Design, implementation, and integration of a portable memory device into a microcomputer software development system.

David John Helffrich

Follow this and additional works at: https://preserve.lehigh.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
https://preserve.lehigh.edu/etd/1971

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.
DESIGN, IMPLEMENTATION, AND INTEGRATION OF A
PORTABLE MEMORY DEVICE INTO A
MICROCOMPUTER SOFTWARE DEVELOPMENT SYSTEM

by
David John Helffrich

A Thesis
Presented to the Graduate Committee
Of Lehigh University
in Candidacy for the Degree of
Master of Science

in
Computer Science

Lehigh University
1982
This thesis is accepted and approved in partial fulfillment of requirements for the degree of Master of Science.

14 May 1989
(dated)

__________________________
Professor in Charge

__________________________
Chairman of Department
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1. Background</td>
<td>3</td>
</tr>
<tr>
<td>2. Approach</td>
<td>13</td>
</tr>
<tr>
<td>3. Requirements</td>
<td>24</td>
</tr>
<tr>
<td>4. Requirements Evaluation</td>
<td>24</td>
</tr>
<tr>
<td>5. System Specification With Alternatives</td>
<td>25</td>
</tr>
<tr>
<td>6. Gross System Design</td>
<td>32</td>
</tr>
<tr>
<td>7. Cost Performance Evaluation</td>
<td>38</td>
</tr>
<tr>
<td>8. Final System Specification</td>
<td>51</td>
</tr>
<tr>
<td>9. Final System Design</td>
<td>54</td>
</tr>
<tr>
<td>10. System Implementation</td>
<td>80</td>
</tr>
<tr>
<td>11. System Test</td>
<td>83</td>
</tr>
<tr>
<td>12. System Performance</td>
<td>86</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Cost Performance Evaluation Results</td>
<td>49</td>
</tr>
<tr>
<td>Table 2</td>
<td>Memory Evaluation Results</td>
<td>79</td>
</tr>
<tr>
<td>Fig.</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>Software Development Cycle</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Object Code Length Comparison</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4040 Assembly vs. Pascal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Memory Bank Switching</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>Link Loader Data Format</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>Intel Data Format</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>Portable Memory System Diagram</td>
<td>58</td>
</tr>
<tr>
<td>7</td>
<td>64K Byte Memory Array – Dynamic</td>
<td>66</td>
</tr>
<tr>
<td>8</td>
<td>Memory Read Cycle of TMS 9995</td>
<td>58</td>
</tr>
<tr>
<td>9</td>
<td>Dynamic RAM Read Cycle</td>
<td>59</td>
</tr>
<tr>
<td>10</td>
<td>64K Byte Memory Array – Bitwide</td>
<td>61</td>
</tr>
<tr>
<td>11</td>
<td>64K Byte Memory Array – Nibblewide</td>
<td>63</td>
</tr>
<tr>
<td>12</td>
<td>64K Byte Memory Array – Bytewide</td>
<td>66</td>
</tr>
<tr>
<td>13</td>
<td>Memory Board Schematic</td>
<td>67</td>
</tr>
<tr>
<td>14</td>
<td>CMOS RAM Data Retention Mode</td>
<td>73</td>
</tr>
<tr>
<td>15</td>
<td>PM Controller Schematic</td>
<td>76</td>
</tr>
</tbody>
</table>
Abstract

During the software development process for a microcomputer controlled system, the use of EPROM (Erasable Programmable Read Only Memory) for program memory can facilitate the debugging process. The reusability of this device keeps costs down while new software versions are tested on the target system.

In a production facility it is important that time spent debugging be minimized. As systems become more complex the required number of debugging iterations for each system increases. Verifying proper performance of systems having EPROM resident program memory becomes unacceptably time consuming; therefore a more efficient software development process is needed.

The following approach is found to fill the above requirement best: A portable nonvolatile data repository is built which eliminates the need for programming or erasing EPROMS in order to test software. The device accepts program data from the software development minicomputer and preserves the data while the device is being carried to the target machine. During target system operation, it emulates program memory via direct umbilicals to the
target microcomputer's data, address, and control busses.

Within the data repository, fast NMOS technology RAM (Random Access Memory) chips are chosen to emulate volatile scratch pad memory while low power CMOS ram chips supported by battery power emulate the nonvolatile program memory. The organization and pinout of these chips are chosen to ease the process of reconfiguring the device for varying RAM/EPROM memory maps.

System flexibility is further enhanced by using a microcomputer to control the data loading process. The microcomputer is utilized to implement data verification during and after the loading process. Its location within the chassis of the Portable Memory device allows data verification at the emulation site without significantly degrading device portability.

In summary, the Portable Memory is a software development tool capable of interfacing directly to both the development minicomputer and the target microcomputer. The Portable Memory is designed and implemented with special considerations for data integrity and operating environment.
0. Introduction

The Bell and Howell Manufacturing Facility in Phillipsburg, N.J., has a real-time software productivity problem. The complexity and size of the software packages produced keeps increasing, and the time spent debugging these packages grows correspondingly. The development of a method to keep this debugging time under control is described in this paper.

The development process is presented in four phases. The first section (Chapter 1) describes the facility where the Portable Memory was developed. The second section (Chapter 2) discusses the alternatives considered for solving the productivity problem. The third section (Chapters 3 through 9) describes the development of the Portable Memory, and section four (Chapters 10 through 12) explains how the Portable Memory operates.

1. Background

The Phillipsburg division of the Bell and Howell Business Equipment Group located in Phillipsburg, New Jersey, manufactures a type of business machine called an inserter. This machine automates the process of mailing large quantities of letters. This process is important to a variety of large businesses. Because of the diversity of organizations within Phillipsburg's customer
base, mailing problems encountered at Phillipsburg take not one but many forms. The ability of the company to respond with specially designed machines per customer requirement is important to the success of the operation.

1.1 Electronic Control

The envelope stuffing process involves a number of run-time decisions. Functions available on the inserter include conditional feeding of inserts, multiple feeding of inserts, and sorting by zip code or by weight. Each process is triggered by computer generated bar code commands which appear on the user's paper medium. A fiber-optic system on the Phillipsburg machine reads the data and an electronic system decodes the information and implements the functions.

1.2 Microcomputer Control

Until 1975, most Phillipsburg systems were special purpose digital logic designs implemented in a high noise threshold technology. The microcomputer was introduced as an alternative system embodiment at that time.

1.2.1 Benefits of Microcomputer Control

The decision to bring the microcomputer to Phillipsburg was motivated by a desire for cost reduction.
Savings were anticipated due to the microcomputer's ability to serve as a general purpose electronic control device.

Costs have in fact been reduced. Because the hardware changes little from system to system, fewer drawings are required. Engineering time has also dropped in those applications which have a closely-related predecessor system available upon which to build. Software containing conditionally assembled statements has been written which can be easily modified to respond to customer demands.

1.2.2 4040 Microcomputer

Soft errors due to noise are liable to occur in an electrically noisy environment such as that of the Phillipsburg inserter; so a high voltage (15 volt) microprocessor was specified, the Intel 4040.

The 4040 is a single chip four bit parallel MOS central processor. It contains the necessary hardware to accept and process single level interrupts, and can address up to 8K bytes of EPROM. The assembly language instruction set has 60 instructions and a 10.8 microsecond cycle time.

The microcomputer is packaged in a single board. It has 64 inputs and 128 outputs, with seven prioritized interrupts. There is an 8K EPROM memory space, a 960
nibble volatile ram space, and a 512 nibble nonvolatile ram space.

1.3 Software Development Cycle

Each machine manufactured at the Phillipsburg facility is designed and debugged by a single applications engineer. First, the hardware design is completed and released to production control so the production process may begin. Then, the software design is begun. A block diagram of the software development cycle appears in Figure 1.

After the system design has been completed in some high level process language or flow chart, the design is coded in 4040 assembly language. Program development then proceeds with the aid of a minicomputer.

1.3.1 PDP 11/04 Minicomputer

There are two PDP 11/04 minicomputers at Phillipsburg. The operating systems resident on these machines provide all the standard software development tools, including an editor and an assembler.

1.3.2 IM1010 Programmer

The International Microsystems Model 1010 Programmer is a microcomputer-controlled instrument capable of pro-
SOFTWARE DEVELOPMENT CYCLE

BEGIN

DESIGN

CODE

COMPOSE/EDIT

ASSEMBLE

1

1

YES

ASSEMBLY ERRORS?

LOAD EPROM PROGRAMMER

PROGRAM EPROMS

PLACE EPROMS IN MICROCOMPUTER

2

2

3

NO

MACHINE WORKS

SHIP MACHINE

END

FIGURE 1
gramming EPROMs. Data enters the device over an RS-232C serial link into a 4K x 8 RAM data buffer. When the data transfer is complete, an EPROM is plugged into the IM1010 and programming proceeds off-line.

The 4K bytes of data fit in two 2716 (2K x 8) EPROMs or one 2732 (4K x 8) EPROM. The time to program each 2716 is approximately two minutes, and the time to program each 2732 is about four minutes.

Application programs larger than 4K require a second assembly. Symbolic references between the two modules are manually linked by assigning addresses to the external references at the beginning of each module.

1.3.3 Target Machine

The target machine itself is used to debug the system software. Once the machine is built, the microcomputer is plugged in and testing begins.

Sample material supplied by the customer is processed on the new system to verify proper operation. Malfunctions at this point, by and large, should be limited to software problems because wiring checks are performed on the machine before the microcomputer is installed.

The debugging of real time process control software is much easier if a logic analyzer is available. The analyzer available to the Phillipsburg engineer is an in-
house design which implements breakpoints and single stepping. The contents of certain registers within the CPU may be displayed as well.

Once a software error has been identified, a correction must be designed and coded, and new EPROMs must be programmed and tested. This cycle continues until the system is fully operational.

1.4 A New Control System

The 4040 Microcomputer has increased engineering productivity, but there are still systems being built which are too large for the 4040. Several systems, in fact, have been implemented using two microcomputers; average system complexity is increasing. A microcomputer with greater memory capacity is needed to help further reduce costs.

The growing volume of incoming orders has also put pressure on engineering to increase productivity. One proven way of improving software productivity is to code in a high level language. The number of lines of code written per unit time remains the same in high level or assembly languages, but each high level line may be the equivalent of three or four lines of assembly code. Of course, to realize this advantage, a compiler which generates executable code is required. High level lan-
guage also allow structured programming which makes pro-
grams easier to de-bug and more readily self-documenting.

A new control system is being assembled in an effort
to respond to these problems. A Texas Instruments TI
990-4 software development system has been purchased to
support a microcomputer featuring a TI family micro-
processor, the TMS 9995.

1.4.1 TI 990-4 Minicomputer

The Texas Instruments 990-4 hosts a multitasking and
- - —
multiprogramming real time operating system, DX/10.
Presently the system is supporting eight video terminals.

1.4.1.1 Microprocessor Pascal

The Texas Instruments Microprocessor Pascal (MPP)
development system executes under the DX/10 system. This
software package supports the development and execution of
a superset of the Pascal language on the 9900 family of
microprocessors.

Pascal was selected as the high level language to be
used because its block structure allows the user to design
a reliable and compact program.

The Texas Instruments' version of pascal supports
special I/O procedures and run-time concurrency. Target
systems produced have an operating system called MPX which
controls aspects of the software's execution such as CPU usage, system memory usage, and routine calling conventions. This package allows the sharing of a single processor by a number of processes during execution.

1.4.1.2 Software Development Tools

The Microprocessor Pascal (MPP) system provides four major tools supporting software development on the host computer. An interactive editor provides for source preparation with a syntax checking capability. A compiler generates interpretive code from MPP source which may be executed on the host computer. A code generator produces object code which may be run on the target system, and an interactive debugging interpreter enables the user to execute and debug MPP code.

1.4.2 TMS 9995 Microprocessor

The TMS 9995 Microprocessor is a single chip 16 bit central processing unit with 256 bytes of on-chip RAM. Its unique memory-to-memory architecture features multiple working register files resident in memory, so the traditional 'load' and 'store' instructions are eliminated. The processor directly addresses 65K bytes (32K words) of memory address space. There are 7 prioritized hardware interrupts and 16 software interrupts supported in
the processor hardware.

1.4.3 9995 Microcomputer

Phillipsburg has designed a two-board microcomputer utilizing the TMS 9995. Since the 9995 is a low voltage (5 volt) processor designed to operate at high frequency (12 megahertz), care was taken to electrically isolate the noisy I/O board from the sensitive CPU board.

Other features of the computer include 128 inputs and 256 outputs, 16 prioritized interrupts, and an RS-232C serial data port. The inputs and outputs may be controlled individually or in groups of up to 16 lines adjacent in address space.

1.4.4 Software Development

It will be possible to develop software with the new system in much the same way as with the old system. Coding in Pascal will no doubt be easier and faster, and there are powerful software development tools within NPP, but the very large object modules generated could create some problems.

The first problem is the size of the IM1010 Programmer's RAM buffer. Most programs will not fit in this space, and software to split object code into 4K blocks must be written. This can be done, but it is cumbersome...
to deal with numerous dumps to create just one version of application software for testing.

The second problem is more serious. The time required to program the EPROMs grows with the size of the application program, and object code from a compiler is normally less space efficient than hand written assembly code. A study of the memory space taken by representative 4040 assembly code software routines compared to the space taken by their Pascal equivalents illustrates the point. (See Figure 2.) In addition, a run-time executive will reside in EPROM in each system. This program overhead has no equivalent in a Phillipsburg 4040 microcomputer system.

Clearly, the programming of EPROMs could become a serious bottleneck in the development process.

2. Approach

A number of technically feasible responses to the software productivity problem are presented in this section. Evaluations are made with respect to product development effort, cost of implementation, and ease of use.

The options to be considered in this chapter are organized as follows:
## Code Length Comparison

<table>
<thead>
<tr>
<th>Name of Routine</th>
<th>4040 Length (Hexadecimal)</th>
<th>Pascal Length (Hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMA</td>
<td>109</td>
<td>228</td>
</tr>
<tr>
<td>SYSWAT</td>
<td>3F</td>
<td>72</td>
</tr>
<tr>
<td>FPL</td>
<td>33</td>
<td>BE</td>
</tr>
<tr>
<td>FRS</td>
<td>92</td>
<td>70</td>
</tr>
<tr>
<td>UCT</td>
<td>24</td>
<td>54</td>
</tr>
<tr>
<td>DET</td>
<td>23</td>
<td>56</td>
</tr>
<tr>
<td>STA</td>
<td>A3</td>
<td>1B4</td>
</tr>
<tr>
<td>RDV</td>
<td>52</td>
<td>8A</td>
</tr>
<tr>
<td>FPl</td>
<td>77</td>
<td>182</td>
</tr>
</tbody>
</table>

**FIGURE 2**
2.1 Software Machine Emulation

2.2 Target Microcomputer Modification

2.2.1 Dual Purpose Microcomputer

2.2.2 Separate Development Microcomputer

2.2.3 Portable Memory

2.1 Software Machine Emulation

An elegant solution would be one which offered a fully debugged software package even before the target machine were built. A machine emulator (in software) is such a system. In this scenario, only one set of EPROMs need be programmed for each application; that is the set to be installed in and delivered with the target system. The overriding time saving aspect of this approach is the fact that the debugging and manufacturing processes may proceed in parallel rather than in series.

The emulator would run on the development minicomputer and provide an executing environment for the application software. It would be a software "drive," providing inputs and interrupts to the program under test, as well as accepting and formatting the outputs for review by the software engineer.

The biggest stumbling block to this approach is the wide diversity of systems produced at the facility. Some functions remain common to all applications, but many
others are not. This means that the emulator would have to be capable of accepting parametric entries which would reconfigure the software "machine" for each system to be debugged.

I/O is another major problem. These control panel inputs which would be available to the real system's operator must be available to the emulator system's user. Certain discretionary actions available to the emulated system's operator must also be duplicated. For example, the removal of paper from the machine (as sensed by the computer through the use of photocells) must be possible in the emulated system.

In order to make the emulator a practical development tool, I/O interfaces to the user must be conceived which are easy to use. All discretionary inputs could be meaningfully implemented with the terminal's keyboard if the positions of the input functions on the keyboard were carefully chosen. For example, inputs from each of the control panels would be grouped together, and the simulated panels would be placed on the keyboard at the same relative positions that they would be found on the machine.

The bar coded information read on paper entering the machine is one type of input which would require special attention. More than likely, a file of data records would be created by the testing engineer which the emulator
software would access at the proper times during system emulation. This file would no doubt be created with the help of a small program which would accept parametric entries for the total number of bars per record and the frequency of occurrence of each bar. There would also be the need for an on-line overriding capability for the testing engineer to modify this file if necessary.

Another important aspect of I/O emulation is the method employed to give the testing engineer feedback on the software machine's operation. It would be possible to present the machine's status at the terminal in the form of a list of all outputs and their respective states ('on' or 'off'), but this approach is extremely user-hostile. For most systems, this list would be very long and difficult to integrate into a meaningful picture. It would be better to create something with more visual impact, like a graphic display of the machine on the development mini's CRT.

The ideal graphic design would present the machine's status at a glance. Microcomputer controlled hardware around the machine could be symbolized at the appropriate locations. Active components would appear in relief compared to their appearance in the inactive state. The machine's rotational position and current speed (cycles per hour) would be displayed. A detail of the system
panel's indicator lamps would be available as well.

Obviously, building a software inserter emulator would be no trivial task; man-years of effort would be involved. This option must be rejected at this time because the engineering resources are not available for its timely implementation.

Practical experience with a test fixture at Phillipsburg designed to test the 4040 microcomputer shows that a less ambitious project is not the answer either.

The 4040 test fixture is equipped with a switch connected to each input and a lamp to each output of the microcomputer under test. Inputs are copied to outputs and the I/O hardware operation of the microcomputer may be verified under the direction of test software. The fixture has been a help in software testing situations. Engineers under some extraordinary scheduling pressures have at times placed a microcomputer with their application software in the fixture, and manipulated the switches to simulate machine inputs. After some painstaking work on the unit, an improved version of software has been brought to the target system and debugging time on the machine has been reduced. This helps when the machine itself is unavailable for testing (e.g. parts shortages) and the shipping deadline draws near, but, by and large, the technique is unpopular because it is so tedious. This
example demonstrates that cutting costs and development time from the emulator project by removing features designed to make it easy to use would jeopardize the practicality of the final product.

2.2 Target Microcomputer Modification

Perhaps the most obvious way of eliminating the EPROM problem is by creating a target microcomputer which can accept the program data directly from the development minicomputer. Before pursuing this approach, it is necessary to present certain attributes of the target microcomputer which are critical to the product's viability.

The first and foremost such attribute is cost. The addition of a significant amount of hardware beyond that already required for controller tasks cannot be tolerated; especially hardware used solely for software development purposes.

Secondly, there is a requirement for some type of nonvolatile storage for program memory within the microcomputer. The machine must maintain its instructions even if it loses power. If the system's cost is to be minimized, the only real alternative for program storage is read-only memory.
2.2.1 Dual Purpose Microcomputer

These constraints on the final system's configuration, however, do not preclude the use of an alternate IC memory during software development. If the target microcomputer's memory space contained CMOS technology rams, for example, the program could be loaded directly from the development minicomputer over the microcomputer's RS-232C interface. The data could be preserved with an on-board battery supply during transportation to the machine. It would be necessary for the RAMs to be pin compatible with the EPROMs in order to avoid the need for extra memory sockets. Compatible JEDEC standard byte wide devices of these types are, in fact, available.

This idea suffers from a number of drawbacks. The first is related to the memory capacity of the physical devices in question. 64K EPROMs (8K x 8) are readily available, but the largest byte wide RAM which supports a battery back-up mode is a 16K (2K x 8) device. A board designed to accept 64K of RAM, then, would require roughly four times as many sockets in the memory section as one designed for 64K of EPROMs.

Secondly, there is the question of the location within the processor's memory space of the loader program. This is the program which would control the processing of data received at the RS-232C port from the minicomputer.
One option would be to dedicate a small area (say, 2K bytes) within the memory space to an EPROM resident loader. This area would then be unavailable to application programs. Another option would be to implement a second bank of memory space using the board's I/O space. (See Figure 3.) This bank would contain the loader program, and the other bank would contain the application program. The few instructions actually implementing the bank switching during the loading process would reside in the 9995's internal RAM space, which is always enabled. This approach would have the advantage of preserving the full 64K byte memory space for application programs, but microcomputer board complexity and cost would increase.

This approach presents packaging problems as well. The microcomputer would have to be carried between the production area and the development laboratory during system debugging. An enclosure would have to be designed which would protect the device from physical and electrostatic damage without rendering the controls on the board inaccessible during operation.

2.2.2 Separate Development Microcomputer

All of the problems associated with creating a single microcomputer well suited for both development and field operation may be solved simply by designing separate
MEMORY BANK SWITCHING

FIGURE 2
boards for each application. The production board would be designed to run with EPROMs, and the development board would have a 64K RAM capacity, an EPROM resident loader, and memory bank switching as described above. The development units could be packaged especially for durability; a limited number would be produced (roughly six units) to handle factory debugging needs indefinitely.

Note that some extra system checkout would be necessary with this approach. After the system were fully operational with the development micro, the production unit would have to be installed and tested in a separate step. This would not be very time-consuming, and is not considered a serious drawback.

2.2.3 Portable Memory

The portable development microcomputer is a workable solution, but a refinement of the approach can cut costs substantially. Comparing the two microcomputers, it appears that the memory sections would be radically different, but the remaining areas of the boards would be essentially the same.

Instead of a separate development microcomputer, it is preferable to design a portable memory module which could operate in conjunction with the production microcomputer during testing. The module would contain 32 RAM
chips, a battery, and a small dedicated controller to handle the loading process. During testing, the microcomputer would contain no EPROMs; rather, the nonvolatile RAM array would be directly accessible to the 9995 over control, data, and address umbilicals.

2.2.4 Conclusion

The last approach is adopted for the following reasons:

1. It adds no cost to the production product.

2. The approach is straightforward. Its simplicity offers a high probability of project success in a reasonable amount of time.

3. Requirements

The minimum requirement which must be met is that there be no need to program EPROMs during the debugging process. Additional capabilities for the Portable Memory will be considered and implemented if found to be effective and within the scope of this project.

4. Requirements Evaluation

The direct beneficiaries of the Portable Memory will be the application software engineers. The prime improvement to the software development process will be a sub-
stantial reduction in the time required to prepare a new software version after discovery of a system bug.

The fulfillment of this requirement also will improve the engineers' effectiveness beyond these measurable time savings. A tedious, distracting task will be eliminated, leaving the user better able to focus on the important problems at hand.

5. System Specification with Alternatives

This section contains a list of functions which the system must perform in order to meet the above requirement. More than one performance level is proposed for each function.

5.1 Download From Minicomputer to Portable Memory

The first major function which must be performed in order to meet the primary system requirement is to download from the minicomputer to the Portable Memory.

5.1.1 Data Stream Format

The format of the serial data stream from the minicomputer must be chosen. There are two reasonable candidates for this format. (See Figures 4 and 5.)

5.1.1.1 Link Loader Format

Object code produced by the minicomputer resident link loader program is in this format.
This is the first alternative to be considered for the data stream design.

5.1.1.2 Intel Format

The IM1010 Programmer requires Intel format encoding. If the programmer is to be used, a formatting program must be created to reformat the link/loader object code. This formatter in turn can be of two different types.

(A) Interpretive Formatter

The formatter program may run interpretively on the minicomputer.

(B) Native Code Formatter

Alternatively, the formatter may be executed directly by the minicomputer's processor.

5.1.2 Portable Memory Load Controller

The controller may be designed as a dedicated special purpose piece of hardware, or it may be a microcomputer.

(A) Dedicated Loader

(B) General Purpose Loader

5.2 Portable Memory Emulates Memory Space

This is the second main function to be performed by the Portable Memory System.

5.2.1 Volatile Memory

The devices within the Portable Memory which are chosen to emulate the static volatile RAM chips may trade
## Link Loader Data Format

<table>
<thead>
<tr>
<th>Field Identifier</th>
<th>Field Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>Entry Point Address</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>Load Address</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>Data Value</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>External Definition Address</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>External Reference Address</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>Checksum</td>
</tr>
<tr>
<td>F</td>
<td>-</td>
<td>End of Record</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>Program Identifier</td>
</tr>
</tbody>
</table>

Figure 4
## Intel Data Format

<table>
<thead>
<tr>
<th>Field Number</th>
<th>Field Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Record Mark (Colon)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Record Length</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>Load Address</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Record Type</td>
</tr>
<tr>
<td>5</td>
<td>32 (typical)</td>
<td>Data Field</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

**Figure 5**
off speed for some other attribute, such as high density. Two levels of performance of the quality of emulation will be considered with respect to device speed.

(A) No Speed Degradation
(B) Some Speed Degradation

5.2.2 Nonvolatile Memory

5.2.2.1 Program Memory

It is impossible to write new data to the emulated chips (EPROMs) during target system operation, but the RAMs which emulate the EPROMs can accept new data. Three performance levels of the program emulating RAMs with respect to the ability to write into the devices are to be considered:

(A) Enable Writing
(B) Disable Writing
(C) Switchable Writing

5.2.2.2 Parametric Data Memory

The possibility of some speed degradation in this area of program memory is considered.

(A) No Speed Degradation
(B) Some Speed Degradation

5.2.3 RAM/ROM Configuration

The location of the three types of memory within the physical address space will vary from application to application. The Portable Memory must work for any applica-
tion, no matter what the RAM/ROM mixture might be. The
reconfiguring of the device may be accomplished a number
of ways.

(A) Manual
The user has to reconfigure the device.

(B) Semi Automatic
The device itself assists in the reconfiguration.

(C) Automatic
The device uses information provided by the link
loader to reconfigure itself.

5.2.4 Data Integrity Verification

This feature is considered to be an important part of
the emulation process because confidence in the device on
the part of the application engineer is essential. A
tough software problem can lead to doubts about program
data integrity. This is when the ability of the device to
perform a quick data check on itself is indispensable.

(A) Go / No Go
Upon command, the Portable Memory gives a go / no go
indication of data integrity.

(B) Error Detection / Correction
Errors are automatically detected and/or corrected by
the Portable Memory.

(C) Display Checksums - Unlimited
The checksum of any section of memory is displayed
upon command.

(D) Display Checksum - Limited

The checksum of any 4K section of program memory located on a 4K boundary is displayed upon command.

5.3 Other Features

The remaining features might be nice to have but are not essential to meet the system requirement. The plan is to implement these features which are judged appropriate as time and resources permit. The order of implementation is determined in the "cost performance evaluation" below.

5.3.1 Ram Display / Edit

Any location in RAM within the Portable Memory may be displayed and/or changed. This could be useful in a situation where a "quick and dirty" change might be preferable to a full edit-recompile-reload process.

5.3.2 Self Test Mode

This feature is being considered because it may help preserve data integrity. Three areas of the Portable Memory are treated.

(A) Memory Section Test
(B) I/O Test
(C) Controller Test
5.3.3. Transmit Data in Intel Format

This would give the user the ability to dump data to the IM1010 Programmer directly from the Portable Memory. This might be done for data inspection or for programming EPROMs in preparation for system shipment.

5.3.4. Support Target System Breakpoints

There is debugging software within the target operating system which is designed to operate in conjunction with commands issued from a data terminal over an RS-232C link. Breakpoints and single stepping are among the features supported by this software. These two features will work only in an environment which allows the software under test to be modified dynamically by the debugging software. The Portable Memory may provide the environment necessary to support these features.

6. Gross System Design

In this chapter, general design approaches are described for each of the alternatives listed above. These preliminary designs help determine the desirability of each approach when the final evaluation is made.

6.1 Download From Minicomputer to Portable Memory
6.1.1 Data Stream Format

6.1.1.1 Link Loader Format

Object code in this format could be transmitted to the Portable Memory without modification.

6.1.1.2 Intel Format

The link loader object code would be processed by a formatting program before transmission.

(A) Pascal

If a formatting program is written, it may be in Pascal. The Pascal compiler produces P-code which may be run interpretively on the minicomputer.

(B) Fortran

Alternatively, the Fortran compiler can produce modules which may be installed as tasks within the operating system. These modules are in native code and may be executed directly by the minicomputer's processor.

6.1.2 Portable Memory Load Controller

(A) Dedicated Controller

Special purpose integrated circuits exist for the purpose of controlling serial data. This design approach could be implemented simply with a UART and some address and data registers.
A microcomputer could serve as a general purpose controller in this application. If a microcomputer is to be used, a processor family must be chosen. Two types of processors are supported by software development systems at the facility, the Intel 4040 and the TI 9900 family processors. For this reason, only these two families can be considered practical alternatives.

(A) TI 9900 Family
(B) Intel 4040

If the microcomputer approach is to be taken, a decision must be made as to whether to design a new board or use a commercially available board. More information about the details of the design requirements are needed to make this determination, and the subject will be raised again later in the paper.

6.2 Portable Memory Emulates Memory Space

6.2.1 Volatile Memory

The operating speed of the emulating memory devices with respect to those memory devices used in the final system configuration will have an effect on the memory selection process. If a relaxed stance in this area can be tolerated, a wider range of devices can be considered.
6.2.2 Nonvolatile Memory

The ability to change the data within the EPROM emulating devices can be implemented in a straightforward manner. This ability can be readily controlled with a single control signal, the "Write Enable."

6.2.3 RAM/ROM Configuration

(A) Manual

The device could be reconfigured by changing the RAM chips at the appropriate locations and by changing the positions of switches.

(B) Semi Automatic - RAM

(C) Semi Automatic - PROM

A programmable site select area would support the creation of a device configuration programming mode. The site select logic could be implemented with RAM devices which could be reprogrammed per application requirements at the command of the user. Alternatively, if PROMs were used, the device could be reconfigured by reprogramming the PROMs.

6.2.3.2 Automatic

This approach would require the use of RAM devices in the memory select logic as well. The reconfiguration commands would be issued by the link/loader software and not
the user.

6.2.4 Data Integrity Verification

(A) Go/No Go

Checksums would have to be generated within the minicomputer resident data formatting software. When the user would request a verification, the Portable Memory resident software would recalculate the checksum and compare it to the checksum supplied by the formatter.

(B) Error Detection/Correction

This approach would require some data redundancy, possibly in the form of parity bits. Special provision would be made in the memory structure to support this function.

(C) Display Checksums - Unlimited

A keypad and multiple digit display on the Portable Memory would be required to implement this function. The Portable Memory software could calculate the checksums in the commanded area.

(D) Display Checksums - Limited

A sixteen position switch could be used in this approach instead of a keypad. Otherwise, this alternative would be implemented similarly to the previous one.

6.3 Other Features
6.3.1 RAM Display/Edit

This function would be supported in Portable Memory software. A keypad would be required.

6.3.2 Self Test Mode

(A) Memory Test

The Portable Memory controller could write various patterns to the memory area and read each one to verify proper operation.

(B) I/O Test

An I/O "mirroring" scheme could be implemented which would mirror the inputs at well defined outputs. This function also would be supported in Portable Memory software.

(C) Controller Test

Signature analysis could be supported.

6.3.3 Transmit Data in Intel Format

This capability could be supported in Portable Memory controller software.

6.3.4 Support Target System Breakpoints

Provision must be made to allow the "Write Enable" control signal to be active at all EPROM emulating devices.
7.0 Cost Performance Evaluation

This chapter contains an analysis of each system function alternative. The results are summarized in Table 1.

Ratings in each case are made in six separate areas of two levels of priority. The ratings of the high priority items are weighted double those of the lower priority items. Each rating is done on a scale of one to five; one is excellent, two is good, three is satisfactory, four is poor, and five is failing. Only those ratings deviating from satisfactory are discussed.

High priority areas are "Relevance to Facility," "Ease of Use," and "Data Integrity."

Lower priority items are "Flexibility," "Product Development Effort," and "Cost."

The lowest total in each category is implemented.

7.1 Data Stream Format

(A) Link Loader Format

"Ease of Use" - good

No formatting program need be invoked if the link loader format is chosen. This simplifies the software development procedure for the user.

"Product Development Effort" - good

38
No formatting program need be designed.

(B) Intel Format

"Relevance to Facility" - good

The user will need to use this formatter to dump data to the IM1010 PROM programmer. It is an advantage to keep the procedures for dumping to the IM1010 and the Portable Memory identical.

7.2 Formatter Language

(A) Pascal

"Relevance to Facility" - good

A program written in Pascal at the Phillipsburg facility is especially easy to maintain because most engineers are familiar with the language.

"Product Development Effort" - good

"Flexibility" - good

Pascal is noted as a well-structured language. It will be easier to get a formatter written in Pascal functioning properly.

(B) Fortran

"Relevance to Facility" - good

The operating system of the TI 990-4 minicomputer at Phillipsburg supports Fortran with directly executable object modules. These modules run
especially quickly. This is important in a production environment.

"Ease of Use" - Good

The Fortran formatter may be called in batch mode, while Pascal modules may not. As a result, the Fortran program will run unattended.

7.3 Controller Type

(A) Special Purpose

"Flexibility" - Poor

A dedicated design cannot be easily modified should the need arise.

(B) Microcomputer

"Flexibility" - Excellent

System changes may be readily implemented in software.

7.4 Microprocessor Type

(A) Texas Instruments 9995

"Product Development Effort" - Good

The 9995 has the capability of addressing the full 64K application program address space directly. This simplifies the hardware design. Furthermore, the power of the Microprocessor Pascal software development system is available if this processor is used.
The 4040 can only directly address 4K of memory address space. All programming for this device must be done in assembly language at Phillipsburg.

7.5 Memory Emulation Speed

(A) No Speed Penalty

"Flexibility" - Poor

The process of specifying the memory ICs to be used within the Portable Memory is more complex if this rigid stance is taken.

(B) Speed Penalty Allowed

"Relevance to Facility" - Fail

Although it is possible to operate the target microcomputer with slower memories by reducing crystal speed, it is not desirable to do so from a standpoint of providing a true system test. The system should be tested at the speed at which it will ultimately operate. Otherwise, there is the risk of missing problems associated with high frequency operation; soft errors within the microcomputer, for example, are more prone to occur at high frequencies. Furthermore, certain hardware on the target board is supported by software which is crystal
speed dependent. Different software would have to be written to be used during testing, creating the potential for shipping systems with the wrong software.

7.6 Program Memory Emulation

(A) Write Enable Always Enabled

"Data Integrity" - Poor

A faulty program could conceivably corrupt itself.

(B) Write Enable Always Disabled

"Data Integrity" - Good

There is no chance of corrupting data accidentally within RAM by leaving the switch in the wrong position.

(C) Optional Write Enable

"Relevance to Facility" - Good

This approach has the advantage of providing a way of revealing otherwise hidden program problems. With this capability, testing begins with the "Write" signal disabled. As a final test, the "Write" signal is enabled. If the system malfunctions at this time, there is something wrong with the program.

7.7 RAM/ROM Programmability

(A) Manual
"Ease of Use" - Satisfactory

Despite the fact that this approach requires that the user change chips and reposition switches, there are mitigating factors. The RAM/ROM object map is under the control of the applications engineer. The link loader program carries out the commands for location of program memory within the memory address space. An infinite number of configurations are conceivable, but a convention is easily established which limits that number.

This convention must satisfy the microcomputer requirements for nonvolatile memory at locations 0000 through OFFF (HEX) and F000 through FFFF. RAM will reside in an area which has address EFFF as an upper bound and grow toward 0000 as application requirements dictate. An initial allocation of 8K of RAM (resident at B000 through EFFF) should be sufficient for the great majority of programs. The rest of the 64K can be allocated to ROM-emulating space. Very little manipulation of the Portable Memory is anticipated until application programs grow to a size near 56K. At that point, a new microcomputer and new Portable Memory will be needed.

"Product Development Effort" - Good

This approach requires little product
development effort because there is a low level of automation involved.

(B) Semi-Automatic - RAM

"Ease of Use" - Good

When reprogramming is required, the user enters a 4K memory address (selectable by a sixteen-position switch) into the Portable Memory. Then a command is issued to allocate that area to RAM or to ROM. The entire process is complete in a short time, even if all 64K of memory space needs to be reconfigured.

"Product Development Effort" - Poor

Software which supports the programming mode must be developed. Very high speed RAM must be used so it will operate at 12MHz). This RAM must be nonvolatile so the device does not "forget" its configuration. This combination of high speed and low power is difficult to find in a memory device.

"Cost" - Poor

The Portable Memory's cost is increased by extra site select RAMs and extra battery capacity for RAM nonvolatility.

(C) Semi-Automatic - ROM

"Ease of Use" - Poor

A new PROM site select chip must be
programmed for each map. The user determines the proper program and programs the PROM on the IM1010. The calculation of the correct data to place within the PROM is not trivial.

(D) Automatic

"Ease of Use" - Excellent

No user actions are required to reconfigure the Portable Memory with this approach.

"Product Development Effort" - Fail

In addition to those tasks presented in the "Semi-Automatic - RAM" section above, it is necessary to modify the formatter program; this program must pass on the ROM requirements of the application program supplied by the link loader.

7.8 Data Integrity Verification

(A) Go / No Go

"Ease of Use" - Good

The user simply presses a "Data Check" control pushbutton to get the verification required.

(B) Error Detection

"Product Development Effort" - Poor

45
Special redundancy circuits must be designed to implement error detection.

(C) Unlimited Checksum

"Cost" - Poor

The user keys in the addresses required for checksum purposes. The keypad required represents a significant increase in device cost.

(D) Limited Checksums

"Flexibility" - Good

If a 4K checksum is available from the Portable Memory, EPROM checksums generated by the IM1010 programmer may be checked after downloading from the Portable Memory to the IM1010.

7.9 Other Features

(A) RAM Display/Edit

"Data Integrity" - Fail

The ability to manually edit data within the Portable Memory jeopardizes data integrity. Furthermore, it is possible that the system's EPROMs will not receive off-line updates and faulty software may be delivered with the system.

(B) Memory Test

"Data Integrity" - Good

Self testing of the Portable Memory
obviously helps ensure data integrity. (This applies to the "Controller test" feature as well.)

(C) Controller Test

"Product Development Effort" - Poor

Signature analysis requires a special software driver program which provides repeatable cycles at each node within the computer hardware. Feedback loops within the microcomputer must be identified and disabled.

It should be noted that certain areas of the microcomputer are tested within the memory test already proposed. Other areas are tested when the checksums are compared after data is transferred.

(D) I/O Test

Since the Portable Memory I/O is not directly involved in the integrity of the data within the device, there is no compelling need for this feature.

(E) Transmit Mode

"Relevance to Facility" - Good

This feature has the attribute of permitting direct loading of the IM1010 Programmer from the Portable Memory. The alternative is to load the IM1010 from the development minicomputer; this involves the development of software to break up the object module into 4K-sized segments.
"Flexibility" - Good

If this capability is implemented on both the Portable Memory and the development minicomputer, each will provide a backup capacity in case of hardware failure.

(F) Breakpoints

"Relevance to Facility" - Good

This feature provides powerful system debugging tools for the user.
<table>
<thead>
<tr>
<th>Section</th>
<th>Category</th>
<th>Relevance to Facility</th>
<th>Ease of Use</th>
<th>Data Integrity</th>
<th>Subtotal</th>
<th>Total</th>
<th>Flexibility</th>
<th>Product Development Effort</th>
<th>Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Data Stream Format</td>
<td>3 2 3 16 3 2 3 24</td>
<td>2 3 2 14 2 3 3 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Link Loader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>Formatter Language</td>
<td>2 3 3 16 2 2 3 23</td>
<td>2 2 3 14 3 3 3 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pascal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fortan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>Controller Type</td>
<td>3 3 3 18 4 3 3 28</td>
<td>3 3 3 18 1 3 3 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Special Purpose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microcomputer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>Microprocessor Type</td>
<td>3 3 3 18 3 2 3 26</td>
<td>3 3 3 18 3 4 3 28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>Memory Emulation</td>
<td>3 3 3 18 3 3 3 27</td>
<td>5 - - - - - - - F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speed No Speed Penalty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Speed Penalty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Index:
- 0 - Relevance to Facility
- 1 - Ease of Use
- 2 - Data Integrity
- 3 - Subtotal
- 4 - Flexibility
- 5 - Product Development Effort
- 6 - Cost
- 7 - Total

* - Alternative chosen for implementation
## Cost Performance Evaluation Results (cont.)

<table>
<thead>
<tr>
<th>7.6 Program Memory Emulation</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable WE</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>Disable WE</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>16</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Optional WE</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>24*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.7 RAM/ROM Programmability</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>26*</td>
</tr>
<tr>
<td>Semi Auto(RAM)</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>27</td>
</tr>
<tr>
<td>Semi Auto(PROM)</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>Automatic</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>16</td>
<td>3</td>
<td>5</td>
<td>-</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.8 Data Integrity Verification</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go/No Go</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>16</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>25*</td>
</tr>
<tr>
<td>Error Detect</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>5</td>
<td>-</td>
<td>F</td>
</tr>
<tr>
<td>Unlimited</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>Checksum</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>26*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.9 Other Features</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Display/edit</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>F</td>
</tr>
<tr>
<td>Memory Test</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>16</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>I/O Test</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>Controller Test</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>16</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>Transmit Mode</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>Breakpoints</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>24</td>
</tr>
</tbody>
</table>

See index on first page of table.

*Table 1 (2 of 2)*

50
8. Final System Specification

The role of the Portable Memory (PM) within the software development system encompasses six modes of operation. The data transfer from the development minicomputer to the PM is called the "Load" mode. The quiescent state of the PM during transportation to the target system is called the "Transport" mode. The operation of the PM with the target microcomputer is called "Emulation"; if the data terminal is connected to the target board as well the mode is called "Debug." Data transmission from the PM to the IM1010 is called "Transmit" mode. The "Load" and "Transmit" modes are both entered out of the final mode, "Manual." In this mode, the user may enter commands to transmit, checksum, or clear data. Figure 6 shows the interactions of the PM with the other elements of the software development system.

The following paragraphs contain the results of the issues treated in the previous chapter and provide the details of the final system specification.

8.1 Data Stream Format

The PM will be loaded in Intel format.

8.2 Formatter Software

At first, this software will be an interpretive Pascal module. The formatter can be made operational quickly this way. Later, a Fortran-sourced module will be created for fast execution.

8.3 Portable Memory Controller
The controller will be a microcomputer based on the TI 9995 microprocessor.

8.4 Memory Emulation

The Portable Memory will operate at the maximum rated speed of the target microcomputer, 12MHz. The user will be provided the option of allowing data to be written to the EPROM-emulating RAMs.

8.5 RAM/ROM Configuration

All IC socket sites within the PM will be configured by the user; provision for placement of shorting plugs on the memory boards will be provided.

8.6 Data Integrity Verification

A "Data Check" control will provide a go/no go verification of PM data. As time permits, the "limited checksum" feature will also be implemented.

8.7 Other Features

The following features will be implemented as time permits in the following order:

1. Transmit Mode
2. Breakpoints
3. Memory Self Test
9. Final System Design

9.1 Selection of Memory ICs

The marketplace is inundated with many types of RAM devices, each with its own strengths and drawbacks. In order to choose the most appropriate type of memory for this application, the systematic approach previously described is used once again.

The high priority criteria for this analysis are "Flexibility," "Speed," and "Nonvolatility." These are directly related to the ability of the memory module to satisfy the system specifications. The other criteria for evaluation are "Density," "Upgradability," "Development Effort," and "Cost."

In order to organize the presentation of the selection process, the chips with the greatest bit density available are considered for each of the three popular memory organizations, bitwise, nibblewise, and bytewise. The evaluation results are summarized in Table 2.

9.1.1 Bitwise

This organization is the first one which appeared when memory ICs were developed. There is one bit at each
uniquely addressable location within the device. Three separate types of RAMs are considered within this category, Dynamic (64K x 1), Static (16K x 1), and a Static/EEPROM hybrid (1K x 1).

9.1.1.1 Bitwide Dynamic

In dynamic memories, each storage cell circuit is comprised of a single transistor and a tiny capacitor. These devices store bits of data in the form of an electric charge. However, since the voltage on the capacitor decays in a very short time, it must be periodically recharged to maintain the required voltage level. This function is termed "refresh" and must be accomplished periodically to prevent loss of data. The refresh function requires additional clocking circuitry and logic.

The 64K device under consideration is in a sixteen pin package and is implemented with row and column addresses multiplexed on the same eight pins. The entire 64K bytes of memory space can be implemented with only eight of these chips; this array has outstanding density. (See Figure 7.)

In the key area of device speed, however, this chip fails to meet system specification. No dynamic RAM (DRAM) presently available can operate at 12MHz with the
64K BYTE MEMORY ARRAY - DYNAMIC

ADDRESS BUS (A0 - A15)

FIGURE 7
9995. The following analysis reveals why.

Figure 8 contains the memory read cycle timing diagram for the 9995 processor at 12MHz. The memory "access time" for this processor is defined as the time the processor allots for a memory device to present valid data to the processor. This time ("Tacc") is 120nS.

In the 64 DRAM read cycle timing diagram reproduced in Figure 9, the row address ("RAS") and column address ("CAS") strobes which are required to implement the address multiplexing are shown. The access time for the device is defined as the time it takes for valid data to appear after the RAS strobe is invoked.

Comparing the two sets of data, it appears that a DRAM with an access time of 120nS (the fastest one available) might be fast enough, but this is not the case. The reason is that the 9995 does not offer an early "Address Valid" signal. Consequently, there is no way of detecting when the address bus has settled into valid data until the next rising edge of the 9995 "Clkout" signal. This reduces the true access time available to only 72nS, even before multiplexing propagation delays are considered.

9.1.1.2 Bitwide Static

The static RAM storage cell is comprised of two
DYNAMIC RAM READ CYCLE

FIGURE 9
transistors cross-coupled to form a flip-flop circuit. This type of storage cell is considerably larger than the dynamic cell, so the static RAMs are less dense. However, the storage cell design of the static RAM enables it to have a much faster access time than the dynamic RAM.

In addition to their superior speed performance, static RAMs have the advantage of not requiring refresh circuitry. Once a data bit is stored in a flip-flop, its contents remain static until changed. While static RAMs are less dense and cost more per bit than dynamic RAMs, they remain the only choice when speed is a critical factor.

The 16K bit density per package yields a reasonable array density. The design approach for a 64K byte array of these devices is shown in Figure 10.

This device is a poor candidate in terms of non-volatility. In the standby mode (with all chip selects disabled), a 64K array of these devices draws about 700mA. To preserve the data in this array for two hours requires a large (1400mAH) capacity battery. Furthermore, no device is available at this time with a battery backup mode, so power transfer circuitry is necessary to preserve data in these chips.

The upgradability of the memory section is defined in this application as the ability to double the memory
64K BYTE MEMORY ARRAY - BITWISE

ADDRESS BUS (A4 TO A15)

SELECT BANK 0

SELECT BANK 1

SELECT BANK 2

SELECT BANK 3

DO

DI

D7

FIGURE 10
capacity of the PM when required; the next generation TI processor, the 99000, directly addresses 128K and is considered the logical future replacement for the 9995.

The 16K x 1 device scores poorly in the area of upgradability. The ideal method of upgrading the PM is to replace all memory chips with chips of double capacity. In the case of the 16K device, however, no strategy for upgrading to 32K on the part of the semiconductor industry can be seen.

9.1.1.3 Bitwide EEPROM/Static

This device is of interest in this application because of its excellent nonvolatility characteristics. It is a nonvolatile RAM which does not require battery backup. Internally, the device is a 1K x 1 static RAM overlaid with an EEPROM (Electrically Erasable Programmable ROM). It has two special control pins, "Store" and "Recall." These signals control movement between the RAM and EEPROM. At the present state of the art, however, devices of this type cannot operate at 12MHz.

9.1.2 Nibblewide Static

The largest available nibblewide device is a 4K x 4 part. A 64K array of these would have a density similar to that of the bitwide static RAMS. (See Figure 11.)
The 4K x 4 static RAMs, like the 16K x 1 devices, are fast and power-hungry.

The biggest problem with this device is the fact that at the present time there is only one manufacturer of this organization at the 4K level. It is obviously unwise to use a sole-source part.

9.1.3 Bytewide Static

As might be expected, the maximum available density of the bytewide organization is also 16K bits (2K x 8). See Figure 12 for the design approach of the 64K byte array.

The most significant attribute of the bytewide device in this application is the fact that each device contains a number of full memory words rather than just a part of a larger number of words. This attribute facilitates the partitioning of the memory array into two separate bytewide devices; one is a low power device for EPROM emulation, and the other is a fast device for 12MHz operation. The bytewide organization offers a high level of flexibility in this respect.

A number of CMOS bytewide devices are available which offer a battery backup mode. The current drawn by these devices in the data retention mode is only 50 microamps.

The bytewide family has JEDEC approved standards for
future growth, including extensions of the existing 24 pin package to 28 pins. The planned 4K x 8 device is essentially pin compatible with the 2K x 8 device, with the addition of one address line.

9.1.4 Conclusion

The memory section of the PM is designed using the bytewide family of memories for the following reasons:

1. Pin compatibility within the family facilitates the reconfiguration of the memory section to respond to varying RAM/ROM mixtures.

2. The family offers an excellent CMOS RAM with a battery backup mode.

3. The PM will be easily upgradable when the bytewide 4K x 8 devices are available.

9.2 Memory Array Design

The 64K byte section of the Portable Memory is organized as shown in Figure 12. There are two printed circuit boards; each one contains 32K bytes. A schematic of the 32K byte board appears in Figure 13.

The issues of special concern within the memory section are memory site programmability and the preservation of data nonvolatility.
64K BYTE MEMORY ARRAY - BITEWIDE

ADDRESS BUS (A0 - A11)

ONE OF SIXTEEN DECODER

DATA BUS (D0 - D7)

ADDRESS BUS (A0 - A11)

ONE OF SIXTEEN DECODER

FIGURE 12
MEMORY BOARD SCHEMATIC

FIGURE 13 (1 OF 4)
MEMORY BOARD SCHEMATIC

FIGURE 13 (3 OF 4)
9.2.1 Site Programmability

Within the memory array, most RAM chips emulate EPROMs and the rest implement fast scratch pad memory. The EPROM-emulating sites are supplied with power backed by a battery and contain CMOS RAM chips. The fast sites are powered only during the load and emulate processes and contain NMOS RAM chips. In order to maximize the flexibility of the device, every site is designed to be programmable to either power source.

A second signal must be programmed to prepare each site for its designated RAM type. EPROM-emulating sites operate with a switchable "Write Enable" (WE) control signal. While the memory is loaded, the WE signal must be enabled, but during the emulation process, the WE signal can be optionally enabled or disabled. Fast sites operate with the WE signal always enabled.

The final option at each site has significance only in the future 128K operating mode, when each site must accept the 4K x 8 devices. In this event, the 2K x 8 "Write Enable" signal becomes the newest address line, "All." The new WE at each site may then be programmed for switchable and non-switchable operation as before.

9.2.2 Data Integrity

According to the manufacturer's specifications, the
data retention mode of the CMOS RAM depends on three conditions. (See Figure 14.) First, before the power is removed from the chip, the "Chip Enable" (CE) signal must be a logic "one"; that is, the chip must be disabled. Secondly, the voltage on the power pin (Vcc) may not fall below 2.0 volts. Finally, the data retention level of CE may not be less than the battery voltage less 0.2 volt. During transport mode, a nominal battery voltage of 3.6 volts is provided for both the Vcc and CE signals. (A voltmeter mounted on the unit displays the battery voltage for the user.)

During the load process, the loader and 64K memory banks are alternately enabled under control of the PM microcomputer. Each 32K board has a "Board Enable" signal which is connected to the "Enable" input of the 1-of-16 decoder. When this signal is a logic "high," all memory site "Chip Enable" signals are also a logic "high." At the conclusion of the load process, both memory boards are automatically disabled in anticipation of power being removed from the PM. All CMOS RAM sites are thus prepared to enter data retention mode.

After the load process and before power is removed from the PM, the WE signal is disabled by the user with the "Data Lock" switch. A pullup resistor tied to battery voltage in the switch affords data protection by preventing changes in this critical signal while the PM is
CMOS RAM DATA RETENTION MODE

<table>
<thead>
<tr>
<th>ITEM</th>
<th>SYMBOL</th>
<th>MIN.</th>
<th>MAX.</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{CC} FOR DATA RETENTION</td>
<td>V_{DR}</td>
<td>2.0</td>
<td>—</td>
<td>V</td>
</tr>
<tr>
<td>DATA RETENTION CURRENT</td>
<td>I_{CCDR}</td>
<td>—</td>
<td>50</td>
<td>μA</td>
</tr>
<tr>
<td>CHIP Deselect TO DATA RETEN. TIME</td>
<td>T_{CDR}</td>
<td>0</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>OPERATION RECOVERY TIME</td>
<td>T_{R}</td>
<td>300</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>

*TRC = READ CYCLE TIME

LOW V_{CC} DATA RETENTION WAVEFORM

![LOW V_{CC} DATA RETENTION WAVEFORM](image)

FIGURE 14
losing power. There is some voltage turbulence introduced on the WE line when the switch is toggled, but no data corruption occurs because no memory chip sites are enabled at this time.

In order to enter the emulation mode without data corruption, the following procedure must be followed. The "Data Lock" switch is placed in the locked position. Another switch, the "Emulate/Load" switch, is placed in the "Emulate" position. When power is applied, this switch signals the PM controller to prepare for emulation. The controller first calculates and verifies the checksum of the nonvolatile program and displays the results. Then it enables both memory boards. The user notes the "Checksum OK" signal, connects the data and address umbilicals, and applies power to the target microcomputer. The microcomputer then begins executing the program stored in the PM.

At the conclusion of the emulation session, the "Emulate/Load" switch must be placed in the "Load" position before power is removed from the PM. The controller then prepares the memory boards for entering data retention mode by disabling all memory site CE signals.

While in emulation mode, if the user wishes to use the data terminal for debugging purposes, the following procedure ensures no PM data is lost. First, power must
be removed from the PM and then the "Data Lock" switch must be unlocked. (Recall that so enabling the WE signal to all RAM sites allows the implementation of software breakpoint setting and single stepping.) Then, the data terminal is connected to the target board's RS-232C port. Power may then be applied; first, to the PM and then to the microcomputer.

9.3 Controller

The PM controller is a single board microcomputer based on the TMS 9995 processor. (See Figure 15.) All processor data, address, and control lines are brought to a connector to permit communication with the memory boards.

This microcomputer has 32 bits of software programmable I/O with an additional 12 bits of inputs. All 44 bits are individually addressable, and they may be addressed in groups of up to 16 contiguous bits. This synchronous I/O space is implemented with two TMS 9901 chips.

The serial RS-232C interface on the microcomputer is based upon another Texas Instruments peripheral LSI chip, the TMS 9902A. This device is a software programmable UART designed to interface with the 9995.

The memory space of the microcomputer consists of two bytewide 28 pin sockets. One socket each is dedicated to EPROM and RAM. The present memory capacity is limited by
FIGURE 15
76
the state of the art in bytewide devices, 64K bytes of EPROM and 2K bytes of RAM per chip. The RAM socket is provided with battery power so the checksum of the application program may be saved. The site select logic consists of a 32 x 8 prom which decodes the highest five address lines to select one of the two memory sockets.

The memories of this controller reside on the same busses as the two 32K memory boards. Memory bank switching between controller and emulator memory is controlled by the processor out of internal 9995 RAM using three bits of I/O space. One bit controls the "Enable" signal of the site select PROM, which in turn controls the "Chip Enable" signals of the controller memory sites. One more I/O bit is dedicated to each 32K memory board; during 64K operation one bit of I/O would suffice, but two are required for the 128K mode.

9.4 Minicomputer Serial Data Interface

Object files of the Intel formatter program are transmitted to the PM over an RS-232C serial link. An existing device service routine (DSR) installed in the operating system is adapted for use with the PM. This DSR was originally designed for use with a printer. Two signals on the TTY/EIA interface board within the minicomputer ("DSR9 and "CTS") are tied to appropriate voltage levels in order to make the adaptation function.
A commercially available microcomputer board, the TMS 990-101, is not suitable for the PM application for a number of reasons. The address and data busses are not available at any connector on the board, so the board would require modification. It also is too large for the application. Finally, it is too expensive.
### Memory Evaluation Results

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitwise Dynamic</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Bitwise Static</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>20</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>34</td>
</tr>
<tr>
<td>Bitwise EEPROM</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Nibblewide</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>22</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>Bytewide</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>23*</td>
</tr>
</tbody>
</table>

**Index:**

0 - Flexibility  
1 - Nonvolatility  
2 - Speed  
3 - Subtotal  
4 - Density  
5 - Upgradability  
6 - Product Development Effort  
7 - Cost  
8 - Total

* - Option is implemented.

**Table 2**
10. System Implementation

10.1 Printed Circuit Boards

Two printed circuit boards are designed and built for the initial configuration of the PM, the microcomputer board and one 32K memory board. The full 64K of memory is not built until the 32K board is debugged. This strategy saves money if changes on the board prove to be required.

In the design of these boards, special considerations are made for high frequency operation of the boards; board layout, placement of bus terminating components, and power distribution elements are all considered.

In both boards, the trace lengths of the high speed areas are kept to a minimum to avoid soft errors due to high frequency "ringing." The critical areas of the layout of the boards are specified, while uncritical areas like the I/O section of the microcomputer are left to the discretion of the layout designer.

All busses on the memory board are terminated with components designed to minimize the high frequency effects of undershoot and overshoot. These components are located at the maximum distance from the bus driving buffers for maximum effect. Furthermore, resistors are placed in series with the high speed busses to tune the
line impedance for minimum ringing.

High frequency noise generated by integrated circuits will tend to be coupled to neighboring chips through the power distribution system. Power distribution elements known as bus bars tend to suppress this type of noise rather than coupling it. Both boards are implemented using bus bars.

10.2 Packaging

The system requirements dictate that the address and data busses of the PM microcomputer and 32K memory board be connected during loading, and the busses of the memory board and the target microcomputer be connected during emulation. In addition, during emulation, the PM microcomputer should be disconnected from the bus. This isolation is required so the two asynchronously operating microcomputers do not interact. These constraints are satisfied by building cables which are permanently terminated on one end at the memory board and which may be connected as required to either of the microcomputers. The umbilicals are terminated on the microcomputer end in DIP plugs which replace existing ICs on the target microcomputer during emulation. This design eliminates the need for a dedicated connector on the target board. The length of these umbilicals, of course, is kept to a
Another packaging issue is whether to implement the PM in one or two modules. The function of the microcomputer is primarily to load the memory boards. If it is placed in a separate module which remains in the development laboratory, the remaining portion of the device will be more portable. On the other hand, if the PM is implemented in one module, the complexity of the packaging design is reduced and the microcomputer becomes available for checksum verification at the emulation site. The latter approach is adopted mainly for emulation site checksumming.

The PM chassis is 15 inches long by 5 inches wide by 8 inches high. It weighs 6 pounds.

10.3 Formatter Software

This software reformats link loader object modules into Intel format. It calculates data load addresses, the number of bytes in each data record, and the checksum of each record, and converts this data into ASCII code.

This program is developed on the 990-4 minicomputer using the MPP development system. The Pascal compiler creates an interpretively executable object module which resides on the minicomputer and is available for execution during microcomputer software development.
10.4 Controller Software

The primary function of this PM-resident module is to control the serial input of data from the development minicomputer. The data is decoded, checksums are verified, and the data is loaded at the proper address using the hardware/software technique of bank switching. Other functions controlled by this software are emulator memory initialization, data integrity verification, and data encoding for transmission to the IM1010 Programmer.

This program is also created on the MPP development system. Unlike the formatter software, however, many of the modules, especially those controlling I/O, are written in assembly language instead of Pascal. Very small native code modules are created using the editor and macro assembler. The larger modules are developed using the development system's Pascal compiler, reverse assembler, editor, assembler, link loader, and Intel formatter programs.

11. System Test

The approach taken to test and debug the completed system is "divide and conquer." The software and hardware components are tested separately to the degree that this is possible. Hardware components are tested using a modular approach.
The power supply is tested first. Then all wiring within the Portable Memory is checked. Before installing any ICS on the printed circuit boards, each socket connection is checked under power for VCC or ground to avoid damaging the ICs.

The 32K memory board is tested using the target microcomputer. Known good target software in ERROM is installed on the memory board, which is connected via the emulation mode umbilicals.

When first powered up, the PM microcomputer is operated at 5MHz to avoid potential high frequency problems. After all areas of the board are functioning, the board is tested at the maximum operating speed of 12MHz.

Synchronous I/O is tested using simple imaging programs. Sophisticated software techniques are avoided in the early stages of hardware testing in order to limit problems encountered to hardware problems.

The interaction between the PM microcomputer and memory board is tested by running I/O test software in the memory board rather than in the microcomputer memory section.

Once the microcomputer hardware is operable, the technique of memory bank switching is tested. A program is devised which loads an EPROM-resident program into 32K RAM at the same physical address as the EPROM test program.
(but in the other bank). Then data integrity is verified by executing the loaded data.

After this program is functioning, the nonvolatility of loaded data in the PM is tested by disabling the "Write Enable" signal at the memory board and running the same program. No data is loaded upon power-up, but the transfer into the 32K RAM still occurs. The test consists of repeatedly turning the PM on and off under these conditions and verifying that the loaded program still works.

The RS232-C port is debugged by transmitting Intel-coded data from the IM1010 to the PM. Software is written to image various status bits within the TMS 9902 asynchronous controller during the load process. The checksum is calculated and displayed at the conclusion of the dump. This checksum is compared to the IM1010 checksum in order to verify proper operation of this mode.

Finally, a program which may be executed by the target microcomputer is dumped into the PM. The emulation mode is tested by running the target board with the PM and verifying proper operation.
12. System Performance

The assembled Portable Memory fulfills the system requirement of making it unnecessary to program EPROMs during the software development process. Application object code may be dumped directly into the PM from the development minicomputer, the data is preserved within the PM under battery power (for many hours if necessary), and the target microcomputer executes this data just as it would if the program were EPROM - resident. Full 12MHz emulation operation is permitted; thus the PM meets the objective of not degrading the performance of the system while it is under test.
12. System Performance

The assembled Portable Memory fulfills the system requirement of making it unnecessary to program EPROMs during the software development process. Application object code may be dumped directly into the PM from the development minicomputer, the data is preserved within the PM under battery power (for many hours if necessary), and the target microcomputer executes this data just as it would if the program were EPROM - resident. Full 12MHz emulation operation is permitted; thus the PM meets the objective of not degrading the performance of the system while it is under test.
CORRECTION

The preceding document has been rephotographed to assure legibility and its image appears immediately hereafter.
12. System Performance

The assembled Portable Memory fulfills the system requirement of making it unnecessary to program EPROMs during the software development process. Application object code may be dumped directly into the PM from the development minicomputer, the data is preserved within the PM under battery power (for many hours if necessary), and the target microcomputer executes this data just as it would if the program were EPROM - resident. Full 12MHz emulation operation is permitted; thus the PM meets the objective of not degrading the performance of the system while it is under test.
References

Technical Manuals


Articles


Vita

The author, Mr. David Helffrich, was born in Allentown, Pennsylvania on January 1, 1951. He graduated from Lehigh University in 1973 with a degree of Bachelor of Science in Electrical Engineering. Mr. Helffrich was employed at the Bethlehem Steel Corporation until 1976. He joined the Bell and Howell Company in Phillipsburg, N.J., in 1977, where he is presently employed as a software engineer.