IBM

General Systems
Division
P. O. Box 2150
Atlanta, Georgia 30301

IBM 5110
Computing System

73-1

January 10, 1978

IBM 5110 NEWSLETTER

Today IBM announced an exciting new, low priced data processing system for the small and large business: The IBM 5110 Computing System. This system combines the experience gained from the popular IBM 5100 with technological improvements to provide capabilities for handling a wide variety of commercial and problem solving applications.

While the IBM 5110 contains new functions and improved performance, it has the interactive languages and ease of use characteristics familiar to the IBM 5100 user. This newsletter focuses on the following topics relative to this new product:

- New Functions and Features

- A Comparison of Features—IBM 5100 vs IBM 5110

- A Comparison of Performance—IBM 5100 vs IBM 5110

- Converting IBM 5100 Programs to the IBM 5110

- Binary Floating Point Arithmetic Considerations

Future issues of this newsletter will contain tips and techniques about using these features. The objective is to provide you with information that will aid you in the productive use of your IBM 5110 Computing System.

NEW FUNCTIONS AND FEATURES

Faster internal performance, enhancements to both the BASIC and APL programming languages, and a high speed diskette attachment highlight the new IBM 5110 capabilities. The IBM 5110 Computing System has many new functions and features designed with the user in mind and with ease of use as the objective.

Ease of Use Factors

- Full Screen Edit

  Provides the ability to read and write anywhere on the display screen. It can help in the implementation of applications that operate in a conversational manner. In developing an order entry application, for example, a form can be displayed on the screen to allow the operator to merely enter data (fill in the blanks) for that order. This will significantly simplify the data entry process and minimize the chance of error.

- Upper and Lowercase

  The 5110 can display and/or print alphabetic characters in both uppercase and lowercase.

- Procedure Files

  You have the ability to build procedure files in the BASIC and APL languages. This will allow programs to be strung together. The system will treat input from these files exactly as it would treat input from an operator at the keyboard. This will assist in designing applications to be run in an unattended mode, and in simplifying the operation of the system.

- Audible Alarm Feature

  An audible alarm is available under program control. This will be useful in cases where the system is left to operate unattended while the operator is performing other functions. The audible alarm can be programmed to indicate the completion of particular tasks or error conditions.

Physical Characteristics

- Direct Access Storage

  The system can be configured to include from one to four diskette drives, each drive, using the IBM Diskette 2D, with a capacity of 1.2 million bytes of storage. These easily removable and storable diskettes provide you with virtually unlimited offline storage capacity.

The capability to use the diskette units in a sequential or direct access fashion can significantly ease system design and operation. The ability to inquire and update against individual records will mean that you can use the system in a mode similar to your manual operation.

-- Data Exchange

The diskettes, when written in the standard interchange mode, also provide a means of exchanging data with other IBM systems having diskettes (eg, 3741, System/32, System/34, and Series/1).

-- Communications Options

You now have a choice of either asynchronous and/or bisynchronous communications on your 5110 system. The bisynchronous communications capability allows you to emulate either a 3741 or 2770 at speeds up to 4800 bps. In asynchronous mode the 5110 emulates a 2741 terminal at both 134.5 and 300 baud.

-- Specialized Input and Output

The 5110 Computing System has optionally available both a Serial Input/Output Adapter and a Parallel Input/Output Adapter (conforms to IEEE Standard 488-1975). Through the use of these features you can add to and control from your 5110 other input/output devices such as plotters, special purpose instrumentation, graphic displays, etc.

-- Customer Support Functions

A comprehensive set of customer support functions are provided with the 5110. These functions provide common routines required by most data processing systems. They will allow you to initialize a diskette, copy from one media to another, and help manage your various data files.

-- Diskette Sort Feature

A common requirement of a commercial data processing system is the ability to sort records. Available as a 5110 option is a diskette sort feature that eliminates the need for you to write your own sort routines.

-- Printer Overlap

The 5110 overlaps printing with calculations. This provides you with improved performance in those jobs where printing on the IBM 5103 printer can be performed while calculations are taking place.

-- Tape Update in Place

This capability adds flexibility and increases performance for 5110 tape operations.

Additional Programming Enhancements

The BASIC and APL programming languages have been significantly extended over that available for the IBM 5100. Following are some highlights of the increased flexibility and power of 5110 BASIC.

-- The DIMension Statement allows you to enter any size array as long as it fits into memory. You can also specify the number of characters for a character variable from 1 to 255 (the default is 18).

-- Error Trapping by the user program can identify most errors. Some examples of errors you could branch on are, End of File, Input/Output Error, Data Conversion.

-- The new FORM Statement can be used in place of an IMAGE statement adding among other uses the capability to automatically float a $ (dollar sign), *, insert commas in numbers over 999.99, and convert a six-digit date to the more familiar MO/DY/YR format.

-- A new system function enables sorts to be performed very quickly in memory.

A COMPARISON OF FEATURES--IBM 5100 vs IBM 5110

Function	5100 Portable Computer	5110 Computing System
Languages		
BASIC	Yes	Yes, with enhancements
APL	Yes	Yes, with enhancements
Tape cartridge	Yes	Yes
Diskette	No	Yes (1.2-4.8 MB)
Media exchange	No	Yes
Communications		
Start/stop	Yes	Yes
Bisync	No	Yes
Print speed	80-120 CPS	80-120 CPS (overlapped)
Print overlap	No	Yes
Upper and lower case	No	Yes
Full screen support	No	Yes
Serial I/O	Yes	Yes
Parallel I/O	No	Yes

A COMPARISON OF PERFORMANCE--IBM 5100 vs IBM 5110

The total time it takes to run a job on any system is a function of many factors. However the 5110 offers significant enhancements in many areas when compared to the 5100.

-- File input/output processing

	5100/5110 Tape	5110 Diskette	Ratio
Read (KB)	2.85	32.8	1:12
Write (KB)	0.95	18.9	1:20
Data transfer (KB)	4.00	62.5	1:16
Average access (Sec)	30.00	0.2	1:50

In addition to the increased speed of the diskette over the 5100 tape, the capability to update data files in place makes application design simpler and allows for better throughput in many application areas.

-- Internal compute

The internal compute power of the IBM 5110 is approximately two to three times as fast as the 5100. Additionally, under program control, the user can turn the CRT on or off. When the CRT is off, up to 18% of the total CPU processing time becomes available to the user rather than being used to refresh the CRT. Since there is no defined relationship between internal performance and overall throughput, no throughput conclusions should be drawn from this information without an actual comparison for a given job.

-- Printer overlap

Print time on the 5110 can be overlapped with processing time, thus enhancing throughput on many programs.

CONVERTING IBM 5100 PROGRAMS TO THE IBM 5110

Conversion of APL Programs

Guidelines for use of 5100 generated APL workspaces on the 5110 are as follows:

1 Only workspaces stored with the)SAVE command on the 5100 can be)LOAD'ed or)COPY'ed onto the 5110. 5100)CONTINUE'd workspaces (file type 6) and the 5110)CONTINUE'd workspaces (file type 26) are not compatible.

2 The 5110 requires 224 more bytes of overhead from the user work area than did the 5100 (6916 bytes vs 6692 bytes). Some programs which were running close to workspace full may not be able to run on the 5110.

3 Shared variable data files on diskette require a simple or complex name, tape files still do not. However, any program written which permits the user to specify the device/file number and defaults to a tape file name of 'DATA' will not run on diskette.

Conversion of BASIC Programs

Any syntactically valid 5100 BASIC statement will be syntactically valid on the 5110. Such a program may run and give compatible results on both systems. However, due to several considerations discussed in this write-up some conversion may be necessary to enable the running of a given 5100 program on the 5110.

The IBM 5110 BASIC is SOURCE code compatible with that of the 5100. This means that programs stored in SOURCE format by use of the SOURCE parameter on the SAVE command can be transferred directly to the 5110 (ie, SAVE 3, SOURCE).

Most existing programs however are not saved in SOURCE, but rather via the 5100 SAVE default option of internal (ie, SAVE 3). Internal means that the program is stored in a compressed state. This compressed state differs between the 5100 and the 5110 and in some cases takes less storage than the SOURCE option. However, since 5100 internal code is different from that of the 5110, programs are not transferable in internal format.

In order to get programs on the 5100 into SOURCE format, the programs must be LOADed on the 5100 and SAVEd in SOURCE. This must be accomplished before moving the tapes to be converted to a 5110 Computing System.

With the programs in SOURCE format you can LOAD them on the 5110. Programs LOADed after being SAVEd in SOURCE are line by line syntax checked, as though they were being keyed in by hand, while being displayed on the CRT. Lines greater than 64 characters in length may exist. These have been created on the 5100 by deleting blank spaces and leading zeros in statement number, in creating BASIC statements. When SAVEd in SOURCE and then LOADed proper syntax is followed, therefore blanks and leading zeros are part of the statement and the total statement length may exceed 64 characters. When this error situation occurs during LOAD, the screen will blink. The user may then modify (by removing extraneous blanks) or change the line and then press execute to continue loading.

In addition, other incompatibilities exist that may affect the user when converting programs from the 5100 to the 5110. These include:

-- Invalid file names. Files on the 5110 must abide by the diskette naming conventions. (See BASIC Reference Manual Chapter 2 File Reference Parameters.)

-- Since the internal character set on the 5110 is different than the 5100 any program which uses a HEX constant (ie A$=X'73') may get different graphics for those characters. Note especially that the characters for new line, end of record, and % (X'E3', X'FF', and X'73' in 5100) have changed for the 5110 (X'15', X'1E' and X'6C'). The first two of these characters would have been used by a program which generated another program or by Serial I/O programs.

-- Device addresses should be changed to take advantage of diskette capabilities.

Conversion of data files can be handled by the Tape to Diskette Copy program. (See Customer Support Functions Reference Manual.) Be aware that file names longer than 8 characters on tape will be noted by the program. The user will have to input a file name consistent with diskette file naming conventions.

Additionally the user should be aware of the following differences between the IBM 5100 and IBM 5110:

1 The internal numeric data format on the 5100 is a packed decimal floating point format, while the data format on the 5110 is a binary floating point format. In many situations this difference will cause slightly different results on the 5110. Be sure to review the Binary Floating Point Arithmetic Considerations write-up in this newsletter.

2 The treatment of local variables in user-defined functions is slightly different in the 5110 versus the 5100. Given the following set of circumstances:

A FNA invokes FNB directly or indirectly.

B FNA uses X as a parameter and local variable.

C FNB does not use X as a parameter but does refer to variable X.

D X is defined globally.

In the above case, when FNB is invoked via FNA the value of the variable X when referred to in FNB will be the value of the parameter X to FNA. When FNB is invoked without going through FNA the value of the variable X in FNB will be the global variable X. In 5100, the value of the variable X in FNB was always the value of the global variable X.

3 5100 BASIC allowed the user to specify a negative value for a file number to allow the user to create a TYPE 2 data file. Since all stream files are created as TYPE 2 on the 5110, it is no longer necessary to provide this interface. If a negative file number is specified on the 5110, it will cause an error when the file is opened. Again, this function would have been used primarily by programs which generated other programs.

4 OPEN statements on the 5110

A File names--5110 tape files follow 5100 file name conventions. 5110 diskette file names have more restrictive conventions. (See Chapter 2--File Reference Parameter--BASIC Reference Manual.)

B File name usage in OPEN statements differs. On OPEN with IN the file name parameter is used as a check that the correct file has been referenced. You cannot use the file name parameter to receive the file name for checking by the program (common practice on the 5100). On OPEN with OUT the file name parameter is used to provide a file name for file type 0. Further, for file type 0 on a diskette the file name parameter must be used. For other file types the name in the file name parameter is checked against the existing file name on the tape or diskette.

5 User-defined keys must be listed, and rekeyed in their entirety on the 5110, as they cannot be saved in SOURCE on the 5100.

6 Programs which just barely fit into 5100 RAM may not fit on the 5110. The amount of system overhead has increased by 241 bytes from 4383 bytes to 4624 bytes. Borderline programs must be investigated one by one.

7 When using Keyboard Generated Data Files the word DATA was required as a separator between the statement number and the data on the 5100. On the 5110 the word DATA has been replaced by the colon (:). Therefore an additional 5 characters can now be keyed per line of a Keyboard Generated Data File.

8 Use of the CHAIN statement on the 5100 causes a single line feed to the printer. On the 5110 it does not.

9 Character variables on the 5100 when used with the USE statement are initialized to blanks. On the 5110 they are binary zeros.

10 The use of RESET END with a Stream I/O file on the 5100 after the file has been OPENed for either IN or OUT operations causes the system to implicitly CLOSE the file, position the pointer to the physical end of the data and implicitly OPEN the file for OUT operations. On the 5110, use of RESET END with Stream I/O files, after the file has been OPENed for IN, provides the same results as does the 5100. That is the file is implicitly CLOSEd, the pointer set to the physical end of the data, and the file OPENed for OUT. If the file on the 5110 has been OPENed for OUT, issuing a RESET END will cause an error.

11 When using CHAIN on a 5100, pressing ATTN while the tape is moving between the files will cause the tape to stop. On the 5110, the pressing of ATTN while the tape drive is in operation between programs will have no effect. In order to stop the tape from moving, it must physically be removed or the user must wait until it finds the program being CHAINed to.

Note this explanation does not discuss at all the recoding of 5100 programs to take advantage of the numerous 5110 language, functional and performance enhancements.

BINARY FLOATING POINT ARITHMETIC CONSIDERATIONS

In the decimal system used by the 5100 the number 32.25 is represented as

32.25

$$(3X10)+(2X1)+(2X\frac{1}{10})+(5X\frac{1}{100})$$

In the binary system used by the 5110 the number 32.25 is represented as

100000.01

(1X32)+(0X16)+(0X8)+(0X4)+(0X2)+(0X1).(0X1/2)+(1X1/4)

In both systems the number (32.25) can be precisely represented.

If the number was 32.3 a different situation exists in binary. The 32 portion of the number is again shown as 100000. The .3 however presents a problem in binary. The fractional part of a number is represented as the sum of some of the following:

1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 ...etc

For .3 it would be 1/4+1/32+1/64+1/512+1/1024 ...etc.

In other words, the binary fraction for .3 will never be an exact representation of the number. A similar problem occurs when we try to determine what 1/3 of a number like 10 is in decimal. We can indicate that the result is 3.3333 but no matter how many 3s are added the result is not precisely equal to 10 divided by 3.

For output the 5110 converts back to decimal so that we can read the number easily. As part of the conversion the number is rounded and the output you would see is 32.3.

The important consideration is that even though the number stored in the 5110 is only approximately .0000000000001 different from exactly 32.3 it is in fact different internally.

Assume that the 5110 is to read some numbers representing how an employee spent his time on various projects during a day. Management expects the entire 8 hours to be accounted for. If the 5110 receives the following numbers:

2.3, 2.5, 3.2

and adds them (the result is 8) it would have a number very slightly less than 8 or 7.999999. If the program compares the calculated number to 8 the 5110 would indicate that 8 hours were not accounted for when in fact they were.

Since whole numbers can be exactly represented in the binary system a simple way to effect exact comparisons is to multiply both numbers by 100 after rounding, and then making the compare on the integer portion:

In other words, instead of:

```
     If A=B GO TO 200
```

use:

```
     If FNC(A)=FNC(B) GO TO 200
```

where:

```
     DEF FNC(X)=INT(100*(X+(SGN(X)*.0001)))
```

A similar situation occurs when the INT function is used:

```
     INT(8)=8
     INT(7.999999)=7
```

To correct this:

Use:

```
     FNI(X) rather than INT(X)
```

where:

```
     DEF FNI(X)=INT(X+SGN(X)*.0001)
```

so that FNI(7.999999)=8.