APPLIED BUSINESS COMPUTER CO.

1885 N. MAIN ST.

ORANGE, CA 92665

Bulk Rate
U.S. POSTAGE
RATE
PERMIT NO.
1231

Applied Business Computer, s

AIM

INTERACTIVE

1885 N. MAIN STREET

ORANGE, CALIFORNIA 92665

(714) 921-0130 921-0131

JUNE, 1982

ISSUE NO. 1

OVERVIEW

Applied Business Computer is proud to announce, that former AIM-65 project engineer Mr. Leo Pardo has joined Applied Business Computer Co. Mr. Pardo was Chief Designer of the AIM-65 and AIM-65/40 microcomputers, his experience and knowledge of the AIM-65 and AIM-65/40 has helped him to develop a more advanced and compatible system to meet the challenges of sophisticated engineering. Applied Business Computer Company's pioneering venture into the independent hardware business became an immediate success with the development of the powerful and advanced ASBC-65 Single Board Computer Controller. Its numerous features allow increased flexibility in software development and the use of numerous hardware packages and peripherals, to make it ideally suited as the foundation of a powerful system.

One of Mr. Pardo's specialties is in industrial control systems. Because of continuing development of the microcomputer coupled with declining costs, the microcomputer is becomming increasingly popular in the industrial world. Microcomputers and microprocessor-based controllers can be used to bring increased productivity to the manufacturing environment. Applied Business Computer, with it's continuing development and research on advance technology can service the industrial community by providing a direct access to present microcomputer technology through it's extensive service, product experience and development department. This is done by carefully studying the requirements of the specific application of the customer and by keeping development costs down maintaining high product reliability.

Applied Business Computer Company's growth and success in hardware technology has forced the company to move from their Anaheim, Ca. location on La Palma Ave. to a new 10,000 square foot facility at 1885 N. Main Street in the city of Orange, Ca. In addition the phone number has been changed. The new number is 714-921-0130 & 921-0131

Due to the immense success of the AIM-65 supported product line and the newly entered of the IBM personal computer product line, more space was needed for research and

new product development. The new facility will also allow all assembly, manufacturing and service operations to be done on the premises.

Recent developments at Applied Business Computer include the ASBC-65 single board computer and monitor modifications allowing you to get lower case letters on the AIM-65 and turning your AIM-65 into a turn-key system. Our products feature innovative designs and a wide range of easy to implement options for flexibility both today and in the future. If you have any questions about our products or are interested in a product or option not described, please feel free to contact us. Our staff is always glad to supply information you desire, while our product engineers are constantly developing new products to meet the demands of an evolving market.

In addition to the new number, a new Modem line will soon be installed in order to make current items of interest available to you through your terminal.

Customer satisfaction remains the foundation of ABCC'S philosophy. Our customers realize all the benefits of a full scale, customer-oriented manufacturer: excellent price/performance ratios, complete documentation and responsiveness to requests for technical support and service.

For the person desiring information on computers for industrial or business application an open line is available : 714-921-0130. All our engineer team is available to provide assistance with your particular application, whether it is the development of a new system or the interfacing of an existing one.

If you have any ideas about the AIM-65 (or IBM Personal Computer) that you want to share, software you would like to see, a new industrial application, etc. or, if you have any ideas on how we can improve our system documentation, please direct all your ideas to:

APPLIED BUSINESS
COMPUTER
1885 MAIN STREET
ORANGE, CALIF. 92665

With the aid of our new extended facility, Leo Pardo and the rest of the staff at Applied Business Computer hope to bring you all of the technical support you need.

NEWS

BASIC DATA STORAGE

The utility "BASHELP CMD" to allow storage and retreival of data from disks during a BASIC program execution is now offered in a EPROM at location $D000 to allow for applications that require a full 4K of RAM for the BASIC program.

EPROM PROGRAMMER IN EPROM

The utility loaded from disks into RAM to control the PP-6432 PROM PROGRAMMER module is offerred in EPROM version at $B000 or $D000. This allows the AIM or the ASBC together with the PP-6432 to be turned into an inexpensive PROGRAMMINMG station without the requirement of the disks.

UPPER / LOWER CASE FOR AIM-65

A modification of the AIM-65 monitor to obtain upper/lower case can be incorporated in the keyboard handler routine. The new code replaces the present code while maintaining all the entry points intact for full compatibility with existing programs.

The new code allows simulation of hardware with Alpha lock key by toggling "Control U". The lower case is specially useful in conjunction with the Editor for text generation. Upon power up the system always comes up with upper case. By pressing "Control U" all Alpha keys generate lower case with access to upper case by pressing the shift key by pressing "Control U" again all Alpha keys generate upper case. The listing of the program is not included in

The listing of the program is not included in this brief newsletter, but it is offered upon request with the cost of handling fee, $5, to the privious customers and $15 for others.

AIM-65 AND ACCESSORY SUPPORT

Applied Business Computer is supporting the AIM-65 not only with software and interface modules manufactured in house but also by providing the AIM-65, AIM-65/40, and the whole family of RM-65 modules. Hardware support includes all the RM-65 line of boards from Rockwell International: fully interfaced floppy controllers, video interfaces, expansion memories, PROM programmers, A/D modules, expansion mother boards, compatible CPU modules and enclosures.

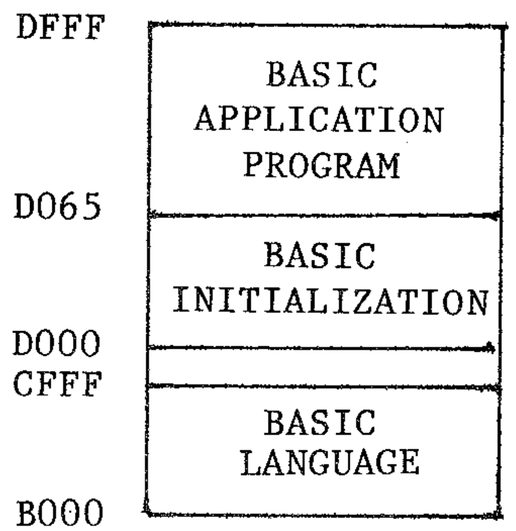
Software Supports includes a powerful operating system, monitor, assembler, editor, PL65, BASIC and FORTH interfaced to the disks. In this manner, Applied Business Computer, as a dealer for the Rockwell product family, can be the single source to satisfy all your needs.

LOADING BASIC APPLICATION PROGRAMS INTO EPROMS

Most applications which are not time critical dependent can be written efficiently and rapidly in BASIC. BASIC is a simple language which is much easier to debug than ASSEMBLY language, resulting in a faster development time. Also the high level instructions allows the user to concentrate on his application. The following description allows the user to put a 4K BASIC application program in EPROM in the AIM-65 Assembler socket (location $D000). This procedure frees 4K bytes of RAM for variables and strings. The procedure is the same for our ASBC Single Board Computer except that 8K of RAM becomes available for variables and strings.

The procedure is outlined in Rockwell's application note R6500 N15 but is described here in more detail with more practical address locations.

BASIC has pointers indicating where the BASIC program and variables reside in memory. Therefore an assembly Initialization program must be executed prior to the BASIC program. The initialization and the BASIC will reside in the 4K starting at $D000, as shown in the following block diagram.



The listing of the BASIC INITIALIZATION PROGRAM at $D000 is shown below. The code can be directly typed or generated using the AIM Editor and Assembler.

BASIC INITIALIZATION

```
0000 0000                ;
0001 0000                ;VARIABLE & STRINGS
0002 0000                RAMBOT  =$0500
0003 0000                                      ; START ADDR
0004 0000                RAMTOP  =$0FFF
0005 0000                                      ; END ADDRESS
0006 0000                ;
0007 0000                        *=$D000
0008 D000  A9 00                 LDA #00       ;PRINTER OFF
0009 D002  8D 11 A4              STA $A411     ; ( OPTIONAL)
0010 D005  A9 E1                 LDA #$E1      ;INITIALIZE
0011 D007  85 82                 STA $82       ;BASIC POINTERS
0012 D009  A2 FE                 LDX #$FE
0013 D00B  9A                    TXS
0014 D00C  D8                    CLD
0015 D00D  A9 4C                 LDA #$4C
0016 D00F  85 00                 STA $00
0017 D011  85 03                 STA $03
0018 D013  85 9C                 STA $9C
0019 D015  85 BB                 STA $BB
0020 D017  A2 87                 LDX #$87      ;ATN VECTOR
0021 D019  A9 BF                 LDA #$BF
0022 D01B  86 BC                 STX $BC
0023 D01D  85 BD                 STA $BD
0024 D01F  A2 87                 LDX #$87      ; USR VECT
0025 D021  A9 BF                 LDA #$BF
0026 D023  86 04                 STX $04
0027 D025  85 05                 STA $05
0028 D027  A9 47                 LDA #$47
0029 D029  85 12                 STA $12
0030 D02B  A9 3C                 LDA #$3C
0031 D02D  85 13                 STA $13
0032 D02F  A2 1C                 LDX #$1C
0033 D031  BD 85 CE     L1       LDA $CE85,X
0034 D034  95 BE                 STA $BE,X
0035 D036  CA                    DEX
0036 D037  D0 F8                 BNE L1
0037 D039  A9 03                 LDA #3
```

```
0038 D03B  85 9B        STA $9B
0039 D03D  A9 00        LDA #0
0040 D03F  85 01        STA $01
0041 D041  85 B0        STA $B0
0042 D043  48           PHA
0043 D044  85 60        STA $60
0044 D046  85 10        STA $10
0045 D048  A9 61        LDA #$61
0046 D04A  85 5E        STA $5E
0047 D04C  A9 B9        LDA #$B9
0048 D04E  85 02        STA $02
0049 D050  A2 6F        LDX #<PGMST     ;BASIC PROGRAM
0050 D052  A9 D0        LDA #>PGMST     ;START ADDRESS
0051 D054  86 73        STX $73
0052 D056  85 74        STA $74
0053 D058  A2 00        LDX #<RAMBOT    ;BASIC VARIABLE
0054 D05A  A9 05        LDA #>RAMBOT    ;START ADDR
0055 D05C  86 75        STX $75
0056 D05E  85 76        STA $76
0057 D060  A2 FF        LDX #<RAMTOP
0058 D062  A9 0F        LDA #>RAMTOP
0059 D064  86 7F        STX $7F
0060 D066  85 80        STA $80
0061 D068  20 7C B4     JSR $B47C       ;CALL RUN
0062 D06B  4C C8 B5     JMP $B5C8       ;1ST STATEMENT
0063 D06E  00           .BYT $00        ;BYTE REQUIRED
0064 D06F               ;
0065 D06F               PGMST   =*
0066 D06F                               ;BASIC START ADDR
0067 D06F               .END
```


Note that the location of the BASIC application program will start at PGMST ($D065) right after the Initialization portion. The user writes and debugs his program as he normally would in RAM. The user must determine where in RAM the BASIC starts so it can be relocated to PGMST at location $D065. To determine the RAM start address for BASIC the user loads the BASIC application program and using the monitor examines locations $73 and $74 as illustrated below.

USING	$73	$74
AIM-65 BASIC ALONE	12	02
FP-950 w. AIM-65 BASIC	03	05
BASHELP. CMD	B4	0B

This vector (locations $73, $74) contains the start address in reverse order of the current BASIC program. For instance, if the FP-950 is used with BASIC therefore value is $503.

By executing the following relocator program all pointer values in the BASIC application program are changed. The BASIC program is not moved, only the pointer values are changed. The new pointer values are adjusted to run the BASIC program starting at $D06F.

BASIC RELOCATOR

```
0000 0000               PGMST   =$D5F6
0001 0000               OLD     =$0BB4          ;LOC $73, $74
0002 0000               ;
0003 0000               ;
0004 0000               ;THIS PROGRAM IS ASSEMBLED TO PAGE 0
0005 0000               ;THEREFORE CODE CANNOT BE PUT
0006 0000               ;DIRECTLY INTO MEMORY WHILE ASSEMBLING
```

```
0007 0000             ;
0008 0000             *=$04          ;USE PAGE 0
0009 0004        NEXT *=*+2
0010 0006        OFF  *=*+2          ;
0011 0008             ;
0012 0008  A9 F6 RELOC LDA #<PGMST    ;NEW PROGRAM
0013 000A  A2 D5       LDX #>PGMST    ;START ADDR
0014 000C  D8          CLD
0015 000D  38          SEC
0016 000E  E9 B4       SBC #<OLD
0017 0010  85 06       STA OFF        ;OLD START ADDR
0018 0012  8A          TXA
0019 0013  E9 0B       SBC #>OLD
0020 0015  85 07       STA OFF+1
0021 0017  A2 00       LDX #0
0022 0019  A0 01       LDY #1
0023 001B  A9 B4       LDA #<OLD
0024 001D  85 04       STA NEXT
0025 001F  A9 0B       LDA #>OLD
0026 0021  85 05       STA NEXT+1
0027 0023  A1 04    J2 LDA (NEXT,X)
0028 0025  11 04       ORA (NEXT),Y
0029 0027  D0 03       BNE J1
0030 0029  4C A1 E1    JMP $E1A1      ;RETURN TO AIM
0031 002C  18       J1 CLC            ;MONITOR
0032 002D  A1 04       LDA (NEXT,X)
0033 002F  48          PHA
0034 0030  65 06       ADC OFF
0035 0032  81 04       STA (NEXT,X)
0036 0034  B1 04       LDA (NEXT),Y
0037 0036  48          PHA
0038 0037  65 07       ADC OFF+1
0039 0039  91 04       STA (NEXT),Y
0040 003B  68          PLA
0041 003C  85 05       STA NEXT+1
0042 003E  68          PLA
0043 003F  85 04       STA NEXT
0044 0041  4C 23 00    JMP J2
0045 0044              .END
```


The BASIC program is still in low memory. Therefore it must be dumped to disk or other type of storage.The BASIC program is then apended to the Initialization program and burned into a 2532 EPROM. This can be easily done using the PP-6432 EPROM programmer.

Starting the BASIC program can be accomplished by typing N. Automatic execution upon power up can be obtained by changing 3 bytes of the MONITOR ROM as explained below.

In order to start execution of the user's BASIC application program from power up, the following bytes in the monitor must be changed.

ADDR	$E142	$E143
CURRENT DATA	$72	$FF
NEW DATA	XX	YY

YY XX is the address where the user's Initialization starts. For instance, in the previous example: YYXX = D000.

FULL CONTROL OF BASIC

APPLICATION PROGRAMS

Running an application program written in AIM-65 BASIC may have disastrous results if the operator hits a carriage return (CR) in response to an INPUT statement. The BASIC program will stop execution and the system goes back to the interactive mode. It is very important to have full control of the machine. For turn key applications where the user does not want to stop execution of his application program. One way around this problem is not to use INPUT statements and use GET statements instead. But this solution could be impractical because so many conversions and setting of variables have to be done by the user.

A) The most practical solution is to change 2 bytes in the AIM-65 BASIC ROM set and burn a new EPROM. The change is as follows:

ADDR	$B9E5	$B9E6
CURRENT DATA	$6B	$B6
NEW DATA	$F5	$B9

This allows the user to maintain full control of the program for INPUT statements in the case of a CR entry.

B) Note that this correction still does not prevent the BASIC program to jump back to the Monitor on an ESCAPE entry. The BASIC calls the READ ($ E93C) routine in the AIM monitor. This situation can be eliminated if the following patch to the AIM monitor is done.

ADDRESS	$E956
CURRENT DATA	$20
NEW DATA	$60

This prevents the monitor from recognizing any ESCAPE keys by the READ subroutine.

AIM FORTH WITH DISKS

FORTH is a language well suited for Industrial applications . Its modularity and extensibility allows for fast development and easy debugging. The FORTH on the AIM-65 as introduced by Rockwell handles data manipulation through cassettes . Although this may be enough for some applications it is not suitable for program development and other applications. Applied Business Computer is the first to offer a fully compatible interface to disks for the AIM-65 FORTH. This allows the user to speed up development time since debugging can be done one "word" at a time. FORTH interface to the disks is accomplished by executing a utility program. After that the user can load FORTH programs directly from disk such as our SCREEN-ORIENTED EDITOR. This program requires our CRT-80 and our FP-950 controllers.

Another software package which is under development is our own FORTH language with a Select compiler. Unlike other FORTH compilers such as the ones called TARGET and META compilers, the user only specifies the higher hierarchy words to the select compiler. This select compiler then figures out the lower level words used and moves them over to another portion in RAM for PROM burning. In this manner there is no overhead in the final product since only the words being used are compiled.

LOADING FORTH APPLICATION PROGRAMS IN EPROMS

For a turn key application system is often desirable to burn a FORTH application program in PROM without requiring compilation from mass storage. The application program is usually ran upon tuning the system on. The following describes the steps to generate FORTH code program to be burned in a EPROM and installed at $D000 (Zockett Z24). A 4K byte program ($D000 to $DFFF) in FORTH is usually more than enough for most applications since compiled code is very compact.

The procedure to generate FORTH code at $D000 requires some means of having RAM memory at $D000. One way of having all the RAM required is by the use of the MB-64 memory module. The procedure outline here also describes FORTH code generation using the disks through the FP-950 controller. The FORTH application program is first compiled and debugged at low memory by loading the appropiate screens from disk.

To relocate the FORTH application program to $D000 all that is required is to change the dictionary pointer (DP) to $D000 and to load (compile) the application program. One problem is that the FORTH compiler checks the value of DP against the value at UFIRST (start up DISK BUFFERS) to see if there is enough memory. Therefore, the value of UFIRST must be set A0 locations higher than the last value of DP ($E0A0). But if UFIRST is altered, the system must be prevented from using a buffer at $E0A0, (ROM). One way to do this is to switch UFIRST to the value E0A0 at the beginning of each screen and reset it to the old value at the end of the screen

The following program defines two words ".." and "-->" to switch the value of UFIRST for each screen. with the SCREEN and address in the stack before calling reolocate. This will load the application program screens into the address specified. For instance if your application program starts at screen 5 you will type "HEX 5 D000 RELOCATE". Each of the user screens must contain the words ".." as the first word and "-->" at the end.

In addition, the first screen of the user application program must contain a small ASSEMBLY start up routine

as shown in screen 3. The last screen must contain the code illustrated in the last lines of screen 4.

To run the following example the user types:

```
2 LOAD      OK
RELOCATE    OK
3 LOAD      OK
```

SCREEN 2

```
0  HEX   ( *** RELOCATOR WORDS *** )
1  0 VARIABLE NUFIRST
2
3  : ..  ( CHANGE UFIRST TO NUFIRST FOR EVERY SCREEN )
4     UFIRST @   ( SAVE IT )  NUFIRST @ UFIRST ! ; IMMEDIATE
5
6  : FIRST-BACK  ( CHANGE UFIRST BACK TO OLD VALUE )
7     UFIRST @ NUFIRST ! ( SAVE FOR NEXT SCREEN)  UFIRST ! ;
8
9  : RELOCATE D000 DP ! E0A0 NUFIRST ! ;
10
11  : -->  FIRST-BACK [COMPILE] --> ; IMMEDIATE
12  : ;S   FIRST-BACK [COMPILE] ;S  ; IMMEDIATE
13
14  FIRST
15  DECIMAL ;S
```


SCREEN 3

```
0  ..  ( FIRST APPLICATION PROGRAM SCREEN )
1  HEX ( *** ASSEMBLY INITIALIZATION DRIVER *** )
2
3   ASSEMBLER  C28F JSR, C2CE JSR, 00 # LDA, 305 STA,
4        00 # LDA, 306 STA, 6 # LDY, UP )Y LDA, TAX, B535 JMP,
5   FORTH   : 1ST ;    ( DUMMY LINK TO LAST WORD IN FORTH )
6  DECIMAL    ( END OF DRIVER )
7
8
9  (  ---  USER DEFINITIONS --- )
10
11  : TEST ." THIS IS A TEST WORD FOR THE RELOCATOR " ;
12
13  -->
14
15
```

SCREEN 4

```
0  ..  ( LAST APPLICATION PROGRAM SCREEN )
1
2  (  ---  )
3
4  ( *** SET PROPER LINK VECTORS *** )
5  HEX
6     ' .S NFA   ' 1ST LFA !     ( SET LINK OF 1ST TO .S WORD )
7    300 DP !   : TASK ;         ( SET TASK AS LAST WORD AT 300 )
8    305 C@  D007 C!
9    306 C@  D00C C!             ( SET DRIVER TO POINT TO TASK )
10  ;S
11
12
13
14
15
```

Relocate must be executed to set DP & NUFIRST. Note that definitions after ".S" will desappear since "lst" is linked directly to the last word in ROM, ".S". The user can do a v list to verify that the vocabulary contains the user words. The code at $D000 can now be stored into the disk using the monitor and burn into EPROM as explained in section of the AIM FORTH user's MANUAL.

Automatic execution of the last user defined word upon power on can be accomplished by setting the IP vector in the Assemble driver. IP must point to CFA of the last word which looks like:

```
: APPLICATION EMPTY-BUFFERS DECIMAL DOAPPLICATION ;
```

To find the address of the word APPLICATION the user simply types:

```
' APPLICATION PFA .
```

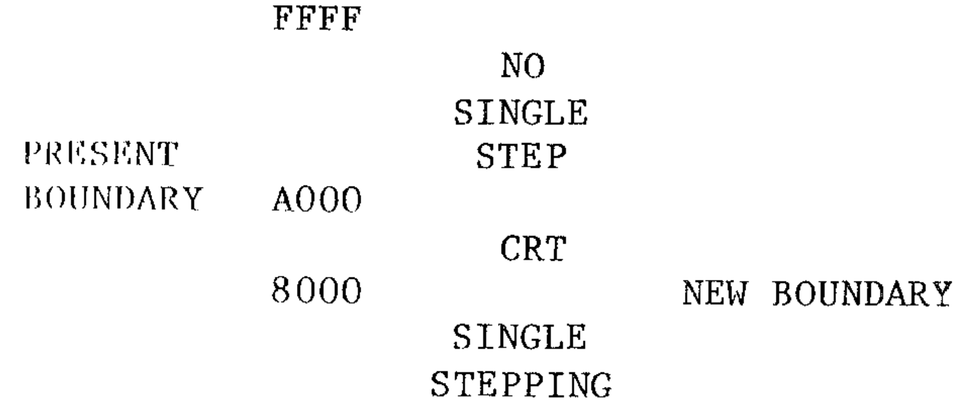
The assembly code then must be modified to store this found value into IP and IPH.

SINGLE STEP USING VIDEO CONTROLLERS

Using the AIM-65 single step feature can prove to be very useful in debugging software programs. The single step feature allows the execution of the user's program one instruction at a time. In this manner the user can examine registers, memory, and I/O variations as a result of executing one specific instruction. Display of register contents and disassembling of following instructions is given by the AIM-65. One draw-back is that the AIM will single step through any program located below address $A000. Programs located at addresses $A000 through $FFFF are not single stepped to enable the AIM-65 to display registers and disassemble the user program.

If a video display driver such as the CRT-80 is added (at location $8000), the single step feature cannot be used. Single stepping through the user program will generate calls to the video controller driver to display registers, but instructions executed for the video driver will also generate single step calls which generates an infinite loop.

One simple way to overcome this problem is to move the single step boundary to $8000. This allows programs residing at $8000 - through $FFFF to be executed at full speed.



In order to accomplish this all the user has to do is to connect a jumper from device Z13 pin 9 to Z13 pin 7.

ASBC SINGLE BOARD COMPUTER

The ASBC single board computer is designed as an economical solution for OEM and Industrial applications. The ASBC is ideal for Data collection terminals, medical instrument controllers, Instrument testers, Remote terminals , Motor controllers, Engine testing and control, Frequency countergenerator and more. The board was introduced by EDN magazine in March 31 1982 issue.

The ASBC is hardware and software compatible with Rockwell's popular AIM-65. Yet the ASBC-65 features numerous improvements over the AIM-65 such as: up to 32K bytes of on-board memory divided into 8K of RAM (NMOS or CMOS) and 24K

of ROM/PROM; on-board back-up battery; true RS-232C serial interface with programmable baud rates from 50 to 19200; fully buffered expansion bus; Interrupt driven keyboard; smaller size and a much lower starting price of $ 230 at OEM quantities.

Retention of 8K bytes of Data after power off allows the ASBC to be used as a smart terminal for collection of Data without the need of cassettes or floppy drives. Data can be transmitted later via the RS-232C port to a host system for processing.

Programmable baud rates from 50 to 19200 make the ASBC suitable for slow transfer rate applications such as interfaces to modems, as well as to very fast transfers of Data from computer to computer or terminals. A total of 76 programmable I/O lines of which 12 can be edge triggered detects, plus 5 timer, provides enough I/O capability to turn the ASBC into a very smart controller. The keyboard interface allows for operator control and Data entry.

The ASBC contains sufficient on-board memory as well as I/O to handle most applications. However, if the user should require to interface the ASBC to additional software/ hardware modules, a fully buffered expansion bus is offered. This bus is EXORciser compatible. The EXORciser bus is supported by our extensive and growing line of system modules such as the FP-950 floppy controller, CRT-80 video controller, MB-64 dynamic RAM module, PP-6432 PROM Programmer and other EXORciser compatible boards.

The ASBC with the addition of the supporting modules mentioned above and the ADOS operating system, becomes an inexpensive development system. Observe that the heart of the development system is identical in nature to the ASBC target system. This feature reduces the development cycle since no down loaders or emulators may be required. The software can be tested immediately.

AIM-65 TURNKEY DEVELOPMENT SYSTEM

The AIM-65 microcomputer is a great machine for the novice in the realm of microcomputers. The machine allows development of assembly programs using the editor and assembler. Serious programmers may find very soon that for longer programs the single line display and cassettes are not very confortable. Applied Business Computer offers a fully intergrated video controller to display 80 characters by 25 lines as opposed to 20 characters by 1 line. A floppy controller is also offered to handle up to 4 double sided double density drives with a maximum capacity of 1.2 megabytes . The floppy controller with it's operating system as well as the Centronic's type printer port on board, are fully integrated with the AIM-65. Additional modules allow the memory capacity to be expanded through our 64K dynamic memory board.

All modules are initialized by executing a jump instruction to $8000 upon power up. The user may want to have this done automatically, in which case replacement of code at location $E141 to 4C 00 80 is required. Substitution of this code can be easily done by reading the ROM at Z23 into RAM memory, changing the three bytes and burning into an 2532 EPROM with a PROM PROGRAMMER such as our PP-6432.

IBM PERSONAL COMPUTER SUPPORT

Applied Business Computer recently introduced several support modules for the rapidly growing market of the IBM personal computer. Our desire to provide our customer with quality products at reasonable prices helped us decide to support this competitive market.

Some of the modules introduced include 1/2 megabyte expansion memory, 18 megabyte hard disk subsystem, EPROM programmer, screen oriented Editor and 64K CMOS battery backed up memory.