Teach operations with a force sensor for a linear robot arm

Citation for published version (APA):

Document status and date:
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 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

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.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
TEACH OPERATIONS WITH A FORCE SENSOR
FOR A LINEAR ROBOT ARM

By: Eric Galet
Université de Technologie de Compiègne

WPB-Report nr. 0255
At the end of my training period, I wish to express my acknowledgements to my coach, P.C. Mulders, for his advice and his kindness.

Greatest thanks also to H. Smit, who by his competence contributed to the good march of my trainee.

I also want to thank Rogier, Ineke for creating with the others a nice atmosphere of work and specially Lia who typed this report.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>I.1</td>
<td>1</td>
</tr>
<tr>
<td>I.2</td>
<td>1</td>
</tr>
<tr>
<td>I.3</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>3</td>
</tr>
<tr>
<td>II.1</td>
<td>3</td>
</tr>
<tr>
<td>II.2</td>
<td>3</td>
</tr>
<tr>
<td>III</td>
<td>6</td>
</tr>
<tr>
<td>III.1</td>
<td>6</td>
</tr>
<tr>
<td>III.2</td>
<td>7</td>
</tr>
<tr>
<td>III.3</td>
<td>8</td>
</tr>
<tr>
<td>IV</td>
<td>10</td>
</tr>
<tr>
<td>IV.1</td>
<td>10</td>
</tr>
<tr>
<td>IV.2</td>
<td>12</td>
</tr>
<tr>
<td>IV.3</td>
<td>15</td>
</tr>
<tr>
<td>V</td>
<td>16</td>
</tr>
<tr>
<td>V.1</td>
<td>16</td>
</tr>
<tr>
<td>V.2</td>
<td>16</td>
</tr>
<tr>
<td>V.3</td>
<td>19</td>
</tr>
<tr>
<td>V.4</td>
<td>21</td>
</tr>
<tr>
<td>V.5</td>
<td>23</td>
</tr>
<tr>
<td>V.6</td>
<td>24</td>
</tr>
<tr>
<td>V.7</td>
<td>25</td>
</tr>
<tr>
<td>VI</td>
<td>26</td>
</tr>
<tr>
<td>VI.1</td>
<td>26</td>
</tr>
<tr>
<td>VI.2</td>
<td>28</td>
</tr>
<tr>
<td>VI.3</td>
<td>29</td>
</tr>
<tr>
<td>VI.4</td>
<td>32</td>
</tr>
<tr>
<td>VI.5</td>
<td>35</td>
</tr>
<tr>
<td>VI.5</td>
<td>36</td>
</tr>
<tr>
<td>VII</td>
<td>37</td>
</tr>
<tr>
<td>VIII</td>
<td>38</td>
</tr>
</tbody>
</table>
I. PRESENTATION OF THE WORK ENVIRONMENT

I 1. TECHNISCHE HGOSCHOOL EINDHOVEN

The Eindhoven University of Technology has been founded in 1956. It offers nine courses of study in which students can qualify as graduate engineers specializing in the following subjects:

- Technology in its social application
- Industrial engineering and management science
- Mathematics
- Computing science
- Technical physics
- Mechanical engineering
- Electrical engineering
- Chemical engineering
- Architecture, structural engineering and urban planning.

Since the Eindhoven University of Technology opened in 1957 more than 6,600 students have graduated from it. The degree of doctor in the technical sciences can be obtained by students submitting a doctorale thesis on research, they have carried out.

I 2. THE EDUCATION AT THE UNIVERSITY

A full University course in the Netherlands used to take at least 4 years. University studies are divided in two phases. The first phase has a duration of 4 years and comprises two examinations: The first or preliminary examination at the end of the first year and the final examination at the end of the fourth year. Students are allowed 2 extra years to complete this first phase. The first examination has to be passed at the end of the second year at the latest. The second phase will be introduced in the academic year 1986/87.

It covers three types of training:
- Further professional training as physician, pharmacist, dentist, veterinarian with a maximum of two years.
- Professional training of teachers for secondary school with a maximum of 1 year.
- Training for research and technological design with a maximum of 1 or 2 years.

I 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 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 its four divisions:

- fundamentals of mechanical engineering
- product design and development
- apparatus design for industrial processing
- production engineering and automation.

In this department, there are about two hundred employees (teachers, technical personnel) and nine hundred students.
II - PRESENTATION OF MY WORK.

II 1. THE RESEARCH PROGRAM 'FAIR'.
   (Flexible Automation and Industrial Robots).

The Research Project 'FAIR' has been established for research in the field of Flexible Automation and Industrial Robot. It's financed and directed by the Dutch government. At the Eindhoven University of Technology, the mechanical and electrical engineering departments are involved in this project. They try to find an approach to improve the flexibility with the aim of designing components for flexible automation equipment. In addition of researchers, students carry out their graduate work on detailed problems of the program. I had to work with some of them.

The project is divided among several parts:

1. General aspects of fixed and flexible automation
2. Parts feeding and handling
3. Kinematics, dynamics, design aspects
4. Drive and control system, programming
5. Sensor control Welding.

II 2. MY PRACTICAL WORK

At the University, in view of Flexible Automation a one dimensional robot arm has been developed in order to get more experienced with problems such as position regulation, welding, etc.

According with that aim, the subject of my work was defined as follows: Development of a force sensor and hardware on a one dimensional robot arm to achieve teach operations.

The interest of that report is to describe the realization of programming a robot arm by teaching. The operator should be able to program the robot using the "Teach" mode, running the robot through the desired movements and then record the moves for later reproductions or play-back. There are several ways to guide the robot through the required path: either the use of a remote manipulation system (joystick, teach box) or by a manual drive of the robot body. Our interest will be for the second possibility.
One might think that the simplest method is by physically grasping the robot end effector and leading it through the required path at the required speed, while simultaneously recording the continuous position. There are robots in which this method can be applied (by disengaging the motors in the case of electric robots or reducing the oil pressure in case of hydraulic ones). However because of the transmission elements (such as gears and leadscrews) in the robot manipulator, it might be impossible to generate a motion of the robot joints by pulling its end effector. Rather than diminishing the effects of the transmission system, the idea is to take advantage of them to move the robot. The figure below describes the general configuration.

During the teach mode the robot is used as a sensory controlled robot. The direction of the desired movement is given by a three-dimensional force sensor grasped by the operator. Then the control of the robot is a function of the information sensed from the end effector: a processor unit drives the different motors and handles at the same time position devices which allow you to store either the path end points or continuous positions.

My practical work was to perform that teach operation for a one-dimensional robot arm fitted with a force sensor.

The project included the design and implementation of teaching operation on an Intel Single Board Computer 86/05. Further requirements are accuracy of recorded positions.
The first part of my job was to decide a planning of my time, or at least to define the different steps to take. My coach and my colleagues helped me much because they knew which equipment might be used.

The first part of my report concerns a general inquiry about teach operations. One can find this information in every book about robotics.

Afterwards, I received precious help from my colleagues, so as to get the required knowledge about the equipment such as the robot. However, that gave me some problems, because the previous reports about this subject are written in Dutch.

Anyway, that first approach permitted me to design a first system. After improving the hardware possibilities, for instance the Intel Single Board Computer ISBC 86/05 and the interfaces (analog to digital, digital to analog converters, incremental linear transducer), I’ve written a program to complete my job.
III . THEORY ABOUT TEACHING OPERATIONS

To realize this teaching operation project, it might be useful to inquire, at first, about the theory of that programming method. When studying the actual methods for programming a robot, teaching operations with a force sensor appear to be a combination of them which includes advantages of each method.

The most commonly used programming method for servo controlled robots is direct training of the robot. This method is called “Teaching by doing”, otherwise programming is carried out using a language. All methods of programming by training are based on a physical demonstration of the task to be performed.

There are three methods of moving the robot:

- manual displacement method (without any power transmitted to the motors)
- the dummy arm method (master-slave)
- telecontrol method (by teach box or syntaxer).

III 1 . MANUAL DISPLACEMENT METHOD

The human operator moves the robot manually. For the 'point to point' robots, when the required configuration is reached, the recording is activated by a command switch. Such method can only be used if the system allows motion by direct action on the axes and not only by the motors. Then the robot is called reversible. The final condition is that the forces required to move the robot must be compatible with the physical strength of the human operator. With 'Recorded Trajectory Robots', the trajectory is recorded by sampling. In both cases, manual movement will not be very precise, because the positions recorded without any power do not correspond to the positions when motors are controlled.
The manual movement method is used with small robots and in case the degree of precision required is not very large.

III 2. THE 'DUMMY' ARM METHOD (Robot simulator - Teaching arm)

The robot is replaced by a mechanical structure with identical geometry but unmotorized and very light. It's equipped with position feedback devices. The operator teaches the end points or the path to the robot and is not hampered by the mass and actuators of the robot.

Although the advantages of the robot simulator method are the simplicity of direct programming and small human force required, it has also some disadvantages:

- The problems of precision are the same as before, with the added differences that may exist between the dummy and the robot (structure, sensors) and between the position of the dummy and that of the robot.
- This method is manual, a high precision cannot be achieved.
- An investment in a simulator is required.
III 3. THE TELECONTROL METHOD

The robot is moved by its actuators which are controlled by the operator. For 'Point to Point' robot control is established using a control box or teach box or teach pendant. The control box is equipped with an array of switches which allow the degrees of freedom (D.O.F) of the robot to be actuated one by one, until the combination of all axial positions gives the direct position of the robot. The technique of moving axes using binary switches is extremely difficult to make. The coordination of several DOF is virtually impossible so the operator works sequentially on one DOF at a time.

In order to provide teaching flexibility, several teach modes are developed using coordinate transformers: world, tool.

In the world mode, the robot moves in a reference coordinate system fixed in the robot arm base. In the tool mode, the tool moves in its own coordinate system. The robot computers calculates in real time the required individual joint motions. With those methods, performance are improved.

A new method is developed to solve more problems: a syntaxer or joystick allows real control of the robot, by acting simultaneously on several DOF without constraint. The syntaxer is like a dummy arm, but total precision is not essential, since the human can observe the effect of his commands. The only fault which is found with this method is that the human operator is not longer in direct contact with the robot.
Because each of those three methods has advantages and disadvantages, it may be interesting to find a compromise of them.

Several aspects may be considered:

- the method must be compatible with the strength of the human (as the dummy method)
- the repeatability of the recorded trajectory and the executed trajectory must be optimal: the actuators must not be inerted
- the operator must have a physical contact with the robot in order to realize the desired task.
- the task of the operator may not be complicated so as to limit the number of trials and to reduce the time while the robot is tied up during teaching mode.

Teach operation method with a force sensor reaches these conditions. It appears as a compromise of manual displacement, robot simulator and telecontrol methods.

While this method has much advantages, it has at least one disadvantage:

- In some applications, when a gripper is used, then it may be difficult to integrate a force sensor method to teach the movement of the gripper. So you may use another device (Teach pendant, syntaxer, dummy gripper). Anyway a welding operation doesn't require any gripper.
IV. FIRST APPROACH, FIRST DESIGN.

I had to apply the theoretical aspect of the teaching operations to the Linear Robot Arm. I studied the possibilities it offers in order to design a first system.

IV 1. THE LINEAR ROBOT ARM.

The linear Robot Arm is constructed of two parts: a mechanical part and a drive system. The figure below shows the Linear Robot Arm.

The mechanical part is one part mobile with a screw and the motor.
The drive unit contains a power amplifier, an actuator (DC servo-motor) and also a tachometer.

The motion is controlled by a closed-loop system. That system compares speed references with feedback signals to determine the error, which is amplified and used to generate drive motions. The control of the motion of the arm commands the rotational motor speed by manipulation of the motor voltage. The robot arm is treated as a load disturbance acting on the motor's shaft. Then the variations in the movement of inertia affect only the time constant of the response but don't result in any unwanted consequences and don't affect the required time to reach the target position. This control approach by manipulating the DC-motor voltage, fits with our application. For the welding the specification of velocity is appropriated.

The figure below shows the analog servo-system.
IV 2. THE EQUIPMENT OF THE LINEAR ROBOT ARM.

In addition to the drive system, the Linear Robot Arm is also equipped with other devices:

- a force sensor
- two 'Hall effect' Digital Switches
- an incremental linear transducer.

**The Force Sensor**

The force sensor has been built by another stagiaire of the THE. It measures the component of the force in the direction of the spindle. The signals sensed by the strain gauges are amplified and sent as a voltage value between -4 and +4 Volts.

An Analog to Digital Converter transforms the measured voltage to a binary data which will be used as input for a computing system.

**The 'hall effect' switches**

The role of the 'hall effect' switches is to be able to sense the end position of the arm. When the end position is reached a pulse of +5 V is sent. (appendix J)

**The incremental linear transducer.**

The incremental linear transducer allows us to measure the position. The output signals \((K_1, K_2)\) of the transducer are two square-wave signal trains, TTL-compatible and a reference mark used for initialization of measures.

Because it's an incremental system, the output signals are not usable, directly, to count. We might transform them in order to get other signals which contain more information, such as the direction of the movement.

A card was realized to perform that purpose. The pulse/edge converter card is described in appendix F in more details.

The figure below shows the principle of the conversion.
OUTPUT SIGNALS FROM THE INCREMENTAL LINEAR TRANSDUCER

K1

K2

pitch 40 \mu m

OUTPUT SIGNALS OF THE PULSE/EDGE CONVERTER

Countup

Countdown

Pitch 10 \mu m

The signals K1 and K2 are transformed to COUNTUP and COUNTDOWN signals. Also the pitch is reduced from 40 \mu m to 10 \mu m, that gives a higher precision.

Although some progress have been done, the position counting is not yet complete. We had to find a solution to use those signals (COUNTUP and COUNTDOWN).

Position counting solutions:

Most of the developments required for our application have typical solutions (such as analog to digital conversion). However we wanted to improve a new method to check the position.
Previous realisations with the robot arm have used a home-made computer board fitted with a 8085 Intel Microprocessor 8-bits. This product will be replaced because it's getting too old and too slow, by the INTEL Single Board Computer 86/05 fitted with an 8086 Intel Microprocessor-16 bits. This computer gives new possibilities for us to handle the development of the equipment. We will describe it more detailed in paragraph The problem of position counting can be managed in several ways. We had at least three solutions with advantages and disadvantages.

- use the on-board counters; This method could be very useful, but for our case, all counters were already used (sample time of the force, serial communication) and then not usable for that purpose.

- another solution can be an external card fitted with required counters and latches to allow parallel I/O communications with the computer: this solution as the previous one is quite easy to handle but the low reliability of the hardware card is a disadvantage.

- at last, I could use a software solution. For example, I can receive the pulses through the interrupt lines of the computer board and count them by some software. In this case we don't need any new hardware. It's an important advantage.

We had to make a choice: it would be the third method, because we did like to have little hardware.
IV 3. DESIGN OF THE SYSTEM.
V. DEVELOPMENT OF THE EQUIPMENT.

In this part, we will describe the development of the required equipment with the help of Intel products environment:

- The Intellec serie III micro computer and its features
- The Isbc Single Board computer 86/05.

and then we'll look at the implement of Interfaces between the computer and the robot's devices:

- Analog to Digital conversion
- Digital to Analog conversion
- The Edge/pulse converter.

V 1. INTELLEC SERIES III MICROCOMPUTER DEVELOPMENT SYSTEM.

The INTELLEC serie III micro-computer development system is more than a keyboard, a video display and disk drives. It's a real tool for designing microcomputer software for the IAPX 86/88 processor.

I used this system to program Assembly for the 8086 and also Pascal '86. We are also able to write programs, debug them, link them, locate them and run them on the system itself, or on its boards.

V 2. THE ISBC 86/05

a) The ISBC 86/05 Single Board Computer and the INTEL environment

The ISBC 86/05 single board computer is a member of Operational Equipment Manufacturers microcomputer systems.

The ISBC 86/05 is a complet computer system on a single printed circuit card. The CPU, System clock, read/write memory, non-volatile ROM, I/O ports and drivers, serial communication interface, priority interrupt logic and programmable timers, all reside on the board. Multibus interface logic is included to
offer capability to communicate with Single Board Computers, expansion memory options, digital and analog I/O expansion board and peripheral controllers.

We used the ISBC 957 Intellec-ISBC 86/05 Interface and execution package which provides the hardware and software required to interface an ISBC 86/05 Board with an Intel Intellec Microcomputer Development system.

So after working the programs on Intel Microcomputer Development System, we run the program on ISBC 86/05.

b) The possibilities of the ISBC 86/05

The ISBC 86/05 Single Board Computer is a complete computer system on a single printed-circuit. It includes a 16-bit central processing unit (CPU), 8 K bytes of dynamic RAM, a serial communication interface, three programmable parallel I/O ports, programmable timers, priority interrupt control, expansion to be interfaced with other boards.

Among its features, I'll speak about those which are of special interest for my project.

- The ISBC 86/05 includes 24 programmable parallel I/O lines implemented by means of an Intel 8255A PPI (Programmable Peripheral Interface). The system software is used to configure the I/O lines in any combination of bidirectional I/O and bidirectional port.

- The RS 232C compatible serial I/O port is controlled and interfaced by an 8251A USART (Universal Synchronous/Asynchronous/Receiver/Transmitter) chip.

- Three independent, fully programmable 16-bit interval/event counters are provided by an INTEL 8253 PIT (Programmable Interval Timer). Each counter is capable of operating in either BCD or binary mode, two of these counters are available to generate accurate timers.

- The Intel 8259A PIC (Programmable Interrupt Controller) can handle up to eight interrupts. By using external PIC's slaved on the on-board 8259 (Master), the interrupt structure can be expanded to handle and reduce priority of up to 64 sources.

The PIC, which can be programmed to respond to edge-sensitive or level-sensitive inputs. After resolving the interrupt priority, it sends a single interrupt request to
the CPU. Interrupt priorities are independently programmable by means of software control.

The CPU includes a non-maskable interrupt (NMI) and a maskable interrupt (INTR).

The 8259A provides an 8-bit identifier of the interrupting source to the CPU. The CPU multiplies the 8-bit-identifier by 4 to device a pointer to the service routine for the interrupting device.

The first step was to carry out the configuration and installation procedures of the Single Board Computer, for our defined environment.

A variety of jumper and switch options allow the user to configure the board for his particular application. You may find in appendix x, the list of most required jumpers switches we used.
V 3. INTERFACE FORCE SENSED/ISBC 86/05 COMPUTER.

The features of the analog force sensed are as follow:
The signal issued from the strain Gages is amplified by a MESVERSTARKER KWS/35-5, in a range from -4 V to +4 V.

That analog bipolar voltage is then input in the analog to Digital Converter Card fitted with a AD 570 (appendix H) which accepts bipolar (-5V to +5V) analog inputs and performs a 8-bit conversion in 25 ns.

Control description of the ADC

As the BLANK and CONVERT input is driven low, the outputs will be open and a conversion will commence. Upon completion of the conversion, the DATA READY line will go low and the data will appear at the output. Pulling the BLANK and CONVERT input high blanks the outputs and readies the ADC for the next conversion. (see figure below).

CONTROL OF THE A.D.C

Then to perform this conversion, I might do the following interfacing to ISBC 86/05:

- The parallel I/O port B, bits 0 to 7, is reserved to input the 8 bits sent by the ADC CARD. To handle those inputs, we've installed input terminators on required IC sockets to interface port B with Intel 8255 A PPI.

- The parallel I/O port C bit 0 handles the output BLANK and CONVERT. A driver SN 7405 interfaces port C bit 0 with Intel 8255 A PPI.

- The parallel I/O port C bit 4 handles the input DATA/READY it is interfaced with intel 8255 A PPI by an terminator. The ADC CARD is detailed in appendix.
A.D.C CARD & ISBC 86/05 CONFIGURATION
V 4 INTERFACE ISBC 86/05 VOLTAGE INPUT OF THE MOTOR DRIVE SYSTEM

The interface computer/input voltage of the motor drive should allow us to command the motor by inputting in its drive system a voltage from -10V to +10V.

The DAC-08 MULTIPLYING CONVERTER (appendix G) performs the Digital to Analog conversion in 85 ns.

We used a bipolar output operation as follow:

<table>
<thead>
<tr>
<th>Hexa</th>
<th>bit</th>
<th>0 1 2 3 4 5 6 7</th>
<th>OUTPUT VOLTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>1 1 1 1 1 1 1 1</td>
<td>- 10 V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>1 0 0 0 0 0 0 0</td>
<td>- 50 mV</td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>0 1 1 1 1 1 1 1</td>
<td>+ 30 mV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0 0 0 0 0 0 0 0</td>
<td>+ 10 V</td>
<td></td>
</tr>
</tbody>
</table>

General description:

The ADC-08 is implanted on a card (see appendix ); you are able to modify gain and offset of the amplification stage, through potentiometers.

The parallel I/O port, bits 0 to 7, handles this output. The interface port A/INTEL 8255A PPI is realized with a bidirectional data buffers.
V 5 . THE SWITCH DEBOUNCER.

To record a wanted position, you have to press a button. The button is a mechanical switch. It sends an interrupt to the on-board Programmable Interrupt Controller. It appeared that the switch contacts did not open or close cleanly: at closure there are several separate contacts over a period of many microseconds. It gave more than one interrupt. A switch debouncer removes the bounce by responding to only one of the voltage excursions during each switch operation.

The figure below shows the switch debouncing circuit using NAND gates (appendix I ).

The output is connected to the interrupt matrix of the ISBC 86/05 Board (see paragraph V.7 ).
V 6. THE 'HALL EFFECT' SWITCHES CARD.

The signals emitted by the two 'Hall Effect' switches don't have a nice shape.

To fit up the waveform, I used a NAND SCHMITT TRIGGER Integrated Circuit (appendix I), for my pulse shape card. At the moment of switching, the circuit increases the edge of a pulse.

The figure below describes the circuit and the waveform correction. The outputs $S_1, S_2$ are connected to the interrupt matrix of the ISBC 86/05 Board. (See paragraph V.7).
V 7. THE HARDWARE INTERRUPTS.

Most of the informations issued from the Linear Robot Arm were used as interrupt.

ZP ............................................. Zero Postion
COUNTUP, COUNTDOWN .................... Incrementation,
                                      Decrementation of the position
S_1 S_2(1) .................................. End switches
REC. ......................................... Recording Button

In addition of these external interrupt requests, we connected the output interrupt line TIMERO INT of a on-board Intel Programmable Interval Timer 8253A.

Further, the purpose of that interrupt will be described. The jumpers of the interrupt matrix description is given in appendix.

(1) The End Switch S_2 is near the Motor on the Linear Robot Arm. S_1 is the other End Switch.
VI. THE SOFTWARE.

VI 1. THE BASIC FLOWCHART OF THE PROGRAM.

The program may be divided in three parts.

- **INITIALIZATION**: The operator enters the parameters for the sample time of the force, for the P.I.D. The hardware devices are initialized (PIT, PIC, PPI).

- **ZERO POSITION**: We search the zero position of the incremental linear sensor. The robot is moving from the left to the right until the ZP interrupt occurs, then immediately the position counting starts.

- **TEACH OPERATIONS**: The value of the force exerted on the gripper by the operator is sampled at a constant time (1 ms), then the corresponding command motor is sent to the drive system.

The operator is enabled to record the positions he desires, to change the parameters and to stop the program.

You may find next page the flowchart of the program. Further, we explain why and how we were induced to use the Assembly Language in the interrupt procedures (particularly for Count Up and Count Down).
THE FLOWCHART OF THE PROGRAM

**Beginning**

- **Initialization of the hardware**
- **Choice of the parameters**

**Interrupt Procedures**

- **Initialization Switch Limit Switch 2 init**

- **ENABLES**

- **Enregistrement Count up Count down Interrupt 1 12 Switches 12 Switch**

**INPUT FORCE**

**RESET COUNTER**

**OUTPUT COMMAND**

**VOLTAGE**

- **? INHRE=**

- **Print recorded Position**
  - **The motor is stopped**

**STOP**
VI 2. EXPLANATIONS OF THE INTERRUPT PROCEDURES.

***PROCEDURE INTZEROINIT

The accuracy of the program depends on that interrupt procedure, because as soon as the ZERO POSITION is found, the position counting should begin. The subroutine sends a False Value to the boolean variable ZERO and then enable the COUNTUP and COUNTDOWN interrupts.

***PROCEDURES SWITCHE1INIT, SWITCHE2INIT.

These procedures change the direction of the displacement of the Linear Robot Arm during the zero position research.

***PROCEDURES COUNTUP, COUNTDOWN

These procedures increment or decrement the position value (Variable POSITION). Each interrupt should be handled quickly, in order to be able to handle the next interrupt, COUNTUP or COUNTDOWN. Because if I miss some interrupt, the reliability of the accuracy decreases.

***PROCEDURE ENREGISTREMENT

It sends the True Value to the boolean variable ENREG.

***PROCEDURE INTTIME1

It sends the True Value to the boolean variable TIME1.

***PROCEDURES I1SWITCHE, I2SWITCHE

When one of these interrupt occurs, the operator has reached the end of the Linear Robot Arm.

These procedures change the direction of the displacement of the Linear Robot Arm and send True Value to the boolean value SWITCH.
VI.3. THE INTERRUPT HANDLING.

In the program, we had to handle an interrupt processing. The mechanism has two states:

- preparation of the PIC for the hardware environment (initialization)
- associate the interrupt procedures written in assembly with each particular interrupt.

a) Initialization

The on-board 8259 A PIC handles eight vectored priority interrupts. During the interrupt processing, it determines which interrupt requests is of the highest priority; It issues an interrupt to the CPU based on the determination of the interrupt level and sends the CPU a vectored restart address.

To realize that operation, the PIC requires an initialization. In our particular environment, that initialization is carried out with three Command Words (ICW1, ICW2 and ICW4). My interest won't be for ICW1 and ICW4; Any information about them may be found in the Intel documentation.

I will detail the command word ICW2.

ICW2 represents the vectoring byte required by the CPU during the processing interrupt. It allows you to choose where in the memory the compiler will write the start address of the different interrupt procedures.

```
ICW2       b7 b6 b5 b4 b3 b2 b1 b0
            MSB       LSB
```

The five most significant bits (b7 to b3) are supplied by the programmer. The three other bits (b2 to b0) correspond with the interrupt level.

The CPU during the interrupt process, multiplies the vector byte by four and that value is used as the address where the start address (4 bytes: offset and base) of the corresponding interrupt is written.

Example
-PIC programming word ICW2:

ICW2: (MSB) 0000 1000 (LSB)
(when programming the bits 0 to 2 are 0)

- Vectoring byte used during the interrupt processing, when the interrupt level request is 2:

<table>
<thead>
<tr>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Choice of the programmer: Interrupt Level

The Vector Byte is (0 A) H.
Then the Vector address is (28) H
In the memory we have:

Memory address: (28)H (29)H (2A)H (2B)H

datas: offset Base

Now we have to associate the interrupt procedures with the desired interrupts. That is done in the program written in PASCAL-86. In PASCAL-86, we can control the compilation by using compiler controls that allow us to specify options.

**INTERRUPT** is a compiler control, it designates procedures as interrupt procedures and generates the interrupt vector. You can optionally specify a number for each procedure which represents, multiplied by 4, the interrupt location of the entry of the interrupt procedure. In my case, I did not write this optional number because further I use the procedure SETINTERRUPT, which overrides it.

More, four procedures are provided to aid interrupt processing

- ENABLEINTERRUPTS
- DISABLEINTERRUPTS
- CAUSEINTERRUPT
- SETINTERRUPT

My interest will be for SETINTERRUPT.

SETINTERRUPT provides a way to dynamically associate an interrupt procedure (declared by the compiler control INTERRUPT) with a
given interrupt number. The number should be the same as the vector byte of the given interrupt level described previously.

I continue my example:

The vector byte of interrupt level 2 was (0A)\textsubscript{H}, (10) decimal. I take two interrupt procedures \texttt{INTER1}, \texttt{INTER2}. Then, we have in the Pascal-86 program:

\begin{verbatim}
INTERRUPT(INTER1, INTER2)

SETINTERRUPT (10, INTER1)

SETINTERRUPT (10, INTER2).
\end{verbatim}

So I have changed the interrupt procedure for interrupt two, this method I used in our program.
VI 4 . THE USE OF ASSEMBLY-86 INTERRUPT PROCEDURES.

The interrupt procedures were written in Assembly-86. Below is explained why and how.

The paragraph IV.2 describes the incremental linear transducer and its converter. We remember that the pitch of the output signals COUNTUP and COUNTDOWN is 10 μm. If the speed of the robot is 1 m/s. The period of the counting becomes 10 μs. Then at the most, we have to handle an interrupt (COUNTUP or COUNTDOWN) every 10 μs. This is a short period and it took me much work to find a reliable software solution to handle these interrupts.

At first I wrote, the interrupt procedures in Pascal-86. The use of the compiler control, "$ CODE" gave the listing of approximated assembly code of those Pascal-86 routines.

The program below is the listing of an incremental operation:

```
INTUP PROC NEAR
PUSH ES
PUSH DS
PUSH AX
PUSH CX
PUSH DX
PUSH BX
PUSH SI
PUSH DI
MOV AX, SEG DATA
MOV DS, AX
PUSH BP
MOV BP, SP
PUSH BP
MOV AX, POSITION
INC AX
MOV POSITION, AX
MOV SP, BP
POP BP
POP DI
POP SI
POP BX
POP DX
POP CX
POP AX
POP DS
POP ES
IRET
```
The execution time of each operation depends on the code and effective address calculation time. The Intel documentation "ASM86 Language Reference Manual" allowed us to calculate the execution time. The program, it takes 260 clock cycles at 8 MHz, this gives 32.5 \( \mu \)s.

We see that the Pascal-86 saves all the registers when an interrupt occurs. In fact this is not necessary, it is enough to save only the registers I may use in my assembly subroutine.

For example, I may save AX Register because it's used in INC operation. To avoid those useless safeguards, I decided to write the interrupt procedure in ASM-86.

More, using the register AX is not necessary if POSITION is an integer. However, Pascal-86 uses it and then it increases the run-time.

You can see the difference in the following listing, which realizes the same incrementation:

```
INTUP PROC NEAR
    PUSH AX
    MOV AX, POSITION
    INC AX
    MOV POSITION, AX
    POP AX
    IRET
```

At the most, this program takes 40 clock cycles 5 \( \mu \)s.

Calling an assembly language subroutine from a Pascal program gave as much problems, because we had to provide the interface specification between the both languages. I had to take care of the data passing and data definition conventions used to link different language programs. Simply put, the assembly language code that talks to Pascal 86 must do what pascal code expects it to do.

I continue my example with the interrupt procedure COUNTUP. The listing below is the program I used with the required interface conventions. I will explain them.
A program written for the 8086 is combined in memory with segments. There are 4 different segments:

- code segment
- data segment
- stack segment
- code segment

I could choose the model of segmentation: the model of segmentation specifies how program segments are archived in memory.

I used the SMALL model which offers the highest code and fastest execution time: then code is one segment, data and stack another. With the control -CONST IN DATA-, the constants are stored in the data segment. With the SMALL model, all procedures should be given type NEAR.

I must also connect the different segment registers (CS, DS and SS) used in assembly declarations with the segments (such as segment CODE) and variables (POSITION) declared in Pascal-86. This was done by the ASSUME directive, which makes the addressability of the code and data.

However, against these precautions, during the linkage of the different modules, a warning appeared. But it was not fatal and then can be ignored in the locating process.
VI 5 . REMARKS ABOUT THE PROGRAM.

I will describe the role of each module I've written.

Each file contains a module:

File :       MAIN.PAS   PARAM.PAS   INTPOS.PAS
Module :     MODUMAIN1   MODUPARAMS   MODURECHERCHEZERO

File :       INIT.PAS   CONVER.PAS   POSASM.ASM
Module :     MODUINITARD   MODUCONVER   MODUSUBASSEMBLEUR

In each module, you have one or more procedures or functions.

MODUMAIN1:

This module is the main module, it calls the other modules.
It contains the procedures:

- INPUTFORCE
- CALCULATION: the function COMMAND MOTOR = f(FORCE) is a
  P.I.D. The P.I.D (the parameters TS, TD and TI) has not
  been improved. Then it may be the reason of further
  studies of the linear robot arm.

MODUPARAMS:

- LECTURE reads and prints 4 characters input from the
  keyboard.
- ASCII4 TO INTEGER4 converts 4 hexadecimal characters in an
  integer.
- CHOICEPARAMETERS inputs the parameters (Sample time of
  the force and P.I.D parameters).

MODURECHERCHEZERO

- INITPOSITION researches the zero position and declares
  the interrupt procedures.

MODUINITARD

- INITSAMPLETIME resets the counter Q
- INITPPI initializes the PPI
- INITPIC initializes PIC

MODUCONVER
- this module contains the procedures to convert an integer in 2, 3 or 6 characters and then prints them on the screen.

MODUSUBASSEMBLEUR contains all the assembly interrupt procedures.

VI 6 . CONCLUSION

The program I've written carries out point-to-point teach operations for the linear robot arm. The position counting by the software method is a success. Although the precision of the recorded positions is reliable, the maximum speed I can reach during the teach operations is less than 1 m/s.

Working with interrupts and with different languages was very interesting and beneficent for me.
VII CONCLUSION

During my practical work at the University of Technology of EINDHOVEN, I learned how to develop a project in research view.

It's important to mention that I met some problems at the beginning, because I had to discover a system for which all the informations are written in dutch. However I performed the job as people expected: then I learned much about Microprocessor, languages and real time problems.

The following step is to improve the system for a continuous path teach operation program and then performing the playback of the recorded movements.
VII. APPENDIX:

- summary in french
- documentation
- equipment informations
Introduction au Pascal by pierre Lebeux, 1980, ed. sybex

The art of the digital design by david Winkel, 1980, ed. prentice-hall

Philips Data handbook, Integrated circuits, 1983

INTEL documentation:

iAPX 86,88 Family utilities user's guide

ASM86 Language reference manual

ASM86 Macro assembler operating instruction

Pascal-86 user's guide

ISIS-II user's guide

Intellec serie III microcomputer development system
console operating instructions

iSBC 957 intellec- iSBC 86/12 interface and execution package

Components data catalog 1981

iSBC 86/05 single board computer hardware reference manual

iSBC 86/12 single board computer hardware reference manual
APPENDIX C

Interrupts Jumpers Configurations

<table>
<thead>
<tr>
<th>FROM</th>
<th>JUMPERS</th>
<th>JUMPERS</th>
<th>TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designation</td>
<td>POSTSin</td>
<td>POSTSout</td>
<td>Designation</td>
</tr>
<tr>
<td>Zero position</td>
<td></td>
<td>132</td>
<td>INT(0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Highest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Priority</td>
</tr>
<tr>
<td>Count Up</td>
<td></td>
<td>133</td>
<td>INT1</td>
</tr>
<tr>
<td>Count Down</td>
<td></td>
<td>124</td>
<td>INT2</td>
</tr>
<tr>
<td>REC</td>
<td></td>
<td>131</td>
<td>INT3</td>
</tr>
<tr>
<td>(S_1) (end Switch)</td>
<td>119(*)</td>
<td></td>
<td>INT4</td>
</tr>
<tr>
<td>(S_2) (end Switch)</td>
<td></td>
<td>130</td>
<td>INT5</td>
</tr>
<tr>
<td>TIMER0 INTR</td>
<td>123</td>
<td>128</td>
<td>INT6</td>
</tr>
<tr>
<td>priority</td>
<td></td>
<td></td>
<td>lowest</td>
</tr>
<tr>
<td>Power Fail INT</td>
<td>129</td>
<td>120</td>
<td>NMI Gate</td>
</tr>
</tbody>
</table>

(*) This interrupt is connected to the recording button via connector J1, pin 50.
RUN
:FI:PASC36 MAIN2.PAS SMALL (- CONST IN DATA -)
:FI:PASC36 PARAMS.PAS SMALL (- CONST IN DATA -)
:FI:PASC36 INITI.PAS SMALL (- CONST IN DATA -)
:FI:PASC36 INIT2.PAS SMALL (- CONST IN DATA -)
:FI:PASC36 CONVER.PAS SMALL (- CONST IN DATA -)
LINK36 MAIN2.OBJ, PARAMS.OBJ, INITI.OBJ, INIT2.OBJ, POSASM.OBJ, CONVER.O
SBCIOS.LIB, :FI:PASC36N0.LIB, :FI:PASC36N1.LIB, &
:FI:RTNULL.LIB, &
:FI:B7NULL.LIB &
TO TEACH.LNK
LOC36 TEACH.LNK TO TEACH RESERVE ( 01180H TO 011D0H )
OH36 TEACH TO TEACH.HEX
EXIT
PUBLIC MODUMAIN1;
CONST
[* PPI DEFINITIONS *]
PORTAADDR = OC8H; [* OUTPUT COMMANDE MOTEUR *]
PORTBADDR = OCAH; [* INPUT FORCE *]
PORTCADDR = OCCH; [* OUTPUT C&B BIT 04 *]
SETBIT4CC = 009H;
RESETBIT4CC = 009H;
IMASKREPORT = OC2H;
TYPESTRING6 = PACKED ARRAY [1..6] OF CHAR;
STRING3 = PACKED ARRAY [1..3] OF CHAR;
STRING2 = PACKED ARRAY [1..2] OF CHAR;
STRING13 = PACKED ARRAY [1..13] OF CHAR;
STRING4 = PACKED ARRAY [1..4] OF CHAR;
VAR
STRING6:
STRING2:
STRING3:
POINTS: PACKED ARRAY [1..100] OF INTEGER;
K,TS,TI,TD:INTEGER; [* VARIABLES USED BY THE PROCEDURE CAL- *
[* CELLED ( COEFFICIENTS OF THE P.I.D ) *]
TSREAL:INTEGER; [* SAMPLE TIME USED BY THE COUNTER *]
POSITION,ENDPOSITION,POSWANTED:INTEGER;
STARTPOSITION:INTEGER;
MODE1,MODE2,SWITCH,ENREG,TIME1,POSREACHED:BOOLEAN;
ZERO:BOOLEAN; [* MISE À FALSE QUAND ON PASSE PAR ZERO *]
COMMZEROINIT:INTEGER;[* VITESSE DU BRAS AU COURS DE LA RECHERCHE *]
[* DU ZERO Pull:010h; Push:0fOh *]
IJ:INTEGER; [* COMPTET TABLEAU DES POSITIONS ENREGISTREES *]
COUNT:RECORD CASE BOOLEAN OF
TRUE:(FULLWORD:WORD);
FALSE:(LOWHIGH:0..255);
END;
PROCEDURE DEBUT;
PUBLIC MODUPARAMS;
PROCEDURE CHOICEPARAMS(VAR K,TS,TI,TD,STARTPOSITION:INTEGER;MODE1,
MODE2:BOOLEAN);
PROCEDURE PRINTASCII13(WORD13:STRING13);
PROCEDURE PRINTASCII4(WORD4:STRING4);
PROCEDURE FINENREGISTREMENT;
PROCEDURE LIN;
PROCEDURE TAB;
PUBLIC SECTIONS;
PROCEDURE CIA(char:CHAR);
FUNCTION CI:CHAR;
PUBLIC MODURECHERCHEZERO;
PROCEDURE INITPOSITION;
PUBLIC MODUSUBASSEMBLEUR;
PROCEDURE INTZEROINIT;
PROCEDURE COUTUP;
PROCEDURE COUTDO;
PROCEDURE COUTUP2;
PROCEDURE COUTDO2;
PROCEDURE SWITCHINIT;
PROCEDURE SWITCH2INIT;
PROCEDURE INTERERNE;
PROCEDURE ENREGISTREMENT;
PROCEDURE INTERTIME1;
PROCEDURE I1SWITCH;
PROCEDURE I2SWITCH;
PUBLIC MODUINITHARD;
PROCEDURE INITPPI;
PROCEDURE INITIALISATION;
MODULE MODUMLAIN; 
$INCLUDEMODUMLAIN; 

PROGRAM MODUMLAIN; 

CONST 
PITCONTRADDR = 0D6H; 
TIMER0CONTRWORD = 030H; 
VAR OKGO, OKGEOCH, OKKREPOS:CHAR; 
READYCHAR, FORCECHAR:CHAR; 
READY, COMOTINT, COMOT, FORCE, FORCEINT, V, S, 0, F:INTEGER; 
CONSTSTI:CONSTDTS:INTEGER; 
I:INTEGER; 

PROCEDURE INPUTFORCE(VAR force:CHAR;VAR forceint:force:INTEGER); 
VAR I:INTEGER; 
BEGIN 
OUTBYT(PORTADDR,10H); 
FOR I:=1 TO 5 DO; 
  INBYT(PORTADDR,FORCECHAR); 
OUTBYT(PORTADDR,00); 
FORCEINT:=ORD(FORCECHAR); 
[* FORCEINT : 0 <---PUSH- 127 ET 128 --PULL---> 255 *] 
  FORCE:=127-FORCEINT; 
[* FORCE : 127 <---PUSH-- 0 ET 1 -PULL---> -128 *] 
END; 

PROCEDURE CALCULATION(FORCE:INTEGER;VAR COMOT,COMOTINT:INTEGER); 
BEGIN 
COMOT:=(K*FORCE+CONSTSTI*(S0+FORCE)+CONSTDTS*(FORCE-F0)) DIV 1000; 
S0:=S0+FORCE;F0:=FORCE; 
[* COMOT : 127 <---PUSH- 0 ET -1 --PULL---> -128 *] 
IF COMOT<128 THEN COMOT:=-128; 
IF COMOT>127 THEN COMOT:= 127; 
IF COMOT<0 THENCOMOTINT:=COMOT+256 
ELSE COMOTINT: = COMOT; 
[* COMOTINT : 127 <---PUSH- 0 ET 255 --PULL---> 128 *] 
END; 

PROCEDURE DEBUT; 
BEGIN 
DISABLEINTERRUPTS; 
[* INITIALISATION DU HARDWARE : PPI , PIT ET PIC *] 
  [*PROCEDURE INITPI; FILE INIT1.PAS *] 
  [*PROCEDURE INITPIC; FILE INIT1.PAS *] 
  [*CONTROLE MODE WORD FOR TIMER0 , P.I.T *] 
INITPIC; 
INITPI; 

[* CHOIX DES PARAMETRES *] 
CHOICEPARAMS(K,TS,TT,TD,STARTPOSITION,MODE1,MODE2); 
CONSTSTI:=1000*(TS DIV (2*TT)); 
CONSTDTS:=1000*(TD DIV TS); 
K:=2*K; 
[* INITIALISATION DU COMPTEUR *] 
COUNT,FULLWORD:=TSREAL; 
[*P.I.T SAMPLE TIME *] 
OUTBYT(PITCONTRADDR,TIMER0CONTRWORD); 
[*P.I.T CONTROL MODE *] 
[* INITIALISATION DES VARIABLES : F0,S0 *] 
FOR I:=0 TO S:=0; 
IJ:=1;
PRINTASCII(' POINT N');
CONVERSION3(IJ,STR3);
PRINTASCII(' = ');
CONVERSION3((POINTSCIJ)*146) DIV 10000,STR3);CO(' .');
CONVERTDECIM2((ABS(POINTSCIJ)*146) MOD 10000) DIV 100 ,STR2);
PRINTASCII(' MM. CONTIN');
PRINTASCII('UE......(Y/N)');
OKRECP0S := CI;
WHILE (ORD(OKRECP0S) <> ORD('Y')) AND
(ORD(OKRECP0S) <> ORD('N'))
DO BEGIN CO(CHR(07H));OKRECP0S := CI;END;
CO(' ');CO(OKRECP0S);LIN;
IF ORD(OKRECP0S) = ORD('N') THEN FinERegistrement;
IJ := IJ + 1;ENREG := FALSE;
END;
END;

END;

END;

BEGIN
DEBU}
CO 'MZEROINIT':=20H;
INPOSITION;

[* OK DEBUT TEACHING *]
LIN:
PRINTASCII13('START TEACH ');
PRINTASCII13(' OPERATIONS ');
PRINTASCII13('.......(Y/N) ');
REPEAT
OKTEACH:=CI
UNTIL ORD(OKTEACH)=ORD('Y');
CO(' ');CO(OKTEACH);LIN;LIN;
TAB;PRINTASCII13('******** MODE ');
PRINTASCII13('TEACH POINT T ');
PRINTASCII13('O POINT IS RU ');
PRINTASCII13('NNING ********');LIN;LIN;TAB;
PRINTASCII13('YOU CAN MOVE ');
PRINTASCII13('THE ARM TO TH ');
PRINTASCII13('E DESIRED POI ');
PRINTASCII13('MENTS,PUSH ');LIN;TAB;
PRINTASCII13('THE "RECORD " ');
PRINTASCII13('BUTTON TO STO ');
PRINTASCII13('RE THE COORDI ');
PRINTASCII13('NATE. ' );LIN;LIN;
OUTBYTE(IMASKREGPORT,09H); [* INTERRUPTS 3 ET 0 SONT MASQUEES*]

[*PROCEDURE INITSAMPLETIME, FILE INIT1.PAS *]
INITSAMPLETIME;
WHILE TRUE DO
BEGIN
 IF SWITCHE THEN
 BEGIN
 OUTBYTE(PORTAADDR,COMMZEROINIT);
 PRINTASCII13('YOU HAVE REA ');
 PRINTASCII13('HED A END SWI ');
 PRINTASCII13('TCH. ');LIN;
 PRINTASCII13('END SWITCH : ');
 CONVERTION(ENPOSITION*146) DIV 10000,STR;CO(' ',');
 CONVERTDECIM2(((ABS(ENPOSITION)*146) MOD 10000) DIV 100,STR2);
 PRINTASCII14(' MM ');TAB;
 CONVERTION(EN)POSITION,STR);
 PRINTASCII13(' UNITS ');
 INITPOSITION;
 SWITCHE:=FALSE;
 [* INTERRUPTS 3 ET 0 SONT MASQUEES*]
 OUTBYTE(IMASKREGPORT,09H);
 INITSAMPLETIME;
 PRINTASCII13('MODE TEACH PO ');
 PRINTASCII13('INT TO POINT ');
 PRINTASCII13('IS RUNNING. ' );LIN;LIN;
END;
 IF TIME1 THEN
 BEGIN
 INITSAMPLETIME;
 TIME1:=FALSE;
 INPUTFORCE(FORCECHAR,FORCEINT,FORCE);
 CALCULATION(FORECUMOT,CUMUTINT);
 OUTBYTE(PORTAADDR,CHR(COMUTINT));
 IF ENREG=TRUE THEN BEGIN
 OUTBYTE(PORTAADDR,00H);

PROCEDURE INITFUNC;
PROCEDURE INITSAMPLETIME;
PUBLIC NODUCONVER;
PROCEDURE CONVERSION(NUMBER:INTEGER; VAR STR:STRING6);
PROCEDURE CONVERSION3(NUMBER:INTEGER; VAR STR3:STRING3);
PROCEDURE CONVERTDECIM2(NUMBER:INTEGER; VAR STR2:STRING2);
PROCEDURE LIN;
BEGIN
CO(CR); CO(LF);
END;

PROCEDURE FINENREGISTREMENT;
VAR Z: INTEGER;
BEGIN
LIN;
Z:=1;
PRINTASCII13('RECORDED POSI');
PRINTASCII13('TIONS');
LIN;
REPEAT
TAB: PRINTASCII13('------------> ');
CONVERSION3((POINTSZ) * 146) DIV 10000 + STR3; CO(' ', ' ');
CONVERDEICIM2((ABS(POINTSZ) * 146) MOD 10000) DIV 100 + STR2;
PRINTASCII13(' MM '); LIN;
Z:=Z + 1;
UNTIL Z=IJ+1;
LIN; PRINTASCII13('TEACH MODE FI');
PRINTASCII13('NISHED. DO-YO');
PRINTASCII13('U WANT TO CON');
PRINTASCII13('TINUE ...(Y/N)');
OKEND:=CI;
WHILE (ORD(OKEND) < ORD('Y')) OR (ORD(OKEND) < ORD('N')) DO:
BEGIN
CO(CHR(07H)); OKEND:=CI; END;
CO(' '); CO(OKEND); LIN;
IF OKEND=OKEND AND ORD('Y') THEN:
BEGIN LIN; LIN; DEBUT; END;
END;

PROCEDURE TAB;
BEGIN
PRINTASCII13(' '); LIN;

PROCEDURE CHICEFARMS(VAR K, TS, TI, TD, STARTPOSITION, INTEGER; MODE, INTEGER; MODE2: BOOLEAN);
BEGIN
[-PARAMETERS-]
LIN; TAB;
PRINTASCII13('******'); PRINTASCII13('******');
PRINTASCII13('******'); PRINTASCII13('******');
PRINTASCII13('* TEACH P'); PRINTASCII13('POINT TO POINT');
PRINTASCII13(' WITH A FORCE'); PRINTASCII13(' SENSOR *');
LIN; TAB;
PRINTASCII13('******'); PRINTASCII13('******');
PRINTASCII13('******'); PRINTASCII13('******');
LIN; TAB;
PRINTASCII13('******');
PRINTASCII13('CHOICE OF '); PRINTASCII13('PARAMETERS');
PRINTASCII13(' ---------- '); LIN;
REPEAT
LIN; TAB;
PRINTASCII13('TIMING VALUE '); PRINTASCII13('MICROSEC');
TSREAL:=(Lecture * 1229) DIV 1000;
LIN; LIN;
TAB; PRINTASCII13('---------- '); PRINTASCII13('CHOICE P, I, D');
PRINTASCII13(' PARAMETERS'); PRINTASCII13(' ---------- '); LIN; LIN;
TAB;PRINTASCII13('PRINTASCII13('PROPORT. K : '))TAB;PRINTASCII13('SAMPLE TIME: ')T$=LECTURE;LIN;
TAB;PRINTASCII13('INTEGRAT. TI: ')T$=LECTURE;LIN;
TAB;PRINTASCII13('DERIVAT. TD: ')TD=LECTURE;LIN;
LIN;TAB;PRINTASCII13('PARAMETERS (OK)');
PRINTASCII13(' .........(Y/N)');
OKPARAMETERS:=CI;
WHILE (ORD(OKPARAMETERS)<ORD('Y')) AND (ORD(OKPARAMETERS)<ORD('N'))
DO BEGIN CO(CHR(07H));OKPARAMETERS:=CI;END;
CU(OKPARAMETERS):=LIN;

[---------------------------------- MODE CHOICE ---------------------------------]

[REPEAT
MODE1:=FALSE;MODE2:=FALSE;
LIN;TAB;
PRINTASCII13('----------- ');
PRINTASCII13('CHOICE RECORD');
PRINTASCII13(' MODE ');
PRINTASCII13(' -------- ');
LIN;TAB;
PRINTASCII13('MODE 1: PUSH');
PRINTASCII13(' THE BUTTON ');
LIN;TAB;
PRINTASCII13('MODE 2: RECO');
PRINTASCII13('RD ALL POSIT');
PRINTASCII13('ONS ');
LIN;TAB;
PRINTASCII13('CHOICE (1/2');
PRINTASCII13('? ');
REPEAT
NUMEROMODE:=CI
UNTIL (ORD(NUMEROMODE)=ORD('1')) OR (ORD(NUMEROMODE)=ORD('2'));
CU(NUMEROMODE):=LIN;
IF NUMEROMODE='1' THEN MODE1:=TRUE
ELSE MODE2:=TRUE;
LIN;TAB;PRINTASCII13('MODE OK (Y/N)');
PRINTASCII13('.........? ');
REPEAT
OKMODE:=CI
UNTIL (ORD(OKMODE)=ORD('Y')) OR (ORD(OKMODE)=ORD('N'));
CU(OKMODE):=LIN;
UNTIL (ORD(OKMODE)=ORD('Y'));

[-------------------------------- STARTPOSITION -------------------------------]

[REPEAT
LIN;TAB;
PRINTASCII13('-------- ');
PRINTASCII13('CHOICE START');
PRINTASCII13(' POSITION ');
PRINTASCII13(' ----------- ');
LIN;TAB;
PRINTASCII13('STARTPOSITION');
PRINTASCII13(' (1 UNIT=0,01')
PRINTASCII13('MM') PRINTASCII13('STARTPOSITION:=LECTURE;LIN;
LIN;TAB;PRINTASCII13('STARTPOSITION '));
PRINTASCII13('OK (Y/N)....? ');
REPEAT
OKSTARTPOSITION:=CI
UNTIL (ORD(OKSTARTPOSITION)=ORD('Y')) OR (ORD(OKSTARTPOSITION)=ORD('N'));
CU(OKSTARTPOSITION):=LIN;
UNTIL (ORD(OKSTARTPOSITION)=ORD('Y'));*
STARTPOSITION:=0;
MODE1:=FALSE;MODE2:=FALSE;

[END]
MODULE MODURECHERCHEZERO;

*INCLUDE(HEADIN.PAS)

PRIVATE MODURECHERCHEZERO;

[XXXXXXXXXXXXXXXXXXXXXXXXX]
PROCEDURE INITPOSITION;

BEGIN [* INITPOSITION *]

POSITION:=3807;

INTERUPTS POUR INITPOSITION [*]

INTERUPT(INTERZERONINIT=0,COUNTUP=1,COUNTDO=2,SWITCHEZINIT=4)

INTERUPT(SWITCHEZINIT=5,INTEREXTERNE=7)

SETINTERUPT(0,INTERZERONINIT);

SETINTERUPT(1,COUNTUP);

SETINTERUPT(2,COUNTDO);

SETINTERUPT(4,SWITCHEZINIT);

SETINTERUPT(5,SWITCHEZINIT);

SETINTERUPT(7,INTEREXTERNE);

OUTBT(1,MASKREGPORT,04OH); [* MASK INIT1,2,3,6 **]

ENABLEINTERUPT;

ZERO:=TRUE;

REPEAT OUTBT(PORTAADDR,COMMZERONINIT)
UNTIL ZERO=FALSE;

OUTBT(PORTAADDR,00H);[* VITESSE NULLE *]

LIN;

PRINTASCII13('ZERO POSITION');

PRINTASCII13(' OF THE ARM');

LIN;

[INTERUPTIONS POUR TEACH-IN OPERATION *]

INTERUPT(ENREGISTEMENT=15,INTERTIME1=6,II_SWITCHE=12,II_SWITCHE=13)

INTERUPT(7,ENREGISTEMENT);

[RECORD CONNECTEE INT EXTERNE *]

INTERUPT(6,INTERTIME1);

[INTERUPT SAMPLETIME *

INTERUPT(4,II_SWITCHE);

[ALARME *

INTERUPT(5,II_SWITCHE);

[ALARME *

END; [* INITPOSITION *]
; CE NO-MAIN MODULE EST APPELE PAR LE MAIN MODULE MAIN1 (FILE MAIN2.PAS)

NAME MODUSUBASSEMBLEUR
EXTRN SWITCHE:BYTE
EXTRN ENREG:BYTE
EXTRN TIME1:BYTE
EXTRN COMMZEROINIT:WORD
EXTRN POSITION:WORD
EXTRN ENDPOSITION:WORD
EXTRN ZERO:BYTE
ASSUME CS:CODE
CODE SEGMENT PUBLIC
;==================================================
ASSUME DS:SEG ZERO
ASSUME SS:SEG ZERO
PUBLIC INTERRUPT PROC NEAR
PUSH AX
  MOV ZERO,0H ; ZERO :=FALSE
  MOV AX,48H ; DEMASK INT1,2 :
  OUT 0C2H,AX ; COUNTUP , COUNTDOWN :
POP AX
IRET
INTZEROINIT ENDP
;==================================================
ASSUME DS:SEG COMMZEROINIT
ASSUME SS:SEG COMMZEROINIT
PUBLIC SWITCHE1INIT ; INTERRUPT FIN DE COURSE DE GAUCHE
SWITCHE1INIT PROC NEAR ; INT4
PUSH AX
  MOV COMMZEROINIT,0EBH ; PULL
POP AX
IRET
SWITCHE1INIT ENDP
;==================================================
PUBLIC SWITCHE2INIT ; INTERRUPT FIN DE COURSE DE DROITE
SWITCHE2INIT PROC NEAR ; INT5
PUSH AX
  MOV COMMZEROINIT,017H ; PUSH
POP AX
IRET
SWITCHE2INIT ENDP
;==================================================
PUBLIC IISWITCHE ; INTERRUPT FIN DE COURSE DE GAUCHE
IISWITCHE PROC NEAR ; INT4.POUR TEACH-IN OPERATIONS.
PUSH AX
ASSUME DS:SEG POSITION
ASSUME SS:SEG POSITION
MOV AX,POSITION
ASSUME DS:SEG ENDPOSITION
ASSUME SS:SEG ENDPOSITION
MOV ENDPOSITION,AX
ASSUME DS:SEG COMMZEROINIT
ASSUME SS:SEG COMMZEROINIT
  MOV COMMZEROINIT,0F6H ; PULL
ASSUME DS:SEG SWITCHE
ASSUME SS:SEG SWITCHE
  MOV SWITCHE,01H
POP AX
IRET
IISWITCHE ENDP
;==================================================
PUBLIC COUNTUP
COUNTUP PROC NEAR
INC POSITION
IRET
COUNTUP ENDP

PUBLIC COUNTDO
COUNTDO PROC NEAR
DEC POSITION
IRET
COUNTDO ENDP

PUBLIC INTERTIME1
INTERTIME1 PROC NEAR
MOV TIME1,01H
IRET
INTERTIME1 ENDP

PUBLIC ENREGISTREMENT
ENREGISTREMENT PROC NEAR
MOV ENREG,01H
IRET
ENREGISTREMENT ENDP

CODE ENDS
SERIES-III 8086 LOCATER, V2.0

INPUT FILE: TEACH.LNK
OUTPUT FILE: TEACH

CONTROLS SPECIFIED IN INVOCATION COMMAND:
TO TEACH

DATE: 11/10/84  TIME:

SYMBOL TABLE OF MODULE MODUMAIN1

<table>
<thead>
<tr>
<th>BASE</th>
<th>OFFSET</th>
<th>TYPE</th>
<th>SYMBOL</th>
<th>BASE</th>
<th>OFFSET</th>
<th>TYPE</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0021H</td>
<td>0533H</td>
<td>PUB</td>
<td>CHICEPARAMS</td>
<td>0021H</td>
<td>0916H</td>
<td>PUB</td>
<td>CI</td>
</tr>
<tr>
<td>0021H</td>
<td>0948H</td>
<td>PUB</td>
<td>CO</td>
<td>0021H</td>
<td>0870H</td>
<td>PUB</td>
<td>CONVERSION</td>
</tr>
<tr>
<td>0021H</td>
<td>086AH</td>
<td>PUB</td>
<td>DEBUT</td>
<td>0021H</td>
<td>0420H</td>
<td>PUB</td>
<td>FINENREGISTRY</td>
</tr>
<tr>
<td>0021H</td>
<td>085CH</td>
<td>PUB</td>
<td>INITPIC</td>
<td>0021H</td>
<td>0740H</td>
<td>PUB</td>
<td>INITPOSITION</td>
</tr>
<tr>
<td>0021H</td>
<td>0850H</td>
<td>PUB</td>
<td>INITPP</td>
<td>0021H</td>
<td>0836H</td>
<td>PUB</td>
<td>INITSAMPLE</td>
</tr>
<tr>
<td>0021H</td>
<td>0419H</td>
<td>PUB</td>
<td>LIN</td>
<td>0021H</td>
<td>02CEH</td>
<td>PUB</td>
<td>MODUMAIN1</td>
</tr>
<tr>
<td>0021H</td>
<td>02E6H</td>
<td>PUB</td>
<td>PRINTASCII13</td>
<td>0021H</td>
<td>0300H</td>
<td>PUB</td>
<td>PRINTASCII4</td>
</tr>
<tr>
<td>0021H</td>
<td>0517H</td>
<td>PUB</td>
<td>TAB</td>
<td>0021H</td>
<td>021H</td>
<td>PUB</td>
<td>COMMZEROINIT</td>
</tr>
<tr>
<td>00E4H</td>
<td>0025H</td>
<td>PUB</td>
<td>ENDPOSITION</td>
<td>00E4H</td>
<td>0101H</td>
<td>PUB</td>
<td>ENREG</td>
</tr>
<tr>
<td>00E4H</td>
<td>001FH</td>
<td>PUB</td>
<td>MODE1</td>
<td>00E4H</td>
<td>027F1H</td>
<td>PUB</td>
<td>MODE2</td>
</tr>
<tr>
<td>00E4H</td>
<td>0104H</td>
<td>PUB</td>
<td>POINTS</td>
<td>00E4H</td>
<td>0209H</td>
<td>PUB</td>
<td>STR</td>
</tr>
<tr>
<td>00E4H</td>
<td>0023H</td>
<td>PUB</td>
<td>STARTPOSITION</td>
<td>00E4H</td>
<td>0029H</td>
<td>PUB</td>
<td>TD</td>
</tr>
<tr>
<td>00E4H</td>
<td>0102H</td>
<td>PUB</td>
<td>SWITCHE</td>
<td>00E4H</td>
<td>0103H</td>
<td>PUB</td>
<td>TIME1</td>
</tr>
<tr>
<td>00E4H</td>
<td>0020H</td>
<td>PUB</td>
<td>TI</td>
<td>00E4H</td>
<td>008DH</td>
<td>PUB</td>
<td>ZERO</td>
</tr>
<tr>
<td>0124H</td>
<td>0054H</td>
<td>PUB</td>
<td>COUNTDO</td>
<td>0124H</td>
<td>0040H</td>
<td>PUB</td>
<td>COUNTUP</td>
</tr>
<tr>
<td>0124H</td>
<td>0061H</td>
<td>PUB</td>
<td>ENREGISTRY</td>
<td>0124H</td>
<td>007FH</td>
<td>PUB</td>
<td>INTEREXTERNAL</td>
</tr>
<tr>
<td>0124H</td>
<td>0033H</td>
<td>PUB</td>
<td>I2SWITCH</td>
<td>0124H</td>
<td>0047H</td>
<td>PUB</td>
<td>INTEREXTERNAL</td>
</tr>
<tr>
<td>0124H</td>
<td>0058H</td>
<td>PUB</td>
<td>INTERTIME1</td>
<td>0124H</td>
<td>0000H</td>
<td>PUB</td>
<td>INTZEROINIT</td>
</tr>
<tr>
<td>0124H</td>
<td>000DH</td>
<td>PUB</td>
<td>SWITCHELINIT</td>
<td>0124H</td>
<td>0016H</td>
<td>PUB</td>
<td>INITZSWITCH</td>
</tr>
<tr>
<td>00BAH</td>
<td>01C0H</td>
<td>PUB</td>
<td>PQCALL&lt;DEVICESDRIVE-VER</td>
<td>00BAH</td>
<td>0169H</td>
<td>PUB</td>
<td>POINTERALWAYS UNTIL-OFFER</td>
</tr>
<tr>
<td>00BAH</td>
<td>0080H</td>
<td>PUB</td>
<td>PQOUTSCLOSEDOW</td>
<td>00BAH</td>
<td>0060H</td>
<td>PUB</td>
<td>PQOUTSTARTUP</td>
</tr>
<tr>
<td>00BAH</td>
<td>0160H</td>
<td>PUB</td>
<td>PQ&lt;FAK&lt;INTERNAL&lt;WRITE&lt;BUFFER</td>
<td>00BAH</td>
<td>0100H</td>
<td>PUB</td>
<td>PQ&lt;GET&lt;LOCAL EN</td>
</tr>
<tr>
<td>00BAH</td>
<td>0120H</td>
<td>PUB</td>
<td>PQ&lt;ISCOSC</td>
<td>00BAH</td>
<td>0110H</td>
<td>PUB</td>
<td>PQ&lt;STORE&lt;LOCAL-OFFER</td>
</tr>
<tr>
<td>00BAH</td>
<td>0220H</td>
<td>PUB</td>
<td>TUALLOCATE</td>
<td>00BAH</td>
<td>0210H</td>
<td>PUB</td>
<td>TQEXIT</td>
</tr>
<tr>
<td>00BAH</td>
<td>0200H</td>
<td>PUB</td>
<td>TOGETHER</td>
<td>00BAH</td>
<td>0100H</td>
<td>PUB</td>
<td>TQINITIALIZATION</td>
</tr>
<tr>
<td>00BAH</td>
<td>0260H</td>
<td>PUB</td>
<td>TUPARSECCL</td>
<td>00BAH</td>
<td>0000H</td>
<td>PUB</td>
<td>TQ&lt;001</td>
</tr>
<tr>
<td>00BAH</td>
<td>0140H</td>
<td>PUB</td>
<td>TQ&lt;302</td>
<td>00BAH</td>
<td>020H</td>
<td>PUB</td>
<td>TQ&lt;999</td>
</tr>
<tr>
<td>00BAH</td>
<td>0040H</td>
<td>PUB</td>
<td>TQ&lt;COPYRIGHT</td>
<td>00F5H</td>
<td>0000H</td>
<td>PUB</td>
<td>TQ&lt;LOCAL</td>
</tr>
<tr>
<td>00F5H</td>
<td>0004H</td>
<td>PUB</td>
<td>TQACCESS</td>
<td>00F5H</td>
<td>002H</td>
<td>PUB</td>
<td>TQCURRENTPTD</td>
</tr>
<tr>
<td>00F5H</td>
<td>0005H</td>
<td>PUB</td>
<td>TDATAAREAFREE</td>
<td>00BAH</td>
<td>0290H</td>
<td>PUB</td>
<td>TODEFAULTPL</td>
</tr>
<tr>
<td>00E4H</td>
<td>0000H</td>
<td>PUB</td>
<td>INITFP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEMORY MAP OF MODULE MODUMAIN1

MODULE START ADDRESS PARAGRAPH = 0020H  OFFSET = 0006H
SEGMENT MAP

<table>
<thead>
<tr>
<th>START</th>
<th>STOP</th>
<th>LENGTH</th>
<th>ALIGN</th>
<th>NAME</th>
<th>CLASS</th>
<th>OVERLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020H</td>
<td>00218H</td>
<td>0019H</td>
<td>G</td>
<td>??LOC86&lt;INITCO</td>
<td>CODE</td>
<td>-DE</td>
</tr>
</tbody>
</table>

-VER
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>GROUP OR SEGMENT NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>00210H</td>
<td>GROUP</td>
</tr>
<tr>
<td></td>
<td>CODE</td>
</tr>
<tr>
<td>00E40H</td>
<td>GROUP</td>
</tr>
<tr>
<td></td>
<td>DATA</td>
</tr>
<tr>
<td></td>
<td>STACK</td>
</tr>
<tr>
<td></td>
<td>CONST</td>
</tr>
<tr>
<td></td>
<td>MEMORY</td>
</tr>
<tr>
<td>00BA0H</td>
<td>GROUP</td>
</tr>
<tr>
<td></td>
<td>UTSCODE</td>
</tr>
</tbody>
</table>
LS 503, LS 503D, LS 513, LS 513D
Gekapseltes, inkrementales Längenmesssystem
Système de mesure linéaire incrémental fermé
Sealed, incremental linear transducer
### 4.2. LS 513, LS 513D

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning principle</td>
<td>photoelectric, transmitted light</td>
</tr>
<tr>
<td>Scale</td>
<td>glass scale with DIADUR graduation</td>
</tr>
<tr>
<td>Grating pitch</td>
<td>LS 513: 40 μm&lt;br&gt;LS 513D: 20 μm</td>
</tr>
<tr>
<td>Light source</td>
<td>long-life miniature filament lamp, prefocused, 5 V/0.6 W</td>
</tr>
<tr>
<td>Scanning elements</td>
<td>silicon solar cells in push-pull arrangement</td>
</tr>
<tr>
<td>Pulse shaping electronics</td>
<td>within mounting block of scanning head. Integra. Schmitt-triggers and output</td>
</tr>
<tr>
<td>System accuracy</td>
<td>± 10 μm/m or ± 5 μm/m measuring length</td>
</tr>
<tr>
<td>Reference mark</td>
<td>on standard units at mid-point of measuring length; optionally: spaced every 50 mm to the left and right from mid-point of measuring length (special version)</td>
</tr>
</tbody>
</table>

| Max. permissible traversing speed in m/min    | LS 513: 60<br>LS 513D: 30                                                   |
| Permissible acceleration when traversing       | 30 m/s²                                                                   |
| Reqd. feed power                               | ≤ 10 N                                                                    |
| Sealed protection of transducer                | IP 53 (DIN 40 050) mounted as instructed                                   |
| Permissible ambient temperature                | 0°C ... + 50°C (32°F ... + 122°F)                                          |

**Storage and transport conditions:**

| Permissible temperature                        | -30°C ... + 80°C (-22°F ... + 176°F)                                       |
| Permissible relative humidity                  | 20% ... 80%                                                               |
| Permissible acceleration                        | 60 m/s²                                                                  |
| Permissible shock load                         | 200 m/s², pulse duration 5 ms                                            |

**Output signals of transducer signals trains:**

- With positive measuring direction:  
  - \( U_a \) "High": \( U_a = 2.4 \text{ V with } l_{\text{load}} = 4 \text{ mA} \)
  - \( U_a \) "Low": \( U_a = 0.4 \text{ V with } l_{\text{sink}} = 4 \text{ mA} \)
  - Switching time \( t \leq 0.5 \mu s \)
  - Lag of \( U_{a0} \) signal to signals \( U_{a1} \) and \( U_{a2} \):
    \( t \leq 1.2 \mu s \)

* inverted signals not shown

**ON-to-OFF ratio**

- up to 10 kHz: \( I_{x} = \pm 25° \)
- up to 30 kHz: \( I_{x} = \pm 45° \)

**Phase angle**

- up to 40 kHz: \( 90° \pm 15° \)
- up to 30 kHz: \( 90° \pm 25° \)

**Supply voltage**

- \( U_p = 5 \text{ V} \pm 5% \)
- \( I = 250 \text{ mA with load } l_{\text{source}} = 4 \text{ mA} \)

**Phase delay of electronics**

- \( t \leq 4 \mu s \)

**Output load capacity**

- \( l_{\text{source}} \leq 6 \text{ mA} \)
- \( l_{\text{sink}} \leq 20 \text{ mA} \)
- \( C_{\text{load}} \leq 1000 \text{ pF at cable end} \)

**Weight of transducer**

- 0.8 kg + 2 kg/m measuring length

**Connecting cable of transducer**

- 0 m

**Extension cable**

- max. 50 m with differential line receiver at input of subsequent electronics and supply voltage remaining constant. When using HEIDENHAIN counters VRZ 116, 126, 136 the voltage drop in the supply lead (extension cable 69.2-3-006) is regulated for a length not exceeding 50 m.
PRINT 2

DAC 1:
AANSTURING VAN MOTOR
A1 = GROUND
A2 = +15 V
A3 = -15 V
A4 = U_{out}
DAC-08

MULTIPLYING D/A CONVERTER

(UNIVERSAL DIGITAL LOGIC INTERFACE)

FEATURES
- Fast Settling Output Current ..................... 65ns
- Full-Scale Current Prescaled to ±1 LSB
- Direct Interface to TTL, CMOS, ECL, HTL, PMOS
- Nonlinearity to .01% Maximum Over Temperature Range
- High Output Impedance and
  Compliance ........................................... -10V to +18V
- Complementary Current Outputs
- Wide Range Multiplying Capability ........ 1MHz Bandwidth
- Low FS Current Drift .............................. ±10ppm/°C
- Wide Power Supply Range ........................ ±4.5V to ±18V
- Low Power Consumption .......................... 33mW @ ±5V
- Low Cost

GENERAL DESCRIPTION
The DAC-08 series of 8-bit monolithic digital-to-analog converters provide very high-speed performance coupled with low cost and outstanding applications flexibility.

Advanced circuit design achieves 85ns settling times with very low "glitch" energy at low power consumption. Monotonic multiplying performance is attained over a wide 20 to 1 reference current range. Matching to within 1 LSB between reference and full-scale currents eliminates the need for full-scale trimming in most applications. Direct interface to all popular logic families with full noise immunity is provided by the high swing, adjustable threshold input.

High voltage compliance complementary current outputs provide, increasing versatility and enabling different operation to effectively double the peak-to-peak analog swing. In many applications, the outputs can be directly converted to voltage without the need for an external opamp.

All DAC-08 series models guarantee full 8-bit monotonicity and nonlinearity as tight as ±0.1% over the entire operational temperature range are available. Device performance remains essentially unchanged over the ±4.5 to ±18V power supply range, with 33mW power consumption attainable at ±5V supplies.

The compact size and low power consumption make DAC-08 attractive for portable and military/aerospace applications; devices processed to MIL-STD-883, Level 8 are available.

DAC-08 applications include 8-bit, 1μs A/D converters, motor and pen drivers, waveform generators, audio encoders and attenuators, analog meter drivers, programmable power supplies, CRT display drivers, high-speed modems and data applications where low cost, high speed and complete output versatility are required.

EQUIVALENT CIRCUIT
### Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>DAC-08A/H</th>
<th>DAC-08C/E</th>
<th>DAC-08C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>R_0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Linearity</td>
<td>R_y</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Temperature</td>
<td>R_x</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Propagation Delay</td>
<td>I_PN</td>
<td>- To ±1/2 LSB, all bits switched</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>V_x</td>
<td>+125°C</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Logic Inputs</td>
<td>V_y</td>
<td>- V+ to V_+ plus 36V</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>P_D</td>
<td>- 500mW</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Note</td>
<td>-</td>
<td>NOTE: Absolute ratings apply to both DICE and packaged parts, unless otherwise noted.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Electrical Characteristics

**Logic Inputs**
- V_y to V_+ plus 36V
- V_+ to V_-

**V_L**
- DAC-08A: -10mA to +10mA
- DAC-08C: -25mA to +25mA

**Reference Input (V_y to V_+)**
- DAC-08A: +2.0mA
- DAC-08C: +5.0mA

**Reference Input Differential Voltage**
- DAC-08A: +2.0mA
- DAC-08C: +5.0mA

**Output Current Range**
- DAC-08A: +10mA, V_+ to -10mA
- DAC-08C: +25mA, V_+ to -12mA

**Output Current Noise**
- DAC-08A: +25µA
- DAC-08C: +25µA

**Logic Input Levels**
- Logic "0" V_y = 0V
- Logic "1" V_y = V_+ 0V

**V_TY**
- DAC-08A: +13.5V
- DAC-08C: +15V

**Reference Bias Current**
- DAC-08A: +1µA
- DAC-08C: +1µA

**Reference Input Slew Rate**
- DAC-08A: 0.0002 µA
- DAC-08C: 0.0002 µA

**Power Supply Sensitivity**
- DAC-08A: 5.0µA
- DAC-08C: 5.0µA
**ELECTRICAL CHARACTERISTICS** at $V_G = \pm 15V$, $I_{REF} = 2.0 mA$, $-55{}^\circ C \leq T_A \leq 125{}^\circ C$ for DAC-08/08A, $0{}^\circ C \leq T_A \leq +70{}^\circ C$ for DAC-08C, E & H, unless otherwise noted. Output characteristics refer to both $I_{OUT}$ and $I_{OUT}$ (Continued)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SYMBOL</th>
<th>CONDITIONS</th>
<th>DAC-08A/H</th>
<th>DAC-08/E</th>
<th>DAC-08C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MIN</td>
<td>TYP</td>
<td>MAX</td>
</tr>
<tr>
<td>Power Supply Current</td>
<td>$V_G = \pm 2V$, $I_{REF} = 1.0mA$</td>
<td>-2.3</td>
<td>3.5</td>
<td>-2.3</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>$V_G = +5V$, $-10V$, $I_{REF} = 2.0mA$</td>
<td>-4.2</td>
<td>-6.5</td>
<td>-4.2</td>
<td>-6.5</td>
</tr>
<tr>
<td></td>
<td>$V_G = \pm 15V$, $I_{REF} = 2.0mA$</td>
<td>2.4</td>
<td>3.8</td>
<td>2.4</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td>$V_G = \pm 15V$, $I_{REF} = 2.0mA$</td>
<td>-8.4</td>
<td>-7.6</td>
<td>-8.4</td>
<td>-7.6</td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>$P_d$</td>
<td>$2.8V$, $I_{REF} = 1.0mA$</td>
<td>33</td>
<td>48</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>$V_G = +5V$, $-10V$, $I_{REF} = 2.0mA$</td>
<td>103</td>
<td>136</td>
<td>103</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>$V_G = \pm 15V$, $I_{REF} = 2.0mA$</td>
<td>136</td>
<td>174</td>
<td>136</td>
<td>174</td>
</tr>
</tbody>
</table>

**NOTE:** Guaranteed by design.

**ORDERING INFORMATION**

<table>
<thead>
<tr>
<th>16-PIN DUAL-IN-LINE PACKAGE</th>
<th>OPERATING TEMPERATURE RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>HERMETIC</td>
</tr>
<tr>
<td>0.1%</td>
<td>DAC080AQ</td>
</tr>
<tr>
<td>0.1%</td>
<td>DAC080Q</td>
</tr>
<tr>
<td>0.3%</td>
<td>DAC080CQ</td>
</tr>
</tbody>
</table>

*Also available with MIL-STD-883B processing. To order add "883" as a suffix to the part number. See Section 3 for screening procedure.

†All commercial and industrial temperature range parts are available with burn-in per MIL-STD-883. See Ordering Information, Section 2.

**PIN CONNECTIONS**

![16-PIN DUAL-IN-LINE PACKAGE](image)
AD570

FEATURES
Complete A/D Converter with Reference and Clock
Fast Successive Approximation Conversion — 25μs
No Missing Codes Over Temperature
0 to +70°C — AD570J
-65°C to +125°C — AD570S
Digital Multiplexing — 3 State Outputs
18 Pin Ceramic DIP
Low Cost Monolithic Construction

PRODUCT DESCRIPTION
The AD570 is an 8-bit successive approximation A/D converter consisting of a DAC, voltage reference, clock, comparator, successive approximation register and output buffers — all fabricated on a single chip. No external components are required to perform a full 8-bit conversion in 25μs.

The AD570 incorporates the most advanced integrated circuit design and processing technology available today. I²L (integrated injection logic) processing in the fabrication of the SAR function along with laser trimming of the high stability SiCr thin film resistor ladder network at the wafer stage (LWT) and a temperature compensated, subsurface Zener reference insures full 8-bit accuracy at low cost.

Operating on supplies of +5V and -15V, the AD570 will accept analog inputs of 0 to +10V unipolar or ±5V bipolar, externally selectable. As the BLANK and CONVERT input is driven low, the three state outputs will be open and a conversion will commence. Upon completion of the conversion, the DATA READY line will go low and the data will appear at the output. Pulling the BLANK and CONVERT input high blanks the outputs and reads the device for the next conversion. The AD570 executes a true 8-bit conversion with no missing codes in approximately 25μs.

The AD570 is available in two versions; the AD570J is specified for the 0 to 70°C temperature range, the AD570S for -55°C to +125°C. Both guarantee full 8-bit accuracy and no missing codes over their respective temperature ranges and are packaged in 18-pin hermatically-sealed ceramic DIP’s.

PRODUCT HIGHLIGHTS
1. The AD570 is a complete 8-bit A/D converter. No external components are required to perform a conversion. Full scale calibration accuracy of ±0.8% (2LSB of 8 bits) is achieved without external trims.
2. The AD570 is a single chip device employing the most advanced IC processing techniques. Thus, the user has at his disposal a truly precision component with the reliability and low cost inherent in monolithic construction.
3. The AD570 accepts either unipolar (0 to +10V) or bipolar (-5V to +5V) analog inputs by simply grounding or opening a single pin.
4. The device offers true 8-bit accuracy and exhibits no missing codes over its entire operating temperature range.
5. Operation is guaranteed with -15V and +5V supplies. The device will also operate with a -12V supply.
6. The AD570S is also available with full processing to MIL-STD-883B, Class B. The single chip construction and functional completeness make the AD570 especially attractive for high reliability applications.
7. Every AD570 is subjected to long-term stabilization bakes, given a powered burn-in at +125°C, and temperature cycled ten times from -65°C to +130°C prior to final test to insure reliability and long-term stability. In addition, all units are tested 100% at the extremes of their respective temperature ranges for all parameters to guarantee full performance.

*Covered by Patent No. 3,940,760, other patents pending.

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

P.O. Box 280; Norwood, Massachusetts 02062 U.S.
Telex: 924491 Cables: ANALOG NORWOODMO
SPECIFICATIONS
(typical @ +25°C with V+ = +5V, V- = -15V, all voltages measured with respect to digital common, unless otherwise indicated)

<table>
<thead>
<tr>
<th>MODEL</th>
<th>AD570JD</th>
<th>AD570SD/AD570SD-883B²</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOLUTION²</td>
<td>8 Bits</td>
<td>*</td>
</tr>
<tr>
<td>RELATIVE ACCURACY @ 25°C,3,4</td>
<td>±1/2LSB max</td>
<td>±1/2LSB max</td>
</tr>
<tr>
<td>Tmin to Tmax</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>FULL SCALE CALIBRATION⁴,⁶ (With 15Ω Resistor In Series With Analog Input)</td>
<td>±2LSB (typ)</td>
<td>*</td>
</tr>
<tr>
<td>UNIPOLAR OFFSET (max)⁴</td>
<td>±1/2LSB</td>
<td>*</td>
</tr>
<tr>
<td>BIPOLAR OFFSET (max)⁴</td>
<td>±1/2LSB</td>
<td>*</td>
</tr>
<tr>
<td>DIFFERENTIAL NONLINEARITY (Resolution for Which no Missing Codes are Guaranteed)</td>
<td>±2LSB</td>
<td>*</td>
</tr>
<tr>
<td>+25°C</td>
<td>8 Bits</td>
<td>*</td>
</tr>
<tr>
<td>Tmin to Tmax</td>
<td>8 Bits</td>
<td>*</td>
</tr>
<tr>
<td>TEMPERATURE RANGE</td>
<td>0 to +70°C</td>
<td>-55°C to +135°C</td>
</tr>
<tr>
<td>TEMPERATURE COEFFICIENTS⁶ Guaranteed max Change Tmin to Tmax Unipolar Offset</td>
<td>±1LSB (88ppm/°C)</td>
<td>±1LSB (40ppm/°C)</td>
</tr>
<tr>
<td>Bipolar Offset</td>
<td>±1LSB (88ppm/°C)</td>
<td>±1LSB (40ppm/°C)</td>
</tr>
<tr>
<td>Full Scale Calibration⁶ (With 15Ω Fixed Resistor or 200Ω Trimmer)</td>
<td>±2LSB (176ppm/°C)</td>
<td>±2LSB (80ppm/°C)</td>
</tr>
<tr>
<td>POWER SUPPLY REJECTION⁴ Max Change In Full Scale Calibration TTL Positive Supply +4.5V&lt;V+&lt;-5.5V</td>
<td>±2LSB max</td>
<td>*</td>
</tr>
<tr>
<td>Negative Supply -16.5V&lt;V-&lt;-13.5V</td>
<td>±2LSB max</td>
<td>*</td>
</tr>
<tr>
<td>ANALOG INPUT RESISTANCE</td>
<td>3kΩ min</td>
<td>*</td>
</tr>
<tr>
<td>5kΩ typ</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>7kΩ max</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ANALOG INPUT RANGES (Analog Input to Analog Common) Unipolar</td>
<td>0 to +10V</td>
<td>*</td>
</tr>
<tr>
<td>Bipolar</td>
<td>-5V to +5V</td>
<td>*</td>
</tr>
<tr>
<td>OUTPUT CODING Unipolar</td>
<td>Positive True Binary</td>
<td>*</td>
</tr>
<tr>
<td>Bipolar</td>
<td>Positive True Offset Binary</td>
<td>*</td>
</tr>
<tr>
<td>LOGIC OUTPUT Bit Outputs and Data Ready Output Sink Current (V&lt;sub&gt;OUT&lt;/sub&gt; = 0.4V max, T&lt;sub&gt;min&lt;/sub&gt; to T&lt;sub&gt;max&lt;/sub&gt;)</td>
<td>3.2mA min</td>
<td>*</td>
</tr>
<tr>
<td>(2TTL Loads)</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Output Source Current (Bit Outputs)⁷ (V&lt;sub&gt;OUT&lt;/sub&gt; = 2.4V min, T&lt;sub&gt;min&lt;/sub&gt; to T&lt;sub&gt;max&lt;/sub&gt;)</td>
<td>0.5mA min</td>
<td>*</td>
</tr>
<tr>
<td>Output Leakage When Blanked</td>
<td>±40μA max</td>
<td>*</td>
</tr>
<tr>
<td>LOGIC INPUT Blank and Convert Input 0&lt;V&lt;sub&gt;in&lt;/sub&gt;&lt;SV+</td>
<td>±40μA max</td>
<td>*</td>
</tr>
<tr>
<td>Blank – Logic &quot;1&quot;</td>
<td>2.0V min</td>
<td>*</td>
</tr>
<tr>
<td>Convert – Logic &quot;0&quot;</td>
<td>0.8V max</td>
<td>*</td>
</tr>
<tr>
<td>CONVERSION TIME</td>
<td>15μs min</td>
<td>*</td>
</tr>
<tr>
<td>25μs typ</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>40μs max</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>
**MODEL**

<table>
<thead>
<tr>
<th>POWER SUPPLY</th>
<th>AD570JD</th>
<th>AD570SD/AD570SD-883B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute Maximum</td>
<td>+7V</td>
<td>•</td>
</tr>
<tr>
<td>V-</td>
<td>-16.5V</td>
<td>•</td>
</tr>
<tr>
<td>Specified Operating - Rated Performance</td>
<td>+5V</td>
<td>•</td>
</tr>
<tr>
<td>V-</td>
<td>-15V</td>
<td>•</td>
</tr>
<tr>
<td>Operating Range</td>
<td>+4.5V to +5.5V</td>
<td>•</td>
</tr>
<tr>
<td>V-</td>
<td>-12.0V to -16.5V</td>
<td>•</td>
</tr>
<tr>
<td>Operating Current</td>
<td>2mA typ (10mA max)</td>
<td>•</td>
</tr>
</tbody>
</table>

**NOTES:**

1. The AD570SD is available fully processed and screened to the requirements of MIL-STD-883B, Class B. A complete list of tests is given on page 6. When ordering, specify the AD570SD/883B.

2. The AD7570 is a selected version of the AD571 10-bit A to D converter. As such, some devices may exhibit 9 or 10 bits of relative accuracy or resolution, but that is neither tested nor guaranteed. Only TTL logic inputs should be connected to pins 1 and 18 (or no connection made) or damage may result.

3. Relative accuracy is defined as the deviation of the code transition point from the ideal transfer point on a straight line from the zero to the full scale of the device.

4. Specifications given in LSB's refer to the weight of a least significant bit at the 8-bit level, which is 0.39% of full-scale.

5. Full scale calibration is guaranteed trimmable to zero with an external 200Ω potentiometer in place of the 15Ω fixed resistor. Full scale is defined as 10 volts minus 1 LSB, or 9.961 volts.

6. Full Scale Calibration Temperature Coefficient includes effects of unipolar offset drift as well as gain drift.

7. The Data output lines have active pull-ups to source 0.5mA. The DATA READY line is open collector with a nominal 6kΩ internal pull-up resistor.

**ABSOLUTE MAXIMUM RATINGS**

- V+ to Digital Common: 0 to +7V
- V- to Digital Common: 0 to -16.5V
- Analog Common to Digital Common: ±1V
- Analog Input to Analog Common: ±15V
- Control Inputs: 0 to V+
- Digital Outputs (Blank Mode): 0 to V+
- Power Dissipation: 800mW

---

**OUTLINE DIMENSIONS**

Dimensions shown in inches and (mm).

---
QUADRUPLE 2-INPUT NAND GATE

The HEF4011UB is a quadruple 2-input NAND gate. This unbuffered single stage version provides a direct implementation of the NAND function. The output impedance and output transition time depends on the input voltage and input rise and fall times applied.

Fig. 1 Functional diagram.

Fig. 2 Pinning diagram.

HEF4011UBP: 14-lead DIL; plastic (SOT-27K, M, T).
HEF4011UBD: 14-lead DIL; ceramic (cerdip) (SOT-73).
HEF4011UBT: 14-lead mini-pack; plastic (SO-14; SOT-108A).

Fig. 3 Schematic diagram (one gate). The splitting-up of the n-transistors provide identical inputs.

FAMILY DATA

| see Family Specifications | I_DD LIMITS category GATES |

Products approved to CECC 90 104-009.

May 1983
A.C. CHARACTERISTICS

V_{ss} = 0 V; T_{amb} = 25 °C; C_L = 50 pF; input transition times < 20 ns

<table>
<thead>
<tr>
<th>Propagation delays</th>
<th>V_{DD}</th>
<th>symbol</th>
<th>typ.</th>
<th>max.</th>
<th>typical extrapolation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_n \rightarrow O_n</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH to LOW</td>
<td>5</td>
<td>t_{PHL}</td>
<td>60</td>
<td>120  ns</td>
<td>25 ns + (0,70 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>25</td>
<td>50   ns</td>
<td>12 ns + (0,27 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>20</td>
<td>40   ns</td>
<td>10 ns + (0,20 ns/pF) C_L</td>
</tr>
<tr>
<td>LOW to HIGH</td>
<td>5</td>
<td>t_{PLH}</td>
<td>35</td>
<td>70   ns</td>
<td>8 ns + (0,55 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>20</td>
<td>40   ns</td>
<td>9 ns + (0,23 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>17</td>
<td>35   ns</td>
<td>9 ns + (0,16 ns/pF) C_L</td>
</tr>
<tr>
<td>Output transition times</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH to LOW</td>
<td>5</td>
<td>t_{THL}</td>
<td>75</td>
<td>150  ns</td>
<td>15 ns + (1,20 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>30</td>
<td>60   ns</td>
<td>6 ns + (0,48 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>20</td>
<td>40   ns</td>
<td>4 ns + (0,32 ns/pF) C_L</td>
</tr>
<tr>
<td>LOW to HIGH</td>
<td>5</td>
<td>t_{TLH}</td>
<td>60</td>
<td>110  ns</td>
<td>10 ns + (1,00 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td>30</td>
<td>60   ns</td>
<td>9 ns + (0,42 ns/pF) C_L</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>20</td>
<td>40   ns</td>
<td>6 ns + (0,28 ns/pF) C_L</td>
</tr>
<tr>
<td>Input capacitance</td>
<td>C_{IN}</td>
<td></td>
<td>10</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic power dissipation per package (P)</th>
<th>V_{DD}</th>
<th>typical formula for P (\mu W)</th>
<th>where</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>500 f_i + \Sigma(f_o C_L) \times V_{DD}^2</td>
<td>f_i = input freq. (MHz)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5000 f_i + \Sigma(f_o C_L) \times V_{DD}^2</td>
<td>f_o = output freq. (MHz)</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>25000 f_i + \Sigma(f_o C_L) \times V_{DD}^2</td>
<td>C_L = load capacitance (pF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\Sigma(f_o C_L) = sum of outputs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>V_{DD} = supply voltage (V)</td>
</tr>
</tbody>
</table>
ANNEX E

Conversion hardware:

Input from ADC 8 bits + 1 output bit. 

Input Voltage DC, 8 bits.

Semiconductors and intercom: actions are: multiple action, count 31, count 31. 2 pin, of course. 1 record.

Semiconductors, intercom: intercom 1. Timer C, 1 to PII, PII.

Clock: 8 MHz.

Input 8 bits, ADC. Port 16 (address CA).

Input 8 bits, ADC. Port 16 (address CB).

Output 8 bits, Port C (address CC).

Output 8 bits, Port C (address CC).

Input 8 bits, intercom 1 to 31.

EXTRACT TO EXIT:

Initialization requires control word + control registers.

Modeled interrupt:

Control word AD = 06

load com 10. DO

read code 0. = DO

Clock Table 2.5

count = Time x F

load 1239. Decimal clock 12345 Hz.

A = 12345

A + 0x100 = 10000 (address CC)

ICW2 (C2)

ICW4 (C2) 000.1.111.1.1

MASK (C2)
Port C0 (output DAC) requires an IC (IC 810) on the single board.
Port CC bit 4 requires a 74LS08 (IC A11) and resistor termination (IC A11).
Port CA ADC requires resistor termination (IC A11).