Teaching operations with a force sensor for a linear robot arm
Janvier, Loic

Published: 01/01/1986

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. Jan. 2019
TEACHING OPERATIONS WITH A FORCE SENSOR FOR A LINEAR ROBOT ARM

Loïc JANVIER
UNIVERSITE DE TECHNOLOGIE DE COMPIEGNE FRANCE

February - June 1986
WPA rapport nr.0306
Thanks

I am gratefully indebted to my coach, Mr P. Mulders, who welcomed me into his group, for his kindness and his advice.

I want also to thank Henk for having introduced me into my work, who could help me in aloney unsolvable problems and who could give me part of his time.

I will never forget the welcome at the heart of this group, owing to Rogier, Jan, Leon, Mike, Ad and so many others I can not mention here and also this small country, so friendly and nice wherever you go.
## CONTENTS

1. **SUMMARY**  
   1.1 introduction  
   1.2 conlusions  

2. **THE WORK ENVIRONMENT**  
   2.1 the engineering education in the Netherlands  
   2.2 the Eindhoven University of Technologie  
   2.3 the mechanical engineering departement  
   2.4 the FAIR project  
   2.5 my traineeship project  

3. **TEACHING AND CONTROL**  
   3.1 the different teaching modes  
      3.1.1 the manuel method  
      3.1.2 the dummy arm method  
      3.1.3 the telecontrol method  
      3.1.4 teaching via a force sensor  
   3.2 the linear robot arm  
   3.3 control theory  
      3.3.1 the PID controller  
      3.3.2 the different actions  
      3.3.3 association of actions  
   3.4 paths recording  
   3.5 calculation of the controller parameters  
   3.6 identification of our process  
      3.6.1 the controller  
      3.6.2 the robot tied to the servo power supply  
      3.6.3 the robot driven by the 86/05, open loop  
      3.6.4 the robot driven by the 86/05, closed loop  

4. **HARDWARE AND EQUIPMENT**  
   4.1 presentation of the iSBC 86/05  
   4.2 memory board 028A  
   4.3 the development system  
   4.4 the interfaces  
      4.4.1 system overview  
      4.4.2 force sensor / iSBC 86/05  
      4.4.3 iSBC 86/05 / robot power supply  
      4.4.4 the switch debouncer  
      4.4.5 the Hall effect switches  
      4.4.6 calculation time measurement  
   4.5 the iSBX standard  
   4.6 the counter/comparator multimodule board  
      4.6.1 the 24-bit counter  
      4.6.2 the edge to pulse converter  
      4.6.3 the comparator  

---

- 3 -
4.6.4 the decoder
4.6.5 remarks

4.7 initialisation of the hardware
  4.7.1 parallel peripheral interface (PPI)
  4.7.2 programmable interrupt timer (PIT)
  4.7.3 programmable interrupt controller (PIC)
  4.7.4 counter / comparator board

5. THE TEACH COMPUTER PROGRAM
  5.1 description of the public data
  5.2 the MAIN module
  5.3 the ROBOT module
  5.4 the HARDWARE module
  5.5 the CONSOLE module
  5.6 the INTERRUPTS module
  5.7 the INTERRUPTPROCEDURES module
  5.8 compiling and linking TEACH programs
  5.9 different versions of TEACH programs
  5.10 running TEACH programs

6. CONCLUSION

7. APPENDIX
APPENDIX CONTENTS

1 : RESUME EN FRANCAIS

2 : INTERFACES
   interface rack                      3
   front panel                        3
   contents                           3
   ADC & DAC responses                4
   DAC board scheme                   5
   ADC board scheme                   6
   Hall effect switches interface     7
   Push button debouncer              8
   iSBX counter/comparator            9
   iSBX bus pin assignment            10
   edge to pulse converter            11
   up / down decoder                  12
   address decoder                    13
   counter / comparator board overview 14
   counter / comparator board chips implementation 15

3 : EQUIPMENT
   servo power amplifier AXODYN 05 LV 05 16
   servo motor MC 19 P                  17

4 : 86/05 JUMPER CONFIGURATION
   interrupt matrix                    18
   iSBX data mode                      18

5 : SYSTEM FREQUENCY RESPONSE
   controller P, P.I., P.D.            19
   Robot + servo power supply          20
   global system in an open loop       21
   global system in a closed loop      22

6 : SOFTWARE
   interrupt procedure code listing    23

7 : REFERENCES
   former applications at THE          24
   control theory books                24
   teaching with a force sensor        24
   interpolations                      25
1. SUMMARY

Teaching a robot arm with a force/torque sensor has been shown to be a powerful instrument for teaching complicated tasks that requires an on-line programming technique.

This method has been applied to a custom linear robot arm, simple mechanical structure, to test different devices and theories, but also in a goal of getting some experience in that view. This arm is driven by a servo power supply and this global system is controlled by an Intel 86/05 single board computer.

The industrial practice has selected the Proportional - Integrative - derivative controller as being a performant and easy to implement controller. We will try to calculate the transfert function of the robot to know our system and to tune the best the different controllers.

A linear digital incremental transducer is used to measure the robot's current position. We will see the design of an interface counter board under the Intel iS8X standard to use this sensor. At the same time, we will check all the interfaces and equipments used in this application.

At last, the TEACH computer program will be introduced. It has been designed to perform the control of the robot arm in a flexible way and it has been written on an Intel development system.
2. THE WORK ENVIRONMENT

The Technische Hogeschool Eindhoven (T.H.E., Eindhoven University of technology) has just celebrated its sixth quinquennium and the cohabitation between students under the former and the new system of studies comes to an end.

2.1 THE ENGINEERING EDUCATION IN THE NETHERLANDS

Engineering education at University level in the Netherlands is concentrated in separate institutes named 'Technische Hogeschool'; at this type of institute only academic engineering degrees are conferred. There are three institutes of that name:
- TH Delft, founded in 1842,
- TH Eindhoven, founded in 1956
- TH Twente, founded in 1961.

They group about 20,000 students. The first degree which terminates a course of studies is the Master's degree. In engineering subjects, the Master's degree entitles the graduate to use the legally protected abbreviation 'Ir' (Ingineer) before his name. For obtaining the Doctor (Dr) degree no normal course of study is scheduled. The candidate must have an engineer's degree, and is required to carry out original research.

2.2 THE EINDHOVEN UNIVERSITY OF TECHNOLOGY

THE offers nine courses of study in which students can qualify as graduate engineers specialising in the following subjects:
- Technology in its social application (60 students)
- Industrial engineering and management science (1148 students)
- Mathematics (360 students)
- Computer science (438 students)
- Technical physics (529 students)
- Mechanical engineering (872 students)
- Electrical engineering (1083 students)
- Chemical engineering (656 students)
- Architecture, structural engineering and urban planning (716 students)

It thus includes 5862 students with only 7% of girls.

Since the University opened, 8000 students have graduated from it.

THE introduced four years ago the new Dutch system of studies. This one is not very well received by all the students.
The former system of studies gave the students almost complete freedom to study or not to study. They could choose more or less freely the semester in which they wanted to take particular examination. This system allowed the students to take time to manage their own life (jobs with or without any educational interest, etc), to practice sport or also to manage the student association. Each department has its own association and their goal is to help students in obtaining extra facilities such as excursions to national companies, an annual excursion to a foreign country (Japan, Norway, Germany, Belgium, France etc which students have to pay by themselves), holding international student meetings and many others else. The nominal duration of the studies is five years but students stay at the university about seven years in average.

The new system decreases the freedom of the students and shortens the studies duration to four years, with a maximum of six years. It is said to be difficult and to require generally an extra year. According to that, students under this system do not take so much time either for many external things or for extra training periods. Their elders are afraid about how the associative student life will change and also for the University. After spending five months at THE, the students I know are in a very large proportion in the former system.

2.3 THE MECHANICAL ENGINEERING DEPARTMENT

Design and production are the two main groups into which the highly varied tasks of mechanical engineers are divided. The nature of the tasks carried out by the mechanical engineers varies from scientific research and development to industrial organisation. Apart 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 departement in its four divisions:
- fundamentals of mechanical engineering
- product design and development
- design for industrial processing
- production engineering and production automation (WPA).

The mechanical department has about two hundred employees (teachers and technical personnel) and nine hundred students. I have worked for five months among them, in the WPA division.
2.4 THE FAIR PROJECT: Flexible Automation and Industrial Robots

The research project FAIR is financed and directed by the Dutch government. Its aim is to get some experience in flexible automation and industrial robot system.

The mechanical and electrical department of the University are involved in this project as well as several private firms.

The project is divided in five parts:
- the general aspects of automation
- the handling of parts
- kinematic and dynamic of mechanical structures
- the drive systems, the control systems and applications of the systems
- the arc-welding and the sensory system

2.5 MY TRAINEESHIP PROJECT

In view of flexible automation, an one axis linear robot arm has been designed at the University to get some experience in position control, arc-welding, etc, on a simple mechanical structure.

According to this aim, my work was defined as follow: development of a computer program and hardware to drive the robot arm to achieve teaching operations with a force sensor. The user runs the arm in different paths by means of a force sensor while the computer records different coordinates for further replays. This work follows Eric Galet's one, student at the University of Compiegne who spend five months at the THE in 85/86.
3. TEACHING AND CONTROL

3.1 THE DIFFERENT TEACHING MODES

A robot trajectory can be taught in different ways. We are interested here in on-line teaching, way in which the user can visualize and appreciate best what he is teaching.

3.1.1 THE MANUEL CONTROL

The manual control of the arm requires a system which allows motion by direct action on the axis as well as by the actuators. Such a robot is called reversible. It follows that the force required to move the robot must be compatible with the human strength, so it is used with small robots where the required accuracy is small.

3.1.2 THE DUMMY ARM METHOD

The dummy arm method uses a mechanical structure identical to the robot arm but unmotorized and very light. It includes the different position and velocity sensors. As we use a different mechanical arm, the accuracy of the copied motion is not so good. This method has the disadvantage of requiring a second mechanical structure.
3.1.3 THE TELECONTROL METHOD

In the telecontrol method, the user drives the robot structure (like the dummy method) via a teach-pendant or a joystick. The user can observe the effect of his commands. The coordination of several degrees of freedom is impossible using a teach-pendant. The only problem with the joystick is that the human operator is no longer in direct contact with the robot.

3.1.4 TEACHING VIA A FORCE SENSOR

A recently carried out method integrates a 6-axis force/torque sensor in the robot. This method allows a very immediate programming of robots either for paths or at the same time for forces and torques to be exerted onto the robot's environment (HYBRID CONTROL). It avoids all the difficulties inherent with any kind of off-line or CAD programming caused by uncertainties in the geometry of robot and material to be processed. This method is based on a force-torque-sensor ball via which the human hand exerts forces/torques that are converted into motion commands in six degrees of freedom involving the cartesian transformation. The sensor ball may be mounted inertially fixed as a kind of telecommand-knob or may be fixed at the robot's wrist for directly pulling or torquing the robot.

The sensor programming technique may be extended to other kinds of sensors. This concept has turned out to be a powerful instrument for teaching complicated assembly and machining tasks that requires fine sensor-based reactions of a robot.
Sensor ball with DfVLR-force-torque-sensor integrated

Path-only-teach-in with robot mounted sensor ball

Bias torques generated by forces

Path-only-teach-in with fixed sensor ball
Simultaneous path-force-torque-teach-in with fixed sensor ball

fig. 4

Simultaneous path-force-torque-teach-in with robot mounted sensor ball

fig. 5
3.2 THE LINEAR ROBOT ARM

Since a 6-axis robot can be considered as 6 times a 1-axis robot (excluding the mathematical transformations), a linear 1-axis robot arm has been designed at the University to test different pieces of equipment and schemes of control.

The design is commonly used in machine-tools: the DC servo-motor drives the carriage by means of a coupling and a spindle. The behaviour of such a freedrive is simple, but the tendency in machine-tools to higher accuracy and speeds requires sophisticated technology to drive this system. The motor is controlled in a closed-loop system by means of an analog servo power supply and a tachometer. The robot arm is treated as a load disturbance acting on the motor’s shaft.

In addition to the drive system, the linear robot arm consists of:
- a home made force sensor mounted on the extremity of the arm,
- one Hall effect switch at each end of the arm to prevent the software of the extremity,
- an incremental linear transducer to measure the position and the velocity.

Previous realisations with that robot arm have used a home-made computer board based on an 8-bit 8085 Intel microprocessor. After several modifications to adapt that board to a new application, it is not safe enough any more. Also, we wanted to change for a faster computer. It has been exchanged for an Intel 86/05 single board computer (16-bit 8086 microprocessor). This equipment is very commonly used in research and industry and it is important to get used of it.
1: robot arm
2: support
3: spindel
4: DC servomotor
5: force sensor
6: Hall effect switch
7: incremental transducer
3.3 CONTROL THEORY

With flexible automation, electromechanical servomotors and controllers are commonly used. The general goal of a controller is to make the output $y$ follow $x$ as close as possible.

The process control necessities require a computer system to achieve not only Direct Digital Control tasks, but also other functions such as supervisory control, adaptive control optimization, communications with lower and higher levels of control, etc...

3.3.1 THE P.I.D. CONTROLLER

The industrial practice has shown that one of the best methods of control is Proportional - Integral - Derivative control, as it usually meets most of the requirements. An extension to the project is to develop an adaptive control for our system to take into consideration the variations of the system's characteristics but we decided to limit our work to a PID controller.

3.3.2 THE DIFFERENT ACTIONS

It can be shown that the Proportional action increases the damping, therefore the stability.

\[ H_c = K \]
The Integral action reduces the steady-state error but decreases the phase shift of the system and therefore the stability:

\[
H_c = \left( \frac{1}{1 + \frac{1}{T_i.s}} \right)
\]

The Derivative action provides a fast response and increases the stability. Nevertheless, it increases the noise level in the system.

\[
H_c = \left( 1 + T_d.s \right)
\]

### 3.3.3 ASSOCIATIONS OF ACTIONS

Several associations are used:
- P only,
- P.I. for a small steady-state error and if the time response is not an important criterion,
- P.D. for a large frequency band-width (fast system),
- P.I.D. to combine all these characteristics. It may be difficult to tune to the optimal parameter settings.

Working with discrete-time signals, we can express the effect by functions of the form:

- Proportional: \( u(t) = K \ast e(t) \)
- Integral: \( u(t) = \frac{1}{T_i} \int_{t=0}^{nT} e(t) \, dt \)

Reducing to first order and using the rectangular integration, we can write:

\[
u(n^*T) = \frac{1}{T_i} \sum_{k=1}^{n} e(k^*T)
\]

- Derivative: \( u(t) = T_d \ast \left( \frac{de}{dt} \right)_{t=nT} \)

\[
u(n^*T) = \frac{e(n^*T) - e((n-1)^*T)}{T}
\]
A digital P.I.D. controller function can be written as follow:

\[ u(n^*T) = K \times (1 + \sum_{k=1}^{n} \frac{e(n^*T)}{Ti} + Td \times \frac{e(n^*T) - e((n-1)T)}{T}) \]

In a computed control loop, we have the following flowchart:

1. \( e = 0 \)
   \( ex-e = 0 \)

BEGIN LOOP

get \( e \)
\( e = e + e \)

\( \sum_{k=1}^{n} \frac{e}{Ti} + Td \times \frac{e - ex-e}{T} \)

\( u = K \times (1 + \sum_{k=1}^{n} \frac{e}{Ti} + Td \times \frac{e - ex-e}{T}) \)

ex-e = e

END of LOOP

We can try to select the controller actions which fit each part of our work:

- **TEACHING**: the human operator drives the robot in several paths. In that case, there are extra feedback loops via the human hand and eyes. We do not require any integrative action.

- **REACH DESIRED POSITION**: the robot has to reach as fast as possible and without any overshoot the desired position. To have the best global resolution within the limitation of the linear sensor, we need an integrative action. The D action is useful in improving the time response of the system. We therefore choose a PID controller.

- **REPLAY**: the robot's position has to follow the recorded position. As above, we use a PID controller.

### 3.4 PATHS RECORDING

Many ways to record the taught path exist and each one has its own advantages.

To obtain the best accuracy in replay, we can record the robot's position at each sample time. This method requires a large memory but few calculations during teaching and replay. We can have a short sample time and so good system performance.

Since the human arm has a small frequency bandwidth, this way of recording wastes some memory. Also, the position recording samplertime
can be different from the control one. A sampling frequency three times higher than the highest human arm's frequency is an average criterion.

As these two sample times can be very different, we need to calculate an artificial curve joining every points recorded. These calculations require interpolation methods. The knowledge of many points, earlier or later than the current one allows many well known and accurate interpolations methods: polynomial, circular, splines, etc (see references). The only problem is the large computation time.

The output of our digital control is the desired velocity. As we have computed it, we can store it also in memory. We know then a second degree of the curve and the interpolation requires less points to obtain the same accuracy. The time-saving may be important.

The program I have written has a calculation time of about 2.5ms, time which is in the same range as the robot's time-constant. I can not spend too much time in calculation as the program records the robot's position and desired velocity at each control sample time. I replay with the same sample time the desired velocity increased of a function taking into consideration the difference between the recorded position and the current position.
3.5 CALCULATION OF THE CONTROLLER PARAMETERS

Tuning the controller first requires knowledge of the system we are dealing with. A common way is to describe the system with a mathematical model and to use the transfer functions.

Each process can be approximated by a product of several first order systems or, within a lower accuracy, a delay and a first order system:

\[ H = K(1 + T_1 s)(1 + T_2 s)(\ldots \exp(-Tv s) \]

or \[ H = K \frac{\exp(-Tv s)}{(1 + T_1 s)} \]

According to the desired accuracy, many empiric or mathematic methods are available:

- Criterion of Ziegler & Nichols: The system is approximated by a delay and a first order system. Recording the step response suffices to calculate P, PI or PID controller parameters. It is a very easy method to apply and it requires little equipment.

\[ H_p = K_p \frac{e^{-td}}{(1 + T_1 s + 1)} \]

<table>
<thead>
<tr>
<th>regelaar</th>
<th>( K, K_p )</th>
<th>( \tau_1 )</th>
<th>( \tau_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>( \frac{\tau_1}{\tau} )</td>
<td>( \tau )</td>
<td>( \tau )</td>
</tr>
<tr>
<td>PI</td>
<td>0,9 ( \frac{\tau_1}{\tau_1} )</td>
<td>3,3( \tau_1 )</td>
<td>( \tau )</td>
</tr>
<tr>
<td>PID</td>
<td>1,2 ( \frac{\tau_1}{\tau_1} )</td>
<td>2( \tau_1 )</td>
<td>0,5( \tau_1 )</td>
</tr>
</tbody>
</table>

- If we can find the second or third order transfer function, we can apply the following method:

Fig. 12.2 Globale instellingen van regelaars.
Gesteld is \( H = H_p \) voor de volgende situaties:

a - P-regeling met \( K, K_p = \tau_1/\tau_2 \)
b - PD-regeling met \( K, K_p = \tau_1/\tau_2 \) en \( \tau = \tau_2 \)
c - PI-regeling met \( K, K_p = \tau_1/\tau_2 \) en \( \tau = \tau_1 \)
d - PID-regeling met \( K, K_p = \tau_1/\tau_2 \) en \( \tau_1 = \tau_1 \) en \( \tau_0 = \tau_2 \)
3.6 IDENTIFICATION OF OUR PROCESS

I have borrowed a Structural Dynamic Analyser HP 5423 A to measure the frequency response of the robot. This measurement device, in one of many other options, generates a digital white noise (signal which contains a very broad range of frequency) and analyses the system response. It draws then the gain and the phase shift as a function of the frequency.

1 - 2 : controller response
2 - 3 : robot + servo power supply response
1 - 3 : global system

3.6.1 THE CONTROLLER

This device has been used to check the running of the digital controller implemented on the 86/05 single board computer. According to different parameters (sample time, gain, integrative and derivative actions) I could confirm the expectations:
- Sampling introduces a delay ( \( H_c = \exp(-T.s/2) \)). The gain is constant but the phase shift decreases with the frequency. Sampling destabilises the system.
- The D action increases gain and phase shift but is limited by the sampling effect.
- The I action is effective only at very low frequencies because of interference in the sampling.

The PID controller is very difficult to tune: the sample time and the D time constant are very close in value and are also close to the I time constant.

3.6.2 THE ROBOT TIED TO THE SERVO POWER SUPPLY

The resonance at \( f = 100 \) Hz makes the model of the system difficult to calculate. Nevertheless, we can give the robot the following characteristics:
\( \varphi = -90^\circ \) : \( f = 31 \) Hz, \( G = -6 \) dB
We can approximate the system to second order:

\[
H = \frac{1}{(1 + T_0 s)^2} = \frac{1}{1 + 2T_0 z s + (T_0 s)^2}
\]

with \( T_0 = 5 \text{ ms} \) (\( f = 31 \text{ Hz} \)), \( z = 1 \).

The resonance peak cannot be neglected as it is very close to the system's bandwidth. A D action would increase this peak and give an unstable system. The I action can be used only around the desired position to prevent instability of the system.

3.6.3 THE ROBOT DRIVEN BY THE 86/05 COMPUTER, OPEN LOOP

(appendix 21)

The two previous systems are tied in an open loop to measure the effect of sampling on the robot.

At \( \phi = -90^\circ \), we measure \( G = -4 \text{ dB} \) for \( f = 22 \text{ Hz} \).

At \( f = 31 \text{ Hz} \), \( G = -6 \text{ dB} \) but \( \phi = -110^\circ \). Sampling decreases the phase only at the higher frequencies.

3.6.4 THE ROBOT DRIVEN BY THE 86/05 COMPUTER, CLOSED LOOP

(appendix 22)

The phase is the same as in open loop and the curve of the gain is smoother.

Being short of time and to avoid overlap with later projects, I did not optimise the control of the robot. The TEACH program uses only proportional actions and the global performances are not so good.
4. HARDWARE AND EQUIPMENT

4.1 PRESENTATION OF THE iSBC 86/05

The Intel iSBC 86/05 Intel Single Board Computer is a complete 16-bit computer system on a single printed circuit assembly. It's a member of Intel's large family of Single Board Computers and provides an economical self-contained computer based solution for applications in the area of process control.

The central processor is an Intel 8086-2 microprocessor whose clock is selectable to 5 or 8 MHz. This single board computer includes 8K bytes of static RAM, up to 64K bytes of ROM, 3 parallel I/O ports, a serial communication interface, 3 timers, an interrupt controller and two ISBX connectors. The 8087 numeric processor can be plugged on the host board to perform real arithmetic and numerous built-in functions such as log, tangent, etc.
Of its features, we will speak about those which are of special interest to our project.

The on-board 8253-5 Programmable Interval Timer (PIT) provides 3 independent timers which may be configured for a variety of applications such as rate generator, interval timer, real-time interrupts, etc. One of them is used as the baud rate clock for the serial communications.

Serial I/O operation is handled by an 8251 Programmable Communication Interface (PCI) device. The board is configured to the RS 232C standard. It is used for the communications between the board and a console.

The iSBC 86/05 board utilizes an 8255A-5 Programmable Peripheral Interface (PPI) device to control the 3 8-bit parallel I/O ports. All 24 lines may be configured for a variety of purpose applications according to the drivers (output mode) or terminators (input mode) tied to these lines.

2 iSBX connectors are provided on the 86/05 board. The iSBX bus and Multimodule has been specially designed to expand the board’s I/O capabilities as well using Intel Multimodule, as a custom peripheral.

Off-board system access is provided by the Multibus connector (P1) and an auxiliary connector (P2) so as to implement the 86/05 board in a more important system.

The 8086 microprocessor can address up to 1M bytes. As the microprocessor registers contains only 16 bits, the 8086 uses two registers to address in memory. The segment register is computed with the offset register to obtain a 20-bit address.

In a common use, the segment register has a fixed value and only the offset may change to address an other piece of data.
The I/O space is separate from the memory space but has the same structure. The I/O space can accommodate up to 64K 8-bit ports or up to 32K 16-bit ports. All ports are considered to be in one same segment. The first 256 ports are directly addressable (with only one instruction) by some I/O instructions and are the most used and the only ones used in our application.

The iSBC 86/05 board provides 9 vectored interrupt levels. The highest level is the Non Maskable Interrupt (NMI) line which is directly tied to the 8086 CPU. At the moment, it is used for signaling power failure and user's interrupt request.

The i8259A Programmable Interrupt Controller (PIC) provides control and vectoring for the next 8 interrupt levels. A selection of four priority processing modes is available.

The PIC accepts interrupt requests from all on-board I/O resources and from the Multibus system bus at an amount of 24 source lines, active ones selected by jumper.

The PIC then resolves requests according to the select mode and the mask register and, if appropriate, issues an interrupt to the CPU.

In Multibus system, the slave PICs may be interfaced to a master one which so can handle 64 interrupt levels.
4.2 MEMORY BOARD 028A

In order to record a path, the iSBC 86/05 requires a RAM expansion. We have implemented a RAM Memory Board through direct Multibus interface.

The 028A memory board contains 128K bytes of read/write memory. The only work necessary to set this board was the configuration of several jumpers to select the starting address.

The iSBC 86/05 includes 8K bytes of RAM whose addresses are from 00000 H to 01FFF H. So as to have a continuous memory, the memory board's starting address has to be 02000 H. This requires that only pins E86 and E79 be connected. These addresses have to be in page 0 (first Mega byte page), so it is necessary to remove all jumpers in pins E139 to E143.

In our application, we do not use the parity flag register which may indicate a data transmission error.

As it is skipped, the use of that memory is transparent to the user and the iSBC 86/05 contains a RAM memory going from 00000 H to 21FFF H.

Address Selection Jumper Configuration
4.3 THE DEVELOPMENT SYSTEM

The Intellec serie III micro-computer development system is a useful tool for designing microcomputer software for the iAPX86,88 processor. We can write programs, debug programs, link them, locate them and run them on single board computers or on the system itself. We can connect an emulator for running our programs in the hardware environment. The emulation is the controlled execution of the prototype software in an artificial environment.

This system offers the possibility to write source programs in high level languages: Pascal, Fortran, PLM and Assembly. These languages are special Intel versions (Pascal86, fortran86, PLM86, Assembly86) to allow specific functions such as Input/Output, interrupts management, etc. The user can divide his program in several parts or 'modules' and in different languages and link them all together with other library files. The program is thus modular and several writers can work on it separately. These different modules are linked together by an 'interface specifications' section. This part declares public all constants, types, variables, procedures and functions which can be called by other modules. In a complete program, the interface specifications part can be common to all the modules.
The compiler converts the user’s instructions into object code modules. Compiler controls allow you to specify options such as including an other module (the common interface specifications, by example) or enabling the assembly code listing. These controls can be included in the source file with the prefix $ or during the compiler invocation. The SMALL control directs the compiler to perform a certain memory addressing technique that help reduce the amount of code produced and so minimize the necessary memory space. LARGE can be used for larger programs.

LINK86 combines 8086 object modules and resolves references between independently translated modules. All the object files involved have to be compiled under the same memory addressing mode control (SMALL or LARGE). LINK86 has also some controls but none is necessary in this application.

LOC86 changes a relocatable 8086 object module into an absolute object module which contains the absolute address of each public part of the program.

OH86 converts an absolute object file to 8086 hexadecimal file loadable by iSBC computers.

Process Relocation

TO RELOCATE AN ENTIRE PROCESS MOVE THE CODE, STACK AND DATA, AND UPDATE THE SEGMENT REGISTER CONTENTS TO POINT TO THE NEW AREAS.
4.4.2 INTERFACE FORCE SENSOR / iSBC 86/05

The force applied to the strain gage is measured and the signal is amplified by a Mesverstarker KW5/35-5 in a range of +/- 4 V DC. That analog bipolar voltage is then driven to the analog-to-digital board which accepts a bipolar +/- 10 V analog input and performs an 8 bit conversion in 25 micro seconds.

THE AD 570 ANALOG-TO-DIGITAL CONVERTER:

As the Blank & Convert/ is driven low, the outputs will remain in high impedance state and the conversion will start. Upon completion of the conversion, the Data-Ready/ will go low and the data will appear at the outputs. Pulling the B & C/ input high blanks the output and readies the device for the next conversion. The AD 570 accepts analog inputs of 0 to +10 V or +/- 5 V bipolar.

THE ADC BOARD: ( appendix 6 )

It has been designed to be an element of the interface rack which provides the power supply. The board includes a signal amplifier so as the input has a high impedance in a range of +/- 10 V. Since the force measurement system can drive a 40 ohm input, the AD 570 low input impedance pin can be used in our application to provide a +/- 5 V input range. The Blank & Convert/ signal is inverted using a NAND gate and the outer Convert signal is high level active (B/ & C).

Communications between the PPI and the ADC board require some terminators to be plugged into the 86/05 board for the ports B and C high (data ready signal, both ports are in input mode) and a driver chip for the port C low (B/ & C, output mode).
4.4.3 INTERFACE iSBC 86/05 / ROBOT POWER SUPPLY (appendix 5)

The robot servo power supply uses a +/- 10 V bipolar analog input as desired velocity signal. This signal is performed by a Digital-to-Analog board which contains a DAC 08 converter.

The DAC 08 performs a very high speed conversion (85 nanoseconds) and no control line is necessary. Since this low cost DAC is very sensitive to the temperature, the user can modify the DAC output offset by means of a precision potentiometer placed on the rack front panel.

The PPI port A, in output mode and connected to the DAC, requires some drivers. These are the default ones, the 86/05 is designed with a 8287 8-bit driver/terminator chip.
4.4.4 THE SWITCH DEBOUNCER

A push button is used to drive the program: emergency stop, start/stop recording, replay, etc. It is connected to the iSBC interrupt matrix via a switch debouncer and the parallel I/O connector.

---

4.4.5 THE HALL EFFECT SWITCHES

One Hall effect switch is placed on each end of the robot arm to prevent the arm extremity by means of interrupts. Two Schmidt triggers read these switches to give the signal a squared shape.
4.4.6 CALCULATION TIME MEASUREMENT

Port C line 1 is tied to the interface rack front panel (BNC connector). This line is set during the calculations to measure how short the sample time can be.

REMARK: These interfaces come from previous applications and have been checked for my work.
4.5 THE iSBX STANDARD

Intel has recently introduced a new line of board products and a new bus which are designed to extend the functional capabilities of single board computers at a much lower cost than previously with the MULTIBUS.

iSBX Multimodule boards are supported by the iSBX bus which allows them to be added directly on the on-board bus. The iSBX bus fits to low cost and compact custom designs.

The iSBC 86/05 board includes two iSBX connectors. iSBX communications are seen as I/O operations. Each connector has 16 8-bit ports or 8 16-bit ports. A jumper selects the 8 or 16 bit mode for each iSBX connector. The host board provides a mechanical support and electrical connections.

THE iSBX BUS

The iSBX bus is divided in six groups of lines, according to the different functions:

- Address and chip select lines: a 86/05 on-board I/O address decoder provides two chip select lines per iSBX connector and the address lines A1, A2, A3 from the address bus.

- Data lines: The 16-bit data bus of the host board is buffered to interface the iSBX board. However the higher and lower parts of the bus are inverted compared with the host board data bus:

```
<table>
<thead>
<tr>
<th>iSBX data bus</th>
<th>host board data bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>data 15</td>
<td>data 7</td>
</tr>
<tr>
<td>data 8</td>
<td>data 0</td>
</tr>
<tr>
<td>data 7</td>
<td>data 15</td>
</tr>
<tr>
<td>data 0</td>
<td>data 8</td>
</tr>
</tbody>
</table>
```

- Interrupt lines: Two high active interrupt lines are tied to the interrupt matrix and are jumper enabled or disabled.

- Option lines: two lines are tied to on-host-board pins and are for the user's use (extra interrupt lines, etc).

- Control lines: this group includes I/O read, I/O write, reset, clock, wait and Multimodule present. I/O R & W can be used in conjunction with the chip select and address lines to select a Multimodule function.
Power lines: +/−12 V and +5 V are provided by the host board.
4.6 THE COUNTER / COMPARATOR MULTIMODULE BOARD

We desire to interface the linear incremental sensor with a multimodule board to have a compact system. According to a previous design, the Multimodule board has to include:
- an edge to pulse converter, to provide the clock to the counter.
- a 24 bit up/down counter.
- a 24 bit comparator which provides interrupts according to the two words (robot's position and a compared position).
- a decoder to select the desired Multimodule function.
- a connector to the linear sensor including power supply lines.

This Multimodule has also to be as standard as possible in order to be used in further applications.

4.6.1 THE 24-BIT COUNTER (appendix 13)

The linear robot arm has a usable length of about 70 cm which corresponds to 70,000 edges from the linear sensor. We thus need at least 17 bits to code the robot's position. Three 8-bit up/down counters (74 AS 867) n-bit cascading are used to do this. These counters have 2-state outputs and need to be interfaced to the data bus by some 3-state latches (74 AS 573). The counter has 2 function lines to provide 4 modes: count up, count down, load start value and reset. The clock and the two select lines are besides tied to three pins to be able to use that Multimodule counter board without the edge-to-pulse converter.
4.6.2 THE EDGE TO PULSE CONVERTER

The linear sensor provides 2 incremental signals, 90° enphased, with a space-period of 40 microns and a mid-position signal. Each one has its own inverse also provided. These signals have to drive a converter to provide a clock (pulse width of about 1 micro-second), an up/down counting mode line and a zero position reset line.

4.6.3 THE COMPARATOR

Once loaded to the desired value, the comparator has to provide an interrupt to inform the program of the robot position. One of the two compared bytes has to be latched from the data bus, the other needs a permanent on-line input to the counter. Also, this type of comparator must be n-bit cascading. Three 74 AS 885 have been chosen.

4.6.4 THE DECODER

A 74 AS 154 4-to-16-line decoders/demultiplexers decodes the 3 address lines, the chip select line and the I/O read & write lines to select the desired function of that Multimodule board:

<table>
<thead>
<tr>
<th>address</th>
<th>function</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>read position bit 0 - 15</td>
<td>I</td>
</tr>
<tr>
<td>A3</td>
<td>read position bit 16 - 23</td>
<td>I</td>
</tr>
<tr>
<td>A5</td>
<td>comparator loads desired position bit 16 - 23</td>
<td>O</td>
</tr>
<tr>
<td>A7</td>
<td>comparator load desired position bit 0 - 15</td>
<td>O</td>
</tr>
<tr>
<td>A9</td>
<td>load position bit 0 - 23 into latches</td>
<td>I or O</td>
</tr>
<tr>
<td>AB</td>
<td>reset counter</td>
<td>I or O</td>
</tr>
</tbody>
</table>
4.6.5 REMARKS

- During the Multimodule board design, I did not know the upper and lower data lines are inverted between the iSBX bus and the host board bus: we therefore need to convert all 16 or 24-bit data before writing or after reading operations.
- Unfortunately, the bytes of the comparator are inverted compared to the counter and latch bytes:
  
  \[
  \begin{align*}
  \text{bit 0} & \longleftrightarrow \text{bit 7} \\
  \vdots & \vdots \\
  \text{bit 7} & \longleftrightarrow \text{bit 0}
  \end{align*}
  \]

The desired position has to be converted before to be written into the comparator.
- The first design includes CMOS technology for the edge-to-pulse converter. This hardware has a propagation delay of about 65 nanoseconds, which means it is more than 10 times slower than the AS one used for the counter, and the delay introduced by each gate generates some noise in the counter function lines. I had to change for the LS technology which gives good results for this application.
- For the same kind of application, the 74 LS 2000 groups the most part of the board: the edge to pulse converter and a 16-bit up/down counter externally resettable whose outputs are latched. Nevertheless, comparison with the desired position is impossible.
4.7 INITIALIZATION OF THE HARDWARE

4.7.1 PARALLEL PERIPHERAL INTERFACE

The PPI has a total of 24 I/O lines grouped into 3 ports A, B and C which addresses are C8, CA and CC. In order to use any of the parallel port lines, the PPI must be initialized to the desired mode. Only one control word is necessary for this operation.

For our application, we need:
- port A : output (to the DAC and then to the robot power supply)
- port B : input (from the ADC and the force transducer)
- port C high (line 4-7) : input (end of force conversion)
- port C low (line 0-3) : output (Blank/ & Convert to the ADC)

On that way, the control bits are as follow:

<table>
<thead>
<tr>
<th>bit</th>
<th>level</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>active mode</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>mode 0 : latched output (port A)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>mode 0 : &quot;&quot;</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>port A : output</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>port C high : input</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>mode 0 : input (port B)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>port B : input</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>port C low : output</td>
</tr>
</tbody>
</table>

Initialization of the PPI: write the control word (8A H) to the control word address (CE H).

Upon PPI initialization, we give the robot a null velocity: write 0 to port A (C8 H) and ready the ADC to convert activating the ADC Blank/: write 0 to port C (CC H).

4.7.2 PROGRAMMABLE INTERRUPT TIMER

We use the timer 0 for sampling: it has to be set in the rate generator, mode 2.
PIT mode control word format:

<table>
<thead>
<tr>
<th>bit</th>
<th>level</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>select counter 0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>read least significant byte first,</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>then lost significant byte</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>} mode 2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>}</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>binary counter (16 bits)</td>
</tr>
</tbody>
</table>

Initialisation: write 34H to PIT control word address (D6H).
To load the counter 0, we have first to write the least significant byte, then the most significant one.

4.7.3 PROGRAMMABLE INTERRUPT CONTROLLER

The PIC functions as an overall manager in an interrupt-driven environment. It may work in a variety of configurations and its initialization requires 3 or 4 Initialization Command Words (ICW).

ICW1 indicates if that PIC takes place in a multi-PIC configuration or not and defines the level or edge triggered input for the 8 interrupts.

<table>
<thead>
<tr>
<th>bit</th>
<th>level</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>edge triggered input</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>single PIC</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

ICW1 = 13H and has C0H for address

ICW2 gives the five most significant bits of the vector byte. As the 86/05 PIC handles interrupts 32 to 39 (see the INTERRUPTS module) and each interrupt procedure address is written on four bytes, the first address is \(4 \times 32 = 128 = 80H\). As we write only the five most significant bits, ICW2 = 20H. Its address is C2H.
ICW3 is used only for a slave PIC and is omitted here.

ICW4 sets:

<table>
<thead>
<tr>
<th>bit</th>
<th>level</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>not fully nested mode</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>}</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>} buffered mode - master</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>auto end of interrupts</td>
</tr>
</tbody>
</table>

ICW4 = 0F H and its address is C2 H.

These 3 bytes have to be written in this order.

Each interrupt can be separately masked via the mask register. During the system initialization, all the interrupts are masked. They will be set according to the desired use during the execution of the program.

4.7.4 COUNTER / COMPARATOR BOARD

The counter / comparator board requests to be initialized after each loading of the program. This requires to move the robot over the sensor zero position. The counter function mode (count up or down, reset) is performed from the linear sensor signals.
5. THE TEACH COMPUTER PROGRAM

A large part of my traineeship concerns the writing of a computer program to drive the robot arm in teaching and replay operations.

So as to be as modular and as easy to read and improve as possible, the program is divided into several parts, or modules, and is written in Pascal. Each module includes different procedures or functions.

MAIN initializes the system, asks for the user's choice and calls one of its three main procedures.

ROBOT includes functions and procedures which deal with the robot plus two controller functions.

HARDWARE groups the initialization procedures and drives the dialogue with the interfaces (force, position).

CONSOLE improves the data communications between the 86/05 board and the user's console. CONSOLE could be converted in a library file but it's use it would not change.

INTERRUPTS attributes an interrupt procedure to each interrupt vector handled by the PIC (hardware interrupts) and resets the interrupt flags.

INTERRUPTPROCEDURES groups the assembly written interrupt procedures.

As the sample time of the control has to be short, I had to take care of the calculation time. I use only integer variables computed on 16 or 32 bits:

\[
\begin{align*}
0 & \leq \text{WORD} \leq +65535 \\
-32768 & \leq \text{INTEGER} \leq +32767 \\
-2^{31} & \leq \text{LONGINT} \leq 2^{31} > 2\text{Mega}
\end{align*}
\]

Calculations are slower with long integers so I use them as little as possible: for position (17 bits) or calculations which use the position.

The interrupt flags are Boolean in Pascal and have to be declared Byte in Assembly to be compatible.
5.1 DESCRIPTION OF THE PUBLIC DATA

CONSTANTS:

- STARTSTOPSTRING is a message to be printed on the console many times in different modules. It has been declared public for an easier use.
- ROBOT is the address of the PPI, port A, connected via the D.A.C. to the servo-power-supply.
- PICMASK is the address of the PIC mask register. It is used many times to enable or disable the desired interrupts.

TYPES:

- STRING30 is a 30 characters string type used for messages written from the 86/05 to the console.
- STRING5 is a 5 characters string type and has the same use as above.

VARIABLES:

- LSWITCH, RSWITCH, STARTSTOP and TIMEO (Boolean) are the different interrupt flags.
- VELOCITY (integer) is the robot's velocity.
- KV, TDTSV, KX, TSTIX, TDTSX, KP, TSTIP and TDTSP (word) are the controllers parameters calculated in the MAIN module and used in the ROBOT module.
- POSITION, WPOSITION, EXPOSITION (long integer) are the robot present position, wanted position and former position.
- SAMPLETIME (word) is the sample time of the control expressed in tenths of milli-seconds.
5.2 THE MAIN MODULE

The bootstrap initializes the system hardware and sets the interrupt procedures. Then, it asks the user's choice.

The TEACHIN procedure consists of reading the force exerted on the force sensor and converting it to the desired robot velocity within a control rule. It is divided in two parts:
- displacement to the start position,
- displacement plus recording of the path.

The user may choose the control parameters and drive the program with the push button. For example, it can be used to stop the displacement of the robot (emergency stop). The Hall effect switches also perform a stop.

POSITION requires at least 17 bits so it is computed as a long integer (32 bits). VELOCITY is in a range of -128 to +127 and needs only 8 bits. COORDINATE groups them in a long integer to save the memory space.

This version of the program records position and velocity each sample time (about 3 milli seconds minimum) in the coordinate vector at an amount of 13,000 points.

Mathematically, we have:
\[
\text{desired velocity} = f(\text{force} - \text{velocity}) \\
\text{coordinate}(i) = \text{position} \times 256 + \text{velocity}
\]

The REPLAY procedure reads in memory the recorded position and velocity and computes the desired robot velocity taking into consideration the current robot position. Indeed, if the load of the robot changes between the teaching and the replay operations, the inertia of the system would be different and replaying the velocity only may perform a different trajectory than the taught one.

Mathematically:
\[
\text{desired velocity} = \text{coordinate}(i) \mod 256 \\
\text{desired position} = \text{coordinate}(i) \div 256
\]

and in output:
\[
\text{velocity order} = \text{des. velocity} + f(\text{des. position} - \text{position})
\]

TESTPOSITION procedure has been written to check the position counter and computation but is now used to move the robot from the current position to a desired position. It asks the user the controller parameters and the desired position (-270 < position < +340 mm). It calls then the REACH procedure (module ROBOT).
READPARAMETERS( TT, PP, IT, DD : CHAR ) reads the control parameters if their corresponding procedure-parameter are set. It is called by the three previous procedures.
TEACHIN FLOWCHART

1. READPARAMETERS
   (Ts, K, Td)
   Set parameters
   Start order
   y
   Start sample time
   Enable STARTSTOPINT,
   TIMEOINT, LSWITCHINT, RSWITCHINT

2. TIME = 0
   y
   Read FORCE
   POSITION
   VELOCITY
   WVELOCITY = PV(VELOCITY)
   i > 1
   n
   y
   Save COORDINATE (i)
   i = i + 1
   i = 1
   and
   STARTSTOP
   y
   Save start position
   STARTSTOP and i > 1
   or i > 10000
   or LSWITCH or RSWITCH
   y
   IMAX = i
   STOP ROBOT

END
REPLAY FLOWCHART

**REPLAY**

**READPARAMETERS** for REACH procedure and for the position error controller

- set parameters
- enable STARTSTOPINT

**start order ?**

- \( n \)
  - REACH start position
  - start sampletime
  - enable STARTSTOPINT, TIMEOINT, LSWITCHINT, RSWITCHINT

**TIME 0 ?**

- \( n \)
  - read POSITION
    - COORDINATE( i ) \( \rightarrow \) WPOSITION
    - \( \rightarrow \) WVELOCITY
  - VELOCITYORDER = WVELOCITY + PIDX( WPOSITION - POSITION )
  - \( i = i + 1 \)
  - STARTSTOP or \( i = imax \) or LSWITCH or RSWITCH ?

- \( n \)
  - STOP ROBOT

**END**
5.3 THE ROBOT MODULE

This module contains two controller functions and procedures which affect directly the robot.

INITPOSITIONCOUNTER is executed once at each running to reset the position counter. The robot goes first to the right Hall effect switch then to the left one and calls STOPROBOT. As the counter board includes a hardware counter reset based on the zero position signal, the counter is initialized.

STOPROBOT is used in case of Hall effect switch or startstop interrupt to stop the robot and to stay in the inner side of the switches. The program generally returns to the menu upon complete execution.

REACH( X : LONGINT, VMAX : INTEGER ) asks the robot to go to the desired position X with a maximum velocity of VMAX. The procedure includes a P.I.D. position controller. It is called by REPLAY to go to the first position of the trajectory and by TESTPOSITION.

PDV( V, EXV : INTEGER ) : INTEGER is a velocity P.D. controller function where V is the error input and EXV is the former error input. It is called by TEACHIN and returns an integer in the range of -128 / +127.

PIDX( X, EXX, SUMX : LONGINT ) : INTEGER is a position P.I.D. controller function for REPLAY where X is the position error, EXX the former position error and SUMX the summation of the position error. It returns an integer in the range of -128 / +127.
5.4 THE HARDWARE MODULE

It groups all functions and procedures which have to do with the hardware.

INITHARD calls three subprocedures to initialize the PPI, the PIC and the PIT (INITPPI, INITPIC, INITPIT). This procedure is executed during the system initialization stage once at each running.

STARTSAMPLETIME (TS : WORD) loads the timer 0 with the sample time value (TS) expressed in tenths of milliseconds (maximum 53.0 ms with the 1.23 MHz clock).

READFORCE reads the force carried on the force sensor and returns a value in a range of -128 / +127 (one byte magnitude).

READVELOCITY calculates the robot's velocity with the actual position, the former position (EXPOSITION) and the sample time. It returns a value in a range of -128 / +127. The actual position then becomes the old one. READPOSITION has to be executed before READVELOCITY.

READPOSITION reads the counters via the latches and converts these data into the true robot's position expressed in tens of micrometres.

- OUTWRD(OA9H, 0) : latches 1, 2 & 3 load the counters 1, 2 & 3 values
- INWRD(OA1H, PL) : As PL is defined as a word, PL is a positive 16-bit value.

\[
\begin{align*}
\text{PL} & : \begin{bmatrix}
\text{bit 15} & | & 8 & 7 & 0 \\
/ & \backslash \\
\text{counter 1} & & \text{counter 2} \\
( \text{bit 0-7} ) & & ( \text{bit 8-15} )
\end{bmatrix} \\
\Rightarrow \text{byte 1} & = \text{PL DIV 256} \\
\text{byte 2} & = \text{PL MOD 256}
\end{align*}
\]

- INWRD(OA3H, PH) : PH is defined as a word.

\[
\begin{align*}
\text{PH} & : \begin{bmatrix}
\text{bit 15} & | & 8 & 7 & 0 \\
/ & \backslash \\
\text{counter 3} & & \text{unsignificant} \\
( \text{bit 16-23} )
\end{bmatrix}
\end{align*}
\]
byte 3 = PH DIV 256

Each byte is in a range of 0 / 255 and they are computed together to return the true position (long integer).
5.5 THE CONSOLE MODULE

Since the iSBC 86/05 monitor includes the minimum functions, the exchanges between that board and the console are only using ASCII characters. I had to write or copy from former applications several functions and procedures to ease these communications. A library file disposes of two integrated function/procedure:

Consol Input reads an ASCII character from the keyboard.

ex: dataascii := CI.

Consol Output writes an ASCII character on the console.

ex: CO( dataascii )
    CO( 'N' ).

PRINT30 ( WORD30 : STRING30 ) writes a 30-character string on the console.

PRINT5 ( WORD5 : STRING5 ) writes a 5-character string on the console.

LECTURE5 returns a positive integer from the keyboard. This function calls a subprocedure ASCII5TOINTEGER5 to convert an ASCII string to an integer.

WRITE5 ( NUMBER : INTEGER ) writes a signed integer on the console.

LINE starts on a new line on the console.

TAB ( LENGTH : INTEGER ) is the horizontal tabulation with a variable magnitude.
5.6 THE INTERRUPTS MODULE

This short module contains only one procedure

STARTINTERRUPTS attributes an interrupt procedure to each interrupt vector handled by the PIC.

I could have used several interrupt procedures per source and set one of them according to the stage of the program for the desired use. As the interrupt procedures are written in Assembly, I decided to write universal procedures, set a flag, and to treat this information in the Pascal program.

I have to set the interrupt procedures only once, during the system initialization. Then, I use the interrupt mask to enable them or not.

STARTINTERRUPTS resets the flags, enables the interrupts and masks all the interrupts handled by the PIC.

Only four interrupt procedures are used:

<table>
<thead>
<tr>
<th>int. vector</th>
<th>int. name</th>
<th>flag</th>
<th>cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>LSWITCHINT</td>
<td>lswitch</td>
<td>left hall effect switch reached</td>
</tr>
<tr>
<td>34</td>
<td>RSWITCHINT</td>
<td>rswitch</td>
<td>right &quot; &quot;</td>
</tr>
<tr>
<td>35</td>
<td>STARTSTOPINT</td>
<td>startstop</td>
<td>press button</td>
</tr>
<tr>
<td>36</td>
<td>TIMEOINT</td>
<td>timeO</td>
<td>timer 0 : sampletime</td>
</tr>
</tbody>
</table>
5.7 THE INTERRUPTPROCEDURES MODULE

To make it easy to read, we decided to write the program in Pascal 86 language. Nevertheless, the interrupt procedures written in Pascal 86 waste too much time in execution: first, all the 8086 registers are pushed in stack and pulled back once the procedure is executed (appendix 23). Some interrupt procedures do not need to save so many registers. These are written in ASM 86 assembly language to execute the only necessary instructions. The scheme is unique, the flag according to the interrupt cause is set. The Pascal program then treats the information and resets the flag for the next interrupt request.

Remarks: The design of the counter/comparator board allows 3 more interrupt sources: position = desired position, <, and >. These three are not used in my program.
5.8 COMPILING AND LINKING TEACH PROGRAMS

Each module includes a common interface specification (HEAD2.PAS) and all of them have to be compiled within the same memory addressing technique. Using the SMALL control limits the addressing area to 64K bytes and so the number of points we can record is about 14,000. The LARGE addressing technique allows larger records, requires a larger memory but disables the interrupts in our application (?).

SBCTO_.LIB contains the two library instruction CO (Consol Output) and CI (Consol Input). Two files are available according to the compiler control Small or Large (_).

- Linking our modules requires other library files:
  - P86RNO.LIB & P86RN1.LIB are both always required,
  - RTNULL.LIB resolves external references,
  - 87NULL.LIB is necessary as we do not use real arithmetic with the 8087 Numeric Data Processor.

Remark : Linking each interrupt procedure written in Assembly generates a warning 65 message ('incorrect declaration of external symbol' ) but does not affect the execution of the program.

During a CPU reset, data is saved in RAM. That area has to be free, or the resident program will be erased. This is done using the RESERVE(01198H to 011B0H) locator control (location given for TEACH2). Locating gives a listing file (.MP2) which can give us informations about the memory space remaining usable.

Conversion into hexadecimal requires no control.
5.9 DIFFERENT VERSIONS OF TEACH PROGRAMS

TEACH2 is the development version. As the addressing mode uses one offset register to address at most 64K bytes, this program can not record more than 14,000 points.

TEACH is the user's version: it looks like TEACH2 but the control parameters are included in the file ( Ts = 3 ms, 10,000 points ).

TEACHF has been written to measure how short the sampletime could be without using any procedures. The fast part concerns only the first stage of TEACHIN, when the robot is driven to the start position. The sampletime can be about 1.7 ms short. This version is impossible to read and to check and as the results are not as good as expected, this 'turbo' version has been forsaken.

TEACHL and all modules whose last letter is 'l', except CONSOL, have been compiled using the LARGE addressing technique. The goal was to record more points but the interrupts do not affect the progress of the program any more.
5.10 RUNNING TEACH PROGRAMS

HARDWARE CONFIGURATION

Connect
- force sensor to the measurement device (DMS induktiv),
- the measurement device (+/- 4 V) to the ADC (ADC),
- DAC (DAC1 STURING) to the servo power supply (SET POINT),
- tachometer to the servo power supply (TACHO IN),
- the servo power supply (SIGNAAL UIT) to the DC servomotor,
- the linear sensor to the counter board with 'LINEAR SENSOR' above,
- the interface rack flat cable to the 86/05 J1 connector with 'J1' above,
- the interface rack Hall effect switch twin black cable to the 86/05 interrupt matrix,
- the Serie III development system to the 86/05 (RS 232 C standard),

check the jumper configuration (appendix 18).

The Intel development system can be used as a console for the single board computer. Calling SBC861 switches control to the 86/05. The communication rate is selected typing 'UU'. The iSBC's monitor answers returning '.' and the program can be loaded by the instruction 'LS,:Fx:TEACHw.HEX' where $k$ is the number of the floppy driver and $w$ a character according to the desired version of TEACH program. After a reset, the user can restart running the program: 'Go 20:6'. The starting address can be fixed at an other location by a locater control.
6. CONCLUSION

Teaching with a force sensor shows here many interesting characteristics. This method does not require any experience from the operator and he is in perfect contact with the robot and the environment.

Five months spend at THE are not too much to work on such a project. Design and realization of hardware may cost you a lot of time, even more if you have to order some components. The development of the program required first to finish all the interfaces. Also, it included many problems inherent with the specificity of the tool used to write the programs.

The following step is to implement a performant adaptative controller. After that, a really flexible and performant tool will be available. Later research are considered on a 6-axis robot to give this work its full magnitude.
Mon projet de fin d'études s'est déroulé à l'Université de Technologie de Eindhoven au sein du groupe Production & Automatisme rattaché au département du Génie Mécanique.

Le contrat gouvernemental FAIR (Flexible Automation and Industrial Robots ) lie l'Université à plusieurs entreprises privées dans le but d'acquérir de l'expérience dans le domaine de l'atelier flexible et de la robotique.

Mon travail a consisté à développer hardware et software pour une récente méthode d'apprentissage : le pilotage d'un robot via un capteur de force. Un capteur de force six axes (force / couple) est monté au niveau du poignet du robot. L'opérateur peut manœuvrer ce capteur et le robot est asservi à ces signaux. Cette méthode permet à l'opérateur de rester en parfait contact avec le robot et la précision de la réexécution peut être excellente.

Un robot linéaire à un seul axe a été développé à l'Université afin d'expérimenter différents équipements et principes de contrôle. Ce robot est équipé d'un capteur de force à un seul axe. Le contrôle de ce robot se fait grâce à un ordinateur complet Intel 86/06 implémenté sur un seul circuit imprimé : il comprend un microprocesseur 16 bits Intel 8086, de la mémoire RAM et ROM, des interfaces de communications parallèles et séries, un timer, un contrôleur d'interruptions... On dispose d'un système de développement Intel afin d'augmenter les facilités d'utilisation : écriture des programmes en langage évolué, lecteurs de disquettes, imprimante ...

Mon travail peut se diviser en trois parties :
- étude de la dynamique du système: recherche d'un modèle mathématique du robot (fonction de transfert) à l'aide d'un analyseur Hewlett Packard, détermination des différents régulateurs employés (Proportionnel - Intégral - Dérivé, un prochain projet étant de mettre au point un contrôleur adaptatif), calcul des paramètres, 
- écriture d'un programme en Pascal d'apprentissage et de réexécution de la trajectoire enregistrée,
- étude et réalisation d'une interface compteur - comparateur 24 bits sous le standard Intel iSBX pour interfacer un codeur incrémental de position.

La première partie citée a posé quelques problèmes en raison d'une résonance du système probablement due à la partie mécanique du bras. De ce fait et par faute de temps, nous n'avons pas pu déterminer les paramètres optimaux et les performances de ce robot piloté par mon programme ne sont pas excellentes.

Cinq mois passés à Eindhoven resterons graves dans ma mémoire autant pour l'intérêt du travail que pour l'environnement humain:
hébergement avec plusieurs étudiants de différents domaines, travail avec des chercheurs-professeurs et des étudiants, le pays...
INTERFACE RACK

FRONT PANEL CONNECTIONS:

<table>
<thead>
<tr>
<th>A.D.C. INPUTS</th>
<th>D.A.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>low impedance, +/- 5 V</td>
<td>* * * output, +/- 10 V</td>
</tr>
<tr>
<td>high impedance, +/- 10 V</td>
<td>* * * offset potentiometer</td>
</tr>
<tr>
<td></td>
<td>(do not touch too much)</td>
</tr>
<tr>
<td></td>
<td>port C, line 1</td>
</tr>
</tbody>
</table>

Also on the front panel, the push button called 'interrupt'.

CONTENT:

- Digital to Analog Converter
- Analog to Digital Converter
- Hall effect switches debouncer
- Push button debouncer
- Power supply +/-15 V, +5 V
ADC & DAC RESPONSE

DAC BOARD: go to the left switch: \[ 0 < v < 127 \]
right \[ 255 > v > 128 \]
\[ \text{slow} \quad \text{fast} \]

\[
\begin{align*}
\text{Digital} & \quad \text{Analog} \\
\downarrow & \quad \downarrow \\
255 & \quad +10V \\
127 & \quad 0 \\
128 & \quad 10V \\
\end{align*}
\]

A.D. Board

ADC BOARD:
\[ 127 > \text{push} > 0 \]
\[ 128 < \text{pull} < 255 \]
\[ \text{low} \quad \text{strong} \]

\[
\begin{align*}
\text{Digital} & \quad \text{Analog} \\
\downarrow & \quad \downarrow \\
255 & \quad +5V \\
127 & \quad 0 \\
128 & \quad +5V \\
\end{align*}
\]

- App 4 -
HALL EFFECT SWITCHES INTERFACE

Switch S1

UGN 3013T
1 2 3

Switch S2

UGN 3013T
1 2 3

5V 0V

2.2 kΩ

4033

- App 7 -
PUSH BUTTON DEBOUNCER

PRESS BUTTON

- 100k
- 10k

+5V

J1 - 50
Ext INTP/
EDGE TO PULSE CONVERTER SCHEME

8 : 4030 quadruple 2-input EXOR gates
9 : 4528 dual MONOSTABLE multivibrator
UP / DOWN MODE DECODER

1', 3' : 74 LS 00 quadruple 2-input positive NAND gates
2', 4' : 74 LS 10 triple 3-input positive NAND gates
5' : 74 LS 20 dual 4-input positive NAND gates
6' : 74 LS 04 hex inverter
COUNTER FUNCTION MODE DECODER

7: 74 LS 08 quadruple 2-input positive NAND gates
8: 4030 quadruple 2-input EXOR gates
COUNTER / COMPARATOR BOARD CHIPS IMPLEMENTATION

- App 14 -
FORCE SENSOR CONNECTIONS

FORCE SENSOR SOCKET: (front view)

- Pink - U2/
- Brown - U1
- Blue / Brown - +5V (0.5 mm²)
- Grey - U2
- White - GND (0.5 mm²)
- Black - UO/
- Green - U1/

COUNTER / COMPARATOR CONNECTOR:

Upper side:
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- GND

Lower side:
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- XXX
- +5 V
- U2/
- U1
- U2
- U1/
- UO/
- UO/
- GND

SENSOR - BOARD EXTRA CABLE:

- U1: orange
- U1/: green
- U2: yellow
- U2/: red
- UO/: blue
- +5 V: red
- GND: black

0.5 mm²
SERVO POWER SUPPLY

BBC AXODYN 05 - LV - 05 :

nominal current  : +/- 20 A
nominal voltage : +/- 45 V
peak current    : 40 A
peak power      : 1800 W

This servo-power supply includes an analog P.I.D. controller and a linear transistorised amplifier. It has been design to drive the MC 19 P DC servo-motor (CEM).

The motor package includes a tachometre F 12 T which performs 6 V at 1000 tr/mn.

The assembly servo power supply + DC servo motor + tachometre has a global time constant of 3.2 ms.
DC SERVO-MOTOR MC 19 P

SERIE M
Type
MC 19 P*

NOVEMBER 1978

1) MOTOR RATINGS (1)
11. Rated Torque
12. Rated Speed
13. Rated Output
14. Rated Voltage
15. Rated Current
16. Maximum at Very Low Speed *
17. Pulse Torque (intermittent operation) *
18. Maximum Speed with no external load **

* Stall motor: ask us for max. current
** For other duty cycles: ask us

2) MOTOR CONSTANTS
21. E.M.F./1000 RPM
22. Torque Constant/Ampere
23. Regulation Constant Voltage/cm.N
24. Friction Torque
25. Damping Constant/1000 RPM
26. Terminal resistance (4)
27. Armature Inductance
28. Total Inertia
29. Mechanical Time Constant
30. Power Rate (5)

3) PERFORMANCE CHARACTERISTICS, CONTINUOUS OPERATION

Data given are in practice independent of the armature temperature

1) Mounting on metal plate with thermal insulation (400x400x10 mm), and pure DC supply. Ambient temperature 0° C to 40° C
2) Loss in motor: 10 mm of H2O
3) Allowed cycle 5S, 30 ms, 12
4) Value included rotor plus contact resistance which does not vary with armature temperature since armature resistance increases when contact resistance decreases
5) Calculated from the formula: (Pulse torque) 2

- App 17 -
Here is summed up the jumper configuration used to run TEACH programs:

**INTERRUPT MATRIX:**

<table>
<thead>
<tr>
<th>INTERRUPT</th>
<th>PINS</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>132 -</td>
<td>Hall effect switch left</td>
</tr>
<tr>
<td>1</td>
<td>133 -</td>
<td>Hall effect switch right</td>
</tr>
<tr>
<td>2</td>
<td>124 -</td>
<td>push button (start / stop)</td>
</tr>
<tr>
<td>3</td>
<td>131 - 119</td>
<td>timer 0</td>
</tr>
<tr>
<td>4</td>
<td>130 - 123</td>
<td>int 0 (=) } iSBX counter /</td>
</tr>
<tr>
<td>5</td>
<td>145 - 117</td>
<td>opt 0 (&gt;) } comparator board</td>
</tr>
<tr>
<td>6</td>
<td>128 - 166</td>
<td>int 1 (&lt;=) } connector J3.</td>
</tr>
<tr>
<td>7</td>
<td>127 - 126</td>
<td></td>
</tr>
</tbody>
</table>

**iSBX DATA MODE SELECTION:**

Each iSBX connector has to be set or on 8-bit or on 16-bit data mode. This 24-bit counter/comparator board requires the 16-bit mode, it means to connect pins 208 & 209.
INTERRUPT PROCEDURE CODE LISTING

STMT LINE NESTING
SOURCE TEXT: INT.PAS

1  1  0  0
MODULE INTERRUPT;

$CODE
$INTERUPT( INT )

2  6  0  0
PUBLIC INTERRUPT;

3  7  0  0
VAR I : BOOLEAN;
[---------------------------]

4  9  0  0
PRIVATE INTERRUPT;

5 11  0  0
PROCEDURE INT;

6 12  1  0
BEGIN

7 13  1  1
I := TRUE;

8 15  0  0

ASSAMBLE LISTING OF GENERATED OBJECT CODE

; STATEMENT # 5

0000 06
INT

0001 1E
PUSH DS

0002 50
PUSH AX

0003 51
PUSH CX

0004 52
PUSH DX

0005 53
PUSH SI

0006 57
PUSH DI

0008 E80000
MOV AX,SEG $DATA

000E 8BEC
MOV BP,SP

0010 53
PUSH BP

; STATEMENT # 6

0011 C606000001
MOV I,1H

; STATEMENT # 7

0016 BBEC
MOV SP,SP

0019 5D
POP BP

001A 5F
POP DL

001B 5E
POP SI

001C 5B
POP IX

001D 59
POP DX

001E 58
POP CX

001F 1F
POP BS

0020 07
POP ES

0021 CF
INT CNOP

12/06/86
REFERENCES

FORMER APPLICATIONS AT T.H.E. :

E. Galet : TEACH OPERATIONS WITH A FORCE SENSOR FOR A LINEAR ROBOT ARM.
February 1986, THE - WPB report nr. 255

P.A. Silberfich : DESIGN AND IMPLEMENTATION OF A 16-BIT DIGITALSEROVOCONTROLLER - PJD ALGORITHM.
August 1984, THE - WPB report nr. 121

Van der Burght R.J.M. : POSITIONERING VAN EEN ROBOTARM.
February 1985, THE - WPB report nr. 165

CONTROL THEORY BOOKS :

Gille, Decaulne & Pelegrin : DYNAMIQUE DE LA COMMANDE LINEAIRE.
Dunod - Paris, 1973
THEORY ET CALCUL DES ASSERVISSEMENT LINEAIRES.
Dunod - Paris, 1971
METHODES MODERNES D'ETUDES DES SYSTEMES ASSERVIS.
Dunod - Paris, 1960

Cool, Schijff & Viersma : REGELTECHNIEK
Agon Elsevier - Amsterdam / Brussel, 1972

H.B. Verbruggen : DIGITAL REGELSYSTEM.
Deltse Universitaire Pers, 1982

J.L. Min & J.J. Schrage : ONTWERPEN VAN ANALOGE EN DIGITALE REGELSYSTEMEN.
Nijgh en Van Ditmar Educatief, 's-Gravenhage 1982

TEACHING WITH A FORCE SENSOR :

G. Hirzinger & J. Heindl : SENSOR PROGRAMMING - A NEW WAY FOR TEACHING A ROBOT PATHS AND FORCES/TORQUES SIMULTANEOUSLY.
November 1983, Robot Vision and Sensory Controls

M.H. Raibert & J.J. Craig : HYBRID POSITION / FORCE CONTROL OF MANIPULATORS.
June 1981, Transaction of the ASME - Journal of dynamic systems, measurement and control

G. Hirzinger & Landzettel: SENSORY FEEDBACK STRUCTURES FOR ROBOTS WITH SUPERVISED LEARNING.
1985, IEEE

D.E. Whithney: HISTORICAL PERSPECTION AND STATE OF THE ART IN ROBOT FORCE CONTROL.
1985, IEEE

G. Hirzinger: ROBOT-TEACHING VIA FORCE-TORQUE-SENSORS.
European meeting on cybernetics and systems research
April 1982

Heindl, Johan and Dr. Hirzinger G.: SIMULTANEOUS PATH-FORCE-TORQUE-TEACH-IN FOR ROBOTS, THE SENSOR PROGRAMMING CONCEPT.
November 1983

G. Hirzinger: DIRECT DIGITAL ROBOT CONTROL USING A FORCE-TORQUE-SENSOR.
IFAC real time digital control applications, 1983

INTERPOLATIONS:

H. de Rosemont: THEORY AND APPLICATIONS OF SPLINE FUNCTIONS FOR A SERVOMECHANISM.
July 1985, THE - WPB report nr 0200

A.W. Nutbourne, T.M. McLellan & R.M.L. Kensit: CURVATURE PROFILES FOR PLANES CURVES.
July 1972, Computer Aided Design

Oyvind Bjorke: INTERPOLATION PROCEDURES IN COMPUTER CONTROLLED MACHINES TOOLS.
January 1973, University of Trondheim, Norway

O. Masory & Y. Koren: 3-D CIRCULAR INTERPOLATIONS FOR CNC MANUFACTURING SYSTEMS.
1979, Manufactering systems
REFERENCE-PULSE CIRCULAR INTERPOLATORS FOR CNC SYSTEMS.
February 1981, Journal of engineering for industry, vol 103

A.W. Nutbourne: CURVE CUTTING ON NC MILLING MACHINES.
Computer aided manufacture and numerical control, Cambridge University

H. Kirchhofer: CNC-STEUERUNG MIT VERBESSERTEN INTERPOLATIONSVERFAHREN
ZwF 76, 1981

D. Schmid: INTERPOLATION BEI NUMERISCHEN BAHNSTEUERUNGEN
Steuerungstechnick 2.Jg, 1969

H. Fischer, R. Klinge & C. Schendel: INTERPOLATIONSVERFAHREN FUR BAHNSTEUERUNGEN BEI WERKZEUGMASCHINEN.
Zeitschrift, Siemens, 1966

H. Lienhard: INTERPOLATION VON FUNKTIONSWERTEN BEI NUMERISCHEN BAHNSTEUERUNGEN.
BESCHREIBUNG DES INTERPOLATIONSVERFAHRENS.
Contraves AG, Zurich

INTEL EQUIPMENT DOCUMENTATION

For Intel equipment, see all Intel user's guides, handbook, reference manuels, etc