the IMSAIder

Dear *IMSAI* Owners:

This is the 1st edition of the IMSAIder, a newsletter dedicated to every *IMSAI* owner. Our purpose is to supply you, the user, with all the new ideas, fixes, software updates, and answers to any problems you are encountering. The IMSAIder will be published every 2 months and will be available on a subscription basis to all registered *IMSAI* owners.

This issue of the IMSAIder contains:

We will answer any of your questions in subsequent issues, and we would like to include your ideas and suggestions for better *IMSAI* service.

Please write us or call your IMSAIder staff with your ideas, stories, and or suggestions, and thank you for buying *IMSAI*.

GETTING THE MOST OUT OF YOUR IMSAI COMPUTER

Part One of a Series

Allow me to introduce myself. My name is Seymour Rubinstein and I am the Director of Marketing for *IMSAI*. In an effort to get acquainted, I can think of no better way than to provide you with some helpful information that will increase your enjoyment of your *IMSAI* computer.

As most of you know, the *IMSAI* I-8080 computing system is provided with two paper tapes and accompanying source listing. These tapes comprise a loader and a self-contained operating system (SCS-1). What many of you may not know is that a significantly more capable and powerful extension to this system is available. On your catalog price sheet this is referred to as PGM 6A and is known as **Self-Contained System II (SCS-2)**. Let us take a look at some of the features and requirements of SCS-2.

1. **Memory Utilization:** Except for the first six bytes of memory, SCS-1 begins at location 40H, allowing the user room to put in his own RSTs (Restarts). However, many people feel that this starting address is unsatisfactory. In SCS-2, the starting address is B000H. The resident code for SCS-2 goes from this address to CFFFH. System RAM and assembler symbol tables are provided from D000H through EFFFH. Memory locations beginning at F000H through the top of memory are available for user written I/O drivers to other peripheral devices.

2. **Device Drivers:** As implied by the preceeding, SCS-2 has the ability to allow the user to add **device drivers** in an orderly fashion for the support of additional peripherals. Thus, for those of you doing software development, you can attach a high-speed line printer to the system such as the *IMSAI* PTR-300A or PTR-300B and specify it as the LIST device. Similarly, I/O drivers can be added to support the *IMSAI* SIO-2 board, where one port of the SIO drives a MODEM and another port drives a CRT or teletype. With this arrangement, it is fairly straight forward to use your *IMSAI* as an intelligent terminal interfacing with a large time-sharing system.

3. **Advanced Paper Handling:** For those of you using a teletype, SCS-2 has a number of advanced paper tape handling features as follows:

A. *PLDR and PEOF Commands:* These commands provide for the punching of a leader as well as an EOF record.

B. *INPT Command:* This command allows the inputting of source programs from paper tapes without the necessity of listing the program.

C. *PNCH Command:* This command allows the punching of an Intel standard hexadecimal format loader tape.

D. *LOAD Command:* The LOAD command provides for SCS-2 loading of a standard hexadecimal Intel loader tape.

4. **SCS-2 Assembler:** The SCS-2 Assembler has been enhanced over SCS-1 with the following additional features:

A. Multiple DB operands
B. ASCII strings (as DB operands)

5. **Debugging Facilities:** The debugging facilities provided by SCS-2 represent the most significant area of enhancement. Among the facilities to be found in the *IMSAI* debugger are the following:

A. HEXADECIMAL ARITHMETIC:
When in DEBUG mode, simply type the hexadecimal expression and receive an immediate answer rather than having to work out address calculations with pencil and paper.

B. SEARCH Command: The search command allows you to find occurrences of particular character strings.

C. Area FILL: Allows you to fill areas with ASCII or hexadecimal values without having to type a re-location.

D. *BREAKPOINT Iteration Counts:* This facility allows you to iterate through a loop some specified number of times before the breakpoints occur.

F. TRACE: This feature facilitates a continuously formatted output of all register contents and memory locations affected during program execution.

G. INPUT and OUTPUT: Input and Output directly from/to any port from the terminal.

H. COMPARE Blocks: Compare 2 blocks of memory, and list all differences.

As you can see from foregoing, facilities of SCS-2 make it a really worthwhile enhancement over SCS-1. In addition to the object tape and a complete user's manual, purchasers of SCS-2 are also provided with complete source listings so that modifications and patches as required by the user can be made easily.

If you have found this article helpful, and would like to see more of its kind, or wish to have other topics covered, please write to me and we will try to answer your request.

COMPUTER SHOPPE OFFERS CLASSES IN MICROCOMPUTERS

Where can you take a comprehensive course about *IMSAI* microprocessors? One of your best bets is Computer Shoppe in New Orleans.

Dr. Charles Beck, head of the Computer Science Dept. of Tulane University teaches a fifteen week course (two 3-hour classes per week) as part of his growing dealership with Dan Ellis and John Parnell in New Orleans, Louisiana.

The Computer Course uses Adam Osborne's INTRODUCTION TO MICROCOMPUTERS VOL. 1 as its basic text, along with some "customized" additions created by Dr. Beck and Dan Ellis to meet the specific needs of *IMSAI* 8080 owners. The classes include instruction in assembly techniques, test procedures, applications information, I/O interfacing, software, system planning and system integration.

Dan Ellis says the classes aren't "limited" to any particular group-the student age range is 13 to 54 years and include lawyers, doctors, students, truckdrivers, engineers and programmers. Eleven computer-builders are enrolled in the initial class. A second session was scheduled to begin late in July.

The class fee of $350.00 has a hidden plus in it-10% of the cost of the class may be applied to the purchase of microcomputer equipment or, if you already have equipment from Computer Shoppe, you can have your 10% of that cost deducted from the class fee. Dan assures us that the class helps provide students with more expertise in choosing peripherals as well as an increased ability to debug and trouble-shoot their own computers. So, if you're in New Orleans, check out the Computer Shoppe people at Computer Shoppe.
Tulane University
C/O Dr. Charles Beck
3828 Veterans Ave.
Metarie, LA 70002
(504) 455-7711

MODS AND FIXES By Duane Hentrich Marketing Technician

Our technical staff receives lots of calls concerning modifications to boards and fixes to problems. This column gives us an excellent opportunity to let everyone know what these mods and fixes are. We will be presenting, in the coming issues, all the mods that we have plus any new modifications which may need to be implemented.

The technicians in Technical Services see lots of different and, at times, interesting problems but admittedly we see few of the total systems in the field. So, if you have spent hours tracking down some hardware or software fault, save some other guy the heartache and frustration of having his system down by jotting down the symptoms and solution and sending it to us. This will help us, other customers, and you-when someone else fixes a problem that you have.

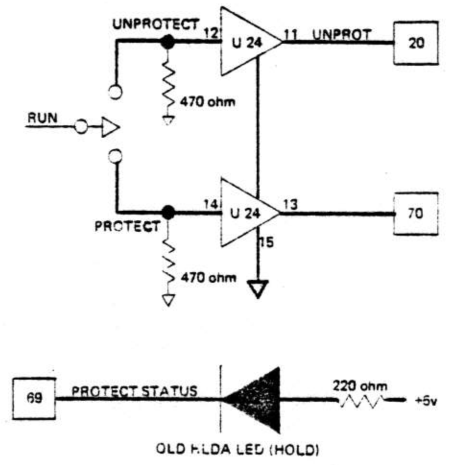
But for right now let's get to some information that may seem old to some of you but great news to others:

I CPA MEMORY PROTECT

The purpose of this modification is to allow the user to WRITE PROTECT or UNPROTECT *IMSAI* RAM 4A-4 boards in 1K blocks from a front panel switch.

Directions:

1. Remove AC cord from wall-socket.
2. Remove AC leads from pads A&B on CP-A, route to a miniature toggle switch mounted on rear of chassis. Connect to center and bottom terminals.
3. Using a solder sucker or copper braid carefully remove the power switch from the CP-A.
4. Cut the following traces:
   a) Between center and upper AC power switch terminal (component side).
   b) Ground lead going to HOLD light(solder side).
   c) Trace to resistor on HOLD light (solder side).
   d) Trace to J1 pin 20 (solder side).
   e) Trace to J1 pin 70 (component side).
5. Install 3-position momentary switch in AC power switch position.
6. Install two 470 ohm, ¼ watt resistors between ground and U24 pins 12 and 14.
7. Install jumpers:
   a) From +5 Volts to resistor from old HOLD light.
   b) From U22 pin 6 to 3-position switch center terminal.
   c) From U24 pin 1 to U24 pin 15.
   d) From bottom terminal of 3-position switch to U24 pin 14.
   e) From top terminal of 3-position switch to U24 pin 12.
   f) From U24 pin 13 to J1 pin 20.
   g) From U24 pin 11 to J1 pin 70.
   h) From cathode of HOLD LED to J1 pin 69.
8. Insert new film label in front panel to change HOLD to MEM. PROTECT and POWER ON/OFF to PROTECT/UN-PROTECT.
9. Insert the following partial schematic on your CP-A sche-matic:



Parts Kits consisting of a paddle switch, a minature toggle switch, 2-470 ohm ¼watt resistors and a front panel label are available from Technical Services to all *IMSAI* chassis owners.

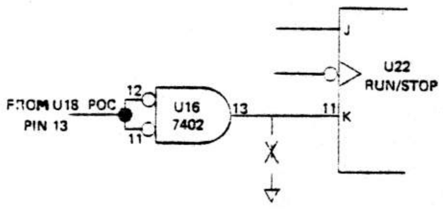
The right hand switch now serves to change the protect status of the currently addressed block of memory (assuming they are *IMSAI* RAM 4A-4's) when the machine is not in RUN mode. The LED which previously showed HOLD status now is lit when the currently addressed block is protected.

II. POWER UP STOP MODIFICATION

The purpose of this mod is to assure that your system will come up in a WAIT state when power is applied to the system. NOTE: not necessary if machine comes up in stop mode. This mode gates POC to the 'K' input of the JK Flip/Flop, U22, which will bring U22 up in a WAIT state.

Directions:

1. If you are assembling the board, before inserting U22 cut the heavy trace between U22 pin 11 and U22 pin 4. If the board is already assembled cut U22 pin 11 from trace on board and let float above board.
2. Add wire from U22 pin 11 to U16 pin 13.
3. Add wire from U18 pin 13 to U16 pin 11 and pin 12.
4. Check all solder joints for good connections and excess or splashed solder.
5. Add the following to your schematic:



III. NON-IMSAI MEMORY MODIFICATION:

The purpose of this mod is to allow the use of non-*IMSAI* memory with the *IMSAI* front panel. This mod gates PWR or $\overline{\text{DEPOSIT}}$ with $\overline{\text{SOUT}}$ to assure that $\overline{\text{MWRITE}}$ will only be active during a Memory Write.

Directions:

1. Cut PC trace that goes from U25 pin 3 to U24 pin 4.
2. Add wire from U25 pin 3 to U25 pin 4.
3. Add wire from U25 pin 6 to U25 pin 9 and pin 10.
4. Add wire from U25 pin 8 to U24 pin 4.
5. Add wire from U25 pin 5 to U16 pin 6.
6. Check all solder joints for good connections and excess or splashed solder.

NOTE: The schematic of CP-A Rev. 4 dated 2/76 shows this modification in place. It is not, however, incorporated in PC Boards which are marked Rev. 4 or earlier. This modification is only necessary for non-*IMSAI* memory. *IMSAI* memory will still operate properly with this mod installed.

IV DEPOSITING TO MEMORY BOARDS USING THE 4200 MEMORY CHIP

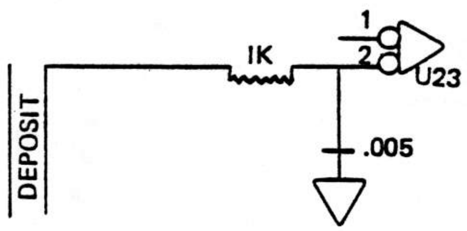
The purpose of this mod is to allow the use of Memory boards using the 4200 memory chip with the *IMSAI* System.

The indications are that you can run programs that write to and read from these boards but cannot deposit to them from your front panel.

This mod slows down $\overline{\text{DEPOSIT}}$ to allow for setup of gates on the memory boards.

Directions:

1. Separate the Deposit signal trace from U25 pin.
2. Insert a 1K ¼ watt resistor between trace and U25 pin 2.
3. Add a .005 mfd capacitor from U25 pin 2 to U25 pin 7 (Ground).
4. Check all solder joints for good connections and excess or splashed solder.
5. Add the following to your schematic:



V RAM 4A-4

Revised Schematic For RAM 4A-4 Rev. 2 or earlier.

The schematic has been redrawn and the section in error has been included below. The 2102 chips were numbered incorrectly, please correct your schematic as shown. There are a few things I would like you to note on the schematic. First, for your information the columns of 2102's are 1K blocks of data.

Therefore:

A16 thru A9 is X000 to X3FF
A8 thru A1 is X400 to X7FF
B16 thru B9 is X800 to XBFF
B8 thru B1 is XC00 to XFFF where X is the board address.

Second , please notice the LED's E0 through E3. These are your green LED's. If you look closely you will see that these LED'S are not only pretty indicators but are used as active pullups for the $\overline{\text{CE}}$ lines to each 1K block of memory. If you have a green LED that doesn't light, REPLACE IT IMMEDIATELY with green, red or any other color LED. A defective LED can cause flakey operation of the board by allowing the 1K block associated with the defective LED to become selected at random intervals.

We hope these mods are a help. If you have others or questions, please write Technical Services.

DYNAMIC RAM AND DMA

The *IMSAI* IFM board REV 6 (used in FIF-Floppy Disk interface and LIF-300 LPM line printer interface) will run in DMA mode with the *IMSAI* 16K, 32K, and 65K dynamic memory boards. A small modification needs to be made to the IFM boards to handle dynamic memory's inherently different timing requirements. Static RAM boards will still operate with this modification. System designers can retain all the advantages of *IMSAI* DMA controllers in the same system with cost effective *IMSAI* dynamic RAM. (The 32K board price is only $749 in kit form). No changes or new signals are required in the S-100 bus definition to enable DMA with the dynamic memory. Any board providing address set up time before the signals listed in the table below, and recognizing a standard false level on the RDY line will run with the dynamic RAM. Refresh is still hidden except for an occasional WAIT state when absolutely required.

Six new lines were added to the S-100 signal definitions to handle addressing the Megabyte Micro Memory System. Address lines 16,17,18, and 19 are on pins 16, 17, 15, and 67 respectively. Pin 67 has been previously used as an address line, albeit with another name.

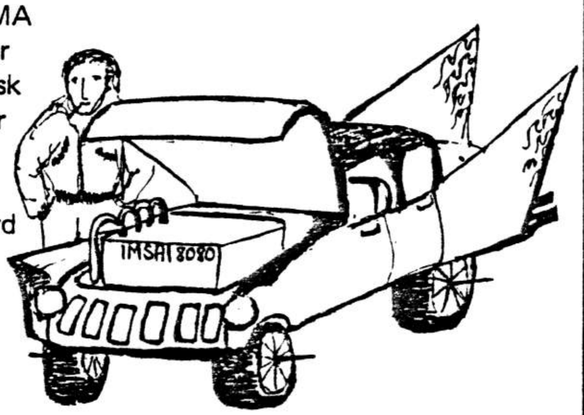
Alternate Address lines 14 and 15 are on pins 14 and 13 respectively. These lines permit complete and powerful control of memory mapping by a separate card. The memory cards can operate on systems without these lines being driven, within a 65K maximum size.

REV 3 or EARLIER IFM BOARDS

To enable those with REV 3 or earlier IFM boards to use dynamic memory with their FIF (or LIF in DMA mode), *IMSAI* extends a special low-cost option to upgrade to the REV 6 IFM. The upgrade costs $100 kit or $150 assembled. For details of the trade contact Technical Services. Note that the

REV 3 or earlier IFM's would require extensive modification to run with dynamic memory.

(**Note:** Although the LIF can DMA complete files from memory for printing, the standard *IMSAI* Disk Operating System Printer driver does not use the DMA feature. Any REV IFM without further modification will run our standard Printer driver).



They told me I could run anything on it.

CYCLE INITIATION CONDITIONS
RAM 16, 32, 65

Read:	**Condition**	**Time**
Normal	Board Enable . MEMR	PSYNC. Ø1 ↑or ↓ jumperable
DMA	Board Enable . STATDSBL	↑PDBIN
Front panel stopped	Board Enable . Run low . /PWR high	after every refresh (to refresh output latches on chips)

Write:	**Condition**	**Time**
Normal	Board Enable	↓/PWR
Front Panel stopped	Board Enable . Run Low . MWRITE high must be valid 1 usec min Data/Add. must hold .4 usec after	edge of internal busy

Board Enable = Address . SINP low. SOUT low. SINTA low

Board Enable must not change during cycle once condition & time met; or during set up time previous to this.

= **rising edge**

= **falling edge**

BASIC-E

The BASIC-E compiler (BASIC E-COM) is now in revision 2.1, the run time monitor (RUN-E. COM) is REV 2.2. This version differs from the previous version (1.4) in the following ways:

BASIC-E.COM REV 2.1

1. Compile options are now specified in the BASIC-E statement instead of as the first line of source code.

   A new option, $F causes the compile listing to be directed to the CP/M LST: device, instead of the sytem CON:.

2 The problem with incorrect compilation of the CLOSE statement has been fixed.

3. The problem with compiling a program longer than one extent (16K) has been fixed.

RUN-E. COM REV 2.2

1. The number of files open at any given time has been increased from 6 to 20.
2. The bug that caused the seventh digit to print improperly (i.e. 81 printed as 81.00001) has been fixed.
3. The VAL function now operates properly.
4. The INP has been fixed.
5. IF END now works for all cases.
6. RUN-E will now accept data files longer than 65K.
7. The problem that caused memory to be "eaten" when doing extensive string manipulations has been fixed.

If any of you working with BASIC-E run across any additional bugs we would appreciate it very much if you would document the problem and send a copy to *IMSAI* Technical Services.

CUSTOMER FIX

We received this letter from Robert Gregoire and are reprinting it here for the benefit of others.

"Last week I wrote to you about an incompatability between your hardware and software i.e. your CRT-2480A terminal and DOS-A. Specifically DOS-A uses CTRL-Z for an EOF marker and the terminal uses CTRL-Z as a clear screen signal.

In the DOS-A User Manual, you recommend setting the terminal (switch on the terminal) "blank screen" to disable. Well, I've found an easier way. I've modified my terminal so that the clear screen signal is CTRL-Y rather than CTRL-Z.

This modification takes about 10 minutes.

Merely (1) On the terminal PCB (bottom side) cut the trace connecting IC No. J2 pin 10 and IC3 K2 pin 13.

(2) Connect a jumper wire from IC No. J2 pin 10 and IC No. L3 pin 2.

That's all there is to it and it works well since CRTL-Y is not used by DOS-A. If you want to know why it works I refer you to the LSI ADM-3 Maintenance Manual Figure 6-1. I'd appreciate it if you put this hint in the next revision of the CP/M users guide page 26 paragraph 9a so that others may benefit.

Robert A. Gregoire"

NOTE: Please send your fixes and suggestions to the IMSAIder. In your letter, please indicate that we have permission to print your letter in the IMSAIder.

TIMING INFORMATION

Mr. Grele kindly sent a copy of this report to Kilobaud and on to us at *IMSAI*. We thank Mr. Grele for this information.

Editor
Kilobaud Magazine
Peterborough, New Hampshire
03458

Sir:

Re the article appearing in your June 1977 publication by Messrs. Rugg and Feldman titled "BASIC Timing Comparisions".

Our store undertook the benchmark tests as specified by Messrs. Rugg and Feldman, using the IMSAI BASIC-E Compiler (DISK BASIC). Two hardware configurations were used. The first was an *IMSAI* I8080 with *IMSAI* memory, (total of 20K) 8080A, the *IMSAI* Dual Drive Floppy, and a DECWRITER II as output. The second configuration was an *IMSAI* I8080 with TDL'S ZPU and 16K memory boards (total of 32K), Digital systems' Dual Drive Floppy and again, A DECWRITER II as output.

The timings were done with a stopwatch using 1/10 second intervals. As the results indicate *IMSAI* Disk BASIC-E performed remarkably well in benchmarks 2-5 compared to the other BASICs. As a matter of fact it had a faster execution time than most of the other BASICs tested. The exception of TRW's running on a CDC Cyber 174, hardly a fair comparision, and Apples Integer BASIC.

IMSAI's BASIC-E is a compilation type BASIC, meaning the object code is executed on a run command, as opposed to BASIC statements being interpreted, as with the Altair or most of the BASICs tested by Messrs. Rugg and Feldman.

R. T. Grele, JRV Corporation

BASIC BENCHMARKS

The following basic comparisons in June of 1977, Kilobaud Magazine published some BASIC timing comparisons by Mssrs. Rugg & Feldman, since *IMSAI* Basic E was not included, JRV Corp. of Hamden, Connecticut undertook to Benchmark test as specified in the article.

A chart of the testing appears below - with the time in seconds.

	BENCHMARKS		Time in Seconds				
HARDWARE	1	2	3	4	5	6	7
IMSAI I8080, 20K IMSAI RAM, 8080A, IMSAI Dual Floppy BASIC-E	6.2	6.5	19.4	17.8	17.1	46.5	72.2
IMSAI I8080, TDL ZPU, TDL 16K RAM (32K); Digital Systems Dual Floppy, BASIC-E	6.1	6.4	18.2	17.0	16.8	45.1	70.4
*Altair Extended BASIC IMSAI I8080, 20K IMSAI RAM, 8080A	2.1	8.7	22.1	24.2	25.7	40.3	61.2

IMSAI DIRECTIONS By Seymour Rubinstein, Marketing Director

In general the atmosphere in a company like *IMSAI* which operates at the very forefront of technology, is incredibly hectic and exciting. Last month was certainly no exception, and for me at any rate, was even more hectic and more exciting than I ever had any business anticipating. From a personal point of view I went from Software Products Manager to Marketing Director just a few days before the NCC in Dallas, Texas. This, plus some other reorganizations in the company, resulted in a frenzied frenzy of activity to bring off the NCC.

In addition to showing off the memory boards we announced a few months ago, three other announcements were made at the show itself.

First and foremost among these is our new **Video Display Board (VIO)**, which is deceptively startling in the incredible number of features and capabilities provided on a single board at an excellent low price. For those of you familiar with the capabilities of the so-called intelligent CRT's on the market, this board can provide your TV set or high quality monitor with not only all of the intelligent terminal features but quite a few other unique features as well. To begin with, the *IMSAI* **VIO produces up to an 80x24 display under program format control,** (yes, 80 characters by 24 lines!), i.e., through simple memory mapped I/O commands you can dynamically change character screen capacity from 80 characters per line to 40 characters per line and from 24 lines to 12 lines. These screen format changes may be requested singly or in combination, thus providing for four different formats, each of which results in a proportional difference in character size. This feature, in addition to allowing the use of different size CRT monitors, allows the programmer to change character sizes for special messages and for viewing at long distances.

Another feature is **character by character, or entire screen, inverse video control**, i.e., white on black or black on white. While the complement of 128 characters is standard, an option provides descenders for lower case characters and a complement of 256 characters which not only comprise the standard 128 character set, but also a line drawing graphics character set. A supplement to this option provides the character sets on three 2708 EPROMS together with instructions to allow you to program your own character FONTS. This feature will be an outstanding aid for those requiring special symbols and foreign alphabets!

So much for the so-called standard characteristics. Now let's get down to the good stuff. The ROM option you can get with this board is an intelligent terminal with features second to none. For example, for text editing and word processing, commands are provided to allow you to do character and line insert and delete. When you insert a character, the text spreads apart to make room. When you delete a character, the text compresses as though the character were never there in the first place. Similarly, when you insert a line, remaining lines still on the screen, are pushed down with the bottom line disappearing. When you delete a line the lines remaining below the line just deleted move up one line. For users with data entry applications, we have implemented protective format capability. This allows you to set up standard business type forms which provide for titled fill-in fields where the space allocated for each fill-in field is predetermined by a user generated format and where the information preceeding the field is protected i.e., is automatically skipped over, during spacing, tabbing and other ordinary typing operations. Additionally, this feature also permits you to set tab stops at any location or locations on the screen. It is interesting to note that after we implemented all of these features our ROM programming genius found he had a lot of available ROM memory left over. So for good measure, if that weren't enough, he threw in a complete 8080 based monitor!

The VIO monitor provides complete facilities for inspecting and modifying RAM locations; filling RAM locations with a predetermined character; searching and printing all locations containing a character string and provide boot strap loading for INTEL hex compatible paper tape and TCOS, our tape cassette operating system. For those of you not familiar with TCOS, this is referred to in our pricing catalog as PGM 2-A, and is an enhanced version of SCS-1 which is the software package that came with your computer kit and which has special commands for saving and restoring files from audio cassette tape when used in conjunction with our MIO board.

Another product we showed at the NCC in Dallas was our I-8080 system, rack-mounted with dual floppies in a standard 19" rack mount cabinet, along with an IBM compatible 9-track 800bpi tape drive. This interesting combination of equipment was put together to demonstrate the capability of using the IMSAI 8080 computer in a commercial environment where utility functions such as tape to print, tape to tape copy, tape edit and tape work stations, including the conversion of Floppy Disk Data to standard IBM compatible tape and vice versa, is important. Of course, application programs to support the manipulation and processing of this data are also quite feasible. Tentative pricing for this product is $5,800 for the tape drive (25 ips at 800bpi) and $2,700 for the controller hardware. Tape drive equipment must be special ordered with a 6 month delivery cycle. The controller hardware will support up to 4 tape drives. Each tape drive and controller combination is furnished with a set of utility software to provide standard copying functions under CP/M, our disk operating system.

Another product that was demonstrated at the NCC was a **full Level 2** (there are no higher levels), **FORTRAN-IV**compiler system. The status of the FORTRAN - IV compiler sytem is that we are still in contract negotiations with the authors of the compiler and hope to complete our negotiations within the next four to six weeks. Following completion of the negotiations, it will be approximately 30 days during which time we will evaluate, document, etc., before we release the product for general sale, Thus, we are looking at being able to fill orders for FORTRAN sometime in the fall. For those of you who are interested, following is a summary of the FORTRAN-IV compiler systems:

1. Generates relocatable code which may be stored in a library.
2. A linking loader program which is part of the package provided for the combining of previously compiled relocatable object modules into load modules.
3. The initial version of the compiler will not have double precision arithmetic, however, this feature will be added within three months.
4. The complete range of FORTRAN-IV statements has been implemented including single and double byte integer, EQUIVALENCE, named COMMON, BLOCK DATA, DATA statements, up to three dimensions in arrays, statement FUNCTION and function subprograms, including INTEGER FUNCTION and REAL FUNCTION. Hexadecimal and octal constants, hollerith strings defined by quotes or H format, and standard print control defined by 1, 0, and +.
5. The compiler generates pure machine code in relocatable form providing high efficiency and easy interface to assembler language routines.
6. Also supplied with the package is an INTEL compatible relocatable assembler. Thus, writing and interfacing assembly language routines with FORTRAN programs becomes an easy matter.

IMSAIder SUBSCRIPTION RATES

*IMSAI*der is a bi-monthly publication of *IMSAI* Manufacturing Corporation and is available to all registered *IMSAI* owners and interested persons. Subscription rates are $4.00 per year (six issues) for Domestic orders and $5.00 per year for International orders.

To order your subscription, please fill out the form below and mail with you check or money order to *IMSAI*, 14860 Wicks Blvd., San Leandro, CA 94577.

Please send me______year(s) of the *IMSAI*der. Enclosed is my check to *IMSAI* Mfg. Corp. for ______ per year. Please send my subscription to the address listed below:

Name:______________________
Company Name:______________
Address:____________________
City:_______________________
State:______________________
Zip:________________________

IMSAIDER CLASSIFIEDS

IMSAIder classified advertising rates are 25 cents per line per issue. Please use the form enclosed for your ad or limit your lines to 26 characters. (please include spaces) Payment must accompany ad copy. **Deadlines for the November December** 1977 issue are **November** 15, 1977.

Please make checks payable to *IMSAI* MFG. CORP. And mail to
IMSAIder
IMSAI Mfg. Corp.
14860 Wicks Blvd.
San Leandro, CA.. 94577

```
REM THIS IS THE IMSAI BASIC E VERSION OF THE GAME   MASTER MIND
REM     2/20/77   AJK   VERSION 1.1
REM
REM     COPYRIGHT 1977 IMSAI MANUFACTURING CORP.
REM     ALL RIGHTS RESERVED, WORLDWIDE
REM
REM
REM
REM VARIABLES:
REM
REM     A$      INPUT & OUTPUT BUFFER STRING
REM     B$      TEMPORARY SINGLE CHAR. STORAGE
REM     C$(I)   GAME PLAYER'S LATEST GUESS, 1 CHAR./POSITION
REM     D$(I)   COMPUTER'S RANDOM STRING
REM     E$      LIST OF ACCEPTABLE CHARACTERS
REM     E       NUMBER OF ACCEPTABLE PLAYING CHARACTERS
REM     F(I)    MATCH INDICATORS FOR D$(I)      0=NO MATCH
REM                                             1=RIGHT CHAR & POSITION
REM                                             2=RIGHT CHAR, WRONG POSITION
REM     G       NUMBER OF 1'S IN F(I)
REM     H       NUMBER OF 2'S IN F(I)
REM     I       LOOP COUNTER
REM     J       LOOP COUNTER
REM     K       LOOP COUNTER FOR GUESS NUMBER
REM     L       GAME COUNTER
REM     M$(I)   MOVE HISTORY
REM     M(I)    MATCH INDICATOR FOR C$(I)       0=NO MATCH
REM                                             1=MATCH
REM     N       TOTAL NUMBER OF MOVES SO FAR
REM     LTH     LENGTH OF RANDOM STRING TO GUESS
REM     DUP     DUPLICATE CHARACTERS OK 0=NOT OK
REM                                    -1=DUPLICATES OK (TRUE)
REM     OFMT    OUTPUT FORMAT, # OF CHAR. BETWEEN SPACES
REM
REM SPECIAL COMMANDS:
REM
REM     ^R      RESTART AVERAGE
REM     ^X      EXIT GAME
REM     ^Q      SETUP NEW PARAMETERS
REM     ^G      PRINT COMPUTER'S STRING & SOUND BELL
REM     ^\      MODIFY AVERAGE
REM
REM
REM INITIAL OUTPUT INFORMATION:
REM
    DIM C$(30),D$(30),F(30),M$(100),M(30)
    PRINT "THIS IS THE IMSAI BASIC E VERSION OF MASTER MIND"
    PRINT
    PRINT "THE OBJECT OF THIS GAME IS FOR YOU TO GUESS THE RANDOM STRING"
    PRINT "OF CHARACTERS I MAKE UP"
```

```
   PRINT
   PRINT "YOU MAKE GUESSES AND I WILL TELL YOU HOW MANY CHARACTERS IN YOUR"
   PRINT "GUESS ARE EXACTLY RIGHT (CHARACTER AND POSITION), AND I WILL ALSO"
   PRINT "TELL YOU HOW MANY OTHERS ARE CHARACTERS I CAN USE TO CONSTRUCT THE"
   PRINT "CORRECT ANSWER, BUT ARE IN THE WRONG PLACE."
   PRINT "IF YOU GUESS MORE THAN I NEED OF ANY CHARACTER, THE EXTRA'S DO NOT"
   PRINT "COUNT FOR MATCHES."
   PRINT "USE A '/' FOR GUARANTEED NO MATCH"
   PRINT
REM
REM INPUT OF GAME SIZE AND OTHER PARAMETERS
REM
REM CHECK FOR STANDARD GAME
REM
8  PRINT
   PRINT "TYPE M FOR STANDARD MASTER MIND - 4 POSITIONS, 123456, DUPLICATES OK"
   PRINT "TYPE S FOR SUPER MASTER MIND - 6 POSITIONS, 12345678, DUPLICATES OK"
   PRINT "TYPE C FOR ENTERING YOUR OWN CUSTOM PARAMETERS"
   PRINT
        INPUT "YOUR CHOICE";A$
        B$=LEFT$(A$,1)
        IF B$="M" THEN\
           E$="123456": E=6: LTH=4: DUP=-1: OFMT=1: GOTO 20
        IF B$="S" THEN\
           E$="12345678": E=8: LTH=6: DUP=-1: OFMT=1: GOTO 20
10      INPUT "FIRST, HOW MANY CHARACTER STRING DO YOU WANT TO GUESS";LTH
        IF LTH>30 THEN\
           PRINT "IF YOU CAN PLAY THAT GAME, YOU CAN MODIFY THE PROGRAM TO":\
           PRINT "PLAY WITH YOU. OTHERWISE, LET'S STAY BELOW 30":\
           PRINT "HERE WE GO AGAIN --------":\
           GOTO 10
REM SET OF POSSIBLE CHARACTERS:
        E$="123456"
        E=LEN(E$)
   PRINT "I'M SET UP TO USE THE ";E;"CHARACTERS '";E$;"' IN MY RANDOM STRING"
        INPUT "TYPE Y OR N, IS THIS OK";A$
        IF A$="N" THEN\
           PRINT "OK, THEN GIVE ME THE CHARACTERS YOU WANT....":\
           PRINT "DO NOT USE A '/' OR A ' '":\
           PRINT "DO NOT DUPLICATE CHARACTERS":\
           PRINT "UP TO 30 DIFFERENT CHARACTERS ARE OK":\
           INPUT "YOUR CHARACTERS";E$:\
           E=LEN(E$)
        IF E>30 THEN\
           PRINT "HEY!  EITHER YOU'RE NOT PAYING ATTENTION, OR YOU'RE JUST":\
           PRINT "GOOFING OFF. NO MORE THAN 30.  LET'S GO AGAIN....":\
           GOTO 10
        INPUT "TYPE Y OR N, CAN I USE DUPLICATES IN MY RANDOM STRING";A$
        IF A$="Y" THEN DUP=-1 ELSE DUP=0
REM MAKE SURE IT'S POSSIBLE
```

```
        IF DUP=0 AND E<LTH THEN\
           PRINT "ALL RIGHT NOW, I DON'T THINK THAT'S POSSIBLE.":\
           PRINT "I'LL GIVE YOU ANOTHER CHANCE":\
           GOTO 10
   PRINT
   PRINT "I PRINT THE CHARACTERS OUT IN GROUPS SO YOU CAN READ THEM"
   PRINT "MORE EASILY. (YOU CAN PUT SPACES IN WHEN YOU TYPE A STRING IF YOU"
   PRINT "WANT, I WON'T COUNT THEM AS CHARACTERS IN THE MIDDLE.)"
   PRINT
        INPUT "HOW MANY CHARACTERS IN EACH GROUP DO YOU WANT";OFMT
        IF OFMT>=LTH THEN\
           PRINT "IT'S OK, BUT NOTE THAT SINCE ";OFMT;">= ";LTH;",":\
           PRINT "THERE WILL BE NO SPACES IN MY OUTPUT."
20 PRINT
   PRINT "REMEMBER - SPACES IN THE MIDDLE OF INPUT ARE OK,"
   PRINT "           - / GIVES A GUARANTEED NO MATCH IF YOU WANT"
   PRINT "AT THE END OF EACH GAME YOUR AVERAGE NUMBER OF MOVES IS PRINTED."
   PRINT
REM
REM FIRST SETUP GAME COUNT
REM
        L=0
        N=0
REM
REM HERE'S WHERE WE COME FOR ADDITIONAL GAMES
REM
30 PRINT
REM
REM SPECIAL COMMANDS INSTRUCTIONS
REM
   PRINT "IF YOU WANT TO RESTART THE AVERAGE, TYPE A CONTROL R"
   PRINT "IF YOU WANT TO QUIT PLAYING MASTER MIND, TYPE A CONTROL X"
   PRINT "IF YOU WANT TO SETUP DIFFERENT PARAMETERS, TYPE A CONTROL Q"
   PRINT
   PRINT
   PRINT "---------- LET'S PLAY!!"
   PRINT
REM   NOW WE'RE READY TO PLAY
REM INCREMENT THE GAME COUNTER
        L=L+1
REM FIRST GET A RANDOM STRING IN D$(I)
        GOSUB 400
REM COUNT THE NUMBER OF MOVES:
        FOR K=1 TO 100
REM GET THE NEXT GUESS IN C(I) AND PRINT IT OUT:
        GOSUB 300
REM CHECK FOR NUMBER OF MATCHES:
        GOSUB 500
REM PRINT NUMBER OF MATCHES
        GOSUB 600
```

```
REM SAVE MOVE AND NUMBER OF MATCHES
        GOSUB 700
REM PRINT A HISTORY OF MOVES THIS GAME
        PRINT
        PRINT
        PRINT
        GOSUB 800
        PRINT
REM CHECK IF WE'RE DONE FOR THIS GAME
        IF G>=LTH THEN\
           PRINT "HOT DAMN..........YOU WIN IN ";K;"MOVES":\
           PRINT:\
           N=N+K:\
           PRINT "SO FAR, IN ";L;"GAMES, YOU AVERAGE ";N/L;"MOVES.":\
           PRINT:\
           K=100
REM CHECK IF THEY'RE DOING OK, GIVE A HINT IF NOT
        IF K=INT(LTH*E/2) THEN\
           PRINT "COME ON, YOU CAN DO IT":\
           PRINT "HERE'S A HINT, THE FIRST ONE IS ";D$(1)
        IF K=INT(LTH*E*3/4) THEN\
           PRINT "LOOKS LIKE YOU COULD REALLY USE SOME HELP!":\
           PRINT "I THINK I'LL GIVE YOU ANOTHER HINT.":\
           PRINT:\
           PRINT "EVERY OTHER CHARACTER IN THE ":\
           GOSUB 480
        IF K=LTH*E THEN\
           PRINT "GOOD GRIEF!!!  TRY GUESSING THE":\
           GOSUB 450
        NEXT K
REM THIS GAME DONE, START NEXT
        GOTO 30
REM
REM CREATE RANDOM STRING OF CHARACTERS TO GUESS
REM
400     RANDOMIZE
        FOR I=1 TO LTH
410     TSUB=INT(E*RND)+1
        IF TSUB=(E+1) THEN 410
        A$=MID$(E$,TSUB,1)
REM ARE DUPLICATES OK?
        IF DUP THEN 420
REM MAKE SURE THERE ARE NO DUPLICATES
        IF I=1 THEN 420
        FOR J=1 TO (I-1)
        IF D$(J)=A$ THEN 410
        NEXT J
REM ADD THIS ONE TO LIST
420     D$(I)=A$
        NEXT I
        RETURN
REM
REM SET UP & OUTPUT THE RANDOM STRING D$(I)
REM
450     J=0
        A$=""
        FOR I=1 TO LTH
        A$=A$+D$(I)
        J=J+1
        IF J>=OFMT THEN A$=A$+" ": J=0
        NEXT I
        PRINT "RANDOM STRING = ";A$;CHR$(07)
        RETURN
REM
REM SET UP & OUTPUT EVERY OTHER CHARACTER IN D$(I) FOR A HINT
REM
480     J=0
        A$=""
        FOR I=1 TO LTH
        IF INT(I/2)=I/2\
           THEN A$=A$+D$(I)\
           ELSE A$=A$+"-"
        J=J+1
        IF J>=OFMT THEN A$=A$+" ": J=0
        NEXT I
        PRINT "RANDOM STRING = ";A$
        RETURN
REM
REM SET INPUT BUFFER TO NULL & GET INPUT
REM
300     A$=""
        PRINT "GUESS # ";K;
        INPUT A$
REM
REM CHECK FOR SPECIAL COMMANDS
REM
        B$=LEFT$(A$,1)
        IF B$=CHR$(18) THEN\
           N=0: L=1: GOTO 300
        IF B$=CHR$(24) THEN STOP
        IF B$=CHR$(17) THEN GOTO 8
        IF B$=CHR$(07) THEN\
           GOSUB 450:\
           GOTO 300
        IF B$=CHR$(28) THEN\
           PRINT "CURRENT TOTAL MOVES = ";N;", TOTAL GAMES = ";L-1;CHR$(07):\
           PRINT "CURRENT AVERAGE = ";N/(L-1);CHR$(07):\
           INPUT "YOUR NEW NUMBER OF MOVES";N:\
           INPUT "YOUR NEW NUMBER OF GAMES";L:\
           L=INT(L+1):\
           PRINT "YOUR NEW AVERAGE IS ";N/(L-1);" IN ";(L-1);" GAMES.":\
           GOTO 300
REM
REM PUT INPUT A$ INTO C$(I), IGNORE SPACES, SHOULD BE LTH CHARACTERS
REM
        J=0
        FOR I=1 TO LTH
100     J=J+1
REM CHECK FOR NOT ENOUGH CHARACTERS
        B=LEN(A$)
        IF J>B THEN GOSUB 110: GOTO 300
        B$=MID$(A$,J,1)
REM PICK UP CHARACTER IF NOT BLANK
        IF B$=" " THEN GOTO 100 ELSE C$(I)=B$
        NEXT I
REM CHECK FOR TOO MANY CHARACTERS
        I=LEN(A$)-J
        IF I>0 THEN GOSUB 140
        GOTO 200
REM
REM ERROR PRINT SUBROUTINES
REM
110     PRINT "COME ON,    ARE YOU INTERESTED IN PLAYING OR NOT?"
        PRINT "WE'RE WORKING WITH ";LTH;"CHARACTERS, NOT ";(I-1);"......."
        PRINT "TRY IT AGAIN ";
        RETURN
REM
140     PRINT "WHAT IS THIS!  I COUNT ";I;"EXTRA CHARACTERS.     ";
        PRINT "I'M GOING TO IGNORE THEM."
        RETURN
REM
REM FORMAT CHARACTERS FOR OUTPUT WITH A SPACE EVERY OFMT
REM
200     J=0
        A$=""
        FOR I=1 TO LTH
        A$=A$+C$(I)
        J=J+1
        IF J>=OFMT THEN A$=A$+" ": J=0
        NEXT I
REM ECHOE MOVE
        PRINT "MOVE # ";K;"   ";A$;
        RETURN
REM
REM CHECK FOR GUESS ITEMS IN CORRECT PLACE
REM PUT 1 IN F(I) IF CORRECT, 0 OTHERWISE
REM PUT 1 IN M(I) FOR EACH C(I) MATCHED TO ANYTHING
REM
500     FOR I=1 TO LTH
        IF D$(I)=C$(I) THEN F(I)=1: M(I)=1 ELSE F(I)=0: M(I)=0
        NEXT I
REM
REM CHECK FOR ADDITIONAL CORRECT CHARACTERS IN WRONG PLACE
REM PUT 2 IN F(J) FOR THOSE
REM DON'T MATCH ANY D(J)'S MORE THAN ONCE
REM PUT 1 IN M(I) FOR ANY C(I) MATCHED
REM DON'T MATCH ANY C(I)'S MORE THAN ONCE
REM
        FOR I=1 TO LTH
        FOR J=1 TO LTH
        IF C$(I)=D$(J) AND F(J)=0 AND M(I)=0 THEN F(J)=2: M(I)=1
        NEXT J,I
REM
REM COUNT NUMBER OF MATCHES TYPE 1 & 2
REM
        G=0: H=0
        FOR I=1 TO LTH
        IF F(I)=2 THEN H=H+1 ELSE G=G+F(I)
        NEXT I
        RETURN
REM
REM PRINT OUT THIS MOVE RESULTS
REM
600     IF G>=LTH THEN RETURN
        PRINT "   ";G;"RIGHT, ";H;"OTHERS OK BUT PLACED WRONG"
        RETURN
REM
REM MAKE AND SAVE PREVIOUS MOVE TABLE
REM
700     M$(K)=A$+"    "+STR$(G)+"    "+STR$(H)
        RETURN
REM
REM PRINT MOVE HISTORY
REM
800     FOR I=1 TO K
        PRINT "MOVE # ";I;TAB(12);M$(I)
        NEXT I
        RETURN
END
```