MICROCONTROLLER ELECTRONICS FOR A
PASSIVE SELF-CONTAINED ABOVE-KNEE PROSTHESIS

by
Karl E. Lindstrom

SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF SCIENCE IN ELECTRICAL SCIENCE AND ENGINEERING

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1987

© Karl E. Lindstrom

The author hereby grants to M.I.T. permission to
reproduce and to distribute copies of this document in
whole or in part.

Signature of Author

Certified by ____________________________
Woodie C. Flowers
Thesis Supervisor

Accepted by ____________________________
David Adler
Chairman, Department Committee

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARCHIVES
JUL 15 1987
LIBRARIES
MICROCONTROLLER ELECTRONICS FOR A
PASSIVE SELF-CONTAINED ABOVE-KNEE PROSTHESIS

by

KARL E. LINDSTROM

Submitted to the Department of Electrical Engineering
on May 18, 1987 in partial fulfillment of the
requirements for the degree of Bachelor of Science.

ABSTRACT

There are a wide range of devices for people who need an above-knee prosthesis. Most of these are constrained to a fixed damping characteristic at the knee joint. A need exists for a device which is controllable and self-contained.

The development of recent technologies in the mechanical and electrical engineering fields offers a unique opportunity to develop this mechanism. Once developed, the mechanism would be fitted on an amputee's leg and through interaction with a personal workstation, the damping characteristics could be "fine-tuned" to the user's specific requirements.

This thesis concerns the controlling electronics for an above-knee prosthesis knee mechanism. At the start of this project, the controller electronics had been designed and tested, but not fully implemented. The goal of the project was to fix any existing bugs, and to finish the implementation so that the knee would be a self-contained and working system.

The feasibility of such a system was tested and demonstrated.

Thesis Supervisor: Woodie C. Flowers

Title: Associate Professor of Mechanical Engineering
Acknowledgements

First of all I would like to thank my mother and father, without whose endless support and enthusiasm, I would not be were I am today.

I would like to thank Professor Flowers for introducing me to this project, I never thought it would be this much fun or as fulfilling as it turned out to be.

Thank you Stuart Schecter for all of the help in bringing me up to speed with the system and for the answering of my many questions on the design considerations in the project. Thank you also for infecting me with the enthusiasm associated with working on this project.

Thanks goes to Bill Murray for the use of his computer systems and whose help around the lab were invaluable during the long nights of implementation.

Thanks Steve, Stuart and Debbie for all your help and support.

Finally, I would like to thank Marco for the use of this computer system, the hardware, the advice given, and for being a friend in the many times I needed one.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>2</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>3</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>4</td>
</tr>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 1 - Introduction and Background</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 2 - Hardware Design</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 3 - Software Design</td>
<td>29</td>
</tr>
<tr>
<td>Chapter 4 - Evaluation and Recommendations</td>
<td>34</td>
</tr>
<tr>
<td>References</td>
<td>37</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
</tr>
<tr>
<td>A - PD78C10 Data Sheet</td>
<td>39</td>
</tr>
<tr>
<td>B - Computer Board Schematic</td>
<td>42</td>
</tr>
<tr>
<td>C - Software Programs</td>
<td>53</td>
</tr>
<tr>
<td>D - Supplier Acknowledgements</td>
<td>68</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>TITLE</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Complete Knee Unit With Shank and Foot Attached</td>
<td>11</td>
</tr>
<tr>
<td>1-2</td>
<td>Schematic of Magnetic Particle Brake</td>
<td>12</td>
</tr>
<tr>
<td>1-3</td>
<td>Magnetic Particle Brake Transmission Unit</td>
<td>13</td>
</tr>
<tr>
<td>1-4</td>
<td>Position Transducer</td>
<td>14</td>
</tr>
<tr>
<td>2-1</td>
<td>Microcomputer Board</td>
<td>24</td>
</tr>
<tr>
<td>2-2</td>
<td>Block Diagram</td>
<td>25</td>
</tr>
<tr>
<td>2-3</td>
<td>Schematic of Electric FET switch</td>
<td>26</td>
</tr>
<tr>
<td>2-4</td>
<td>Serial Port Details</td>
<td>27</td>
</tr>
<tr>
<td>2-5</td>
<td>Ideal Transfer Character of a Band - Pass Filter</td>
<td>28</td>
</tr>
<tr>
<td>3-1</td>
<td>Monitor Program Command Details</td>
<td>32</td>
</tr>
<tr>
<td>3-2</td>
<td>Figure not available</td>
<td>na</td>
</tr>
<tr>
<td>3-3</td>
<td>Details for Programmable Square Wave</td>
<td>33</td>
</tr>
<tr>
<td>4-1</td>
<td>Computer Board Attached to the Knee Unit</td>
<td>36</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Background

An above-the-knee (A/K) prosthesis is a device which allows a person who has lost their leg below the thigh to walk. Such a device could be as simple as a two-bar linkage which swings freely around the knee-joint. But that particular solution is rather simple and barely adequate for subjects' needs. To solve this problem, there has been ongoing research to improve prosthesis characteristics. This thesis is part of the ongoing prosthesis research and development at the Laboratory for Biomechanical and Human Research at M.I.T.

The passive, self-contained, above-knee prosthesis is an outgrowth of some of the 15 years of research at Laboratory for Biomechanical and Human Research. The research has looked into two major areas of A/K prosthesis design. The first area concerned designs which use "active" mechanisms (such as a motor to introduce energy into the system). The second dealt with passive mechanisms (such as a brake to dissipate energy in the system). For a detailed account of such research, see Schecter (26).

One of the major problems with both active and
passive research areas included the need for relatively "large" amounts of power. This meant that the systems had to have power delivered through umbilical cords or else the power supplies had to be carried by the subjects. Another problem encountered was the inflexibility of the systems. The knee-mechanisms lacked the ability to respond to the changing needs of the user. For example, these mechanisms could not change their attributes to accommodate walking up stairs versus walking along a hall (two actions which require very different characteristics in a knee).

Although these problems had been solved separately, there had, as yet, been no system which totally solved the problems of self-containment and of flexibility. The passive, self-contained, microcomputer controlled A/K prosthesis developed by Schecter (26) represents an effort to achieve such a system (see Figure 1-1).

Introduction to the System

At the heart of that mechanism is the magnetic-particle brake (MPB) (see Figure 1-2). It is a small, commercially available, lightweight, electrically controllable, energy dissipator. The MPB dissipates the necessary energy levels, and has a relatively quick response time (see Figure 1-3). Because the brake is electrically controllable it is relatively easy to
interface with a microcontroller. Both the brake unit and the microcomputer system have relatively low power consumption (on the order of 50mA at 5V) and can be powered by a rechargeable battery pack.

Rechargeable batteries were chosen to give the brake a life cycle of about 12 days in before recharging is necessary.

The housing was built with many considerations in mind. First of all, it was designed from a strong, durable, and lightweight material. Good-looks and clean lines were important to inspire user confidence. The housing design also had to facilitate the placement of transducers. The specific transducers accounted for were strain gages and a goniometer (a potentiometer which measures angle). The strain gages were designed to lie on specific beams carved in the housing where there was relevant strain information (see Schecter (26)). Specific spots were designated for the MPB and the battery pack as well. Finally, the housing was designed so that at full extension, the mechanism would not require support from the Magnetic Particle Brake.

The angle transducer is a Bourns thin film, 0.5 percent linearity, ball bearing potentiometer. It is connected to the knee axis through a timing belt and pulleys at a 2.25:1 ratio. This gearing was chosen to maximize the range of differentiable information from the
transducer.

As mentioned before, the mechanism was designed to be user-tunable through interaction with a personal computer. The design also accounted for user control through switches which were either on board (such as dip switches) or connected to (such as muscle-activated switches) the system. Lastly, the mechanism was designed to implement a wide range of control algorithms using various types of physical information.

All of these features meant that the microcontroller had to have certain governing characteristics. It was necessary to have the ability to read and convert the information from transducers. The microcomputer had to have consistent and linear control of the magnetic particle brake's damping character. A volatile memory space was necessary to store important variables so that the variables could be interactively changed. And perhaps the most important characteristic required was the ability to interact with, and be controlled by, the outside world through a serial communication port.

This Thesis's Goals

When this thesis project began, the mechanical part of the project had been fully designed and implemented. But, the microcontroller, while it had been designed, was not fully implemented. Part of this thesis involved
fixing and possibly redesigning the controller circuit hardware and the accompanying software. The final goal was to achieve a self-contained knee-apparatus which demonstrated a viscous (velocity dependent) damping profile.

More specifically, this involved:

1. Learning the use and design of the system.

2. Learning how to use the computer system on which the assembly language programs were written.

3. Learning the Assembly language for the microprocessor.

4. Fixing the serial communications port.

5. Examining and fixing any wiring mistakes.

6. Rewriting the software for the system.

7. Redesigning and building the circuits for the transducer information.

8. Redesigning and building the circuit to control the magnetic particle brake's characteristics.

9. Writing a viscous damping program.

10. Making the system self-contained by mounting the microcomputer board and the batteries on the housing.
Figure 1-1  COMPLETE KNEE UNIT WITH SHANK AND FOOT ATTACHED
Figure 1-2  SCHEMATIC OF MAGNETIC PARTICLE BRAKE
Figure 1-3 MAGNETIC PARTICLE BRAKE TRANSMISSION UNIT
Figure 1-4  POSITION TRANSDUCER
Chapter 2

HARDWARE DESIGN

The hardware was designed and successfully implemented. Discussed below are the constraints on the hardware and the system hardware features.

Constraints on Hardware

There are many constraints on the hardware. These constraints are imposed by the need for a self-contained prosthesis with programmable characteristics. Many of these constraints are not immediately obvious and affect various parts of the system design. The important ones are outlined below along with the design reaction to those constraints.

1. Low power consumption:

The reasons behind low power consumption are fairly straightforward. If the A/K prosthesis is to be a self-contained system, the power supply has to be as small and as lightweight as possible. Because power packages generally increase in size as you demand more power, low consumption is an issue. Also, lower power draw is a good way to increase the time between battery recharges.

For those reasons the design used primarily CMOS
technology and the smallest number of chips possible. With less peripheral chips, the microcomputer had to do most of the work and accordingly it had to be versatile and very powerful. Another concession to this point was the elimination of on-board LED's because they would draw too much power. Also, the design tries to emphasize high resistor values so less current is drawn. 2. Small size:

Here again, the use of the smallest number of chips and the smallest package size of a chip is critical to achieving a self-contained system. Again, the microcomputer has to be powerful to eliminate the need for peripheral hardware. Other design considerations include the use of small connectors, planning multi-purpose use of switches and moving the LED's off the board to help conserve critical board space.

3. Flexible and expandible:

Since this is an ongoing and developing project, flexibility is important to meet the demands of changes in design. This meant the system had to leave space for a second level of chips. It meant the system had to have extra, unused op-amps for new transducers. Switches were placed on Port A of the microcomputer to allow for a way to designate new states and for tweaking various parameters manually. Finally, the microprocessor chip had to be very flexible and powerful.

4. 0 to 5V supply:
The batteries used for the microcomputer board were constrained to a limit of 0 to +5V and thus, there were special compensations designed. In particular, the system uses special charge-pump chips to generate the voltage levels necessary for serial communication and op-amp operation.

5. Must interface with physical data:

This meant that the board has to have some sort of Analog to Digital conversion ability because transducer information is generally in the form of analog voltage signals. Op-amps are also necessary to buffer the conversion stage from the input signals.

6. Must have communications:

There has to be a serial port communications facility on board the microcontroller.

7. Fast micro-computer:

The hardware is required to be fast enough to read in the transducer data, to process it and then to modify the brake's characteristic before the next sampling cycle. This means a relatively fast microprocessor must be used. The NEC chip used a clock crystal of 11.059 MHz to achieve an instruction cycle time of about 1 us.
System Features

Below is a listing of the major systems and features of the actual designed microcontroller electronics (see APPENDIX B for a detailed look at circuits). An overall block diagram can be seen in Figure 2-2.

1. Wire-wrap board:
   The electronics are mounted on a flexible, pre-cut wire-wrap board (see Figure 2-1). This board is shaped to fit on the housing of the knee-mechanism. The board is mounted using non-conductive spacers so that none of the pins are touching the housing.

2. Microprocessor:
   These were the major factors which determined the choice of this particular microprocessor. The NEC uMP78C10 chip:
   a. Allows access to 64K bytes of memory
   b. Has a versatile arithmetic/logic unit including multiply and divide commands
   c. Has 8 channels of analog to digital conversion as well as edge triggering capabilities,
   d. Has an on board general serial interface
   e. Is able to run at 11 MHz
   f. Requires only 64 pins in a 1.63 in. by 0.99 in
package.

g. Is available in a CMOS package.

3. Serial interface:

The chip used was a MAXIM 232 chip which uses charge-pumped capacitors to raise the voltage levels to the proper +10 V voltage levels. It also converts the RS232 input signals to the proper 0 to 5V levels for CMOS/TTL UART compatibility (see figure 2-4 for original configuration).

4. Program memory:

The chips used are a CMOS EPROM (4K x 8) and a static RAM chip (8K x 8). The Eprom is programmed with the controlling assembly program hex file. The monitor program accepts hex files sent over the serial communications port and loads them into the RAM. This way, programs can be run on the fast access times of the static RAM. With both of these memory chips there is a usable space of 12 Kilobytes for programs--more than enough for this project's software demands.

The RAM provides a volatile memory space where important variables may be stored and easily changed.

5. +5V to -5V Conversion:

The system uses a MAXIM 7660 chip to convert +5V to the -5V level necessary for proper Op-amp operation. The
chip is monolithic, and the actual output voltage level was measured to be around \(-4V\). This implies an Op-Amp current draw of about 20mA according to manufacture information (see charts in Appendix B).

6. A-to-D op-amp buffer subsection:

The op-amps buffer the transducer data from the potentiometer on the knee. This information is translated into a 0 to 5V voltage reading which is not entirely stable due to inconsistencies in power and potentiometer and the op-amps. That information is fed to the A-to-D section of the microcomputer (at AN0) were it is resolved into one of 256 values (8-bit A/D conversion). This produces the angle information for the microprocessor.

The velocity signal is generated by sending the angle signal through an analog differentiation circuit using op-amps and resistors (see figure in APPENDIX B). The circuit is basically a band-pass filter (see Figure 2-5). The filter produces a \(-5\) to +5 V signal which must be raised to a 0 to +5V signal so that it may be read by the A-to-D section of the NEC chip (at AN2). That is then read in as the velocity signal for the computer.

There is a final op-amp which is driven to the highest value possible and fed in as the reference voltage for the A/D section of the microprocessor (at
VAref). This is done because the maximum voltage output of the op-amps is measured to be less than 5V. If the reference voltage was 5V, the microcomputer would lose some signal resolution. Since the A/D converter is always buffered by an op-amp, this reference signal assures maximum resolution no matter how the op-amps perform.

7. D-to-A buffer (MOSFET switches):

This circuit controls the operation of the magnetic particle brake. It consists of a set of 3 cascaded MOSFET switches. This configuration is necessary because of the large current draw induced by the magnetic particle brake. When only one MOSFET switch was used the current draw on the gate to source was too large to allow an op-amp signal to close the switch. Experimental evidence shows that 3 cascade levels are necessary before the switch will close on typical CMOS outputs current levels from the op-amps.

The MOSFET circuit is controlled by an op-amp in a Schmitt-trigger configuration. This configuration forces the op-amp into positive or negative saturation if the signal at the positive input of the op-amp is less than or greater than the voltage at the negative input to the op-amp (respectively). The voltage at the negative input has set at 2.5 V. This allows a CMOS 0 to +5V square
wave to trigger the switches into full on (at 0V) or full off (at 5V) at all times. This signal is being generated by a port C output pin (PC7) which can generate a controllable square-wave signal. Control of this signal is further discussed in chapter 3 (see figure 2-3 for original hardware design).

8. Switches and LED's:

These switches can be used for either turning systems on and off or to control the state of the machine. With these switches for control there doesn't need to be terminal to control the system. This is important because a terminal is not always readily available. Thus, these switches could be used to tweak the system's damping characteristics in emergencies (especially the 3-position, 2 pull switch (S8, S9)).

LED's are off the board because of space and power considerations and should be used only for debugging the system or when a 5V power supply is available.

9. Power Considerations:

The microcomputer electronics were tested to have a 50mA draw at 5V. This drained the two 6V photograph batteries in less than 2 hours time.

The magnetic particle brake was tested and found to have a maximum draw of 120 mA at 9V. At an average of
about a 50% duty cycle, the 9V transistor battery lasted about 4 hours.

For further power analysis see Chapter 4.
Figure 2-2 Block Diagram
Figure 2-3  SCHEMATIC OF ELECTRONIC FET SWITCH
Figure 2-4  SERIAL PORT DETAILS
\[
C_1 = 1 \mu F \quad R_S C_1 = 2 \times 10^{-2} \text{ s}
\]
\[
C_2 = 0.1 \mu F \quad R_F C_2 = 1 \times 10^{-2} \text{ s}
\]
\[
R_F = 100 \text{K} \quad \frac{R_F}{R_S} = 5
\]

\[
\frac{V_o}{V_i} = -\frac{sC_1R_F}{(1+sC_1R_S)(1+sC_2R_F)}
\]

Transfer Function

\begin{figure}
\centering
\includegraphics[width=\textwidth]{pole-zero-diagram}
\caption{Pole-zero diagram}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{magnitude-phase-diagram}
\caption{Magnitude and Phase diagrams}
\end{figure}

**Figure 2-5** Ideal Transfer Character of a Band-Pass Filter
Chapter 3

SOFTWARE DESIGN

Constraints on the Software

1. Initialization:

Since the microprocessor is so powerful and versatile it allows the user control of all its functions. The only drawback is that it means that the programmer has to initialize the function of every subsystem before it can be used. This meant a large percentage of programming time was spent writing code to initialize and then control the reading/writing cycles for all of the Input/Output ports (serial, A/D, and Square Wave generation.)

2. Assembly language:

The programming is done at the assembly level. The code is 8085-compatible, and mainly features 8-bit operations. But there are also 16-bit operations including fast 16-bit multiply and divide commands. The monitor program was developed on an older CPM 20 system in the Laboratory for Biomechanics and Human Research. It was developed with a NEC 7800 non-relocatable assembler-debugger.

3. 4Kbyte instruction limit
This limit is imposed due to the EPROM size chosen in the design.

**Monitor Program**

This program allows the user to communicate and to the knee mechanism through the serial port of a terminal. The transmission rate is set at 9600 baud. The monitor program allows the user to examine and modify information stored in the microcontroller.

The program starts with an initial menu screen. Once an option is chosen the program proceeds to another screen which may prompt the user for certain hex values. The hex values must end in an "H" and they are constrained to a limited range of values. The load option is designed to accept the transmission of a standard ascii hex file in INTEL standard format. For a brief summary of options and functions see Figure 3-1.

**Viscous Damping program**

The basic idea behind the program is for the microprocessor to constantly generate a 1 Khz square wave, whose positive pulse-width can be changed by controlling a certain register's value. The signal is coming out of Port C.7 of the microprocessor chip. The value in the register is generated by multiplying the velocity of the knee mechanism by a constant.
The program had to control Port C.7 (refer to APPENDIX B) and output that signal to the D-to-A section of the system. If you refer to figure 3-3 you will see that the program varied to value of 'n' while holding the value of 'm' constant.
C[LEAR] [<LOC>][,<NUM>]
Clear <NUM> bytes of RAM starting at <LOC>.

E[XAMINE][<REG>]
Examine 7810 registers as specified by
<REG>. <REG> = 1  MAIN REGISTERS
<REG> = 2  MAIN & ALT REGISTERS
<REG> = 3  MAIN & ALTERNATE & SPECIAL REGISTERS

H[ALT]
Halts computer without saving registers.

L[OAD]
Sets up computer to receive a HEX file and
stays in this mode until an "end of file"
delimiter is found (ASCII "=" character).

P[INT] [<LOC>][,<NUM>]
Sends <NUM> bytes of memory starting
at <LOC>.

R[UN] [<LOC>]
Sets the program counter to <LOC>.

[ ]- bracket means default values may be used.

DEFAULT VALUES:
<LOC> = 2000H  <NUM> = 3FFFH  <REG> = 1

Figure 3-1  COMMAND DETAILS
Output Control Circuit

Reference clock (fAs)

CP0

CP1

CO0

Start

Remarks: ETM0 register = m
ETM1 register = n

(m < n)
(m and n are count values.)

Square Wave Output

Figure 3-3 Details for Programmable Square Wave
Chapter 4

EVALUATION AND RECOMMENDATIONS

Conclusion

The goals of the thesis project were accomplished! The knee was assembled and ran under its own power entirely. It was self-contained and ran a viscous damping program of control. The monitor program and communication facilities were fixed and demonstrated to work.

There is still more work to be done on this project. The software control can be more improved to process more information. The hardware could be updated to provide the microprocessor with additional transducer information. This would help the system by making more "ideal" walking gait algorithms possible. Finally, the interface with an IBM PC still needs to be written.

Future Recommendations

Beyond these general comments, I would like to make some specific recommendations for the near future.

a. Fix up hardware

-- Neaten the wiring. Construct better connectors for power, brake, and transducers. Stabilize the 3-position switch on the board. All of this will help to inspire user confidence.
b. Trim power consumption (250 mW)
-- Monitor consumption of different board subsystems.
Find optimal frequencies for reading in data and for the pulse-width modulation scheme.

c. Finish monitor program
-- Neaten the output so it is fully functional and brings up professional-looking screen to monitors.

d. Finish damping program
-- Allow the user to interactively change the damping profile of the knee by either a terminal or switches.

e. Write IBM PC software serial interface
-- It should be written to communicate with the leg in an simple and direct fashion.
Figure 4-1  COMPUTER BOARD ATTACHED TO THE KNEE UNIT
REFERENCES


APPENDIX A

uP78C10 DATA SHEET
μPD78C10/78C11

CMOS Single-Chip 8-Bit Microcomputer with A/D Converter
Description

The NEC uPD78C10/uPD78C11 is a high performance, single chip microcomputer integrating sophisticated on-chip peripheral functionality normally provided by external components. The device's internal 16-bit ALU and data paths, combined with a powerful instruction set and addressing, make the uPD78C10/uPD78C11 appropriate in data processing as well as control applications. The device integrates a 16-bit ALU, 4K-ROM, 256-byte RAM with an 8-channel A/D converter, a multifunctional 16-bit timer/event counter, two 8-bit timers, a USART and two zero-cross detect inputs on a single die, allowing its use in fast, high end processing applications. This involves analog signal interface and processing.

The uPD78C11 is the mask-ROM high volume production device embedded with custom customer program. The uPD78C10 is a ROM-less version for prototyping and small volume production.

Features

- CMOS technology
  - 2.5 to 6.0 V operating range
  - 17 mA operating supply current
- Complete single chip microcomputer
  - 16-bit ALU
  - 4K-ROM
  - 256-byte RAM
- 44 I/O lines
- Two zero-cross detect inputs
- Two 8-bit timers
- Multifunction 16-bit timer/event counter
- Expansion capabilities
  - 8085A bus-compatible
  - 60K-byte external memory address range
- 8-channel, 8-bit A/D converter
  - Autoscan mode
  - Channel select mode
- Full duplex USART
  - Synchronous and asynchronous
- 153 instruction set
  - 16-bit arithmetic, multiply and divide
- 1 us instruction cycle time (12 MHz operation)
- Prioritized interrupt structure
  - 3 external
  - 8 internal
- 2 Standby modes
  - HALT mode uses 9 mA supply current
  - STOP mode uses less than 1 uA supply current
- On-chip clock generator
<table>
<thead>
<tr>
<th>Pin No.</th>
<th>DIP/QUIP</th>
<th>Flat</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8</td>
<td>1, 2</td>
<td>59-64</td>
<td>PA0-P7</td>
<td>Port A: (Three-state input/output) 8-bit programmable I/O port. Each line independently programmable as an input or output. Reset places all lines of Port A in input mode.</td>
</tr>
<tr>
<td>9-16</td>
<td>3-10</td>
<td></td>
<td>PB0-P7</td>
<td>Port B: (Three-state input/output) 8-bit programmable I/O port. Each line independently programmable as an input or output. Reset places all lines of Port B in input mode.</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td></td>
<td>PC0</td>
<td>Transmit Data (TxD): Serial data output terminal.</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td></td>
<td>PC1</td>
<td>Receive Data (RxD): Serial data input terminal.</td>
</tr>
<tr>
<td>Pin No.</td>
<td>DIP/QUIP</td>
<td>Flat</td>
<td>Symbol</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>PC2</td>
<td></td>
<td>Port C:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Three-state</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>input/output)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8-bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>programmable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I/O port.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Each line</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>independently</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>programmable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>as an input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>or output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Alternatively</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Port C may be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>used as con-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>trol lines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>for USART</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and timer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reset puts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Port C in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port mode and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>all lines in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>input mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Serial Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(SCK): Serial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clock input/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>output ter-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>minal. When</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>internal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clock is used,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the output can</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>be selected;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when an external</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clock is used,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the input can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>selected.</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>PC3</td>
<td></td>
<td>Timer Input (TI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>/interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>request input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(INT2): Timer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>clock input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>terminal; can</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>also be used as</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>falling edge,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>maskable inter-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rupt input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>terminal and AC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>input zero-cross</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detection ter-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>minal.</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>PC4</td>
<td></td>
<td>Timer Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(TO): This</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>output signal is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a square wave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>whose frequency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is determined by</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the timer/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>counter.</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>PC5</td>
<td></td>
<td>Counter Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(CI): External</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>pulse input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>terminal to the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>timer/event</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>counter.</td>
</tr>
<tr>
<td>Pin No.</td>
<td>Flat</td>
<td>Symbol</td>
<td>Function</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>DIP/QUIP</td>
<td>Flat</td>
<td>Symbol</td>
<td>Function</td>
<td></td>
</tr>
<tr>
<td>23,24</td>
<td>17,18</td>
<td>PC6,PC7</td>
<td>Counting Outputs 0,1 (CO0-CO1): Programmable rectangular wave output terminal based on timer/event counter.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>NMI</td>
<td>Falling edge, nonmaskable interrupt (NMI) input.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>20</td>
<td>INT1</td>
<td>This signal is a rising edge, maskable interrupt input. This input is also used to make the zero-cross detection AC input.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>21</td>
<td>MODE0</td>
<td>Used as input in conjunction with MODE0 to select appropriate memory expansion mode. Also outputs M1 signal during each opcode fetch.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>22</td>
<td>RESET</td>
<td>(Input, active low), RESET initializes the uPD7811.</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>23</td>
<td>MODE1</td>
<td>Used as input in conjunction with MODE1 to select appropriate memory expansion mode. Also used to output IO/H.</td>
<td></td>
</tr>
<tr>
<td>30,31</td>
<td>24,25</td>
<td>X2,X1</td>
<td>This is a crystal connection terminal for system clock oscillation. When an external clock is supplied X1 is the input.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(crystal)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>26</td>
<td>VSS</td>
<td>Power supply ground potential.</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>27</td>
<td>AVSS</td>
<td>A/D converter power supply ground potential.</td>
<td></td>
</tr>
<tr>
<td>34-41</td>
<td>28-35</td>
<td>AN0-AN7</td>
<td>Eight analog inputs to the A/D converter. AN7-AN4 can also be used as a digital input port for falling edge detection.</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>36</td>
<td>VAREF</td>
<td>Reference voltage for A/D converter. Sets conversion range upper limit.</td>
<td></td>
</tr>
<tr>
<td>Pin No.</td>
<td>Dip/Quip</td>
<td>Flat</td>
<td>Symbol</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>43</td>
<td>37</td>
<td>AVCC</td>
<td></td>
<td>Power supply voltage for A/D converter.</td>
</tr>
<tr>
<td>44</td>
<td>38</td>
<td>RD</td>
<td></td>
<td>(Three-state output, active low) RD is used as a strobe to gate data from external devices onto the data bus. RD goes high during Reset.</td>
</tr>
<tr>
<td>45</td>
<td>39</td>
<td>WR</td>
<td></td>
<td>(Three-state output, active low) WR, when active, indicates that the data bus holds valid data. Used as a strobe signal for external memory or I/O write operations. WR goes high during Reset.</td>
</tr>
<tr>
<td>46</td>
<td>40</td>
<td>ALE</td>
<td></td>
<td>The strobe signal is for latching the address signal to the output from PD7–PD0 when accessing external expansion memory.</td>
</tr>
<tr>
<td>47-54</td>
<td>41-48</td>
<td>PF0-PF7</td>
<td></td>
<td>Port F: Address Bus: (Three-state input/output) 8-bit programmable I/O port. Each line configurable independently as an input or output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Port D: 8-bit Address Bus: When external programmable expansion memory is used, multiplexed I/O port. address/data bus can be selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This byte can be designated as either input or output.</td>
</tr>
<tr>
<td>55-62</td>
<td>49-56</td>
<td>PD0-PD7</td>
<td></td>
<td>Applying a low-level input to this pin stops the oscillator, and puts the device in a low-power-consumption mode.</td>
</tr>
<tr>
<td>63</td>
<td>57</td>
<td>STOP</td>
<td></td>
<td>+5 V power supply</td>
</tr>
<tr>
<td>64</td>
<td>58</td>
<td>VCC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
uPD78C18/78C11
Block Diagram
Advance Product Information
uPD78C10/78C11
Functional Index

Memory Map

The uPD78C11 can directly address up to 64k-bytes of memory. Except for the on-chip ROM (0-4095) and RAM (65280-65535), any memory location can be used as ROM or RAM. The memory map, figure 1, defines the 0 to 64K-byte memory space for the uPD78C11.
Standby Function

The 78C10/78C11 has two standby modes: HALT and STOP.

The HALT mode reduces power consumption to less than 50% of normal operating requirements, while maintaining the contents of on-chip registers, RAM, and control status. The system clock and on-board peripherals continue to operate, but the CPU stops executing instructions. The HALT mode is initiated by executing the HLT instruction. The HALT mode can be released by any non-masked interrupt or by RESET.

The STOP mode reduces power consumption to less than one tenth of 1% of normal operating requirements. There are two types of the STOP mode.

Type A is initiated by executing a STOP instruction. If VCC is maintained within the operating range (2.5 to 6.0 V), on-board RAM and CPU register contents are saved. If VCC is held above 2.0 V (but less than 2.5 V), only on-board RAM is saved. The oscillator is stopped. The STOP mode can be released by an input on NMI or RESET. The user can program oscillator stabilization time via Timer 1. By checking the standby flag (SB), the user can determine whether the processor has been in the Standby mode.

Type B is initiated by inputting a low level on the STOP input. Only RAM contents are saved, not the CPU register contents. The oscillator is stopped. The STOP mode is released by raising STOP to a high level. The oscillator stabilization time is fixed at 65 ms; 65 ms after STOP is raised, instruction execution will begin at location 0. You can increase the stabilization time by holding RESET low for the required time period.
Standby Conditions

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>HALT MODE</th>
<th>STOP MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>By Instruction</td>
<td>By STOP (Input)</td>
</tr>
<tr>
<td>Oscillator</td>
<td>Run</td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>Internal System Clock</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>PA, PB, PC</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>PD Single-chip Mode</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Expand Mode</td>
<td>HIGH-Z</td>
<td>HIGH-Z</td>
</tr>
<tr>
<td>FF Single-chip Mode</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Address Bus</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Expand Mode</td>
<td>(Next Addr)</td>
<td>(Next Addr)</td>
</tr>
<tr>
<td>Mode Port</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td></td>
<td>(Port Output)</td>
<td>(Port Output)</td>
</tr>
<tr>
<td>ALE, RD, WR</td>
<td>HIGH</td>
<td>HIGH</td>
</tr>
<tr>
<td>On-chip RAM</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Memory Mapping</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Register (MM)</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Program Counter (PC)</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Stack Pointer (SP)</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>General Register</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Program Status Word (PSW)</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>Timer Upcounter</td>
<td>Run</td>
<td>Undefined</td>
</tr>
<tr>
<td>Timer Mode Register (TMM)</td>
<td>Hold</td>
<td>Active</td>
</tr>
<tr>
<td>TM0, TM1</td>
<td>Hold</td>
<td>Inactive</td>
</tr>
<tr>
<td>Interrupt Control Circuit</td>
<td>Run</td>
<td>Stop</td>
</tr>
<tr>
<td>Pending Interrupts (INTFX)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>Interrupt Mask Register</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>NMI (Input)</td>
<td>Hold</td>
<td>Active</td>
</tr>
<tr>
<td>INT1, INT2, (Input)</td>
<td>Hold</td>
<td>Inactive</td>
</tr>
<tr>
<td>Timer/Event Counter Circuit</td>
<td>Run</td>
<td>Stop</td>
</tr>
<tr>
<td>Mode Register (ECM, ETMM)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>ECNT, ETM0, ETM1</td>
<td>Run</td>
<td>Hold</td>
</tr>
<tr>
<td>Serial Interface Circuit</td>
<td>Run</td>
<td>Stop</td>
</tr>
<tr>
<td>Mode Register (SMB, SML)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>RxB, TxB Serial Registers</td>
<td>Run</td>
<td>Undefined</td>
</tr>
<tr>
<td>A/D Converter Circuit</td>
<td>Run</td>
<td>Stop</td>
</tr>
<tr>
<td>Mode Register (ANM)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>CR0, CR1, CR2, CR3</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>STANDBY Flag (SB)</td>
<td>Hold</td>
<td>Hold</td>
</tr>
<tr>
<td>STOP Input</td>
<td>Hold</td>
<td>Active</td>
</tr>
<tr>
<td>Zero Cross Mode Reg. (ECM)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>Text Flags (Except SB)</td>
<td>Hold</td>
<td>Undefined</td>
</tr>
<tr>
<td>Reset Input</td>
<td>Active</td>
<td>Active</td>
</tr>
</tbody>
</table>

Note: *1: During Stop Mode : 011000xxB
After Release : 11111111B
APPENDIX B

HARDWARE SCHEMATICS
BLOCK DIAGRAM
SWITCHES, LED'S

SERIAL COMMUNICATION SUBSYSTEM
RESET CIRCUIT AND CLOCK CIRCUIT

ANALOG REFERENCE VOLTAGE CIRCUIT

POWER SUPPLIES

VOLTAGE CONVERSION CIRCUIT
PROGRAM AND MEMORY SPACE SUBSYSTEM
D/A BUFFER SUBSYSTEM

A/D BUFFER SUBSYSTEM
PARTS LIST AND BOARD LAYOUT

- **Uo** - uPD 78C10: NEC 8-bit Microcomputer
- **U1** - 74C244: Latch Buffer
- **U2** - 27C32: 4K x 8 EPROM
- **U3** - NM616: 8K x 8 static RAM
- **U4** - MAX232: RS232 Serial Port
- **U5, U6** - MAX7660: Monolithic Voltage Converter
- **U7, U6**, **-** - LF444: Quad Operational Amplifier

- **M1-Ma** - IRFD110: HEXFET
- **J1.0-J1.7** - 16-PIN: Extender Board Connector
- **J2.1-J2.18** - 18-PIN: I/O Patch Panel
- **J3** - 3-Conductor: RS232 Connector
- **D1-D6** - IN8666: Diode
- **Xo** - 11.059 MHz: Crystal

- **S0-Sy** - 8-STSP: Dip Switch
- **Sx-Sw** - 2-TFDP: 3-Position, Normal Off Switch
- **Sx-A** - 1-AAA: 2-Position, Normal Off Switch
- **Lx-Ly** - LED's
- **B0** - 3V Lithium Battery Holder

- **R0-R7** - 5.1 K
- **R8-R10** - 100 K
- **R12** - 510 K
- **R13** - 220 K
- **R14** - 100 K
- **R15** - 15 K
- **R17** - 20 K: Trim pot.
- **R18-R20** - 47 K

- **C1, C2** - 10 pf
- **C3, C4** - 1 uF
- **C5, C6** - 10 uF
- **C7** - .1 uF
- **C8-C16** - .05 uF
- (Bypass Cap. 's)
NMC6164/6164L 8192 x 8-Bit Static RAM

General Description

The NMC6164/6164L is a 8192-word by 8-bit, new-generation static RAM. It is fabricated with National's proprietary microCMOS double-poly silicon technology which combines high performance and high density with low power consumption and excellent reliability.

The NMC6164/6164L operates with a single 5V power supply with ±10% tolerance. Additional battery back-up operation is available (L version) for data retention down to 2V, with low standby current.

Packaging is in standard 28-pin DIP and is available in both plastic and CERDIP.

In addition to the inputs and outputs being TTL compatible, the outputs are also CMOS compatible, in that capacitive loads are driven to VCC or VSS.

Features

- Single power supply: 5V ± 10%
- Fast access time 100 ns/120 ns/150 ns max
- Equal access and cycle times
- Completely static RAM: no clock or timing strobe required
- Low standby power and low power operation
  - Standby: 10 µW, typical
  - Operation: 15 mW/MHz, typical
- Battery back-up operation available (L version) with data retention supply voltage: 2V-5.5V
- Common data input and output, TRI-STATE* output
- TTL compatible: all inputs and outputs
- CMOS compatible: outputs drive capacitive loads to VCC or VSS
- Standard 28-pin package configuration

Block and Connection Diagrams

Truth Table

<table>
<thead>
<tr>
<th>Mode</th>
<th>WE</th>
<th>CS1</th>
<th>CS2</th>
<th>OE</th>
<th>I/O</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Selected</td>
<td></td>
<td>H</td>
<td>H</td>
<td></td>
<td>HI-Z</td>
<td>I/O, I/O</td>
</tr>
<tr>
<td>(Power Down)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Disabled</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>HI-Z</td>
<td>I/O, I/O</td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>DO</td>
<td>I/O, I/O</td>
</tr>
<tr>
<td>Write</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td></td>
<td>DOUT</td>
<td>I/O, I/O</td>
</tr>
</tbody>
</table>

*Don't care (H or L) H = Logic HIGH Level L = Logic LOW Level

Order Number NMC6164J (NMC6164LJ)
NS Package Number 828A
Order Number NMC6164N (NMC6164LN)
NS Package Number N28A
**TABLE I. Mode Selection**

<table>
<thead>
<tr>
<th>Mode</th>
<th>CE (20)</th>
<th>OE (22)</th>
<th>V_{PP} (11)</th>
<th>V_{CC} (28)</th>
<th>Outputs (11-13, 15-19)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>V_{IL}</td>
<td>V_{IL}</td>
<td>V_{CC}</td>
<td>V_{CC}</td>
<td>D_{OUT}</td>
</tr>
<tr>
<td>Output Disable</td>
<td>V_{IL}</td>
<td>V_{IH}</td>
<td>V_{CC}</td>
<td>V_{CC}</td>
<td>High Z</td>
</tr>
<tr>
<td>Standby</td>
<td>V_{IH}</td>
<td>X</td>
<td>V_{CC}</td>
<td>V_{CC}</td>
<td>High Z</td>
</tr>
<tr>
<td>Verify</td>
<td>V_{IH}</td>
<td>V_{IL}</td>
<td>V_{PP}</td>
<td>V_{CC}</td>
<td>D_{OUT}</td>
</tr>
<tr>
<td>Program Inhibit</td>
<td>V_{IH}</td>
<td>V_{IH}</td>
<td>V_{PP}</td>
<td>V_{CC}</td>
<td>High Z</td>
</tr>
</tbody>
</table>

Note: X can be V_{IH} or V_{IL}.

**Connection Diagram**

Note: National's socket compatible EPROM pin configurations are shown in the blocks apparent to the NM27C256 pins.

NS Package Number J24A-Q
APPENDIX C

SOFTWARE LISTINGS
Monitor Program
A: 10H
F: 1FH
COM

### EXEC

**MARCH 30, 1987**

**APRIL 2, 1987**

**FT7.PGM**

********************************************************************
**MENU/LOAD/CLEAR/HALT/PRINT/RUN/EXAMINE**
**FPG FOR 7801 MICRO IN ARM PROSTHESIS**
********************************************************************

* WITH CALL ROUTINES
* WITH VECTORED INTERRUPTS
* WITH ZEROING RAMS
* WITH A CHANGED LOADING ROUTINE.
* WITH A DELAY ROUTINE.

---

**POWER-UP AND RESET**

**STICK:** EQU 0FFFH
**INITRAM:** EQU 1000H

**BASE:** EQU 2000H
**LF:** EQU 0AH
**CR:** EQU 0DH
**CRLF:** EQU 3000H
**BSSP:** EQU 68H
**BS:** EQU 08H

**RAMST:** EQU 0FFFH
**RAMEND:** EQU 0FFFH
**RAMN:** EQU RAMEND-RAMST+1
**RAM?:** EQU RAMN / 2

---

**INTERRUPT JUMP TABLE**

<table>
<thead>
<tr>
<th>ORG</th>
<th>JMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000H</td>
<td>0000H</td>
</tr>
<tr>
<td>0001H</td>
<td>0001H</td>
</tr>
<tr>
<td>0002H</td>
<td>0002H</td>
</tr>
<tr>
<td>0003H</td>
<td>0003H</td>
</tr>
<tr>
<td>0004H</td>
<td>0004H</td>
</tr>
<tr>
<td>0005H</td>
<td>0005H</td>
</tr>
<tr>
<td>0006H</td>
<td>0006H</td>
</tr>
<tr>
<td>0007H</td>
<td>0007H</td>
</tr>
<tr>
<td>0008H</td>
<td>0008H</td>
</tr>
<tr>
<td>0009H</td>
<td>0009H</td>
</tr>
<tr>
<td>000AH</td>
<td>000AH</td>
</tr>
<tr>
<td>000BH</td>
<td>000BH</td>
</tr>
<tr>
<td>000CH</td>
<td>000CH</td>
</tr>
<tr>
<td>000DH</td>
<td>000DH</td>
</tr>
<tr>
<td>000EH</td>
<td>000EH</td>
</tr>
<tr>
<td>000FH</td>
<td>000FH</td>
</tr>
<tr>
<td>0010H</td>
<td>0010H</td>
</tr>
<tr>
<td>0011H</td>
<td>0011H</td>
</tr>
<tr>
<td>0012H</td>
<td>0012H</td>
</tr>
<tr>
<td>0013H</td>
<td>0013H</td>
</tr>
<tr>
<td>0014H</td>
<td>0014H</td>
</tr>
<tr>
<td>0015H</td>
<td>0015H</td>
</tr>
<tr>
<td>0016H</td>
<td>0016H</td>
</tr>
<tr>
<td>0017H</td>
<td>0017H</td>
</tr>
<tr>
<td>0018H</td>
<td>0018H</td>
</tr>
<tr>
<td>0019H</td>
<td>0019H</td>
</tr>
<tr>
<td>001AH</td>
<td>001AH</td>
</tr>
<tr>
<td>001BH</td>
<td>001BH</td>
</tr>
<tr>
<td>001CH</td>
<td>001CH</td>
</tr>
<tr>
<td>001DH</td>
<td>001DH</td>
</tr>
<tr>
<td>001EH</td>
<td>001EH</td>
</tr>
<tr>
<td>001FH</td>
<td>001FH</td>
</tr>
<tr>
<td>0020H</td>
<td>0020H</td>
</tr>
</tbody>
</table>

**POWERUP:**

**LJ:** SP, STCY+1
**XFA:** A, A
**MAY:** M4, A
**MIN:** MI, A
**MT:** A, M1, M4

---

**SOFTWARE RESET:**

**EXTERNAL SHUTDOWN:**

**INTEGRATED TIMER:**

**INTERNAL TIMING EVENTS:**

**INTERNAL A/D:**

**INTERNAL RXD OR TXD:**

---

**PROGRAM END**
INITIALIZE THE INTERNAL RAM *********************************************************************

; INITIALIZE RAM BY CLEARING IT. (I.E. FILLING IT WITH "0"S)

LDL #0, RAM1
LDL #4, RAM1+1
MOV B, RAM1+2
; STORE STARTS AT RAM'S FIRST LOCATION
; FILL REGION TO THE ZEROS IN THE RAM
; REGISTER B IS THE COUNTER.

STAL B, MEM
; SET IT STORE, DUMMY INC. ADDRESS
DBF B
; DEC'REMEN'T COUNTER
JRF #MEM
; LOOP THROUGH UNTIL B=0 (UNTIL CARRY FLAG)
; INITIALIZE PORT C AT 9600 BAUD.

INITIALIZE PORT C AS INPUT ********************************************************************

CPU1: MOV A, 7
MOV M, M, 0
MOV A, M
MOV PORTC, 0
; PORT C FOR RXD, TXD & SCK (BAUD TIMER PIN)
; INITIALIZE MODE C CONTROL REGISTER (PORT C MODE)
; LEAVING 5 MSR AS DISABLED OUTPUT CHANNELS.
; INITIALIZE MODE C REGISTER (RXD AS INPUT)

USE TIMECRAND XTAL TO GENERATE 1200 BAUD (SEE TABLE IN MANUAL --
; SERIAL MODE SECTION)

MOV A, 3
STOR 11, ASK, MH7 XTL
MOV PORTA, A
; SET TIM FOR THE COUNTER CLOCK
; FRACTIONS OUT OF REGISTER EXCEPT TIMER1.

INITIALIZE SERIAL PORT REGISTER -- ALL 0's Data.
; 2 STOP BITS, 8-BIT MODE, 13-13 OFF RST, NO PARITY CHECK.
; (SEE MANUAL FOR FORMULAS
; --SERIAL MODE SECTION.)

MOV A, 0
; WHILE THE RECEIVED, TRANSMIT AND SELECT
MOV A, 0
; THE INTERNAL BIT "TO" OUTPUT FOR /SCK.
MOV A, 0
; 0-SET OUT STATE ABOVE. (0-BIT, ETC.)

*****************************************************************************
PRINT BEGINNING SUBROUTINE
*****************************************************************************

CALL PRINT

PRINT: MOV A, 0
; MOVE DOWN 4 SPACES
MOV B, 1
LDA FLOW
MOV A, 72
; MOVE ACROSS 12 SPACES
MOV B, 10
CALL MUP
LDI H, 0
; PRINT MENU
LDL PND

LDA H, PND
; PRINT ENTER MESSAGE
CALL PND
CALL PND
CALL PND
CALL PND
CALL PND
CALL INCH
; DELAY BEFORE READING MESSAGE.
CALL INCH
; WAIT FOR ENTRY
HIT A, "A"
JMP PEND
; IF "C" GOTO CLEAR RAM SUBROUTINE
NEI A, "H"
JMP PEND
; IF "H" GOTO HALT SUBROUTINE
NEI A, "E"
JMP PEND
; IF "F" GOTO LYRINE SUBROUTINE.
JMP PEND
; IF "F" GOTO LOAD PROGRAM SUBROUTINE
NEI A, "P"
JMP PEND
; IF "L" GOTO PRINT ROUTINE.
NEI A, "P"
JMP PEND
; IF "F" GOTO PRINT ROUTINE.
HALT ?BC10 subroutine

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.

**************

Continue clearing.
**DOWNLOADING HEXFILL SUBROUTINE**

**L0APP:**

LXI A, 1234H  ; PRINT LOADING MESSAGE.
CALL P000H    ; STORE RETURN VALUE FOR LOADER ROUTINE.
SHLD OFF5H    ; SET LOWEST INTERNAL RAM VALUE.

**L0APP1:**

NOP
CALL 1234H    ; WAIT FOR FIRST CHARACTER.
EQU 1234H     ; FIRST CHARACTER SHOULD BE A '1'.
JMP 2000H     ; ELSE ERROR.
MOV P000H     ; PUT IT IN THE LED'S.
MOV 1234H     ; PHF=F. ERR FOR REST OF RECORD.
CLC            ; CLEAR CHECKSUM (REGISTER B).
CLC            ; CLEAR BYTE COUNT (REGISTER C).
CALL INBYTE   ; GET BYTE FROM NEXT 2 CHARACTERS.
NL 1           ; IF NOT '1' THEN CONTINUE.
JMP LASTREC   ; ELSE TEST FOR LAST REC.
MOV 1234H     ; SET RECORD LENGTH.
CLD            ; SET HIGH BYTE OF ADDRESS.
LXI H, 1234H   ; MOVE IT INTO HIGH BYTE.
MOV 1234H     ; GET LOW BYTE OF ADDRESS.
LXI L, 1234H   ; MOVE IT INTO LOW BYTE.
CALL INBYTE   ; NEXT BYTE MUST BE '9'.
EQU 1234H     ; ELSE ERROR.
JMP LASTREC   ; ELSE CONTINUE.

**L0APP2:**

CALL 1234H    ; SET BASE OF INTERNAL RAM, INCL. ADDR.
MOV 1234H     ; BYTE LENGTH.
MOV 1234H     ; NOT 1234H.
JMP 1234H     ; NOT THE NEXT BYTE.
CALL 1234H    ; GET THE 1234H.
LXI H, 1234H   ; AT THE END CONTINUE.
LXI L, 1234H   ; AT THE END CONTINUE.
JMP 1234H     ; AT THE END CONTINUE.

**L0APP3:**

LXI H, 1234H   ; AT THE END CONTINUE.
LXI L, 1234H   ; AT THE END CONTINUE.
JMP 1234H     ; AT THE END CONTINUE.

**L0APP4:**

LXI H, 1234H   ; AT THE END CONTINUE.
LXI L, 1234H   ; AT THE END CONTINUE.
JMP 1234H     ; AT THE END CONTINUE.

**L0APP5:**

LXI H, 1234H   ; AT THE END CONTINUE.
LXI L, 1234H   ; AT THE END CONTINUE.
JMP 1234H     ; AT THE END CONTINUE.
**PRINT MEMORY BYTES SUBROUTINE**

*PRINT "PRINT" MESSAGE*
CALL FM56
LDX H,CPPM
CALL FM56
LDI H,PPRI
SHD 64H

*PRINT "ENTER" MESSAGE*
CALL FM56
LDX H,CPPM
CALL FM56
LDI H,200H
DEP La,1
PUSH DE
PUSH H
PUSH L

*SET UP V FOR COMPARISON*

*SET UP D FOR LOOPING (20 BYTES PER LINE)*

*CLEAR MEMORY BYT*

*PRINT A HEX BYTE*

*DECREMENT COUNTER*

*IF E = 0 THEN CONTINUE CLEARING*

*ELSE JMP TO PPR2*

*DECEDEMENT LINE PRINTING COUNTER.*

*IF E = 0 JMP PPR4,*

*CONTINUE CLEARING.*

*AFTER 16 BYTES WRITE A CRLF*

*AND CONTINUE PRINTING.*

*WAIT FOR ANY CHARACTER BEFORE*

*RETURNING TO MENU.*
LETI  A, 44H  ;RETURN RETURN VALUE FOR LOADER

LETI  A, 88H  ;PRINT FUN ENTER MESSAGE

CALL  POS63  ;GET ADDRESS

SKM:   CALL  INQWRD  ;GET ADDRESS
CALL  INCH
EOT  A, "H"
JMP  LOADER
CALL  INCH
EOT  A, 94H
JMP  LOADER

JMP  TO ADDRESS

*INCH*
; THIS SUBROUTINE WRITES THE ASCII VALUE RECEIVED TO THE
; TRANSMIT CHANNEL OF PORT C WHEN THAT CHARACTER IS RECEIVED FROM THE MONITOR.

INCH:  MOV  PS, 0 ;TEST RECEIVE INTERRUPT FLAG.
JR  XS
MOV  P, 0 ;JUMP THEY IF NOT SET, WAIT FOR A CHARACTER.
MOV  P, A  ;GET CHARACTER.
MOV  P, 0  ;PUT IT OUT ON PORT B LED'S.

INH:     MOV  PS, 1 ;TEST XMIT FLAG
JR  TXN
MOV  TXN, A
BRET ;ONCE CHARACTER HAS BEEN ECHOED, RETURN.

**INHE**
; IT IS RECEIVING AN ASCII VALUE CHARACTER IN REGISTER A.
; IT TESTS TO SEE WHETHER IT IS A HEX CHARACTER OR NOT.
; IF IT IS NOT A HEX CHARACTER THEN GO TO ERROR ROUTINE.
; IF IT IS A HEX CHARACTER CONVERT IT TO ITS HEX DECIMAL VALUE.

; AND LEAVE IT IN REGISTER A.
INHE:   CALL  INCH  ;GET CHARACTER FROM SERIAL PORT
EOT  A, 0  ;SPECIAL ROUTINE FOR CR
MOVT  L, 0  ;ENTER IF A-

MOVH  A, "9"  ;SUBTRACT THE VALUE OF A FROM CHARACTER
JMP  LOADER  ;IF 60 TO 90 CHAR'S (CH. ASCII<40) ERROR.
G:     MOV  A, "7"  ;IF GREATER THAN 7 THEN CONTINUE.
; GOTO 70H WITH RETURN VALUE.

MOVH  A, "0"  ;SUBTRACT THE VALUE OF A FROM CHARACTER
JMP  LOADER  ;IF 60 TO 90 CHAR'S (CH. ASCII>65) ERROR.
L:     MOV  A, 1  ;IF A- THEN CONTINUE (ADD 10 TO VALUE)
JMP  LOADER  ;IF GREATER THAN ASCII THEN ERROR.
G:     ADD 10  ;AD E TO VALUE FOR A-F.
JMP  A

**INHE**
; THIS ROUTINE GETS 7 HEX CHARACTERS, TURNS THEM INTO A BYTE
; AND RETURNS THEM IN REGISTER A, ADDS THEM TO REGISTER B (CHECKSUM)
; IT USES REGISTER E.

INHEE:  CALL  INHE  ;GET A HEX CHARACTER
MOV  P, A  ;PUT IT ON THE LED'S
SUI  A  ;SWITCH IT INTO HIGH HALF OF BYTE
SUL  A
SUI  A
SUL  A
MOV  P, A  ;SAVE IT IN REGISTER E
INT  INHEE  ;SWITCH IT INTO HIGH HALF OF BYTE
ADD B, H
MOV PH,A
REI

**** INITIAL****

THIS ROUTINE GETS 2 BYTES AND PLACES THEM IN THE EA
FIRST BYTE goE IN R1N, SECOND GOES IN R1AL

INDIR: CALL INBYTE
MOV EA, A
CALL INBYTE
MOV EA, A
REI

**** INITIAL ****

THIS SUBROUTINE INPUTS VALUES IN A CERTAIN FORMAT
HEXWORD "H", "H" HEXWORD "H", "H" OR "H", "H"
HEXWORD -- AL, HEXWORD -- EA

INPALS: CALL INWORD
MOV H, EA
CALL INCH
EQU A, "H"
JMP LOADERR
CALL INCH
EXIT: CALL INWORD
LOAD ERR: CALL INCH
EXIT: CALL INCH
EQU A, "H"
JMP LOADERR
CALL INCH
EXIT: CALL INCH
EQU A, "H"
JMP LOADERR
REI

**** DELAY ROUTINE ****

DELAY: DT 1, INH FFH
EXIT: INH H, I

**** \******

MESSAGES FOR SYSTEM

**************

MESSAGES:

DW CALL
CRLF
"PROSTHESIS MENU"
CRLF
DW 'OPTIONS:
CRLF
DW 'C - CLEAR RAM
CRLF
DW 'H - HALT
CRLF
DW 'L - LOAD HI FILE RAM
CRLF
DW 'P - PRINT RAM
ENTMSG:  DW  CR LF
          DW  "ENTER OPTIMA:"
          DW  CR LF
          DB  0
LDMSG:  DW  CR LF
          DW  "LOAD DATA FILE:"
          DW  CR LF
          DB  0
LDERR:  DW  CR LF
          DW  "LINE NOT ACCEPTED"
          DW  CR LF
          DB  0
CLPRMSR: DW  CR LF
          DW  "CLEAR"
          DW  0
CLRMN:  DW  DR  "NUM RAM BYTES, STARTING AT:"
          DW  CR LF
          DW  "(NUM START AT 2000H, 2000H BYTES LONG)"
          DW  CR LF
          DW  "USE ONLY 16, EX: 2000H, 2000H CR"
          DW  0
LMN:    DW  CR LF
          DW  "ENTER VALUES:""
Viscous Damping Program
** ECHO TEST PGM FOR 78C16 MICRO ON A/F PROSTHESIS

** WITH A CHANGE FOR CALL ROUTINES
** WITH A CHANGE FOR VECTORED INTERRUPTS
** WITH A CHANGE FOR ZEROING RAMS.
** WITH A CHANGE FOR A/D CONVERSION CHANNELs
** WITH A CHANGE FOR CONTROLLING THE PULSE WIDTH OF AN OUTPUT SIGNAL.
** A BETTER CONTROLLING SCHEME.
** GIVES VERSION.

** POWER-UP AND RESET:

SICK EQU $0EFFH  \( \text{CHANNEL FOR CALL ENABLING.} \)
INTRAM EQU $1000H  \( \text{BIT SELECTS INTERNAL RAM.} \)
BASE EQU $2000H  \( \text{BASE ADDRESS IN RAM FOR VECTORED INTERRUPTS.} \)
LJ EQU $04H  \( \text{LIFO FIELD.} \)
LE EQU $0DH  \( \text{CALLAGE RETURN.} \)
RAM$1 EQU $0EFFH  \( \text{PHYSICAL/ HARDWARE LOCATIONS FOR STATIC RAM} \)
RAM$0 EQU $0EFFH  \( \text{ARE FROM 2000H TO 4000H BUT THERE IS} \)
RAM(?:i) EQU $0eff$+$0(i)  \( \text{INTERNAL RAM IN LOCATIONS 0$EFFH$T 0$EFFH.}\)
RAM$2 EQU $0ef$+2  \( \text{SO TO CLEAR RAM MUST SPECIFY ALL LOCATIONS.}\)
\( \text{SEE INITIALIZ RAM SUBROUTINE.}\)

** INITIALIZATION VARIABLES FOR THE SQUARE WAVE GENERATOR:

PERIOD EQU $0400H  \( \text{PERIOD OF ABOUT 1000US=IMILLISECOND} \)
PERIOD2 EQU PERIOD  \( \text{PERIOD OF ABOUT 1000US=IMILLISECOND} \)
PERIOD1 EQU PERIOD2 / 2  \( \text{INITIAL DUTY CYCLE OF 50 PERCENT.}\)

** INTERRUPT JUMP TABLE

<table>
<thead>
<tr>
<th>JMP</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000AH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>000DH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>1000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>2000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>3000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>4000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>5000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>6000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
</tbody>
</table>

** INTERRUPT JUMP TABLE

<table>
<thead>
<tr>
<th>JMP</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000AH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>000DH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>1000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>2000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>3000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>4000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>5000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>6000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
</tbody>
</table>

** INTERRUPT JUMP TABLE

<table>
<thead>
<tr>
<th>JMP</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000AH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>000DH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>1000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>2000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>3000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>4000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>5000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>6000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
</tbody>
</table>

** INTERRUPT JUMP TABLE

<table>
<thead>
<tr>
<th>JMP</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000AH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>000DH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>1000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>2000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>3000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>4000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>5000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>6000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
</tbody>
</table>

** INTERRUPT JUMP TABLE

<table>
<thead>
<tr>
<th>JMP</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000AH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>000DH</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>1000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>2000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>3000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>4000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>5000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
<tr>
<td>6000H</td>
<td>INITIAL HELP JUMP TO POWER UP.</td>
</tr>
</tbody>
</table>
; INITIALIZE THE RAM BY CLEARING IT (I.E., FILLING IT WITH 0'S)

LXI H, RAM1
LXI EA, A
MVI H, RAM2
ZMEN: STEAK M++
DIF B
JMP ZHP

; INITIALIZE THE RAM WITH INITIAL VALUES

LXI H, PRSH
MOV A, 0H
MOV PB, A
JMP VISC

; INITIALIZE PORT C AS INPUT

CPURQ: MOV A, M
MOV MCC, A
MOV A, 1
MOV RC, A

; USE TIMERS AND EXTERNAL TO GENERATE 9600 BAUD (SEE TABLE IN MANUAL)

MVI A, 0H
MOV TRH, A
MVI TMH, 0001H
SET TMH-FLAG FOR COUNTER CLOCK
ENABLES ALL OF REGISTER EXCEPT TIMER1.

; INITIALIZE SERIAL MODE REGISTER -- BAUD RATE.
; STOP BITS, ASYNCH MODE, 11.0592 MHZ, NO PARITY CHECK.
; (SEE MANUAL FOR FORMULAS)

; SERIAL MODE SECTION.

MVI A, 00EH
SMH 4CH
; ENABLE THE FIFO, TRANSMIT AND SELECT THE INTERNAL ( IF "IO" OUTPUT FOR SCK.
MVI A, 00EH
SMI 5EH
; SLEEP AS STATED ABOVE, (8-HIT, ETC.)
REI
; GO TO INITIALIZE.

; INITIALIZE THE A TO D CONVERTER PORT IIC -- SCAN MODE, READING CHANNELS
; A+/- INTO REGISTERS CRIE IN SUCCESSIVELY, INITIATIONS FOR AN 11 MHZ CLOCK.

ATOD: MOV A, 00EH
REI

; INITIALIZE PROGRAMMABLE SQUARE WAVES WITH A PROGRAMMABLE PHASE OFFSET.

SUBAY: MOV A, 00H
CMC RC FOR R6D & R7D COUNTER
MOV A, 07H
SFI CDI AND CDI OF PORT C TO THE CONTROL MODE.
MOV M1, A

; INITIAL SW: MOV CD1=1 AND CD0=0 AND SELECT INVERSION FOR LOW AND CD1

LOCTR: LVI EA, PERM02
DREV ETM1, EN
LXI EA, PHAS1
DREV ETMB, EA

; SUBAY: MOV A, 00H
MOV ETIN, EA
REI

; CLEAR ECNT WHEN = ETM1, CD0 OUTPUT CHANGE AT
; ETINT = ETM1, AND CD1 OUTPUT CHANGE AT
; NOW FOR A LOOP WHICH WRITES THE ASCII VALUE RECEIVED TO THE
; TRANSMIT CHANNEL OF PORT 1. WHEN A CHARACTER IS RECEIVED FROM THE MONITOR,
; RXDI:
; SF  FSH  ; TEST RECEIVE INTERRUPT FLAG.
; JR RXDI  ; JUMP BACK IF NOT SET, WAIT FOR A CHARACTER.
; MOV A, RXB  ; GET CHARACTER.
; MOV PB, A  ; PUT IT OUT ON PORT B LED'S.
; TXDI:
; SF  FSH  ; TEST PUT FLAG
; JR TXDI  ; WAIT UNTIL ITS SET (READY TO TXD AGAIN)
; ; IF AN A IS NOT SENT THEN CONTINUE.
; ; ELSE JUMP TO A/D CONVERSION ROUTINE.
; ; IF NOT A N THEN CONTINUE.
; CALL FDOWN  ; ELSE DECREASE THE SQ. WAVE FREQUENCY.
; CALL FUP  ; ELSE INCREASE THE SQ. WAVE FREQUENCY.
; JME VIOL  ; ELSE INITIATE VISCOUS DAMPING.
; ; FLOOR INFINITELY AS AN ECHO/DISPAY PGM.
; ; IF A D FLAG
; MOV A, CR  ; LOOP FOREVER UNTIL A CONVERSION (FLAG IS SET)
; DMOV EA, H  ; FETCH A/D CONVERSION READING.
; DMOV EA, R  ; PUT IT OUT ON PORT A LED'S (POSITION)
; MOV EA, CR2  ; FETCH A/D CONVERSION READING.
; DMOV EA, R  ; PUT IT OUT ON PORT A LED'S (VELOCITY)
; DMOV EA, H  ; LOOP INFINITELY BACK UNTIL RESET
; ; FDOWN:
; CALL DUTY  ; DISPLAY DUTY CYCLE
; DMOV EA, H  ; SET UP TO ADD MORE TO THE PERIOD
; INCR EA, H  ; AND 4 COUNTS (US) TO PERIOD
; INCR EA, R  ; AND SAVING IT AGAIN IN HL.
; MOV EA, H  ; THEN BRKE CHARACTER.
; ; FUP:
; CALL DUTY  ; DISPLAY DUTY CYCLE
; DMOV EA, H  ; SET THE CURRENT PULSE WIDTH
; INCR EA, H  ; ADD SET UP TO SUBTRACT FROM PERIOD/2
; DMOV EA, R  ; SUBTRACT 4 COUNTS/US FROM PERIOD/2
; INCR EA, R  ; AND SAVE IT DOWN IN HL.
; MOV EA, H  ; THEN BRKE CHARACTER.

; BRAKE CONTROLLER ROUTINE: ACCEPTS A VALUE IN HL AND PLACES IT IN
; THE BRAKE CONTROLLING REGISTER (ETM8)
; ; VALUE ranges are FROM 1 TO (PERIOD/2 - 1)
; ; IN THIS VERSION FROM 1 TO 3FFH
; ; BRAKE:
; MVI A, AH  ; CLEAR AND STOP
; MOV EMM, R  ; ECNT COUNTER
; MVI ECM, AH  ; SET C1=1 AND C0=1 SELECT INVERSION
; (SEE MANUAL)
; DMOV EA, H  ; GET THE NEW PULSE WIDTH
; DMOV ETM8, EA  ; CHANGE THE ETM8 AND THUS THE SQ.WV.
; MVI A, OFCH  ; INSET THE ECNT COUNTER
; MOV ETM8, A  ; TO ITS ORIGINAL STATUS.
DIVIDE IT BY 4 TO GET A RANGE FROM 0 TO 256
FROM THE ORIGINAL RANGE OF 0 TO 1024
GET THE RESULT FROM THE LOW HALF OF EA
DISPLAY IT ON THE LED's

TEST INTERRUPT FLAG.
JUMP TO ANALOG IF NOT SET,
GET CHARACTER,
IF NOT AN 'R' THEN CONTINUE,
ELSE RESET IF AN 'R'

TEST A/D FLAG
LOOP TO CHARACTER LOOP IF FLAG IS NOT SET

STEP A/D CONVERSION READING (VELOCITY)
IF A = 128 THEN CONTINUE
ELSE JUMP TO V1
SUBTRACT 128 TO GET ABS. VELOCITY
AND SKIP OVER V1 SECTION,
IF A <= 128 THEN SUBTRACT 128
TO GET ABSOLUTE VELOCITY,
ADD 40 TO MULTI BY 8 TO START AT 8140H.

MULTIPLY VELOCITY BY A FACTOR TO
ACHIEVE LINEAR RANGE FROM 320 TO 1K
SUBTRACT THIS FROM 1K TO GET PROPORTIONAL
DAMPING
SUBTRACT EA, H
MULTIPLY D5 INTO H/4 REGISTER
ADD MODIFIED CHARACTER,
WAIT THAN WAIT 100 A/D CYCLES
WAIT FOR AN EVENT TIMER INT.
BEFORE GETTING THE NEXT A/D READING.

LOOP INFINITELY BACK UNTIL RESET
APPENDIX D

SUPPLIER ACKNOWLEDGEMENTS
Part: μPD78C10 Microcomputer
Supplier: COMPASS TECHNOLOGY
215 Salem Street
Woburn, MA 01810

contact: Steve Miner
tel: (617) 933-3336

Part: Anti-Backlash Screw-nut Assembly
Supplier: KERK MOTION PRODUCTS
1 Kerk Drive
Hollis, NH 03049

contact: Keith or Ken Olsen
tel: (603) 465-7227

Part: 3200 B.TBGH 10mm Double-row Ball Bearing
Supplier: FAG BEARING
118 Hamilton Avenue
Stamford, CN 06904

contact: A. Gunst
tel: (203) 327-1960

Part: MJ-451 1/4" Needle Bearing
Supplier: TORRINGTON BEARING
59 Field Street
Torrington, CN 06790

contact: Geore Rusiecki
tel: (203) 482-9511

Part: P/N B20SF9 S/N 101 Magnetic Particle Brake
Supplier: FORCE LIMITED
2217 Main Street
Santa Monica, CA 90405

contact: Paul Atkins
tel: (213) 392-3021
Part: 5-Pack AA NiCad Batteries  
Supplier: GENERAL ELECTRIC BATTERY  
5 Militia Drive  
Lexington, MA 02173  

contact: Margaret Huelskamp  
tel: (617) 861-6382  

Part: Bourns 6534-S-1-203 Potentiometer  
Supplier: J.E. BOEING COMPANY  
18 Mussey Street  
Lexington, MA 02173  

contact: Marie Hill  
tel: (617) 862-2500  

Part: National Semiconductor 27C32 EPROM  
Supplier: A/D SYSTEMS  
594 Marrett Road  
Lexington, MA 02173  

contact: Barbara Flanagan  
tel: (617) 861-6371