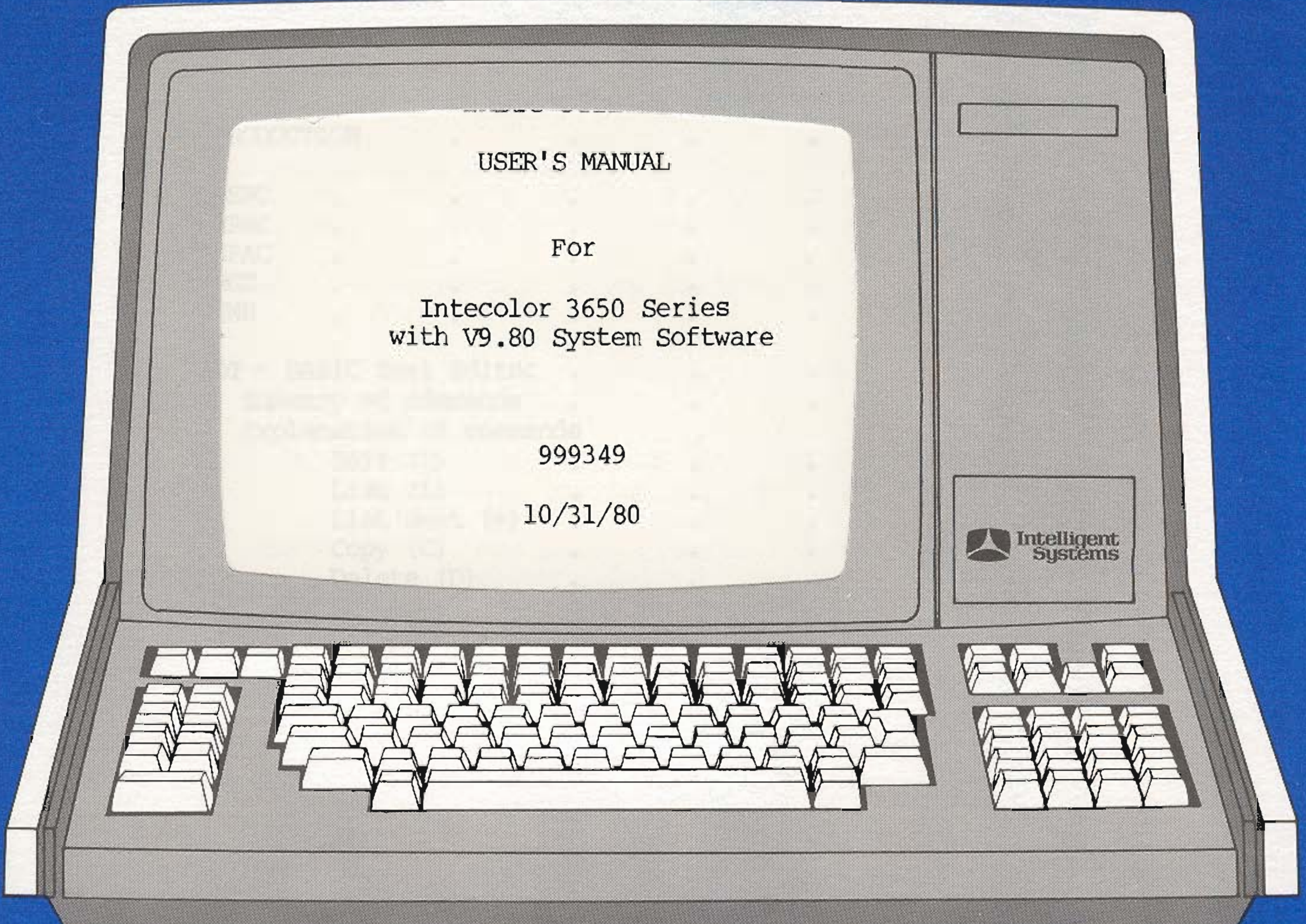

USER'S MANUAL

For

Intecolor 3650 Series
with V9.80 System Software

999349

10/31/80

Intecolor

AN INTELLIGENT SYSTEMS COMPANY

225 Technology Park/Atlanta • Norcross, GA 300921 • Telephone 404/449-5961 • TWX 810-766-1581

BASIC UTILITIES

USER'S MANUAL

For

Intecolor 3650 Series
with V9.80 System Software

999349

10/31/80

Intelligent Systems Corp.
Intecolor Drive
225 Technology Park/Atlanta
Norcross, Georgia, 30092
Telephone: 404/449-5961
TWX: 810/766-1581

TABLE OF CONTENTS

INTRODUCTION

Basic Utilities is a group of programs designed to aid the Basic programmer by performing functions that are not available in Disk Basic V9.80. With the exception of the BASIC Text Editor, all of the utilities are run independent of Basic and each modifies Basic files in specific ways.

The Basic utilities are listed below with brief descriptions of what each does. Unless otherwise noted, all references to a Basic program or file refer to Disk Basic 8001 V9.80.

1) BASSRC - converts a Basic program file to an ASCII source file.

2) COMPAC - removes all unnecessary spaces from a Basic program file.

3) REMPAC - removes all Remark statements (REM) and all unnecessary spaces from a Basic program file.

4) MERGE - merges two Basic program files into a third Basic program file.

5) RNUMB - renumbers all or part of a Basic program file.

6) BEDT - the BASIC Text Editor that is callable from Basic.

The Basic utilities are divided into two groups dependent on the way in which they are used. As stated before, all of the utilities except for the BASIC Text Editor are run independent of Basic and they create as their output a new modified file while leaving the old file(s) intact and unchanged. This group of utilities have similar characteristics in their restrictions and in the commands that they accept.

At run-time, the user can specify the name and device for the file that will be created. This will be of the form:

prompt> source file spec **TO** destination file spec

The exact default conditions will depend on the utility and these will be specified in their respective sections. At the time of the 'prompt>', the user can abort and go back to FCS by typing a 'Control C'. The message 'CONSOLE INTERRUPT !' will be displayed followed by the FCS prompt. The program can be re-executed by typing 'ESC T'. This is applicable even after a utility has been successfully executed.

While running one of these utilities, it is possible to break and/or abort the execution of a program. To break the execution, the BREAK key or a 'Control B' is struck. To abort the execution, a 'Linefeed' (down arrow) or a 'Control C' is used. After a break, execution of the program may be resumed by striking any key except the 'break' or 'abort' keys. An abort can be performed either during program execution or after a break. When the program is aborted, the message 'CONSOLE INTERRUPT !' is displayed and the user is returned to FCS. The only restriction to these commands is that they are disabled during disk access.

The other group of utilities consists of the two versions of the BASIC Text Editor (BEDT16 and BEDT32). These utilities are RAM resident while the user is programming in Basic. In order to use them, the user must first initialize Basic and then the appropriate version (16K or 32K) must be run from FCS. At that time, the user is free to work on Basic programs either in Basic or in the BASIC Text Editor as the editor is available for use from Basic through the 'ESC ^' call.

NOTES: 1) Care should be taken by the user to make sure that there is sufficient room on the disk to hold the converted program file.

2) Since each of the utilities in the first group run in user RAM, the user must remember to save the Basic program that is presently in memory **before** running the utility; otherwise, the Basic program will be lost.

3) If the user wishes to return to Basic after running any of the utilities except for the editor, Basic must be re-initialized and the editor, if desired, must be reloaded.

BASSRC

The BASSRC utility allows the user to convert a Basic program file (.BAS) into an ASCII source file (.SRC). Since Basic programs are stored in a compressed form, commands to view or print a Basic program must be executed from Basic. After the conversion of a Basic program file, the ASCII source file will contain the text as it would be printed complete with carriage returns and linefeeds. In this form, it is often easier to transfer Basic programs to other computers.

When the BASSRC utility is executed, all Basic tokens are expanded into ASCII strings, and all control codes (<32) that are outside quoted strings are thrown away except for 'HT' (horizontal tab) and 'LF' (linefeed) which are passed intact. All control codes within quoted strings are passed intact. Quoted strings in DATA statements or INPUT statements are not allowed to contain any control characters except for 'HT' and 'LF'. At the end of each line there is placed a 'CR' (carriage return) and a 'LF'.

To run BASSRC, the user must be in FCS and type:

```
FCS>BASSRC
```

The run-time header and prompt for BASSRC is:

```
BAS/SRC V1.30 FOR V9.80
BASIC FILE>
```

The user then enters the Basic program file to be converted as:

```
     BASIC FILE>file spec (TO file spec)
```

The default destination 'file spec' is the filename with the .SRC type and the same version number as the .BAS file. If another filename is desired, then it must be specified at this time and its default version number will be '01'. Other general run-time information is described in the Introduction section of this manual.

COMPAC

The COMPAC utility is used to remove unnecessary spaces from the text portion of a Basic program. Keywords are displayed with spaces between them even if none are present.

To run COMPAC, the user must be in FCS and type:

```
**FCS>**COMPAC
```

The run-time header and prompt for COMPAC is:

```
BASIC SPACE REMOVER V1.30 FOR V9.80
COMPACT>
```

The user then enters the Basic program file to be compacted as:

```
COMPACT>file spec (TO file spec)
```

The default destination 'file spec' is the filename with the .BAS type and the next higher version number than the original .BAS file. If another filename is desired, then it must be specified at this time and its default version number will be '01'. Other general run-time information is described in the Introduction section of this manual.

REMPAC

The REMPAC utility is used to remove Remark (REM) statements and unnecessary spaces from a Basic program. Keywords are displayed with spaces between them even if none are present. If the Remark statement is the only statement on a line, then the remark is removed but the REM is left so as to preserve the line number in case there is a reference to it. If the Remark statement is preceded by another statement(s) in the line, then the remark, the REM and the preceeding colon are removed.

To run REMPAC the user must be in FCS and type:

```
FCS>REMPAC
```

The run-time header and prompt for REMPAC is:

```
BASIC REMARK AND SPACE REMOVER V1.30 FOR V9.80
COMPACT>
```

The user then enters the Basic program file to be compacted as:

```
COMPACT>file spec (TO file spec)
```

The default destination 'file spec' is the filename with the .BAS type and the next higher version number than the original .BAS file. If another filename is desired, then it must be specified at this time and its default version number will be '01'. Other general run-time information is described in the Introduction section of this manual.

MERGE

The MERGE utility merges two Basic program files. The resulting output file will contain the lines of both of the input files. In the event that there are duplicate line numbers, the line placed in the output file will be taken from the second input file.

To run MERGE the user must be in FCS and type:

```
     FCS>MERGE
```

The run-time header and prompt for MERGE is:

```
     BASIC MERGE V1.30 FOR V9.80
     MERGE>
```

The user then enters the Basic program files to be merged as:

```
     MERGE>file spec 1,file spec 2 TO file spec 3
```

All of the 'file specs' must be entered explicitly as there are no default conditions allowed.

In MERGE, different disk drives may be used as in the following example:

```
     MERGE>0:FILEA,1:FILEB TO 1:FILEC
```

FILEA will be taken from drive 0: and merged with FILEB from drive 1: and the result will be placed on drive 1: in FILEC.

Other general run-time information is described in the Introduction section of this manual.

**** Care should be taken by the user to ensure that there is sufficient disk space for the output file from MERGE.

RNUMB

The RNUMB utility allows the user to renumber all or sections of a Basic program. All line number references to the section being renumbered will be changed also.

RNUMB will only change line numbers and will not allow the re-ordering of lines within the program. For this reason, the sections to be renumbered must be entered in order.

To run RNUMB the user must be in FCS and type:

```
FCS>RNUMB
```

The run-time header and prompt for RNUMB is:

```
BASIC RENUMBER V1.30 FOR V9.80
RENUMBER>
```

The user then enters the Basic program file to be renumbered as:

```
RENUMBER>file spec (TO file spec)
```

The default destination 'file spec' is the filename with the .BAS type and the next higher version number of the original .BAS file. If another filename is desired, then it must be specified at this time and its default version number will be '01'. Other general run-time information is described in the Introduction section of this manual.

After the filename is entered, the program will respond with:

```
     NEWLINE,INCREMENT,OLDSTART-OLDEND>
```

These refer to the first line number that you wish to have in the program or section, the increment between lines, the old line number from which you wish the renumbering to begin, and the old line number where you want it to end. The default (simply pressing RETURN) is to renumber by 10 starting with new line 100.

Except for the cases noted below, the renumber utility will not allow a previously runable program to be made unrunable. RNUMB will issue the error message **'PASS 1 SEQUENCE ERROR'** if the user attempts to renumber a program in such a manner as to change the sequence of the statements. After the error message, the first NEWLINE prompt will be displayed.

Restrictions:

Unresolved line numbers (GOTOs to nowhere) will be left unchanged.

Programs which LOAD other programs and then GOTO a line number rather than RUNning the other programs will have the line number(s) that reference other program(s) renumbered also.

The FILE"TRAP" statement is legal in Basic but not in RNUMB therefore FILE"T" must be used.

The FILE"E" statement will not be handled properly if the line number returned **in** the statement is placed in a variable **and** later used for comparison. The following example will **not** be handled properly.

```
100 FILE"E",FL,ER,LN
120 IF LN=50 THEN GOTO 200
```

The line number 50 used in the comparison will not be renumbered correctly.

Examples of Renumbering:

Renumber a program by 10 starting with line 20.

```
NEWLINE,INCREMENT,OLDSTART-OLDEND>20
```

RNUMB will assume that the user wishes to renumber the entire program by 10.

Renumber a program by 100 starting with line 1000.

```
NEWLINE,INCREMENT,OLDSTART-OLDEND>1000,100
```

RNUMB will assume that the user wishes to renumber the entire program.

Renumber the last portion of a program by 10 starting with line 1000.

```
NEWLINE,INCREMENT,OLDSTART-OLDEND>1000,10,1000
```

Renumber a program by 10 but leave sections starting on even 1000s.

```
NEWLINE,INCREMENT,OLDSTART-OLDEND>10,10,0-999
NEWLINE,INCREMENT,OLDSTART-OLDEND>1000,10,1000-1999
NEWLINE,INCREMENT,OLDSTART-OLDEND>2000,10,2000-2999
etc. to include all lines of the program
```

BEDT - BASIC Text Editor

The BASIC Text Editor is a line editor that will assist a user in the modification of existing Basic programs and in the creation of new ones.

The BASIC text editor will allow the user to modify any line of a program without having to retype the line; to list any portion of a program (in pages), or list only those lines containing a defined 'search string'; to copy a line from one location to another and then edit the new line; to delete a line or group of lines by specifying the first line and last line of the range to be deleted; to insert new lines one at a time; or to enter new lines or sections with the editor providing the line numbers that are user specified as to beginning line number and increment between line numbers.

In order to aid the user, all lines listed by the BASIC text editor will be shown with the text in green and control characters in red 'lower-case'. These 'lower-case' characters will be determined by the type present in the user's machine.

To run the BASIC Text Editor, the user must first initialize Basic. Now from FCS, the user should type:

```
     FCS>BEDT16 (16K of user RAM)

                 or

     FCS>BEDT32 (32K of user RAM)
```

BEDT16 or BEDT32 will respond with a header which will include the command to 'call' the BASIC Text Editor from Basic. After the header, the Basic 'READY' prompt will appear and the user is now in Basic. A previously saved Basic program may be loaded at this time or a new program may be entered. To call the BASIC editor from Basic, use the command 'ESC ^'.

BEDT16 uses memory locations AEE0 to BFFF and BEDT32 uses memory locations EEE0 to FFFF. Basic is located 'below' this area.

SUMMARY OF COMMANDS

COMMAND CODE	DESCRIPTION
E	Edit an existing line of a program.
L	List one 'page' of program text.
+	List next 'page' of program (follows L).
C	Copy a line to a new location and allow the new line to be edited.
D	Delete a line or range of lines.
I	Insert one line.
A	Automatic line numbering mode for sequential insertion.
B	Return to Basic.
ERASE PAGE	Clear screen.

Any of the above 'single key' commands may be used when the editor returns with the prompt:

```
EDITOR READY
>
```

Edit:

The 'Edit' command allows the user to edit the text and control characters, but not the line number of an existing line of a Basic program.

After typing the 'Edit' command (E), the following prompt will appear:

```
LINE NUMBER>
```

The acceptable responses to the prompt are:

1) An existing line number

2) An 'F' for the first line of the program

3) An 'L' for the last line of the program

4) A '+' (plus) or a 'RETURN' for the next line

5) A '-' (minus) for the previous line

6) A '.' (period) for the same line

All of the responses above (except for RETURN) must be followed by a 'RETURN'.

If the specified line is found, it will appear near the top of the screen on the fifth line (and on the sixth line if the line to be edited contains more than 64 characters). The 'edit cursor' will be positioned at the first character of the line to be edited and the prompt 'PLEASE EDIT:' will be displayed on the seventh line of the screen.

The 'edit cursor' will be indicated in one of the following ways:

1) A blinking CYAN character for a character of text.

2) A blinking MAGENTA 'lower-case' character for a control character.

3) A solid CYAN block for a space.

4) A solid MAGENTA block for the 'End of Line' marker.

Whenever the 'PLEASE EDIT:' prompt is on the screen (on the seventh line), the user will be able to edit the line in the 'edit buffer' which consists of the fifth and sixth lines of the screen. The following list contains the legal 'Edit' commands:

Edit Command	ACTION
—>	Move edit cursor right.
<—	Move edit cursor left.
]	Move cursor to end of line.
[	Move cursor to beginning of line.
'DEL CHAR' or 'SHIFT _'	Delete the character before cursor.
'BELL' (CTRL G)	Convert all lower-case characters from right of cursor to end of line to upper-case.
'LINEFEED' or 'CTRL J'	Abort edit - do not change original line - return to command mode.

Edit Command	ACTION
'RETURN' or 'CTRL M'	Place new edited line into Basic program and return to command mode. If the new line is null (empty), then edit is aborted and line is **not** replaced.
'NULL' (CTRL @)	Insert next character regardless of what it may be. Usually for inserting one of the above control characters.
Other character	Insert character before cursor.

Warning:

The text may extend up to 127 characters but the editor **will not** check for an overflow of the edit buffer. If an overflow occurs, the edit **must** be aborted by 'ERASE LINE', 'LINEFEED', or 'CPU RESET'; otherwise the Basic program **may** be lost.

NOTE: 'DELETE CHAR' will not delete characters in the event of an overflow.

List:

The 'List' command allows the user to list any part of a program beginning with a specified line or to list just those lines beginning with a specified line number that contain the specified 'search string'. The text is listed on the screen in 'pages' of 21 lines each.

After typing the 'List' command (L), the following prompts appear:

```
LINE NUMBER>
SEARCH STRING>
```

The response to 'LINE NUMBER>' is the line number to be listed first. The response to 'SEARCH STRING>' is a string or series of characters which the editor will search for. If a line whose number is greater than or equal to the line number that was entered is found that contains this string, then it will be listed. The search will continue to the end of the Basic program. If the 'search string' is null (by just pressing 'RETURN'), then the specified line and the next 20 lines following it will be listed.

The BASIC text editor will only list one 'page' of lines at a time. A 'page' consists of 21 lines.

List Next:

The 'List Next' command (+) allows the user to 'List' the **next** page of a program, beginning with the next line. The 'List Next' command also 'keys' on the 'search string' to determine which lines will be displayed. This command must always be preceeded by the 'List' command.

Copy:

The 'Copy' command enables the user to duplicate a specific line anywhere in a Basic program.

After typing the 'Copy' command (C), the following prompts will appear:

```
            OLD LINE NUMBER>
            NEW LINE NUMBER>
```

The parameters explicitly required are an **old line number** (which exists), and the **new line number** where the new line should be placed (which should not exist). If the 'Copy' command is accepted, a copy of the old line will be placed at the new line number and the user will be placed in the 'Edit' mode so that the new line can be edited, if necessary. The 'Copy' is completed when the 'Edit' is completed by either some editing followed by a 'RETURN', or by accepting the line as is with a 'RETURN' or a 'LINEFEED'. When the 'Copy' is completed, the new line will be listed on the first available line in the 'listing' area of the screen.

Delete:

The 'Delete' command enables the user to delete a line or range of lines from a Basic program.

After typing the 'Delete' command (D), the following prompts will appear:

```
FIRST LINE>
LAST LINE>
```

Both of the parameters must be explicitly entered, both line numbers must exist, and the first line number must be less than or equal to the last line number for the 'Delete' command to be accepted and executed. If the 'Delete' is accepted, all of the lines within the specified range, including the line numbers specified will be deleted. In order to delete only one line, the line number must be typed for both prompts.

Insert:

The 'Insert' command enables the user to insert a new line into the Basic program.

After typing the 'Insert' command (I), the editor responds with:

```
                LINE NUMBER>
```

The user must explicitly enter the number of a line that does not already exist in the program. If the 'Insert' is accepted, then the user is placed in the 'Edit' mode to enter the new line. At this point, the new line is null and the user must enter the desired text followed by 'RETURN' to complete the 'Insert'. After a successful 'Insert', the new line will be displayed on the first available line in the 'listing area' of the screen. If a null line is entered or 'LINEFEED' is pressed, then the 'Insert' is aborted.

Auto-Numbering:

The 'Auto-Numbering' command allows the user to enter new lines of a Basic program with the editor supplying the line numbers, with the user having initially specified the beginning line number and the increment.

After typing the 'Auto-Numbering' command (A), the following prompts are displayed:

```
LINE NUMBER>
INCREMENT>
```

The **line number** must be entered explicitly and must not already exist. If defaulted, the **line number** will be '0'. The initial default value for the **increment** is 10 and if any other value is desired, it must be entered explicitly. In subsequent 'Auto-Numbering', the default value of the **increment** will be the last value used.

'Auto-Numbering' will stop automatically if there is a collision with an existing line number; however, if there is a line number that is in the path of the new lines but does not collide with a new line, then there is no warning that a new line has been placed both before and after the already existing line.

The 'Auto-Numbering' mode is exited by striking the 'LINEFEED' or the 'RETURN' key after a null line.

Back to Basic:

By using the 'Back to Basic' command (B), the user will be returned to Basic and the 'READY' prompt for Basic will be displayed. The return path to the BASIC text editor is 'ESC ^'.

Clear Screen:

The user may clear the screen by pressing the 'ERASE PAGE' key. This will erase all user entries or listed text from the screen while leaving the editor's screen format intact.