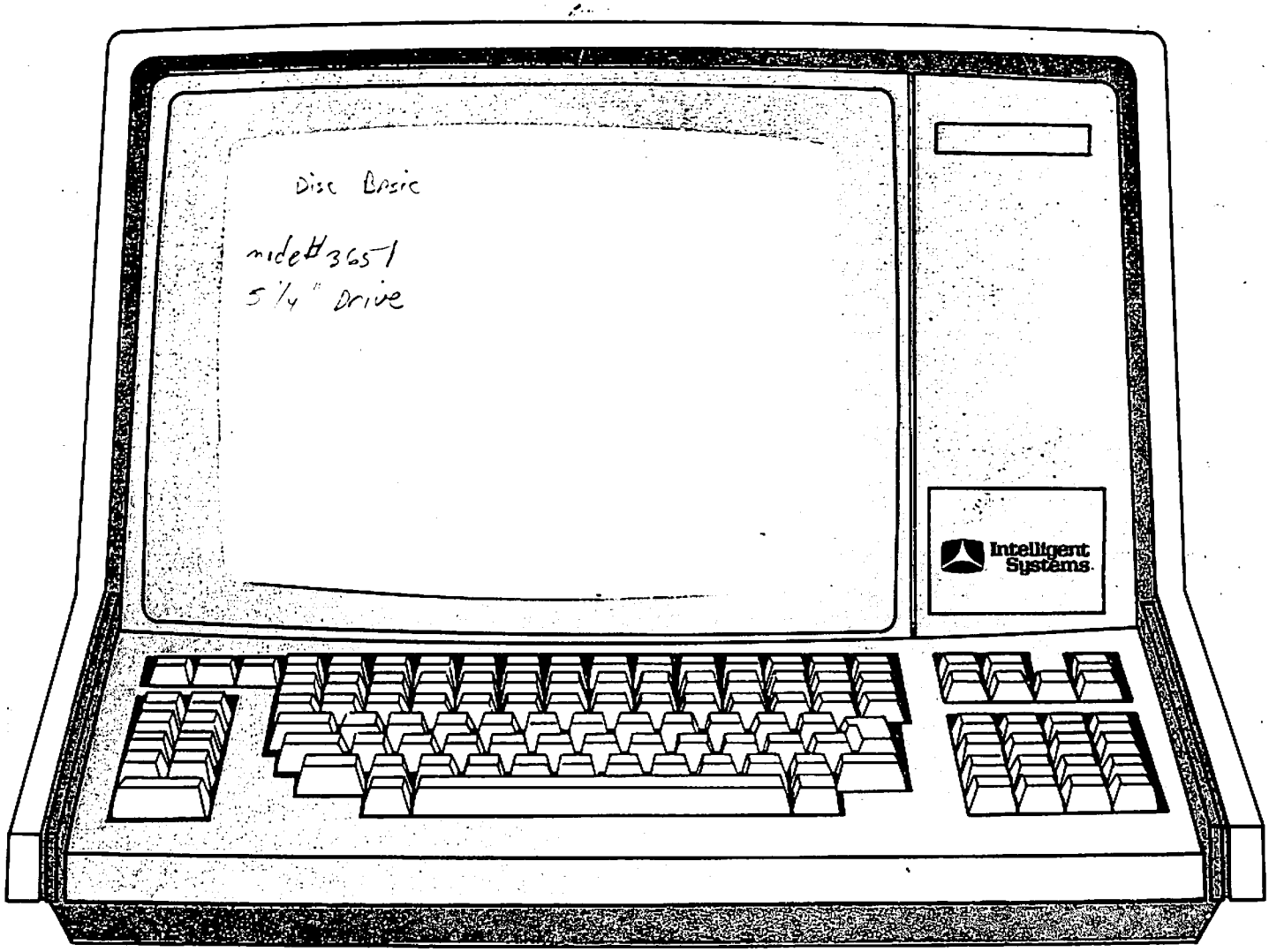


Intecolor®

AN INTELLIGENT SYSTEMS COMPANY

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

USER'S MANUAL

for the

INTECOLOR 3650 SERIES

Desktop Computers

(With FCS Utilities)

999276 1/15/84

WARNING

This equipment generates, uses, and can radiate radio frequency energy and may cause interference to radio communications if not installed and used in accordance with the instructions in this manual. The equipment's radio frequency emissions have been measured and found to be within the limits established in FCC Rules, Part 15, Subpart J, for Class A computing devices. These rules are designed to limit interference by such devices to levels considered reasonable for commercial environments. If the user operates this equipment in a residential environment he will be likely to cause interference for which he can be required to take corrective action at his own expense.

Intecolor Corporation
225 Technology Park/Atlanta
Norcross, Georgia 30092
Telephone: 404-449-5961
TWX: 810-766-1581

NOTICE

To report errors in the manual or software bugs, please complete and return the Software Problem Report form at the back of this manual.

Special Note to the User

Intecolor 3600 Series units manufactured after January 6, 1983 are equipped with keyboards having two-tone grey keycaps. Functionally, these keyboards differ from older keyboards in one important respect: The CPU Reset key must be operated while the Shift key is held down to effect reset. This feature has been added to help avoid accidental reset. References to CPU Reset in the text of this manual do not take this new feature into account.

Before attempting to operate the 3600, see the Installation Notes at Appendix H. These notes are supplementary to Chapter One of this manual and contain important information about other new features of the unit.

This printing of the 3650 Series Manual contains a new alignment procedure at Appendix I and new schematic drawings for the keyboard, analog and video circuitry. The new procedure and schematics are appropriate only to units manufactured after January 6, 1983, which incorporate extensive hardware design changes.

The new hardware includes an in-line gun CRT with automatic screen degaussing equipment in place of the delta-gun CRT in older units, a new analog board (power supply and CRT deflection circuitry), and a new video amplifier assembly. The logic circuitry in the new units remains unchanged, and no changes have been made in software.

Intecolor Corporation will continue to provide technical information on older 3650 Series units on request.

TABLE OF CONTENTS

6. MORE STATEMENTS, COMMANDS, AND FEATURES

7. FUNCTIONS AND SUBROUTINES

8. ARRAYS

9. FURTHER SOPHISTICATION

10. DISK FEATURES

11. COLOR, GRAPHICS, AND OTHER TERMINAL FEATURES

12. THE FILE CONTROL SYSTEM

APPENDICES

1. INTRODUCTION - INITIAL TEST

Your 3650/9650 Series Intecolor unit is a complete desktop computer with eight-color CRT display, 16K or 32K of user memory, the simple but powerful Extended Disk BASIC programming language, one or more disk drives for program and data storage on Sof-Disk, a versatile File Control System and color graphics display capability. When connected through a modem it can be used for data communications over common voice telephone lines. The serial I/O port also can be used with certain types of printers. Other types of printers and peripherals can be interfaced to the system Extension Bus.

The 3651 Intecolor is a completely self-contained unit, with a 5.25" Mini-Disk Drive included in the console. The 3652 adds another 5.25" Mini-Disk externally. The 3653 and 3654 have dual external 8" Floppy Disk Drives, single head and double head respectively. The full size keyboard — 72-key standard, 101-key and 117-key versions optional — is built in.

In the 9650 Series the keyboard is detachable but fits against the console to simulate a one-piece unit. Disk drive combinations are similar to those in the 3650 Series.

Each unit features a color CRT (13" diagonal in the 3650 Series, 19" in the 9650 Series) which can display eight foreground and eight background colors -- red, green, blue, yellow, magenta, cyan, black and white. Convergence is adjusted from a recessed front panel with nine-sector control. Further specifications are included in Appendix A at the rear of this manual.

1.01 Initial Operation

After unpacking your new 3650/9650 Series Intecolor Desktop Computer, please check it out in this order initially:

First, verify that the console-keyboard combination is working correctly before connecting any other device.

Then connect any external disk drives and/or other peripherals.

1.01.01 Connecting the Keyboard to the Console

3651 Intecolor

Everything is in the console in this model. Plug the power cord into the socket on the console rear panel. Verify that the lower edge of the white power switch nearby is depressed. Connect the power cord to a source of AC power (standard is 60 Hz, 115 VAC, 3-wire). Now proceed with the INITIAL TEST section, page 2.

3652, 3653 or 3654 Intecolor

Proceed as above for the 3651 Intecolor. Do not connect the external disk drive yet.

9650 Series Intecolor

In the 9650 Series units the keyboard is separate from the console, though shaped to fit snugly against the console front if desired. Connect the keyboard cable edge connector to the Logic Module at the rear of the console. The dark color edge of the flat flexible cable must be to the left as viewed from the console rear. The cable may be routed under the console, staying clear of the feet. Plug the power cord into the socket on the rear panel. Verify that the lower edge of the white power switch nearby is depressed. Connect the power cord to a source of AC power (standard units use 60 Hz, 115 VAC, 3-wire). You're ready for initial test!

1.01.02 Initial Test

Before applying power, verify that there is no diskette in the disk drive(s). (Of course you don't expect one to be there in a new unit. However, it's a good habit to begin forming, verifying that disk drives are empty before turning power on or off. Power up or power down with a diskette in the drive sometimes damages information on the diskette.)

Depress the upper edge of the power switch to turn power on. Within less than a minute, warm-up should be complete and the screen display in red, cyan, yellow and green double-height characters reads:

```
DISK BASIC V9.80 COPYRIGHT (C)
MAXIMUM RAM AVAILABLE?
15665 BYTES FREE
READY
```

or something quite similar. The above display is for a 16K machine. A 32K unit will show 32049 BYTES FREE. The V9.80 indicates the vintage of the software, which may be revised from time to time.

The white cursor will be seen blinking at the left edge of the screen a couple of lines below.

IF THE SCREEN DISPLAY DOES NOT APPEAR, STRIKE THE CPU RESET KEY.

If the correct display appeared or was in process, operation of the CPU RESET key (upper right corner of the keyboard) will result in a green and cyan display in normal height characters:

```
CRT MODE V9.80
```

Operate CPU RESET and check the display size by erasing the screen with a background color other than black:

1. Operate the **CAPS LOCK** key to the locked **down** position. (This affects only the alpha characters and results in the upper case used in many operations.)

2. Operate the **BG ON/FLG ON** key. (To enable background color selection.)

3. With the **CONTROL** key held operated, operate the red **Q** key. (To select red as the background color. In the 101-key and 117-key keyboards, use of the red key in the color pad at left without CONTROL does the job.)

4. Operate the **ERASE PAGE** key. (To erase the entire page to the selected background color. The blinking white cursor is still visible at the upper left corner of the screen.)

The raster should be approximately 7.25" x 9.50" (10" x 13" in the 9650 Series), of fairly uniform red color.

Check convergence with a full-screen pattern of white letters against a black background:

1. With the **CONTROL** key held operated, operate the black **P** key. (The earlier BG ON key operation is still in effect. CONTROL P — or the black color pad key in 101/117-key keyboards — selects the background color black.)

2. Operate the **FG ON/FLG OFF** key. (To enable selection of foreground color.)

3. With the **CONTROL** key held operated, operate the white **W** key. (Or just the white key in the color pad; to select white as foreground.)

4. Select X2 characters by operating the **A7 ON** key. (This results later in 16 lines of double-height characters.)

5. In sequence operate the **(ESC)**, the **(TEST)** **Y** and the **L** keys. (The sequence of (ESC) Y and another key causes the screen to fill with the last character struck.)

There should be a full-screen display of double-height L characters, in white against a black background. The letters should be a fairly uniform white. (Pronounced colored edges indicate need for convergence adjustment. This simple adjustment, made from the front of the console, sometimes is required after shipment. See the Adjustments, Appendix H.

Operate the ERASE PAGE key to clear the screen. Then with CONTROL and SHIFT keys (or just the COMMAND key in the 101-key or 117-key keyboards) held operated, operate the CPU RESET key. The initial display should return. The unit is in the BASIC mode, ready to accept programming in BASIC. A short program may be used for test. With the CAPS KEY still in the locked down position, type an instruction for immediate execution such as:

```
PRINT "IT WORKS!"
```

Or put some other short message in quotes. (CAPS LOCK key down takes care of the upper case alpha characters, but SHIFT is still required to get the quotation marks and exclamation points.)

Now operate the RETURN/ENTER key (designated in most instructions by the symbol <CR>). The message between the quotation marks is displayed and then another READY.

Everything OK so far? In the 3651, check of the disk drive is the next step. In other units of the 3650/9650 Series, turn power off before connecting external disk drives or other peripherals.

1.01.03 Connecting the External Disk Drive

AC power should be off in both the Intecolor console and the external disk drive when connecting the drive unit. (The lower edge of white power switch depressed is the off position in each case.)

Connect the edge connector of the flat fexible connector to the EXT DISK board area at left rear of the console, dark colored stripe to the left as viewed from the console rear. (Standard configurations have no more than one external disk drive unit. However, additional drives may be added through multiples off this same connection.)

Connect the disk drive unit's power cable to a source of AC power (standard is 60 Hz, 115 VAC, 3-wire).

Depress the upper edge of the white power switches on the Intecolor console and then the external disk drive to apply power. The computer is ready for further check after a one-minute warm-up.

1.01.04 Checking Disk Drive Operation

Perhaps the simplest way to check disk drive operation is by use of the SAMPLER diskette, which has a "menu" of several recorded programs. Insert the programmed diskette into the built-in drive (or the left-hand drive of a dual drive unit). With the drive door open, insert the diskette all the way into the disk drive unit with the label side up (or to the right in the internal drive). Close the drive door. (The door of the 5 1/4" drive tilts outward or upward to open, inward or downward to close. The door of the 8" drive normally is open; if closed, depress the small bar inward to release the gate to the up, or open, position. To close the door of the 8" drive, depress the wide upper bar downward until it locks closed.)

With the ISC programmed diskette in the drive and the drive door closed, strike the AUTO key. Following drive operation, the screen will display a "Menu" showing the diskette content and instructions for use.

A few notes regarding handling of the diskettes may be appropriate at this point:

1. NEVER touch the diskette magnetic surface (exposed through the oval window slot) with the fingers or thumb.

2. Always place the diskette back in its protective envelope after use. Keep it away from dust, dirt and liquids — these can cause loss of data.

3. Avoid bending the diskette -- handle carefully.

4. Do not leave diskettes lying around the video unit, near electric motors or adjacent to any other types of apparatus which generate magnetic fields that can alter data.

5. Use a felt tip pen when making notes on the diskette label -- use of a ballpoint pen or a pencil can damage the diskette.

6. Avoid storage of diskettes in abnormally cold or hot areas. (Stay within the range of 10° to 52° C. -- 50° to 125° F.)

7. Never insert a diskette into the disk drive before powering up the drive; never power down the drive while a diskette is still inserted.

8. Never remove a diskette from the disk drive while the red "busy" light is on (or from the internal 5.25" drive while the motor is running).

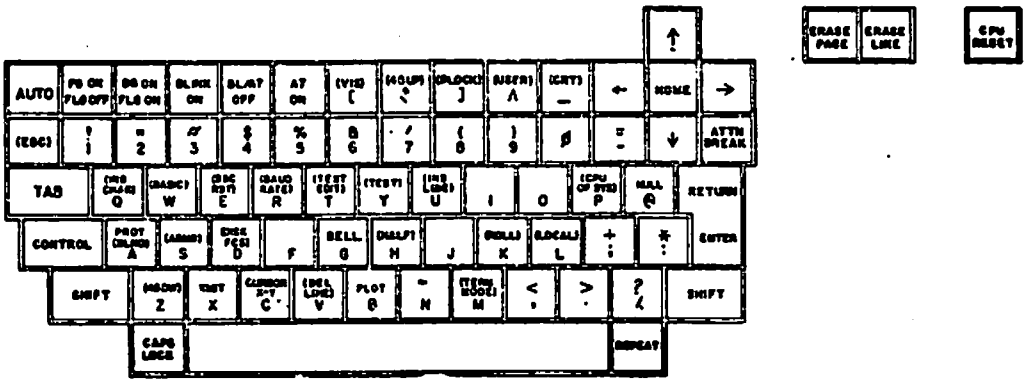
9. When inserting the diskette in the disk drive, note that it "clicks" into place before closing the compartment door.

1.02 Learning to Use Your Intecolor Computer

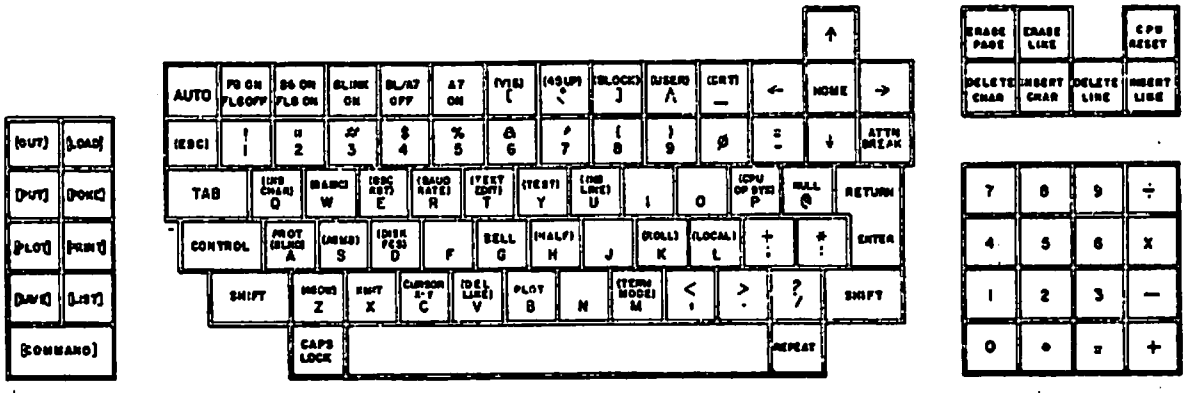
When you have assembled your 3650/9650 Series Intecolor Desktop Computer as outlined above, everything likely is operating as it should. The next section of this manual describes the functions of the keyboard and initial usage of the disk drive. Succeeding sections of this manual describe the use of the computer in Extended Disk BASIC, the generation of color graphics, the File Control System and use as a terminal.

Appendix A near the rear of this manual outlines the specifications for the 3650/9650 Series Intecolor units. Appendix E contains details regarding the pin-out of the rear panel connections for a modem and for peripherals using the Extension Bus.

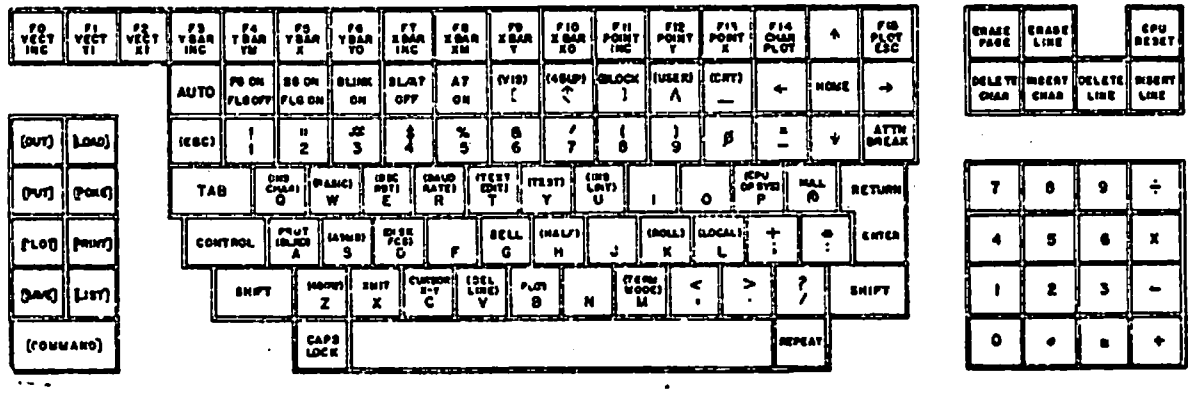
If you have any problem, please contact the sales representative from whom the system was purchased.



72 - KEY



101 - KEY



117 - KEY

2. USING THE KEYBOARD AND DISK DRIVE

2.01 The Intecolor Keyboards

The Intecolor keyboards are similar to a typewriter keyboard in many respects. However, there are some differences and many keys serve more than one function. The key layouts for the three versions of the keyboard are shown on the page to the left. During the familiarization phase it probably is best to use the CRT Mode — obtained by striking CPU RESET.

CPU RESET — The **CPU RESET** key provides a reset signal to the 8080 CPU. Its primary function is to allow the operator to regain control if the unit goes into an endless loop. It automatically forces the unit to return to the CRT Mode initialized state. The CRT refresh RAM is cleared, but the user random access memory is not cleared -- a program already entered into memory is not lost. The unit can be returned to BASIC, with the program intact, by operating the (ESC) and E keys in sequence. (An ESC W sequence, on the other hand, clears the user RAM while initializing BASIC.)

CAPS LOCK — The **CAPS LOCK** key affects only the alpha keys. When it is in the locked down position, operation of the letter keys will result in upper case characters. When it is released to the up position, operation of the letter keys will result in special characters (or lower case letters, if that option is present) as determined by the reverse field flag setting -- more about that flag later. Use of the SHIFT key when the CAPS LOCK key is down negates the effect of the CAPS LOCK key — i.e., operation with both the SHIFT and the CAPS LOCK keys down is the same as though neither were down. For operation in BASIC, the CAPS LOCK key in the locked down position is most convenient.

SHIFT — The **SHIFT** key must be held down while the selected key is operated if the shifted function is desired. For generation of certain codes, both the SHIFT and CONTROL keys must be held down while another key is operated. Note that the SHIFT key does not lock in the down position and it does not release the CAPS LOCK key.

CONTROL — The **CONTROL** key must be held down while operating the selected character key to implement a desired control function. The control functions, listed and described later, are indicated by key colors or by engraving (without parentheses) on several of the key tops.

(ESC) — The **(ESC)** key initiates generation of standard escape codes. The (ESC) key is depressed and released and then the character key for the desired escape code is depressed and released. Escape functions are indicated by engraving within parentheses on the key tops. Escape codes are listed and described later in the manual.

RETURN ENTER — Operation of the **RETURN ENTER** key results in a carriage return -- the cursor returns to the beginning of the line. In the BASIC mode, it enters a numbered program line into memory and sends the cursor to the start of the next line.

ATTN BREAK — Operation of the **ATTN BREAK** key can interrupt certain operations, such as a program listing, and can change Full Duplex terminal mode to Half Duplex. More on this later.

TAB — The **TAB** key moves the cursor to the next tab position. Considering the character position at the extreme left of the screen as position zero, the tab positions are 0, 8, 16, 24, 32, 40, 48 and 56. Beyond character position 56, operation of the TAB key moves the cursor to the start of the next line. When in the BASIC mode, operation of the TAB key with SHIFT results in RUN (the same as typing RUN).

A7 ON — Operation of the **A7 ON** key initiates generation of double height characters. It serves other functions which will be described later.

BLINK ON — Operation of the **BLINK ON** key causes foreground blink in display generated after operation of the key.

BL/A7 OFF — Operation of the **BL/A7 OFF** key negates the effect of both the A7 ON and the BLINK ON operations. So if one wishes to cancel blink but maintain x2 character height, it is necessary to operate the A7 ON key again after operating the BL/A7 OFF key.

FG ON FLG OFF — Operation of the **FG ON FLG OFF** key followed by operation of a color key with CONTROL down -- or simply by operation of a color key in the color pad -- results in selection of foreground color for display created subsequently. The setting of the flag also affects character generation, to be discussed later.

BG ON FLG ON — Similar to FG ON FLG OFF but for background color.

REPEAT — If a character or function key is held down and the **REPEAT** key is also held down, the character or function will repeat while the REPEAT key is held down.

AUTO — When a diskette having a "Menu" is in the default disk drive and BASIC has been initialized, operation of the **AUTO** key results in disk operation and display of the diskette menu.

ERASE PAGE — Operation of the **ERASE PAGE** key clears the screen to the background color and returns the cursor to the home position at screen upper left. (I.e., if the present background color is blue, the entire display goes blue. The cursor is still seen at upper left.)

ERASE LINE — Operation of the **ERASE LINE** key causes return of the cursor to the beginning of the line it is on. The entire line goes to the present background color.

CURSOR KEYS — The **Cursor Control** keys move the cursor in the direction indicated by the arrow on the key top. The HOME key moves the cursor to the "home" position at the upper left corner of the screen. In the CRT Mode the cursor direction controls move the cursor as indicated by the arrows. However, when operating in BASIC, use of the cursor controls is somewhat different. Cursor up, cursor down and HOME remain the same. Cursor right will not move the cursor to the right farther than the character position just to the right of character positions already filled. Cursor left will move the cursor only back over character positions already filled. This feature is useful in correcting typing errors detected before the RETURN key is operated. The cursor left key is used to back up to the position where the error was made. At this position the error is corrected by striking the correct key. Then the cursor right key is operated to move the cursor back to the end of the typed area, with the original typing restored as the cursor moves right. (Note: Repeated operation of the cursor right key in an effort to make the cursor move when it is not supposed to can sometimes result in erratic cursor movement. When in BASIC, and the cursor is at the beginning of a line, operation of the cursor left key will cause the cursor to move to the start of the next line.)

NUMBER PAD — In the **Number Pad** included in the optional 101-key and 117-key keyboards, most of the keys simply duplicate, in a different physical layout, the functions of other keys. However, the Multiplication, Add and Equal keys combine the effects of a key strike and SHIFT.

COLOR PAD — In the **Color Pad** at left on the optional 101-key and 117-key keyboards, operation of one of the eight color keys combines the effects of normal color key operation with the CONTROL key. The Color Pad also is useful in character plot and for programming in BASIC.

COMMAND — In BASIC, with the **COMMAND** or the SHIFT key held down, operation of one of the keys in the color pad results in generation of the BASIC command word shown on the key top in brackets. Operation of certain other keys with the COMMAND key held operated causes other BASIC command words to be generated (see Appendix B.7).

FUNCTION KEYS — The 117-key keyboard has 16 **Function Keys** F0 through F15 labeled with graphic plot submodes. Use of these keys is described in the section of Graphic Plot Submodes.

The four keys immediately above the number pad are not used as such in the 3650/9650 Series. However, they may take on user-defined functions.

Other than the space bar, the remaining keys are character keys. With the CAPS LOCK key down, the indicated character, number or symbol is generated when the key is struck. If the SHIFT key is held down while the key is struck, a shifted character or symbol is generated. In the case of the number keys, the symbol is shown on the key top above the number. (There

is no shifted symbol for the zero key.) Operation of the other alpha and symbol keys with shift results in generation of special characters or lower case characters, depending upon the character option present. In the standard 3650/9650 Series unit with the Intelligent Systems "C" character set, the special characters shown on the character sheet (page 11) are generated.

On the left side of the character sheet the 128 characters are shown arranged in eight columns, numbered 0 through 7. With the CAPS LOCK key down and without shift, operation of the 26 alpha keys and six other symbol keys result in generation of the characters shown in columns 4 and 5 (and as indicated by the key top engraving). With SHIFT, the characters shown in columns 0 and 1 are generated if the flag is on (i.e., if the most recent flag key operation was BG ON FLG ON). The characters in columns 6 and 7 are generated if the flag is off (if the most recent flag key operation was FG ON FLG OFF).

The following exercise illustrates a possible use of the special characters to generate "quad-size" letters. The alternate quad-sized A shown at the bottom of the character sheet can be generated as follows:

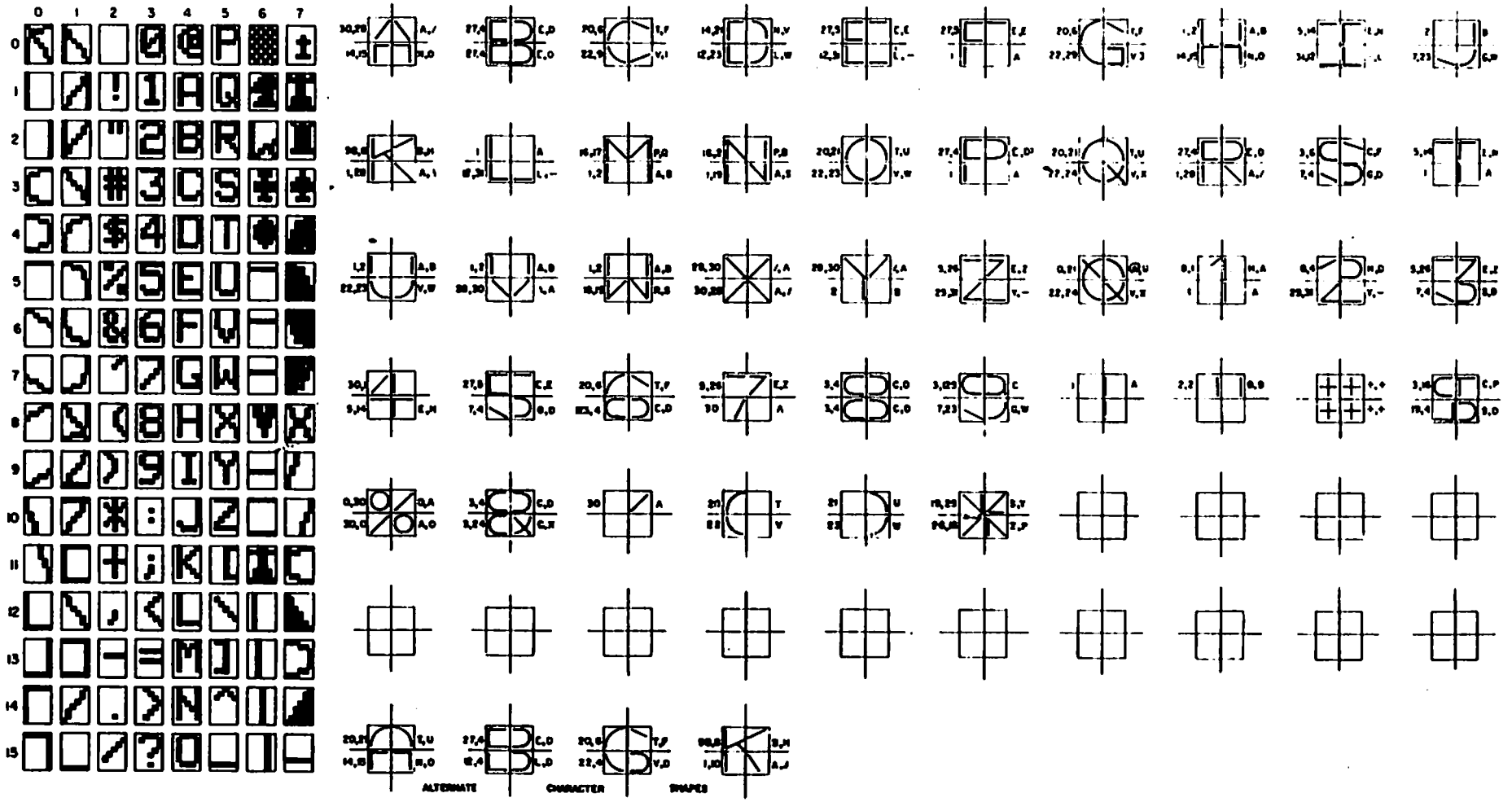
CPU RESET	To enter the CRT Mode
A7 ON	To set X2 character height
BG ON FLG ON	To set the reverse field flag on
T with SHIFT	To generate upper left quadrant
U with SHIFT	To generate upper right quadrant
Cursor down	To move down to the next line
Cursor left	Operate two times to be in position
N with SHIFT	To generate lower left quadrant
O with SHIFT	To generate lower right quadrant

In BASIC, program plot statements would be used as outlined in the Programming Manual; once programmed, the quad-height characters would flow rapidly. E.g, enter BASIC via (ESC) W and type:

```
PLOT 12,14,30,116,117,10,26,26,110,111
```

The screen is erased (12), X2 character height is set (14), the flag is set on (30) and the special characters generated.

If the 3650/9650 Series computer has the lower case option, the special characters shown in columns 6 and 7 are replaced with lower case alpha characters and certain symbols. When the unit is being operated as a word processor or as a terminal, the CAPS LOCK key may be in the up position most of the time. The operation of alpha keys result in lower case characters. Upper case characters are obtained by use of shift (as in a typewriter). However, when using the unit as a computer in BASIC, operation is more convenient with the CAPS LOCK key in the down position.



A-3 CHARACTER SET AND X4 CHARACTER COMPOSITION

2.Ø2 Using the Disk Drives

First, a reminder:

> Always remove a diskette from the disk drive after use. Before system power turn-on or turn-off, verify that the disk drives are empty. Damage to the recorded program occasionally occurs when a diskette is in the drive at power up or power down.

The internally mounted 5.25" disk drive is ready for use after the console power switch is turned on. An external disk drive must have its separate power switch operated on. The door of a 5.25" drive opens upward (or outward to the right in the case of the internally mounted drive). The door of an 8" disk drive is released to the open position by depressing the short release bar inward.

The diskette normally is inserted into the drive with its label up (or to the right in the case of the internal drive) — see illustration on page 4. Insert the diskette all the way into the drive (until it is heard or felt to "click") and close the drive door. The door of the 5.25" drive closes downward (or inward to left for the internal drive). The door of the 8" drive is closed by pressing down firmly on the long door closure bar.

If the diskette is a Sof-Disk with a program and a "menu" recorded by ISC, operate the AUTO Key to cause program load and display of the menu. The menu display includes instructions for proceeding. (Note: If BASIC is not initialized, use the (ESC) E sequence prior to operating AUTO.)

When a new formatted diskette (one with no program on it) is used for the first time to save a BASIC program that has been entered into computer memory, the diskette first must be initialized.

> **CAUTION:** Initialization results in a "clean" diskette. Any program already on the diskette is lost.

To initialize the diskette and record a BASIC program:

1. Insert the new formatted diskette ino the disk drive and close the door.

2. On the keyboard, verify that the CAPS LOCK key is down. Operate the (ESC) key and then the D key. The screen will respond with **FCS>** to indicate entry into the File Control System.

3. If the 5.25" internal disk drive is the one in use, type

   > INI MDØ:SAMPLENAME (In place of SAMPLENAME you can use any name for the disk up to 1Ø characters in length.)

   (For other disk drives, note the type and number of the drive being used. The 3652 has the 5.25" MD; the 3653 has the 8" FD; and the 3654 has the 8" DF. The external drive of the 3652 is 1; in the dual disk drive unit of the 3653 and 3654, the left-hand drive is Ø; the right-hand is 1.)

4. Operate the RETURN key. The diskette will be initialized, the screen will display the diskette directory, and then the FCS> prompt will appear.

5. Operate the (ESC) and E keys in sequence to return to BASIC without clearing the user memory.

6. To record the BASIC program in memory, continue with

   SAVE "PROGRM" (or any name up to six characters)

   and strike the RETURN key. (Be sure to include the quotation marks around the program name.)

After saving a program, it is good practice to reenter FCS (by an (ESC) D sequence) and obtain a directory listing by typing DIR and striking RETURN. (Use upper case letters for the DIR.) If the program was saved correctly, the directory will list it by name. If the effort to save the program failed, try once more. If there is still a problem or if an error message (in red) appears, see Chapter 10 and Chapter 12.

In order to load a previously saved program from a diskette, first verify that the computer is initialized in BASIC. Then type

LOAD "PROGRM" (or other name given the program)

and strike the RETURN key. The recorded program is loaded into user memory and can be used by typing RUN and striking the RETURN key.

Later sections of this manual deal with BASIC and the File Control System. They include the above information and many other features of Extended Disk BASIC and FCS. The brief description given here enables one to get started. Any programs written during study of BASIC can be saved if desired.

3. INTRODUCTION TO BASIC

3.01 Extended Disk BASIC

Extended Disk BASIC is a single user, conversational programming language which uses simple statements and familiar mathematical notations to perform operations. BASIC is one of the simplest computer languages to learn, and once learned provides the facility found in more advanced techniques to perform intricate manipulations and express problems efficiently.

Like any other language, BASIC has a prescribed grammar to which the user must adhere in order to produce statements and commands intelligible to the computer. The following pages provide a quick but complete introduction to the BASIC language. Careful reading and liberal experimentation with examples will enable a user to start programming in a short time. Adopting a leisurely pace with the text will ensure that the new user will find programming much easier than suspected.

3.02 Initializing and Running BASIC

When the 3650/9650 Series Intecolor is turned on, the screen display will be:

```
DISK BASIC V9.80 COPYRIGHT (C)
MAXIMUM RAM AVAILABLE?
15655 BYTES FREE
READY
```

or something quite similar. In a 32K unit the number of free bytes will be 32049. The READY message indicates that the machine is now ready to accept any BASIC programming statements that the user wishes to enter.

If, when the machine is powered on, the proper message does not appear, the user should hold the SHIFT and CONTROL keys down while striking the CPU RESET key. This should produce the correct screen display; however, there may be a delay of 5 or more seconds before it appears. On the deluxe or extended keyboards the COMMAND key can be held down in place of the combined CONTROL SHIFT sequence.

It may often be necessary to reset BASIC after the machine has been turned on and a program or two has been run. The first step to reinitializing BASIC is striking the CPU RESET key. The screen will output:

```
CRT MODE V9.80
```

Then, the (ESC) and W(BASIC) keys are hit in sequence. The machine will print the message:

```
DISK BASIC V9.80 COPYRIGHT (C)
MAXIMUM RAM AVAILABLE?
```

If the user desires no specific amount of memory, then simply striking the RETURN key will bring the ____BYTES FREE and READY message to the screen. The beginning user may just as well proceed in this fashion. The machine is now ready, as when it is first turned on, to either accept a user's program or load a program from a diskette.

(If, however, a certain amount of memory needs to be specified — as is necessary in some applications — the user may type in a number, subtracting any amount of space to be reserved as not for use by BASIC. The user then strikes the RETURN key and the machine will return the number of free bytes and the READY message. The minimum amount of RAM that may be specified is 763 bytes. Any lesser amount will result in an error message. Note also that BASIC uses 719 bytes for purposes such as references and for storage of pointers. These bytes must be allowed for when specifying the amount of RAM.)

If it is necessary to leave BASIC by using the CPU RESET key or by entering the File Control System, BASIC can be re-entered without losing the program in its workspace by operating the (ESC) and E (BSC RST) keys in sequence. (Remember that the ESC W sequence for initializing BASIC wipes the RAM workspace clean!)

If the machine will not return the proper messages and/or numbers, the local representative should be contacted for assistance.

3.Ø3 Using a Recorded Program

If the user wishes to use a prepared diskette — the SAMPLER diskette, for example -- it can be put into the disk drive (see section 2.Ø2, page 12) and the door closed. After BASIC is initialized, depressing the AUTO key (the brown key on the upper left of the keyboard) will result in a program named "MENU" being loaded and run. This feature normally is used to present a list of the programs available on the diskette. The menu normally includes instructions for selection of a program for use.

3.Ø4 Using the Manual

Extended Disk BASIC has thirty (3Ø) key word program, editing and command statements, eighteen (18) mathematical functions, nine (9) string functions and thirty (3Ø) two-letter error messages. These features are described in detail in the next chapters, thus providing a ready reference to BASIC's capabilities. If the user is unfamiliar with the BASIC language, then the remaining portion of this manual should be studied in sequence while having a 3650/9650 Series Intecolor available to run the examples given.

4. ESSENTIALS FOR SIMPLE PROGRAMMING

4.01 Variables

Extended Disk BASIC uses variables as a basis for conveying values in programming statements. The variable is an algebraic symbol representing a number which the user assigns to it. A variable is formed in one of three ways. It can be a letter alone, a letter followed by a number, or two letters. For example:

Acceptable Variables	Unacceptable Variables
A	3F - begins with a digit
C2	25 - numeric constant
XY	

A variable longer than 2 characters will be accepted by BASIC, but BASIC will only read the first two characters. Thus, these must be distinct from any other variables used in the program. For example, CAT is not a new variable in a program already using the variable CANCEL. Words used as specific commands or statements in BASIC are reserved, and cannot be used as variable names (e.g. LIST, RUN, READ, etc.). If such a word is used, BASIC will not accept it as a variable, and will usually return an error message. Certain other special purpose variables are acceptable in BASIC, and will be described in later sections of this manual.

When the user assigns a value to a variable, it will retain that value until it is changed by a later statement or calculation in the program. All numeric variables, until given a value by the user, are assumed by the computer to have the value 0. String variables are initially assumed to be equal to the null string (a string of length 0). This assures that later changes or additions will not misinterpret values.

4.02 Numbers

BASIC treats all numbers (real and integer) as decimal numbers, that is, it accepts any decimal number and assumes a decimal point after an integer. The advantage of treating all numbers as decimal numbers is that any number or symbol can be used in any mathematical expression without regard to its type. Numbers used must be in the approximate range $10^{-38} < N < 10^{38}$.

In addition to integers and real numbers, a third format for numbers is recognized and accepted by BASIC. This is the scientific or "E-type" notation, and in this format a number is expressed as a decimal number times some power of 10. The form is:

xxEn

where E represents "times 10 to the power of"; thus the number is read, "xx times 10 to the power of n." For example:

25.8E2 = 25.8 * 100 = 2580

PRINT 2*25.8E2 yields the same result as PRINT 2*2580

Data may be input in any one or all three of these forms. Results of computations are output as decimals if they are within the range .01<n<999999; otherwise, they are output in E format. BASIC handles seven significant digits in normal operation and prints 6 decimal digits as illustrated below:

Value Typed In	Value Output by BASIC
.01	.01
.0099	9.9E-03
999999	999999
1000000	1E+06

BASIC automatically suppresses the printing of leading and trailing zeroes in integer and decimal numbers, and, as can be seen from the preceding examples, formats all floating point numbers in the form:

(sign) x.xxxxxE (+ or -)n

where x represents the number carried to six decimal places; E stands for "times 10 to the power of"; and n represents the value of the exponent. For example:

-3.47021E+08	is equal to	-347,021,000
7.26E-04	is equal to	.000726

The floating point format is used when storing and calculating most numbers.

NOTE: Because memory size limitations prohibit the storage of infinite binary numbers, some numbers cannot be expressed exactly in BASIC. Accuracy is approximately seven digits, and errors in the 6th digit can occur. For example; .999998 may be the result of some functions instead of 1. Discrepancies of this type are magnified when such a number is used in mathematical operations.

4.03 Arithmetic Operations

BASIC performs addition, subtraction, multiplication, division and exponentiation. Formulas to be evaluated are represented in a format similar to standard mathematical notation. The five operators used in writing most formulas are:

Symbol Operator	Example	Meaning
+	X+Y	Add Y to X
-	X-Y	Subtract Y from X
*	X*Y	Multiply X by Y
/	X/Y	Divide X by Y
^	X^Y	Raise X to Yth power

BASIC also permits the use of unary plus and minus. The - in -A+B, or the + in +X-Y are examples of such usage. Unary plus is ignored, while unary minus is treated as a zero minus the variable. The expression -A+B is processed as 0-A+B.

4.03.01 Priority of Arithmetic Operations

When more than one operation is to be performed in a single formula, as is most often the case, certain rules must be observed as to the precedence of operators. In any given mathematical formula, BASIC performs the arithmetic operations in the following order of evaluation:

1. Parentheses receive top priority. Any expression within parentheses is evaluated before an unparenthesized expression
2. Exponentiation
3. Unary minus
4. Multiplication and division (of equal priority)
5. Addition and Subtraction (of equal priority)
6. Logical operators in the order NOT, AND, then OR. (see Section 6.7)

If the rules above do not clearly designate the order of priority, then the evaluation of the expression proceeds from left to right. The expression A^B^C is evaluated from left to right as follows:

```
1. A^B                    = step 1
2. (result of step 1)^C   = answer
```

The expression A/B*C is also evaluated from left to right since multiplication and division are of equal priority:

```
1. A/B                    = step 1
2. (result of step 1)*C   = answer
```

The expression A+B*C^D is evaluated as:

```
1. C^D                    = step 1
2. (result of step 1)*B   = step 2
3. (result of step 2)+A   = answer
```

Parentheses may be nested, or enclosed by a second set (or more) of parentheses. In this case, the expression within the innermost parentheses is evaluated first, and then the next innermost, and so on, until all have been evaluated. In the following example:

```
A = 7 * ((B^2+4) / X)
```

the order of evaluation is:

```
1. B^2                    = step 1
2. (result of step 1)+4   = step 2
3. (result of step 2)/X   = step 3
4. (result of step 3)*7   = A
```

Parentheses also prevent any confusion or doubt as to how the expression is evaluated. For example:

```
A*B^2/7+B/C*D^2                    ((A*B^2)/7)+((B/C)*D^2)
```

Both of these formulas are executed in the same way, but the order of evaluaton in the second is made more clear by the use of parentheses.

Spaces may be used in a similar manner. Since the BASIC interpreter ignores spaces (except when enclosed in quotation marks), the two statements:

```
B = D^2 + 1                        B=D^2+1
```

are identical in meaning and consequence, but spaces in the first statement provide ease in reading when the line is entered. When such a statement is subsequently printed by the computer, spaces entered on input are ignored, and the spacing will appear differently on the screen.

4.03 The Assignment Statement

The user assigns a value to a variable by the use of the equals (=) sign. The variable must appear on the left of the statement and its value on the right. For example:

```
A = 2
Q4 = 7.5
```

The statements 2=A , and 7.5=Q4 , although algebraically equivalent to the above examples, are not legal in BASIC, because the machine always takes the value on the right of the equals sign and assigns it to the variable on the left of the sign. The number 2 is not an acceptable variable, and the machine cannot replace its value with that of "A". The fundamental difference in meaning and use of the equals sign in algebra and in BASIC must be clearly understood to avoid confusion. In algebraic notation, the formula X=X+1 is meaningless. However, in BASIC (and in most other computer languages), the equals sign designates replacement rather than equality. Thus, this formula is actually translated: "add one to the current value of X and store the new result back in the same variable X." Whatever value has previously been assigned to X will be combined with the value 1. An expression such as A=B+C instructs the computer to add the values of B and C and store the result in a third variable A. The variable A is not being evaluated in terms of any previously assigned value, but only in terms of B and C. Therefore, if A has been assigned any value prior to its use in this statement, the old value is lost; it is instead replaced by the value B+C. For example:

X=2	Assigns the value 2 to the variable X.
X=X+1+Y	Adds 1 to the current value of X, then adds the value of Y to the result and assigns that value to X.

5. BEGINNING TO PROGRAM

5.01 Sample Program

The lines below form an acceptable BASIC program which the machine will understand and compute. The numbers at the start of each line are an essential part of the program. Each statement must have a line number in order to be executed when the program runs on the machine. The computer will process each line in ascending numerical order, regardless of the order in which it is typed into the machine.

```
10 A=8
20 B=7
30 C=A+B
40 PRINT C
```

The line number itself may be any integer from 0 to 65529, and lines may be numbered in increments as low as 1, but it is a good programming practice to number program statements in increments of 10 or 100. This leaves adequate room for insertion of statements at a later time without the necessity of renumbering the entire program. Hitting the return key at the end of a numbered line automatically enters that line into the computer and stores it in memory.

5.02 The PRINT Statement

Line 40 of the above program is a PRINT statement. This statement is necessary in order to retrieve the calculation the machine has made. After line 30, the computer has solved the problem and assigned the value 15 to the variable C. Without the PRINT statement, however, it will simply store that information for future use, and it will not be visible to the user. The PRINT statement need not always give the value of a single variable; it may contain an expression. Therefore, in the preceding program, the same result would have appeared if the program had read:

```
10 A=8
20 B=7
30 PRINT A+B
```

Other examples of the use of expressions in PRINT statements are:

```
10 A=400
20 PRINT A*975
```

```
10 R = 5
20 P = 3.14159
30 PRINT P*R^2
```

The PRINT statement can also be used to print a message or string of characters, either alone, or together with the evaluation and printing of numeric values. Characters to be printed are enclosed in double quotation marks. For example:

```
10 PRINT "CLASSIFIED"
20 PRINT "INFORMATION"
```

gives:

```
CLASSIFIED
INFORMATION
```

and:

```
10 A=50
20 PRINT "THE NEXT NUMBER IS",A
```

gives:

```
THE NEXT NUMBER IS      50
```

The comma following the quote marks resulted in a tab. And the number itself allowed a leading space for a possible negative sign. This latter point can be illustrated by changing line 20 as follows:

```
20 PRINT "THE NEXT NUMBER IS"A
```

When RUN, the display is:

```
THE NEXT NUMBER IS 50
```

A change in line 10 to:

```
10 A=-50
```

gives a print of:

```
THE NEXT NUMBER IS-50
```

In order to provide a space between "is" and the number, a space may be included within the quotes:

```
20 PRINT "THE NEXT NUMBER IS "A
```

to obtain:

```
THE NEXT NUMBER IS -50
```

When a character string is printed, only the characters between the quotes appear; no leading or trailing spaces are added. Leading and trailing spaces can be added within the quotation marks using the keyboard space bar; spaces appear in the printout exactly as they are typed within the quotation marks.

A convenient shortcut in DISK BASIC is the use of the question mark (?) in place of the word "PRINT" in any PRINT statement. For example:

```
10 ?A          is equal to          10 PRINT A
30 ?"MAGIC"    is equal to          30 PRINT "MAGIC"
```

When the program is listed by the machine, however, the question mark is replaced by the word PRINT. (For a more detailed description of the PRINT statement, see Section 9.01)

5.03 The RUN Command

Once a program has been properly written and entered into the computer, the use of the RUN command will cause it to be processed by the machine and return the result of the program. When the last program line is typed and entered, the user types RUN (or strikes the TAB key with SHIFT held operated) and then hits RETURN. Because RUN is a command and not part of the actual program, it needs no line number. The machine will return the result and the message READY. The READY message indicates that the machine is prepared to accept further additions or changes to the program. For example:

```
Program                          Machine Response

10 R=50
20 T=50
30 PRINT R*T                     2500
RUN                              READY
```

If the user desires to write a completely new program, the machine must be cleared of existing data by re-initializing BASIC. (See 3.02.)

5.04 Corrections

Corrections can be made easily while programming. If, while typing a line, the user makes a mistake, the ← can be used to delete the last character typed. The ← moves the cursor back one space at a time, and it can be struck repeatedly until the error is erased. The line is then retyped from that point, or, if the rest of the original line was correct, the → can be used to restore that portion of the line removed by the ←.

If the line containing the error is already entered, a correction is made by retyping the line correctly, using the same line number. The computer will replace the faulty line with the one most recently typed. If the user desires to delete an entire line from the program, entering that line number and hitting RETURN will remove it from the program. The line currently being entered can be deleted by typing the ERASE LINE key.

The ERASE PAGE key will clear the entire CRT screen, but it does not change or disturb any BASIC statements in any way. It is often used to obtain a blank workspace on the screen while programming.

5.05 The REM Statement

It is often desirable to insert notes and messages within a program. Such data as the name and purpose of the program, how it is used, how certain parts of the program work, and expected results at various points are useful things to have present in the program for ready reference by anyone using that program.

The REMARK or REM statement is used to insert remarks or comments into a program without these comments affecting execution. Remarks do, however, use memory in the user area which may be needed by an exceptionally long program.

The REM statement must be preceded by a line number. The message itself can contain any legal character on the keyboard, including some of the control characters. BASIC ignores anything in a line following the letters REM. Typical REM statements are shown below:

```
10 REM THIS PROGRAM COMPUTES THE
15 REM ROOTS OF A QUADRATIC EQUATION
```

However, if a control character is included following REM, BASIC will "honor" it when doing a listing. For example, if just before typing in QUADRATIC in the illustration above, a CONTROL Q (striking the Q key while the CONTROL key is held down) is executed and then a CONTROL R following the typing of QUADRATIC, the word QUADRATIC in the listing will be in red. Control characters should be used with caution, however; some produce surprising or unwanted results.

5.06 The LIST Command

The user can see a listing of his program on the screen by typing LIST and hitting RETURN. Such a listing makes finding errors much easier, and facilitates additions and changes to the program. A portion of any program may be viewed by typing LIST followed by a line number. The screen will show a listing of that line and all following lines in the program. Because the machine will scroll the program very rapidly, the user may wish to stop the listing at some point for a closer look. Hitting the BREAK key will cause the scrolling to halt. Hitting the RETURN key will resume the listing. To stop the listing altogether, so that the user can edit or change the program, the LINEFEED key (↓) is struck. This will produce the message READY.

The above assumes the Intecolor is still initialized in Scroll Mode. If Page Mode (non-scrolling) were set, a listing of more than a few lines could become very confusing. The listing can be stopped by striking BREAK; then an (ESC) K sequence brings in Scroll Mode; striking RETURN causes the listing to resume.

5.07 The END Statement

The optional END statement is of the form:

```
END
```

Upon executing an END statement, program execution is terminated and the READY message is printed. Program execution can be continued at the statement immediately following the END statement by entering a CONT command. For example, executing the following lines:

```
10 PRINT 1: END: PRINT 2
20 PRINT 3
```

gives the following response:

```
RUN
 1

READY

CONT
 2
 3

READY
```

In this fashion the END statement can be used to generate program breaks to facilitate debugging a program.

Program execution will also terminate automatically when the program runs out of statements. Note that in both cases currently open files are not closed.

5.Ø8 The CONT Command

The CONT command is of the form:

```
CONT
```

This command is used to continue program execution at the next statement after a program break or error is detected. Execution can be restarted at a specific line number by using a GOTO statement instead of CONT.

A CN error message is printed if it is impossible to continue execution after a program break. This message will appear if no program exists or a new or corrected line was entered into the program.

5.Ø9 Multiple Statement Lines

For convenience in programming, DISK BASIC allows the user to place more than one statement on a single numbered line. The general form is:

```
statement:statement: ... :statement
```

where 'statement' is any permissible BASIC statement. Any number of statements may be put together on one line, with the restriction that line length must not exceed 96 characters. The colon (:) denotes new statements and separates them from one another. The statements are executed in order from left to right.

The user must take note of a few statements whose use in multiple statement lines requires some caution.

Because BASIC ignores anything after REM, in the following statement:

```
A=5Ø:B=25:C=4:REM THIS PROGRAM ADDS:PRINT A+B+C
```

the result of A+B+C will never be computed and printed.

Because GOTO causes an immediate and unconditional transfer of control, anything following GOTO in a multiple statement line will never be executed. DATA statements that appear after GOTO's will, however, be read by any corresponding READ statements.

Care must be taken when IF...THEN statements are used in multiple statement lines. If the result of the test is false, control will not pass to the next statement in the line, but rather to the next numbered statement. For example:

```
50 C=2:A=5:IF A=6 THEN PRINT 1:PRINT 2
60 IF C=2 THEN PRINT 3:PRINT 4
```

This program will print out the numbers 3 and 4. If the IF...THEN statement comparison is true and does not pass control to a specific line number, the next statement to the right in the multilple statement line will be executed. For example:

```
40 A=10: IF A=10 THEN B=500: PRINT A+B
```

will result in setting B to 500 and the printing of the result of A+B.

5.10 Introduction to Strings

The previous sections described the manipulation of numerical information only; however, DISK BASIC also processes information in the form of character strings. A string, in this context, is a sequence of characters treated as a unit. A string is composed of alphabetic, numeric, or special characters. The maximum length of quoted strings and strings entered using the INPUT statement is determined by the length of the input line buffer which is 96 characters or bytes. (Characters in excess of 96 will be rejected. However, if a longer string is needed it can be broken into two or more strings and the strings concatenated, as in the example at the end of this section.)

Any variable name followed by a dollar sign ($) character indicates a string variable. For example:

```
A$
C7$
LONG$
```

are simple string variables and can be used as follows:

```
10 A$="HELLO"
20 PRINT A$
```

Note that the string variable A$ is separate and distinct from the variable A. In DISK BASIC, all control characters above control code C (or 3) are legal characters within quotes (") except for the following:

Control Code K or 11 or erase line
Control Code L or 12 or erase page
Control Code M or 13 or return/enter
Control Code Y or 25 or cursor right
Control Code Z or 26 or cursor left

Concatenation is a string operator that puts one string after another without any intervening characters. It is specified by a plus sign (+) and works only with strings. The maximum length of a concatenated string is 255 characters. In each of the following examples, D$ contains the result of concatenating the strings A$, B$, and C$.

```
10 A$ = "33"
20 B$ = "22"
30 C$ = "44"
40 D$ = A$+B$+C$
50 PRINT D$

RUN
332244
```

```
10 A$ = "I AM"
20 B$ = " A CLEVER"
30 C$ = " INTECOLOR"
40 D$ = A$+B$+C$
50 PRINT D$

RUN
I AM A CLEVER INTECOLOR
```

5.11 The CLEAR Statement

The CLEAR statement clears all the user's variables including simple variables and arrays. The CLEAR statement has two forms as shown below:

```
CLEAR
```

and

```
CLEAR expression
```

The difference between the two forms is that the form with the expression specifies the new number of bytes in the string space. Upon entry to BASIC the string space is initialized to 50 bytes. For example, in programs that heavily use strings, this allocation can be changed by executing a CLEAR 250; it should be one of the first executed statements in a program because it also clears all the variables. It should be noted that memory CLEARED for use by string variables decreases the amount of memory available to BASIC. It would be possible to allocate BASIC's RAM for use by string variables, leaving no space for other variables or BASIC commands — a bit far-fetched, but possible. For further information on how strings are allocated in the string space, see Section 9.04.

5.12 Immediate Mode

It is not necessary to write a complete program to use BASIC. Most of the statements discussed in this manual can be included in a program for later execution or given as commands which are immediately executed by the DISK BASIC interpreter. This latter facility makes BASIC an extremely powerful calculator.

BASIC distinguishes between lines entered for later execution and those entered for immediate execution solely by the presence (or absence) of line numbers. Statements which begin with line numbers are stored as part of the program; statements without line numbers are executed immediately upon being entered into the system. Thus the line:

```
10 PRINT "THIS IS AN INTECOLOR 3651"
```

produces no action at the console upon entry, while the statement:

```
PRINT "THIS IS AN INTECOLOR 3651"
```

causes the immediate output:

```
THIS IS AN INTECOLOR 3651
```

Multiple statements can be used on a single line in immediate mode. For example:

```
A=1:PRINT A     gives:          1
```

Program loops are also allowed in immediate mode; thus a table of squares can be produced as follows: (For a description of FOR NEXT loops, see Section 6.9)

```
FOR I=1 TO 10: PRINT I, I^2:NEXT I

 1        1
 2        4
 3        9
 4        16
 5        25
 6        36
 7        49
 8        64
 9        81
 10       100

READY
```

5.13 Samples and Examples

In order to become more adept at programming, any user previously unfamiliar with BASIC should set aside some time for experimentation with the information thus far provided in this manual. Simple programs such as the ones on the next page make good practice efforts.

A

```
10 REM THIS PROGRAM COMPUTES
20 REM THE AREA OF A CIRCLE
30 REM THE FORMULA IS:
40 REM AREA = PI * RADIUS ^ 2
50 PI = 3.14159
60 R = 25
70 A = PI * R ^ 2
80 PRINT "AREA = ",A
```

B

```
10 REM THIS PROGRAM AVERAGES
20 REM FIVE NUMBERS
30 A=23
40 B=1
50 C=188
60 D=5
70 E=89
80 T=A+B+C+D+E
90 AV=T/5
95 PRINT "AVERAGE =",AV
```

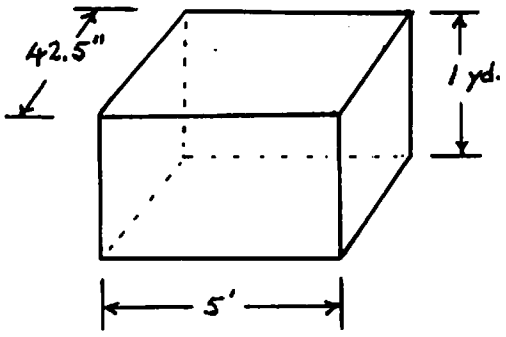
Write programs to solve these problems:

A

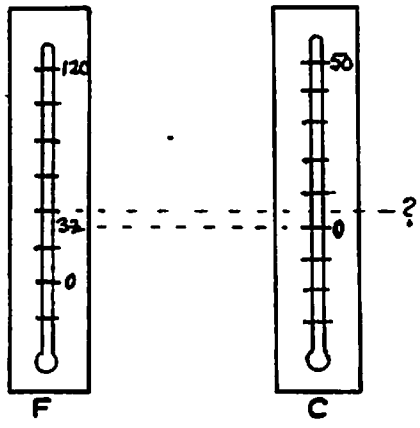
How many cubic yards of soil can be into put into a box that measures 5 feet by 42.5 inches by 1 yard?

B

Convert 40 degrees Fahrenheit into degrees Celsius using the formula C = (5/9)*(F-32)



A. Box



B. Temperature Conversion

6. MORE STATEMENTS, COMMANDS, AND FEATURES

6.01 The INPUT Statement

The INPUT statement is used when data values are to be entered from the terminal keyboard during program execution. The form of the statement is:

```
INPUT list
```

where 'list' is a list of variable names separated by commas. For example:

```
10 INPUT A,B,C
```

causes the computer to pause during execution, print a question mark, and wait for the entry of three numeric values separated by commas. The values are input to the computer by typing the RETURN key.

If too few values are entered, BASIC prints another ? to indicate that more data values are needed. If too many values are used, the excess data values on that line are ignored, but the program will continue. The values entered in response to the INPUT statement cannot be continued on another line and are terminated by the RETURN key. Values must be separated by commas if more than one value is entered on the same line.

When reading numeric values, spaces are ignored. When a non-space is found, it is assumed to be part of a number; if not, then the question mark is repeated. The number is terminated by a comma, colon, or carriage return.

When reading string items, leading spaces are ignored. When a non-space character is found, it is assumed to be the start of a string item. If this first character is a quotation mark ("), the item is taken as being a quoted string and all characters between the first double quote (") and a matching double quote or carriage return are returned as characters in the string. Thus, quoted strings may contain any legal character except double quote. If the first non-space character is not a double quote, then it is assumed to be an unquoted string constant. The string will terminate with a comma, colon, or carriage return.

When there are several values to be entered via the INPUT statement, it is helpful to print a message explaining the data needed. For example:

```
10 PRINT "YOUR AGE IS"
20 INPUT A
```

The INPUT statement can also contain quoted strings. The above example could be written:

```
10 INPUT "YOUR AGE IS? ";A
```

Note that when a quoted string is included in an INPUT statement, the normal ? is not printed as a prompt character, and if desired, must be included as shown within the quotes above.

The INPUT statement allows BASIC to be programmed to accept direct questions and answers as well as fill-in-the-blank applications.

If the user wishes to stop a program while it is waiting at an input statement, LINEFEED and RETURN must be typed in sequence. If RETURN is typed in response to the INPUT prompt (?), DISK BASIC will assume the value Ø for numeric variables, and "" (the quotation marks, but no information between them) for string variables. If there are additional variables in the INPUT list, a question mark (?) will be printed as discussed above.

6.Ø2 The DATA Statement

The DATA statement is used in conjunction with the READ statement to enter data into an executing program. One statement is never used without the other. The form of the statement is:

```
DATA value list
```

where value list contains the numbers or strings to be assigned to the variables listed in a READ statement. Individual items in the value list are separated by commas; strings are usually enclosed in quotation marks. For example:

```
15Ø DATA 4,7,2,3,"ABC"
17Ø DATA 1,34E-3,3,171311
```

The scanning of numeric and string items is identical to that described above in the INPUT statement. An SN error message can result from an improperly formatted DATA list.

The location of DATA statements is arbitrary as long as they appear in the correct order; however, it is good practice to collect all related DATA statements near each other.

When the RUN command is executed, BASIC searches for the first DATA statement and saves a pointer to its location. Each time a READ statement is encountered in the program, the next value in the DATA statement is assigned to the designated variable. If there are no more values in that DATA statement, BASIC looks for the next DATA statement.

6.Ø3 The READ Statement

A READ statement is used to assign the values listed in the DATA statements to the specified variables. The READ statement is of the form:

```
READ variable list
```

The items in the variable list may be simple variable names or string variable names and are separated by commas. For example:

```
1Ø READ A, B$, C
2Ø DATA 12, "42", .12E2
```

Since data must be read before it can be used in a program, READ statements generally occur near the beginning of the program. A READ statement can be placed anywhere in a multiple statement line.

If there are no data values available in the DATA statements for the READ to store, the out of data message below is printed:

```
OD ERROR IN xxxxx
READY
```

Items in the data list in excess of those needed by the program's READ statements are ignored.

6.04 The RESTORE Statement

The RESTORE statement causes the program to reuse the data from the first DATA statement, or, if a line number is specified, from the first DATA statement on or after the specified line. The two forms of the RESTORE statement are as follows:

```
RESTORE
```

and

```
RESTORE line number
```

For example:

```
100 RESTORE 50
```

causes the next READ statement to start reading data from the first DATA statement on or after line 50. The following example shows how the RESTORE statement functions:

```
10 INPUT "ENTER 1 FOR NUMERIC, 2 FOR STRINGS:"; A
20 IF A = 2 THEN 200
100 RESTORE 190
110 FOR I = 1 TO 5 READ B: PRINT B: NEXT I
120 GOTO 10
190 DATA 10, 20, 30, 40, 50, 60
200 RESTORE 290
210 FOR I = 1 TO 5 READ B$: PRINT B$: NEXT I
220 GOTO 10
290 DATA "APPLE", "BOY", "CAT", "DOG", "ELEPHANT", "FOX"
```

If a 2 is entered, the first 5 string data values in line 290 are printed; otherwise, the first 5 numeric data values on line 190 are printed. The sixth data items in lines 190 and 290 are not read.

6.05 The GOTO Statement

The GOTO statement is used when it is desired to unconditionally transfer to some line other than the next sequential line in the program. In other words, a GOTO statement causes an immediate jump to a specified line, out

of the normal consecutive line number order of execution. The general form of the statement is as follows:

```
GOTO line number
```

The line number to which the program jumps can be either greater or lower than the current line number. It is thus possible to jump forward or backward within a program. For example:

```
10 A=2
20 GOTO 50
30 A=SQR(A+14)
50 PRINT A,A*A
RUN
```

causes the following output:

```
 2        4
```

When the program encounters line 20, control transfers to line 50; line 50 is executed, control then continues to the line following line 50. Line 30 is never executed. Any number of lines can be skipped in either direction.

When written as part of a multiple statement line, GOTO should always be the last executable statement on the line, since any statement following the GOTO on the same line is never executed. For example:

```
110 A=ATN(B2):PRINT A:GOTO 50
```

However, REM and DATA statements can follow a GOTO on the same line because they are non-executable statements.

6.06 Relational Operators

Relational operators allow comparison of two values and are usually used to compare arithmetic expressions or strings in an IF...THEN statement. The relational operators are:

MATHEMATICAL SYMBOL	BASIC SYMBOL	EXAMPLE	MEANING
=	=	A=B	A is equal to B
<	<	A<B	A is less than B
≤	<= , =<	A<=B	A is less than or equal to B
>	>	A>B	A is greater than B
≥	>= , =>	A>=B	A is greater than or equal to B
≠	<> , ><	A<>B	A is not equal to B

The result of the relational operators is -1 for true and 0 for false.

6.06.01 Relational Operators in Strings

When applied to string operands, the relational operators test alphabetic sequence. Comparison is made character by character on the basis of the ASCII codes (See Appendix E) until a difference is found. If, while the comparison is proceeding, the end of one string is reached, the shorter string is considered to be smaller. For example:

```
55 IF A$<B$ THEN 100
```

When line 55 is executed, the first characters of each string (A$ and B$) are compared, then the second characters of each string, and so on until the character in A$ is less than the corresponding character in B$. If this test is true, execution continues at line 100. Essentially, the strings are compared for alphabetic order. Below is a list of the relational operators and their string interpretations.

In any string comparison, leading and trailing blanks are significant (i.e., "ABC" is not equivalent to "ABC ").

OPERATOR	EXAMPLE	MEANING
=	A$=B$	The strings A$ and B$ are alphabetically equal.
<	A$<B$	The string A$ alphabetically precedes B$.
>	A$>B$	The string A$ alphabetically follows B$.
<=	A$<=B$	The string A$ is equivalent to or precedes B$ alphabetically.
>=	A$>=B$	The string A$ is equivalent to or follows B$ alphabetically.
<>	A$<>B$	The strings A$ and B$ are not alphabetically equal.

6.07 Logical Operators

Logical operators are typically used as Boolean operators in relational expressions. For example, consider the following two sequences of statements:

```
100 IF A = B THEN 150
110 IF C < D THEN 150
```

and

```
200 IF A <> 5 THEN 220
210 IF B = 10 THEN 250
220 ...
```

In both cases the sequences can be simplified by using the logical operators AND and OR. The first two statements can be combined into a single statement:

```
100 IF A = B OR C < D THEN 150
```

Similarly, the second sequence of statements is equivalent to:

```
200 IF A = 5 THEN IF B = 10 THEN 250
220 ...
```

This can be further simplified to:

```
200 IF A = 5 AND B = 10 THEN 250
```

Following the rules of Boolean algebra, the unary operator NOT will change true into false and vice versa. For example:

```
100 IF A <> 5 THEN 150
```

is equivalent to:

```
100 IF NOT (A=5) THEN 150
```

More complex expressions can be constructed by using combinations of the AND, OR, and NOT operators.

Logical operators may also be used for bit manipulation and Boolean algebraic functions. The AND, OR, and NOT operators convert their arguments into sixteen bit, signed, two's complement integers in the range -32768 to 32767. After the operations are performed, the result is returned in the same form and range. If the arguments are not in this range, a CF error message will be printed and execution will be terminated. Truth tables for the logical operators appear below. The operations are performed bitwise, that is, corresponding bits of each argument are examined and the result computed one bit at a time. In binary operations, bit 7 is the most significant bit of a byte and bit 0 is the least significant.

AND

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

NOT

X	NOT X
1	0
0	1

Some examples will serve to show how the logical operators work:

63 AND 16=16	63 = binary 111111 and 16 = binary 10000, so 63 AND 16 = 16
15 AND 14=14	15 = binary 1111 and 14 = binary 1110 , so 15 AND 14 = binary 1110 = 14
-1 AND 8=8	-1=binary 1111111111111111 and 8=binary 1000, so -1 AND 8 = 8
4 OR 2=6	4 = binary 100 and 2 = binary 10 so 4 OR 2 = binary 110 = 6
10 OR 10=10	binary 1010 OR'd with itself is 1010=10
-1 OR 2=-1	-1 = binary 1111111111111111 and -2 = 1111111111111110, so -1 OR -2 = -1
NOT 0=-1	the bit complement of sixteen zeros is sixteen ones, which is the two's complement representation of -1
NOT X=-(X+1)	the two's complement of any number is the bit complement plus one.

A typical use of logical operations is "masking"—testing a binary number for some predetermined pattern of bits. Such numbers might come from the computer's input ports and would then reflect the condition of some external device.

6.08 The IF...THEN and IF...GOTO Statements

The IF-THEN statement is used to transfer control conditionally from the normal consecutive order of statement numbers, depending upon the truth of some mathematical relation or relations. The basic form of the IF statement is as follows:

IF expression THEN line number
IF expression GOTO line number

where 'expression' is an arithmetic expression. If the result of the expression is nonzero (true), execution begins at the line number given and proceeds as usual. If the value of the expression is zero (false), the next statement in numerical order will be executed. Usually the statement is of the form:

```
                                        THEN
IF expression  rel. op. expression           line number
                                        GOTO
```

In this case, expressions cannot be mixed; both must be string or both must be numeric. Numeric comparisons are handled as described in 6.06. String comparisons are performed on the ASCII values of the strings as described in 6.06.01 and Appendix E. The rel. op. (relational operator) must be as described in 6.06, and the line number is the line of the program to which control is conditionally passed.

If the value of the expression is true, control passes to the line number specified. If the value of the expression is false, control passes to the next statement in sequence. For example:

```
30 IF A=B THEN 20                    40 IF A<>71 GOTO 20
40 PRINT A+B                         55 PRINT A
50 PRINT A^2                         60 D=A+B+*C
```

An alternate form of the IF...THEN statement is as follows:

```
IF expression THEN statement
```

where the statement is any valid DISK BASIC statement. Note that multiple statements can follow the THEN if they are separated by colons (:). With this form of the IF...THEN statement, if the expression evaluates to non-zero (true), the statements following the THEN are executed. Otherwise, control passes to the next numbered line. For example:

```
10 A=10
20 IF A=10 THEN PRINT "TRUE":GOTO 40
30 PRINT "FALSE"
40 END
```

6.09 The FOR and NEXT Statements

FOR and NEXT statements define the beginning and end of a loop. (A loop is a set of instructions which are repeated over and over again, each time being modified in some way until a terminal condition is reached.) The FOR statement is of the form:

```
FOR variable = expression1 TO expression2 STEP expression3
```

where the variable is the index, expression1 is the initial value, expression2 is the terminal value, and expression3 is the incremental value. For example:

```
15 FOR K=2 TO 20 STEP 2
```

causes the program to execute the designated loop as long as K is less than or equal to 20. Each time through the loop, K is incremented by 2, so the loop is executed a total of 10 times. After executing the loop, when K=20, program control passes to the line following the associated NEXT statement, and the value of K is 22.

The index variable must be unsubscripted, although such loops are commonly used in dealing with subscripted variables. In such a case the control variable is used as the subscript of a previously defined variable. The expressions in the FOR statement can be any acceptable BASIC expression.

The NEXT statement signals the end of the loop which began with the FOR statement. The NEXT statement is of the form:

```
NEXT variable
```

where the variable is the same variable specified in the FOR statement. The variable is actually optional, since any NEXT statement encountered is assumed by the computer to be closing the loop for the appropriate FOR variable. Together the FOR and NEXT statements define the boundaries of a program loop. When execution encounters the NEXT statement, the computer adds the STEP expression value to the variable and checks to see if the variable is still less than or equal to the terminal expression value. When the variable exceeds the terminal expression value, control falls through the loop to the statement following the NEXT statement. Note that the variable is not necessary since when a NEXT statement is encountered it is assumed it is for the appropriate FOR loop variable.

If the STEP expression and the word STEP are omitted from the FOR statement, +1 is the assumed value. Since +1 is a common STEP value, that portion of the statement is frequently omitted.

The expressions within the FOR statement are evaluated once upon initial entry into the loop. The test for completion of the loop is made after each execution of the loop. (If the test fails initially, the loop is still executed once.)

The index variable can be modified within the loop. When control falls through the loop, the index variable retains the value used to fall through the loop.

The following is a demonstration of a simple FOR-NEXT loop. The loop is executed 10 times; the value of I is 11 when control leaves the loop; and +1 is the assumed STEP value:

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
40 PRINT I
```

works

The loop itself is defined by lines 10 through 30. The numbers 1 through 10 are printed when the loop is executed. After I=10, control passes to line 40 which causes 11 to be printed. If line 10 had been:

```
10 FOR I = 10 TO 1 STEP -1
```

the value printed by line 40 would have been 0.

The following loop is executed only once since the value of I=44 has been reached and the termination condition is satisfied.

```
10 FOR I = 2 TO 44 STEP 2
20 I = 44
30 NEXT I
```

If the initial value of the variable is greater than the terminal value, the loop is still executed once. The loop set up by the statement:

```
10 FOR I = 20 TO 2 STEP 2
```

will be executed only once although a statement like the following will initialize execution of a loop properly:

```
10 FOR I = 20 TO 2 STEP -2
```

For positive STEP values, the loop is executed until the control variable is greater than its final value. For negative STEP values, the loop continues until the control variable is less than its final value.

FOR loops can be nested but not overlapped. The depth of nesting depends upon the amount of user storage space available; in other words, upon the size of the user program and the amount of RAM available. Nesting is a programming technique in which one or more loops are completely within another loop. The field of one loop (the numbered lines from the FOR statement to the corresponding NEXT statement, inclusive) must not cross the field of another loop. For example:

ACCEPTABLE NESTING TECHNIQUES

Two-Level Nesting

```
10 FOR I1 = 1 TO 10
20 FOR I2 = 1 TO 10
30 NEXT I2
40 FOR I3 = 1 TO 10
50 NEXT I3
60 NEXT I1
```

Three-Level Nesting

```
10 FOR I1 = 1 TO 10
20 FOR I2 = 1 TO 10
30 FOR I3 = 1 TO 10
40 NEXT I3
50 FOR I4 = 1 TO 10
60 NEXT I4
70 NEXT I2
80 NEXT I1
```

UNACCEPTABLE NESTING TECHNIQUES

Two-Level Nesting

```
10 FOR I1 = 1 TO 10
20 FOR I2 = 1 TO 10
30 NEXT I1
40 NEXT I2
```

Three-Level Nesting

```
10 FOR I1 = 1 TO 10
20 FOR I2 = 1 TO 10
30 FOR I3 = 1 TO 10
40 NEXT I3
50 FOR I4 = 1 TO 10
60 NEXT I4
70 NEXT I1
80 NEXT I2
```

It is possible to exit from a FOR-NEXT loop without the control variable reaching the termination value. A conditional or unconditional transfer can be used to leave a loop. Control can only transfer into a loop which has been left earlier without being completed, ensuring that termination and STEP values are assigned.

Both FOR and NEXT statements can appear anywhere in a multiple statement line. For example:

```
10 FOR I = 1 TO 10 STEP 5: NEXT I: PRINT "I="; I
```

causes:

```
I= 11
```

to be printed when executed.

In the case of nested loops which have the same endpoint, a single NEXT statement of the following form can be used:

```
NEXT variable 1, ... , variable N
```

The first variable in the list must be that of the most recent loop, the second most recent, and so on. For example:

```
10 FOR I=1 TO 10
20 FOR J=1 TO 10
30 ...
100 NEXT J,I
```

7. FUNCTIONS AND SUBROUTINES

7.01 Functions

BASIC provides functions to perform certain standard mathematical operations which are frequently used and time-consuming to program. These functions have three or four letter call names followed by a parenthesized argument. They are pre-defined and may be used anywhere in a program.

Call Name	Function
ABS(x)	Returns the absolute value of x.
ATN(x)	Returns the arctangent of x as an angle in radians in range $+\pi/2$), where π = 3.14159.
CALL(x)	Call the user machine language program at decimal location 33282. (8202 HEX) The D,E registers have value of X upon entry and value of Y upon return from machine language routine.
COS(x)	Returns the cosine of x radians.
EXP(x)	Returns the value of e^x where e = 2.71828.
FRE(x)	Returns the number of free bytes not in use.
INT(x)	Returns the greatest integer less than or equal to x.
INP(x)	Returns a byte from input port x. The range for x is 0 to 255.
LOG(x)	Returns the natural logarithm of x.
PEEK(x)	Returns a byte from memory address -32768<x<65535; or if x is negative the memory address is 65536+x.
POS(x)	Returns the value of the current cursor character position between 0 and 63 (X value only).
RND(x)	Returns a pseudo random number between 0 and 1.
SGN(x)	Returns a -1, 0, or 1, indicating the sign of x.
SIN(X)	Returns the sine of x radians.
SPC(x)	Causes x spaces to be generated. (Valid only in a PRINT statement).
SQR(x)	Returns the square root of x.
TAB(x)	Causes the cursor to space over to column number x. (Valid in PRINT statement only).
TAN(x)	Returns the tangent of x radians.

The argument x to the functions can be a constant, a variable, an expression, or another function. Square brackets cannot be used as the enclosing characters for the argument x, e.g. SIN[x] is illegal.

Function calls, consisting of the function name followed by a parenthesized argument, can be used as expressions anywhere that expressions are legal.

Values produced by the functions SIN(x), COS(x), ATN(x), SQR(x), EXP(x), and LOG(x) have six significant digits.

7.01.01 The Sine and Cosine Functions; SIN(x) and COS(x)

The SIN and COS functions require an argument angle expressed in radians. If the angle is stated in degrees, conversion to radians may be done using the identity:

radians = degrees * (π/180)

In the following program, 3.14159 is used as a nominal value for π. P is set equal to this value at line 20. At line 40 the above relationship is used to convert the input value into radians. Note the use of the TAB function to produce a more legible printout.

```
10 REM CONVERT ANGLE (X) TO RADIANS, AND
11 REM FIND SIN AND COS
20 P = 3.14159
25 PRINT "DEGREES",, "RADIANS",, "SINE",, "COSINE"
30 FOR X = 0 TO 90 STEP 15
40 Y = X*(P/180)
60 PRINT X,,Y;TAB(32); SIN(Y); TAB(48); COS(Y)
70 NEXT X

RUN
DEGREES         RADIANS         SINE            COSINE
 0               0               0               1
 15              .261799         .258819         .965926
 30              .523598         .5              .866026
 45              .785398         .707106         .707107
 60              1.0472          .866025         .500001
 75              1.309           .965926         .25882
 90              1.5708          1               1.12352E-06
```

7.01.02 The Arctangent and Tangent Functions; ATN(x) and TAN(x)

The arctangent function returns a value in radian measure, in the range - π/2 to + π/2 corresponding to the value of a tangent supplied as the argument (x).

In the following program, the input is an angle in degrees. Degrees are then converted to radians at line 50. At line 70 the tangent value, Z, is supplied as the argument to the ATN function to derive the value found on column 4 of the printout under the label ATN(x). Also in line 70 the

radian value of the arctangent function is converted back to degrees and printed in the fifth column of the printout as a check against the input value shown in the first column.

```
10 P = 3.14159
15 PRINT
20 PRINT "ANGLE","ANGLE";TAB(20);"TAN(X)";
21 PRINT TAB(32);"ATAN(X)",,"ATAN(X)"
25 PRINT "(DEGS)", "(RADS)",,,"(RADS)",,"(DEGS)"
30 FOR X = 0 TO 45 STEP 15
35 PRINT
40 FOR X = 0 TO 75 STEP 15
50 Y = X*P/180
60 Z = TAN(Y)
70 PRINT X,Y;TAB(20);Z;TAB(32);ATN(Z);TAB(48);ATN(Z)*180/P
80 NEXT X
RUN
```

ANGLE (DEGS)	ANGLE (RADS)	TAN(X)	ATAN(X) (RADS)	ATAN(X) (DEGS)
0	0	0	0	0
15	.21799	.267949	.261799	15
30	.523598	.57735	.523598	30
45	.785398	.999999	.785398	45
60	1.0472	1.73205	1.0472	60
75	1.309	3.73204	1.309	75

7.01.03 The Square Root Function; SQR(x)

This function derives the square root of any positive number as shown below:

```
10 INPUT X
20 X = SQR(X)
30 PRINT X
40 GOTO 10
RUN
?16
 4
?1000
 31.6228
?(LINEFEED) (RETURN)
READY
```

If the argument is negative, a CF error will result.

7.01.04 The Exponential and Logarithmic Functions; EXP(x) and LOG(x)

The exponential function raises the number e to the power x. EXP is the inverse of the LOG function. The relationship is:

```
LOG(EXP(X)) = X = EXP(LOG(X))
```

The following program prints the exponential equivalent of an input value.

```
10 INPUT X
20 PRINT EXP(X), LOG(EXP(X)), EXP(LOG(x))
30 GOTO 10
RUN

?87
 6.07601E+37      87        87
?.0033
 1.00331          3.2999E-03        3.3E-03
?1
 2.71828          1         1
```

Logarithms to the base e may easily be converted to any other base using the following formula:

$$\log \ N = \frac{\log_{e} N}{\log_{e} a}$$

where a represents the desired base and e = 2.71828. The following program illustrates conversion to the bases 10 and 2.

```
10 PRINT "VALUE","BASE E LOG","BASE 10 LOG","BASE 2 LOG"
20 INPUT X
30 PRINT X,LOG(X);TAB(24);LOG(X)/LOG(10);
40 PRINT TAB(40);LOG(X)/LOG(2)
50 GOTO 20
RUN
VALUE    BASE E LOG       BASE 10 LOG      BASE 2 LOG
?1
 1        0                0                0
?4
 4        1.38629          .60206           2
?10
 10       2.30259          1                3.32193
?1000
 1000     6.90776          3                9.96579
```

An attempt to find the LOG of zero or of a negative number causes a CF error message.

7.01.05 The Absolute Value Function; ABS(x)

The ABS function returns the absolute value of any argument. The absolute value is the argument itself with a positive sign. For example the absolute value of both 3 and -3 is 3. The ABS function may be illustrated as follows:

```
PRINT ABS(12.34),ABS(-23.65)
 12.34    23.65
```

7.01.06 The Greatest Integer Function; INT(x)

The greatest integer function returns the value of the greatest integer not greater than x. For example:

```
PRINT INT(34.67)
 34

PRINT INT(11)
 11
```

The INT of a negative number is a negative number with the same or larger absolute value, i.e., the same or smaller algebraic value. For example:

```
PRINT INT(-23.45)
-24

PRINT INT(-11)
-11
```

The INT function can be used to round numbers to the nearest integer, using INT(X+.5). For example:

```
PRINT INT(34.67+.5)
 35

PRINT INT(-5.1+.5)
-5
```

INT(x) can also be used to round to any given decimal place or integral power of 10, by using the following expression as an argument:

```
(X*10^D+.5)/10^D
```

where D is an integer supplied by the user.

```
10 REM INT FUNCTION EXAMPLE
15 PRINT
20 PRINT "NUMBER TO BE ROUNDED:"
25 INPUT A
40 PRINT "NO. OF DECIMAL PLACES:"
45 INPUT D
60 B = INT(A*10^D +.5)/10^D
70 PRINT "NUMBER ROUNDED = " ;B
80 GOTO 15
RUN

NUMBER TO BE ROUNDED
?55.65842
NO. OF DECIMAL PLACES:
?2
NUMBER ROUNDED =  55.66
```

```
NUMBER TO BE ROUNDED
?78.375
NO. OF DECIMAL PLACES:
?-2
NUMBER ROUNDED =  100

NUMBER TO BE ROUNDED
?67.38
NO. OF DECIMAL PLACES:
?-1
NUMBER ROUNDED =  70

NUMBER TO BE ROUNDED
?(LINEFEED) (RETURN)

READY
```

7.01.07 The Random Number Function; RND(x)

The random number function produces a random number, or random number set between 0 and 1. (Perhaps "pseudo random" would be a better term. In many cases the first number is not truly random but is likely the same each time the equipment is turned on.) The numbers are reproducible in the same order after the ESC, E sequence if X>0 for later checking of a program. In DISK BASIC the form RND without arguments is not legal. For example:

```
10 PRINT "RANDOM NUMBERS:
30 FOR I = 1 TO 8
40 PRINT RND(I),
50 NEXT I
RUN
RANDOM NUMBERS:
 .100250        .968134         .886657         .636444         .839019
 .306121        .285553         .285534
```

(The print format may vary a bit, depending on whether the numbers have less than six digits.)

To obtain random digits from 0 to 9, line 40 can be changed to read:

```
40 PRINT INT(10*RND(1)),
```

This time the results will be printed as follows:

```
RANDOM NUMBERS:
 8      9       3       5       6       1       8       2
```

It is possible to generate random numbers over a given range. If the open range (A,B) is desired, use the expression:

```
(B-A)*RND(1)+A
```

to produce a random number in the range A<n<B. The following program produces a random number set in the open range (4,6). The extremes, 4 and 6, are never reached.

```
10 REM RANDOM NUMBER SET IN OPEN RANGE 4,6.
20 FOR B = 1 TO 8
30 A = (6-4) * RND(1) + 4
40 PRINT A,
50 NEXT B
RUN
 4.20054        5.92962        5.77325        5.27288
 4.99125        5.02420        4.18825        5.99989
```

Negative arguments, i.e. RND(-123), will start a new random number sequence, while RND(0) will always generate the last random number.

7.01.08 The Sign Function; SGN(x)

The sign function returns the value 1 if x is a positive number, 0 if x is 0 and -1 if x is negative. For example:

```
10 REM SGN FUNCTION EXAMPLE
20 READ A,B,C
25 PRINT "A = "A,"B = "B,"C = "C
30 PRINT "SGN(A) = "SGN(A), "SGN(B) = "SGN(B),
40 PRINT "SGN(C) = "SGN(C)
50 DATA -7.32, .44, 0
RUN
A = -7.32       B =  .44        C =  0
SGN(A) = -1     SGN(B) =  1     SGN(C) =  0
```

7.01.09 The Position Function; POS(x)

The POS function returns the current x coordinate of the cursor's position. It is most often used to determine whether or not a particular program result, either string or numeric, will fit on a given line. By use of the POS(x) function, the correct placement of the answer can be easily determined.

7.02 User Defined Functions

In some programs it may be necessary to execute the same sequence of statements or mathematical formulas in several different places. BASIC allows definition of unique operations or expessions and the calling of these functions in the same way as the predefined standard mathematical functions.

These user defined functions are described by a function name, the first two letters of which are FN followed by any acceptable BASIC variable name. For example:

Legal	Illegal
FNA	FNA$
FNAA	FNZ
FNA1	

Each function is defined once and the definition may appear anywhere in the program. The defining or DEF statement is formed as follows:

```
DEF FNA (argument) = expression
```

where A is a variable name. The argument must be a simple variable. The expression may contain the argument variable and any other program variables. For example:

```
10 DEF FNA(S) = S^2
```

causes a later statement:

```
20 R = FNA(4)+1
```

to be evaluated as R = 17. As another example:

```
50 DEF FNB(A) = A+X^2
60 Y= FNB(14)
```

causes the function to be evaluated with the current value of the variable X within the program. The two following programs:

```
10 DEF FNS(A) = A^A
20 FOR I=1 TO 5
30 PRINT I, FNS(I)
40 NEXT I
```

```
10 DEF FNS(X) = X^X
20 FOR I=1 TO 5
30 PRINT I, FNS(I)
40 NEXT I
```

cause the same output:

```
RUN
 1        1
 2        4
 3        27
 4        256
 5        3125
```

User defined functions cannot have several arguments, as shown below:

```
25 DEF FNL(X,Y,Z) = SQR(X^2 + Y^2 + Z^2)
```

Such a statement will cause an error of the type:

```
SN ERROR IN 25
```

When calling a user defined function, the parenthesized argument can be any legal expression. The value of the expression is substituted for the argument variable. For example:

```
10 DEF FNZ(X) = X^2
20 A=2
30 PRINT FNZ(2+A)
```

Line 30 causes the result 16 to be printed.

If the same function name is defined more than once, then the last defintion (the one with the higher line number) will be used. The program below:

```
10 DEF FNX(X) = X^2
20 DEF FNX(X) = X+X
30 A=5
40 PRINT FNX(A)
```

will cause 10 to be printed.

The function variable need not appear in the function expression as shown below:

```
10 DEF FNA (X) = 4+2
20 R=FNA(10)+1
30 PRINT R
RUN
 7
```

7.03 BASIC String Functions

Like the intrinsic mathematical functions described above, BASIC contains various functions for use with character strings. These functions allow the program to concatenate two strings, access part of a string, determine the number of characters in a string, generate a character string corresponding to a given number or vice versa, and perform other useful operations. The various functions available are summarized in the following table.

STRING FUNCTIONS

Call Name	Function
ASC(x$)	Returns the eight bit internal ASCII code (0-255) for the one-character string. If the argument contains more than one character, then the code for the first character in the string is returned. A value of 0 is returned if the argument is a null string (LEN(x$)=0). (See ASCII codes in Appendix F).
CHR$(x)	Generates a one-character string having the ASCII value of x where x is a number greater than or equal to 0 and less than or equal to 255. Only one character can be generated.
FRE(x$)	Returns number of free string bytes. (See CLEAR statement in 5.11)
LEFT$(x$,I)	Returns left-most I characters of string (x$). If I>LEN(x$), then x$ is returned.
LEN(x$)	Returns the number of characters in the string x$, with non-printing characters and blanks being counted.

MID$(x$,I,J) J is optional. Without J, returns right-most characters from x$ beginning with the Ith character. If I>LEN(x$), MID$ returns the null string. With 3 arguments, it returns a string of length J of characters from x$ beginning with the Ith character. If J is greater than the number if characters in x$ to the right of I, MID$ returns the rest of the string. Argument ranges: 0<I<=255, 0<=J<=255.

RIGHT$(x$,I) Returns right-most I characters of string (x$). If I>LEN(x$), then x$ is returned.

STR$(x) Returns the string which represents the numeric value of x as it would be printed by a PRINT statement.

VAL(x$) Returns the number represented by the string x$. If the first character of x$ is not +, -, or a digit, then the value 0 is returned.

In the above descriptions, x$ represents any legal string expression, and I and J represent any legal arithmetic expressions.

NOTE: Unlike the mathematical functions, character string functions cannot be defined by the user. Similar results can be obtained by the use of subroutines, as described in Section 7.04.

7.04 Subroutines

A subroutine is a section of a program performing some operation required at more than one point in the program. Sometimes a complicated I/O operation for a volume of data, a mathematical evaluation which is too complex for a user defined function, or any number of other processes may be best performed in a subroutine.

More than one subroutine can be used in a single program, in which case they are best placed one after the other in line number sequence before the DATA statements. It is a useful practice to assign distinctive line numbers to subroutines. For example, if the main program uses line numbers up to 199, use 200 and 300 as the first line numbers of two subroutines. When subroutines are included in a program, the program begins execution and continues until it encounters a GOSUB statement of the form:

```
GOSUB line number
```

where the line number following the word GOSUB is that of the first line of the subroutine. Control then transfers to that line of the subroutine. For example:

```
50 GOSUB 200
```

Control is transferred to line 200 in the user program. The first line in the subroutine can be a remark or any other valid BASIC statement.

Having reached the line containing a GOSUB statement, control transfers to the line indicated after GOSUB; the subroutine is processed until BASIC encounters a RETURN statement of the form:

```
RETURN
```

which causes control to return to the statement following the original GOSUB statement. A subroutine must always be exited via a RETURN statement.

Before transferring to the subroutine, BASIC internally records the next sequential statement to be processed after the GOSUB statement; the RETURN statement is a signal to transfer control to this statement. In this way, no matter how many subroutines there are or how many times they are called, BASIC always knows where to transfer control next. The following program demonstrates the use of GOSUB and RETURN.

```
1  REM THIS PROGRAM ILLUSTRATES GOSUB AND RETURN
10 DEF FNA(X) = ABS(INT(X))
20 INPUT A,B,C
30 GOSUB 100
40 A=FNA(A)
50 B=FNA(B)
60 C=FNA(C)
70 PRINT
80 GOSUB 100
90 END
100 REM THIS SUBROUTINE PRINTS OUT THE SOLUTIONS
110 REM OF THE EQUATION: AX^2 + BX + C = 0
120 PRINT "THE EQUATION IS "A "*X^2 + " B"*X + "C
130 D=B*B -4*A*C
140 IF D<>0 THEN 170
150 PRINT"ONLY ONE SOLUTION...X "; -B/(2*A)
160 RETURN
170 IF D<0 THEN 200
180 PRINT "TWO SOLUTIONS...X=";
185 PRINT (-B+SQR(D))/(2*A); " AND "; (-B-SQR(D))/(2*A)
190 RETURN
200 PRINT "IMAGINARY SOLUTIONS...X =(";
205 PRINT -B/(2*A) "," SQR(-D)/(2*A) ") AND (";
207 PRINT -B/(2*A) ","; -SQR(-D)/(2*A) ")"
210 RETURN
900 END
```

Subroutines can be nested; that is, one subroutine can call another subroutine. If the execution of a subroutine encounters a RETURN statement, it returns control to the statement following the GOSUB which called that subroutine. Therefore, a subroutine can call another subroutine, even itself. Subroutines can be entered at any point and can have more than one RETURN statement. It is possible to transfer to the beginning or any part of a subroutine; multiple entry points and RETURN's make a subroutine more versatile.

7.05 The ON GOTO and ON GOSUB Statements

The ON...GOTO statement provides another type of conditional branching. Its form is as follows:

ON expression GOTO line number list

After the value of the expression is truncated to an integer in the range 0-255, say I, the statement causes BASIC to branch to the line whose number is Ith in the list. If I=0 or is greater than the number of lines in the list, execution will continue at the next line after the ON...GOTO statement. If I is less than 0 or greater than 255, a CF error will result. For example, the following sequence of IF statements can be replaced by a single ON...GOTO statement. Thus;

```
100 IF X=1 THEN 1000
110 IF X=2 THEN 2000
120 IF X=3 THEN 3000
130 IF X=4 THEN 4000
140 IF X=6 THEN 6000
150 Y=10
```

can be replaced by:

```
100 ON X GOTO 1000, 2000, 3000, 4000, 150, 6000
150 Y=10
```

Note that there was no IF statement for X=5, so in the ON...GOTO statement the corresponding line number is 150, which is the next line.

Subroutines may be called conditionally by use of the ON...GOSUB statement. Its form is as follows:

ON expression GOSUB line number list

The execution is the same as ON...GOTO except that the line numbers are those of the first lines of subroutines. Execution continues at the next statement after the ON...GOSUB upon return from one of the subroutines.

Note that ON...GOTO and ON...GOSUB statements do not have to be the last executable statements on a line.

8. ARRAYS

8.01 Introduction to Arrays

Arrays or subscripted variables are most frequently used for storing lists of information in a program using a single name to refer to the list as a whole and using subscripts to refer to individual items. For example, consider the following list of 12 numbers corresponding to the number of days in each month in a non-leap year:

31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

The notion of subscripts follows naturally. For instance, the 5th item in the list corresponds to the number of days in May. Using an array (list) of size 12, named M, to refer to all the entries in the list as a whole, the fifth item of M can be simply denoted as M(5). Similarly, the number of days in February is denoted by M(2). If the number of days in the Ith month is desired, then M(I) contains that value.

In the following example, the data values are read into an array which is dimensioned to size 12 in line 10. (See Section 8.4.)

```
10 DIM M(12)
20 FOR I=1 TO 12: READ M(I): NEXT I
30 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
35 REM PRINT THE NUMBER OF THE MONTH AND DAYS IN EACH MONTH
36 REM ADD UP THE NUMBER OF DAYS IN THE MONTHS
40 D=0
50 FOR I=1 TO 12
60 PRINT I, M(I)
70 D=D+M(I)
80 NEXT I
90 PRINT "TOTAL DAYS =", D
```

The resulting output from this program is:

```
RUN
 1        31
 2        28
 3        31
 4        30
 5        31
 6        30
 7        31
 8        31
 9        30
 10       31
 11       30
 12       31
TOTAL DAYS =      365
```

If the above program were expanded past line 90 the values in M would be accessible at any point during the execution of the program unless they were changed by an assignment or input statement.

8.02 Subscripted Variables

The name of a subscripted variable is any acceptable BASIC variable name followed by one or more integer expressions in parentheses within the range Ø - 32767. Subscripted variable names follow the same naming conventions as simple variables with the first 2 characters being significant. For example, a list might be described as A(I), where I goes from Ø to 5 as shown below:

```
A(Ø),A(1),A(2),A(3),A(4),A(5)
```

This allows reference to each of the six elements in the list, and can be considered a one dimensional algebraic matrix as follows:

```
A(Ø)
A(1)
A(2)
A(3)
A(4)
A(5)
```

A two-dimensional matrix B (I,J) can be defined in a similar manner:

```
B(Ø,Ø),B(Ø,1),B(Ø,2),...,B(Ø,J),...B(I,J)
```

and graphically illustrated as follows:

```
B(Ø,Ø)  B(Ø,1)  ...  B(Ø,J)
B(1,Ø)  B(1,1)  ...  B(1,J)
   .       .          .
   .       .          .
   .       .          .
B(I,Ø)  B(I,1)  ...  B(I,J)
```

Higher dimensional arrays can also be formed. The upper limit is determined by the size of the input buffer giving a practical limit of 40.

Subscripts used with subscripted variables throughout a program can be explicitly stated or they can be any legal expression. If the value of the expression is non-integer, the value is truncated so that the subscript is an integer.

It is possible to use the same variable name as both a subscripted and unsubscripted variable. Both A and A(I) are valid variables and can be used in the same program. The variable A has no relationship to any element of the matrix A(I). Subscripted arrays of character strings may also be defined, and their variable names are distinct. A$(I) bears no relation to A(I) or A.

A dimension (DIM) statement is used with subscripted variables to define the maximum number of elements in a matrix.

If a subscripted variable is used without appearing in a DIM statement, it is assumed to be dimensioned to length 1Ø in each dimension (that is, having eleven elements in each dimension, Ø through 1Ø). However, all matrices should be correctly dimensioned in a program.

8.03 Subscripted String Variables

Any list or matrix variable name followed by the $ character denotes the string form of that variable. For example:

```
V$(n)          M2$(n)
C$(m,n)        G1$(m,n)
```

where m and n indicate the position of the matrix element within the whole.

The same name can be used as a numeric variable and as a string variable in the same program with no restriction. Simple variables and dimensioned variables can also have the same name. For example:

```
A              A(n)
A$             A$(m,n)
```

can all be used in the same program; however, A(n,m) could not be used, because it redefines the size of A(n).

String lists and matrices are defined with the DIM statement as are numerical lists and matrices.

8.04 The DIM Statement

The DIM statement is used to define the maximum number of elements in a matrix. The DIM statement is of the form:

```
DIM variable(n), variable(n,m), variable$(n), variable$(n,m)
```

where variables specified are indicated with their maximum subscript value(s). For example:

```
10 DIM X(5),Y(4,2), A(10,10)
12 DIM A4(100), A$(25)
```

Arrays can be dynamically dimensioned by using numeric expressions instead of integer constants to define the size of an array. Any number of matrices can be defined in a single DIM statement as long as they are separated by commas.

The first element of every matrix is automatically assumed to have a subscript of zero. Dimensioning A(6,10) sets up room for a matrix with 7 rows and 11 columns. This zero element is illustrated in the following program:

```
10 REM MATRIX CHECK PROGRAM
20 DIM A(6,4)
30 FOR I=0 TO 6
40 A(I,0) = I
50 FOR J=0 TO 4
60 A(0,J) =J
70 PRINT A(I,J);
80 NEXT J:PRINT:NEXT I
90 END
RUN
 0 1 2 3 4
 1 0 0 0 0
 2 0 0 0 0
 3 0 0 0 0
 4 0 0 0 0
 5 0 0 0 0
 6 0 0 0 0
```

Notice that a variable has a value of zero until it is assigned another value.

Whenever an array is dimensioned (m,n), the matrix is allocated with (m+1,n+1) elements. Memory space can be conserved by using the 0th element of the matrix. For example, DIM A(5,9) dimensions a 6 * 10 martrix which would then be referenced beginning with the A(0,0) element.

The size and number of matrices which can be defined depend upon the amount of storage space available.

A DIM statement can be placed anywhere in a multiple statement line and can appear anywhere in the program. A matrix can only be dimensioned once. DIM statements must appear prior to the first reference to an array. DIM statements are generally among the first statements of a program to allow them to be easily found if any alterations are later required.

All arrays specified in DIM statements are allocated space when the DIM statement is executed. All other arrays are declared at the first reference executed.

9. FURTHER SOPHISTICATION

9.01 Formatting the Printout

Often, the purpose of a program will require that results be printed out in a particular format, rather than simply in a list or line at the end of a program run. BASIC provides certain facilities for use in formatting the printout, so that the desired result can be achieved.

When a comma separates a text string from another PRINT list item, the item is printed at the beginning of the next available print zone. Semicolons separating text strings from other items are ignored. The screen is divided into 8 print zones of 8 characters each. A comma or semicolon appearing as the last item of a PRINT list always suppresses the carriage return/line feed operation. BASIC does an automatic carriage return/line feed if a string is printed past column 64. Examples of the use of comma include:

```
10 A=3
20 B=2
30 PRINT A,B,A+B,A*B,A-B,B-A
```

When the preceding lines are executed, the computer will print:

```
 3       2       5       6       1      -1
```

Notice that each character is eight spaces from the next character. Two commas together in a PRINT statement cause a print zone to be skipped, as in:

```
10 A=1
20 B=2
30 PRINT A,B,,A+B
RUN
 1       2               3

READY
```

If the last item in a PRINT statement is followed by a comma, no carriage return/linefeed is output, and the next value to be printed (by a later PRINT statement) appears in the next available print zone. For example:

```
10 A=1:B=2:C=3
20 PRINT A, :PRINT B: PRINT C
RUN
 1       2
 3
READY
```

If a tighter packing of printed values is desired, the semicolon can be used in place of the comma. A semicolon causes no spaces to be output other than the leading space automatically output with each non-negative number. A comma causes the cursor to move at least one space to the next

print zone or perform a carriage return/line feed if the string prints past column 64. The following example shows the effects of the semicolon and comma.

```
10 A=1:B=2:C=3
20 PRINT A;B;C;
30 PRINT A+1;B+1;C+1
40 PRINT A,B,C
RUN
 1 2 3 2 3 4
 1       2       3
READY
```

The following example demonstrates the use of the formatting characters , and ; with text strings:

```
120 PRINT "STUDENT"X; " GRADE ="G;" AVG. ="A;
130 PRINT " NO. IN CLASS ="N
```

Assuming that calculations had been done prior to these lines, the following would result:

```
STUDENT 119050 GRADE = 87 AVG. = 85.44 NO. IN CLASS = 26
```

9.01.01 The Tabulator Function; TAB(x)

The TAB function is used in a PRINT statement to write spaces to the specified column on the output device. The columns on the screen are numbered 1 to 64. The form of the command is:

```
PRINT TAB(x)
```

where (x) is the column number in the range 0 - 255. (If x exceeds 64, however, every other consecutive line is tabbed until the number of specified spaces are printed. If (x) is greater than 255 or negative, an error message is printed as follows:

```
CF ERROR
READY
```

If (x) is non-integer, only the integer portion of the number is used. If the column number (x) specified is less than or equal to the current column number, the TAB function has no effect.

9.01.02 The Space Function; SPC(x)

The SPC function can be used in much the same fashion as TAB in PRINT statements. This function prints the number of spaces indicated by (x) which must be in the range 0-255; otherwise a CF error results.

Note that if either a TAB(x) or SPC(x) is the last item in a print list the carrige return/line feed is suppressed.

9.02 Immediate Mode and Debugging

Immediate mode operation is especially useful for program debugging (error removal), and performing simple calculations in situations which do not occur with sufficient frequency or with sufficient complication to justify writing a program.

In order to facilitate debugging a program, END statements can be liberally placed throughout the program. Each END statement causes the program to halt, at which time the various data values can be examined and perhaps changed in immediate mode. The command:

```
GOTO xxxxx
```

is used to continue program execution (where xxxxx is the number of the next program line to be executed). GOSUB and IF commands can also be used. The values assigned to the variables when the RUN command is executed remain intact until a CLEAR statement or another RUN command is executed.

When using immediate mode, nearly all of the standard statements can be used to generate or print results.

If LINEFEED is used to halt program execution, the GOTO xxxx or CONT command can be used to continue execution. Since CTRL/J or LINEFEED does print the number of the line where execution stopped, it is easy to know where to resume the program. Note that if a BASIC program statement is entered or altered, it is not possible to continue execution.

9.02.01 Restrictions on Immediate Mode

The INPUT and DEF statements cannot be used in immediate mode and such use results in the following error message:

```
ID ERROR
READY
```

Certain other commands, while not illegal, make no logical sense when used in immediate mode. Commands in this category are DIM and DATA.

Although the standard mathematical functions are permissible, user functions are not defined until the program is executed, and therefore any references to user defined functions in immediate mode cause an error unless the program containing the definition was previously executed.

Thus, the following dialogue might result if a function were defined in a user program and then referenced in immediate mode.

```
10 DEF FNA(X) = X^2 + 2*X:REM SAVED STATEMENT
PRINT FNA(1):REM IMMEDIATE MODE

UF ERROR
READY
```

but if the sequence of statements were:

```
10 DEF FNA(X) = X^2+2*X:REM SAVED STATEMENT
RUN

READY

PRINT FNA(1)
 3

READY
```

the immediate mode statement would be executed.

9.03 Machine Level Interfaces with DISK BASIC

DISK BASIC has several features that allow the user access to the machine level input/output of the microprocessor. By using the WAIT and OUT statements and the INP function, various input/output operations can be performed. Other machine dependent features allow access to the memory and assembly language subprograms. (See Appendices E.1 and E.2 for Key Memory Locations and Port Assignments.)

9.03.01 The WAIT Statement

The status of memory ports can be monitored by the WAIT statement which has the following forms:

```
WAIT I,J
WAIT I,J,K
```

where I is the number of the port being monitored, and J and K are integer expressions. The port status is exclusive OR'ed with K if present and the result is AND'ed with J. Execution is suspended until a non-zero value results. In other words, J picks the bits of port I to be tested and execution resumes at the next statement after the WAIT. If K is omitted, it is assumed to be zero. I, J, and K must be in the range 0 to 255; otherwise, a CF error results.

9.03.02 The OUT Statement

The form of the OUT statement is as follows:

```
OUT I,J
```

where I and J are integer expressions in the range 0 to 255. OUT sends the 8 bit quantity (byte) signified by J to output port I.

9.03.03 The Input Function; INP(x)

The INP function is the counterpart of the OUT statement. Its form is as follows:

```
X = INP(I)
```

INP reads a byte (8 bit quantity) from port I where I is an integer expression in the range 0 to 255.

9.03.04 The Peek Function; PEEK(x)

The PEEK Function is called as follows:

```
J = PEEK(I)
```

where J is the integer value returned in the range 0-255 that is to be stored in the memory location specified by the integer expression I. The range of I is -32768 to 65535. If I is negative, then the address is 65536+I; and if I is positive, the address is I.

9.03.05 The POKE Statement

The form of the POKE statement is as follows:

```
POKE I,J
```

where J is an integer expression in the range 0 to 255 that is to be stored in the memory location specified by the integer expression I. The range of I is -32768 to 65535. If I is negative, then the address is 65536+I; and if I is positive, the address is I.

9.03.06 The User Call Function; CALL(x)

The CALL function is used for interfacing with 8080 machine language subroutines. The function can be used in the same manner as the other mathematical functions. The form is as follows:

```
Y = CALL(x)
```

where the assignment x must be in the range -32768 to 65535. The value Y returned is in the range -32768 to 32767.

The CALL function converts the argument into a 2-byte integer and stores the result in the 8080's D and E registers (high byte in D, low byte in E). The BASIC interpreter then executes an 8080 CALL instruction to location 33282 (8202 HEX), which, unless modifed by the user, contains a jump to the CF ERROR message routine. The user must modify the locations 33282 through 33284 so that they contain a JMP to the desired machine language routine. Upon return, the 2 byte integer in the D,E registers is converted back into floating point format. The stack level must be preserved at the same point at which the user entered the CALL, and the H and L registers must be preserved. All other 8080 registers can be modified.

For example, consider the following assembly language subroutine which negates the contents of the D and E registers.

```
        ORG     08202H  ;33282
        JMP     NEGATE

        ORG     09FF0H  ;40944
NEGATE: MOV     A,D     ;COMPLEMENT
        CMA             ;HIGH
        MOV     D,A     ;BYTE
        MOV     A,E     ;COMPLEMENT
        CMA             ;LOW
        MOV     E,A     ;BYTE
        INX     D       ;INCREMENT AND  FORM 2'S COMPLEMENT
        RET             ;RETURN - HL UNCHANGED
```

This subroutine could be assembled using the Intecolor Assembler or "hand" assembled and entered using the POKE statement in BASIC.

To enter this subroutine in BASIC, the user must first hit CPU RESET then re-enter BASIC by using the ESCAPE W sequence. The number 8176 must be entered in response to the MAXIMUM RAM AVAILABLE prompt. This leaves 16 bytes free for the machine language subroutine. The following program loads the machine language subroutine and demonstrates the CALL function.

```
5  REM CHANGE JUMP ADDRESS AT 8203-4 HEX, 8202H CONTAINS JUMP
10 POKE 33283, 240 : POKE 33284, 159
15 REM PROGRAM BYTES AT 9FF0 HEX
20 DATA 122, 47, 87, 123, 47, 95, 19, 201
30 FOR AD = 40944 TO 40951
40 READ VL: POKE AD, VL
50 NEXT AD
100 INPUT "ENTER X ";X : Y=CALL(X)
110 PRINT "-X = ";Y : GOTO 100
```

9.04 String Space Allocation

Understanding how the string space is used is important in deciding how much string space is necessary for the execution of a program. First, all strings entered in immediate mode or by the INPUT statement (see Section 6.01) are allocated in the string space because the input line buffer can be modifed by subsequent inputs.

String functions and the string concatenation operator "+" always return their results in the string space. Assigning a string a constant value in a program through a READ or assignment statement does not use any string space since the string value is part of the program itself. In general, copying is done when a string value is in the input line buffer, or it is in the string space and there is an active reference to it by a string variable. Thus, A$ = B$ will cause copying if B$ has its string data in the string space. The assignment A$ = STR$(105) (see Section 7.03 for STR$) will use four bytes of string space to store the new four character string, " 105", created by the STR$ function, but the assignment itself .

does not cause copying since the only reference to the new string was created as a temporary reference by the formula evaluator. The temporary references disappear when the assignment is done. The copying is done in this manner because the string garbage collection does not allow two references to the same area in the string space.

10. DISK FEATURES

10.01 Loading and Saving Programs

Programs and data can be loaded and saved by the Intecolor 3650 Series unit so that they can be stored and used, edited, or updated in the future. The general forms of the LOAD and SAVE statements are:

```
LOAD  string expression
SAVE  string expression
```

where the string can be a string variable such as A$ or a quoted literal string such as "NAME". There are three FILE types that can be loaded and saved. They are BASIC source (BAS), numeric ARRAYS (ARY), and memory DATA (DAT). If no file type is specified, then the default type is BAS. The BAS file type can be in the form as shown below. Each of the following examples will save the same BASIC source.

```
SAVE "TEST" :REM SAVES BASIC SOURCE WITH NAME TEST ON DISK
SAVE "TEST.BAS"
SAVE "TEST.BAS;1"
SAVE A$: REM WHERE A$ IS A STRING VARIABLE
SAVE "MD1:" + A$: REM WHERE "MD1:" SPECIFIES OPTIONAL DISK
```

NOTE: A diskette must be initialized before it can be used to save a program. See Section 2.02 in Chapter 2 or the File Control System instructions in Chapter 12.

Each of the following examples will cause a BASIC source program to be loaded.

```
LOAD "TEST:REM LOADS A BASIC_SOURCE PROGRAM BY NAME OF TEST
LOAD "TEST.BAS"
LOAD "TEST.BAS;1"
LOAD "MD1:" + A$: REM WHERE "MD1:" SPECIFIES THE SECOND DISK
LOAD A$:REM WHERE A$ IS A STRING VARIABLE
```

The ARY file type can be in the same form as BAS except that ARY must be in the string after the file name. Also the file name must be a dimensioned or previously used array by the same first two letters of the file name. If a one letter variable name is used, then the file name must be that letter only.

```
10 DIM ST (100,10),T(3),TT(11,15,38)
20 SAVE "STEST.ARY"
30 SAVE "T.ARY;1"
40 END
```

The above program will save the numbered arrays ST and T. The following program will cause a (100,10) array to be loaded even though it was originally set at 1200, since 1200 > 101 * 11.

```
10 DIM ST (1200)
20 LOAD "STEST.ARY": REM DIM ST (100,10)
30 END
```

The DAT file type can be in the same form as ARY. It will look at the two-byte integer stored in locations 32940 and 32941 (32940 low byte and 32941 high byte) as a pointer to memory. It adds 1 to this pointer and takes the next two bytes in memory as the number of bytes to be loaded into memory or saved on disk. The locations 32940 and 32941 specify the end of BASIC memory space, so all memory above that location can be used to save data via BASIC using the POKE command. Also note that only one DAT file may be read in at any one time without changing the pointers at 32940 and 32941.

Note that it is recommended that programs use the random file capability of DISK BASIC instead of loading and saving DAT files.

10.01.01 Program Chaining

A series of different programs can be executed as a single program by using a technique commonly known as program chaining. In DISK BASIC, two types of program chaining are possible. The first and easiest method uses the LOAD statement in combination with the RUN command as follows:

```
LOAD "PROGRM": RUN
```

Executing this statement in either a program or immediate mode causes the specified BASIC program to be loaded and executed. The RUN command clears all the variables from the previous program. A line number can optionally be specified on the RUN command.

The second method uses the LOAD statement in combination with the GOTO statement as follows:

```
LOAD "PROGRM": GOTO line number
```

Executing this statement in a program causes the specified program to be loaded and executed starting at the specified line number in the GOTO command. This method does not clear the variables from the previous program; however, two restrictions must be satisfied to ensure proper exection of the program. First, the program with the largest source in the chain must be loaded and executed first. Second, string variables whose data values were part of the program source will contain incorrect references when a subsequent program is loaded because the program source will not be the same as the previous program. If these restrictions are satisfied, then the series of programs should execute properly. Clearly, this second method of program chaining is the least desirable because of the possible difficulties. See Section 11.04 for a description of how strings are allocated before using this method.

10.01.02 MENU Programs

With the 3650 Series Intecolor it is possible to create a program that is loaded and executed by pressing a single key. The AUTO key automatically loads and executes a BASIC program called MENU.BAS from the default disk drive (internal drive of a 3651 or 3652, left drive of a 3653 or 3654, unless the default device has been changed). Initialization in BASIC is assumed. (Chapter 12 has more detail on device and file specifications.)

The MENU program can be used to run and control a large application system composed of several programs such as a payroll system. In this case the MENU program asks which function is to be performed next and directs the execution to the proper program or section of BASIC code. Similarly, the MENU program can control a number of unrelated applications by displaying a "menu" of applications accessible on the diskette. This technique is used on many of the diskette albums. Thus, by depressing the AUTO key, the MENU program is loaded and executed displaying a "menu" of programs. The user simply selects a program by number or name, and then the MENU program chains to the desired program. When the selected program is finished it can chain back to the MENU program.

For example, consider the following three programs:

MENU.BAS

```
10 PRINT "MENU PROGRAM"
20 PRINT
30 PRINT"1 - PRINT TABLE OF POWERS"
40 PRINT"2 - PRINT TABLE OF SINE FUNCTIONS"
50 PRINT
60 INPUT "ENTER NUMBER OF DESIRED PROGRAM"; N
70 N = INT(N)
80 IF N<1 OR N>2 THEN 60
90 ON N GOTO 100,200
100 LOAD "POWERS":RUN : REM EXECUTE POWERS
200 LOAD "SINE": RUN : REM EXECUTE TRIG
999 END
```

POWERS.BAS

```
10 PRINT "N","N^2","N^3"
30 FOR N = 1 TO 10
40 PRINT N,N^2,N^3
50 NEXT N
60 PRINT
100 LOAD "MENU":RUN:REM RETURN TO MENU
```

SINE.BAS

```
10 PI = 3.14159265: REM BASIC ROUNDS TO APPROX. 7 PLACES
20 PRINT "DEG","SINE"
30 PRINT
40 FOR DEG = 0 TO 360 STEP 15
50 RAD = DEG * PI/180
60 PRINT DEG,SIN(RAD)
70 NEXT DEG
80 PRINT
100 LOAD "MENU":RUN :REM RETURN TO MENU
```

To try this example the user must first enter and save each of the three programs, being careful to remember to reinitialize BASIC before entering a new program after saving the old one. After saving all three programs the AUTO key must be struck. This causes MENU to be loaded and executed. MENU then asks for either 1 or 2 and executes the selected program. Note that these programs return back to the MENU program after they have performed their specified function. This makes the MENU program an effective tool for controlling and demonstrating a system or diskette of programs.

It may be well to note, however, that if programs other than .BAS have been run (e.g., .PRG, .LDA), it is possible that BASIC has been altered in memory so that other menu stringed programs cannot be loaded until BASIC has been reinitialized and the menu loaded.

10.02 Using the File Control System through BASIC

The PRINT STRING command preceded by PLOT 27 and PLOT 4 or PLOT 68 (ESC,D for FCS) will enable the user to exercise all of the FCS disk commands through BASIC. Thus every command available to the File Control System is also available to BASIC, by letting the string become the FCS command. The following examples show how to retrieve a disk directory through BASIC. (For a description of the PLOT statement, see Section 11.01)

```
10 PLOT 27,4
20 PRINT "DIR"
30 PLOT 27,27
40 END
```

or

```
10 PLOT 27,4: PRINT"DIR": PLOT 27,27
```

or

```
10 PLOT 27:PLOT 68:PRINT A$:REM WHERE A$ IS
20 REM A STRING VARIABLE EQUAL TO DIR.
```

If the directory of the disk were as follows:

```
TEST.ARY;01
TEST.ARY;02
```

then the BASIC program below would delete version 1 of the TEST.ARY file, rename version 2 to version 1, update the array, and save it as version 2 so it can be used again.

```
5 DIM TEST(1000)
10 LOAD "TEST.ARY;2"
20 PLOT 27: PLOT 4: REM SELECT FCS MODE
30 PRINT "DELETE TEST.ARY;1"
50 PRINT "RENAME TEST.ARY;2 TO TEST.ARY1"
60 PLOT 27: PLOT 27: REM SELECT VISIBLE CURSOR MODE
80 REM UPDATE TEST ARRAY
90 SAVE "TEST.ARY"
95 END
```

All string functions that are available to BASIC can be used in the PRINT statement containing the FCS command.

To escape from the File Control System and return to one of the other CRT modes, an escape sequence must be given; such as ESC,ESC for visible CRT cursor mode. The FCS responds only to printing ASCII characters and the following control codes:

11	ERASE LINE
13	CARRIAGE RETURN
26	CURSOR LEFT
27	ESCAPE

All other control codes will cause an FCS error if they appear in a string. A complete description of the FCS commands appears in Chapter 12 and Appendix C.1.

10.02.01 Loading and Saving Displays in BASIC

The FCS interface with BASIC makes it very easy to load and save screen displays. However, in order to do this there must be in the default disk drive a diskette with the SVSCR.COM file. (See section 12.04.) The Intecolor should be in page mode (see Section 11.04); otherwise, if the screen has scrolled at all then the saved display will be wrapped around. After the display has been generated via a BASIC program, the user simply includes a sequence of statements similar to that shown below.

```
900 PLOT 27,4 : REM ENTER FCS
910 PRINT "SVSCR SCREEN 7000 1000"
920 PLOT 27,27 : REM RETURN FROM FCS
```

Line 910 saves a copy of screen refresh memory which is located from 7000 HEX to 7FFF HEX in a file called SCREEN.PIC. In a system with 5.25" disk drives, the display can now be loaded at any time by executing the following sequence of statements.

```
1000 PLOT 12 : REM UNROLL SCREEN MEMORY BY ERASE PAGE
1010 PLOT 27,4 : REM ENTER FCS
1020 PRINT "LOAD SCREEN.PIC 7000" : REM LOAD DISPLAY
1030 PLOT 27,27 : REM RETURN FROM FCS
```

Systems with other types of disk drives must use a .COM file command. For line 1020 above, substitute:

```
1020 PRINT "LDSCR SCREEN.PIC 7000"
```

These sequences can be tailored to fit any needs by simply changing the line numbers and name of the file containing the display.

Displays that are generated in CRT mode can also be saved using BASIC. First, initialize BASIC with the (ESC) W RETURN sequence. Before creating a display, the user should enter the following one line BASIC program.

```
10 PLOT 27,4 : PRINT "SVSCR SCREEN 7000 1000" : END
```

Then save this one-line program as a menu by use of the SAVE command:

```
SAVE "MENU"
```

Now enter the CRT Mode via CPU RESET. Then set Page Mode with an (ESC) X sequence and clear the screen with ERASE PAGE. Then create the display. After the display is finished, simply depress the AUTO key. The one-line program will load and run, saving the display.

10.03 Introduction to Random Files

EXTENDED DISK BASIC has three statements which implement a powerful random access file capability. The FILE statement performs various functions including creating, opening and closing random files. The GET and PUT statements read, write, and update records in a random file.

Random files are organized into physical blocks containing a fixed number of fixed length records. If a physical block is not a multiple of 128, then the excess length up to the next multiple of 128 is not used. The blocking factor and record size of a file can be changed to allow different types of access. For example, a 100 record file of 80 byte records with a blocking factor of 3 will use only the first 240 bytes of 256 available in 2 disk sectors. The last 16 bytes are unused. Logical records do not cross physical block boundaries. Thus, for the 100 record file 2*34=68 sectors are needed. In this case a 102 record file could have been allocated in the same amount of disk space.

There can be up to 127 random files open simultaneously subject to memory limitations. Memory space for files is allocated dynamically from the user's workspace. Each file can contain from 1 to 32767 records and the record size range is 1 to 32767 bytes. The record size must be small enough to fit into the user's workspace, giving a practical maximum of 30000 bytes. The default filename type for random files is .RND.

10.04 The FILE Statement

The basic form of the FILE statement is:

```
FILE "string expression", extra information
```

The FILE statement is a versatile statement that has the ability to perform a number of functions. The first character of the string expression determines what the FILE statement will do. The following sections describe the FILE statement's uses and functions.

10.04.01 Random File Creation

The Random File Creation statement is of the form:

```
FILE "N", filename, records, record size, blocking factor
```

where 'filename' is a string expression containing a valid FCS filename; 'records' is the number of logical records (1-32767); 'record size' is the size in bytes of logical records (1-32767); and 'blocking factor' is the number of logical records per physical block (1-255).

The specified file must not exist. If no version number is specified, then FCS will choose the next larger version number. The user is responsible for choosing proper values of the parameters. Any of the file specifications can be overridden when the file is opened with the FILE "R" statement. For example:

```
FILE "N", "CHECKS", 200, 32, 8
```

creates a file containing 200 32-byte records with 8 records per block.

10.04.02 Random File Open

The form of the Random File Open statement is:

```
FILE "R",file,name,buffers<;records,rec size,blocking factor>
```

where 'file' is the logical number of the file (1-127), 'name' is a string expression containing a valid FCS filename, and 'buffers' is the number of buffers in memory (1-255).

The items between the angle brackets are optional and redefine the file size. The elements are: 'records', which is the number of logical records (1-32767); 'rec size', which is the size in bytes of logical records (1-32767); and 'blocking factor', which is the number of logical records per physical block (1-255).

The specified file must already exist. It is possible to open any type of file, but they are best created with the FILE "N" statement. Files not created in BASIC can be accessed by overriding the number of records, the record size, and the blocking factor, but the directory will not contain valid information about the number of records, record size, or blocking factor. For example:

```
FILE "R",1,"CHECKS", 2
```

opens the file "CHECKS.RND" and allocates enough buffer space for 2 physical blocks or 16 records.

10.04.03 Random File Close

The Random File Close statement is of the form:

```
FILE "C", file 1 <,...,file N>
```

where 'file' is the number of the file to be closed. The items between the angle brackets are optional, and merely describe the format for closing more than one file at a time.

Each file that has been opened must be closed to ensure that the buffers in memory are written to the disk if they have been modified. Closing a file frees up its buffer space in memory. For example:

```
FILE "C", 1
```

closes file 1.

10.04.04 Dump File Buffers

The form of the Dump File Buffers statement is:

```
FILE "D", file 1 <,...,file N>
```

where 'file' is the number of the file (1-127); and the optional items between the angle brackets are other files that can be included in the same statement.

This statement writes any modified buffers to the disk for the specified files. It can be used to ensure that modifications to a file are recorded immediately. It is similar to FILE "C" except that the buffer space is not freed up and the file remains open. For example:

```
FILE "D",4,6
```

writes any modified buffers back to the disk for files 4 and 6.

10.04.05 File Attributes

The form of the File Attributes statement is:

```
FILE "A",file,cur record <,records,rec size,blocking factor>
```

where 'file' is the number of the file (1-127); and 'cur record' is the variable that is assigned the most recently accessed record number. The items between the angle brackets are optional and include 'records', which is the variable that is assigned the number of records in the file; 'rec size', which is the variable that is assigned the record size in bytes; and 'blocking factor', which is the number of logical records per physical block (1-255).

This statement is used when the file size and other attributes of a file are unknown. For example, the attributes of file 1 may be determined as follows:

```
FILE "A", 1, CR, NR, RS, BF
```

10.04.06 File Error Trapping

The form of the File Error Trapping statement is:

```
FILE "T" <,line number>
```

where the optional line number is a line number in the range 0 to 65529.If the file "T" statement is executed with the line number specified, then when a disk error occurs it will be trapped and execution will continue at the specified line number. All information about nested GOSUB's and FOR-NEXT loops will be lost. In most cases this will not be a problem. In the other cases, assuming good programming practices, the disk error will probably be a hardware failure which requires some type of special recovery procedure. If the line number is not specified, then the error trapping facility will be disabled. For example:

```
FILE "T",32000
```

causes the program to go to line 32000 whenever a disk error occurs.

10.04.07 File Error Determination

The form of the File Error Determination statement is:

```
FILE "E", file, error, line number
```

where 'file' is the file number at the time of the error (this number may be incorrect for bad file name errors and errors within the FILE "N" statement); 'error' is the disk error number (for explanations see Appendix B.6); and 'line number' is the line number in which the error occurred.

This statement lets the user determine what type of disk error occurred. It is used in conjunction with the FILE "T" statement. For example:

```
FILE "E", FL, ER, LN
```

returns the file, error, and line number of the current random file error.

10.05 The GET Statement

The GET statement is of the form:

```
GET file <,record <,first>> ; variable list
```

where 'file' is the logical file number (1-127); and the 'variable list' contains one or more of the following entries:

numeric variable - reads 4 bytes into the numeric variable;
string variable [byte count] - reads the specified number of bytes into the string variable. The byte count range is 1 to 255.

The items between the angle brackets are optional and include 'record', which is the record number to be read (if 0 or omitted, then the record number is 1 greater than that used for the last access to the file); and 'first', which is the first byte of the record to be read (1-record size). If no value is given for 'first', then first defaults to 1.

The GET statement allows a file to be randomly accessed. By using the first field, different parts of the record can be immediately accessed. For example:

```
GET 1,R;ACCOUNT,AMOUNT,DATE,PAYEE$[20]
```

will read ACCOUNT, AMOUNT, and DATE as numeric entries, and PAYEE as a 20 byte string.

10.06 The PUT Statement

The PUT statement is of the form:

```
PUT file <,record <,first>> ; expression list
```

where 'file' is the logical file number (1-127); and the 'expression list' contains 1 or more of the following entries:

numeric expression - writes 4 bytes containing the value of the expression;

string expression [byte count] - writes the specified number of bytes. The value of the string expression is truncated or blank filled on the right. The byte count range is 1-255.

Items between the angle brackets are optional and include: 'record', which is the record number to be written or updated (if 0 or omitted, then the record number is 1 greater than that used for the last access to the file); and 'first', which is the first byte of record to be written (1-record size). If no value is given, then first defaults to 1.

The PUT statement allows random records to be written or updated. For example:

```
PUT,1,R,13; "MORTGAGE COMPANY"[20]
```

updates 20 bytes of record R starting at the 13th byte.

10.07 Improving File Access

The random files in DISK BASIC are oriented towards fast random reads and updates. Sequential file input and output can easily be simulated; however, there is a time penalty for sequential output because the PUT statement updates information on a record. The file accessing time in a program can often be greatly reduced if the program takes advantage of the flexibility offered.

The file accessing scheme in DISK BASIC is different from the random accessing scheme commonly used in most microcomputers. When a record is accessed that is not present in one of the buffers in memory, the physical block containing the logical record is read into memory in an unused buffer or, if all buffers are in use, the least recently used (LRU) buffer. If the least recently used buffer has been modified, it is rewritten to disk before the next block is read into the buffer. This type of a buffer management scheme is very similar to the LRU virtual memory paging schemes used on large computers.

The first method of improving file access is by increasing the number of file buffers allocated in the FILE "R" statement. Changing this number from 1 to a larger number does not alter the results of execution; it only alters the number of times the disk has to be physically accessed. The difference in time can be quite substantial. However, for sequential access or random access which uniformly accesses all parts of a large file

there is little advantage to be gained by increasing the number of buffers beyond 1. However, the trade-off between use of more memory for increasing the file buffers and decreasing the number of disk accesses can best be determined by the user.

The second method of improving file access is by varying the record size and blocking factor of a file. Ideally, the record size should be a power of 2. By choosing an appropriate blocking factor the block size will be a multiple of 128. For example, a 32 byte record can be blocked 4, 8, or 12, giving block sizes of 128, 256, or 384 bytes, respectively. For sequential access a blocking factor of 1 allocates 1 record to a physical block. Thus, to read records sequentially, 1 physical access and disk read is necessary for each record. With a blocking factor of 8, physical disk access is only necesary for every 8 records read, which is 1/8 as many disk accesses as necessitated by a blocking factor of 1.

If the record sizes are not a power of two, the blocking factor should be chosen carefully. For example, with 80 byte records a blocking factor of 1 will waste 48 bytes of disk space for each record because the 80 byte record is contained in a 128 byte disk sector. By using a blocking factor of 3, only 16 bytes (256-3*80) will be wasted for every 2 128 byte sectors. Again, with a blocking factor of 8, 640 bytes are used with no wasted space because 5 disk sectors hold exactly 640 bytes. Whether or not to choose 1, 3, or 8 should be determined by the type of application for which the file is used. If the program is large and uses most of the workspace, either 1 or 3 would be best. If the program is small, allocating 678 (34+4+640) bytes may be quite acceptable and improve the speed of the program. Choosing the best values for the number of buffers, record size, and blocking factor is often difficult. The user is following a reasonable guideline if he allocates 1 buffer for sequential files with a larger blocking factor and more buffers with smaller blocking factors for random files. For often used applications a little experimentation and fine tuning of the parameters can improve the disk access time.

10.08 Storage Requirements

When random files are used, they are allocated from the user's free workspace. The storage requirements in bytes are as follows:

error trapping - 10 bytes

open files -

4+30+BUF*(4+128*INT((RECSIZ*BLKFAC+127)/128)) bytes

where

BUF = the number of allocated physical block buffers,
RECSIZ = the number of bytes per record,
BLKFAC = the number of records per block.

Thus, opening a file with 80 byte records and a blocking factor of 3 and 1 buffer requires 34 + 1 * (4+256) = 294 bytes. With 4 buffers the requirement is 34 + 4 * (4+256) = 1074 bytes.

11. COLOR, GRAPHICS AND OTHER FEATURES

11.Ø1 The PLOT Statement

The PLOT Statement is used to output the 8 bit value of an expression to the screen. The form of the PLOT statement is as follows:

PLOT expression

or

PLOT expression,expression,...,expression

The expressions in the expression list must evaluate to a quantity in the range Ø to 255. Other values will cause a CF error.

For example, the following statement will cause the letters ABCDEF to be displayed on the screen.

PLOT 65,66,67,68,69,7Ø

The PLOT statement is usually used to send control codes, escape codes, and other graphics information to the screen. For further examples, see the following sections in this chapter, and for information about CRT commands and ASCII codes, see Appendices D and F.

11.Ø2 Color

The color displays that can be achieved on Intecolor are an important feature of the machine. The color controls are easy to operate and add a new dimension to traditional programming.

Both the foreground and background can be set to a desired color. The foreground can be made to blink, and in addition, characters may be either single or double height.

Color, blink and character size can each be set in one of two ways. The first method involves the use of the color and special keys. To set the background color, the BG ON key is pressed. Then the actual color is set by simultaneously striking the CONTROL key and the letter key corresponding to the desired color. They are as follows:

BLACK:	P	BLUE:	T
RED:	Q	MAGENTA:	U
GREEN:	R	CYAN:	V
YELLOW:	S	WHITE:	W

On the deluxe and extended keyboards, the color keys are in a separate pad and are simply struck to select color.

The foreground can be set by depressing the FG ON key and selecting a color as for the background.

The BLINK ON key sets the blink in motion and the BL/A7 OFF key turns it off. The double-height characters can be set by the A7ON key and small characters are reset by the BL/A7 OFF key. Because this key controls both blink and character height, if the user wishes to turn the blink off while using the larger characters, and continue typing in large characters, the BL/A7 OFF key and the A7 ON key must be struck in immediate succession.

While these codes can be used in the CRT mode to test color combinatons and display appearances, etc., the characters will only be accepted in BASIC if they are contained in quoted strings or REMARK statements. If not so contained, they will cause a syntax error (SN).

Color can be selected without being contained in a quoted string by the second method of setting color, blink and character height. This is done through the use of the PLOT statement, as shown below:

```
PLOT 29        (sets foreground color)
PLOT 30        (sets background color)
PLOT 31        (sets blink on)
PLOT 14        (sets large characters)
PLOT 15        (sets blink and large characters off)
```

The individual colors are selected by PLOT statements using the internal code of each color key, as shown below:

```
PLOT 16  (black)          PLOT 20  (blue)
PLOT 17  (red)            PLOT 21  (magenta)
PLOT 18  (green)          PLOT 22  (cyan)
PLOT 19  (yellow)         PLOT 23  (white)
```

Because blink off and standard character height are controlled by the same code, retaining double character height while turning off the blink will require PLOT 15 and PLOT 14 statements in immediate sequence. The PLOT commands can be used in a BASIC program to set the color of the screen output.

The PLOT character set, BLINK, BACKGROUND COLOR, and FOREGROUND COLOR can also be selected by means of the CCI Control Code or PLOT 6 statement. The general form in BASIC is as shown:

```
PLOT 6,number
```

where number must be an integer between 0 and 255 representing the visible CCI status. This number is represented in binary up to eight bits long and arranged in a table as shown below. (Also shown in Appendix D.2)

128 A7	64 A6	32 A5	16 A4	8 A3	4 A2	2 A1	1 A0
PLOT	BLINK	BACKGROUND			FOREGROUND		
		BLUE	GREEN	RED	BLUE	GREEN	RED

The foreground and background colors are formed, as in a color television, by combinations of the blue, red, and green color guns. When the binary number is placed in the eight bit location, a 1 in any position turns that bit on. The formula for determining the desired number in decimal is:

```
PLOT*128 + BLINK*64 + BACKGROUND*8 + FOREGROUND
```

The program below illustrates the various results that can be achieved with the PLOT 6 command.

```
10 PLOT 6,6: REM SET CYAN FOREGROUND AND BLACK BACKGROUND
20 PRINT "PLOT(0-1),BLINK(0-1),BCKGRD(0-7),FORGRD(0-7): ";
25 INPUT "";PL,BL,BG,FG
30 PLOT 6,PL*128+BL*64+BG*8+FG
40 REM LINE 30 SETS THE COLOR INFORMATION YOU SELECTED
50 PRINT "THIS IS WHAT YOU SELECTED";: PLOT6,6: PRINT
60 REM RESET COLOR BEFORE LINEFEED
70 GOTO 20
```

11.03 Special Characters

The 3650 Series Intecolor has 64 special characters which are actually two groups of 32 special characters. A group is selected depending upon the condition of the Flag Bit. If the Flag Bit is off, then the ASCII codes from 96 to 127 are not changed when they are placed in the CRT refresh memory. If the Flag Bit is on, then these codes have 96 subtracted from them before they are replaced in the CRT refresh memory. Therefore, they are mapped into 0 to 31 within the refresh memory.

The characters in the range 96 to 127 are generated by changing the shift of the alphabetic characters @, A, ..., Z, [, \,], ^, and _. If the CAPS LOCK key is down, then the SHIFT key will also have to be depressed to generate these characters. If the CAPS LOCK key is up, then these special characters are generated whenever an alphabetic key is struck. (Remember, CAPS LOCK affects alpha keys only, not [, etc.)

The condition of the Flag Bit is changed by depressing either the FG ON/FLAG OFF key or the BG ON/FLAG ON key. Thus, if the FG ON/FLAG OFF key is struck, the characters in columns 6 and 7 in the Intecolor Character Set (shown in Appendix F) are displayed whenever ASCII codes in the range 96 to 127 are received. If the BG ON/FLAG ON key is struck, the characters in columns 0 and 1 are selected.

In BASIC the two sets of special characters can be selected as follows:

PLOT CODE	KEY	CHARACTER SET
PLOT 29	FG ON/FLAG OFF	COLUMNS 6 AND 7
PLOT 30	BG ON/FLAG ON	COLUMNS 0 AND 1

Intecolor has an alternate set of 256 characters. These characters are used for the graphics plot modes where each character position is composed of eight plot blocks - four high by two wide. These plot blocks can also be accessed through the character plot via color pad mode entered by the ESC B sequence. This mode uses the eight color keys to intensify each of

the eight plot blocks within a character. The one to one correspondence between the 4 x 2 color select pad on the extended and deluxe keyboards and the 4 x 2 character plot blocks is shown below.

CHARACTER PLOT BLOCK ARRAY WITH CORRESPONDING COLOR CODES

BLACK	BLUE
RED	MAGENTA
GREEN	CYAN
YELLOW	WHITE

This mode is designed especially for use by the keyboard to simplify the drawing of graphs or the correcting of graphs. Once this mode is entered a block at the top right hand corner of the present cursor position can be intensified by depressing the BLUE key at the top right hand corner of the color selection pad or CONTROL T (2Ø).

Once a plot block is intensified, any other plot block in the same character positon can also be intensified since the cursor does not automatically advance. If a color key corresponding to an intensified plot block is pressed, the plot block is turned off. This allows plot blocks to be erased. After all the desired plot blocks have been intensified or extinguished, the cursor can be moved using the cursor control keys without leaving this mode. In fact, all of the control codes are effective while in this mode except for the color select control codes, and any of the ASCII text characters (32 to 127) can be entered and displayed. Any code that requires a two key or more sequence (such as CURSOR X-Y, CCI, and ESC) terminates this mode. It should be noted that the ASCII text characters when entered and displayed advance the cursor. Therefore, when a character position has been used to display plot blocks, a cursor command must be given to advance the cursor to the next character positon.

11.Ø4 Cursor Controls

Intecolor has two cursor modes available to the user. The most commonly used mode is the visible cursor mode where the blinking cursor on the screen shows the current visible cursor position. In this mode all the cursor control features of Intecolor are available. A second cursor mode, called the blind cursor mode, allows the use of a second invisible cursor with only XY cursor position allowed. The two modes are described in the following sections.

11.04.01 Visible Cursor Mode

In the visible cursor mode the following PLOT statements move the visible cursor on the screen:

PLOT 10	CURSOR DOWN / LINEFEED (moves the cursor 1 space down)
PLOT 25	CURSOR RIGHT (moves cursor one space to the right)
PLOT 28	CURSOR UP (moves cursor one space up)
PLOT 26	CURSOR LEFT (moves cursor one space to the left)
PLOT 8	HOME (moves cursor to topmost left of screen)
PLOT 9	TAB (moves cursor to start of next print zone)

The X,Y position of the visible cursor can be changed using the CURSOR X,Y control code sequence or the following PLOT statement:

```
PLOT 3,X,Y
```

where X and Y are the desired X,Y coordinates. The range for X is 0 to 64 and the range for Y is 0 to 31. Where X=0 and Y=0 is the HOME position at the upper left hand corner of the screen. The X coordinate determines the column position and the Y coordinate determines the line on the screen.

If the cursor is positioned at X=64, then the blinking visible cursor will disappear. But if a character is typed, it will be positioned at the beginning of the line specified by Y+1, and the cursor then reappears in character position (X=1). Any cursor movement command forces the cursor to reappear at the proper position relative to character position 0, line Y+1.

If the X value is greater than 80, then the blind cursor addressing mode is entered.

Page mode, which is entered from the keyboard via ESC X, writes characters left to right, and does not scroll the screen. From BASIC it is entered with a PLOT 27,24 statement.

Scroll mode, which is entered via ESC K, writes left to right and scrolls the screen for a continuous readout. It is entered in BASIC by PLOT 27,11.

Vertical mode, which is entered via ESC J writes top to bottom in one column only. It does not scroll the display. This mode can be reached through BASIC by the PLOT 27,10 statement.

ERASE PAGE code replaces the contents of the entire screen with spaces that have the same color and composite status as the present visible CCI status. Both the visible and blind cursors are positioned at HOME. In BASIC a PLOT 12 erases the screen.

The ERASE LINE code does a carriage return and replaces the line containing the visible cursor with spaces having the same color and composite status as the present visible CCI status. The cursor is sent to the beginning of the line. The current line is erased through BASIC by PLOT 11. The following program illustrates the use of some cursor controls.

```
10 DEF FNR(X) = INT (X*RND(1))
20 FOR I = 0 TO 3: READ D(I): NEXT I
30 DATA 10,25,28,26: REM CURSOR CONTROL VALUES
40 PLOT 6,0,12,27,24: REM ERASE PAGE AND SET PAGE MODE
50 PLOT 3, FNR(64), FNR(32): REM SELECT RANDOM STARTING POINT
60 FOR I = 1 TO 1000
70 PLOT 6, (FNR(7)+1)*8: REM SET VISIBLE BACKGROUND COLOR
80 PLOT 20,26: REM OUTPUT SPACE, THEN BACKSPACE
90 PLOT D(FNR(4)): REM OUTPUT A RANDOM DIRECTION
100 NEXT I
110 PLOT 6,2,8: REM SET COLOR AND RETURN HOME
120 END
```

11.04.02 Blind Cursor Mode

The blind cursor mode can be entered in two ways. The first is by using the CURSOR X,Y control sequence. If the X value is 81 or larger, then the terminal ignores this as the visible cursor X value and sends the unit into the blind cursor addressing mode. Once in the blind cursor X,Y addressing mode, three additional bytes must be sent. They are the blind cursor X value, the blind cursor Y value, and the blind status word. The blind X value must be in the range 0-63 and the blind Y value must be in the range 0-31. The blind status word has the same format as that required for the CCI code (PLOT 6) as described in Section 11.02.

The Blind A7 Bit will be set on by sending values from 128 to 255 instead of 65 to 127 when going from the visible cursor X,Y mode to the blind cursor X,Y mode. The Blind A7 Bit is set off when a value from 65 to 127 is used.

It should be noted that the X and Y cursor values are masked to 0-63 ad 0-31 respectively.

After receiving the five byte blind cursor X,Y sequence, the terminal is left in the blind cursor mode for whatever input device caused the mode to be entered. After CPU RESET, the keyboard and RS-232C serial port are placed in visible cursor mode. If the keyboard causes the blind cursor XY to be addressed, then the keyboard will be left in the blind cursor mode while the RS-232C serial is still in the visible cursor mode. This allows the keyboard and the RS-232C to use two different cursors.

It is important to note that most of the control codes affect only the visible cursor mode including all of the cursor positioning codes except, of course, CURSOR X,Y, which can affect both modes, and ERASE PAGE which

resets both the visible and blind cursor to the home position (Ø,Ø). The setting of the Flag Bit is used by both the blind and visible cursor modes to select the proper special characters.

The blind cursor mode can also be entered by using the ESC A sequence, and ESC ESC returns the input to the visible cursor mode without changing the cursor address, composite color status word, or A7 Bit of the two cursor modes.

In BASIC the blind cursor X,Y addressing is used as follows:

```
PLOT 3, BC, X, Y, CL
```

where BC is in the range 65 to 127 for A7 OFF and small characters, and from 128 to 255 for A7 ON and large characters. X and Y are the cursor positions, and CL is the color status word (Ø-255). Blind cursor mode can be exited and visible cursor mode entered by:

```
PLOT 27,27      (ESC ESC)
```

and blind cursor mode can be re-entered by:

```
PLOT 27,1       (ESC A)
```

With the two cursor modes, Intecolor offers a great deal of flexibility as a color display device and a terminal. The blind cursor mode is useful for generating displays in character mode without the cursor interfering with the display. For example, the differences are shown by the hunter and turkey program below:

```
Ø REM TURKEY AND THE HUNTER
1Ø REM VISIBLE AND BLIND CURSOR DEMONSTRATION
2Ø PLOT 6,2,12: INPUT "VISIBLE OR BLIND CURSOR (V/B)?";A$
3Ø BC=MID$(A$,1,1)="B": VC=MID$(A$,1,1)="V"
4Ø IF BC+VC<>-1 THEN 2Ø
5Ø REM DRAW BORDER AROUND SCREEN
6Ø PLOT 27,24: REM PAGE MODE
61 PLOT 15: REM A7 OFF - SMALL CHARACTERS
62 PLOT 6,Ø: REM SET COLOR - BLACK FG/BLACK BG
63 PLOT 12: REM ERASE PAGE
7Ø PLOT 6,15: REM SET COLOR - WHITE FG/RED BG
71 FOR I=1 TO 64: PLOT 32:NEXT:REM DRAW TOP LINE
72 PLOT 3,Ø,31: REM MOVE CURSOR TO BOTTOM LINE
73 FOR I=1 TO 64: PLOT 32: NEXT: REM DRAW BOTTOM LINE
74 PLOT 27,1Ø: REM WRITE VERTICAL MODE
75 PLOT 8: REM MOVE CURSOR TO HOME
76 FOR I=1 TO 32: PLOT 32: NEXT: REM DRAW LEFT SIDE
77 PLOT 3,63,Ø: REM MOVE CURSOR TO TOP RIGHT
78 FOR I=1 TO 32: PLOT 32: NEXT: REM DRAW RIGHT SIDE
79 PLOT 27,24: REM PAGE MODE
9Ø PLOT 3,64,Ø: REM MOVE BLINKING CURSOR OFF SCREEN
1ØØ REM SET UP GAME PARAMETERS
11Ø HX=1: HY=1: REM HUNTER INITIAL POSITION
12Ø TX=32: TY=16: REM TURKEY INITIAL POSITION
13Ø TS=2: REM TURKEY SPEED
```

```
150 HC=39: TC=15
180 REM DEFINE FNR TO RETURN RANDOM INTEGER IN RANGE -X TO X
190 DEF FN R(X)= -X+INT((2*X+1)*RND(1))
200 REM MOVE CURSOR TO TURKEY'S OLD POSITION
201 IF VC THEN PLOT 3,TX,TY,6,0: REM VISIBLE
202 IF BC THEN PLOT 3,127,TX,TY,0: REM BLIND
210 TX=TX+FNR(TS): REM CHANGE TURKEY X POSITION
220 TY=TY+FNR(TS): REM CHANGE TURKEY Y POSITION
230 IF TX<1 OR TX>62 OR TY<1 OR TY>30 THEN 1000: REM ESCAPE!
240 PLOT 32: REM CLEAR TURKEY'S LAST POSITION
250 REM MOVE CURSOR TO TURKEY'S NEW POSITION
251 IF VC THEN PLOT 3,TX,TY,6,TC: REM VISIBLE
252 IF BC THEN PLOT3,127,TX,TY,TC: REM BLIND
260 PLOT ASC("T"): REM OUTPUT TURKEY SYMBOL
300 REM MOVE CURSOR TO HUNTER'S OLD POSITION
301 IF VC THEN PLOT 3,HX,HY,6,0: REM VISIBLE
302 IF BC THEN PLOT 3,127,HX,HY,0: REM BLIND
310 REM RANDOM SELECT HUNTER'S MOVE IN X OR Y DIRECTION
320  IF  RND(1)>(ABS(TY-HY)+1)/(ABS(TY-HY)+ABS(TX-HX)+2)THEN500
400 HY=HY+SGN(TY-HY): REM MOVE TOWARD TURKEY IN Y DIRECTION
410 GOTO 600
500 HX=HX+SGN(TX-HX): REM MOVE TOWARD TURKEY IN X DIRECTION
600 PLOT 32: REM CLEAR HUNTER'S LAST POSITION
700 REM MOVE CURSOR TO HUNTER'S LAST POSITION
701 IF VC THEN PLOT 3,HX,HY,6,HC: REM VISIBLE
702 IF BC THEN PLOT 3,127,HX,HY,HC: REM BLIND
710 PLOT ASC("H"): REM OUTPUT HUNTER SYMBOL
720 IF HX=TX AND HY=TY THEN 2000: REM HUNTER CATCHES TURKEY
800 GOTO 200
1000 REM TURKEY ESCAPES
1010 PLOT 27,27: REM VISIBLE CURSOR MODE
1020 PLOT 6,2: REM SET COLOR - GREEN FG/BLACK BG
1030 PLOT 8: REM CURSOR HOME
1040 PRINT "TURKEY ESCAPES !!!!   "
1050 GOTO 3000
2000 REM HUNTER CATCHES TURKEY
2010 PLOT 27,27: REM VISIBLE CURSOR MODE
2020 PLOT 6,2: REM SET COLOR - GREEN FG/BLACK BG
2030 PLOT 8: REM CURSOR HOME
2040 PRINT "GOBBLE GOBBLE .....  "
3000 FOR I=1 TO 1000: NEXT: REM DELAY FOR A WHILE
3010 RUN
```

11.05 Vector Graphics

The vector graphics capability of Intecolor allows the user to draw almost any desired display. The vector graphics are enabled by entering the graphic plot mode by depressing CONTROL B (binary 2) from the keyboard when in the CRT mode, or by executing PLOT 2 in BASIC. While in the graphic plot submode the user can choose from sixteen (16) plot submodes that perform a variety of graphic functions. The initial plot submode is the XY Point Plot mode. In this mode the user can turn on and off individual plot blocks on the screen. Other plot submodes can easily be entered by a binary code from 240 to 255.

An additional feature is available to allow a graphic plot to be erased by simply setting the FLAG bit on before entering the plot mode. This causes a logical XOR function to be used in setting the plot blocks. Thus, if the same point is plotted a second time, it is erased. Also, any plot submode may be entered from any other plot submode except Character Plot mode. The various submodes and their interactions are explained in detail below.

Colors may be defined on a character by character basis only and the color of an individual plot block as well as other intensified plot blocks within a character will be the most recent color defined when a new plot block within that character is turned on. To change color, it is necessary to exit the current plot submode, set the new color, and re-enter the plot mode.

The character grid on the screen is 64 characters wide and 32 characters high. The zero reference point for all plotting is the lower left hand corner of the screen. Each character is further subdivided into 8 plot blocks — 2 blocks wide and 4 blocks high. This gives a 128 by 128 grid of plot blocks which may be individually set. All plot submodes operate on this grid size and have the same reference point (Ø,Ø). Positive directions are up and to the right, and negative directions are down and to the left.

All plot submodes and the general Plot Mode are terminated or exited by the binary code 255. When ever this code is issued, the plot mode is terminated and must be re-entered by issuing a CONTROL B or binary 2.

On the deluxe keyboards there are sixteen (16) special functions keys labelled FØ through F15. Using these keys the various plot submodes can be entered directly in the CRT mode (not in BASIC.) The FØ key produces a binary 240 code, F1 a 241, etc., up to the F15 key which produces a 255. In BASIC these plot submodes are entered by using the PLOT statement as described below.

Plot Mode Escape - (255 binary)

This code is used to exit from the Plot Mode or any of the plot submodes. On the deluxe keyboards the F15 function key performs a Plot Mode Escape. On other keyboards, the combination of CONTROL SHIFT ? or COMMAND ? serves the purpose.

Character Plot - (254 binary)

The Character Plot Submode is entered by a 254 after the general Plot Mode is entered. All subsequent characters issued are treated as plot characters except for 255 which is the Plot Mode Escape. Thus, other plot submodes can not be entered directly from this mode. The plot characters are constructed by ORing together the selected plot blocks to form the composite character as follows:

01 HEX	1 0	10 HEX	0 1
1 Dec	0 0	16 Dec	0 0
	0 0		0 0
	0 0		0 0
02 HEX	0 0	20 HEX	0 0
2 Dec	1 0	32 Dec	0 1
	0 0		0 0
	0 0		0 0
04 HEX	0 0	40 HEX	0 0
4 Dec	0 0	64 Dec	0 0
	1 0		0 1
	0 0		0 0
08 HEX	0 0	80 HEX	0 0
8 Dec	0 0	128 Dec	0 0
	0 0		0 0
	1 0		0 1

The Character Plot causes the the 6 wide by 8 high dot matrix to be divided into 8 blocks organized 2 blocks wide and 4 blocks high. Each block consists of a dot matrix 3 dots wide and 2 dots high. Each block corresponds to an individual bit of the 8 bit plot character. Large characters may also be formed by using the plot blocks in several character positions to create a large 5 by 7 matrix or any other desired size.

X Point Plot - (binary 253)

The X Point Plot is automatically entered upon receipt of the general Plot Mode code, binary 2 or CONTROL B. It may also be entered directly from any of the other plot submodes. After entering the X Point Plot submode, the next byte received defines the X value of the block that is desired to be plotted. The X value may range from 0 to 127 and all other values will cause 128 to be subtracted from the value of X.

The X Point Plot may be terminated by the code 255 which also causes the the general Plot Mode to be terminated. Any of the other plot submodes may be entered directly from the X Point Plot by simply entering the appropriate plot submode codes from 240 to 255.

It should be noted that this plot submode does not cause a plot block to be intensified, it only defines the X value. Once the X value is received, the Intecolor unit is automatically placed in the Y Point Plot mode. Thus, the next code sent will be the Y value which may range from 0 to 127.

The procedure for entering and exiting the X Point Plot mode is shown in the following table:

Function	Code
Plot Mode	2
Xl Value	Ø to 127
Yl Value	Ø to 127
.	.
.	.
.	.
Xn Value	Ø to 127
Yn Value	Ø to 127
Plot Escape	255
or	
Plot Submode	240 to 254

The X Point Plot in conjunction with the Y Point Plot allows any block on a 128 by 128 block matrix to be intensified. Thus, in BASIC the above sequence becomes:

```
PLOT 2,X1,Y1, ... ,XN,YN,255
```

The following statement will plot points at the screen's four corners:

```
PLOT 2, Ø,Ø, Ø,127, 127,127, 127,Ø, 255
```

Y Point Plot - (binary 252)

The Y Point Plot is entered by a binary 252 code after the general Plot Mode is entered or automatically from the X Point Plot submode after the X value has been sent. The next byte received after entering the Y Point Plot submode defines the Y value of the block to be plotted and intensifies that block. If the new block is within a character position that contains an ASCII character, then the ASCII character is replaced completely by the new block and its associated color.

XY Incremental Point Plot - (binary 251)

The XY Incremental Point Plot submode is entered by a binary 251 code while in the general Plot Mode. The next byte defines the next two (2) increments as shown below. This byte may take on values in the range Ø to 239 since the binary codes from 240 to 255 are used for the plot submodes.

```
b7  b6  b5  b4  b3  b2  b1  bØ

[ X ]   [ Y ]   [ X ]   [ Y ]
  1       1       2       2

Plot Block 1    Plot Block 2
```

The 4 two bit codes are defined as follows:

Ø	No change
1	Negative increment
2	Positive increment
3	No change

If b0 through b3 are "0"s, then the plot block will not plot, but will still increment according to the coding of b4 through b7. This allows skipping a plot increment by plotting an "invisible" block. The XY Incremental Plot mode may be terminated by the Plot Mode Escape code 255. The following sample program will do a random walk using the Incremental Point Plot mode.

```
10 DEF FNR(X)=INT(X*RND(1))
20 PLOT 12,6,6: REM CLEAR SCREEN AND PLOT IN LIGHT BLUE
30 PLOT 2,63,63: REM PLOT POINT IN THE MIDDLE OF THE SCREEN
40 PLOT 251: REM ENTER INCREMENTAL POINT PLOT MODE
50 FOR I=1 TO 1000
60 INC=FNR(3)*64+FNR(3)*16+FNR(3)*4+FNR(3)
70 REM  USE ONLY THE FIRST THREE DIRECTION CODES
75 IF (INC AND 15)=0 THEN 60: REM NO ALLOW INVISIBLE BLOCKS
80 PLOT INC
90 NEXT I
100 PLOT 255: REM ESCAPE FROM PLOT MODE
110 END
```

X Bar Graph, X0 Value - (250 binary)

The X Bar Graph, X0 Value plot submode is entered by a binary 250 code after the general Plot Mode is entered. It may also be entered directly from any of the other plot submodes except for Character Plot. After entering the X Bar Graph, X0 Value submode, the next byte defines the X0 value or the left horizontal start block of the horizontal bar graph. The X0 may range in value from 0 to 127 and all other values have 128 subtracted giving a new X0 value in the range 0 to 127.

Upon receiving the X0 value, the value of X0 is stored in memory and the Intecolor is automatically placed in the X Bar Graph, Y Value plot submode (249 binary.) After receiving the next byte as the Y value, the Intecolor is automatically placed in the X Bar Graph, X Max Value plot submode (248 binary.) After receiving the X Max value the horizontal bar graph is drawn on the screen and the Intecolor is placed back in the X Bar Graph, Y Value plot submode ready to receive new Y and X Max value pairs until a new plot submode is entered. Note that once an X0 value is defined it is unnecessary to respecify it for each horizontal bar in the graph. This process is shown in the following example.

Function	Code
Plot Mode	2
or	
Plot Submode	240 to 253
X Bar Graph, X0 Value	250
X0 value	0 to 127
Y value - line 1	0 to 127
X Max value - line 1	0 to 127
.	.
.	.

```
  .                          .
  .                          .
Y value - line n          Ø to 127
X Max value - line n      Ø to 127

Plot Escape               255
  or
Plot Submode              24Ø to 254
```

For example, from BASIC a horizontal bar graph plotting a sine function can be drawn as follows:

```
1Ø PLOT 6,6,12  : REM SET COLOR TO CYAN AND CLEAR SCREEN
2Ø XØ = 1Ø      : REM SET XØ VALUE
3Ø PLOT 2,25Ø,XØ: REM ENTER X BAR GRAPH SUBMODE - SET XØ
4Ø FOR Y=Ø TO 127 STEP 2: REM SET Y VALUES
5Ø PLOT Y,XØ+5Ø*(1+SIN(Y/1Ø)): REM SCALE SINE FUNCTION
6Ø NEXT Y
7Ø PLOT 255     : REM PLOT ESCAPE
```

As can be seen from the above examples, once in the X Bar Graph, XØ mode, it is necessary to define only two points for each new bar graph. The bar graph is drawn after receiving the X Max value. Any of the other plot submodes can be entered directly from the three X Bar Graph submodes. Multiple colored bar graphs can be drawn by leaving plot mode, changing the color, and re-entering the X Bar Graph, Y Value submode (249 binary.) In this case the original XØ value would be preserved. Bars drawn in this mode are one plot block wide; thicker bars can be drawn by changing the Y value by 1 and replotting it along with the same X Max value or using the X Incremental Bar Graph submode.

X Bar Graph, Y Value - (249 binary)

The X Bar Graph, Y Value plot submode is entered by a binary 249 code or automatically from the X Bar Graph, XØ Value plot submode. After entering this submode the next byte is used as the Y value of the next bar in the graph to be plotted, and the Intecolor is automatically placed into the X Bar Graph, X Max Value plot submode (248 binary.) Any of the other plot submodes can be entered directly from this submode. For more information on this submode see the description of the X Bar Graph, XØ Value submode (25Ø binary.)

X Bar Graph, X Max Value - (248 binary)

The X Bar Graph, X Max Value plot submode is entered by a binary 248 code or automatically from the X Bar Graph, Y Value plot submode. After entering this submode the next byte is used as the X Max value of the bar in the graph. The bar is plotted, and the Intecolor is automatically placed into the X Bar Graph, Y Value plot submode (249 binary) which allows the next bar to be defined and drawn. Any of the other plot submodes can be entered directly from this submode. For more information on this submode see the description of the X Bar Graph, XØ value submode (25Ø binary.)

X Incremental Bar Graph - (247 binary)

The X Incremental Bar Graph plot submode is entered by a binary 247 code. After entering this submode the next byte defines the next two horizontal and vertical increments for two horizontal bar graphs. Thus, it is possible to position a bar graph on either side of the present location by adding or subtracting an increment to the bar graph previously defined. The coding and composition of the incremental direction code is the same as that defined in the XY Incremental Point Plot submode (251 binary.) Any of the other plot submodes can be entered directly from this submode.

Y Bar Graph, Y0 Value - (246 binary)

The Y Bar Graph, Y0 Value plot submode is entered by a binary 246 code after the general Plot Mode is entered. It may also be entered directly from any of the other plot submodes except for Character Plot. After entering the Y Bar Graph, Y0 Value submode, the next byte defines the Y0 value or the bottom vertical start block of the vertical bar graph. The Y0 may range in value from 0 to 127 and all other values have 128 subtracted giving a new Y0 value in the range 0 to 127.

Upon receiving the Y0 value, the value of Y0 is stored in memory and the Intecolor is automatically placed in the Y Bar Graph, X Value plot submode (245 binary.) After receiving the next byte as the X value, the Intecolor is automatically placed in the Y Bar Graph, Y Max Value plot submode (244 binary.) After receiving the Y Max value the vertical bar graph is drawn on the screen and the Intecolor is placed back in the Y Bar Graph, X Value plot submode ready to receive new X and Y Max value pairs until a new plot submode is entered. Note that once a Y0 value is defined, it need not be respecified for each vertical bar in the graph. This is shown in the following example.

Function	Code
Plot Mode	2
or	
Plot Submode	240 to 253
Y Bar Graph, Y0 Value	246
Y0 value	0 to 127
X value - line 1	0 to 127
Y Max value - line 1	0 to 127
.	.
.	.
.	.
X value - line n	0 to 127
Y Max value - line n	0 to 127
Plot Escape	255
or	
Plot Submode	240 to 254

For example, from BASIC a vertical bar graph plotting the area under a

random function can be drawn as follows:

```
10 PLOT 6,6,12  : REM SET COLOR TO CYAN AND CLEAR SCREEN
20 Y0 = 10      : REM SET Y0 VALUE
30 PLOT 2,246,Y0: REM ENTER Y BAR GRAPH SUBMODE - SET Y0
40 FOR X=0 TO 127 STEP 2: REM SET X VALUES
50 PLOT X,Y0+100*RND(1) : REM SCALE RANDOM FUNCTION
60 NEXT X
70 PLOT 255     : REM PLOT ESCAPE
```

As can be seen from the above examples, once in the Y Bar Graph, Y0 mode, it is necessary to define only two points for each new bar in the graph. The bar graph is drawn after receiving the Y Max value. Any of the other plot submodes can be entered directly from the three Y Bar Graph submodes. Multiple colored bar graphs can be drawn by leaving plot mode, changing the color, and re-entering the Y Bar Graph, X Value submode (245 binary.) In this case the original Y0 value is preserved. Bars drawn in this mode are one plot block wide; thicker bars can be drawn by changing the X value by 1 and replotting it along with the same Y Max value or using the Y Incremental Bar Graph submode.

Y Bar Graph, X Value - (245 binary)

The Y Bar Graph, X Value plot submode is entered by a binary 245 code or automatically from the Y Bar Graph, Y0 Value plot submode. After entering this submode the next byte is used as the X value of the next bar to be plotted, and the Intecolor is automatically placed into the Y Bar Graph, Y Max Value plot submode (244 binary.) Any of the other plot submodes can be entered directly from this submode. For more information on this submode see the description of the Y Bar Graph, Y0 Value submode (246 binary.)

Y Bar Graph, Y Max Value - (244 binary)

The Y Bar Graph, Y Max Value plot submode is entered by a binary 244 code or automatically from the Y Bar Graph, X Value plot submode. After entering this submode the next byte is used as the Y Max value of the bar in the graph. The bar is plotted, and the Intecolor is automatically placed into the Y Bar Graph, X Value plot submode (245 binary) which allows the next bar to be defined and drawn. Any of the other plot submodes can be entered directly from this submode. For more information on this submode see the description of the Y Bar Graph, Y0 value submode (246 binary.)

Y Incremental Bar Graph - (243 binary)

The Y Incremental Bar Graph plot submode is entered by a binary 243 code. After entering this submode the next byte defines the next two vertical and horizontal increments for two vertical bar graphs. Thus, it is possible to position a bar graph on either side of the present location by adding or subtracting an increment to the bar graph previously defined. The coding and composition of the incremental direction code is the same as that defined in the XY Incremental Point Plot submode (251 binary.) Any of the other plot submodes can be entered directly from this submode.

XØ Vector Plot - (242 binary)

The XØ Vector Plot submode is entered by a binary 242 code after the general Plot Mode is entered. After entering the XØ Vector Mode the next byte defines the XØ point of the vector being drawn. The vector mode requires two endpoints to be defined (i.e. XØ,YØ and X1,Y1.) The X1,Y1 values should be previously defined by way of the X and Y Point Plot submodes (253 and 252 binary.) Upon receiving the XØ value the Intecolor is automatically placed into YØ Vector Plot submode. After receiving the YØ value the Intecolor plots the best fitting straight line between XØ,YØ and X1,Y1 using the plot blocks and returns to the XØ Vector Plot submode, ready to plot vectors between successive XØ,YØ pairs. This process is shown below:

Function	Code
Plot Mode	2
or	
X Point Plot	253
X1 Vector point 1	Ø to 127
Y1 Vector point 1	Ø to 127
XØ Vector Plot	242
XØ Vector point 1	Ø to 127
YØ Vector point 1	Ø to 127
.	.
.	.
.	.
XØ Vector point n	Ø to 127
YØ Vector point n	Ø to 127
Plot Escape	255
or	
Plot Submode	240 to 254

Thus, in BASIC the above sequence becomes

```
100 PLOT 2, X1,Y1
110 PLOT 242
120 FOR I=1 TO N
130 PLOT XØ(I),YØ(I)
140 NEXT I
150 PLOT 255
```

To plot a rectangle around the entire screen simply execute the statement

```
PLOT 2, Ø,Ø, 242, Ø,127, 127,127, 127,Ø, Ø,Ø, 255
```

YØ Vector Plot - (241 binary)

The YØ Vector Plot submode is entered by a binary 241 code after the general Plot Mode is entered. After entering this submode the next byte defines the YØ value of the vector being drawn. There is no restriction on YØ except that it must be in the range Ø to 127. Upon receiving the YØ

value a vector is plotted from X1,Y1 to X0,Y0 with X0,Y0 replacing the old X1,Y1 endpoint. If the next vector has a X1,Y1 value equal to the old X0,Y0 value, then only the new X0,Y0 values need be sent. This effectively draws a vector from the present X0,Y0 position to the new X0,Y0 position. For more information on this submode see the description of the X0 Vector Plot submode (242 binary.)

Incremental Vector Plot - (240 binary)

The Incremental Vector Plot submode is entered by a binary 240 code after the general Plot Mode is entered. After entering this submode the next byte defines the increments in the X0,Y0 and X1,Y1 values for the vector from X1,Y1 to X0,Y0. The values for the increments are defined as follows:

```
b7  b6  b5  b4  b3  b2  b1  b0

[ X ] [ Y ] [ X ] [ Y ]
  1     1     0     0
```

The 4 two bit codes for the increments are defined as follows:

0	No change
1	Negative increment
2	Positive increment
3	No change

The incremental direction codes are similar to those used for the other increment plot submodes. Furthermore, if either half of the word is all zeroes, then the corresponding X,Y values will be changed but no vector will be drawn. This allows endpoints for the vectors to be skipped. The only time a vector is drawn is when both halfs of the word are non-zero. The Incremental Vector Plot submode does not automatically transfer control to any other plot submode. Therefore, a series of incremental movements in both X1,Y1 and X0,Y0 can be made by sending consecutive incremental direction codes.

11.06 RS-232C Interface

The RS-232C interface allows the user to connect many types of RS-232C compatible devices to the Intecolor unit. For example, this feature enables some serial printers to be interfaced without any additional software. Communication with other terminals is possible either by direct wiring or via modems connected to telephone lines. Chapter 13 describes use of the Intecolor unit as a terminal. The following paragraphs describe use of the RS-232C Interface under control of BASIC.

The RS-232C port is controlled by several escape codes which set the baud rate of the serial output and direct all output to the serial port. The default baud rate of the serial port is 9600 baud with 1 stop bit. This rate can be changed by using the ESC R sequence. The setting of the A7 Bit determines the number of stop bits when the ESC R sequence is given. A7 OFF gives 2 stop bits (normal for 110 baud) and A7 ON gives 1 stop bit. The baud rate is selected by issuing the sequence ESC R followed by a

character in the range 1 through 7 which specifies the baud rate as shown in the table following.

BAUD RATE SELECTION

NUMBER KEY	1	2	3	4	5	6	7
BAUD RATE	110	150	300	1200	2400	4800	9600

After the baud rate has been properly set, data can be transmitted to the RS-232C serial port by executing an ESC M sequence. Once this escape sequence has been issued, all inputs to the CRT display drives (including all keyboard inputs and BASIC outputs) are directed to the RS-232C port instead of the CRT screen. Thus, the only way to break out of this mode from the CRT mode is via the CPU RESET key or in Disk BASIC by executing a POKE 33265,0 statement which resets the BASIC Output Flag to send characters to the screen in visible cursor mode. In BASIC the Intecolor unit can be reset to the previous output mode by saving the contents of the BASIC Output Flag with an X=PEEK(33265) function before issuing an ESC M sequence and then restoring the BASIC Output Flag by executing the statement POKE 33265,X as follows:

```
10 REM SET BAUD RATE = 300, 1 STOP BIT
20 PLOT 14,27,18,3,15
100 GOSUB 9000: REM DIRECT OUTPUT TO RS-232C PORT
110 FOR I=1 TO 10
120 PRINT "THIS IS AN RS-232C TEST"
130 NEXT
140 GOSUB 9500: REM RESET OUTPUT TO CRT
150 PRINT "BACK TO THE CRT"
160 END
9000 TMP = PEEK(33265): REM SAVE BASIC OUTPUT FLAG
9010 POKE 33265,14: OUT 8,199: REM BASIC OUTPUT TO RS-232C
9015 REM DISABLE KEYBOARD
9020 RETURN
9500 POKE 33265, TMP: OUT 8,207: REM RESET OUTPUT TO CRT
9505 REM ENABLE KEYBOARD
9510 RETURN
```

Several other control and escape codes interact with the RS-232C serial port. The ESC C sequence transmits the current cursor location and the color status to the RS-232C port using the following 7-byte sequence:

3, X, Y, 6, Status, ASCII Character, 13

The X,Y values are the current cursor position, Status is the color status word of the ASCII Character stored at the cursor position. Using the CONTROL X code, every text character on the screen is transmitted to the RS-232C port from the visible cursor to the end of the page or until an FF,00 sequence is found in the screen refresh RAM. The text characters are sent in lines terminated by a linefeed and carriage return. The color status is not transmitted.

NOTE - When interfacing any device to the RS-232C serial port directly (rather than via modems) it may be necessary to switch the transmit and receive data lines at the device end of the cable (lines 2 and 3). See Appendix E.4 for pin assignments on the Intecolor connectors.

11.07 Miscellaneous Escape Codes

Intecolor has several additional escape codes to test the display, and jump to fixed and user defined memory locations. A test pattern can be generated on the CRT screen using the ESC Y test mode sequence. By issuing an ESC Y followed by a character, the entire screen is filled with that character using the current visible status word as the visible status word for each character on the CRT screen.

Several of the remaining ESCAPE codes have been pre-programmed to execute JMP's to certain memory locations as outlined below.

ESCAPE CODE	MEMORY LOCATION	
	HEX	DECIMAL
I	9000	36864
S	A000	40960
T	8200	33280
^	81BF	33215

The ESC ^ is a user definable escape code. By POKEing an 8080 JMP instruction into the three bytes starting at 33215 a jump to any location in memory can be defined. The sequence in this case is placement of the JMP intruction in location 33215, the low byte address in 33216 and the high byte address in 33217. For example, if ESC ^ is to be used to jump to a user-defined routine or program at memory location F000 Hex (61,440 decimal), the sequence would be:

```
POKE 33215,195: REM DECIMAL VALUE FOR JMP INSTRUCTION
POKE 33217,240: REM HIGH BYTE OF MEMORY ADDRESS - DECIMAL
POKE 33216,00 : REM LOW BYTE OF MEMORY ADDRESS
```

The decimal version of the high byte F0 is 240; the decimal version of the low byte 00 is 00.

(Refer to Appendix G.1, page 174 for definition of the unconditional jump instruction JMP.)

12. FILE CONTROL SYSTEM

12.01 Introduction to FCS

The File Control System used in the 3650 Series Intecolor, FCS for short, is used to manage the Sof-Disks which store programs. The File Control System enables the user to store or save and retrieve programs, screen displays and other data.

The Escape D sequence is used to enter FCS -- first strike the (ESC) key and then the D key. The screen will display the prompt **FCS>**. The unit is then ready to respond to the commands of the File Control System. The commands use upper case letters. (The codes for the lower case letters also are acceptable; however, the standard 3651 does not have lower case and the special characters would be difficult to read.)

There are two types of FCS commands: internal commands which are executed by routines within FCS itself and external commands which are executed by disk file programs of type .COM. Both commands are used in the same manner, but any external command used must exist as a .COM type file on a diskette in the default disk drive (or a specified disk drive in a 3650 Series system with multiple disk drives).

12.02 The FCS Command Format

Following are definitions of the elements used in describing the FCS commands:

( ) denotes a mandatory element.
[ ] denotes an optional element which, if not specified, will result in the default type.

(Device name:) means [Device type] [Number] (:)
'Device type' is MD, DM. FD or DF. The 'Number' is 0 (for an internal disk drive or for the left hand drive of a dual drive unit), 1, 2 or 3. After a default type device has been selected, only the number of the device is required.

(Memory spec) means (Load address) (byte count) or
(Load address) (-end address)
All memory addresses are in Hex format.

(File spec) means (File name) [.Type] [;Version]
'File name' may be up to six and 'Type' up to three characters and/or digits. The default type is .PRG when RUNning a program and .LDA when LOAding a program. 'Version' is 01 to FF Hex.

Device Type

The 3650 Series Intecolor offers disk storage in a number of sizes and types of drives. A standard system can support the following devices:

MD type	5.25" Single-sided Mini-Disk	90K bytes/drive
DM type	5.25" Double-sided Mini-Disk	180K bytes/drive
FD type	8" Single-sided Floppy Disk	290K bytes/drive
DF type	8" Double-sided Floppy Disk	580K bytes/drive

The number of bytes shown above is the number accessible without turning the diskette over. The diskettes for the single-headed MD type drive can be formatted on both sides to provide a total of 180K bytes of storage. However, with the diskette in the drive, only 90K is accessible at a time. Diskettes for the single-headed FD type drive may be formatted on one side only. The double-headed DM and DF type drives utilize both sides of the diskettes in flip-flop fashion, using first one head and then the other.

Standard software supports drive numbers of 0 through 3 for up to four 8" drives and 0 through 2 for up to three 5.25" drives.

Memory Spec

The 'Memory Spec' is the 'Start Address' in hex followed by the number of bytes or followed by hyphen (-) and the 'End Address'. Alternately it can be the 'Start Address' and the length in bytes -- both values in hex.

File Spec

The 'File Spec' is the name that the user has assigned to the file, followed by the file type (.PRG, .LDA, .BAS, etc.) and, optionally, a semicolon (;) followed by a version number in the range 01 to FF hex. If the specified file is being read, then the default version is the file with the largest version number. For files being written, the default version number is one higher than the largest version number of an existing file on the specified device. If no file currently exists on the disk with the specified name, then the default version number is 01.

File types, as noted above, can be any three characters. Some common varieties are .BAS, .PRG and .LDA.

A .BAS type file results when a BASIC program is SAVED in BASIC, as described in section 10.01.

A .PRG type file is created with the FCS SAVE command. It is a machine-code program in "memory image" form. The information in the file is a contiguous memory image of the program. The RUN command loads a .PRG file into memory starting at the specified Load Address in the file's directory entry, and begins execution at the Start Address specified in the file's directory entry. A .PRG file is loaded into memory more quickly than an .LDA file. Therefore, once a program is working, it should be saved in .PRG form with the SAVE command, so that subsequent RUN's of the program will be quicker.

An .LDA type file is created by the INTECOLOR 8080 Assembler. The file consists of one or more data records and is terminated by one end record. Each data record specifies a load address for the record, and one or more data bytes to be loaded sequentially into memory starting at the load address. The end record specifies the starting (execution) address for the program (the operand of the END statement in the source program).

An .SRC type file is created by the INTECOLOR Text Editor or Screen Editor. Such a file can be assembled by the INTECOLOR 8080 Assembler to become an .LDA type. It can then be SAVED in FCS, to become a .PRG type file.

Paragraph 12.03.04 describing the DIRectory command includes information regarding the location of the starting address and the load address in the directory.

12.03 Internal FCS Commands

This section describes and gives examples of internal FCS commands. In the use of FCS commands, only the first three letters of any internal command are required. Many FCS commands assume a disk has been initialized.

A word of CAUTION: Note that the execution of certain commands results in loss of data already on a disk.

12.03.01 The COPY Command

The COPY command allows the user to copy a file, possibly to another disk drive. It uses the disk buffer. It is of the form:

COPY [Device Name:] (File Spec) TO [Device Name:] (File Spec)

For example:

COP MD0:TEST.PRG TO MD1:ABC

This command copies the latest version of TEST.PRG on device MD0: to file name ABC.PRG on device MD1:.

Another example:

COP TEST.PRG;02 TO 1:

This command copies version 02 of TEST.PRG from a disk in the default device to a similar device number 1. The disk in the default device may contain higher numbered versions of TEST.PRG but with this command the selected version 02 is the one copied. It will retain the specification TEST.PRG;02 on the new disk.

The COPY command can be used to transfer a file from one type of device to another:

COP MD0:TEST.PRG;02 TO FD0:

12.Ø3.Ø2 The DELETE Command

The DELETE command allows any file on the Sof-Disk to be deleted and is of the form:

DELETE [Device Name:] (File Spec)

For example:

```
DEL TEST.BAS;1  (or DEL TEST.BAS;Ø1)
DEL 1:TEST.PRG;2
DEL FD1:NAME.RND;1
```

The complete File Spec is needed to delete a file. This form of file protection is provided to prevent accidental erasures. (The exact wording of the program name and the version can be found by typing DIR and a RETURN after the FCS> prompt.) The DELETE command repacks the disk and directory, using the disk buffer in the process.

12.Ø3.Ø3 The DEVICE Command

The DEVICE command allows the user to change the default device or drive, and is of the form:

DEVICE [Device Name:]

If the Device Name is not specified, then the current default device is listed. For example, if one has just powered up an Intecolor system with two single sided 8" drives and types DEV after the FCS> prompt, the display is:

```
FCS>DEV
DEFAULT DEVICE FDØ:
```

However, if one types DEV FD1: the display is:

```
FCS>DEV FD1:
DEFAULT DEVICE FD1:
```

The default device has been changed from the initial FDØ: (left hand drive) to FD1: (the right hand drive). In the examples above the bold face type **FCS>** shows the screen response to the (ESC) D key sequence for entering FCS; the standard type shows the display of the typed command; then the second line of bold face shows the computer's response to the command.

The default device can be changed to another number or type device by entry of the appropriate command after the FCS> prompt at any time.

12.Ø3.Ø4 The DIRECTORY Command

The DIRECTORY command lists all the programs on the Sof-Disk in any device, and is of the form:

DIRECTORY [Device Name:]

For example:

```
DIR
DIR FD1:
```

The first example would result in the listing of the directory of the Sof-Disk in the default device. (At power-up this is normally the internal 5.25" drive or the left hand drive in a system with two 8" disk drives. Later it may be a different drive if the default device has been changed via a DEVICE command.) The second example would result in the listing of the directory of the diskette in device FD1:, no matter what the default device.

If the system has only one type of drive, the command may take the simpler form DIR Ø: or DIR 1: for listing the directory of the Sof-Disk in the internal or the external 5.25" drive respectively -- or the left hand drive or the right drive in a system with dual 8" drives.

A list of file names and associated parameters appears on the screen headed by the device type and number, the disk volume name (given when it was initialized), the number of blocks reserved for the directory and several column headings:

```
ATR  NAME  TYPE  VR  SBLK  SIZE  LBC  LADR  SADR
```

ATR (attribute) is a number assigned to the type of directory entry. The attribute of an ordinary file is 3 and the free space entry is 1.

NAME is the name of a file and consists of up to 6 letters and/or digits.

TYPE is an extension of the name and consists of up to 3 letters and/or digits. It is usually an indicator of the type of file and much software assumes certain 'type' names when files are used. For example, BASIC software assumes programs are of type BAS and the ASSEMBLER assumes source files are of type SRC. In cases where a file called for is not of the assumed (default) type, then a period (.) followed by the type must be appended to the name.

VR is the version number of the file. Version numbers are hexadecimal numbers between 1 and FF inclusive. Whenever a file is put on a disk, it is automatically assigned the version number following the highest one for any file of the same name and type already on the disk. The first time a file of a given name and type is put on a disk, it is automatically assigned version number Ø1. When a file already on disk is referred to in an FCS command, the one of the highest version number is assumed to be the intended file. If a file of a lower version number is intended, then a semicolon (;) followed by the desired version number must be appended to the name in the FCS command (following 'type' if designated).

SBLK is the hexadecimal number of the starting (128 byte) block of the disk where the file is stored.

SIZE is the hexadecimal number of blocks of the disk occupied by the file.

LBC is the hexadecimal number of bytes used in the last block of the file.

LADR is the hexadecimal loading address of the file and is the beginning memory address where the file will be loaded into memory when called for if no other address is indicated. The number is not used (or meaningful) for some kinds of files such as BASIC programs. For type .RND file, NREC is the number of logical records in the file.

SADR is the hexadecimal number of the starting address for the file after it is loaded into memory if it is a program. However, it is not used (or meaningful) for many files including BASIC programs and data files. For type .RND file, RSIZ is the number of bytes in each logical record.

Following these parameters for all the files on disk is given the starting block and number of blocks of disk space that remain available for additional files.

A directory listing may be halted by striking the BREAK key or CTRL/B key, and it may be resumed by striking the RETURN key. If the CTRL/C key is struck the directory is stopped and FCS is ready to receive another command.

12.03.05 The EXECUTE Command

The EXECUTE command allows a file consisting of FCS commands to be executed one after another in batch mode fashion and is of the form:

EXECUTE (File Spec) [parameter 1][,parameter 2]...[,parameter 9]

Such files are of default type .FCS and may be created using an editor such as ISC's SCREEN EDITOR. For any parameters called for in FCS commands of an FCS file, references to parameter in the EXE command line may be used instead: $1 for paramter 1, $2 for parameter 2, etc. For example, if DRUCK.FCS is a file consisting of

```
PRI $1
PRI $2
DEL $1
DEL TEST.PRG;01
```

and the diskette content includes the .COM file command program PRI.COM and, among any other files, the files UNU.SRC;01, DU.SRC and DRUCK.FCS, then execution of the command

EXE DRUCK UNU.SRC;01,DU

causes the file UNU.SRC;01 to be printed, followed by DU.SRC. Next, UNU.SRC;01 is deleted and finally TEST.PRG;01 is deleted. See the appropriate sections for complete descriptions of the PRI and DEL commands. Note that PRI is one of the standard .COM file commands rather than an internal one. Thus the default drive must have a diskette containing the program PRI.COM when the PRI command is used.

12.Ø3.Ø6 The INITIALIZE Command

The INITIALIZE command allows the user to assign a name of up to 1Ø characters to a Sof-Disk and optionally to assign the number of allotted directory blocks. This command clears all the directory information on a diskette, effectively deleting all files on the Sof-Disk. It should be used only when a "clean" Sof-Disk is desired.

The command is of the form:

INITIALIZE (Device Name:) (Volume Name) [No. of DIR blocks]

For example:

```
INI FD1:SAMPLENAME
INI DFØ:TESTDISKØ1 1Ø  (the 1Ø is optional) →
```

Each directory block can hold information on 6 files; however, 2 entries are necessary for the Volume Name and free space entries. Thus specifying 1Ø Hex as the number of directory blocks allows up to 94 entries, because 94 = (16 * 6) - 2. The maximum number of directory blocks is 1F hex.

The single-sided 5 1/4" Mini-Disk defaults to 5 blocks of directory space, the double-headed Mini-Disk to 11 blocks. An 8" single-sided Floppy Disk defaults to 18 blocks and a double-headed 8" to 31 blocks. The first and last blocks hold five file entries each; other blocks hold six.

12.Ø3.Ø7 The LOAD Command

The LOAD command allows the user to load any type file into any RAM location. The LOAD command uses the same format as the SAVE command. But the LOAD command operates differently depending on the file type loaded. The default type is .LDA.

To LOAD a file type other than .LDA, the command is of the form:

LOAD [Device Name:] (File Spec) [Load Address]

The file is assumed to be a "memory image" file and is loaded contiguously into memory starting either at the load address in the file's directory entry or at the load adress specified in the command line.

To load a file of type .LDA, the command is of the form:

LOAD [Device Name:] (File Spec) [Lowest Address [Memory Spec]]

Each data record in the file is loaded into memory. If Lowest Address and Memory Spec are not specified, then each record is loaded at the address specified in the record.

If Lowest Address and Memory Spec are specified, the default Memory Spec is B000-FFFF. A "memory range" will be determined as follows:

1. If the Memory Spec is omitted, the range will be B000-FFFF.

2. If one number, i.e. C000, is given for the Memory Spec, then the range will be specified by the given number as the low limit and FFFF as the high limit of the range.

3. If two numbers, separated by a hyphen are given for the Memory Spec, then the range is specified by those numbers.

4. If two numbers, separated by a space or comma, are given for the Memory Spec, then the first number will be the low limit of the range, and the second number is the byte count used to calculate the high limit of the range. For example, D000 400 will give a range D000-D3FF.

An "offset" will be calculated as "low limit of memory range" minus "Lowest Address". Each data record will then be loaded at the address specified in the record plus the "offset". Data will be loaded only within the "memory range" as determined above. If the "Lowest Adress" is not specified, then user programs can load at the start of the user RAM area, A120 HEX.

NOTE: BASIC programs should be LOADed and SAVEd in BASIC, not in FCS.

12.03.08 The READ Command

The READ command allows retrieval of information on any part of the Sof-Disk without regard to the directory or program boundaries. The command is of the form:

READ [Device Name:] (Start Block) (Memory Spec)

For example:

READ FD0: 20 8200-8FFF

reads 3584 bytes (0E00 Hex) from the drive 0 starting at block 32 (20 Hex) into the memory at 8200 to 8FFF.

12.03.09 The RENAME Command

The RENAME command allows the user, in one step, to change the file name, file, type and the version number separately or collectively without changing the information stored in the program. The statement is of the form:

RENAME [Device Name:] (File Spec) TO (File Spec)

For example:

```
REN TEST.PRG;1 TO NWTEST.PRG;2
```

renames the file TEST.PRG;1 to NWTEST.PRG;2.

12.03.10 The RUN Command

The RUN command is used to load and execute machine-code programs. Only two file types are permitted with the RUN command: .PRG and .LDA. The default file type is .PRG. To execute an .LDA file, the .LDA extension must be specified. The RUN command is of the form:

```
RUN [Device Name:] (File Spec)
```

For example:

```
RUN CHESS
```

loads and executes a file CHESS.PRG from the default device.

12.03.11 The SAVE Command

The SAVE command allows the user to save any type of data or program in a file on a Sof-Disk. The command is of the form:

```
SAVE [Device Name:] (File Spec) (Memory Spec) [Start Address
      [Actual Address]]
```

For example:

```
SAVE USERAM.XYZ 8120 1E00
```

or

```
SAVE USERAM.XYZ 8120-9F20
```

will save a file called USERAM.XYZ.

12.03.12 The WRITE Command

The WRITE command allows information to be written anywhere on the Sof-Disk without regard to the directory or previous program boundaries. It is of the form:

```
WRITE [Device Name:] (Start Block Number) (Memory Spec)
```

CAUTION: It is possible to destroy the FCS directory information using the WRITE command. Care should always be taken when using this command.

12.Ø4 External FCS Commands

This section describes and gives examples of the external FCS commands. Routines for execution of the external commands are recorded on Sof-Disk. They are not included in the software within 365Ø Series Intecolor itself. Therefore, use of the external commands requires the presence of a diskette with the appropriate .COM type file in the default disk drive. If there is no diskette in the default drive, or if the diskette in the default drive does not have the required .COM type file, the screen will respond with an error message EIVC (Invalid Command) when the external command is given.

(NOTES: External commands must be entered as they are listed in the directory of the .COM file diskette.

The external command names must not start with the same three letters in order as these letters appear in any internal command.

If the diskette with the .COM type file is in a disk drive other than the default drive, the device type must be entered prior to the external command.)

12.Ø4.Ø1 The DUPLICATE Command (DUP)

The DUPLICATE command, an external FCS command, allows all the files on one Sof-Disk, including the directory, in one disk drive to be copied to another Sof-Disk in another disk drive. The two specified devices must be of the same type, but have different numbers. The command is of the form:

DUP (Device Name:) TO (Device Name:)

For example:

DUP Ø: TO 1:

CAUTION: The DUP and the DUA commands, like the INI command, result in erasure and loss of all information previously on the destination disk. Before using any of these commands, be sure to COPY to another Sof-Disk any files you wish to retain from that disk.

12.Ø4.Ø2 The DUPLICATE ALL Command (DUA)

The DUPLICATE ALL command copies the entire contents, including the free space area, of one Sof-Disk in one disk drive to another diskette in another disk drive. (The free-space area might include, for example, data previously entered via a WRITE command.) The two specified devices must be of the same type, but have different numbers. The command is of the form:

DUA (Device Name:) TO (Device Name:)

For example:

DUA FD1: to FDØ:

CAUTION: The DUA command results is erasure and loss of all information previously on the destination disk. Before using this command, be sure to COPY to another Sof-Disk any files you may wish to retain.

The **BACKUP Command (BACKUP)** also allows transfer of files on one diskette to another diskette. It differs from the DUP and DUA commands in that the disk drives are not required to be of the same type and in that only the latest version of each file is transferred. For example:

```
FD0:BACKUP FD0: TO MD0:
```

where the default drive is MD0 and .COM file BACKUP resides in FD0.

12.04.03 The FILE Command (FIL)

The FILE command creates an empty file of a specified size on a Sof-Disk. It is similar to the FILE "N" statement in Extended Disk BASIC. The command is of the form:

```
FIL (File Spec) (# Records) [Record Size  Blocking Factor]
```

If the optional Record Size and Blocking Factor are not specified, then they default to 80 Hex and 1 respectively. For random access files in Extended Disk BASIC the parameters should match the desired size for the file. Note that all the parameters are hexadecimal numbers.

For example:

```
FIL NAMES.RND 64

FIL PARTS.IDX 200 20 8
```

In the first example a 100 record file NAMES.RND with 128 byte records is opened. The second command creates a file PARTS.IDX containing 512 32-byte records with a blocking factor of 8.

For further description of the optional parameters see Section 10.

12.04.04 The MERGE Command (MER)

The MERGE command is used to create a new file consisting of the contents of two or more existing files. The input files are concatenated sequentially in the order in which they are specified in the command. Each file specification may optionally specify a device and/or a file type. The default file type for the first file is .SRC. The default type for each subsequent file in the command is the type used for the previous file. The command is of the form:

```
MER (file1),(file2),...,[fileN] TO (New File Spec)
```

where the file1 through fileN are any valid file specifications.

For example:

```
MER PART1,PART2 TO FINAL

MER 0:SYS.LDA,1:DRV,0:GRAPH TO 1:DISPL
```

NOTE: BASIC programs should not be merged using the FCS MERGE command. However, there is a BASIC Merge program on the BASIC Utilities Disk.

12.04.05 The PRINT Command (PRI)

The PRINT command is used to print a file to the CRT, or to the printer if it is on. The default file type is .SRC; any other file type must be specified explicitly. The command is of the form:

PRI (File Spec)

For example:

PRI HELP.SRC

or

PRI FD1: HELP.SRC

where the .COM file PRI resides in the default drive and HELP in FD1.

12.04.06 The SAVE SCREEN Command (SVSCR)

The SAVE SCREEN command is used to save a copy of the screen display in a disk file. The default type is .PIC; any other file type must be specified explicitly. The command is of the form:

SVSCR (File Spec)

For example:

SVSCR BILD

saves a copy of the screen in a disk file named BILD.PIC.

12.04.07 The LOAD SCREEN Command (LDSCR)

The LOAD SCREEN command is used to put a display on the screen from a disk file previously created by a SVSCR command (see above). The default type is .PIC; any other file type must be specified explicitly. The command is of the form:

LDSCR (File Spec)

For example:

LDSCR SEITE.PRG

sends the data in a file named SEITE.PRG to the screen.

NOTE: Additional external commands may be added from time to time.

12.05 FCS Error Codes

Appendix C.2 lists the FCS Error Codes and includes a list of usual causes and possible solutions to problems.

13. OPERATING IN THE TERMINAL MODE

13.01 Terminal Modes

The 3650/9650 Series Intecolor units can be used as data communications terminals with the RS-232C interface. The terminal mode also can be useful for experimental work in graphics composition.

Operation of the CPU RESET key places the Intecolor unit in the CRT Mode (or terminal mode of operation). The unit is initialized with conditions as follows:

Visible foreground	Green
Visible background	Black
Reverse field flag	"0"
Visible A7 bit	"0" (for x1 characters)
Plot bit	"0"
Operation	Scroll mode
Terminal mode	Local
Baud rate	9600 with one stop bit
Write mode	Left to right, visible cursor

In the Local Mode of terminal operation, keyboard operations are displayed on the CRT screen. Commands entered via the keyboard have priority; however, the unit also will respond to binary code words input to the RS-232C serial port. For example, if the binary equivalents of ASCII decimal codes 65, 66 and 67 are received through the RS-232C serial input port, the screen will display the letters A, B and C in upper case. The appropriate code sequences can cause the Intecolor unit to enter the graphic submodes and display graphics patterns determined by the input through the RS-232C port. The screen display responds to either the RS-232C input or to the keyboard input, with the keyboard having priority.

The Half Duplex Mode of terminal operation is entered via an (ESC) H sequence (either by operation of the (ESC) and H keys or by input of decimal codes 27 and 8 to the RS-232C port). In this mode keyboard operations are sent to both the screen and the RS-232C serial output.

The Full Duplex Mode is entered with an (ESC) F sequence from the keyboard or by input of codes 27 and 6 to the RS-232C port. In full duplex all keyboard inputs are directed only to the RS-232C port. The results of keyboard operations will be seen on the screen only if echoed back by the computer or terminal at the other end. If control of the CRT display via the keyboard is desired, operation of the ATTN BREAK key places the unit in the Half Duplex Mode; operation of the CPU RESET key returns the unit to the Local Mode.

13.02 Serial I/O Port Connections

The terminal connections for the RS-232C port are listed in Appendix E.4.

The connections assume connection to a host computer or other terminal via modems. If direct wire connections are used (recommended cable length 50 feet maximum), the connecting cable likely must provide interchange of the

transmit and receive leads. The Intecolor unit's transmit output goes to the host computer's receive input and the Intecolor's receive input comes from the host's transmit output. (Note: This reversal of transmit and receive leads sometimes is necessary when the serial I/O port is used with certain peripheral devices — e.g., some types of serial printers.)

13.03 Parity

The standard 3650/9650 Series Intecolor units do not use parity. They will receive from any system of parity. The Intecolor unit transmits without parity.

In code received through the serial port, the most significant bit (A7) is used only in the Graphic Plot Mode. For a host transmitting in parity, use of this bit for graphics still is possible — the host can exit the Plot Mode temporarily to use code 14 to set the A7 bit or code 15 to reset it to zero. However, if the host at the other end of a communications loop is expecting to receive odd or even parity, approximately half of the codes 0 through 127 sent by the Intecolor will be considered invalid.

13.04 Binary Code Generation

When certain Control Codes are used -- e.g., Plot, Cursor XY, CCI -- following words must be binary codes. These binary code words can be generated by keystrokes or combinations of keystrokes. The table of Appendix F.3 shows how the binary equivalents of decimal codes zero through 255 are generated. (The 72-key and 101-key keyboards are coded for 0 through 191; the 117-key keyboard is coded for 0 through 255.) Appendix F.4 shows the keystroke(s) for decimal codes 0 through 255.

Following are a number of exercises illustrating the use of the keyboard for generation of binary codes when operating in the CRT Mode.

13.04.01 Cursor XY Positioning

The Cursor XY mode for positioning the visible cursor may be used to send the cursor to a point near screen center by these keyboard operations:

CPU RESET	To be in the CRT Mode
CONTROL C	To enter the Cursor XY Mode
SHIFT 0	To generate the binary code for character position 32
CONTROL P	To generate the binary word for line 16

The cursor moves to character position 32 of line 16. (In cursor XY positioning, the top line is designated Line 0; the leftmost character position is Character Position 0.)

13.04.02 The CCI Mode for Color Control

When the CCI Mode is entered via a CONTROL F sequence, the next byte provides the 8-bit visible status word specifying foreground color, background color, blink and plot. The lower order three bits represent foreground color, the next three bits background color, the A6 bit determines foreground blink, and A7 is a plot character bit which causes the display to interpret the ASCII word as a 2x4 plot array.

(128) A7	(64) A6	(32) A5	(16) A4	(8) A3	(4) A2	(2) A1	(1) A0
PLOT	BLINK	BACKGROUND			FOREGROUND		
		BLUE	GREEN	RED	BLUE	GREEN	RED

In the above table the figures in parentheses represent the decimal value of the binary digit. For example, a decimal code of 97 corresponds to the binary code

0 1 1 0 0 0 0 1

This code specifies red foreground, blue background and blink. In the CRT Mode the keyboard operations are:

CPU RESET	To be in the CRT Mode
CONTROL F	To enter the CCI Mode
SHIFT A	To generate the binary code corresponding to decimal 97

Subsequent typing will be displayed as blinking red characters against a blue background.

The effect is similar to the results of a BASIC plot statement of PLOT 6, 97. (Do a CPU RESET with both CONTROL and SHIFT held operated to get into BASIC. The screen displays READY in green against a black background. Then type PLOT 6,97 and hit the RETURN/ENTER key. The screen acknowledges execution of this command with READY — but now the READY is in blinking red characters against a blue background.)

13.04.03 Binary Code Generation in the Plot Mode

This section will include several examples of the use of the keyboard in the generation of graphics. First, a practice exercise in the generation of binary codes may be of help. A diagonal plotting exercise may be used to go through the tables of Appendices E.3 and E.4 step by step. (Note: This isn't the normal way to plot a diagonal — it's suggested here only to provide feedback of the results of the keyboard operations.)

In the XY Point Plot Modes the screen is treated as a grid 128 x 128. Point (Ø,Ø) is the lower left corner of the screen. Point (127,127) is the upper right corner. The X Point Plot Mode is entered at the same time the General Plot Mode is entered through CONTROL B. The next binary word is the X coordinate. Then the unit shifts automatically to the Y Point Plot Mode and the next binary word is the Y coordinate. The unit returns automatically to the X Point Plot Mode for another cycle. If the flag is on prior to entering the plot mode, a second plot of a given point will cause the point to be erased. Now proceed as follows:

CPU RESET	To initialize the CRT Mode
FLG ON	To set the reverse field flag on
CONTROL B	To enter the General Plot Mode (and the X Point Plot submode)
CONTROL NULL	To generate the code for zero (our X value)
CONTROL NULL	To generate the code for zero (our Y value)

A small block is plotted at the lower left corner of the screen. Now with the CONTROL key held down, strike the A key twice to generate the codes for position (1,1). Another small block appears just above and to the right of the first one. Continue in order through the alphabet, holding the CONTROL key down and striking each alpha key twice. With the CONTROL key still down, perform dual operations of the [\ ] ^ and _ keys (all in the same row as the HOME key). The plot blocks now form a diagonal a quarter of the way across the screen.

Next, with the SHIFT key held down (CONTROL is released), operate the Ø key twice and then follow with dual operations of keys 1 through 9 and * and +. The diagonal continues step by step. Release the SHIFT key and perform dual operations of the , - . and / keys, then the Ø and 1 through 9 keys and the : and ; keys. Then with SHIFT held operated, continue with the < = > and ? keys. The diagonal is now half way across the screen. The last point plotted was (63,63).

With both CONTROL and SHIFT released, operate the @ key twice. Continue with dual operations of A through Z, followed by the [\ ] ^ and _ keys. Next repeat the sequence of dual operations of keys @ and A through Z and [\ ] ^ and _ with just the SHIFT key held operated. The diagonal is now complete, with the last point at (127,127).

The same @ through _ sequence with both CONTROL and SHIFT held down will generate the codes for 128 through 159. When plot of coordinates above 127 is attempted, the machine automatically subtracts 128 from the value. Therefore, when the @ key is struck twice with both CONTROL and SHIFT held down, point (128,128) is plotted as (Ø,Ø). This second plot of (Ø,Ø) results in erasure of the lower left corner block. Continuing the sequence of A through _ (both CONTROL and SHIFT down) erases the first quarter of the diagonal. CONTROL SHIFT Ø and 1 through 9 and * and + continues the erasure in order. Release SHIFT and continue with CONTROL Ø and 1 through 9 and : and ;. With both CONTROL and SHIFT down, continue with < = > and ?. Half of the diagonal has been erased. If the keyboard does not have the 16 special function keys, the operation of ? with both CONTROL and SHIFT resulted in exit from the Plot Mode. The unit will now respond to operation of the ERASE PAGE key to erase the screen to the background color.

If the keyboard does have the 16 function keys, the exercise may be continued. With only CONTROL held down, continue with dual operations of the function keys FØ through F15 (skip the up arrow key). Repeat FØ through F15 with only the SHIFT key held down. Then go through FØ through F15 again with both CONTROL and SHIFT held down. That's as far as this exercise goes. A small portion of the diagonal remains, but we can't use codes 24Ø through 255 to continue our erasure because these particular codes result in other plot submodes. So just strike the F15 key alone to escape from the Plot Mode or operate CPU RESET.

13.Ø5 Graphics in the CRT Mode

Section 11 of this manual describes the several Plot Submodes. Examples are given for use of the codes involved when programming in BASIC. The format examples show the codes used via the RS-232C port. In the CRT Mode, the decimal equivalent of the binary codes required may be generated from the keyboard. Following are a number of examples illustrating the keyboard operations for generating graphics.

The keystrokes used to enter the plot submodes depend upon the type of keyboard present in the system. With the 117-key keyboard, the special function keys FØ-F15 are used. When the function keys are not present, the combinations of CONTROL Ø through 9 and the symbols : ; < = > and ? are used. The last four symbols require SHIFT as well. Thus with the function keys, one-key entry to the plot submodes is possible; without the function keys, two-key or three-key combinations must be used. The examples show both methods, along with the decimal value of the binary code used via the RS-232C port.

Plot Mode Escape — Code 255

F15 or the combination of CONTROL SHIFT ?.

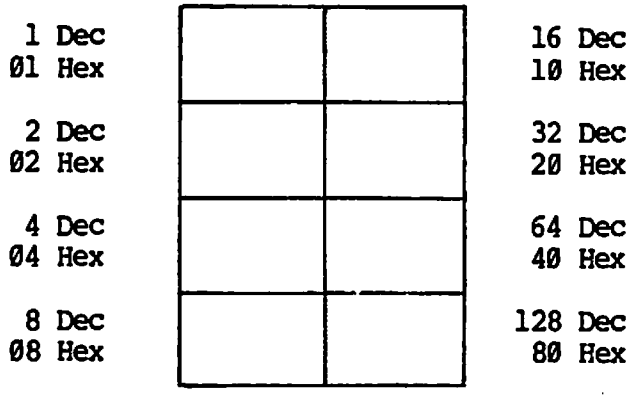
Character Plot — Code 254

The Character Plot submode is entered by code 254 after the general plot mode is entered via code 2 or CONTROL B. CONTROL > (which requires the SHIFT key to be held down also) or F14 is used at the keyboard.

After Character Plot is entered, the next word is treated as a plot character except for code 255 binary (all eight bits 1's). Receipt of code 255 terminates the plot mode.

Other plot submodes may not be entered from the Character Plot submode. To enter other plot submodes it is necessary to terminate the plot mode, re-enter the general plot mode and enter the desired plot submode.

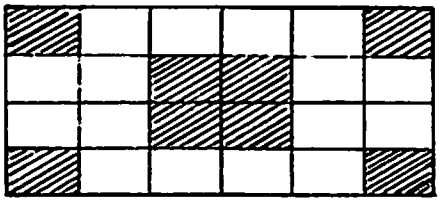
Plot characters are defined by codes Ø through 254. The sketch on the next page shows the eight individual blocks within the character matrix to be defined by decimal codes 1, 2, 4, 8, 16, 32, 64 and 128.



The codes may be ORed to define more than one block within a character matrix — e.g., code 9 (generated by TAB or CONTROL I) will cause plotting of the upper left and lower left plot blocks in the following illustration. (The terminal is assumed to be in the CRT Mode after CPU RESET. The cursor controls and TAB may be used to position the cursor nearer screen center if desired.)

Function	By Code Via RS-232C	By Keyboard Operations
Entrance into general plot mode	2	CONTROL B
Enter Character Plot submode	254	CONTROL SHIFT > or F14
Define first character	9	CONTROL I (or TAB)
Define next character	102	SHIFT F
Define third character	144	CONTROL SHIFT P
Exit plot mode	255	CONTROL SHIFT ? or F15

The plot should appear as shown below.



The A7 Bit should be off when in the Character Plot mode or it will cause the lower right plot block to be intensified along with the selected block. Intensification of all eight blocks by entry of one code is not possible — this would require code 255, which results in exit from the plot mode. However, this result can be achieved in other ways, such as entry of a character space in an appropriate background color.

X Point Plot — Code 253
Y Point Plot — Code 252

The X Point Plot submode is entered automatically upon receipt of the general plot mode code 2 or CONTROL B. It also may be entered directly from any other plot submode except Character Plot via code 253 through the serial input port or generated by the keyboard combination CONTROL = (SHIFT also held down) or F13. After X Point Plot is entered the next word defines the X value of the block to be plotted. The X value may be in the range of 0 to 127 -- a larger value will have 128 subtracted to obtain a valid X value. Refer to Appendix F.4 for the keyboard operations used to obtain these codes.

Entry of the X value does not cause the block to be intensified — only the X coordinate has been defined. Instead the terminal automatically goes to the Y Point Plot submode and awaits the Y value. Upon receipt of the Y value the now defined plot block is intensified on the screen. The terminal automatically reverts to the X Point Plot submode and the next word will be treated as an X value. Therefore, once in the X Point Plot submode, new blocks may be defined simply by sending X values and Y values consecutively. There is no need to send codes for re-entry of the X or the Y Point Plot submodes.

The following sequence will intensify plot blocks near the four corners of the display:

Function	By Code Via RS-232C	By Keyboard Operations
Entrance into general plot mode and X Point Plot submode	2	CONTROL B
Define lower left corner block	0 0	CONTROL NULL CONTROL NULL
Define lower right corner block	127 0	SHIFT _ (key with CRT) CONTROL NULL
Define upper right block	127 127	SHIFT _ SHIFT _
Define upper left block	0 127	CONTROL NULL SHIFT _
Exit plot mode	255	CONTROL SHIFT ? or F15

The X Point Plot in conjuction with the Y Point Plot allows any block on the 128 by 128 grid to be positioned and intensified. If the new block is within a character position that contains a previously intensified ASCII character, that ASCII character will be completely replaced by the new block and its associated color. If a new block is within a character position that has a previously intensified plot block, the previously intensified plot block will take on the color of the new plot block.

The X Point Plot submode may be terminated by code 255 (which causes the general plot mode to be terminated as well), or by keystrokes F15 or CONTROL SHIFT ?. Any of the other plot submodes may be entered directly from the X Point Plot submode simply by entry of the appropriate code in the range of 240 to 254.

The Y Point Plot submode normally is entered automatically from the X Point Plot submode, as in the example above. However, it may be entered directly by code 252 after the general plot mode has been entered or via a keyboard CONTROL SHIFT < or F12. Following entry into Y Point Plot, the next word defines the Y value of the block to be plotted. The terminal uses whatever previous X value may be in memory to complete definition of the block to be plotted and the block is intensified. The Y value may be in the range of 0 to 127 — larger values will cause subtraction of 128 for calculation of a valid Y value.

Upon receipt of the Y value, the terminal automatically enters the X Point Plot submode. The X value of the next block to be plotted may then be entered as described earlier.

The Y Point Plot submode is terminated by code 255 (which causes the general plot mode to be terminated also), by keystrokes F15 or CONTROL SHIFT ?. Any of the other plot submodes may be entered directly from Y Point Plot simply by entry of the appropriate code in the range of 240 to 254 or the keystrokes which result in generation of these codes.

XY Incremental Point Plot — Code 251

The XY Incremental Point Plot submode is entered by code 251 after entry into the general plot mode — or via a keyboard CONTROL ; or F11. It may be entered directly from any of the other plot submodes except Character Plot. After entry into XY Incremental Point Plot, the next word defines the next two increments as shown in the table below. This word may be in the range of 0 to 239 binary — binary 240 through 255 codes are used for plot submodes.

128 B7	64 B6	32 B5	16 B4	8 B3	4 B2	2 B1	1 B0
$\Delta X1$		$\Delta Y1$		$\Delta X2$		$\Delta Y2$	
Plot Block 1				Plot Block 2			

B(n+1)	B(n)	
0	0	No change
1	0	Positive increment
0	1	Negative increment
1	1	No change

Where n = 0, 2, 4, 6

For example, creation of horizontal line that zig-zags up and down as it crosses the page requires that all X increments be positive (to keep the points moving to the right) and that the Y increments alternate between positive and negative (to move the points up and then down). After one block is plotted as a starting point and then the XY Incremental Point Plot Mode is entered, the four bits 1010 define the location of the next block as one block to the right and one block up. Four similar bits define the next point to be located an additional block right and up. The binary code word

1 0 1 0 1 0 1 0

(decimal value 128 + 32 + 8 + 2 = 170) causes two blocks to be plotted to the right and upward from the starting point.

In Appendix F.3, looking at the bit patterns at the top of each column on the chart, column A (the 10th column) has the required pattern for the first four bits. Then, looking at the bit patterns at the left of each line, line A has the required pattern for the second four bits. The intersection of line and column shows that operation of the "*" key with both CONTROL and SHIFT held down will generate the desired code.

The binary code word

1 0 0 1 1 0 0 1

(decimal value 128 + 16 + 8 + 1 = 153) causes two blocks to be plotted to the right and downward from the last point plotted. Reference to Appendix F.3 indicates that operation of the "Y" key with both CONTROL and SHIFT held down will generate this code.

To create the zigzag line, then, the code entries through the serial port or the keyboard operations are:

Function	By Code Via RS-232C	By Keyboard Operations
Enter general plot mode	2	CONTROL B
Define starting point at left center	0 64	CONTROL NULL @ (the NULL key)
Enter XY Incremental Point Plot submode	251	CONTROL ; or F11
Increment up and right (for two blocks)	170	CONTROL SHIFT *
Increment down and right (for two blocks)	153	CONTROL SHIFT Y

Continue alternating the last two operations to continue the pattern across the page.

The B0-B3 bits do not have to duplicate the B4-B7 bits. The binary code word

1 0 1 0 1 0 0 1

(decimal value 128 + 32 + 8 + 1 = 169) locates the first additional block right and up and the second additional block right and down. Operate ERASE PAGE and repeat the above example with code 169, keyboard operation CONTROL SHIFT ")", instead of codes 170 and 153.

If only one incremental block (rather than two) is desired, the B4-B7 bits are made zero and the B0-B3 bits are used for the increment. The binary code word

0 0 0 0 1 0 0 1

(decimal value 8 + 1 = 9) locates one additional block right and down.

If "skipping" of a block is desired, bits B0-B3 are made zero and B4-B7 are used for the increment. The binary code word

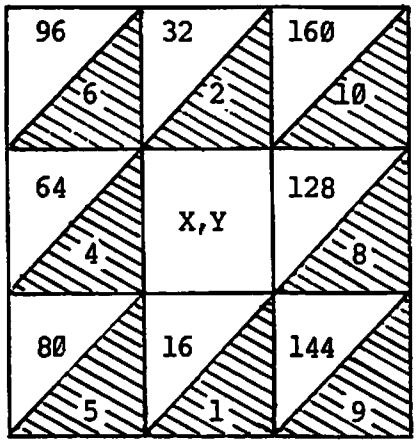
0 1 1 0 0 0 0 0

(decimal value 64 + 32 = 96) will increment left and up to plot an invisible block — i.e., leave the new block in the background color. For example, a broken line left and upward from a starting point may be created as follows:

Function	By Code Via RS-232C	By Keyboard Operations
Enter General Plot Mode	2	CONTROL B
Define starting point in lower right quadrant	100 16	SHIFT D CONTROL P
Enter XY Incremental Point Plot submode	251	CONTROL ; or F11
Increment left and up to plot one block	6	CONTROL F
Increment left and up but not plot	96	SHIFT NULL
Increment left and up and plot two blocks	102	SHIFT F
Increment left and up but not plot	96	SHIFT NULL
Increment left and up and plot two blocks	102	SHIFT F

Alternate codes 96 and 102 to continue the broken line. Use code 255 (keyboard CONTROL SHIFT ? or F15) to exit the plot mode.

The sketch below shows the codes for one-block increments. The code in the shaded area causes the block to be intensified. The code in the unshaded area results in an increment to the block but no plot ("invisible" block).



The XY Incremental Point Plot submode does not automatically transfer the terminal to any other plot submode upon receipt of an incremental change word. It remains in the XY Incremental Point Plot submode ready to receive another incremental change word. Thus a series of incremental movements may be made by sending consecutive incremental change words with or without points being plotted.

The XY Incremental Point Plot submode may be terminated by code 255 (keyboard F15 or CONTROL SHIFT ?), which also causes the general plot mode to be terminated. Any of the other plot submodes may be entered directly from XY Incremental Point Plot simply by entry of the appropriate code in the range of 240 to 254.

X Bar Graph, XØ Value — Code 250
X Bar Graph, Y Value — Code 249
X Bar Graph, X Max Value — Code 248

After entry into the general plot mode, the X Bar Graph, XØ Value submode is entered by a binary 250 code or via a keyboard CONTROL : or F10. It may be entered from any of the other plot submodes except Character Plot. Following entry into the X Bar Graph, XØ Value submode, the next word defines the XØ Value, the left horizontal start block of the horizontal bar graph. The graph grid is referenced to the lower left corner of the screen. The XØ Value may be in the range of Ø to 127 — larger values will have 128 subtracted for calculation of a valid position.

Following receipt of the XØ Value, this value of XØ is stored in memory and the terminal is placed automatically in the X Bar Graph, Y Value submode. It now awaits the next word to define the Y position of the bar graph. Upon receipt of the Y Value, the terminal then is placed automatically in the X Bar Graph, X Max Value mode and awaits the next word to define the X Max Value. Upon receipt of the X Max Value, the bar is drawn on the screen. The terminal is placed automatically back in the X Bar Graph, Y Value submode, ready to receive a new Y value for another bar.

As can be seen from the above process, once in the X Bar Graph, XØ Value submode and an XØ Value received, only two words (Y and X Max) are required to define bar graphs with the same XØ horizontally. After the first bar graph sequence, additional bar graphs can be described simply by entry of the new Y position and the X Max Value. Each bar is drawn, using the original XØ Value, upon receipt of the X Max Value.

This mode allows bar graphs in any of the colors to be drawn with a width as small as one plot block. Areas under curves may be filled in easily. If multiple color bars are to be drawn, the bars should be character multiples in height — only one foreground color and one background color per character position may be used.

Any of the three X Bar Graph submodes may be entered independently and any of the other plot submodes may be entered directly by entry of the appropriate binary code in the range of 240 to 254. The submode and the general plot mode are terminated on receipt of code 255.

As an example, to plot a bar across screen center, starting a quarter of the way in from the left and continuing to a point a quarter of the way in from the right, the sequence may be as follows:

Function	By Code Via RS-232C	By Keyboard Operations
Enter general plot mode	2	CONTROL B
Enter X Bar Graph, XØ	250	CONTROL : (colon) or F10
Define XØ as 31	31	BLINK ON (or CONTROL _)
Define Y as 63	63	SHIFT ?
Define X Max as 96	96	SHIFT NULL
Exit plot mode	255	CONTROL SHIFT ? or F15

Prior to exit from the plot mode, the bar could have been made double height (or "heavier") by additional codes 64 (@) and 96 (SHIFT @).

The X Bar Graph, Y Value submode is more commonly entered automatically from the X Bar Graph, XØ Value submode or the X Bar Graph, X Max Value submode, as outlined above. However, it may be entered by a binary code 249 or via a keyboard CONTROL "9" or F9, if already in the general plot mode. After entry into the X Bar Graph, Y Value submode, the next word defines the Y Value, the vertical position of the horizontal bar graph being drawn. This Y Value may be in the range of Ø to 127 — larger values result in subtraction of 128 for calculation of a valid Y Value.

Upon receipt of the Y Value, this value of Y is stored in memory and the terminal is placed automatically in the X Bar Graph, X Max Value submode.

Any of the other plot submodes may be entered directly from X Bar Graph, Y Value by entry of the appropriate code in the range of 240 to 254. Receipt of code 255 terminates the submode and the general plot mode.

The X Bar Graph, X Max Value submode is more commonly entered automatically from the X Bar Graph, Y Value submode. However, it may be entered by code 248 or via a keyboad CONTROL "8" or F8, if already in the general plot mode. It may be entered from any other plot submode except Character Plot. Following entry into the X Bar Graph, X Max Value submode, the next word defines the X Max horizontal point of the horizontal bar graph being drawn. This value may be in the range of 0 to 127 — larger values will result in subtraction of 128 for calculation of a valid X Max Value.

Upon receipt of the X Max Value the bar graph is drawn on the screen in the predefined color in accordance with the X0 and Y values stored in memory from previous operations. The terminal then is placed automatically in the X Bar Graph, Y Value submode.

Any of the other plot submodes may be entered directly from X Bar Graph, X Max Value by entry of the appropriate code in the range of 240 to 254. Code 255 terminates the submode and the general plot mode.

X Incremental Bar Graph — Code 247

After entry into the general plot mode, X Incremental Bar Graph is entered by a binary 246 code or via a keyboard CONTROL "7" OR F7. It may be entered from any of the other plot submodes except Character Plot. Following entry into the X Incremental Bar Graph submode, the next word defines the next two horizontal and vertical increments for two horizontal bar graphs. In this manner a bar graph may be positioned on either side of the present location with an increment added to or subtracted from the original previously defined. The coding of the word is similar to that for the XY Incremental Point Plot submode (code 251).

For example, going back to the bar graph example on the previous page, the X Incremental Bar Graph Mode could be used to make the bar graph several blocks high. Using the same initial operations, but prior to exit from plot via code 255:

Function	By Code Via RS-232C	By Keyboard Operations
To enter X Incremental Bar Graph Mode	247	CONTROL 7 or F7
To increment downward twice, X values unchanged	17	CONTROL Q
To repeat above	17	CONTROL Q
To exit plot mode	255	CONTROL SHIFT ? or F15

Rather than making a "heavier" bar graph line, a shaded area could be created by using the plot skip technique in the X Incremental Bar Graph Mode. Again, starting as in the example on the previous page:

Enter X Incremental	247	CONTROL 7 or F7
Increment downward without plotting	16	CONTROL P
Repeat X bar line after skip	1	CONTROL A
Increment downward without plotting	16	CONTROL P
Repeat bar	1	CONTROL A
Exit plot mode	255	CONTROL SHIFT ? or F15

Other methods can be used for large shaded areas, but this example serves to illustrate the use of the X Incremental Bar Graph Mode.

Y Bar Graph, YØ Value —— Code 246 binary
Y Bar Graph, X Value —— Code 245 binary
Y Bar Graph, Y Max Value - Code 244 binary
Y Incremental Bar Graph — Code 243 binary

The above four submodes operate in much the same manner as their corresponding X Bar Graph submodes. After entry into the general plot mode, any of them may be entered by receipt of the appropriate code or via a keyboard CONTROL "6", "5", "4" or "3" or F6, F5, F4 or F3 respectively.

Vector Mode, XØ Value — Code 242
Vector Mode, YØ Value — Code 241

After entry into the general plot mode, the Vector Mode, XØ Value is entered by a binary 242 code or via a keyboard CONTROL "2" or F2. It may be entered from any other plot submode except Character Plot. Following entry into the Vector Mode, XØ Value, the next word defines the XØ Value point of the vector being drawn. This XØ Value must be in the range of Ø to 127 (larger values will have 128 subtracted).

The Vector Mode requires definition of two end points — XØ,YØ and X1,Y1. The X1,Y1 point should have been defined previously by way of the X and Y Point Plot submodes (codes 253 and 252).

Upon receipt of the XØ Value the terminal is placed automatically in the Vector Mode, YØ Value (code 241) and awaits receipt of the YØ Value. Upon receipt of the YØ Value, the terminal then determines the best straight line fit between XØ,YØ and X1,Y1 using plot blocks. The vector is drawn in the predefined color. The terminal then reverts to the Vector Mode, XØ Value, ready to receive a new XØ Value for another vector. Unless a new X1,Y1 point is defined by way of the X and Y Point Plot routine, this new vector will use the last XØ,YØ point as its X1,Y1.

Code 255 terminates the submode and the general plot mode. Other plot submodes may be entered by use of the appropriate code.

As an example, the following sequence generates a diagonal from screen upper left to lower right:

Function	By Code Via RS-232C	By Keyboard Operation
Enter general plot mode (and X Point Plot)	2	CONTROL B
Define X1,Y1 as 0,127	0	CONTROL NULL
	127	SHIFT _ (underline)
Enter Vector Mode, X0	242	CONTROL 2 or F2
Define X0,Y0 as 127,0	127	SHIFT _
	0	CONTROL NULL

To continue a vector up the right side of the screen and back across the top to the upper left corner before exiting the plot mode, the sequence following 127,0 would be:

Define new X0,Y0 as 127,127	127	SHIFT _
	127	SHIFT _
Define new X0,Y0 as 0,127	0	CONTROL NULL
	127	SHIFT _
Exit plot mode	255	CONTROL SHIFT ? or F15

The Vector Mode, Y0 Value is more commonly entered automatically from Vector Mode, X0 Value, as just described. However, it may be entered by a binary 241 code or via a keyboard CONTROL "1" or F1, if already in the general plot mode. It may be entered directly from any other plot submode except Character Plot. Following entry into the Vector Mode, Y0 Value submode, the next word defines the Y0 Value of the vector being drawn. This Y0 value must be in the range of 0 to 127 (larger values will have 128 subtracted). Following receipt of the Y0 Value, the vector is drawn and the terminal is placed automatically in the Vector Mode, X0 Value submode.

Code 255 terminates the submode and the general plot mode. Other plot submodes may be entered directly from Vector Mode, Y0 Value by entry of the appropriate code.

Incremental Vector Mode — Code 240

After entry into the general plot mode, the Incremental Vector Mode is entered by a binary 240 code or via a keyboard CONTROL "0" or F0. It may be entered also from any other plot submode except Character Plot. However, when entered from any submode other than the Vector Mode the results may be somewhat difficult to predict.

Following entry into the Incremental Vector Mode, the next word defines the increments in X0, Y0, X1 and Y1 point values for the vector from X1,Y1 to X0,Y0. This word may be in the range of 0 to 239. (Codes 240 through 255 are used for plot submodes.)

As in the XY Incremental Point Plot submode, the word has two bits available for each of the four point coordinates (X0, Y0, X1 and Y1). In this case a "1" in the bit location defines the incremental change as shown in the table following:

B7	B6	B5	B4	B3	B2	B1	B0
X1+1	X1-1	Y1+1	Y1-1	X0+1	X0-1	Y0+1	Y0-1

If both bits describing a point are 1 or 0, no increment will take place — e.g., if both B5 and B4 are 1 or both are 0, then there will be no increment for the Y1 point.

A vector will be drawn only when both halves (B0 through B3 and B4 through B7) are non-zero. If either half of the word is all zero, then the X,Y corresponding to the other half will be changed but no vector drawn. This enables "skip" of points.

The Incremental Vector Mode does not transfer control automatically to any other mode. It remains in this incremental mode until terminated by a plot submode code. Therefore, a series of incremental movements in both X0,Y0 and X1,Y1 may be made by sending consecutive incremental change words.

Any other plot submode may be entered directly from the Incremental Vector Mode by entry of the appropriate code. Code 255 terminates both the submode and the general plot mode.

As one example, the Incremental Vector Mode may be used to create a solid triangular area:

Function	By Code Via RS-232C	By Keyboard Operations
Enter Plot Mode	2	CONTROL B
Define X1,Y1 as 65,65	65 65	A (upper case letter) A
Enter Vector Mode, X0	242	CONTROL 2 or F2
Define X0,Y0 as 90,90	90 90	Z Z

(vector drawn from 65,65 to 90,90)

Enter Incremental Vector Mode	240	CONTROL Ø (zero) or FØ
Increment X1, decrement YØ and plot new vector	129	CONTROL SHIFT A

(the binary code word is 1 Ø Ø Ø Ø Ø Ø 1 which redefines the vector end points as 66,65 and 9Ø,89 and plots a new vector between these points)

Continue process until triangle is complete	129	CONTROL SHIFT A twenty-five more times

(additional vectors are plotted, the final "vector" being a one-block plot)

Exit plot mode	255	CONTROL ? or F15

As another example the following, which creates a small diagonally shaded rectangular area, uses a number of the features of the Incremental Vector Mode, including the skip feature. The area is begun with one point. Then a second point three blocks higher is defined. Next the Vector Mode is entered and a vector plotted. The Incremental Vector Mode then is entered and the original vector end points moved (first one end, then the other) without plotting. Then both end points are moved and a new vector plotted.

Function	By Code Via RS-232C	By Keyboard Operations
Enter general plot mode	2	CONTROL B
Define the lower right corner of rectangle	8Ø 8Ø	P (upper case letter) P

(Plot block intensified at point 8Ø,8Ø)

Define one end point for start of a vector	8Ø 83	P S

(Plot block intensified at point 8Ø,83)

Enter Vector mode, XØ	242	CONTROL 2 or F2
Define XØ,YØ as 77,8Ø	77 8Ø	M (upper case letter) P

(Vector drawn from 8Ø,83 to 77,8Ø)

Enter Incremental Vector Mode	240	CONTROL Ø (zero) or FØ
Increment X1,Y1 upward twice without plotting	32 32	SHIFT Ø (zero) SHIFT Ø

Decrement X0,Y0 left	4	CONTROL D
twice without plotting	4	CONTROL D

(The binary code words for these two movements are 0 0 1 0 0 0 0 0 and 0 0 0 0 0 1 0 0. When either B0-B3 or B4-B7 are all zero, there is point movement but no plot.)

Increment Y1, decrement X0 and plot	36	SHIFT 4

(The binary code word 0 0 1 0 0 1 0 0 moves both end points and causes a new vector to be drawn from 80,86 to 74,80.)

Increment without plot	32,32	SHIFT 0 two times
Decrement without plot	4, 4	CONTROL D two times
Increment, decrement, plot	36	SHIFT 4

(Vector drawn from 80,89 to 71,80.)

Increment without plot	32,32	SHIFT 0 two times
Decrement without plot	4, 4	CONTROL D two times
Increment, decrement, plot	36	SHIFT 4

(Vector drawn from 80,92 to 68,80.)

Decrement X1,Y1 left	64	@
twice without plotting	64	@

(Here the binary code is 0 1 0 0 0 0 0 0, which decrements X1, leaves Y1 unchanged and does not plot — bits B0-B3 are all zero.)

Decrement X0,Y0 left	4	CONTROL D
twice without plotting	4	CONTROL D
Decrement both and plot	68	D

(The binary code word 0 1 0 0 0 1 0 0 leaves the Y values unchanged, decrements both X1 and X0 and plots a new vector from 77,92 to 65,80.)

Decrement without plot	64,64	@ two times
Decrement without plot	4, 4	CONTROL D two times
Decrement both and plot	68	D

(Vector drawn from 74,92 to 62,80.)

Decrement X1,Y1 left	64	@
twice without plotting	64	@
Increment XØ,YØ upward	2	CONTROL B
twice without plotting	2	CONTROL B
Decrement, increment, plot	66	B

(Vector drawn from 71,92 to 62,83.)

Decrement without plot	64,64	@ two times
Increment without plot	2, 2	CONTROL B two times
Decrement, Increment, plot	66	B

(Vector drawn from 68,92 to 62,86.)

Decrement without plot	64,64	@ two times
Increment without plot	2, 2	CONTROL B two times
Decrement, increment, plot	66	B

(Vector drawn from 65,92 to 62,89.)

Decrement without plot	64,64	@ two times
Increment without plot	2, 2	CONTROL B two times
Decrement, increment, plot	66	B

("Vector" drawn from 62,92 to 62,92 — one block.)

Exit plot mode	255	CONTROL SHIFT ? or F15

The result should be a series of diagonal lines defining a rectangular area.

The examples just given are intended solely to illustrate the effects possible with the graphics features of the Intecolor 3650 Series. The somewhat tedious step-by-step approach of some of them is included only to demonstrate the effects in simple form. In actual practice, BASIC programs with FOR ... NEXT statements or a mathematical formula or some other form of program -- with the user's skill and imagination almost the only limiting factors — are much the better way to go. The possibilities are nearly endless.

14. FILE TRANSFER FROM OTHER ISC SYSTEMS

Users of other types of ISC systems may have substantial files which they wish to use in the 3650/9650 Series system. In most cases a transfer of the data to a diskette for use in the 3650/9650 Series unit is possible. The local Intecolor dealer or representative should be consulted regarding the procedure to be used in a particular situation.

14.01 Intecolor 3621 and Compucolor II Files

A backup program (CDBKUP.COM), with user documentation, is available for transfer of the latest revision of each program on a Micro-Disk recorded diskette to a diskette for use in one of the disk drives supported by the 3650/9650 software. The procedure requires use of an external 4-phase Micro-Disk drive, a Sof-Disk with the program for formatting diskettes for use in the 3650/9650 Series and a Sof-Disk with the CDBKUP.COM program. Instructions included with the CDBKUP.COM diskette outline the procedure. Please consult your Intecolor representative.

14.02 Intecolqr 8000 Series Files

Files on diskettes recorded by the standard disk drives supported by the Intecolor 8000 Series software can, in most cases, be read, copied or duped in a 3650/9650 Series unit having disk drives of a similar type. The reverse is not the case — e.g., diskettes recorded in the 3650/9650 Series 5.25" drives very likely will not be usable in an 8000 Series system because of the different format. The two File Control Systems do have certain differences (e.g., different size directories, use of .COM file commands in the 3650/9650, etc.). Therefore caution must be observed when attempting to use 8000-Series recorded diskettes in a 3650/9650 Series system. Again, consultation with the local Intecolor representative is recommended.

APPENDICES

APPENDIX A
3650/9650 SERIES INTECOLOR SPECIFICATIONS

Intecolor units in the 3650/9650 Series are intended for use on a desk, table or similar vibration-free horizontal surface that is free from dust and lint. They may be installed adjacent to most types of electrical and electronic equipment, provided that it is not located within a strong magnetic field. (Operation within a strong magnetic field may affect the quality of the display.) Although no special cooling provisions need be made for the microcomputer or disk drives, there must be free flow of air around the console and vents in the bottom of the cabinet must not be blocked; vents at the rear and sides of the external disk drives also must be free. The units may be used under normal room lighting conditions, but direct sunlight and excessively bright room lighting should be avoided. Power outlets (115 VAC, 60 Hz standard) must be within reach of the 6 to 7-foot 3-wire power cords of the console and the external disk drives.

Operating Conditions

Power	105-125 volts, 50/60 Hz., 100 watts. (Optionally 210-250 VAC)
Temperature	+10°C to +40°C operating -30°C to +70°C for storage only
Humidity	0 to 95% non-condensing

Physical Dimensions

	3650 Series	9650 Series
Console	13 3/4" high 19 3/4" wide 26 5/8" deep	17 1/2" high 26 1/8" wide 23 1/2" deep
	51 pounds	70 pounds
Keyboard	Built-in	25 3/4" wide 8 3/4" deep 3" high
		5.5 pounds
Screen size	13" diag. CRT 90 sq. in.	19" diag. CRT 186 sq. in.

Display

Dimensions	9.5" x 7.25" 10" x 13"
Colors	Eight foreground and eight background (red, blue, green, yellow, magenta, cyan, black and white).
Format	64 characters per line, 32 lines per page (or 16 lines with 2X character height).
Characters	64 ASCII characters, 5 x 7 dot matrix within a 6 x 8 dot pattern, plus 64 special characters in a 6 x 8 matrix. 128 x 128 graphics and vector generating software.
Cursor	White blinking underscore.
Refresh rate	60 times per second (optionally 50).

Keyboard

Type	Gold crossbar commercial key switches. 72 keys standard. Coded with 192 codes. Optionally 101 keys or 117 keys (latter coded with 256 codes).
Features	Cursor controls, CPU Reset, Automatic Disk Loading. 101-key keyboard includes 16-key number/mathematics pad and 9-key color/command pad. 117-key keyboard has further addition of 16 plot function keys.

Microcomputer

CPU	8080 2-microsecond CPU, total address capability 64K.
Memory	16K bytes of non-destructive read only memory. Optional add-on board has four sockets for additional ROM (up to 8K).
	4K bytes of random access memory for CRT screen refresh.
	4K bytes of random access memory for disk buffer.
	16K bytes of user random access memory standard. Optionally 32K.

I/O Ports

One RS-232C serial asynchronous channel for a modem or printer. Disk drive I/O capable of supporting up to three 5.25" Mini-Disk Drives and four 8" Floppy Disk Drives, single or double headed. Other ports available for user implementation via the X-Bus or 50-pin Extension Bus.

Baud rate is keyboard selectable for one of seven rates, ranging from 110 baud to 9600 baud. One or two stop bits. No parity.

Editing

Editing features include Page Roll Mode, Erase Line, Erase Page, Tab, Caps Lock, CPU Reset and Color Selection.

Language

Extended Disk BASIC interpreter in ROM includes 27 statement types: CLEAR, DATA, DEF, DIM, END, FILE, FOR, GET, GOSUB, GOTO, IF, INPUT, NEXT, ON, OUT, PLOT, POKE, PRINT, PUT, READ, REM, RETURN, RESTORE, STEP, THEN, TO and WAIT.

Five command types: CONT, LIST, LOAD, RUN and SAVE.

There are 18 mathematical functions: ABS(x), ATN(x), CALL(x), COS(x), EXP(x), FRE(x), INT(x), INP(x), LOG(x), PEEK(x), POS(x), RND(x), SGN(x), SIN(x), SPC(x), SQR(x), TAB(x) and TAN(x).

Nine string functions: ASC(x$), CHR$(x), FRE(x$), LEFT$(x$,I), LEN(x$), MID$(x$,I,J), RIGHT$(x$,I), STR$(x) and VAL(x$).

File Control

Internal Disk File commands are: COPY, DELETE, DEVICE, DIRECTORY, EXECUTE, INITIALIZE, LOAD, READ, RENAME, RUN, SAVE and WRITE.

External Disk File commands include: BACKUP, DUPlicate, DUplicate ALL (DUA), FILe, MERge, PRInt, Save SCReen (SVSCR) and Load SCReen (LDSCR).

Disk Drives

Medium

5.25" Sof-disk using 40 tracks, 18 sectors.
8.00" Sof-disk using 77 tracks, 30 sectors.

Capacity

Accessible with disk(s) in drive(s),

Single Mini-Disk Drive	— 92,160 bytes
Dual Mini-Disk Drive	— 184,320 bytes
Dual 8" Floppy Disk	-- 295,680 bytes
Dual Double Headed 8"	— 591,360 bytes

Data Transfer	125 Kilobits per second in 5.25" drives and 250 Kilobits per second in 8" drives. The 5.25" drive requires approximately one second for speed stabilization.
3651 Intecolor	One built-in 5.25" single headed disk drive
3652 Intecolor	One built-in and one external 5.25" single headed disk drives. The external drive has a separate power cord, is 3 3/4"H x 6"W x 12 1/2"D, 8 pounds.
3653 Intecolor	External dual 8" single headed disk drives. 5 1/2"H x 21"W x 23 1/2"D. 48 pounds. Separate power cord.
3654 Intecolor	Same as 3653 Intecolor except drives are double headed.
Options	All units in series support up to three 5.25" disk drives and four 8" disk drives.

APPENDIX B. DISK BASIC

B.1 BASIC Statements

The following summary of BASIC statements defines the general format for each statement and gives a brief explanation. Optional items are enclosed in angle brackets, '<' and '>'. The following items in the syntax descriptions are used to represent different types of variables and expressions:

var	- numeric or string variable
nvar	- numeric variable
svar	- string varaible
expr	- numeric or string expression
nexpr	- numeric expression
sexpr	- string expression

Statement	Syntax and Description
CLEAR	CLEAR <nexpr> Clears all variables and optionally sets the string space size to nexpr bytes.
CONT	CONT Continues execution after CTRL/J or ↓ (LINEFEED).
DATA	DATA value list Defines data values to be read using the READ statement.
DEF	DEF FN nvar (nvar) = nexpr Defines a user function to be used in the program.
DIM	DIM var(nexpr <,...,nexpr>) <,...> Reserves space for lists and tables according to subscripts specified after variable name. Up to 255 dimensions.
END	END Terminates program execution.
FILE "N"	FILE "N",filename,records,record size,blocking factor Creates a new random file with the specified number of records (1-32767), record size (1-32767 bytes), and blocking factor (1-255). File name is a string expression containing a valid FCS file name.
FILE "R"	FILE "R",filenumber,filename,buffers <;records,record size, blocking factor> Opens a random file with the specified file number (1-127) and number of buffers (1-255).
FILE "A"	FILE "A",file,current record <,records, record size, blocking factor> Finds the attributes for the specified file.

FILE "C" FILE "C",file1 <,...>
Closes the specified files and releases the buffer space.

FILE "D" FILE "D",file1 <,...>
Writes any modified buffers for the specified files immediately to the corresponding devices.

FILE "T" FILE "T" <,line number>
Causes file errors to trap to the specified line number. No line number turns the file error trapping off.

FILE "E" FILE "E",file,error,line number
Finds the disk error number and location of the last file error.

FOR FOR nvar = nexpr1 TO nexpr2 <STEP nexpr3>
Sets up a loop to be executed the specified number of times.

GET GET file<,record<,first>>;nvar,svar[byte count],...
Reads from the record in the file starting from the first byte into the variables in the list. String variables must have a byte count (1-255).

GOSUB GOSUB line number
Used to transfer control to the specified line number of a subroutine.

GOTO GOTO line number
Used to unconditionally transfer control to the specified line number.

IF IF nexpr GOTO line number
IF nexpr THEN line number
Used to conditionally transfer control to the specified line number.

IF nexpr THEN statement <:statement:...>
Used to conditionally execute BASIC statements

INPUT INPUT <"string";> var <,var,...>
Used to input data from the terminal, prompts with either "?" or the optional quoted string as the prompt.

LIST LIST <line number>
Prints the user program currently in memory on the CRT display, optionally, starting from the specified line number.

LOAD LOAD filename
Loads the specified file. If no extension is specified, then a BASIC program is loaded; otherwise, the .ARY extension loads the specified numeric array, and the .DAT extension loads the specified data into memory after BASIC's workspace.

NEXT NEXT <nvar <,nvar,...>>
Placed at the end of a FOR loop to return control to the FOR statement.

ON

ON nexpr GOSUB line number <,line number,...>
Multiple GOSUB statement. Transfers control to the line number specified by nexpr.

ON nexpr GOTO line number <,line number,...>
Multiple GOTO statement. Transfers control to the line number specified by nexpr.

OUT

OUT port,nexpr
Outputs the specified nexpr (0-255) to the 8080 port (0-255). CAUTION: Do not output to the CRT controller chips ports (96-111).

PLOT

PLOT nexpr <,nexpr,...>
Sends the one byte results (0-255) of the expressions to the CRT display.

POKE

POKE location,nexpr
Causes the one byte result of nexpr to be placed in the specified memory location (-32768 to 65535).

PRINT

PRINT expr <,expr,...>
PRINT expr <;expr;...>
Prints the results of the expressions in the list. Commas are used for normal spacing, and semicolons are used for compressed spacing. If either a comma or a semicolon is the last item in the print list, the carriage return is suppressed.

PRINT SPC(nexpr)
Prints the specified number of spaces. May be placed anywhere in the print list.

PRINT TAB(nexpr)
Tabs to the specified column. May be placed anywhere in the print list.

?

Equivalent to the keyword PRINT.

PUT

PUT file <,record<,first>>; nexpr,sexpr[byte count] <,...>
Writes the expressions in the list to the record in the file starting from the first byte. String expressions must have a byte count.

READ

READ var <,var,...>
Used to assign the values in DATA statements to the variables specified in the list.

REM

REM comment
Used to insert explanatory comments in a BASIC program.

RESTORE

RESTORE <line number>
Resets the data pointer to either the first DATA statement or optionally to the specified line number.

RETURN — RETURN
Returns program control to the statement following the last executed GOSUB statement.

RUN — RUN <line number>
Executes the BASIC program in memory, optionally, starting at the specified line number.

SAVE — SAVE filename
Saves the specified file. If no extension is specified, the current BASIC program in memory is saved; otherwise, the .ARY extension saves the specified numeric array, and the .DAT extension saves the data in memory after BASIC's workspace.

WAIT — WAIT port, nexprl <,nexpr2>
Reads from the specified 8080 port and exclusive OR's the result with nexpr2 (0 if not present), and then AND's with nexprl. The program waits until the result is zero before continuing.

: — statement : statement < : statement : ... >
A colon is used to separate statements in a multiple statement line.

B.2 BASIC Operators

Symbol	Function
=	Assignment or equality test (DISK BASIC does not allow the LET statement)
-	Negation or Subtraction
+	Addition or String Concatenation
*	Multiplication
/	Division
^	Exponentiation
NOT	Logical or One's complement (2 byte integer)
AND	Logical or Bitwise AND (2 byte integer)
OR	Logical or Bitwise OR (2 byte integer)
=,<,>,<=, =<,>=,=>, <>	Relational tests (result is TRUE = -1 or FALSE = 0)

The precedence of operators is:

1. Expressions in parentheses
2. Exponentiation (A^B)
3. Negation (-X)
4. *,/
5. +,-
6. Relational Operators (=,<>,<,>,<=,>=)
7. NOT
8. AND
9. OR

B.3 Standard Mathematical Functions

BASIC provides functions to perform certain standard mathematical operations such as square roots, logarithms, etc.

These functions have three or four letter call names followed by a parenthesized argument. They are predefined and may be used anywhere in a program.

Call Name	Function
ABS(x)	Returns the absolute value of x.
ATN(x)	Returns the arctangent of x as an angle in radians in range + /2), where = 3.14159.
CALL(x)	Call the user machine language program at decimal location 33282. (8202 HEX) D,E registers have value of X and D,E registers must have Y on return from machine language routine.
COS(x)	Returns the cosine of x radians.
EXP(x)	Returns the value of e^x where e = 2.71828.
FRE(x)	Returns number of free bytes not in use.
INT(x)	Returns the greatest integer less than or equal to x.
INP(x)	Returns a byte from input port x. The range for x is 0 to 255.

LOG(x)	Returns the natural logarithm of x.
PEEK(x)	Returns a byte from memory address -32768<x<65535; if x is negative the memory address is 65536+x.
POS(x)	Returns the value of the current cursor position between Ø and 63.
RND(x)	Returns a random number between Ø and 1.
SGN(x)	Returns a -1, Ø, or 1, indicating the sign of x.
SIN(X)	Returns the sine of x radians.
SPC(x)	Causes x spaces to be generated. (Valid only in a PRINT statement).
SQR(x)	Returns the square root of x.
TAB(x)	Causes the cursor to space over to column number x. (Valid only in a PRINT statement).
TAN(x)	Returns the tangent of x radians.

The argument x to the functions can be a constant, a variable, an expression, or another function. Square brackets cannot be used as the enclosing characters for the argument x, e.g. SIN[x] is illegal.

Function calls, consisting of the function name followed by a parenthesized argument, can be used as expressions anywhere that expressions are legal.

Values produced by the functions SIN(x), COS(x), ATN(x), SQR(x), EXP(x), and LOG(x) have six significant digits.

B.4 Standard String Functions

Like the intrinsic mathematical functions (e.g., SIN, LOG), BASIC contains various functions for use with character strings. These functions allow the program to access parts of a string, determine the number of characters in a string, generate a character string corresponding to a given number or vice versa, and perform other useful operations. The various functions available are summarized in the following table.

Call Name	Function
ASC(x$)	Returns the eight bit internal ASCII code (0-255) for the one-character string. If the argument contains more than one character, then the code for the first character in the string is returned. A value of 0 is returned if the argument is a null string (LEN(x$) = 0). See ASCII codes in Appendix E.
CHR$(x)	Generates a one-character string having the ASCII value of x where x is a number in the range 0 to 255. Only one character can be generated.
FRE(x$)	Returns number of free string bytes. (See CLEAR statement in 5.11)
LEFT$(x$,I)	Returns left-most I characters of string (x$). If I>LEN(x$), then x$ is returned.
LEN(x$)	Returns the number of characters in the string x$, with non-printing characters and blanks being counted.
MID$(x$,I,J)	J is optional. Without J, returns right-most characters from x$ beginning with the Ith character. If I>LEN(x$), MID$ returns the null string. With 3 arguments, it returns a string of length J of characters from x$ beginning with the Ith character. If J is greater than the number if characters in x$ to the right of I, MID$ returns the rest of the string. Argument ranges: 0<I<=255, 0<=J<=255.
RIGHT$(x$,I)	Returns right-most I characters of string (x$). If I>LEN(x$), then x$ is returned.
STR$(x)	Returns the string which represents the numeric value of x as it would be printed by a PRINT statement.
VAL(x$)	Returns the number represented by the string x$. If the first character of x$ is not +, -, or a digit, then the value 0 is returned.

In the above example, x$ and y$ represent any legal string expressions, and I and J represent any legal arithmetic expressions.

B.5 BASIC Error Codes

After an error occurs, BASIC returns to command level and types READY. Variable values and the program text remain intact, but the program cannot be continued and all GOSUB and FOR context is lost.

When an error occurs in a statement executed in immediate mode, no line number is printed.

Format of error messages:

Stored BASIC statement	XX ERROR IN YYYY
Immediate mode statement	XX ERROR

In both of the above examples, "XX" is the error code. The "YYYY" is the line number in which the error occurred in the indirect statement.

The following are the possible error codes and their meanings:

Error Meaning

BS Bad Subscript. An attempt was made to reference a matrix element which is outside the dimension of the matrix. This error can occur if the wrong number of dimensions is used in a matrix reference. For instance, A (1,1,1)=Z when A has been dimensioned DIM A(2,2).

DD Double Dimension. After a matrix was dimensioned, another dimension statement for the same matrix was encountered. This error often occurs if a matrix has been given the default dimension 10 because a statement like A(I)=3 is encountered and then later in the program a DIM A(100) is found.

CF Call Function error. The parameter passed to a mathematical or string function was out of range. CF errors can occur due to:

1. a negative matrix subscript (A(-1)=0)
2. an unreasonably large matrix subscript (>32767)
3. LOG with a negative or zero argument
4. SQR with a negative argument
5. A^B with A negative and B not an integer
6. a CALL(x) before the address of the machine language subroutine has been patched in
7. calls to MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, PLOT, TAB, SPC or ON...GOTO/GOSUB with an improper argument

ID Illegal Direct. You cannot use an INPUT or DEF statement in immediate mode.

NF NEXT without FOR. The variable in a NEXT statement corresponds to no previously mentioned FOR statement.

OD Out of Data. A READ statement was executed but all of the DATA statements in the program have already been read. The program tried to read too much data or an insufficient number of data values were included in the program.

OM Out of Memory. Program too large, too many variables, or too many FOR loops, too many GOSUB's, too complicated an expression, or any combination of the above.

OV Overflow. The result of a calculation was too large to be represented in BASIC's numeric format. If an underflow occurs, zero is given as the result and execution continues without any error message being printed.

SN Syntax error. Missing parenthesis in an expression, illegal character in a line, incorrect punctuation, etc.

RG RETURN without GOSUB. A RETURN statement was encountered without a previous GOSUB statement being executed.

US Undefined Statement. An attempt was made to GOTO, GOSUB, or THEN to a statement which does not exist.

/0 Division by Zero.

CN Continue error. Attempt to continue a program when none exists, an error occurred, or after a new line was typed into the program.

LS Long String. Attempt was made by use of the concatenation operator to create a string more than 255 characters long.

OS Out of String Space. Use the CLEAR X statement to allocate more string space or use smaller strings or fewer string variables.

SL SAVE/LOAD error. (From disk operation.) Other error messages may also appear from the File Control System. See Appendix B.2.

ST String Temporaries. A string expression was too complex. Break it into two or more shorter expressions.

TM Type Mismatch. The left hand side of an assignment statement was a numeric variable and the right hand side was string, or vice versa, or, a function which expected a string argument was given a numeric one or vice versa.

UF Undefined Function. Reference was made to a user defined function which was never defined.

B.6 BASIC Random File Error Codes

Error	Number	Meaning
EV		No error vector. No file error trap line number has been set with a FILE "T" statement.
BF	2	Bad file name. Improper FCS file name.
NO	4	File not open. The specified file number is not open.
AO	6	File already open. The specified file number is already in use.
FS	8	File size error. The file being created with the FILE "N" statement is too large or the file parameters on the file being opened with the FILE "R" statement are improper.
RO	10	Record overflow. Too many data bytes were either read from or written to the current record.
EF	12	End of file. Tried to read or write past the end of the file.
CO	14	Cant't open file. The specified file does not exist on the specified device. (Possibly a Sof-Disk or hardware problem.)
CC	16	Can't close file. The specified file can not be closed. (Usually a disk or hardware problem.)
RE	18	FCS READ error. (Usually a disk or hardware problem.)
WE	20	FCS WRITE error. (Usually a disk or hardware problem.)

B.7 BASIC "Tokens"

Intecolor reduces BASIC commands to one-byte symbolic representations or tokens to reduce space requirements and make programs run more efficiently. The key tops in the color pad (on 101 and 117-key keyboards) indicate the BASIC commands which can be generated by operating the key while the COMMAND key is held down. Other commands can be generated by use of other keys with COMMAND (or both CONTROL and SHIFT) or CONTROL held operated.

Command	COMMAND Key with	Hex Value	Decimal Value
END	@	80	128
FOR	A	81	129
NEXT	B	82	130
DATA	C	83	131
INPUT	D	84	132
DIM	E	85	133
READ	F	86	134
FILE	G	87	135
GOTO	H	88	136
RUN	I	89	137
IF	J	8A	138
RESTORE	K	8B	139
GOSUB	L	8C	140
RETURN	M	8D	141
REM	N	8E	142
GET	O	8F	143
OUT	P	90	144
PUT	Q	91	145
PLOT	R	92	146
SAVE	S	93	147
LOAD	T	94	148
POKE	U	95	149
PRINT	V	96	150
LIST	W	97	151
CONT	X	98	152
CLEAR	Y	99	153
DEF	Z	9A	154
WAIT	[	9B	155
ON	\	9C	156
TAB(	]	9D	157
TO	^	9E	158
FN	_	9F	159
SPC(	0	A0	160
THEN	1	A1	161
NOT	2	A2	162
STEP	3	A3	163
+	4	A4	164
-	5	A5	165
*	6	A6	166
/	7	A7	167
^	8	A8	168
AND	9	A9	169
OR	:	AA	170
>	;	AB	171

Command	CONTROL Key with	Hex Value	Decimal Value
=	,	AC	172
<	-	AD	173
SGN	.	AE	174
INT	/	AF	175
ABS	0	B0	176
CALL	1	B1	177
FRE	2	B2	178
INP	3	B3	179
POS	4	B4	180
SQR	5	B5	181
RND	6	B6	182
LOG	7	B7	183
EXP	8	B8	184
COS	9	B9	185
SIN	:	BA	186
TAN	;	BB	187
	COMMAND Key with		
ATN	,	BC	188
PEEK	-	BD	189
LEN	.	BE	190
STR$	/	BF	191
	CONTROL Key with		
VAL	F0	C0	192
ASC	F1	C1	193
CHR$	F2	C2	194
LEFT$	F3	C3	195
RIGHT$	F4	C4	196
MID$	F5	C5	197

APPENDIX C. FCS (File Control System)

C.1 FCS Commands

The File Control System is entered by pressing (ESC) then D from the keyboard, or PLOT 27,4 from BASIC. If (ESC) D is from the keyboard then BASIC is terminated and must be re-entered by an (ESC) E key sequence. When entering FCS commands, only the first three letters of an internal command need to be typed in.

The following definitions will be used to describe the FCS commands:

( ) denotes mandatory element;
[ ] denotes optional element and if not specified, will result in the default type.

(Device name:) = [Device type] [Number] (:)
Standard device types are MD, DM, FD and DF for the 3650 Series Intecolor and number is either 0, 1, 2 or 3.

(Memory spec) = (Load address)(byte count)
or (Load address)(-end address)
All memory addresses are in HEX format.

(File Spec.) = (File name) [.Type] [;Version]
File name is any 6 characters. Type can be any three characters and PRG is the default type. Version is 00 to FF HEX.
NOTE: After a default device type has been selected only the number of the device is required.

Type of Command	Syntax and Description
Copy	COP [Device Name:] (File Spec) TO [Device Name:] [File Spec] Copies the specified file, usually, to another device.
Delete	DEL [Device Name:] (File Spec) All File Spec options are required. Deletes the specified file, and repacks the disk and directory.
Device	DEV [Device Name:] Sets and displays the current default Device Name.
Directory	DIR [Device Name:] Lists the directory for the default or specified device.
Execute	EXE (File Spec) [parameter 1][,parameter 2]...[parameter 9] Allows a file consisting of FCS commands to be executed one after another in batch mode fashion.
Exit "FCS"	ESC ESC or ESC E to return to BASIC. In BASIC, use PLOT 27,27.
Initialize	INI (Device Name:) (Volume Name) [No. Dir. Blocks] Initializes the directory on the disk currently in the specified device. Use only when "clean" diskette is wanted.

Load	LOA [Device Name:] (File Spec) [Load Address] or LOA [Device Name:] (File Spec) [Low Addr [Memory Spec]] Loads memory with a program. Defaults to .LDA type files written by the Intecolor Assembler. (See Section 12.02.)
Read	REA [Device Name:] (Start Block No.) (Memory Spec) Reads into memory from anywhere on the disk without regard to the directory or program boundaries.
Rename	REN [Device Name:] (File Spec) TO (File Spec) Allows any file to be renamed without changing any information in the file itself.
Run	RUN [Device Name:] (File Spec) Loads and executes the specified program. (Default type .PRG)
Save	SAV [Device Name:] (File Spec) (Memory Spec) [Start Address [Actual Address]] Saves memory image in the specified file. The Start Address and Actual Address default to the lower limit of the Memory Spec.
Write	WRI [Name:] (Start Block No.) (Memory Spec) Writes memory image to the specified block on a disk without regard to the FCS directory information and file boundaries. CAUTION: It is possible to destroy the FCS directory and file information on a disk with this command.
Duplicate	DUP (Device Name:) TO (Device Name:) This external command duplicates all the files on one disk to another disk in a second disk drive.
Duplicate All	DUA (Device Name:) TO (Device Name:) This external command copies one disk's entire contents, including the free space, to another disk.
File	FIL (File Spec) (# Records) [Record Size Blocking Factor] This external command creates an empty file of the specified size.
Merge	MER (file1),(file2),...,[fileN] TO (New File Spec) This external command creates a new file consisting of the contents of two or more existing files.
Print	PRI (File Spec) This external command is used to print a file to the CRT or the printer, if it is on.
Save Screen	SVSCR (File Spec) This external command saves a copy of the screen display in a disk file. (Default type .PIC)
Load Screen	LDSCR (File Spec) This external command puts a display on the screen from a disk file previously created by an SVSCR command. (If not of default type .PIC, file type must be specified.)

C.2 FCS Error Codes

The numbers to the right of the code meanings, under the heading CAUSES, refer to a list which follows this code list.

Message	Meaning	Causes
EBLF	BAD LOAD FILE SPEC	2
EBLK	INVALID BLOCK NUMBER	2
ECOP	ERROR DURING COPY	1, 3
ECFB	CAN'T FIND BLOCK	3
EDCS	DATA CRC ERROR	3
EDEL	DELETE ERROR	1, 3
EDFN	DUPLICATE FILE NAME	2
EDIR	DIRECTORY ERROR	1, 2
EDRF	DIRECTORY FULL	4
EDSY	DATA SYNC CHARACTER ERROR	1, 3
EDUP	ERROR DURING DUPLICATE	1, 3
EFNF	FILE NOT FOUND	2
EFRD	FILE READ ERROR	3
EFWR	FILE WRITE ERROR	3
EHCS	HEADER CRC ERROR	3
EIVC	INVALID COMMAND	2
EIVD	INVALID DEVICE	2
EIVF	INVALID FUNCTION	2
EIVP	INVALID PARAMETERS	2
EIVU	INVALID UNIT	2
EKBA	KEYBOARD ABORT	4
EMDV	MISSING DEVICE NAME	2
EMEM	MEMORY ERROR DURING READ	4
EMFN	MISSING FILE NAME	2

EMVN	MISSING VOLUME NAME	2
EMVR	MISSING VERSION	2
ENSA	NO START ADDRESS	2
ENVE	NO VOLUME ENTRY IN DIRECTORY	5
ERSZ	FILE TOO LARGE TO READ INTO ALLOCATED MEMORY	2, 4
ESIZ	DEVICE SIZES NOT SAME	1
ESKF	SEEK FAILURE	1
ESYN	SYNTAX ERROR	2
EVFY	VERIFY FAILURE DURING WRITE	3
EVOV	VERSION NUMBER OVERFLOW	4
EWRF	WRITE FAILURE	3
EWSF	FILE TOO LARGE TO WRITE ON DISKette	2, 4

Causes of FCS Errors

1. Mechanical Problem -- Jammed READ/WRITE head, loose disk drive, internal I/O connectors.

2. Invalid User Input -- Incorrect entry from user. Refer to FCS Commands, Appendix Section C.1.

3. Diskette Failure — Try a different diskette.

4. Error Message is self-explanatory.

5. Disk Not Initialized — you need to initialize the disk; see the FCS INITIALIZE command.

D. CRT COMMAND SUMMARIES

D.1 Control Codes

To enter a control code, hold down the CONTROL key while depressing the desired character key.

Control Code	Control Key	Section	Explanation
Ø	@		NULL-Has no effect.
1	A	10.01.02	AUTO - Loads and runs a BASIC program named "MENU" from the disk drive.
2	B	11.05	PLOT - Enters graphic plot mode (see plot submodes), not allowed as a BASIC input character.
3	C	11.04	CURSOR X,Y - Enters X-Y cursor address mode for either visible cursor or blind cursor, used to go from BASIC to CRT MODE when typed as a BASIC input character.
4	D		Not used.
5	E		Not used.
6	F	11.02	CCI - The following character provides the 8 bit visible status word. Specifies Foreground, Background, Blink and Plot. (See Appendix D.2)
7	G		Not used.
8	H	11.04	HOME - Moves the cursor to top left corner of display.
9	I	11.04	TAB - Causes cursor to advance to next column--the tab columns are every 8 characters.
10	J	11.04	CURSOR DOWN or LINEFEED - Causes a break in BASIC execution of a program, causes the cursor to move down one line.
11	K	11.04	ERASE LINE - Causes the cursor to return to the beginning of the line and causes the complete line to be erased. Also causes the BASIC input line to be ignored.

12	L	11.04	ERASE PAGE - Causes the complete screen to be erased and the cursor to be moved to the home position. BASIC input ignores this character.
13	M	11.04	CARRIAGE RETURN - Causes the cursor to move to the beginning of the line it is presently on. Causes BASIC input to accept the typed line and process as a statement or input data.
14	N	11.02	A7 ON - Turns the A7 flag on. (2x character height and also stop bit.)
15	O	11.02	BLINK/A7 OFF - Turns the blink bit and A7 flag off.
16	P	11.02	BLACK KEY - Sets foreground color black if flag is off and background black if flag is on. (See codes 29 and 30 below.)
17	Q	11.02	RED KEY - Same as above with color red.
18	R	11.02	GREEN KEY - Same as above with color green.
19	S	11.02	YELLOW KEY - Same as above with color yellow.
20	T	11.02	BLUE KEY - Same as above with color blue.
21	U	11.02	MAGENTA KEY - Same as above with color magenta.
22	V	11.02	CYAN KEY - Same as above with color cyan.
23	W	11.02	WHITE KEY - Same as above with color white.
24	X	11.06	XMIT - Causes data to be transmitted from the visible cursor to the end of the page or until an FF,00 sequence is found in refresh RAM. Sends text characters with a linefeed and carriage return at end of each line. NOTE: Color status is not sent.
25	Y	11.04	CURSOR RIGHT - Causes the cursor to move right 1 position. On BASIC input displays previous character input.
26	Z	11.04	CURSOR LEFT - Causes the cursor to move left 1 position. On BASIC input deletes previous character from input buffer.

27	[		ESC - Provides an entry to the escape code table -- must be followed by one or more codes for proper operaton.
28	/	11.Ø4	CURSOR UP - Causes the cursor to move up one line.
29	]	11.Ø2	FG ON/FLAG OFF - Sets the flag bit off. If followed by one of the color keys it will set the foreground to that color. Also, does not change input codes in the range 96 to 127 that are to be stored in the display memory, i.e. the shifted alphabetic characters are displayed as shown in columns 6 and 7 in the Intecolor character set in Appendix F. In plot mode OR's "ON" bits.
3Ø	^	11.Ø2	BG ON/FLAG ON - Sets the flag bit on. If followed by one of the color keys it will set the background to that color. With the FLAG on the shifted alphabetic characters 96 to 127 are converted into Ø to 31 when stored in the display memory, i.e. the characters displayed are shown in columns Ø and 1 in Appendix F.2. In plot mode XOR's "ON" bits.
31	_	11.Ø2	BLINK ON - Turns on the blink bit which will blink the foreground color against the backround color.

D.2 Status Word Format

A7	A6	A5	A4	A3	A2	A1	A0
PLOT	BLINK	BACKGROUND COLOR			FOREGROUND COLOR		
		BLUE	GREEN	RED	BLUE	GREEN	RED

D.3 Escape Codes

To enter an escape code sequence, operate the ESC key and then follow with operation of the desired character key.

Escape Code	Key	Section	Explanation
Ø	@		Used for terminal control—not available for any other use.
1	A	11.Ø4	Blind cursor mode.
2	B	11.Ø3	Plot via color pad.
3	C	11.Ø6	Transmit cursor X,Y position to RS-232C port.
4	D	12.Ø1	Enters Disk File Control System (FCS) with CRT as output.
5	E	3.Ø2	Re-entry to DISK BASIC.
6	F	11.Ø7	Sets full duplex mode, not functional when in BASIC. The BREAK key restores to half duplex when in terminal mode.
7	G	12.Ø1	Enters Disk File Control System (FCS) with RS-232C port as output.
8	H	11.Ø7	Sets half duplex mode.
9	I	11.Ø8	Causes a program jump to location 36864. (9ØØØ HEX)
1Ø	J	11.Ø4	Sets write vertical mode.
11	K	11.Ø4	Sets scroll up and write left to right mode.
12	L	11.Ø7	Sets local mode.
13	M	11.Ø6	Sends all output to the RS-232C port.
14	N		Set to ignore all inputs. In BASIC POKE must be used to reset back to normal, or hit CPU RESET.
15	O		Not used.
16	P		Not used.
17	Q		Not used.

18	R	11.06	Baud rate selection mode. A7 on = 1 stop bit, A7 off = 2 stop bits. See Appendix C.4 below for the next key to specify the baud rate.
19	S	11.08	Causes a program jump to location 40960. (A000 HEX)
20	T	11.08	Causes a program jump to location 33280. (8200 HEX)
21	U		Not used.
22	V		Not used.
23	W	3.02	Initializes and transfers control to DISK BASIC 8001.
24	X	11.04	Sets terminal to page mode and write left to right mode.
25	Y	11.08	Test mode — fill page with next character.
26	Z		Not used.
27	[	11.04	Visible cursor mode. Also used to exit FCS.
28	/		Not used.
29	]		Not used.
30	^	11.08	User definable escape code. Causes a program jump to locaton 33215. (81BF HEX)
31	_		Transfer control to the CRT mode.

D.4 Baud Rate Selection

Number Key	1	2	3	4	5	6	7
Baud Rate	110	150	300	1200	2400	4800	9600

D.5 Graphic Plot Submodes

Disk BASIC Plot or RS-232C Code	Plot Submode	Standard Keyboard	Optional Function Keyboard
255	Plot Mode Escape	CONTROL SHIFT ?	F 15
254	Character Plot	CONTROL SHIFT >	F 14
253	X Point Plot	CONTROL SHIFT =	F 13
252	Y Point Plot	CONTROL SHIFT <	F 12
251	X-Y Incremental Point Plot	CONTROL ;	F 11
250	XØ of X Bar Graph	CONTROL :	F 1Ø
249	Y of X Bar Graph	CONTROL 9	F 9
248	X max of X Bar Graph	CONTROL 8	F 8
247	Incremental X Bar Graph	CONTROL 7	F 7
246	YØ of Y Bar Graph	CONTROL 6	F 6
245	X of Y Bar Graph	CONTROL 5	F 5
244	Y max of Y Bar Graph	CONTROL 4	F 4
243	Incremental Y Bar Graph	CONTROL 3	F 3
242	XØ Vector Plot	CONTROL 2	F 2
241	YØ Vector Plot	CONTROL 1	F 1
240	Incremental Vector Plot	CONTROL Ø	F Ø

For incremental plot submodes see the format of the incremental direction codes below.

D.6 Incremental Direction Codes (Decimal Values)

ΔX1		ΔY1		ΔX2		ΔY2	
A7	A6	A5	A4	A3	A2	A1	AØ
+	-	+	-	+	-	+	-
128	64	32	16	8	4	2	1

APPENDIX E. INTERNAL FEATURES

E.1 Key Memory Locations

24576 to 28671 = Disk Buffer RAM 6000-6FFF HEX
28672 to 32767 = Screen Refresh RAM 7000-7FFF HEX
32940 = Points to maximum RAM used by BASIC
32980 = Points to start of BASIC source
32982 = Points to end of source and start of variables
32984 = Points to end of variables and start of arrays
32986 = Points to end of arrays
33209 = 0 to 59 seconds of Real Time Clock
33210 = 0 to 59 minutes of Real Time Clock
33211 = 0 to 23 hours of Real Time Clock
33215 = User ESCAPE jump vector
33218 = User output FLAG jump vector
33221 = User input FLAG jump vector
33224 = User timer no. 2 jump vector
33228 = External output port buffer
33247 = Keyboard FLAG.
33249 = FCS output FLAG
33251 = Input port FLAG
33265 = BASIC output FLAG
33272 = Output port FLAG
33273 = LIST output FLAG
33278 = Keyboard character
33279 = Keyboard character ready FLAG
33282 = Location of CALL(x) jump
33285 = BASIC output vector location
33289 = Number of characters on terminal output
33433 = Start of BASIC source code
65535 = Maximum Amount of RAM

E.2 Port Assignments

PORT # HEX	I/O PORT ADDRESS
0 - F	TMS 5501
10 - 1F	TMS 5501 Duplicate Addresses
20 - 2F	FD1771-B Floppy Disk Controller
30 - 4F	Not Assigned
50 - 5F	Tester
60 - 7F	SMC 5048-005 CRT Controller
80 - FF	X-Bus

PORT # HEX DEC	TMS 5501 I/O CHIP (See Appendix G.2)	
0 - 0	Read Serial Data in from RS-232C interface	
1 - 1	Read Parallel Data from keyboard and disk	
2 - 2	Read Interrupt Address on TMS 5501	
3 - 3	Read Status on TMS 5501	
4 - 4	Issue Discrete Command	
5 - 5	Set Baud Rate on Serial I/O	
6 - 6	Transmit Serial Data to RS-232C interface	
7 - 7	Transmit Parallel Data to keyboard and disk (also controls Disk R/W)	
8 - 8	Load Interrupt Mask Register	
9 - 9	Interval Timer #1	(Power-up)
A - 10	Interval Timer #2	(Free — available to user)
B - 11	Interval Timer #3	(Keyboard)
C - 12	Interval Timer #4	(Bell)
D - 13	Interval Timer #5	(CRT)
E - 14	No Function	
F - 15	No Function	
	FD1771 Floppy Disk Controller (See Appendix G.4)	
20 - 32	Read Disk Status	
21 - 33	Read Disk Track	
22 - 34	Read Disk Sector	
23 - 35	Read Disk Data	
24 - 36	Write Command	
25 - 37	Write Track	
26 - 38	Write Sector	
27 - 39	Write Data	
28 - 40	Write Drive Select	
29 - 41	Read Interrupt Address	
2A - 42	Put CPU on Hold	
	SMC 5048 CRT CHIP (See Appendix G.3)	
60 - 96	Load Register 0 - Don't Load	These six registers are masked — not changeable by user
61 - 97	Load Register 1 - Don't Load	
62 - 98	Load Register 2 - Don't Load	
63 - 99	Load Register 3 - Don't Load	
64 - 100	Load Register 4 - Don't Load	
65 - 101	Load Register 5 - Don't Load	
66 - 102	Load Register 6 - Roll Register#	
67 - 103	Processor Load Command - Don't Use	
68 - 104	Read Cursor X Register	
69 - 105	Read Cursor Y Register	
70 - 106	Issue Reset Command - Don't Issue	
6B - 107	Scroll up 1 line	
6C - 108	Load Cursor X Register	
6D - 109	Load Cursor Y Register	
6E - 110	Load Start Timing - Don't Load	
6F - 111	Self Load Commad - Don't Use	

E.3 The Extension Bus (X-Bus)

Pin	Designation	Pin	Dsignation
1	Ground	14	Ground
2	XA0 (address LSB)	15	XB2 (data bus bit)
3	Ground	16	X I/O W (write)
4	XA1 (address bit)	17	XB3 (data bus bit)
5	Ground	18	Ground
6	XA2 (address bit)	19	XB4 (data bus bit)
7	+5 VDC (200MA max)	20	XA5 (address bit)
8	XA3 (address bit)	21	XB5 (data bus bit)
9	XB1 (data bus bit)	22	Ground
10	XA4 (address bit)	23	XB6 (data bus bit)
11	XB0 (data bus bit)	24	Not used
12	X I/O R (read)	25	XB7 (data bus bit)
13	Ground	26	Ground

The buffered data, address and signal leads are capable of sinking five TTL loads. The +5V is primarily for sensing.

The connector for the X-Bus is a 50-pin edge connector. In addition to the X-Bus connections listed above, several other connections are available for user-designed devices. The additional address lines are not buffered. Any device using them should provide buffering. The additional power outputs should be loaded only lightly if at all. The additional connections are as follows:

Pin	Designation	Pin	Designation
33	MR (memory read)	42	A13 (address bit)
34	O2 (timing, TTL level)	43	A8 (address bit)
35	-12 VDC	44	A10 (address bit)
36	MW (memory write)	45	A9 (address bit)
37	A6 (address bit)	46	A11 (address bit)
38	SYNC (from 8080)	47	A12 (address bit)
39	A7 (address bit)	48	+12 VDC
40	XT WAIT (input)	49	A14 (address bit)
41	A15 (address bit)	50	-5 VDC

E.4 RS-232C INTERFACE

Terminal	Signal
1	AA Protective Ground
2	BA Transmitted Data
3	BB Received Data
4	CA Request to Send
5	CB Clear to Send
7	AB Signal Ground
20	CD Data Terminal Ready

The preceding table lists the usual modem connections. However, the manufacturer's instructions for the particular modem or other serial type device to be connected to the RS-232C Interface should be consulted.

The length of the cable connected between the Intecolor and the device should not exceed 50 feet. For direct connection to a host computer ("direct", rather than through a modem or other interfacing device), the Transmit and Receive data line connections likely must be reversed. (The host computer's Receive may be the Intecolor's Transmit and the host computer's Transmit may be the Intecolor's Receive.) A similar situation sometimes occurs when a serial type printer is connected with the Intecolor's serial I/O.

APPENDIX F. DECIMAL CODE INFORMATION

F.1 Intecolor Code Set

The chart at right shows the codes generated by the various keystroke combinations. Decimal values are listed in the key blocks; binary and hexadecimal values are listed at the top of the columns and at the left side of the rows — the most significant bits are at the top.

For example, the decimal value for the upper case letter Y is 89. Going up to the top of the column, the most significant bits in binary are seen to be 0 1 0 1. Going to the left on the row containing Y, the other four bits in binary are 1 0 0 1. Thus the binary code for Y is

0 1 0 1 1 0 0 1

and the Hexadecimal equivalent is 59.

																	OPTIONAL FUNCTION KEYS				
			A_7-A_4 UNIQUE KEYBOARD CODE		9	D	6	4	8	C	A	E	8	F	7	5	1	2	3	0	
			HEXADECIMAL		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
			A7		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
			A6		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
			A5		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
			A4		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
HEXADECIMAL	A3	A2	A1	A0	CONTROL @ TO 0	CONTROL P TO _	SHIFT SP TO ?				SHIFT @ TO 0	SHIFT P TO _	CONT-SHIFT @ TO 0	CONT-SHIFT P TO _	CONT-SHIFT SP TO ?	CONT @ TO ?	CONTROL F0-FF	SHIFT F0-FF	CONT-SHIFT F0-FF	F0-FF	
0	0	0	0	0	NULL 0	BLACK 16	SPACE 32	0 48	@ 64	P 80	`/GO 96	p/G16 112	@ 128	P 144	SPACE 160	0 176	F0 192	F0 208	F0 224	F0 240	Xo-Yo INCREMENTAL VECTOR
1	0	0	0	1	PROTECT 1	RED 17	! 33	1 49	A 65	Q 81	a/G1 97	q/G17 113	A 129	Q 145	! 161	1 177	F1 193	F1 209	F1 225	F1 241	Yo VECTOR PLOT
2	0	0	1	0	PLOT MODE 2	GREEN 18	" 34	2 50	B 66	R 82	b/G2 98	r/G18 114	B 130	R 146	" 162	2 178	F2 194	F2 210	F2 226	F2 242	Xo VECTOR PLOT
3	0	0	1	1	CURSOR X-Y MODE 3	YELLOW 19	# 35	3 51	C 67	S 83	c/G3 99	s/G19 115	C 131	S 147	# 163	3 179	F3 195	F3 211	F3 227	F3 243	INCREMENTAL Y BAR
4	0	1	0	0	FREE 4	BLUE 20	$ 36	4 52	D 68	T 84	d/G4 100	t/GO 116	D 132	T 148	$ 164	4 180	F4 196	F4 212	F4 228	F4 244	Ym-YBAR
5	0	1	0	1	FREE 5	MAGENTA 21	% 37	5 53	E 69	U 85	e/G5 101	u/G21 117	E 133	U 149	% 165	5 181	F5 197	F5 213	F5 229	F5 245	X-Y BAR
6	0	1	1	0	CCI 6	CYAN 22	& 38	6 54	F 70	V 86	f/G6 102	v/G22 118	F 134	V 150	& 166	6 182	F6 198	F6 214	F6 230	F6 246	Yo-Y BAR
7	0	1	1	1	BELL 7	WHITE 23	' 39	7 55	G 71	W 87	g/G7 103	w/G23 119	G 135	W 151	' 167	7 183	F7 199	F7 215	F7 231	F7 247	INCREMENTAL X BAR
8	1	0	0	0	HOME 8	TRANSMIT 24	(40	8 56	H 72	X 88	h/G8 104	x/G24 120	H 136	X 152	(168	8 184	F8 200	F8 216	F8 232	F8 248	Xm-X BAR
9	1	0	0	1	TAB 9	CURSOR RIGHT 25	) 41	9 57	I 73	Y 89	i/G9 105	y/G25 121	I 137	Y 153	) 169	9 185	F9 201	F9 217	F9 233	F9 249	Y-X BAR
A	1	0	1	0	LINE FEED 10	CURSOR LEFT 26	* 42	: 58	J 74	Z 90	j/G10 106	z/G26 122	J 138	Z 154	* 170	: 186	F10 202	F10 218	F10 234	F10 250	Xo-X BAR
B	1	0	1	1	ERASE LINE 11	ESC 27	+ 43	; 59	K 75	[91	k/G11 107	{/G27 123	K 139	[155	+ 171	; 187	F11 203	F11 219	F11 235	F11 251	X-Y INCREMENTAL PLOT
C	1	1	0	0	ERASE PAGE 12	CURSOR UP 28	, 44	< 60	L 76	\ 92	l/G12 108	|/G28 124	L 140	\ 156	, 172	< 188	F12 204	F12 220	F12 236	F12 252	Y-POINT PLOT
D	1	1	0	1	CR 13	FOREGND ON FLAG OFF 29	- 45	= 61	M 77	] 93	m/G13 109	}/G29 125	M 141	] 157	- 173	= 189	F13 205	F13 221	F13 237	F13 253	X-POINT PLOT
E	1	1	1	0	A7 ON 14	BACKGND ON FLAG ON 30	. 46	> 62	N 78	^ 94	n/G14 110	~/G30 126	N 142	↑ 158	. 174	> 190	F14 206	F14 222	F14 238	F14 254	CHAR PLOT
F	1	1	1	1	BLINK A7 OFF 15	BLINK ON 31	/ 47	? 63	O 79	_ 95	o/G15 111	■/G31 127	O 143	← 159	/ 175	? 191	F15 207	F15 223	BREAK (F15) 239	F15 255	PLOT ESCAPE

	7	6	5	4	3	2	1	0
STATUS	PLOT	BLINK	B8	B6	B4	B2	B1	B0

NOTE THE TERMINAL ACCEPTS ALL B0 TO BF HEX CODES FROM THE KEYBOARD AND REASSIGNS THEM F0 TO FF HEX WHEN IN THE PLOT MODE, UNLESS THE OPTIONAL KEYS ARE INSTALLED THEREFORE WITHOUT THE FUNCTION KEYS THE KEYBOARD CAN PLOT IN A RANGE OF 0 TO 175.

* COLUMNS 6 AND 7 WILL BE TRANSLATED TO COLUMNS 0 AND 1 RESPECTIVELY IN THE CRT REFRESH RAM IF THE FLAG ON HAS BEEN SET BEFORE ENTERING THESE CODES. THEY WILL THEN APPEAR AS THE SECOND GROUP OF 64 CHARACTERS IF THAT OPTION IS SUPPLIED.

REV 1
REVISED NOV 25 77
C&D

F.2. ASCII VALUES

Decimal	Character	Decimal	Character	Decimal	Character
000	NULL	048	0	096	`
001	AUTO	049	1	097	a
002	PLOT	050	2	098	b
003	CURSOR X,Y	051	3	099	c
004	(not used)	052	4	100	d
005	(not used)	053	5	101	e
006	CCI	054	6	102	f
007	(not used)	055	7	103	g
008	HOME	056	8	104	h
009	TAB	057	9	105	i
010	LINEFEED	058	:	106	j
011	ERASE LINE	059	;	107	k
012	ERASE PAGE	060	<	108	l
013	RETURN	061	=	109	m
014	A7 ON	062	>	110	n
015	BLINK/A7 OFF	063	?	111	o
016	BLACK KEY	064	@	112	p
017	RED KEY	065	A	113	q
018	GREEN KEY	066	B	114	r
019	YELLOW KEY	067	C	115	s
020	BLUE KEY	068	D	116	t
021	MAGENTA KEY	069	E	117	u
022	CYAN KEY	070	F	118	v
023	WHITE KEY	071	G	119	w
024	XMIT	072	H	120	x
025	CURSOR RIGHT	073	I	121	y
026	CURSOR LEFT	074	J	122	z
027	ESC	075	K	123	{
028	CURSOR UP	076	L	124	|
029	FG ON/FLAG OFF	077	M	125	}
030	BG ON/FLAG ON	078	N	126	~
031	BLINK ON	079	O	127	DEL
032	SPACE	080	P		
033	!	081	Q		
034	"	082	R		
035	#	083	S		
036	$	084	T		
037	%	085	U		
038	&	086	V		
039	'	087	W		
040	(	088	X		
041	)	089	Y		
042	*	090	Z		
043	+	091	[		
044	,	092	\		
045	-	093	]		
046	.	094	^		
047	/	095	_		

GENERATION OF DECIMAL CODES FROM THE KEYBOARD

Use keystroke combinations as indicated

CAPS LOCK key locked down at all times

SHIFT required for characters such as ! $ etc.

0	Ctr @	64	@	128	Ctr-Sh @	192	Ctr F0
1	Ctr A	65	A	129	Ctr-Sh A	193	Ctr F1
2	Ctr B	66	B	130	Ctr-Sh B	194	Ctr F2
3	Ctr C	67	C	131	Ctr-Sh C	195	Ctr F3
4	Ctr D	68	D	132	Ctr-Sh D	196	Ctr F4
5	Ctr E	69	E	133	Ctr-Sh E	197	Ctr F5
6	Ctr F	70	F	134	Ctr-Sh F	198	Ctr F6
7	Ctr G	71	G	135	Ctr-Sh G	199	Ctr F7
8	Ctr H	72	H	136	Ctr-Sh H	200	Ctr F8
9	Ctr I	73	I	137	Ctr-Sh I	201	Ctr F9
10	Ctr J	74	J	138	Ctr-Sh J	202	Ctr F10
11	Ctr K	75	K	139	Ctr-Sh K	203	Ctr F11
12	Ctr L	76	L	140	Ctr-Sh L	204	Ctr F12
13	Ctr M	77	M	141	Ctr-Sh M	205	Ctr F13
14	Ctr N	78	N	142	Ctr-Sh N	206	Ctr F14
15	Ctr O	79	O	143	Ctr-Sh O	207	Ctr F15
16	Ctr P	80	P	144	Ctr-Sh P	208	Sh F0
17	Ctr Q	81	Q	145	Ctr-Sh Q	209	Sh F1
18	Ctr R	82	R	146	Ctr-Sh R	210	Sh F2
19	Ctr S	83	S	147	Ctr-Sh S	211	Sh F3
20	Ctr T	84	T	148	Ctr-Sh T	212	Sh F4
21	Ctr U	85	U	149	Ctr-Sh U	213	Sh F5
22	Ctr V	86	V	150	Ctr-Sh V	214	Sh F6
23	Ctr W	87	W	151	Ctr-Sh W	215	Sh F7
24	Ctr X	88	X	152	Ctr-Sh X	216	Sh F8
25	Ctr Y	89	Y	153	Ctr-Sh Y	217	Sh F9
26	Ctr Z	90	Z	154	Ctr-Sh Z	218	Sh F10
27	Ctr [	91	[	155	Ctr-Sh [	219	Sh F11
28	Ctr \	92	\	156	Ctr-Sh \	220	Sh F12
29	Ctr]	93	]	157	Ctr-Sh]	221	Sh F13
30	Ctr ^	94	^	158	Ctr-Sh ^	222	Sh F14
31	Ctr _	95	_	159	Ctr-Sh _	223	Sh F15
32	Sh Ø	96	Sh @	160	Ctr-Sh Ø	224	Ctr-Sh F0
33	!	97	Sh A	161	Ctr !	225	Ctr-Sh F1
34	"	98	Sh B	162	Ctr "	226	Ctr-Sh F2
35	#	99	Sh C	163	Ctr #	227	Ctr-Sh F3
36	$	100	Sh D	164	Ctr $	228	Ctr-Sh F4
37	%	101	Sh E	165	Ctr %	229	Ctr-Sh F5
38	&	102	Sh F	166	Ctr &	230	Ctr-Sh F6
39	'	103	Sh G	167	Ctr '	231	Ctr-Sh F7
40	(	104	Sh H	168	Ctr (	232	Ctr-Sh F8
41	)	105	Sh I	169	Ctr)	233	Ctr-Sh F9
42	*	106	Sh J	170	Ctr *	234	Ctr-Sh F10
43	+	107	Sh K	171	Ctr +	235	Ctr-Sh F11
44	,	108	Sh L	172	Ctr ,	236	Ctr-Sh F12
45	-	109	Sh M	173	Ctr -	237	Ctr-Sh F13
46	.	110	Sh N	174	Ctr .	238	Ctr-Sh F14
47	/	111	Sh O	175	Ctr /	239	Ctr-Sh F15
48	Ø	112	Sh P	176	Ctr Ø	240	F0
49	1	113	Sh Q	177	Ctr 1	241	F1
50	2	114	Sh R	178	Ctr 2	242	F2
51	3	115	Sh S	179	Ctr 3	243	F3
52	4	116	Sh T	180	Ctr 4	244	F4
53	5	117	Sh U	181	Ctr 5	245	F5
54	6	118	Sh V	182	Ctr 6	246	F6
55	7	119	Sh W	183	Ctr 7	247	F7
56	8	120	Sh X	184	Ctr 8	248	F8
57	9	121	Sh Y	185	Ctr 9	249	F9
58	:	122	Sh Z	186	Ctr :	250	F10
59	;	123	Sh [	187	Ctr ;	251	F11
60	<	124	Sh \	188	Ctr <	252	F12
61	=	125	Sh]	189	Ctr =	253	F13
62	>	126	Sh ^	190	Ctr >	254	F14
63	?	127	Sh _	191	Ctr ?	255	F15

F.4 COMPOSITE COLOR CODES
(Used following a PLOT 6)

Background BLACK

Foreground	Plot	Blink
BLACK	0	64
RED	1	65
GREEN	2	66
YELLOW	3	67
BLUE	4	68
MAGENTA	5	69
CYAN	6	70
WHITE	7	71

Background BLUE

Foreground	Plot	Blink
BLACK	32	96
RED	33	97
GREEN	34	98
YELLOW	35	99
BLUE	36	100
MAGENTA	37	101
CYAN	38	102
WHITE	39	103

Background RED

Foreground	Plot	Blink
BLACK	8	72
RED	9	73
GREEN	10	74
YELLOW	11	75
BLUE	12	76
MAGENTA	13	77
CYAN	14	78
WHITE	15	79

Background MAGENTA

Foreground	Plot	Blink
BLACK	40	104
RED	41	105
GREEN	42	106
YELLOW	43	107
BLUE	44	108
MAGENTA	45	109
CYAN	46	110
WHITE	47	111

Background GREEN

Foreground	Plot	Blink
BLACK	16	80
RED	17	81
GREEN	18	82
YELLOW	19	83
BLUE	20	84
MAGENTA	21	85
CYAN	22	86
WHITE	23	87

Background CYAN

Foreground	Plot	Blink
BLACK	48	112
RED	49	113
GREEN	50	114
YELLOW	51	115
BLUE	52	116
MAGENTA	53	117
CYAN	54	118
WHITE	55	119

Background YELLOW

Foreground	Plot	Blink
BLACK	24	88
RED	25	89
GREEN	26	90
YELLOW	27	91
BLUE	28	92
MAGENTA	29	93
CYAN	30	94
WHITE	31	95

Background WHITE

Foreground	Plot	Blink
BLACK	56	120
RED	57	121
GREEN	58	122
YELLOW	59	123
BLUE	60	124
MAGENTA	61	125
CYAN	62	126
WHITE	63	127

Appendix G.1

TMS 8080 Microprocessor

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TMS 8080 MICROPROCESSOR

1. ARCHITECTURE

1.1 INTRODUCTION

The TMS 8080 is an 8-bit parallel central processing unit (CPU) fabricated on a single chip using a high-speed N-channel silicon-gate process. (See Figure 1). A complete microcomputer system with a 2-μs instruction cycle can be formed by interfacing this circuit with any appropriate memory. Separate 8-bit data and 16-bit address buses simplify the interface and allow direct addressing of 65,536 bytes of memory. Up to 256 input and 256 output ports are also provided with direct addressing. Control signals are brought directly out of the processor and all signals, excluding clocks, are TTL compatible.

1.2 THE STACK

The TMS 8080 incorporates a stack architecture in which a portion of external memory is used as a pushdown stack for storing data from working registers and internal machine status. A 16-bit stack pointer (SP) is provided to facilitate stack location in the memory and to allow almost unlimited interrupt handling capability. The CALL and RST (restart) instructions use the SP to store the program counter (PC) into the stack. The RET (return) instruction uses the SP to acquire the previous PC value. Additional instructions allow data from registers and flags to be saved in the stack.

1.3 REGISTERS

The TMS 8080 has three categories of registers: general registers, program control registers, and internal registers. The general registers and program control registers are listed in Table 1. The internal registers are not accessible by the programmer. They include the instruction register, which holds the present instruction, and several temporary storage registers to hold internal data or latch input and output addresses and data.

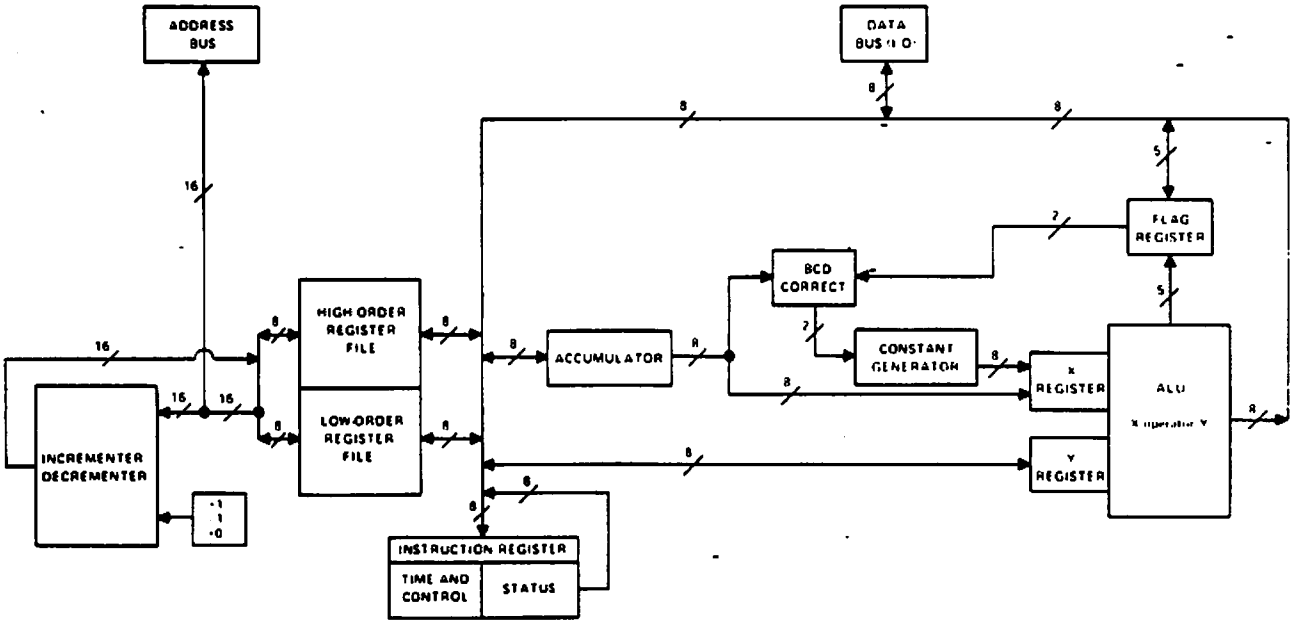


FIGURE 1—TMS 8080 FUNCTIONAL BLOCK DIAGRAM

1.4 THE ARITHMETIC UNIT

Arithmetic operations are performed in an 8-bit parallel arithmetic unit that has both binary and decimal capabilities. Four testable internal flag bits are provided to facilitate program control, and a fifth flag is used for decimal corrections. Table 2 defines these flags and their operation. Decimal corrections are performed with the DAA instruction. The DAA corrects the result of binary arithmetic operation on BCD data as shown in Table 3.

1.5 STATUS AND CONTROL

Two types of status are provided by the TMS8080. Certain status is indicated by dedicated control lines. Additional status is transmitted on the data bus during the beginning of each instruction cycle (machine cycle). Table 4 indicates the pin functions of the TMS8080. Table 5 defines the status information that is presented during the beginning of each machine cycle (SYNC time) on the data bus.

1.6 I/O OPERATIONS

Input/output operations (I/O) are performed using the IN and OUT instructions. The second byte of these instructions indicates the device address (256 device addresses). When an IN instruction is executed, the input device address appears in duplicate on A7 through A0 and A15 through A8, along with $\overline{WO}$ and INP status on the data bus. The addressed input device then puts its input data on the data bus for entry into the accumulator. When an OUT instruction is executed, the same operation occurs except that the data bus has OUT status and then has output data.

Direct memory access channels (DMA) can be OR-tied directly with the data and address buses through the use of the HOLD and HLDA (hold acknowledge) controls. When a HOLD request is accepted by the CPU, HLDA goes high, the address and data lines are forced to a high-impedance or "floating" condition, and the CPU stops until the HOLD request is removed.

Interfacing with different speed memories is easily accomplished by use of the WAIT and READY pins. During each machine cycle, the CPU polls the READY input and enters a wait condition until the READY line becomes true. When the WAIT output pin is high, it indicates that the CPU has entered the wait state.

Designing interrupt driven systems is simplified through the use of vectored interrupts. At the end of each instruction, the CPU polls the INT input to determine if an interrupt request is being made. This action does not occur if the CPU is in the HOLD state or if interrupts are disabled. The INTE output indicates if the interrupt logic is enabled (INTE is high). When a request is honored, the INTA status bit becomes high, and an RST instruction may be inserted to force the CPU to jump to one of eight possible locations. Enabling or disabling interrupts is controlled by special instructions (EI or DI). The interrupt input is automatically disabled when an interrupt request is accepted or when a RESET signal is received.

1.7 INSTRUCTION TIMING

The execution time of the instructions varies depending on the operation required and the number of memory references needed. A machine cycle is defined to be a memory referencing operation and is either 3, 4, or 5 state times long. A state time (designated S) is a full cycle of clocks ϕ1 and ϕ2. (NOTE: The exception to this rule is the DAD instruction, which consists of 1 memory reference in 10 state times). The first machine cycle (designated M1) is either 4 or 5 state times long and is the "instruction fetch" cycle with the program counter appearing on the address bus. The CPU then continues with as many M cycles as necessary to complete the execution of the instruction (up to a maximum of 5). Thus the instruction execution time varies from 4 state times (several including ADDr) to 18 (XTHL). The WAIT or HOLD conditions may affect the execution time since they can be used to control the machine (for example to "single step") and the HALT instruction forces the CPU to stop until an interrupt is received. As the instruction execution is completed (or in the HALT state) the INT pin is polled for an interrupt. In the event of an interrupt, the PC will not he incremented during the next M1 and an RST instruction can be inserted.

TABLE 1
TMS 8080 REGISTERS

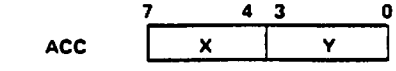
NAME	DESIGNATOR	LENGTH	PURPOSE
Accumulator	A	8	Used for arithmetic, logical, and I/O operations
B Register	B	8	General or most significant 8 bits of double register BC
C Register	C	8	General or least significant 8 bits of double register BC
D Register	D	8	General or most significant 8 bits of double register DE
E Register	E	8	General or least significant 8 bits of double register DE
H Register	H	8	General or most significant 8 bits of double register HL
L Register	L	8	General or least significant 8 bits of double register HL
Program Counter	PC	16	Contains address of next byte to be fetched
Stack Pointer	SP	16	Contains address of the last byte of data saved in the memory stack
Flag Register	F	5	Five flags (C, Z, S, P, C1)

NOTE Registers B and C may be used together as a single 16 bit register. Likewise D and E, and H and L

TABLE 2
FLAG DESCRIPTIONS

SYMBOL	TESTABLE	DESCRIPTION
C	YES	C is the carry/borrow out of the MSB (most significant bit) of the ALU (Arithmetic Logic Unit). A TRUE condition (C = 1) indicates overflow for addition or underflow for subtraction.
Z	YES	A TRUE condition (Z = 1) indicates that the output of the ALU is equal to zero
S	YES	A TRUE condition (S = 1) indicates that the MSB of the ALU output is equal to a one (1)
P	YES	A TRUE condition (P = 1) indicates that the output of the ALU has even parity (the number of bits equal to one is even)
C1	NO	C1 is the carry out of the fourth bit of the ALU (TRUE condition). C1 is used only for BCD correction with the DAA instruction

TABLE 3
FUNCTION OF THE DAA INSTRUCTION
Assume the accumulator (A) contains two BCD digits, X and Y



ACCUMULATOR BEFORE DAA				ACCUMULATOR AFTER DAA			
C	$A_7 \ldots A_4$	C1	$A_3 \ldots A_0$	C	$A_7 \ldots A_4$	C1	$A_3 \ldots A_0$
0	X · 10	0	Y · 10	0	X	0	Y
0	X · 10	1	Y · 10	0	X	0	Y + 6
0	X · 9	0	Y 10	0	X + 1	1	Y + 6
1	X · 10	0	Y · 10	1	X + 6	0	Y
1	X · 10	1	Y 10	1	X + 6	0	Y + 6
1	X 10	0	Y 10	1	X + 7	1	Y + 6
0	X 10	0	Y 10	1	X + 6	0	Y
0	X 10	1	Y 10	1	X + 6	0	Y + 6
0	X 9	0	Y 10	1	X + 7	1	Y + 6

NOTE The corrections shown in Table 3 are sufficient for addition. For subtraction, the programmer must account for the borrow condition that can occur and give erroneous results. The most straight forward method is to set A = 99_{16}, and carry = 1. Then add the minuend to A after subtracting the subtrahend from A.

TABLE 4
TMS 8080 PIN DEFINITIONS

SIGNATURE	PIN	I/O	DESCRIPTION
A15 (MSB)	36	OUT	A15 through A0 comprise the address bus. True memory or I/O device addresses appear on this 3-state bus during the first state time of each instruction cycle.
A14	39	OUT	
A13	38	OUT	
A12	37	OUT	
A11	40	OUT	
A10	1	OUT	
A9	35	OUT	
A8	34	OUT	
A7	33	OUT	
A6	32	OUT	
A5	31	OUT	
A4	30	OUT	
A3	29	OUT	
A2	27	OUT	
A1	26	OUT	
A0 (LSB)	25	OUT	
D7 (MSB)	6	IN/OUT	D7 through D0 comprise the bidirectional 3-state data bus. Memory, status, or I/O data is transferred on this bus.
D6	5	IN/OUT	
D5	4	IN/OUT	
D4	3	IN/OUT	
D3	7	IN/OUT	
D2	8	IN/OUT	
D1	9	IN/OUT	
D0 (LSB)	10	IN/OUT	
V_{SS}	2		Ground reference
V_{BB}	11		Supply voltage (−5 V nominal)
V_{CC}	20		Supply voltage (5 V nominal)
V_{DD}	28		Supply voltage (12 V nominal)
$\phi 1$	22	IN	Phase 1 clock.
$\phi 2$	15	IN	Phase 2 clock. See page 19 for $\phi 1$ and $\phi 2$ timing.
RESET	12	IN	Reset. When active (high) for a minimum of 3 clock cycles, the RESET input causes the TMS 8080 to be reset. PC is cleared, interrupts are disabled, and after RESET, instruction execution starts at memory location 0. To prevent a lockup condition, a HALT instruction must not be used in location 0.
HOLD	13	IN	Hold signal. When active (high) HOLD causes the TMS 8080 to enter a hold state and float (put the 3-state address and data bus in a high-impedance state). The chip acknowledges entering the hold state with the HLDA signal and will not accept interrupts until it leaves the hold state.
INT	14	IN	Interrupt request. When active (high) INT indicates to the TMS8080 that an interrupt is being requested. The TMS8080 polls INT during a HALT or at the end of an instruction. The request will be accepted except when INTE is low or the CPU is in the HOLD condition.
INTE	16	OUT	Interrupts enabled. INTE indicates that an interrupt will be accepted by the TMS 8080 unless it is in the hold state. INTE is set to a high logic level by the EI (Enable Interrupt) instruction and reset to a low logic level by the DI (Disable Interrupt) instruction. INTE is also reset when an interrupt is accepted and by a high on RESET.
DBIN	17	OUT	Data bus in. DBIN indicates whether the data bus is in an input or an output mode. (high = input, low = output).

TABLE 4 (CONTINUED)

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{WR}$	18	OUT	Write. When active (low) $\overline{WR}$ indicates a write operation on the data bus to memory or to an I/O port.
SYNC	19	OUT	Synchronizing control line. When active (high) SYNC indicates the beginning of each machine cycle of the TMS8080. Status information is also present on the data bus during SYNC for external latches.
HLDA	21	OUT	Hold acknowledge. When active (high) HLDA indicates that the TMS8080 is in a hold state.
READY	23	IN	Ready control line. An active (high) level indicates to the TMS 8080 that an external device has completed the transfer of data to or from the data bus. READY is used in conjunction with WAIT for different memory speeds.
WAIT	24	OUT	Wait status. When active (high) WAIT indicates that the TMS8080 has entered a wait state pending a READY signal from memory.

TABLE 5

TMS 8080 STATUS

SIGNATURE	DATA BUS BIT	DESCRIPTION
INTA	D0	Interrupt acknowledge.
$\overline{WO}$	D1	Indicates that current machine cycle will be a read (input) (high = read) or a write (output) (low = write) operation.
STACK	D2	Indicates that address is stack address from the SP.
HLTA	D3	HALT instruction acknowledge.
OUT	D4	Indicates that the address bus has an output device address and the data bus has output data.
M1	D5	Indicates instruction acquisition for first byte.
INP	D6	Indicates address bus has address of input device.
MEMR	D7	Indicates that data bus will be used for memory read data.

2. TMS 8080 INSTRUCTION SET

2.1 INSTRUCTION FORMATS

TMS 8080 instructions are either one, two, or three bytes long and are stored as binary integers in successive memory locations in the format shown below.

One-Byte Instructions

D7 D6 D5 D4 D3 D2 D1 D0	OP CODE

Two-Byte Instructions

D7 D6 D5 D4 D3 D2 D1 D0	OP CODE
D7 D8 D5 D4 D3 D2 D1 D0	OPERAND

Three-Byte Instructions

D7 D6 D5 D4 D3 D2 D1 D0	OP CODE
D7 D6 D5 D4 D3 D2 D1 D0	LOW ADDRESS OR OPERAND 1
D7 D6 D5 D4 D3 D2 D1 D0	HIGH ADDRESS OR OPERAND 2

2.2 INSTRUCTION SET DESCRIPTION

Operations resulting from the execution of TMS 8080 instructions are described in this section. The flags that are affected by each instruction are given after the description.

2.2.1 INSTRUCTION SYMBOLS

SYMBOL	DESCRIPTION
$<b_2>$	Second byte of instruction
$<b_3>$	Third byte of instruction

r_a

Register =	Register Name
000	B
001	C
010	D
011	E
100	H
101	L
111	A

r_b

Register =	Register Name
00	BC
01	DE
10	HL
11	SP

r_c

Register =	Register Name
0	BC
1	DE

r_d

Register =	Register Name
00	BC
01	DE
10	HL

r_{dL}	Least significant 8 bits of r_d
r_{dH}	Most significant 8 bits of r_d
f	Flags — True condition
	Zero (Z) — Result is zero
	Carry (C) — Carry/borrow out of MSB is one
	Parity (P) — Parity of result is even
	Sign (S) — MSB of result is one
	Carry 1(C1) — Carry out of fourth bit is one
M	Memory address defined by registers H and L
( )	Contents of specified address or register
[ ]	Contents at address contained in specified register
←	Is transferred to
↔	Exchange
Am	Bit m of A register (accumulator)
{ }	Flags affected
b_2	Single byte immediate operand
b_3b_2	Double byte immediate operand
$(nnn)_8$	(nnn) is an octal (base 8) number

2.2.2 ACCUMULATOR GROUP INSTRUCTIONS

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
ACI	b_2	2	2/7	(A) ← (A) + b_2 +(carry), add the second byte of the instruction and the contents of the carry flag to register A and place in A. ¦C,Z,S,P,C1¦
ADC	M	1	2/7	(A) ← (A) + (M) + (carry). ¦C,Z,S,P,C1¦
ADC	r_a	1	1/4	(A) ← (A) + (r_a) + (carry). ¦C,Z,S,P,C1¦
ADD	M	1	2/7	(A) ← (A) + (M), add the contents of M to register A and place in A. ¦C,Z,S,P,C1¦
ADD	r_a	1	1/4	(A) ← (A) + (r_a). ¦C,Z,S,P,C1¦
ADI	b_2	2	2/7	(A) ← (A) + b_2 . ¦C,Z,S,P,C1¦
ANA	M	1	2/7	(A) ← (A) AND (M), take the logical AND of M and register A and place in A. The carry flag will be reset low. ¦C,Z,S,P,C1¦
ANA	r_a	1	1/4	(A) ← (A) AND (r_a). ¦C,Z,S,P,C1¦
ANI	b_2	2	2/7	(A) ← (A) AND b_2 . ¦C,Z,S,P,C1¦
CMA		1	1/4	(A) ← ($\bar{A}$), complement A.
CMC		1	1/4	(carry) ← ($\overline{carry}$), complement the carry flag. ¦C¦
CMP	M	1	2/7	(A) − (M), compare the contents of M to register A and set the flags accordingly. ¦C,Z,S,P,C1¦ (A) < (M) Z = 1 (A) ≠ (M) Z = 0 (A) < (M) C = 1 (A) ≥ (M) C = 0
CMP	r_a	1	1/4	(A) − (r_a). ¦C,Z,S,P,C1¦
CPI	b_2	2	2/7	(A) − b_2 . ¦C,Z,S,P,C1¦
DAA		1	1/4	(A) ← BCD correction of (A). The 8 bit A contents is corrected to form two 4 bit BCD digits after a binary arithmetic operation. A fifth flag C1 indicates the overflow from A_3. The carry flag C indicates the overflow from A_7 (See Table 3). ¦C,Z,S,P,C1¦
DAD	r_b	1	1/10	(HL) ← (HL) + (r_b), add the contents of double register r_b to double register HL and place in HL. ¦C¦
LDA	b_3b_2	3	4/13	(A) ← ¦ b_3 b_2 ¦
LDAX	r_c	1	2/7	(A) ← ¦(r_c)¦
ORA	M	1	2/7	(A) ← (A) OR (M), take the logical OR of the contents of M and register A and place in A. The carry flag will be reset. ¦C,Z,S,P,C1¦
ORA	r_a	1	1/4	(A) ← (A) OR (r_a). ¦C,Z,S,P,C1¦
ORI	b_2	2	2/7	(A) ← (A) OR b_2 . ¦C,Z,S,P,C1¦
RAL		1	1/4	A_{m+1}←A_m, A_0← (carry), (carry)← (A_7). Shift the contents of register A to the left one bit through the carry flag. ¦C¦
RAR		1	1/4	A_m←A_{m+1}, A_7← (carry), (carry)← A_0. ¦C¦
RLC		1	1/4	A_{m+1}←A_m, A_0←A_7 (carry)←(A_7). Shift the contents of register A to the left one bit. Shift A_7 into A and into the carry flag. ¦C¦
RRC		1	1/4	A_m←A_{m+1}, A_7← A_0, (carry)← (A_0). ¦C¦

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
SBB	M	1	2/7	(A)←(A)−(M)−(carry), subtract the contents of M and the contents of the carry flag from register A and place in A. Two's complement subtraction is used and a true borrow causes the carry flag to be set (underflow condition). {C,Z,S,P,C1}
SBB	r_a	1	1/4	(A)←(A)−(r_a)−(carry). {C,Z,S,P,C1}
SBI	b_2	2	2/7	(A)←(A)−<b_2>−(carry). {C,Z,S,P,C1}
STA	b_3b_2	3	4/13	[<b_3> <b_2>]←(A), store contents of A in memory address given in bytes 2 and 3.
STAX	r_c	1	2/7	[(r_c)]←(A), store contents of A in memory address given in BC or DE.
STC		1	1/4	(carry)←1, set carry flag to a 1 (true condition).
SUB	M	1	2/7	(A)←(A)−(M), subtract the contents of M from register A and place in A. Two's complement subtraction is used and a true borrow causes the carry flag to be set (underflow condition). {C,Z,S,P,C1}
SUB	r_a	1	1/4	(A)←(A)−(r_a). {C,Z,S,P,C1}
SUI	b_2	2	2/7	(A)←(A)−<b_2>. {C,Z,S,P,C1}
XRA	M	1	2/7	(A)←(A) XOR (M), take the exclusive OR of the contents of M and register A and place in A. The carry flag will be reset. {C,Z,S,P,C1}
XRA	r_a	1	1/4	(A)←(A) XOR (r_a). {C,Z,S,P,C1}
XRI	b_2	2	2/7	(A)←(A) XOR <b_2>. {C,Z,S,P,C1}

2.2.3 INPUT/OUTPUT INSTRUCTIONS

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
IN	b_2	2	3/10	(A)←(input data from data bus), byte 2 is sent on bits A7-A0 and A15-A8 as the input device address. INP status is given on the data bus.
OUT	b_2	2	3/10	(Output data)←(A), byte 2 is sent on bits A7-A0 and A15-A8 as the output device address. OUT status is given on the data bus.

2.2.4 MACHINE INSTRUCTIONS

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
HLT		1	2/7	Halt, all machine operations stop. All registers are maintained. Only an interrupt can return the TMS 8080 to the run mode. Note that a HLT should not be placed in location zero, otherwise after the reset pin is active, the TMS 8080 will enter a nonrecoverable state (until power is removed), i.e., in halt with interrupts disabled. This condition also occurs if a HLT is executed while interrupts are disabled. HLTA status is given on the data bus.
NOP		1	1/4	(PC)←(PC)+1, no operation.

2.2.5 PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
CALL	b_3b_2	3	5/17	[(SP)-1] [(SP)-2]←(PC), (SP)←(SP)-2, (PC)←b_3 b_2, transfer PC to the stack address given by SP, decrement SP twice, and jump unconditionally to address given in bytes 2 and 3.
Conditional call instructions for true flags:				
(f)			5/17 (Pass)	If (f) = 1, [(SP)-1] [(SP)-2]←(PC), (SP)←(SP)-2, (PC)←b_3 b_2, otherwise (PC)←(PC)+3. If the flag specified, f, is 1, then execute a call. Otherwise, execute the next instruction.
CC (carry)	b_3b_2	3	3/11 (Fail)	
CPE (parity)	b_3b_2	3		
CM (sign)	b_3b_2	3		
CZ (zero)	b_3b_2	3		
Conditional call instructions for false flags:				
(f)			5/17 (Pass)	If (f) = 0, [(SP)-1] [(SP)-2]←(PC), (SP)←(SP)-2, (PC)←b_3 b_2, otherwise (PC)←(PC)+3.
CNC (carry)	b_3b_2	3	3/11 (Fail)	
CPO (parity)	b_3b_2	3		
CP (sign)	b_3b_2	3		
CNZ (zero)	b_3b_2	3		
DI		1	1/4	Disable interrupts. INTE is driven false to indicate that no interrupts will be accepted.
EI		1	1/4	Enable interrupts. INTE is driven true to indicate that an interrupt will be accepted. Execution of this instruction is delayed to allow the next instruction to be executed before the INT input is polled.
JMP	b_3b_2	3	3/10	(PC)←b_3 b_2, jump unconditionally to address given in bytes 2 and 3.
Conditional jump instructions for true flags:				
(f)			3/10	If (f) = 1, (PC)←b_3 b_2, otherwise (PC)←(PC)+3. If the flag specified, f, is 1, execute a JMP. Otherwise, execute the next instruction.
JC (carry)	b_3b_2	3		
JPE (parity)	b_3b_2	3		
JM (sign)	b_3b_2	3		
JZ (zero)	b_3b_2	3		
Conditional jump instructions for false flags:				
(f)			3/10	If (f) = 0, (PC)←b_3 b_2, otherwise (PC)←(PC)+3.
JNC (carry)	b_3b_2	3		
JPO (parity)	b_3b_2	3		
JM (sign)	b_3b_2	3		
JNZ (zero)	b_3b_2	3		
PCHL		1	1/5	(PC)←(HL)
POP	PSW	1	3/10	(F)←[(SP)], (A)←[(SP)+1], (SP)←(SP)+2, restore the last stack values addressed by SP into A and F. Increment SP twice.
POP	r_d	1	3/10	(r_{dL})←[(SP)], (r_{dH})←[(SP)+1], (SP)←(SP)+2.
PUSH	PSW	1	3/11	[(SP)-1]←(A), [(SP)-2]←(F), (SP)←(SP)-2, save the contents of A and F into the stack addressed by SP. Decrement SP twice.
PUSH	r_d	1	3/11	[(SP)-1]←(r_{dL}), [(SP)-2]←(r_{dH}), (SP)←(SP)-2.
RET		1	3/10	(PC)←[(SP)] [(SP)+1], (SP)←(SP)+2, return to program at memory address given by last values in the stack. The SP is incremented by two.

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
Conditional return instructions for true flags				
(f)			3/11 (Pass)	If (f) = 1, (PC)← [(SP)] [(SP+1)], (SP)← (SP)+2 If the flag specified, f, is 1, execute a RET. Otherwise, execute the next instruction.
RC (carry)	C	1	1/5 (Fail)	
RPE (parity)	P	1		
RM (sign)	S	1		
RZ (zero)	Z	1		
Conditional return instructions for false flags				
(f)			3/11 (Pass)	If (f) = 0, (PC)← [(SP)] [(SP)+1], (SP)← (SP)+2
RNC (carry)	C	1	1/5 (Fail)	
RPO (parity)	P	1		
RP (sign)	S	1		
RNZ (zero)	Z	1		
RST		1	3/11	[(SP)-1] [(SP)-2] ← (PC) (SP)← (SP)-2, (PC)←0000R0$_8$ where R is a 3 bit field in RST (RST 3R7$_8$). Transfer PC to the stack address given by SP, decrement SP twice, and jump to the address specified by R
SPHL		1	1/5	(SP)← (HL)

2.2.6 REGISTER GROUP INSTRUCTIONS

MNEMONIC	OPERANDS	BYTES	M CYCLES/ STATES	DESCRIPTION
DCR	M	1	3/10	(M)← (M)−1, decrement the contents of memory location specified by H and L {Z,S,P,C1}
DCR	r_a	1	1/5	(r_a)←(r_a)−1, decrement the contents of register r_a. {Z,S,P,C1}
DCX	r_b	1	1/5	(r_b)←(r_b)−1, decrement double registers BC, DE, HL, or SP.
INR	M	1	3/10	(M)← (M)+1, increment the contents of memory location specified by H and L {Z,S,P,C1}
INR	r_a	1	1/5	(r_a)← (r_a)+1, increment the contents of register r_a. {Z,S,P,C1}
INX	r_b	1	1/5	(r_b)← (r_b)+1, increment double registers BC, DE, HL, or SP
LHLD	b_3b_2	3	5/16	(L)← [<b_3><b_2>], (H)← [<b_3><b_2>+1], load registers H and L with contents of the two memory locations specified by bytes 3 and 2
LXI	$r_bb_3b_2$	3	3/10	(r_bH)←<b_3>, (r_bL)← b_2, load double registers BC, DE, HL, or SP immediate with bytes 3, 2, respectively
MVI	M,b_2	2	3/10	(M)← b_2, store immediate byte 2 in the address specified by HL
MVI	r_ab_2	2	2/7	(r_a)←<b_2>, load register r_a immediate with byte 2 of the instruction
MOV	Mr_a	1	2/7	(M)←(r_a), store register r_a in the memory location addressed by H and L
MOV	r_aM	1	2/7	(r_a)←(M), load register r_a with contents of memory addressed by HL.
MOV	$r_{a1}r_{a2}$	1	1/5	(r_{a1})←(r_{a2}), load register r_{a1} with contents of r_{a2}, r_{a2} contents remain unchanged.
SHLD	b_3b_2	3	5/16	[<b_3><b_2>]←(L), [<b_3><b_2>+1]←(H), store the contents of H and L into two successive memory locations specified by bytes 3 and 2.
XCHG		1	1/4	(H)↔(D), (L)↔(E), exchange double registers HL and DE
XTHL		1	5/18	(L)↔[(SP)], (H)↔[(SP)+1], (SP)←(SP), exchange the top of the stack with register HL

2.3 INSTRUCTION SET OPCODES ALPHABETICALLY LISTED

MNEMONIC	BYTES	DESCRIPTION	REGISTER AFFECTED	POSITIVE-LOGIC HEX OPCODE D7–D4	POSITIVE-LOGIC HEX OPCODE D3–D0	CLOCK CYCLES*
ACI	2	Add immediate to A with carry†		C	E	7
ADC M	1	Add memory to A with carry†		8	E	7
ADC r	1	Add register to A with carry†	B	8	8	4
			C	8	9	
			D	8	A	
			E	8	B	
			H	8	C	
			L	8	D	
			A	8	F	
ADD M	1	Add memory to A†		8	6	7
ADD r	1	Add register to A†	B	8	0	4
			C	8	1	
			D	8	2	
			E	8	3	
			H	8	4	
			L	8	5	
			A	8	7	
ADI	2	Add immediate to A†		C	6	7
ANA M	1	AND memory with A†		A	6	7
ANA r	1	AND register with A†	B	A	0	4
			C	A	1	
			D	A	2	
			E	A	3	
			H	A	4	
			L	A	5	
			A	A	7	
ANI	2	AND immediate with A†		E	6	7
CALL	3	Call unconditional		C	D	17
CC	3	Call on carry		D	C	11 17
CM	3	Call on minus		F	C	11 17
CMA	1	Complement A		2	F	4
CMC	1	Complement carry		3	F	4
CMP M	1	Compare memory with A†		B	E	7
CMP r	1	Compare register with A				
			B	B	8	4
			C	B	9	
			D	B	A	
			E	B	B	
			H	B	C	
			L	B	D	
			A	B	F	
CNC	3	Call on no carry		D	4	11 17
CNZ	3	Call on no zero		C	4	11 17
CP	3	Call on positive		F	4	11 17
CPE	3	Call on parity even		E	C	11 17
CPI	2	Compare immediate with A†		F	E	7
CPO	3	Call on parity odd		E	4	11 17
CZ	3	Call on zero		C	C	11 17
DAA	1	Decimal adjust A†		2	7	4

*Two possible cycle times (11 17) indicate instruction cycles dependent on condition flags
†All flags (C, Z, S, P, CY) affected
Only carry flag affected

MNEMONIC	BYTES	DESCRIPTION	REGISTER AFFECTED	POSITIVE-LOGIC HEX OPCODE D7–D4	POSITIVE-LOGIC HEX OPCODE D3–D0	CLOCK CYCLES
DAD B	1	Add B&C to H&L		0	9	10
DAD D	1	Add D&E to H&L		1	9	10
DAD H	1	Add H&L to H&L		2	9	10
DAD SP	1	Add stack pointer to H&L		3	9	10
DCR M	1	Decrement Memory [2]		3	5	10
DCR r	1	Decrement Register [2]	B	0	5	5
			C	0	D	
			D	1	5	
			E	1	D	
			H	2	5	
			L	2	D	
			A	3	D	
DCX B	1	Decrement B&C		0	B	5
DCX D	1	Decrement D&E		1	B	5
DCX H	1	Decrement H&L		2	B	5
DCX SP	1	Decrement stack pointer		3	B	5
DI	1	Disable interrupts		F	3	4
EI	1	Enable interrupts		F	B	4
HLT	1	Halt		7	6	7
IN	2	Input		D	B	10
INR M	1	Increment memory [2]		3	4	10
INR r	1	Increment register [2]	B	0	4	5
			C	0	C	
			D	1	4	
			E	1	C	
			H	2	4	
			L	2	C	
			A	3	C	
INX B	1	Increment B&C register		0	3	5
INX D	1	Increment D&E register		1	3	5
INX H	1	Increment H&L register		2	3	5
INX SP	1	Increment stack pointer		3	3	5
JC	3	Jump on carry		D	A	10
JM	3	Jump on minus		F	A	10
JMP	3	Jump unconditional		C	3	10
JNC	3	Jump on no carry		D	2	10
JNZ	3	Jump on no zero		C	2	10
JP	3	Jump on positive		F	2	10
JPE	3	Jump on parity even		E	A	10
JPO	3	Jump on parity odd		E	2	10
JZ	3	Jump on zero		C	A	10
LDA	1	Load A direct		3	A	13
LDAX B	1	Load A indirect		0	A	7
LDAX D	1	Load A indirect		1	A	7
LHLD	3	Load H&L direct		2	A	16
LXI B	3	Load immediate register pair B&C		0	1	10
LXI D	3	Load immediate register pair D&E		1	1	10
LXI H	3	Load immediate register		2	1	10
LXI SP	3	Load immediate stack pointer		3	1	10

[1] Only carry flag affected
[2] All flags except carry affected

MNEMONIC	BYTES	DESCRIPTION	REGISTER AFFECTED	POSITIVE-LOGIC HEX OPCODE D7-D4	POSITIVE-LOGIC HEX OPCODE D3-D0	CLOCK CYCLES
MOV M,r	1	Move register to memory	B	7	0	7
			C	7	1	
			D	7	2	
			E	7	3	
			H	7	4	
			L	7	5	
			A	7	7	
MOV r,M	1	Move memory to register	B	4	6	7
			C	4	E	
			D	5	6	
			E	5	E	
			H	6	6	
			L	6	E	
			A	7	E	
MOV r_1, r_2	1	Move register to register	B,B	4	0	5
			B,C	4	1	
			B,D	4	2	
			B,E	4	3	
			B,H	4	4	
			B,L	4	5	
			B,A	4	7	
			C,B	4	8	
			C,C	4	9	
			C,D	4	A	
			C,E	4	B	
			C,H	4	C	
			C,L	4	D	
			C,A	4	F	
			D,B	5	0	
			D,C	5	1	
			D,D	5	2	
			D,E	5	3	
			D,H	5	4	
			H,L	5	5	
			D,A	5	7	
			E,B	5	8	
			E,C	5	9	
			E,D	5	A	
			E,E	5	B	
			E,H	5	C	
			E,L	5	D	
			E,A	5	F	
			H,B	6	0	
			H,C	6	1	
			H,D	6	2	
			H,E	6	3	
			H,H	6	4	
			H,L	6	5	
			H,A	6	7	
			L,B	6	8	

MNEMONIC	BYTES	DESCRIPTION	REGISTER AFFECTED	POSITIVE-LOGIC HEX OPCODE D7–D4	POSITIVE-LOGIC HEX OPCODE D3–D0	CLOCK CYCLES*
MOV r_1, r_2	1	Move register to register (continued)	L,C.	6	9	
			L,D	6	A	
			L,E	6	B	
			L,H	6	C	
			L,L	6	D	
			L,A	6	F	
			A,B	7	8	
			A,C	7	9	
			A,D	7	A	
			A,E	7	8	
			A,H	7	C	
			A,L	7	D	
			A,A	7	F	
MVI M	2	Move immediate memory		3	6	10
MVI r	2	Move immediate register	B	0	6	7
			C	0	E	
			D	1	6	
			E	1	E	
			H	2	6	
			L	2	E	
			A	3	E	
NOP	1	No operation	4	0	0	4
ORA M	1	OR memory with A†		B	6	7
ORA r	1	OR register with A†	B	B	0	4
			C	B	1	
			D	B	2	
			E	B	3	
			H	B	4	
			L	B	5	
			A	B	7	
ORI	2	OR immediate with A†		F	6	7
OUT	2	Output		D	3	10
PCHL	1	H&L to program counter		E	9	5
POP B	1	Pop register pair B&C off stack		C	1	10
POP D	1	Pop register pair D&E off stack		D	1	10
POP H	1	Pop register pair H&L off stack		E	1	10
POP PSW	1	Pop A and flags off stack†		F	1	10
PUSH B	1	Push register pair B&C		C	5	11
PUSH D	1	Push register pair D&C		D	5	11
PUSH H	2	Push register pair H&L on stack		E	5	11
PUSH PSW	1	Push A and Flags on stack		F	5	11
RAL	1	Rotate A left through carry‡		1	7	4
RAR	1	Rotate A right through carry‡		1	F	4
RC	1	Return on carry		D	8	5/11
RET	1	Return		C	9	10
RLC	1	Rotate A left‡		0	7	4
RM	1	Return on minus		F	8	5/11
RNC	1	Return on no carry		D	0	5/11
RNZ	1	Return on no zero		C	0	5/11
RP	1	Return on positive		F	0	5/11

*Two possible cycles times (11/17) indicate instruction cycles dependent on condition flags.
†All flags (C, Z, S, P, C1) affected.
‡Only carry flag affected.

MNEMONIC	BYTES	DESCRIPTION	REGISTER AFFECTED	POSITIVE-LOGIC HEX OPCODE D7–D4	POSITIVE-LOGIC HEX OPCODE D3–D0	CLOCK CYCLES*
RPE	1	Return on parity even		E	8	5/11
RPO	1	Return on parity odd		E	0	5/11
RRC	1	Rotate A right[1]		0	F	4
RST	1	Restart				11
			$PC-0000_{16}$	C	7	
			$PC-0008_{16}$	C	F	
			$PC-0010_{16}$	D	7	
			$PC-0018_{16}$	D	F	
			$PC-0020_{16}$	E	7	
			$PC-0028_{16}$	E	F	
			$PC-0030_{16}$	F	7	
			$PC-0038_{16}$	F	F	
RZ	1	Return on Zero		C	8	5/11
SBB M	1	Subtract memory from A with borrow[†]		9	E	7
SBB r	1	Subtract register from A with borrow[†]	B	9	8	4
			C	9	9	
			D	9	A	
			E	9	B	
			H	9	C	
			L	9	D	
			A	9	F	
SBI	2	Subtract immediate from A with borrow[†]		D	E	7
SHLD	3	Store H&L direct		2	2	16
SPHL	1	H&L to stack pointer		F	9	5
STA	3	Store A direct		3	2	13
STAX B	1	Store A indirect		0	2	7
STAX D	1	Store A indirect		1	2	7
STC	1	Set carry[‡]		3	7	4
SUB M	1	Subtract memory from A[†]		9	6	7
SUB r	1	Subtract register from A[†]	B	9	0	4
			C	9	1	
			D	9	2	
			E	9	3	
			H	9	4	
			L	9	5	
			A	9	7	
SUI	2	Subtract immediate from A[†]		D	6	7
XCHG	1	Exchange D&E, H&L registers		E	8	4
XRA M	1	Exclusive OR memory with A[†]		A	E	7
XRA r	1	Exclusive OR register with A[†]	B	A	8	4
			C	A	9	
			D	A	A	
			E	A	B	
			H	A	C	
			L	A	D	
			A	A	F	
XRI	2	Exclusive OR immediate with A[†]		E	E	7
XTHL	1	Exchange top of stack H&L		E	3	18

*Two possible cycles times (11/17) indicate instruction cycles dependent on condition flags

[†]All flags (C, Z, S, P, CY) affected

[‡]Only carry flag affected

3. TMS 8080 ELECTRICAL AND MECHANICAL SPECIFICATIONS

3.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1)	−0.3 V to 20 V
Supply voltage, V_{DD} (see Note 1	−0.3 V to 20 V
Supply voltage, V_{SS} (see Note 1)	−0.3 V to 20 V
All input and output voltages (see Note 1)	−0.3 V to 20 V
Continuous power dissipation	1.5 W
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1 Under absolute maximum ratings voltage values are with respect to the normally most negative supply voltage, V_{BB} (substrate). Throughout the remainder of this data sheet, voltage values are with respect to V_{SS} unless otherwise noted.

3.2 RECOMMENDED OPERATING CONDITIONS

	MIN	NOM	MAX	UNIT
Supply voltage, V_{BB}	−4.75	−5	−5.25	V
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{DD}	11.4	12	12.6	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH} (all inputs except clocks) (see Note 2)	3.3		V_{CC}+1	V
High-level clock input voltage, $V_{IH(\phi)}$	V_{DD}−1		V_{DD}+1	V
Low-level input voltage, V_{IL} (all inputs except clocks) (see Note 3)	−1		0.8	V
Low-level clock input voltage, $V_{IL(\phi)}$ (see Note 3)	−1		0.6	V
Operating free-air temperature, T_A	0		70	C

3.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (UNLESS OTHERWISE NOTED)

PARAMETER		TEST CONDITIONS	MIN	TYP†	MAX	UNIT
I_I	Input current (any input except clocks and data bus)	V_I = 0 V to V_{CC}			±10	μA
$I_{I(\phi)}$	Clock input current	$V_{I(\phi)}$ = 0 V to V_{DD}			±10	μA
$I_{I(DB)}$	Input current, data bus	$V_{I(DB)}$ = 0 V to V_{CC}			−100	μA
$I_{I(hold)}$	Address or data bus input current during hold	$V_{I(ad)}$ or $V_{I(DB)}$ = V_{CC}			10	μA
		$V_{I(ad)}$ or $V_{I(DB)}$ = 0 V			−100	
V_{OH}	High-level output voltage	I_{OH} = 100 μA	3.7			V
V_{OL}	Low-level output voltage	$I_{OL(DB)}$ = 1.7 mA, I_{OL} = 0.75 mA (any output except DB)			0.45	V
$I_{BB(av)}$	Average supply current from V_{BB}	Operating at $t_{c(\phi)}$ = 480 ns, T_A = 25 C		−0.01	−1	mA
$I_{CC(av)}$	Average supply current from V_{CC}			60	75	
$I_{DD(av)}$	Average supply current from V_{DD}			40	67	
C_i	Capacitance, any input except clock	V_{CC} = V_{DD} = V_{SS} = 0 V, V_{BB} = −4.75 to −5.25 V, f = 1 MHz, All other pins at 0 V		10	20	pF
$C_{i(\phi)}$	Clock input capacitance			5	10	
C_o	Output capacitance			10	20	

†All typical values are at T_A = 25 C and nominal voltages

NOTES 2 Active pull up resistors of nominally 2 kΩ will be switched onto the data bus when DBIN is high and the data input voltage is more positive than V_{IH} min

3 The algebraic convention where the most negative limit is designated as minimum is used in this specification for logic voltage levels only

3.4 TIMING REQUIREMENTS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (SEE FIGURE 2)

		MIN	MAX	UNIT
$t_{c(\phi)}$	Clock cycle time (see Note 5)	480	2000	ns
$t_{r(\phi)}$	Clock rise time	5	50	ns
$t_{f(\phi)}$	Clock fall time	5	50	ns
$t_{w(\phi 1)}$	Pulse width, clock 1 high	60		ns
$t_{w(\phi 2)}$	Pulse width, clock 2 high	220		ns
$t_{d(\phi 1L-\phi 2)}$	Delay time, clock 1 low to clock 2	0		ns
$t_{d(\phi 2-\phi 1)}$	Delay time, clock 2 to clock 1	70		ns
$t_{d(\phi 1H-\phi 2)}$	Delay time, clock 1 high to clock 2 (time between leading edges)	130		ns
$t_{su(da-\phi 1)}$	Data setup time with respect to clock 1	50		ns
$t_{su(da-\phi 2)}$	Data setup time with respect to clock 2	150		ns
$t_{su(hold)}$	Hold input setup time	140		ns
$t_{su(int)}$	Interrupt input setup time	180		ns
$t_{su(rdy)}$	Ready input setup time	120		ns
$t_{h(da)}$	Data hold time (see Note 6)	$t_{PD(DBI)}$		ns
$t_{h(hold)}$	Hold input hold time	0		ns
$t_{h(int)}$	Interrupt input hold time	0		ns
$t_{h(rdy)}$	Ready input hold time	0		ns

NOTES 5. $t_{c(\phi)} + t_{d(\phi 1L-\phi 2)} + t_{r(\phi 2)} + t_{w(\phi 2)} + t_{f(\phi 2)} + t_{d(\phi 2-\phi 1)} + t_{r(\phi 1)} \geq 480$ ns, $t_{c(\phi)} \leq 2000$ ns

6. The data input should be enabled using the DBIN status signal. No bus conflict can then occur and the data hold time requirement is thus assured.

3.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (SEE FIGURE 2)

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
$t_{PD(ad)}$	Propagation delay time, clock 2 to address outputs	C_L = 100 pF, R_L = 1.3 kΩ		200	ns
$t_{PD(da)}$	Propagation delay time, clock 2 to data bus			220	ns
$t_{PD(cont)}$	Propagation delay time, clocks to control outputs			120	ns
$t_{PD(DBI)}$	Propagation delay time, clock 2 to DBIN output		25	140	ns
$t_{PD(int)}$	Propagation delay time, clock 2 to INTE output			200	ns
t_{DI}	Time for data bus to enter input mode			$t_{PD(DBI)}$	ns
t_{PXZ}	Disable time to high-impedance state during hold (address outputs and data bus)			120	ns

The time that the address outputs and output data will remain stable after $\overline{WR}$ goes high, t_{WA} and $t_{WD} = t_{d(\phi 1H-\phi 2)}$

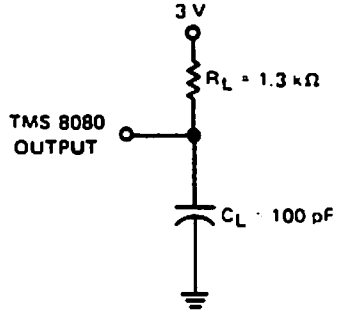
The time between address outputs becoming stable and WR going low, $t_{AW} = 2\,t_{c(\phi)} - t_{d(\phi 1H-\phi 2)} - t_{r(\phi)} \geq 120$ ns

The time between output data becoming stable and $\overline{WR}$ going low, $t_{DW} = t_{c(\phi)} - t_{d(\phi 1H-\phi 2)} - t_{r(\phi)} \geq 150$ ns

The following are relevant when interfacing to devices requiring V_{IH} min of 3.3 V

a) Maximum output rise time (t_{TLH}) from 0.8 V to 3.3 V is 140 ns with C_L as specified for the propagation delay times above.

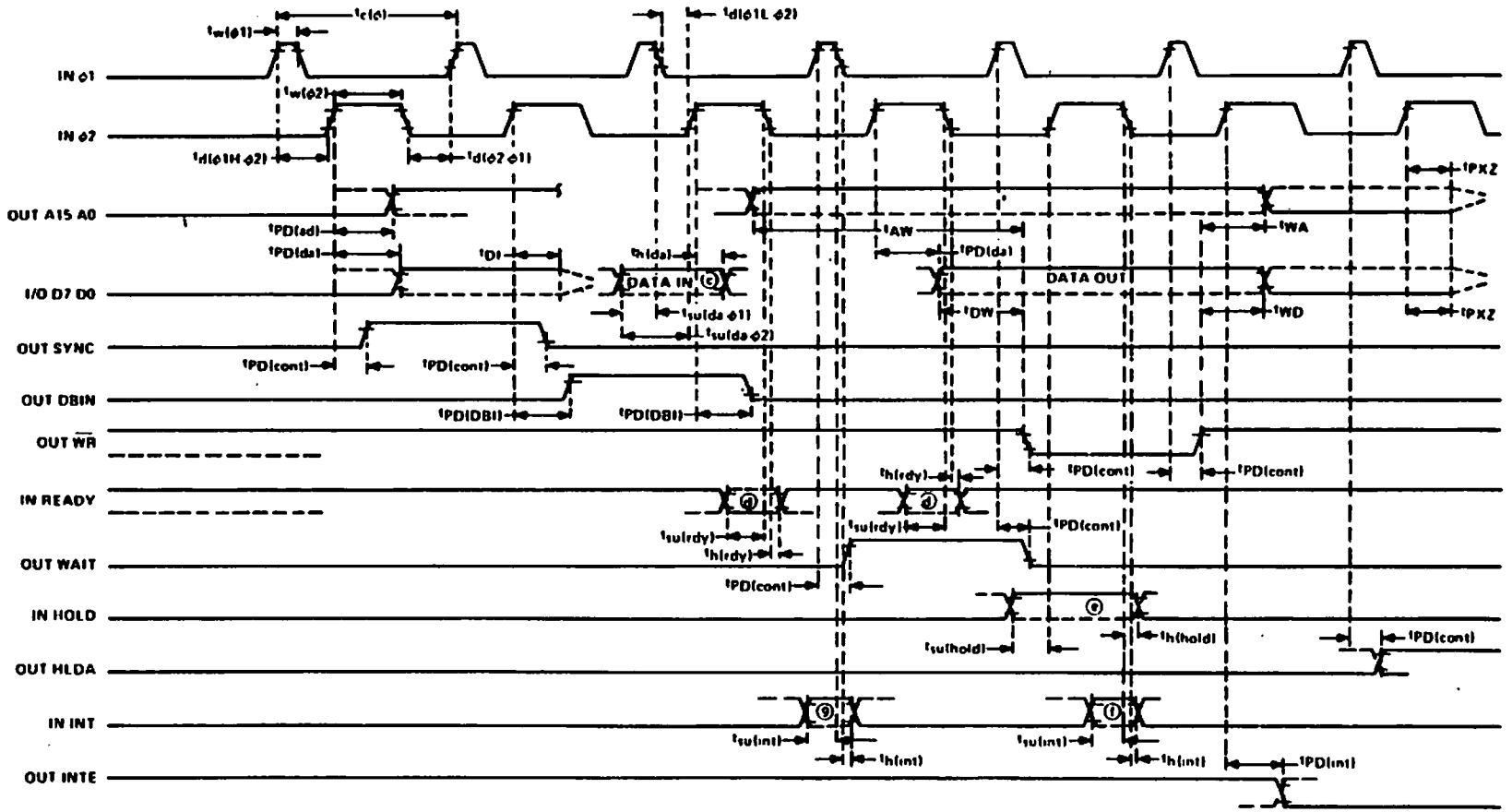
b) Maximum propagation delay times when measured to $V_{ref(H)}$ = 3 V (instead of 2 V) will be 60 ns more than as specified above with C_L as specified.



C_L includes probe and jig capacitance

LOAD CIRCUIT

voltage waveforms (see notes a and b)



NOTES: a. This timing diagram shows timing relationships only, it does not represent any specific machine cycle.

b. Time measurements are made at the following reference voltages: Clock, $V_{ref(H)}$ = 9.5 V, $V_{ref(L)}$ = 1 V. Other inputs, $V_{ref(H)}$ = 2 V, $V_{ref(L)}$ = 0.8 V.

c. Data in must be stable for this period when DBIN is high during S3. Requirements for both $t_{su(da\ \phi1)}$ and $t_{su(da\ \phi2)}$ must be satisfied.

d. The ready signal must be stable for this period during S2 or SW. This requires external synchronization.

e. The hold signal must be stable for this period during S2 or SW when entering the hold mode and during S3, S4, S5 and SWH when in the hold mode. This requires external synchronization.

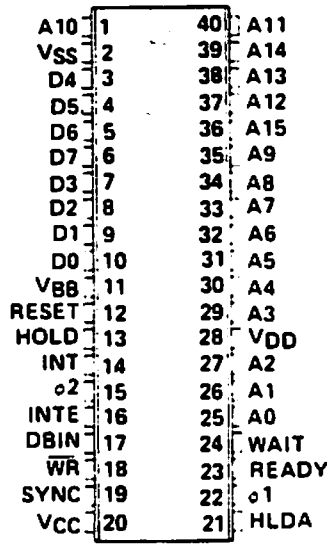
f. The interrupt signal must be stable during this period on the last clock cycle of any instruction to be recognized on the following instruction. External synchronization is not required.

g. During halt mode only, timing is with respect to the clock 1 falling edge.

FIGURE 2

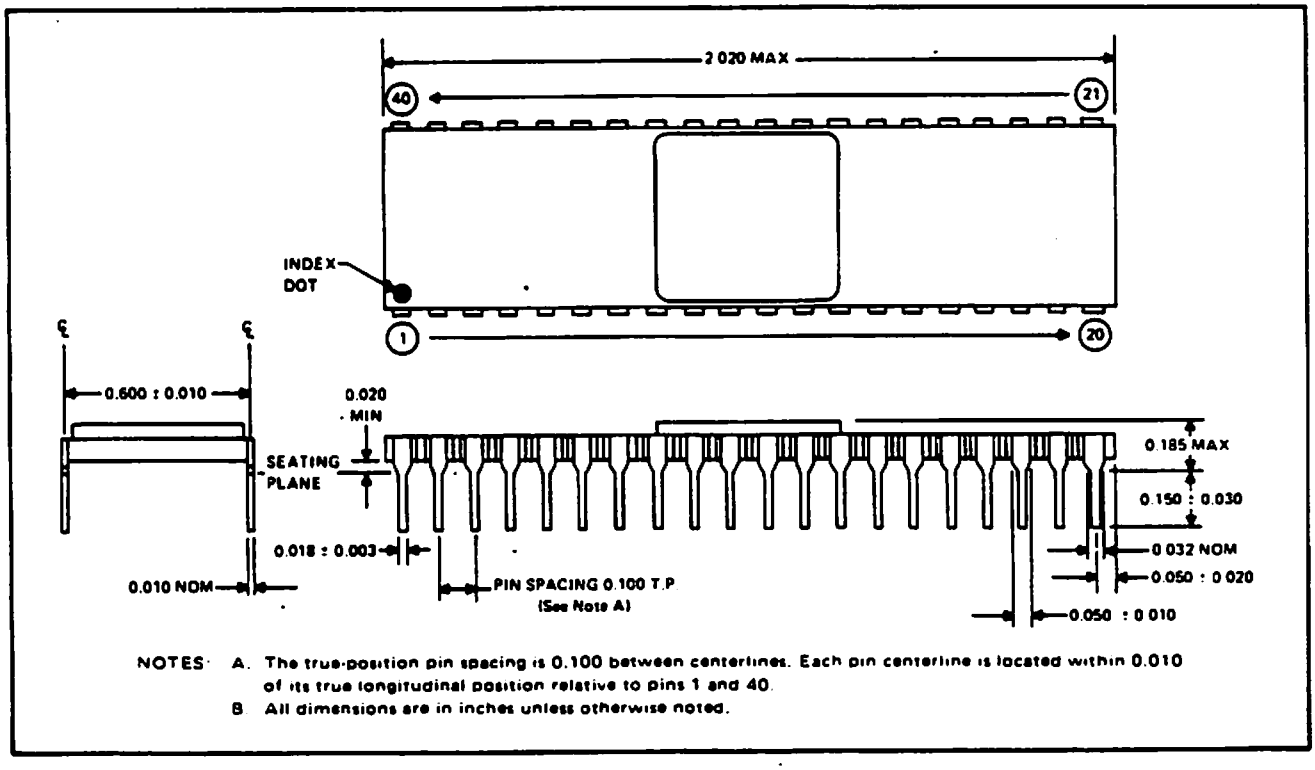
3.6 TERMINAL ASSIGNMENTS

TMS 8080



3.7 MECHANICAL DATA

40-PIN CERAMIC PACKAGE



NOTES: A. The true-position pin spacing is 0.100 between centerlines. Each pin centerline is located within 0.010 of its true longitudinal position relative to pins 1 and 40.

B. All dimensions are in inches unless otherwise noted.

Appendix G.2

TMS 5501 Multifunction Input/Output Controller

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TMS 5501 MULTIFUNCTION INPUT/OUTPUT CONTROLLER

1. INTRODUCTION

1.1 DESCRIPTION

The TMS 5501 is a multifunction input/output circuit for use with TI's TMS 8080 CPU. It is fabricated with the same N-channel silicon-gate process as the TMS 8080 and has compatible timing, signal levels, and power supply requirements. The TMS 5501 provides a TMS 8080 microprocessor system with an asynchronous communications interface, data I/O buffers, interrupt control logic, and interval timers.

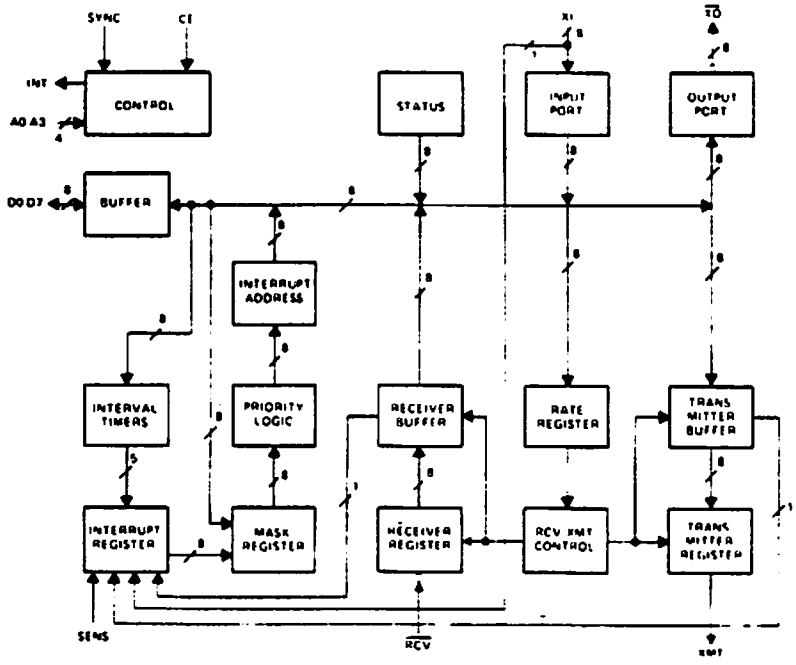


FIGURE 1–TMS 5501 BLOCK DIAGRAM

The I/O section of the TMS 5501 contains an eight-bit parallel input port and a separate eight-bit parallel output port with storage register. Five programmable interval timers provide time intervals from 64 μs to 16.32 ms.

The interrupt system allows the processor to effectively communicate with the interval timers, external signals, and the communications interface by providing TMS 8080-compatible interrupt logic with masking capability.

Data transfers between the TMS 5501 and the CPU are carried by the data bus and controlled by the interrupt, chip enable, sync, and address lines. The TMS 8080 uses four of its memory-address lines to select one of 14 commands to which the TMS 5501 will respond. These commands allow the CPU to:

- read the receiver buffer
- read the input port
- read the interrupt address
- read TMS 5501 status
- issue discrete commands
- load baud rate register
- load the transmitter buffer
- load the output port
- load the mask register
- load an interval timer

The commands are generated by executing memory referencing instructions such as MOV (register to memory) with the memory address being the TMS 5501 command. This provides a high degree of flexibility for I/O operations by letting the systems programmer use a variety of instructions.

1.2 SUMMARY OF OPERATION

Addressing the TMS 5501

A convenient method for addressing the TMS 5501 is to tie the chip enable input to the highest order address line of the CPU's 16-bit address bus and the four TMS 5501 address inputs to the four lowest order bits of the bus. This, of course, limits the system to 32,768 words of memory but in many applications the full 65,536 word memory addressing capability of the TMS 8080 is not required.

Communications Functions

The communications section of the TMS 5501 is an asynchronous transmitter and receiver for serial communications and provides the following functions:

Programmable baud rate – A CPU command selects a baud rate of 110, 150, 300, 1200, 2400, 4800, or 9600 baud.

Incoming character detection – The receiver detects the start and stop bits of an incoming character and places the character in the receive buffer.

Character transmission – The transmitter generates start and stop bits for a character received from the CPU and shifts it out.

Status and command signals – Via the data bus, the TMS 5501 signals the status of: framing error and overrun error flags; data in the receiver and transmitter buffers; start and data bit detectors; and end-of-transmission (break) signals from external equipment. It also issues break signals to external equipment.

Data Interface

The TMS 5501 moves data between the CPU and external devices through its internal data bus, input port, and output port. When data is present on the bus that is to be sent to an external device, a Load Output Port (LOP) command from the CPU puts the data on the $\overline{XO}$ pins of the TMS 5501 by latching it in the output port. The data remains in the port until another LOP command is received. When the CPU requires data that is present on the External Input (XI) lines, it issues a command that gates the data onto the internal data bus of the TMS 5501 and consequently onto the CPU's data bus at the correct time during the CPU cycles.

Interval Timers

To start a countdown by any of the five interval timers, the program selects the particular timer by an address to the TMS 5501 and loads the required interval into the timer via the data bus. Loading the timer activates it and it counts down in increments of 64 microseconds. The 8-bit counters provide intervals that vary in duration from 64 to 16,320 microseconds. Much longer intervals can be generated by cascading the timers through software. When a timer reaches zero, it generates an interrupt that typically will be used to point to a subroutine that performs a servicing function such as polling a peripheral or scanning a keyboard. Loading an interval value of zero causes an immediate interrupt. A new value loaded while the interval timer is counting overrides the previous value and the interval timer starts counting down the new interval. When an interval timer reaches zero it remains inactive until a new interval is loaded.

Servicing Interrupts

The TMS 5501 provides a TMS 8080 system with several interrupt control functions by receiving external interrupt signals, generating interrupt signals, masking out undersired interrupts, establishing the priority of interrupts, and generating RST instructions for the TMS 8080. An external interrupt is received on pin 22, SENS. An additional external interrupt can be received on pin 32, XI7, if selected by a discrete command from the TMS 8080 (See Figure 4). The TMS 5501 generates an interrupt when any of the five interval timers count to zero. Interrupts are also generated when the receiver buffer is loaded and when the transmitter buffer is empty.

When an interrupt signal is received by the interrupt register from a particular source, a corresponding bit is set and gated to the mask register. A pattern will have previously been set in the mask register by a load-mask-register command from the TMS 8080. This pattern determines which interrupts will pass through to the priority logic. The priority logic allows an interrupt to generate an RST instruction to the TMS 8080 only if there is no higher priority interrupt that has not been accepted by the TMS 8080. The TMS 5501 prioritizes interrupts in the order shown below:

1st – Interval Timer =1
2nd – Interval Timer =2
3rd – External Sensor
4th – Interval Timer =3
5th – Receiver Buffer Loaded
6th – Transmitter Buffer Emptied
7th – Interval Timer =4
8th – Interval Timer =5 or an External Input (XI 7)

The highest priority interrupt passes through to the interrupt address logic, which generates the RST instruction to be read by the TMS 8080. See Table 3 for relationship of interrupt sources to RST instructions and Figures 6 and 8 for timing relationships.

The TMS 5501 provides two methods of servicing interrupts; an interrupt-driven system or a polled-interrupt system. In an interrupt-driven system, the INT signal of the TMS 5501 is tied to the INT input of the TMS 8080. The sequence of events will be: (1) The TMS 5501 receives (or generates) an interrupt signal and readies the appropriate RST instruction. (2) The TMS 5501 INT output, tied to the TMS 8080 INT input, goes high signaling the TMS 8080 that an interrupt has occured. (3) If the TMS 8080 is enabled to accept interrupts, it sets the INTA (interrupt acknowledge) status bit high at SYNC time of the next machine cycle. (4) If the TMS 5501 has previously received an interrupt-acknowledge-enable command from the CPU (see Bit 3, Paragraph 2.2.5), the RST instruction is transferred to the data bus.

In a polled-interrupt system, INT is not used and the sequence of events will be: (1) The TMS 5501 receives (or generates) an interrupt and readies the RST instruction. (2) The TMS 5501 interrupt-pending status bit (see Bit 5, Paragraph 2.2.4) is set high (the interrupt-pending status bit and the INT output go high simultaneously). (3) At the prescribed time, the TMS 8080 polls the TMS 5501 to see if an interrupt has occurred by issuing a read-TMS 5501-status command and reading the interrupt-pending bit. (4) If the bit is high, the TMS 8080 will then issue a read-interrupt-address command, which causes the TMS 5501 to transfer the RST instruction to the data bus as data for the instruction being executed by the TMS 8080.

1.3 APPLICATIONS

Communications Terminals

The functions of the TMS 5501 make it particularly useful in TMS 8080-based communications terminals and generally applicable in systems requiring periodic or random servicing of interrupts, generation of control signals to external devices, buffering of data, and transmission and reception of asynchronous serial data. As an example, a system configuration such as shown in Figure 2 can function as the controller for a terminal that governs employee entrance into a plant or security areas within a plant. Each terminal is identified by a central computer through ID switches. The central system supplies each terminal's RAM with up to 16 employee access categories applicable to that terminal. These categories are compared with an employee's badge character when he inserts his badge into the badge sensor. If a

match is not found, a reject light will be activated. If a match is found, the terminal will transmit the employee's badge number and access category to the central system, and a door unlock solenoid will be activated for 4 seconds. The central computer then may take the transmitted information and record it along with time and date of access.

The TMS 4700 is a 1024 x 8 ROM that contains the system program, and the TMS 4036 is a 64 x 8 RAM that serves as the stack for the TMS 8080 and storage for the access category information. TTL circuits control chip-enable information carried by the address bus. Signals from the CPU gate the address bits from the ROM, the RAM, or the TMS 5501 onto the data bus at the correct time in the CPU cycle. The clock generator consists of four TTL circuits along with a crystal, needed to maintain accurate serial data assembly and disassembly with the central computer.

The TMS 5501 handles the asynchronous serial communication between the TMS 8080 and the central system and gates data from the badge reader onto the data bus. It also gates control and status data from the TMS 8080 to the door lock and badge reader and controls the time that the door lock remains open. The TMS 5501 signals the TMS 8080 when the badge reader or the communication lines need service. The functions that the TMS 5501 is to perform are selected by an address from the TMS 8080 with the highest order address line tied to the TMS 5501 chip enable input and the four lowest order lines tied to the address inputs.

2. OPERATIONAL AND FUNCTIONAL DESCRIPTION

This detailed description of the TMS 5501 consists of:

INTERFACE SIGNALS – a definition of each of the circuit's external connections

COMMANDS – the address required to select each of the TMS 5501 commands and a description of the response to the command.

2.1 INTERFACE SIGNALS

The TMS 5501 communicates with the TMS 8080 via four address lines: a chip enable line, an eight-bit bidirectional data bus, an interrupt line, and a sync line. It communicates with system components other than the CPU via eight external inputs, eight external outputs, a serial receiver input, a serial transmitter output, and an external sensor input. Table 1 defines the TMS 5501 pin assignments and describes the function of each pin.

TABLE 1
TMS 5501 PIN ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	DESCRIPTION
		INPUTS
CE	18	Chip enable–When CE is low, the TMS 5501 address decoding is inhibited, which prevents execution of any of the TMS 5501 commands.
A3	17	Address bus–A3 through A0 are the lines that are addressed by the TMS 8080 to select a particular
A2	16	TMS 5501 function.
A1	15	
A0	14	
SYNC	19	Synchronizing signal–The SYNC signal is issued by the TMS 8080 and indicates the beginning of a machine cycle and availability of machine status. When the SYNC signal is active (high), the TMS 5501 will monitor the data bus bits D0 (interrupt acknowledge) and D1 ($\overline{WO}$, data output function).
$\overline{RCV}$	5	Receiver serial data input line–$\overline{RCV}$ must be held in the inactive (high) state when not receiving data. A transition from high to low will activate the receive circuitry.

TABLE 1 (continued)
TMS 5501 PIN ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	DESCRIPTION
		INPUTS
XI 0	39	External inputs—These eight external inputs are gated to the data bus when the read-external-inputs function is addressed. External input n is gated to data bus bit n without conversion.
XI 1	38	
XI 2	37	
XI 3	36	
XI 4	35	
XI 5	34	
XI 6	33	
XI 7	32	
SENS	22	External interrupt sensing – A transition from low to high at SENS sets a bit in the interrupt register, which, if enabled, generates an interrupt to the TMS 8080.
		OUTPUTS
$\overline{XO}$ 0	24	External outputs—These eight external outputs are driven by the complement of the output register; i.e., if output register bit n is loaded with a high (low) from data bus bit n by a load-output register command, the external output n will be a low (high). The external outputs change only when a load-output-register function is addressed.
$\overline{XO}$ 1	25	
$\overline{XO}$ 2	26	
$\overline{XO}$ 3	27	
$\overline{XO}$ 4	28	
$\overline{XO}$ 5	29	
$\overline{XO}$ 6	30	
$\overline{XO}$ 7	31	
XMT	40	Transmitter serial data output line—This line remains high when the TMS 5501 is not transmitting.
		DATA BUS INPUT/OUTPUT
D0	13	Data bus – Data transfers between the TMS 5501 and the TMS 8080 are made via the 8-bit bidirectional data bus. D0 is the LSB. D7 is the MSB.
D1	12	
D2	11	
D3	10	
D4	9	
D5	8	
D6	7	
D7	6	
INT	23	Interrupt—When active (high), the INT output indicates that at least one of the interrupt conditions has occurred and that its corresponding mask-register bit is set.
		POWER AND CLOCKS
V_{SS}	4	Ground reference
V_{BB}	1	Supply voltage (−5 V nominal)
V_{CC}	2	Supply voltage (5 V nominal)
V_{DD}	3	Supply voltage (12 V nominal)
$\phi 1$	20	Phase 1 clock
$\phi 2$	21	Phase 2 clock

2.2 TMS 5501 COMMANDS

The TMS 5501 operates as input/output device for the TMS 8080. Functions are initiated via the TMS 8080 address bus and the TMS 5501 address inputs. Address decoding to determine the command function being issued is defined in Table 2.

TABLE 2
COMMAND ADDRESS DECODING
When Chip Enable Is High

#2 5501 PORT NO.	#1 5501 PORT NO.	A3	A2	A1	A0	COMMAND	FUNCTION	PARAGRAPH
16	0	L	L	L	L	Read receiver buffer	RBn → Dn	2.2.1
17	1	L	L	L	H	Read external inputs	XIn → Dn	2.2.2
18	2	L	L	H	L	Read interrupt address	RST → Dn	2.2.3
19	3	L	L	H	H	Read TMS 5501 status	(Status) → Dn	2.2.4
20	4	L	H	L	L	Issue discrete commands	See Fig.4	2.2.5
21	5	L	H	L	H	Load rate register	See Fig.4	2.2.6
22	6	L	H	H	L	Load transmitter buffer	Dn → TBn	2.2.7*
23	7	L	H	H	H	Load Output port	Dn → XOn	2.2.8
24	8	H	L	L	L	Load mask register	Dn → MRn	2.2.9
25	9	H	L	L	H	Load interval timer 1	Dn → Timer 1	2.2.10
26	10	H	L	H	L	Load interval timer 2	Dn → Timer 2	2.2.10
27	11	H	L	H	H	Load interval timer 3	Dn → Timer 3	2.2.10
28	12	H	H	L	L	Load interval timer 4	Dn → Timer 4	2.2.10
29	13	H	H	L	H	Load interval timer 5	Dn → Timer 5	2.2.10
30	14	H	H	H	L	No function		
31	15	H	H	H	H	No function		

* Important

RBn Receiver buffer bit n
Dn Data bus I/O terminal n
XIn External input terminal n
RST 11 (IA_5) (IA_4) (IA_3) 1 1 1 (see Table 3)
TBn Transmit buffer bit n
XOn Output register bit n
MRn Mask register bit n

TABLE 3
RST INSTRUCTIONS

DATA BUS BIT 0	1	2	3	4	5	6	7	INTERRUPT CAUSED BY	#1 TMS 5501
H	H	H	L	L	L	H	H	Interval Timer 1	Power Up
H	H	H	H	L	L	H	H	Interval Timer 2	User Timer
H	H	H	L	H	L	H	H	External Sensor	Keyboard
H	H	H	H	H	L	H	H	Interval Timer 3	Repeat Key
H	H	H	L	L	H	H	H	Receiver Buffer	Rx RS-232
H	H	H	H	L	H	H	H	Transmitter Buffer	Tx RS-232
H	H	H	L	H	H	H	H	Interval Timer 4	Bell Timer
H	H	H	H	H	H	H	H	Interval Timer 5 of X17	CRT Executive Loop

The following paragraphs define the functions of the TMS 5501 commands.

2.2.1 Read receiver buffer

Addressing the read-receiver-buffer function causes the receiver buffer contents to be transferred to the TMS 8080 and clears the receiver-buffer-loaded flag.

2.2.2 Read external input lines

Addressing the read-external-inputs function transfers the states of the eight external input lines to the TMS 8080

2.2.3 Read interrupt address

Addressing the read interrupt address function transfers the current highest priority interrupt address onto the data bus as read data. After the read operation is completed, the corresponding bit in the interrupt register is reset.

If the read-interrupt-address function is addressed when there is no interrupt pending, a false interrupt address will be read. TMS 5501 status function should be addressed in order to determine whether or not an interrupt condition is pending.

2.2.4 Read TMS 5501 status

Addressing the read-TMS 5501-status function gates the various status conditions of the TMS 5501 onto the data bus. The status conditions, available as indicated in Figure 3, are described in the following paragraphs.

BIT:	7	6	5	4	3	2	1	0
	START BIT DETECT	FULL BIT DETECT	INTRPT PENDING	XMIT BUFFER EMPTY	RCV BUFFER LOADED	SERIAL RCVD	OVERRUN ERROR	FRAME ERROR

FIGURE 3–DATA BUS ASSIGNMENTS FOR TMS 5501 STATUS

Bit 0, framing error

A high in bit 0 indicates that a framing error was detected on the last character received (either one or both stop bits were in error). The framing error flag is updated at the end of each character. Bit 0 of the TMS 5501 status will remain high until the next valid character is received.

Bit 1, overrun error

A high in bit 1 indicates that a new character was loaded into the receiver buffer before a previous character was read out. The overrun error flag is cleared each time the read-I/O-status function is addressed or a reset command is issued.

Bit 2, serial received data

Bit 2 monitors the receiver serial data input line. This line is provided as a status input for use in detecting a break and for test purposes. Bit 2 is normally high when no data is being received.

Bit 3, receiver buffer loaded

A high in bit 3 indicates that the receiver buffer is loaded with a new character. The receiver-buffer-loaded flag remains high until the read-receiver-buffer function is addressed (at which time the flag is cleared). The reset function also clears this flag.

Bit 4, transmitter buffer empty
A high in bit 4 indicates that the transmitter buffer register is empty and ready to accept a character. Note, however, that the serial transmitter register may be in the process of shifting out a character. The reset function sets the transmitter-buffer-empty flag high.

Bit 5, interrupt pending
A high in bit 5 indicates that one or more of the interrupt conditions has occured and the corresponding interrupt is enabled. This bit is the status of the interrupt signal INT.

Bit 6, full bit detected
A high in bit 6 indicates that the first data bit of a receive-data character has been detected. This bit remains high until the entire character has been received or until a reset is issued and is provided for test purposes.

Bit 7, start bit detected
A high in bit 7 indicates that the start bit of an incoming data character has been detected. This bit remains high until the entire character has been received or until a reset is issued and is provided for test purposes.

2.2.5 Issue discrete commands

Addressing the discrete command function causes the TMS 5501 to interpret the data bus information according to the following descriptions. See Figure 4 for the discrete command format. Bits 1 through 5 are latched until a different discrete command is received.

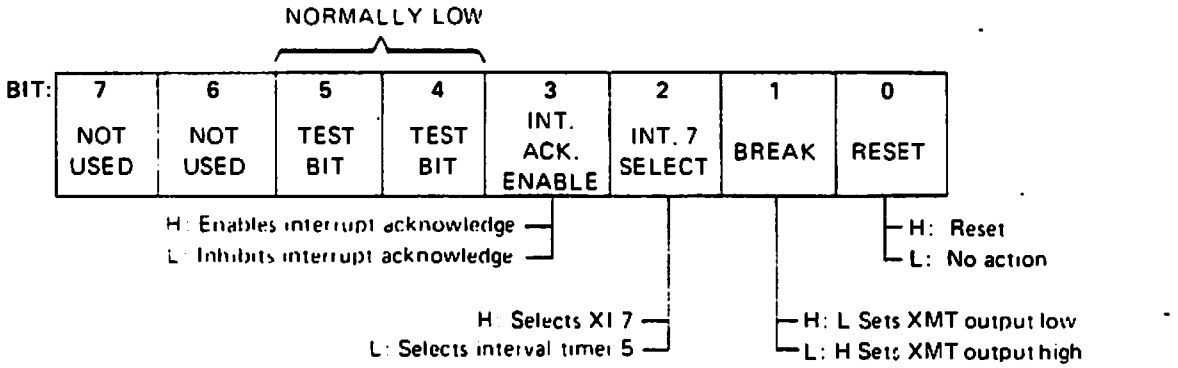


FIGURE 4–DISCRETE COMMAND FORMAT

Bit 0, reset
A high in bit 0 will cause the following:

1) The receiver buffer and register are cleared to the search mode including the receiver-buffer-loaded flag, the start-bit-detected flag, the full-bit-detected flag, and the overrun-error flag. The receiver buffer is not cleared and will contain the last character received.
2) The transmitter data output is set high (marking). The transmitter-buffer-empty flag is set high indicating that the transmitter buffer is ready to accept a character from the TMS 8080.
3) The interrupt register is cleared except for the bit corresponding to the transmitter buffer interrupt, which is set high.
4) The interval timers are inhibited.

A low in bit 0 causes no action. The reset function has no affect on the output port, the external inputs, interrupt acknowledge enable, the mask register, the rate register, the transmitter register, or the transmitter buffer.

Bit 1, break
A low in bit 1 causes the transmitter data output to be reset low (spacing).

If bit 0 and bit 1 are both high, the reset function will override.

Bit 2, interrupt 7 select
Interrupt 7 may be generated either by a low to high transition of external input 7 or by interval timer 5.

A high in bit 2 selects the interrupt 7 source to be the transition of external input 7. A low in bit 2 selects the interrupt 7 source to be interval timer 5.

Bit 3, interrupt acknowledge enable
The TMS 5501 decodes data bus (CPU status) bit 0 at SYNC of each machine cycle to determine if an interrupt acknowledge is being issued.

A high in bit 3 enables the TMS 5501 to accept the interrupt acknowledge decode. A low in bit 3 causes the TMS 5501 to ignore the interrupt acknowledge decode.

Bit 4 and bit 5 are used only during testing of the TMS 5501. For correct system operation both bits must be kept low.

Bit 6 and bit 7 are not used and can assume any value.

2.2.6 Load rate register

Addressing the load-rate-register function causes the TMS 5501 to load the rate register from the data bus and interpret the data bits (See Figure 5) as follows.

BIT:	7	6	5	4	3	2	1	0
	STOP BIT(s)	9600 baud	4800 baud	2400 baud	1200 baud	300 baud	150 baud	110 baud

H: One stop bit
L: Two stop bits

FIGURE 5–DATA BUS ASSIGNMENTS FOR RATE COMMANDS

Bits 0 through 6, rate select

The rate select bits (bits 0 through 6) are mutually exclusive, i.e., only one bit may be high. A high in bits 0 through 6 will select the baud rate for both the transmitter and receiver circuitry as defined below and in Figure 5:

Bit 0 110 baud
Bit 1 150 baud
Bit 2 300 baud
Bit 3 1200 baud
Bit 4 2400 baud
Bit 5 4800 baud
Bit 6 9600 baud

If more than one bit is high, the highest rate indicated will result. If bits 0 through 6 are all low, both the receiver and the transmitter circuitry will be inhibited.

Bit 7, stop bits

Bit 7 determines whether one or two stop bits are to be used by both the transmitter and receiver circuitry. A high in bit 7 selects one stop bit. A low in bit 7 selects two stop bits.

2.2.7 Load transmitter buffer

Addressing the load-transmitter-buffer function transfers the state of the data bus into the transmitter buffer.

2.2.8 Load output port

Addressing the load-output-port function transfers the state of the data bus into the output port. The data is latched and remains on $\overline{XO\ 0}$ through $\overline{XO\ 7}$ as the complement of the data bus until new data is loaded.

2.2.9 Load mask register

Addressing the load-mask-register function loads the contents of the data bus into the mask register. A high in data bus bit n enables interrupt n. A low inhibits the corresponding interrupt.

2.2.10 Load timer n

Addressing the load-timer-n function loads the contents of the data bus into the appropriate interval timer. Time intervals of from 64 μs (data bus = LLLLLLLH) to 16,320 μs (data bus HHHHHHHH) are counted in 64-μs steps. When the count of interval timer n reaches 0, the bit in the interrupt register that corresponds to timer n is set and an interrupt is generated. Loading all lows causes an interrupt immediately.

3. TMS 5501 ELECTRICAL AND MECHANICAL SPECIFICATIONS

3.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1)	−0.3 V to 20 V
Supply voltage, V_{DD} (see Note 1	−0.3 V to 20 V
Supply voltage, V_{SS} (see Note 1)	−0.3 V to 20 V
All input and output voltages (see Note 1)	−0.3 V to 20 V
Continuous power dissipation	1.1 W
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Under absolute maximum ratings voltage values are with respect to the normally most negative supply voltage, V_{BB} (substrate). Throughout the remainder of this data sheet, voltage values are with respect to V_{SS} unless otherwise noted.

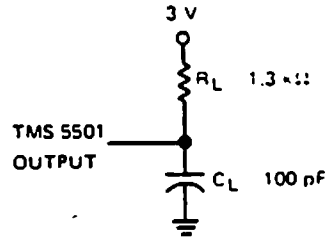
3.2 RECOMMENDED OPERATING CONDITIONS

	MIN	NOM	MAX	UNIT
Supply voltage, V_{BB}	−4.75	−5	−5.25	V
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{DD}	11.4	12	12.6	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH} (all inputs except clocks)	3.3		V_{CC}+1	V
High-level clock input voltage, $V_{IH(\phi)}$	V_{DD}−1		V_{DD}+1	V
Low-level input voltage, V_{IL} (all inputs except clocks) (see Note 2)	−1		0.8	V
Low-level clock input voltage, $V_{IL(\phi)}$ (see Note 2)	−1		0.6	V
Operating free-air temperature, T_A	0		70	C

NOTE 2: The algebraic convention where the most negative limit is designated as minimum is used in this specification for logic voltage levels only.

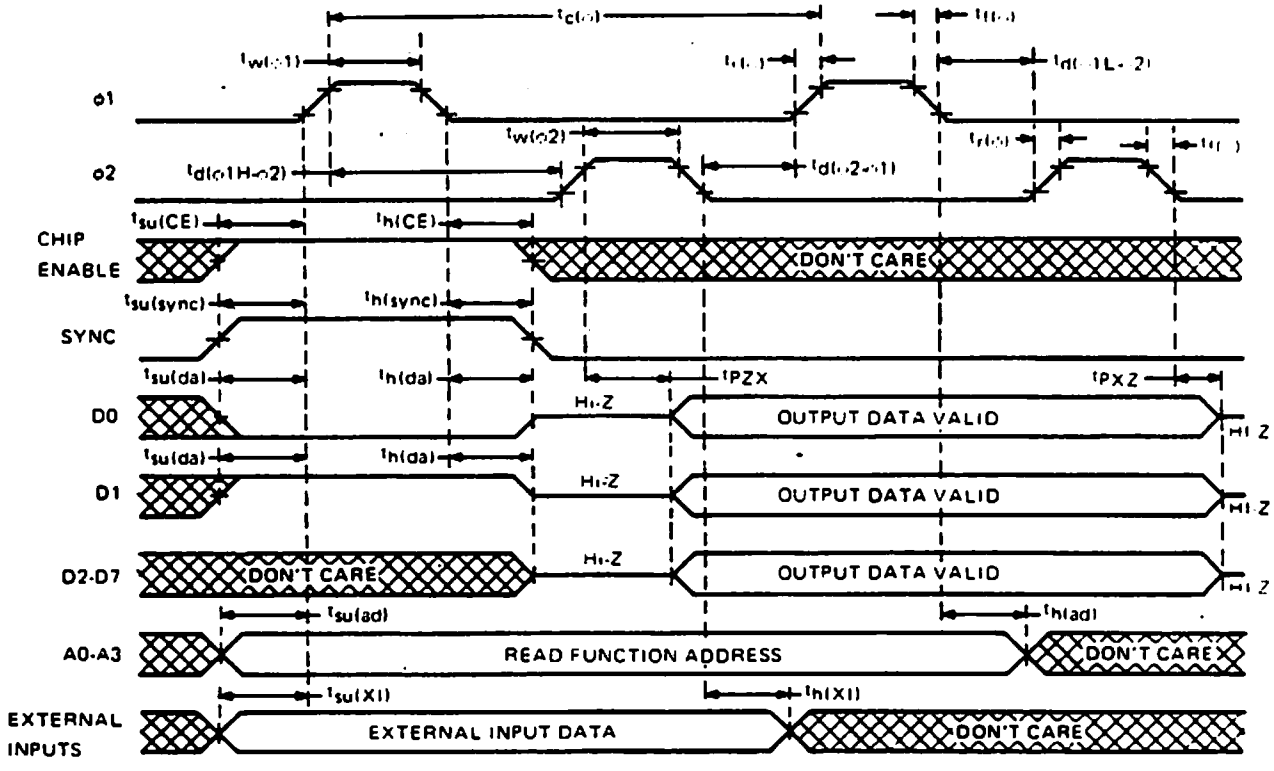
3.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (SEE FIGURES 6 AND 7)

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{PZX}	Data bus output enable time	C_L 100 pF, R_L 1.3 kΩ		200	ns
t_{PXZ}	Data bus output disable time to high-impedance state			180	ns
t_{PD}	External data output propagation delay time from Φ2			200	ns



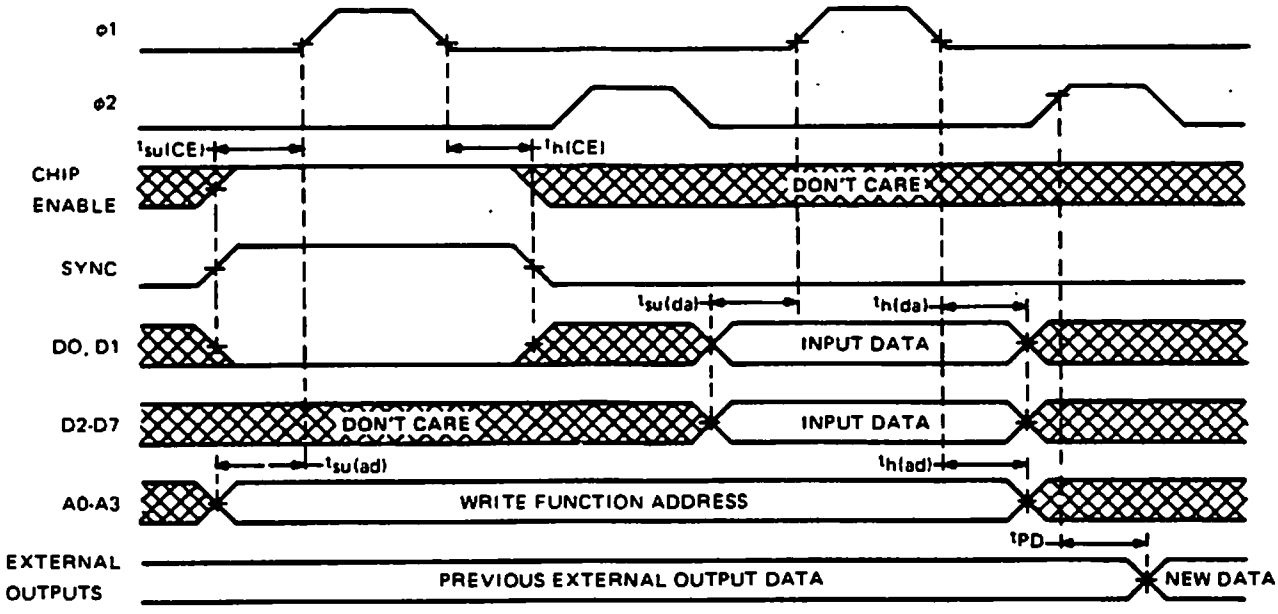
C_L includes probe and jig capacitance

LOAD CIRCUIT



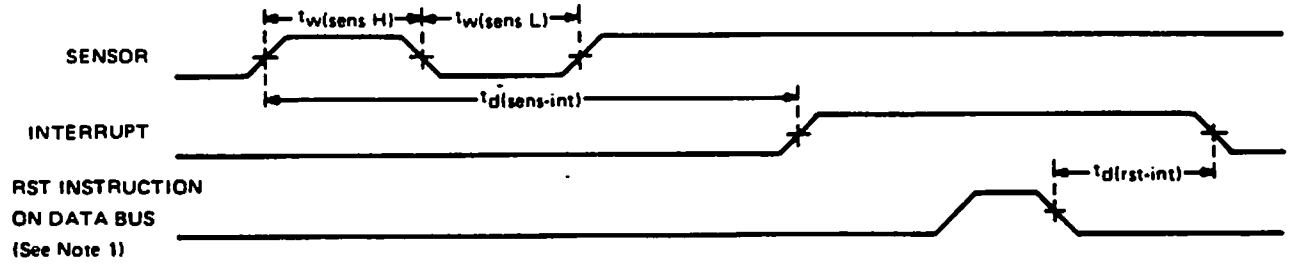
NOTE For Φ1 or Φ2 inputs, high and low timing points are 90% and 10% of $V_{IH(\min)}$. All other timing points are the 50% level.

FIGURE 6–READ CYCLE TIMING



NOTE For ø1 and ø2 inputs, high and low timing points are 90% and 10% of $V_{IH(\phi)}$. All other timing points are the 50% level.

FIGURE 7–WRITE CYCLE TIMING



NOTES 1 The RST instruction occurs during the output data valid time of the read cycle.
2 All timing points are 50% of V_{IH}.

FIGURE 8–SENSOR/INTERRUPT TIMING

Appendix G.3

CRT Controller

The 5048 CRT Controller is similar to the 5027. It differs in that pin 10 is used for 50/60 Hz. Sync input rather than for composite sync output. The 5048-004 is masked for the 80 x 48 display format as used by ISC and does not require loading of some of the registers at power-up. The 5048-003 is used for the 64 x 32 format.

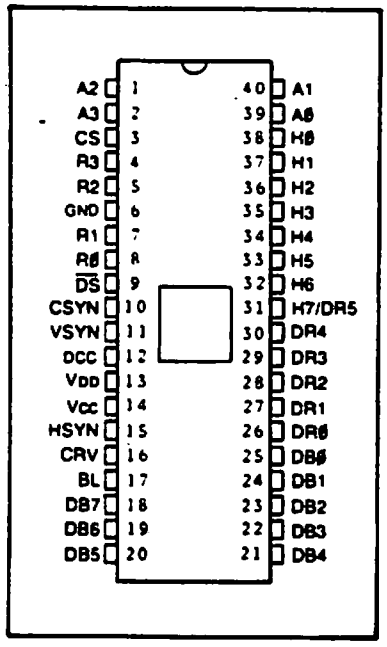
CRT Video Timer-Controller VTAC

5027

FEATURES

- ☐ Fully Programmable Display Format
  - Characters per data row
  - Data rows per frame
  - Raster scans per data row
  - Raster scans per frame
- ☐ Fully Programmable Monitor Format
  - Horizontal Sync
  - Vertical Sync
  - Composite Sync
- ☐ Programmed via:
  - Processor data bus
  - External PROM
  - Mask option ROM
- ☐ Standard or Non-Standard CRT Monitor Compatible
- ☐ Scrolling
- ☐ Generation of Cursor Video
- ☐ Interlaced and Non-interlaced Operation
- ☐ Vertical Data Positioning
- ☐ TTL Compatibility
- ☐ High Speed Operation
- ☐ COPLAMOS® N-Channel Silicon Gate Technology

PIN CONFIGURATION



General Description

The CRT Video Timer-Controller Chip (VTAC) is a user programmable 40-pin COPLAMOS® n channel MOS/LSI device containing the logic functions required to generate all the timing signals for the presentation and formatting of interlaced and non-interlaced video data on a standard or non-standard CRT monitor.

With the exception of the dot counter, which may be clocked at a video frequency above 25 MHz and therefore not recommended for MOS implementation, all frame formatting, such as horizontal, vertical, and composite sync, characters per data row, data rows per frame, and raster scans per data row and per frame are totally user programmable. The data row counter has been designed to facilitate scrolling.

Programming is effected by loading seven 8 bit control registers directly off an 8 bit bidirectional data bus. Four register address lines and a chip select line provide complete microprocessor compatibility for program controlled set up. The device can be "self loaded" via an external PROM tied on the data bus as described in the OPERATION section. Formatting can also be programmed by a single mask option.

In addition to the seven control registers two additional registers are provided to store the cursor character and data row addresses for generation of the cursor video signal. The contents of these two registers can also be read out onto the bus for update by the program.

MAXIMUM GUARANTEED RATINGS*

Operating Temperature Range	0°C to + 70°C
Storage Temperature Range	−55°C to +150°C
Lead Temperature (soldering, 10 sec.)	+325°C
Positive Voltage on any Pin, with respect to ground	+18.0V
Negative Voltage on any Pin, with respect to ground	−0.3V

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

ELECTRICAL CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = +12V ±5%, unless otherwise noted)

Parameter	Min.	Typ.	Max.	Unit	Comments
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low Level, V_{IL}			0.8	V	
High Level, V_{IH}	Vcc - 1.5		Vcc	V	
OUTPUT VOLTAGE LEVELS					
Low Level—V_{OL} for RØ-3			0.4	V	I_{OL} = 3.2ma
Low Level—V_{OL} all others			0.4	V	I_{OL} = 1.6ma
High Level—V_{OH} for RØ-3	2.4				I_{OH} = 80μa
High Level—V_{OH} all others	2.4				I_{OH} = 40μa
INPUT CURRENT					
Low Level, I_{IL}					
High Level, I_{IH}					
INPUT CAPACITANCE					
Data Bus, C_{IN}				pf	
Clock, C_{IN}				pf	
All other, C_{IN}				pf	
DATA BUS LEAKAGE in INPUT MODE					
I_{DB}					
I_{DB}					
POWER SUPPLY CURRENT					
I_{CC}				ma	
I_{DD}				ma	
A.C. CHARACTERISTICS					T_A = 25°C
DOT COUNTER CARRY					
frequency	0.2	4.0		MHz	Figure 1
PW_H	35			ns	Figure 1
PW_L	190			ns	Figure 1
t_r, t_f		10		ns	Figure 1
DATA STROBE					
$PW_{\overline{DS}}$		150		ns	Figure 2
ADDRESS, CHIP SELECT					
Set-up time		100		ns	Figure 2
Hold time		50		ns	Figure 2
DATA BUS—LOADING					
Set-up time		100		ns	Figure 2
Hold time		75		ns	Figure 2
DATA BUS—READING					
T_{DEL2}		100		ns	Figure 2, CL = 50pf
OUTPUTS: HØ-7, HS, VS, BL, CRV, CS-T_{DEL1}		100		ns	Figure 1, CL = 20pf
OUTPUTS: RØ-3, DRØ-5					
T_{DEL3}		1.0		μs	Figure 3, CL = 20pf

Restrictions

1. Only one pin is available for strobing data into the device via the data bus. The cursor X and Y coordinates are therefore loaded into the chip by presenting one set of addresses and outputed by presenting a different set of addresses. Therefore the standard WRITE and READ control signals from most microprocessors must be "NORed" externally to present a single strobe ($\overline{DS}$) signal to the device.

2. An even number of scan lines per character row must be programmed in interlace mode. This is again due to pin count limitations which require that the least significant bit of the scan counter serve as the odd/even field indicator.

3. In interlaced mode the total number of character slots assigned to the horizontal scan must be even to insure that vertical sync occurs precisely between horizontal sync pulses.

Operation

The design philosophy employed was to allow the device to interface effectively with either a microprocessor based or hardwire logic system. The device is programmed by the user in one of two ways; via the processor data bus as part of the system initialization routine, or during power up via a PROM tied on the data bus and addressed directly by the Row Select outputs of the chip. (See figure 4). Seven 8 bit words are required to fully program the chip. Bit assignments for these words are shown in Table 1. The information contained in these seven words consists of the following:

Horizontal Formatting:

Characters/Data Row	A 3 bit code providing 8 mask programmable character lengths from 20 to 132. The standard device will be masked for the following character lengths. 20, 32, 40, 64, 72, 80, 96, and 132
Horizontal Sync Delay	3 bits assigned providing up to 8 character times for generation of "front porch".
Horizontal Sync Width	4 bits assigned providing up to 16 character times for generation of horizontal sync width.
Horizontal Line Count	8 bits assigned providing up to 256 character times for total horizontal formatting
Skew Bits	A 2 bit code providing from a 0 to 2 character skew between the horizontal address counter and the horizontal blank and sync signals to allow for retiming of video data prior to generation of composite video signal. The Cursor Video signal is also skewed as a function of this code

Vertical Formatting

Interlaced/Non-interlaced	This bit provides for data presentation with odd even field formatting for interlaced systems. It modifies the vertical timing counters as described below
Scans/Frame	8 bits assigned, defined according to the following equations: Let X = value of 8 assigned bits. 1) in interlaced mode—scans/frame = 2X + 513. Therefore for 525 scans, program X = 6 (00000110). Vertical sync will occur precisely every 262.5 scans, thereby producing two interlaced fields. Range = 513 to 1023 scans/frame, odd counts only. 2) in non-interlaced mode—scans/frame = 2X + 256. Therefore for 262 scans, program X = 3 (00000011). Range = 256 to 766 scans/frame, even counts only. In either mode, vertical sync width is fixed at three horizontal scans (= 3H).
Vertical Data Start	8 bits assigned providing scan line resolution in vertical data positioning with respect to vertical sync. The Data Row Counter is reset at vertical sync and will not begin counting until the scan line selected by these eight bits.
Data Rows/Frame	6 bits assigned providing up to 64 data rows per frame.
Last Data Row	6 bits to allow up or down scrolling via a preload defining the count of the last displayed data row.
Scans/Data Row	4 bits assigned providing up to 16 scan lines per data row.

Additional Features

Device Initialization

Under microprocessor control—The device can be reset under system or program control by presenting a Ø1Ø1 address on AØ-3. The device will remain reset at the top of the even field page until a start command is executed by presenting a Ø111 address on AØ-3.

Via "Self Loading"—In a non-processor environment, the self loading sequence is effected by presenting and holding the 1111 address on AØ-3, and is initiated by the receipt of the strobe pulse ($\overline{DS}$). The 1111 address should be maintained long enough to insure that all seven registers have been loaded (in most applications under one millisecond). The timing sequence will begin one line scan after the 1111 address is removed. In processor based systems, self loading is initiated by presenting the 111Ø address to the device. Self loading is terminated by presenting the start command to the device which also initiates the timing chain.

Scrolling—In addition to the Register 6 storage of the last displayed data row a "scroll" command (address 11Ø1) presented to the device will increment the first displayed data row count to facilitate up scrolling in certain applications.

Description of Pin Functions

Pin No.	Symbol	Name	Input/ Output	Function
25-18	DBØ-7	Data Bus	I/O	Data bus. Input bus for control words from microprocessor or PROM. Bidirectional bus for cursor address.
3	CS	Chip Select	I	Signals chip that it is being addressed
39, 40, 1, 2	AØ-3	Register Address	I	Register address bits for selecting one of seven control registers or either of the cursor address registers
9	$\overline{DS}$	Data Strobe	I	Strobes DBØ-7 into the appropriate register or outputs the cursor character address or cursor line address onto the data bus
12	DCC	DOT Counter Carry	I	Carry from off chip dot counter establishing basic character clock rate.
38-32	HØ-6	Character Counter Outputs	O	Character counter outputs.
7, 5, 4	R1-3	Scan Counter Outputs	O	Three most significant bits of the Scan Counter; row select inputs to character generator.
31	H7/DR5	H7 DR5	O	Pin definition is user programmable. Output is MSB of Character Counter if MSB of Characters/Data Row word is a "1", otherwise output is MSB of Data Row Counter
8	RØ	Scan Counter LSB (Odd/Even Field)	O	Least significant bit of the scan counter. In interlaced mode this bit defines the odd or even field. In this way, odd scan lines of the character font are selected during the odd field and even scans during the even field.
26-30	DRØ-4	Data Row Counter Outputs	O	Data Row counter outputs.
17	BL	Blank	O	Defines non active portion of horizontal and vertical scans.
15	HSYN	Horizontal Sync	O	Initiates horizontal retrace.
11	VSYN	Vertical Sync	O	Initiates vertical retrace.
10	CSYN	Composite Sync	O	Active in non-interlaced mode only. Provides a true RS-170 composite sync waveform.
16	CRV	Cursor Video	O	Defines cursor location in data field.
14	V_{CC}	Power Supply	PS	+5 volt Power Supply
13	V_{DD}	Power Supply	PS	+12 volt Power Supply

Timing Diagrams

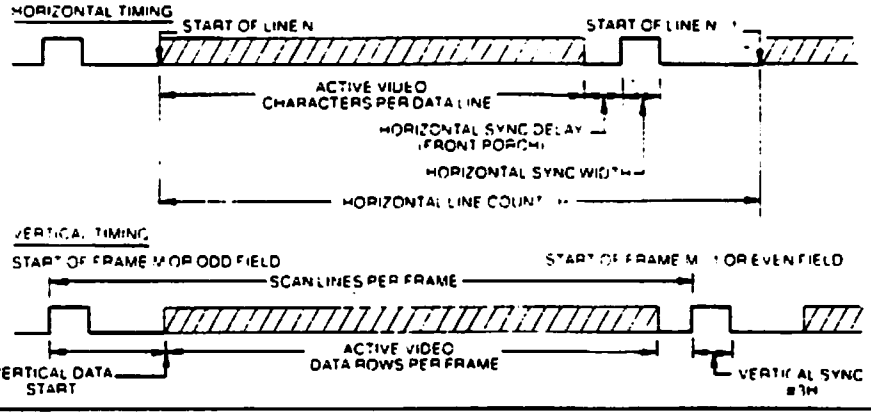
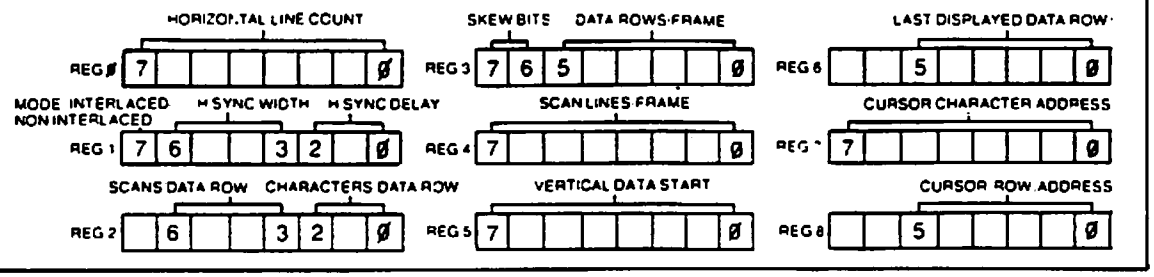


TABLE 1

BIT ASSIGNMENT CHART



Register Selects/Command Codes

A3	A2	A1	AØ	Select/Command	Description
0	0	0	0	Load Control Register Ø	See Table 1
0	0	0	1	Load Control Register 1	
0	0	1	0	Load Control Register 2	
0	0	1	1	Load Control Register 3	
0	1	0	0	Load Control Register 4	
0	1	0	1	Load Control Register 5	
0	1	1	0	Load Control Register 6	
0	1	1	1	Processor Self Load	Command from processor instructing CRT 5027 to enter Self Load Mode
1	0	0	0	Read Cursor Character Address	
1	0	0	1	Read Cursor Line Address	
1	0	1	0	Reset	Resets timing chain to top left of page. Reset is latched on chip by $\overline{DS}$ and counters are held until released by start command.
1	0	1	1	Up Scroll	Increments address of first displayed data row on page. ie; prior to receipt of scroll command—top line = 0, bottom line = 23. After receipt of Scroll Command—top line = 1, bottom line = 0.
1	1	0	0	Load Cursor Character Address	
1	1	0	1	Load Cursor Line Address	
1	1	1	0	Start Timing Chain	Receipt of this command after a Reset or Processor Self Load command will release the timing chain approximately one scan line later. In applications requiring synchronous operation of more than one CRT 5027 the dot counter carry should be held low during the $\overline{DS}$ for this command.
1	1	1	1	Non-Processor Self Load	Device will begin self load via PROM when $\overline{DS}$ goes low. The 1111 command should be maintained on AØ-3 long enough to guarantee self load. (Scan counter should cycle through at least once). Self load is automatically terminated and timing chain initiated when the all "1's" condition is removed, independent of $\overline{DS}$. For synchronous operation of more than one CRT 5027, the Dot Counter Carry should be held low when this command is removed.

NOTE: During Self Load, the scan counter states corresponding to the nine load command addresses will load the appropriate register. Therefore if reseting of the cursor X and Y position registers is required via self load the PROM words for address 11ØØ and 11Ø1 should be programmed as all zeros.

AC TIMING DIAGRAMS

FIGURE 1 VIDEO TIMING

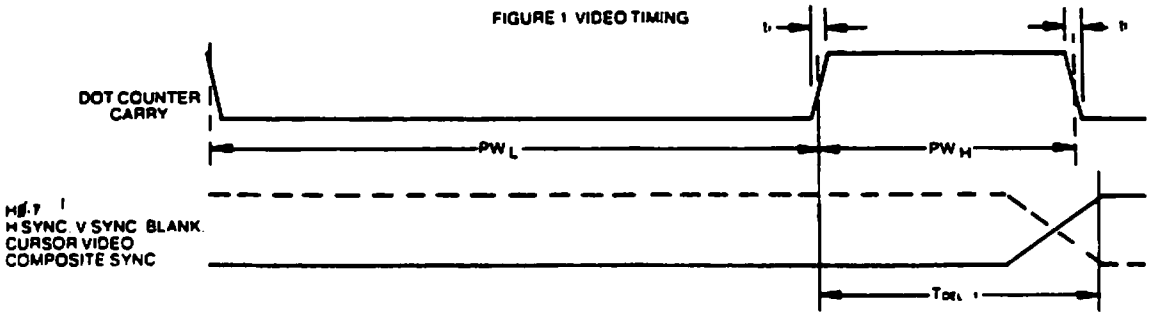


FIGURE 2 LOAD/READ TIMING

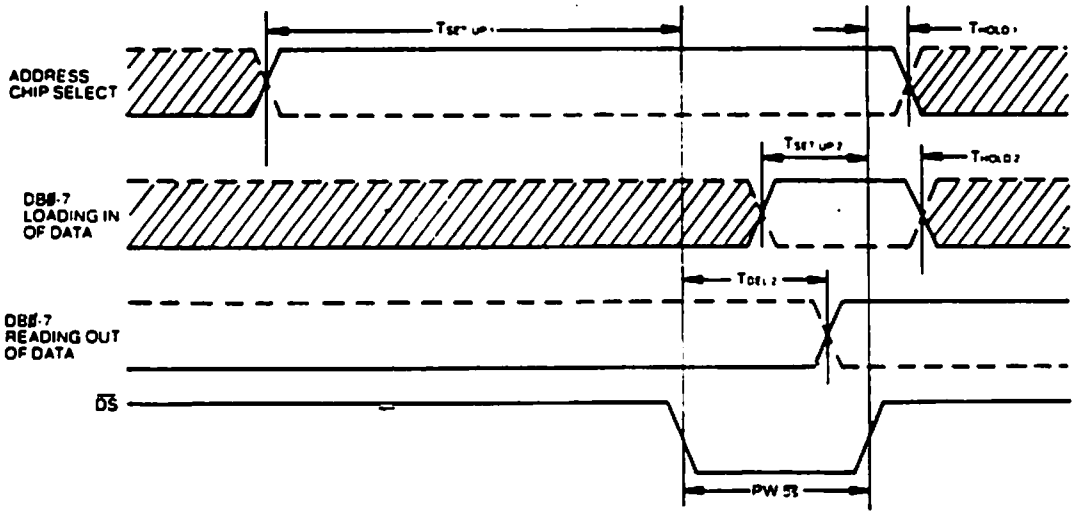


FIGURE 3 SCAN AND DATA ROW COUNTER TIMING

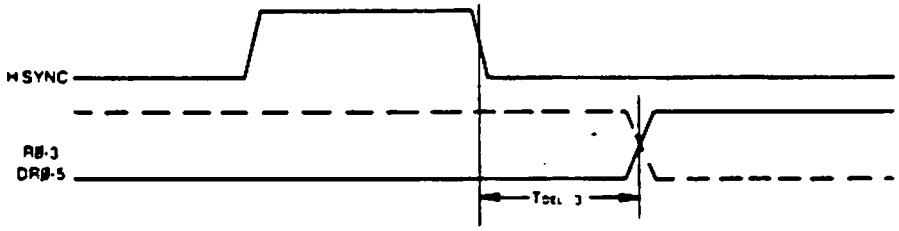
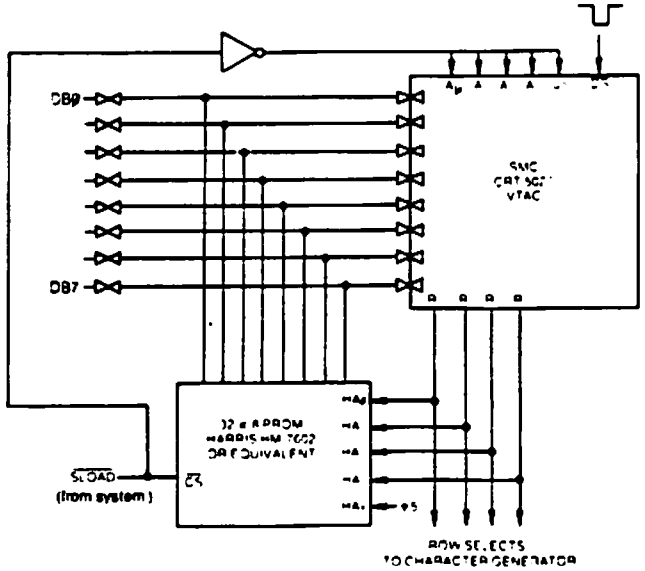


Figure 4. SELF LOADING SCHEME FOR CRT 5027 SET-UP



FD1771 A/B - 01

DATA SHEET — FLOPPY DISK FORMATTER/CONTROLLER

GENERAL DESCRIPTION

The FD1771 is a MOS/LSI device that performs the functions of a Floppy Disk Controller/Formatter. The device is designed to be included in the disk drive electronics, and contains a flexible interface organization that accomodates the interface signals from most drive manufactures. The FD1771 is compatible with the IBM 3740 data entry system format.

The processor interface consists of a 8-bit bi-directional bus for data, status, and control word transfers. The FD1771 is set up to operate on a multiplexed bus with other bus-oriented devices.

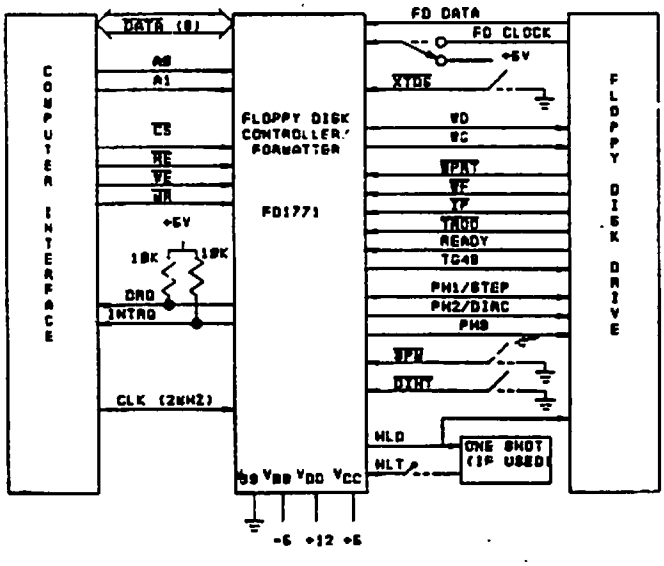
The FD1771 is fabricated in N-channel Silicon Gate MOS technology and is TTL compatible on all inputs and outputs.

APPLICATIONS

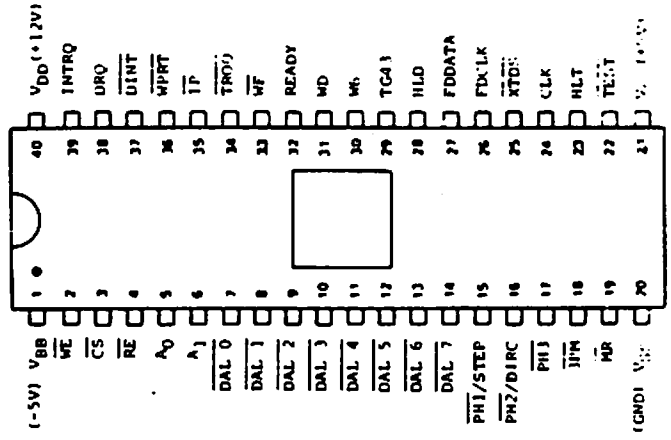
- o FLOPPY DISK DRIVE INTERFACE
- o SINGLE OR MULTIPLE DRIVE CONTROLLER/FORMATTER
- o NEW MINI-FLOPPY CONTROLLER

FEATURES

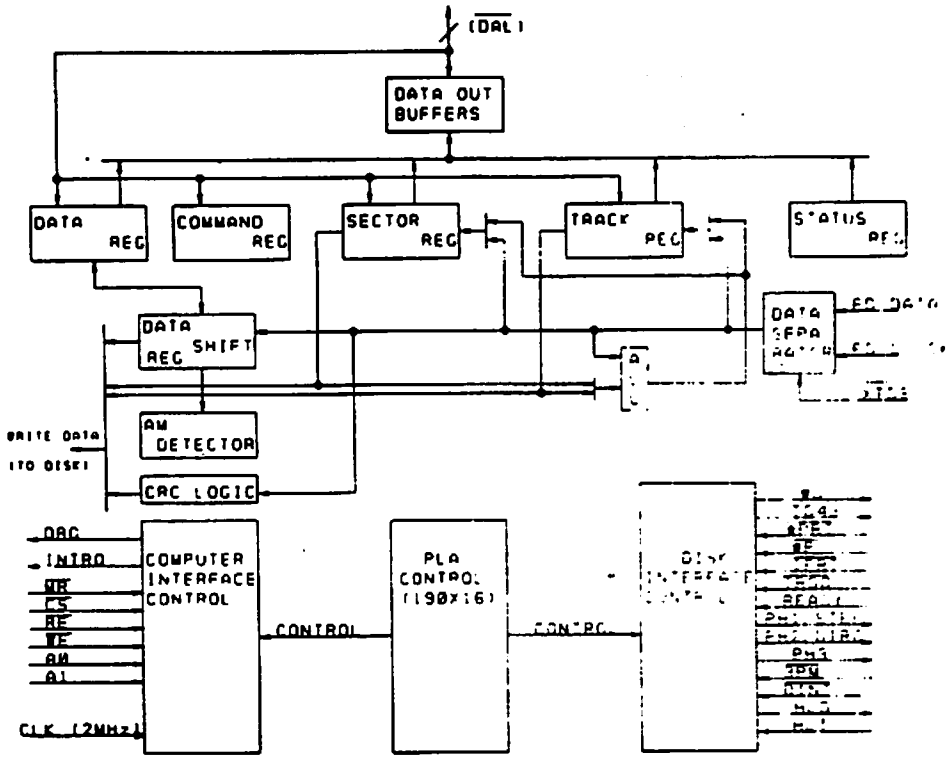
- o SOFT SECTOR FORMAT COMPATIBILITY
- o AUTOMATIC TRACK SEEK WITH VERIFICATION
- o READ MODE
  - Single/Multiple Record Read with Automatic Sector Search or Entire Track Read
  - Selectable 128 Byte or Variable Length Record
- o WRITE MODE
  - Single/Multiple Record Write with Automatic Sector Search
  - Entire Track Write for Diskette Initialization
- o PROGRAMMABLE CONTROLS
  - Selectable Track to Track Stepping Time
  - Selectable Head Settling and Head Engage Times
  - Selectable Three Phase or Step and Direction and Head Positioning Motor Controls
- o SYSTEM COMPATIBILITY
  - Double Buffering of Data 8 Bit Bi-Directional Bus for Data, Control and status
  - DMA or Programmed Data Transfers
  - All Inputs and Outputs are TTL Compatible



FD1771 SYSTEM BLOCK DIAGRAM
FIG 1



FD1771 PIN CONNECTIONS
FIG 2



FD1771 BLOCK DIAGRAM
FIG 3

ORGANIZATION

The Floppy Disk Formatter block diagram is illustrated on Page 2. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register - This 8-bit register assembles serial data from the Read Data input (FDDATA) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register - This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register can be loaded from the DAL and gated onto the DAL under processor control.

Track Register - This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in (towards track 76) and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when this device is busy.

Sector Register (SR) - This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy

Command Register (CR) - This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the execution of the current command is to be overridden. This latter action results in an interrupt The command register can be loaded from the DAL, but not read onto the DAL.

Status Register (STR) - This 8-bit register holds device Status information. The meaning of the Status bits are a function of the contents of the Command Register. This register can be read onto the DAL but not loaded from the DAL.

CRC Logic This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is: $G(x) = x^{16} + x^{12} + x^5 + 1$

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) - The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

AM Detector - The Address Mark detector is used to detect ID, Data, and Index address marks during Read and Write operations.

Timing and Control - All computer and Floppy Disk Interface controls are generated through this logic. The internal device timing is generated from a 2.0 MHz external crystal clock.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the FD1771. The DAL are three state buffers that are enabled as output drivers when Chip Select (CS) and Read Enable (RE) are active (low logic state) or act as input receivers when CS and Write Enable (WE) are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and $\overline{CS}$ is made low. The least-significant address bits A1 and A0, combined with the signals $\overline{RE}$ during a Read operation or $\overline{WE}$ during a Write operation are interpreted as selecting the following registers:

A1-A0		READ ($\overline{RE}$)	WRITE ($\overline{WE}$)
0	0	Status Register	Command Register
0	1	Track Register	Track Register
1	0	Sector Register	Sector Register
1	1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the FD1771 and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operation continues until the end of sector is reached.

On Disk Write operations the Data Request is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

The Lost Data bit and certain other bits in the Status Register will activate the interrupt request (INTRQ). The interrupt line is also activated with normal completion or abnormal termination of all controller operations. The INTRQ signal remains active until reset by reading the Status Register to the processor or by the loading of the Command Register. In addition, the INTRQ is generated if a Force Interrupt command condition is met.

FLOPPY DISK INTERFACE

The Floppy Disk interface consists of head positioning controls, write gate controls, and data transfers. A 2.0 MHz ± 1% square wave clock is required at the CLK input for internal control timing, (may be 1.0 MHz for mini floppy.)

HEAD POSITIONING

Four commands cause positioning of the Read-Write head (see Command Section). The period of each positioning step is specified by the r field in bits 1 and 0 of the command word. After the last directional step an additional 10 milliseconds of head settling time takes place. The four programmable stepping rates are tabulated below.

The rates (shown in Table 1) can be applied to a Three Phase Motor or a Step-Direction Motor through the device interface. When the 3PM input is connected to ground the device operates with a three-phase motor control interface, with one active low signal per phase on the three output signals PH1, PH2 and PH3. The stepping sequence, when stepping in, is Phases 1-2-3-1, and when stepping out, Phases 1-3-2-1. Phase 1 is active low after Master Reset. Note: PH3 needs an inverter if used.

The Step-Direction Motor Control interface is activated by leaving input 3PM open or connecting it to +5V. The Phase 1 pin PH1 becomes a Step pulse of 4 microseconds width. The Phase 2 pin PH2 becomes a direction control with a high voltage on this pin indicating a Step In, and a low voltage indicating a Step Out. The Direction output is valid a minimum of 24 us prior to the activation of the Step pulse.

When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by setting bit 2 in the command word to a logic 1. The verification operation begins at the end of the 10 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track

Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete. If track comparison is not made but the CRC checks, an interrupt is generated, the Seek Error status (Bit 4) is set and the Busy status bit is preset.

TABLE 1
STEPPING RATES

r1	r0	1771-01 CLK=2MHZ TEST=1	1771-01 CLK=1MHZ TEST=1	1771 or-01 CLK=2MHZ TEST=0	1771 or-01 CLK=1MHZ TEST=0
0	0	6ms	12ms	*APPROX. 400us	*APPROX. 800us
0	1	6ms	12ms		
1	0	10ms	20ms		
1	1	20ms	40ms		

*For exact times consult WDC.

The Head Load (HDL) output controls the movement of the read/write head against the disk for data recording or retrieval. It is activated at the beginning of a Read, Write (E Flag On) or Verify Operation, or a Seek or Step operation with the head load bit, h, a logic one remains activated until the third index pulse following the last operation which uses the read/write head. Reading or Writing does not occur until a minimum of 10 msec delay after the HDL signal is made active. If executing the type 2 commands with the E flag off, there is no 10 msec delay and the head is sssumed to be engaged. The delay is determined by sampling of the Head Load Timing (HLT) input after 10 msec. A low logic state input, generated from the Head Load output transition and delayed externally, identifies engagement of the head against the disk. In the Seek and Step commands, the head is loaded at the start of the command execution when the h bit is a logic one. In a verify command the head is loaded before stepping to the destination track on the disk whenever the h bit is a logic zero.

DISK READ OPERATION

The 2.0 MHz external clock provided to the device is internally divided by 4 to form the 500 KHz clock rate for data transfer. When reading data from a diskette this divider is synchronized to transitions of the Read Data (FDDATA) input. When a transition does not occur on the 500 KHz clock active state, the clock divider circuit injects a clock to maintain a continuous 500 KHz data clock. The 500 KHz data clock is further divided by 2 internally to separate the clock and information bits. The divider is phased to the information by the detection of the address mark.

In the internal data read and separation mode the Read Data input toggles from one state to the opposite state for each logic one bit of clock or information. This signal can be derived from the amplified, differentiated, and sliced Read Head signal, or by the output of a flip-flop toggling on the Read Data pulses. This input is sampled by the 2 MHz clock to detect transitions.

The chip can also operate on externally separated data, as supplied by methods such as Phase Lock loop. One Shots, or variable frequency oscillators. This is accomplished by grounding the External Data Separator (XTDS) INPUT. When the Read Data input makes a high to-low transition, the information input to the FDDATA line is clocked into the Data Shift Register. The assembled 8 bit data from the Data Shift Register are then transferred to the Data Register.

The normal sector length for Read or Write operations with the IBM 3740 format is 128 bytes. This format or binary multiples of 128 bytes will be adopted by setting a logic 1 in Bit 3 of the Read and Write commands. Additionally, a variable sector length feature is provided which allows an indicator recorded in the ID Field to control the length of the sector. Variable sector lengths can be read or written in Read or Write commands respectively by setting a logic o in Bit 3 of the command word. The sector length indicator specifies the number of 16 byte groups or 16 x N, where N is equal to 1 to 256 groups. An indicator of all zeroes is interpreted as 256 sixteen byte groups.

DISK WRITE OPERATION

After data is loaded from the processor into the Data Register, and is transferred to the Data Shift Register, data will be shifted serially through the Write Data (WD) output. Interlaced with each bit of data is a positive clock pulse of 0.5 usec duration. This signal may be used to externally toggle a flip-flop to control the direction of Write Current flow.

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the FD1771 before the Write Gate signal can be activated.

Writing is inhibited when the Write Protect input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set. The Write Fault input, when activated, signifies a writing fault condition detected in disk drive electronics such as failure to detect write current flow when the Write Gate is activated. On detection of this fault the FD1771 terminated the current command, and sets the Write Fault bit (bit 5) in the Status Word. The Write Fault input should be made inactive when the Write Gate output becomes inactive.

Whenever a Read or Write command is received the FD1771 samples the READY input. If this input is logic ow the command is not executed and an interrupt is generated. The Seek or Step commands are performed regardless of the state of the READY input.

COMMAND DESCRIPTION

The FD1771 will accept and execute eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in table 2.

COMMAND SUMMARY*

		BITS							
TYPE	COMMAND	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step In	0	1	0	u	h	V	r_1	r_0
I	Step Out	0	1	1	u	h	V	r_1	r_0
II	Read Command	1	0	0	m	b	E	0	0
II	Write Command	1	0	1	m	b	E	a_1	a_0
III	Read Address	1	1	0	0	0	1	0	0
III	Read Track	1	1	1	0	0	1	0	s
III	Write Track	1	1	1	1	0	1	0	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

TABLE 2

• = Shown in true form.

FLAG SUMMARY

TYPE 1

h = Head Load Flag (Bit 3)
- h=1, Load head at beginning
- h=0, Do not load head at beginning

V = Verify flag (Bit 2)
- V=1, Verify on last track
- V=0, No verify

r_1r_0 = Stepping motor rate (Bits 1-0)
- Refer to Table 1 for rate summary

u = Update flag (Bit 4)
- u=1, Update Track register
- u=0, No update

TABLE 3

TYPE II

m = Multiple Record flag (Bit 4)
- m = 0, Single Record
- m = 1, Multiple Records

b = Block length flag (Bit 3)
- b = 1, IBM format (128 to 1024 bytes)
- b = 0, Non-IBM format (16 to 4096 bytes)

a_1a_0 = Data Address Mark (Bits 1-0)
- a_1a_0 = 00, FB (Data Mark)
- a_1a_0 = 01, FA (User defined)
- a_1a_0 = 10, F9 (User defined)
- a_1a_0 = 11, F8 (Deleted Data Mark)

TABLE 4

TYPE III

s = Synchronize flag (Bit 0)
- s=0, Synchronize to AM
- s=1, Do Not Synchronize to AM

TYPE IV

I_i = Interrupt Condition flags (Bits 3-0)
- I_0=1, Not Ready to Ready Transition
- I_1=1, Ready to Not Ready Transition
- I_2=1, Index Pulse
- I_3=1, Immedate interrupt

E = Enable HLD and 10 msec Delay
- E=1, Enable HLD, HLT and 10 msec Delay
- E=0, Head is assumed Engaged and there is no 10 msec Delay.

TABLE 5

TYPE 1 COMMANDS

The Type 1 Commands include the RESTORE, SEEK STEP, STEP-IN, AND STEP-OUT commands. Each of the Type 1 Commands contain a rate field (r_0r_1), which determines the stepping motor rate as defined in Table 1, page four.

The type 1 Commands contain a head load flag (h) which determines if the head is to be loaded at the beginning of the command. If h = 1, the head is loaded at the beginning of the command HLD output is made active). If h = 0, HLD is deactivated. Once the head is loaded, the head will remain engaged until the FD1771 receives a command that specifically disengages the head. If the FD1771 does not receive any commands after two revolutions of the disk, the head will be automatically disengaged (HLD made inactive). The Head Load Timing Input is sampled after a 10 ms delay, when reading or writing on the disk is to occur.

The Type 1 Commands also contain a verification (V) flag which determines if a verification operation is to take place on the destination track. If V = 1, a verification is performed, if V = 0, no verification is performed.

During verification, the head is loaded and after an internal 10 ms delay, the HLT input is sampled. When HLT is active (logic true), the first encountered ID field is read off the disk. The track address of the ID field is then compared to the Track Register; if there is a match and a valid ID CRC, the verification is complete, an interrupt is generated and the BUSY status bit is reset. If there is not a match but there is valid ID CRC, an interrupt is generated, the Seek Error status bit (Status bit 4) is set and the BUSY status bit is reset. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3), and the next encountered ID field is read from the disk for the verification operation. If an ID field with a valid CRC cannot be found after two revolutions of the disk, the FD1771 terminates the operation and sends an interrupt, (INTRQ).

The STEP, STEP-IN, and STEP-OUT commands contain an UPDATE flag (U). When U = 1, the track register is updated by one for each step. When U = o, the track register is not updated.

RESTORE (SEEK TRACK 0)

Upon receipt of this command the Track 00 ($\overline{TROO}$) input is sampled. If $\overline{TROO}$ is active low indicating the Read-Write head is positioned over track 0, the .Track Register is loaded with zeroes and an interrupt is generated. If $\overline{TROO}$ is not active low, stepping pulses (pins 15 to 17) at a rate specified by the r1r0 field are issued until the $\overline{TROO}$ input is activated. At this time the TR is loaded with zeroes and an interrupt is generated. If the TROO input does not go active low after 255 stepping pulses, the FD1771 terminates operation, interrupts, and sets the Seek error status bit. Note that the RESTORE command is executed when MR goes from an active to an inactive state. A verification operation takes place if the V flag is set. The h bit allows the head to be loaded at the start of command.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read-Write head and the Data Register contains the desired track number. The FD1771 will update the Track register and issue stepping pulses in the appropriate direction until the contents of the Track register are equal to the contents of the data register (the desired track location). A verification operation takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP

Upon receipt of this command, the FD1771 issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the r1r0 field, a verification takes place if the V flag is on. If the u flag is on, the TR is updated. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

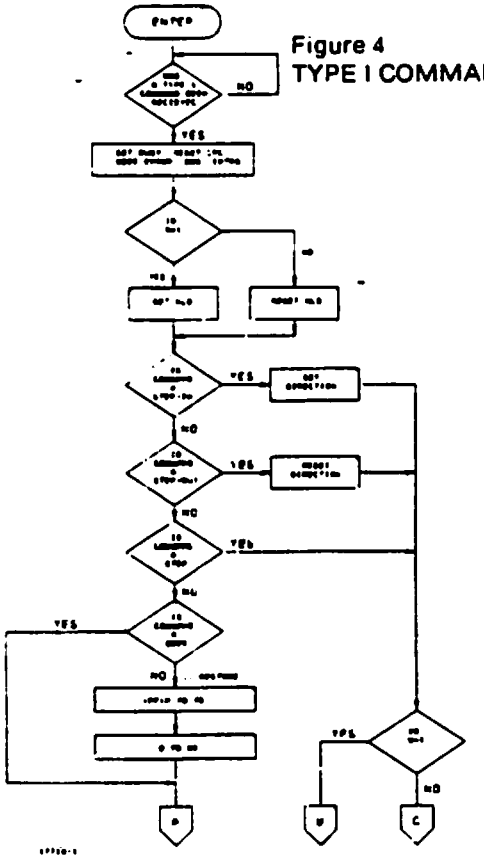


Figure 4
TYPE I COMMAND FLOW

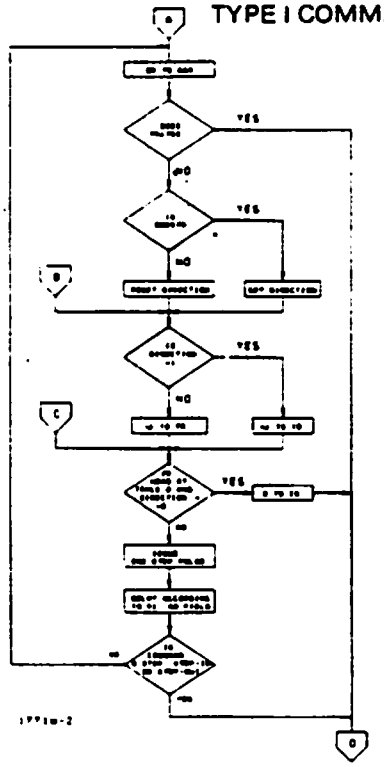


Figure 5
TYPE I COMMAND FLOW

STEP-IN

Upon receipt of this command, the FD1771 issues one stepping pulse in the direction towards track 76. If the u flag is on the Track Register is incremented by one. After a delay determined by the r_1 r_0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP-OUT

Upon receipt of this command, the FD1771 issues one stepping pulse in the direction towards track 0. If the u flag is on, the TR is decremented by one. After a delay determined by the r1r0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

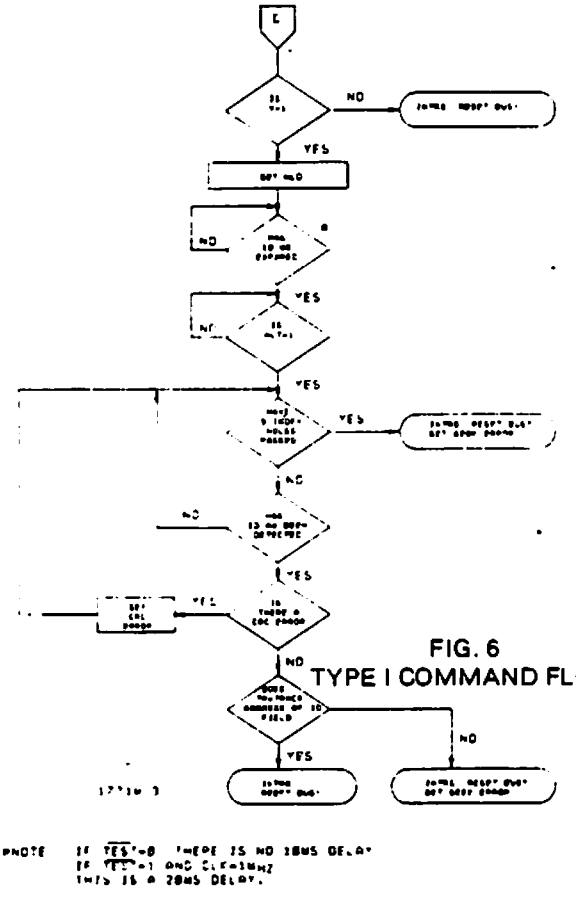


FIG. 6
TYPE I COMMAND FLOW

TYPE II COMMANDS

The Type II Commands include the Read Sector (s) and Write Sector (s) commands. Prior to loading the type II command into the COMMAND REGISTER, the computer must load the Sector Register with the desired sector number. Upon receipt of the type II command, the busy status Bit is set. If the E flag = 1 (this is the normal case) HLD is made active and HLT is sampled after a 10 msec delay. If the E flag is0, the head is assumed to be engaged and there is no 10 msec delay. The ID field and Data Field format are shown below:

When an ID field is located on the disk, the FD1771 compares the Track Number of the ID field with the Track register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The FD1771 must find an Id field with a Track number, Sector number, and CRC within two revolutions of the disk; otherwise, the Record not found status bit is set (Status bit 3) and the command is terminated with an interrupt.

GAP	ID AM	TRACK NUMBER	ZEROS	SECTOR NUMBER	SECTOR LENGTH	CRC 1	CRC 2	GAP	DATA AM	DATA FIELD	CRC 1	CRC 2
ID FIELD									DATA FIELD			

IDAM = ID Address Mark – DATA=$(FE)_{16}$ CLK = $(C7)_{16}$
Data AM = Data Address Mark – DATA=(F8, F9, FA, or FB), CLK = $(C7)_{16}$

Each of the Type II Commands contain a (b) flag which in conjunction with the sector length field contents of the ID determines the length (number of characters) of the Data field.

For IBM 3740 compatibility, the b flag should equal 1. The numbers of bytes in the data field (sector) is then 128 x 2^n where n - 0,1,2,3.

For b = 1

Sector Length Field (hex)	Number of bytes in sector (decimal)
00	128
01	256
02	512
03	1024

When the b flag equals zero, the sector length field (n) multiplied by 16 determines the number of bytes in the sector or data field as shown below:

For b = 0

Sector Length Field (hex)	Number of bytes in sector (decimal)
01	16
02	32
03	48
04	64
•	•
•	•
•	•
FF	4080
00	4096

Each of the type II commands also contain a (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m = o a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1 , multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The FD1771 will continue to read or write multiple records and update the sector register until the sector register exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the command register, which terminated the command and generates an interrupt.

FIG. 7
TYPE II COMMAND

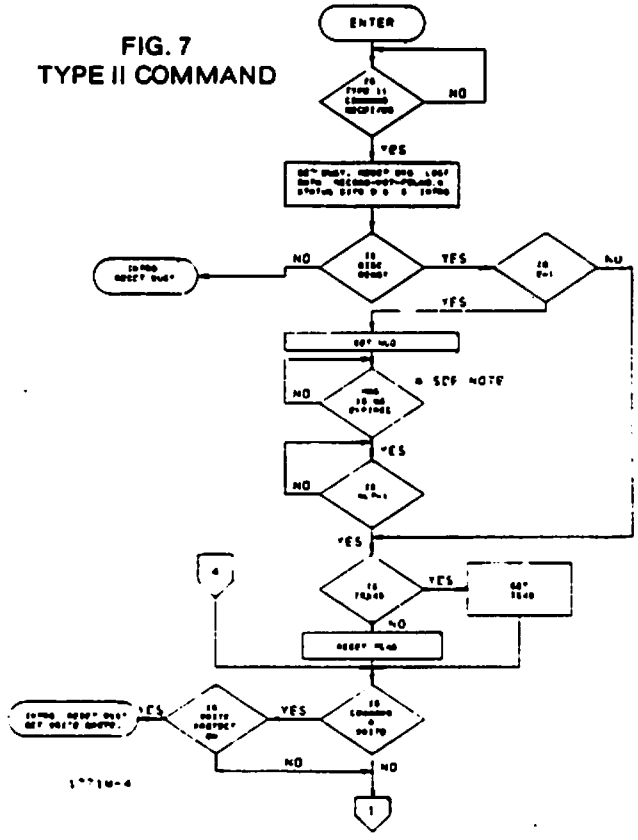
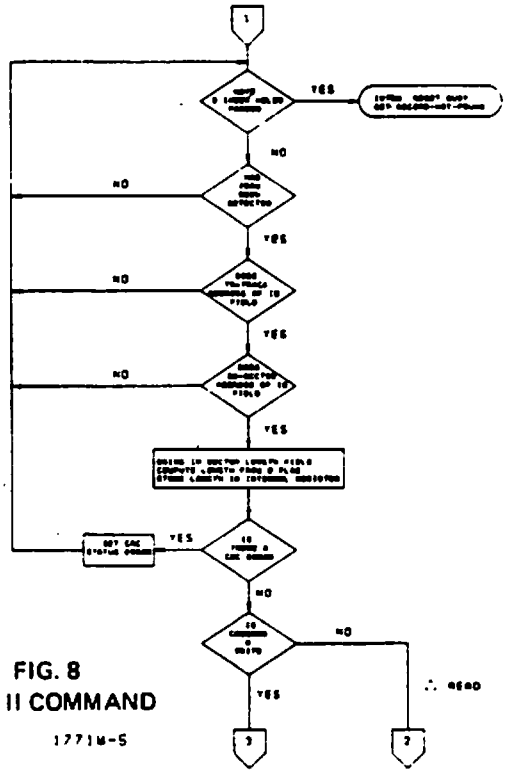


FIG. 8
TYPE II COMMAND



READ COMMAND

Upon receipt of the Read command, the head is loaded, the BUSY status bit set, and when an ID field is encountered that has the correct track number, correct sector number, and correct CRC, the data field is presented to the computer. The Data Address Mark of the data field must be found within 28 bytes of the correct field; if not, the Record Not Found status bit is set and the operation is terminated. When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the Computer has not read the previous contents of the DR before a new character is transferred that character is lost and the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bits 5 and 6) as shown below:

Status Bit 5	Status Bit 6	Data AM (HEX)
0	0	FB
0	1	FA
1	0	F9
1	1	F8

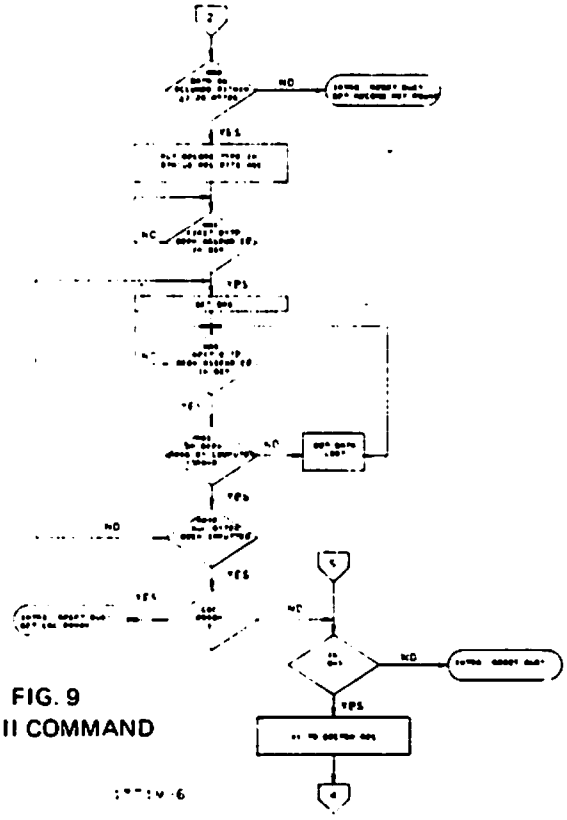


FIG. 9
TYPE II COMMAND

WRITE COMMAND

Upon receipt of the Write command, the head is loaded (HLD active) and the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, and correct CRC, a DRQ is generated. The FD1771 counts off 11 bytes from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeros are then written on the disk. At this time the Data Address Mark is then written on the disk as determined by the a^1a^0 field of the command as shown below:

a1	a0	DATA MARK (HEX)	CLOCK MARK (HEX)
0	0	FB	C7
0	1	FA	C7
1	0	F9	C7
1	1	F8	C7

The FD1771 then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit is set and a byte of zeros is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte gap of logic ones. The WG output is then deactivated.

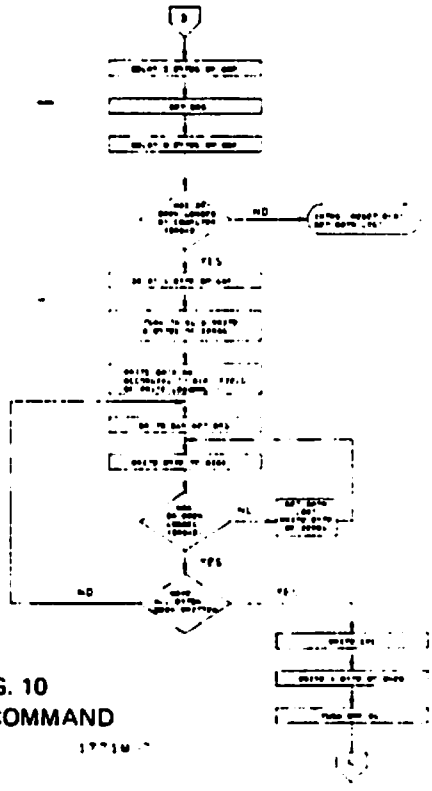


FIG. 10
TYPE II COMMAND

TYPE III COMMANDS

READ ADDRESS

Upon receipt of the Read Address command, the head is loaded and the BUSY Status Bit is set. The next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

TRACK ADDR	ZEROS	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

Although the CRC characters are transferred to the computer, the FD1771 checks for validity and the CRC error status bit is set if there is a CRC error. The Sector Address of the ID field is written into the sector register. At the end of the operation an interrupt is generated and the BUSY Status is reset.

READ TRACK

Upon receipt of the Read Track command, the head is loaded and the BUSY Status bit is set. Reading starts with the leading edge of the first encountered index mark and continues until the next index pulse. As each byte is assembled it is transferred to the Data Register and the Data Request is generated for each byte. No CRC checking is performed. Gaps are included in the input data stream. If bit 0 (S) of the command is a 0, the accumulation of bytes is synchronized to each Address Mark encountered. Upon completion of the command, the interrupt is activated.

WRITE TRACK

Upon receipt of the Write Track command, the head is loaded and the BUSY Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon recieving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded by the time the index pulse is encountered the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the Interrupt is activated. If a byte is not present in the DR When needed, a byte of zeros is substituted. Address Marks and CRC characters are written on the disk by detecting certain data byte patterns in the outgoing data stream as shown in the table below. The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR.

CONTROL BYTES FOR INITIALIZATION

DATA PATTERN (HEX)	INTERPRETATION	CLOCK MARK (HEX)
F7	Write CRC Char.	FF
F8	Data Addr. Mark	C7
F9	Data Addr. Mark	C7
FA	Data Addr. Mark	C7
FB	Data Addr. Mark	C7
FC	Index Addr. Mark	D7
FD	Spare	
FE	ID Addr. Mark	C7

The Write Track command will not execute if the DINT input is grounded; instead if the Write Protect Status bit is set and the interrupt is activated. Note that one F7 pattern generates 2 CRC characters.

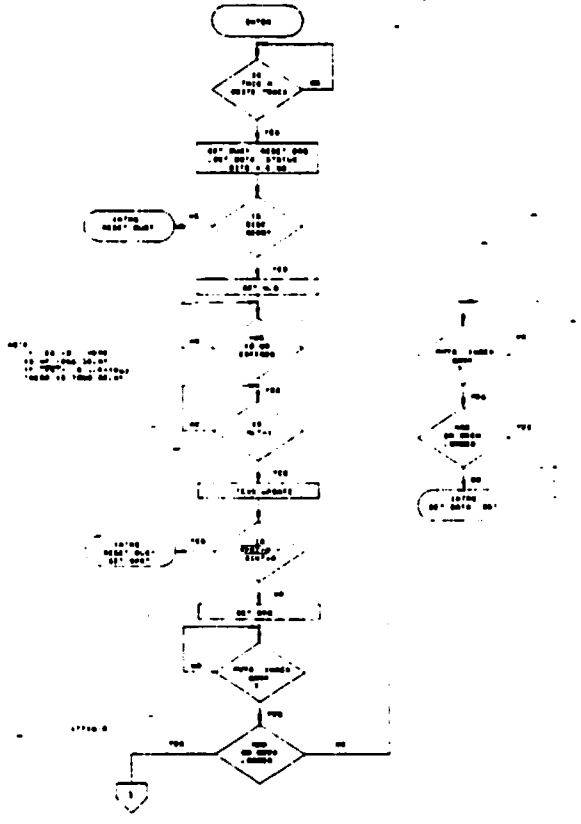


FIG. 11
TYPE III COMMAND
WRITE TRACK

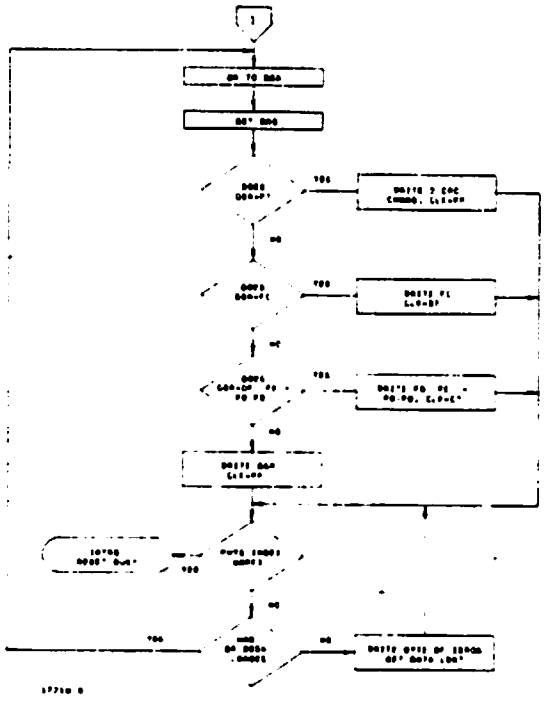


FIG. 12
TYPE III COMMAND
WRITE TRACK

TYPE IV COMMAND

FORCE INTERRUPT

This command can be loaded into the command register at any time. If there is a current command under execution (Busy Status Bit set), the command will be terminated and an interrupt will be generated when the condition specified in the I_0 through I_3 field is detected. The interrupt conditions are shown below:

I_0 = Not-Ready-To-Ready Transition
I_1 = Ready-To-Not-Ready Transition
I_2 = Every Index Pulse
I_3 = Immediate Interrupt

NOTE: If I_0I_3=0, there is no interrupt generated but the current command is terminated and busy is reset.

STATUS DESCRIPTION

Upon receipt of any command, except the Force Interrupt command, the BUSY Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the BUSY status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, theBUSY Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The format of the Status Register is shown below:

(BITS)

7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table 6.

STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ	READ TRACK	WRITE	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	RECORD TYPE	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD ENGAGED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	ID NOT FOUND	RECORD NOT FOUND	0	RECORD NOT FOUND	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

TABLE 6

STATUS FOR TYPE I COMMANDS

BIT NAME	MEANING
S7 Not Ready	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the READY input and logically 'ored' with MR.
S6 PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy of $\overline{WRPT}$ input.
S5 HEAD LOADED	When set, it indicates the head is loaded and engaged. This bit is a logical "and" of HLD and HLT signals.
S4 SEEK ERROR	When set, the desired track was not verified. This bit is reset to 0 when updated.
S3 CRC ERROR	When set, there was one or more CRC errors encountered on an unsuccessful track verification operation. This bit is reset to 0 when updated.
S2 Track 00	When set, indicates Read Write head is positioned to Track 0. This bit is an inverted copy of the $\overline{TROO}$ input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the $\overline{IP}$ input.
S0 BUSY	When set command is in progress. When reset no command is in progress.

STATUS BITS FOR TYPE II AND III COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset, it indicates that the drive is ready. This bit is an inverted copy of the READY input and 'ored' with MR. The TYPE II and III Commands will not execute unless the drive is ready.
S6 RECORD TYPE/ WRITE PROTECT	On read Record: In indicates the MSB of record-type code from data field address mark. On Read Track: Not Used. On any Write Track: It indicates a Write Protect This bit is reset when updated.
S5 RECORD TYPE/ WRITE FAULT	On Read Record: It indicates the LSB of record-type code from data field address mark. On Read Track: Not Used. On any Write Track: It indicates a Write Fault. This bit is reset when updated.
S4 RECORD NOT FOUND	When set, it indicates that the desired track and sector were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated.
S1 DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read operation or the DR is empty on a Write operation. This bit is reset to zero when updated.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA control with a large amount of memory. When operating under DMA with limited amount of memory, formatting is a more difficult task. This is because gaps as well as data must be provided at the computer interface.

Formatting the disk is accomplished by positioning the R/W head over the desired track number and issuing the Write Track command. Upon receipt of the Write Track command, the FD 1771 raises the data request signal. At this point in time, the user loads the data register with desired data to be written on the disk. For every byte of information to be written on the disk, a data request is generated. This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a clock mark of $(FF)_{16}$. However, if the FD1771 detects a data pattern on F7 thru FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation. For instance, an FE pattern will be interpreted as an ID address mark (DATA-FE, CLK-C7) and the CRC will be initialized. An F7 pattern will generate two CRC characters. As a consequence, the patterns F7 thru Fe must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by a F7 pattern.

Disks may be formatted in IBM 3740 formats with sector lengths of 128, 256, 512, or 1024 bytes, or may be formatted in non-IBM 3740 with sectors length of 16 to 4096 bytes in 16 byte increments. IBM 3740 at the present time only defines two formats. One format with 128 bytes/sector and the other with 256 bytes/sector. The next section deals with the IBM 3740 format with 128 bytes/sector and the following section details non-IBM formats.

IBM 3740 FORMATS - 128 BYTES/SECTOR

Shown in Figure 13, is the IBM format with 128 bytes/sector. In order to format this format, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	00 or FF
6	00
1	FC (Index Mark)
26	00 or FF
* 6	00
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	00
1	Sector Number (1 thru 1A)
1	00
1	F7 (2 CRC's written)
11	00 or FF
6	00
1	FB (Data Address Mark
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	00 or FF
247**	00 or FF

* Write bracketed field 26 times

** Continue writing until FD1771 interrupts out. Approx. 247 bytes.

NON-IBM FORMATS

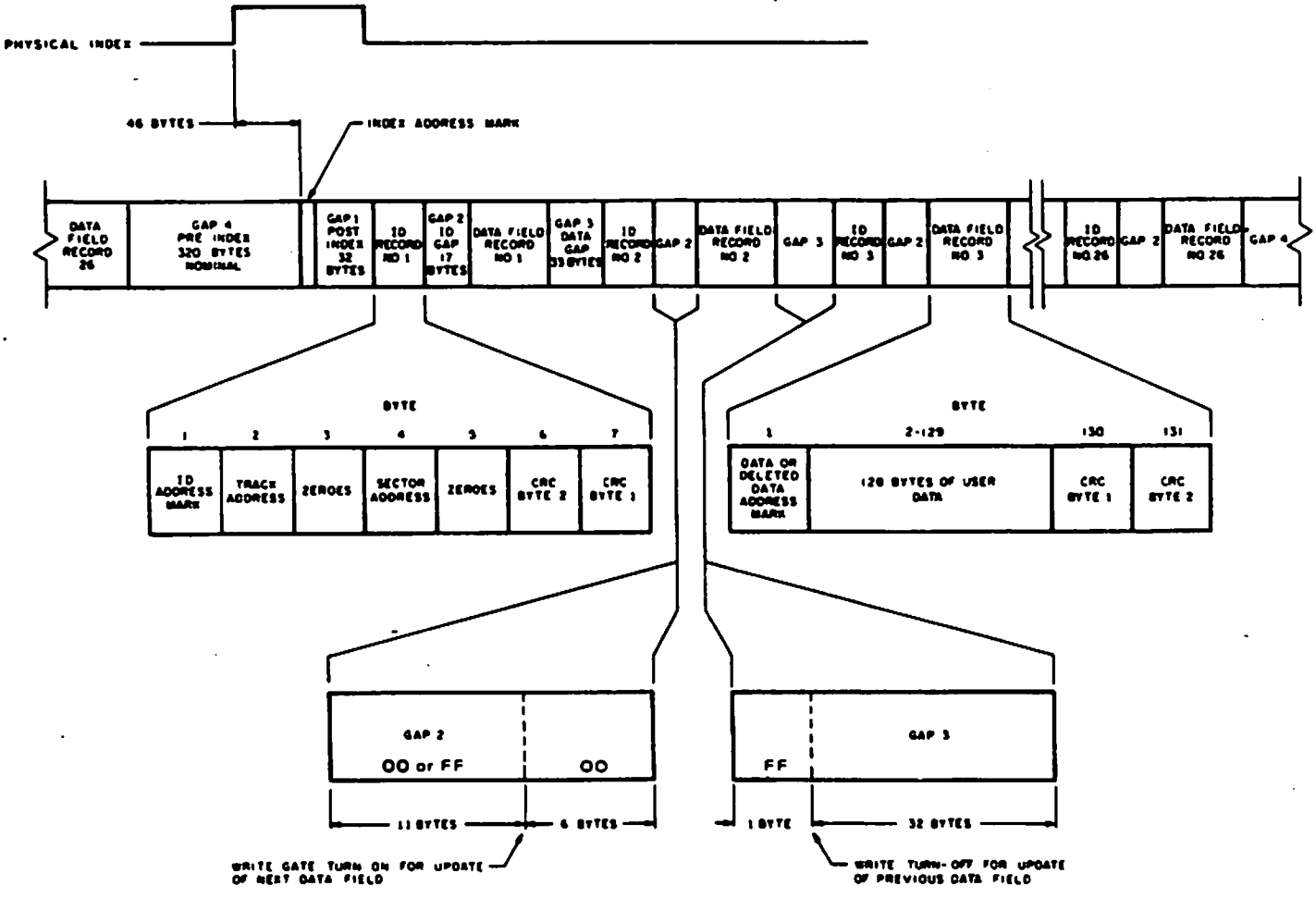
Non IBM Formats are very similar to the IBM formats except a different algorithum is used to ascertain the sector lengt from the sector length byte in the ID field. This permits a wide range of sector lengths from 16 to 4096 bytes. Refer to section V, Type II commands with b flag equal to zero. Note that F7 thru FE must not appear in the sector length byte of the ID field.

In formatting the FD1771, only two requirements regarding GAP sizes must be met. GAP 2 (i.e., the gap between the ID field and data field must be 17 bytes of which the last 6 bytes must be zero and that every address mark be preceded by at least one byte of zeros. However, it is recommended that every GAP be at least 17 bytes long with 6 bytes of zeros. The FD1771 does not require the index address mark (i.e., DATA = FC, CLK = D7) and need not be present.

References:

1) IBM Diskette OEM Information GA21-9190-1
2) SA900 IBM Compatibility Reference Manual - Shugart Associates.

FIG.13
TRACK FORMAT



ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

V_{DD} With Respect to V_{BB} (Ground)	+20 to -0.3V
Max Voltage to Any Input With Respect to V_{BB}	+20 to -0.3V
Operating Temperature	0°C to 70°C
Storage Temperature	-55°C to +125°C

OPERATING CHARACTERISTICS (DC)

T_A = 0°C to 70°C, V_{DD} = +12.0V ± .6V, V_{BB} = −5.0 ± .5V, V_{SS}= 0V, V_{CC} = +5V ± .25V
V_{DD} = 10 ma Nominal, V_{CC} = 30 ma Nominal, V_{BB} = 0.4 ua Nominal

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
I_{LI}	Input Leagage			10	uA	$V_{IN} = V_{DD}$
I_{LO}	Output Leakage			10	uA	$V_{OUT} = V_{DD}$
V_{IH}	Input High Voltage	2.6			V	
V_{IL}	Input Low Voltage (All Inputs)			0.8	V	
V_{OH}	Output High Voltage	2.8			V	I_O = −100 uA
V_{OL}	Output Low Voltage			0.45	V	I_O = 1.6 mA

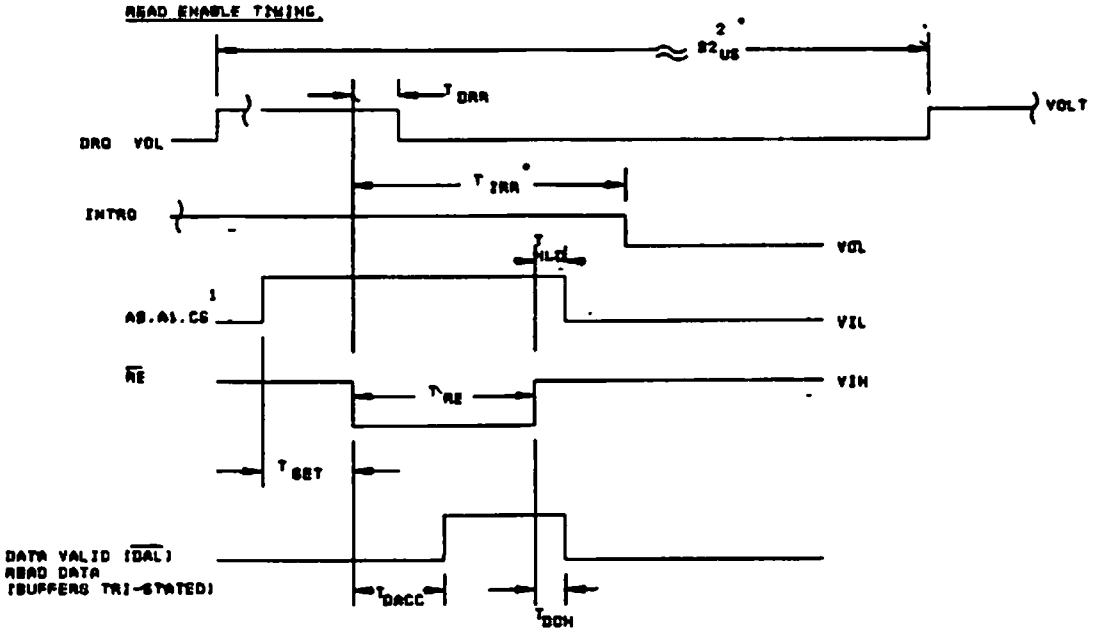
NOTE: Vol ≤.4V when interfacing with low Power Schottky parts (1_O<1 ma)

TIMING CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± .6V, V_{BB} = −5 ± .25V, V_{SS} = 0V, V_{CC} = +5 ± .25V
NOTE: Timings are given for 2 MHZ Clock. For those timings noted, values will double when chip is operated at 1 MHZ. Use 1 MHZ when using mini-floppy.

Read Operations

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TET	Setup ADDR & CS to $\overline{RE}$	100			nsec	
THLD	Hold ADDR & CS from $\overline{RE}$	10			nsec	
TRE	$\overline{RE}$ Pulse Width	500			nsec	C_L = 25 pf
TDRR	DRQ Reset from $\overline{RE}$			150	nsec	
TIRR	INTRQ Reset from $\overline{RE}$			3000	nsec	
TDACC	Data Access from $\overline{RE}$			450	nsec	C_L= 25 pf
TDOH	Data Hold From $\overline{RE}$	50		150	nsec	C_L = 25 pf



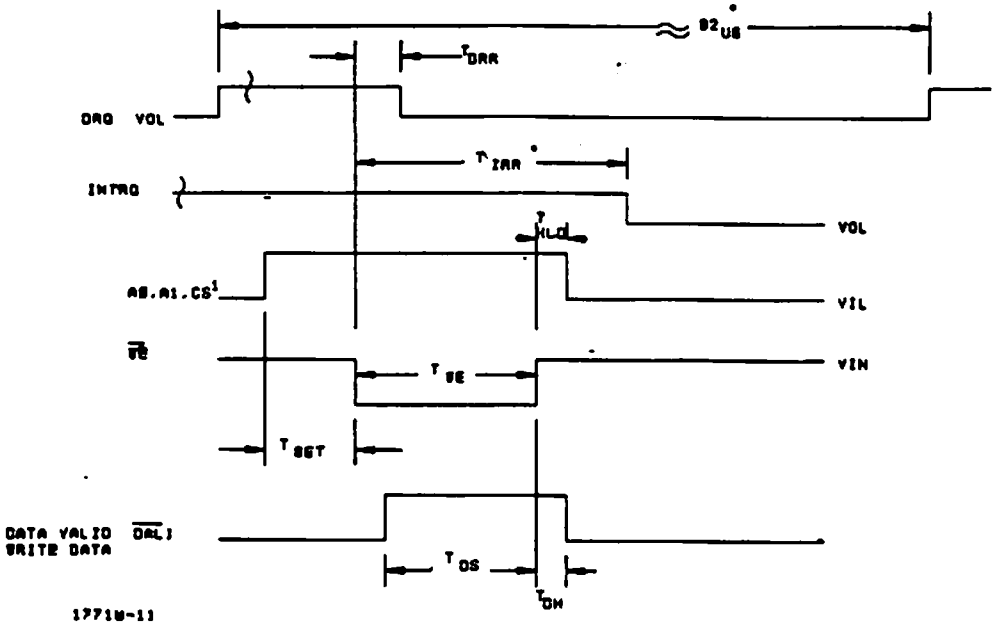
NOTE: 1. $\overline{CS}$ MAY BE PERMANENTLY TIED LOW IF DESIRED.
2. FOR READ TRACK COMMAND, THIS TIME MAY BE 12 TO 32 USEC WHEN 8-8.

* TIME DOUBLES WHEN CLK=1 MHZ

Write Operations

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TSET	Setup ADDR & CS to $\overline{WE}$	100			nsec	
THLD	Hold ADDR & CS from $\overline{WE}$	10			nsec	
TWE	$\overline{WE}$ Pulse Width	350			nsec	
TDRR	DRQ Reset from $\overline{WE}$			150	nsec	See Note
TIRR	INTRQ Reset from $\overline{WE}$			3000	nsec	
TDS	Data Setup to $\overline{WE}$	250			nsec	
TDH	Data Hold from $\overline{WE}$	150			nsec	

WRITE ENABLE TIMING



1771M-11

NOTE: 1. $\overline{CS}$ MAY BE PERMANENTLY TIED LOW IF DESIRED.
2. WHEN WRITING DATA INTO SECTOR, TRACK, OR DATA REGISTER, USER CANNOT READ THIS REGISTER UNTIL AT LEAST 8 usec AFTER THE RISING EDGE OF $\overline{WE}$ WHEN WRITING INTO THE COMMAND REGISTER, STATUS IS NOT VALID UNTIL SOME 12 usec LATER. THESE TIMES ARE DOUBLED WHEN CLK=1 MHz.

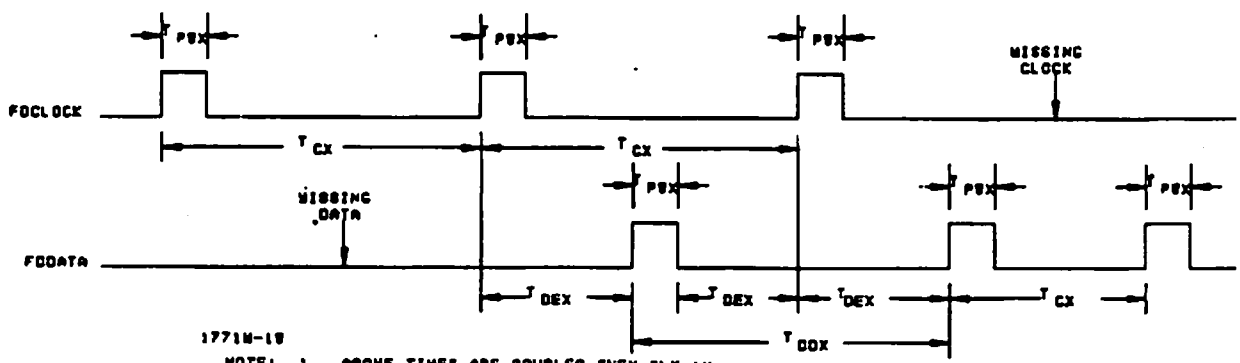
* = Time doubles when CLK = 1 MHZ

External Data Separation ($\overline{XTDS}$ = 0)

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TPWX	Pulse Width Rd Data & Rd Clock	150		350	nsec	
TCX	Clock Cycle Ext	2500			nsec	
TDEX	Data to Clock	500			nsec	
TDDX	Data to Data Cycle	2500			nsec	

READ TIMING

$\overline{XTDS}$=0
EXTERNAL DATA SEPERATION



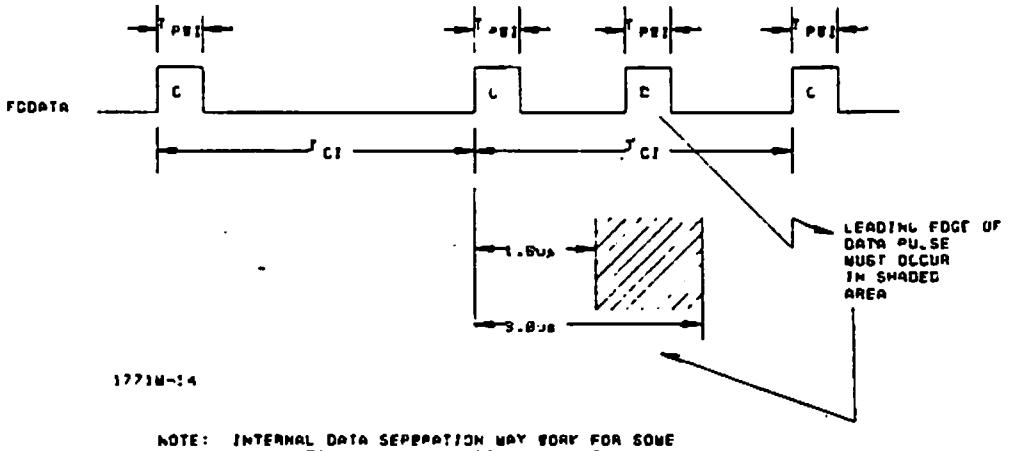
1771M-19

NOTE: 1. ABOVE TIMES ARE DOUBLED WHEN CLK=1MHZ.
2. CONTACT WDC FOR EXTERNAL CLOCK/DATA SEPERATOR CIRCUITS.

Internal Data Separation (XTDS = 1)

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TPWI	Pulse Width Data & Clock	150		1000	nsec	
TCI	Clock Cycle Internal	3500		5000	nsec	

READ TIMING
XTDS=1
INTERNAL DATA SEPERATION
FDCLOCK MUST BE TIED HIGH

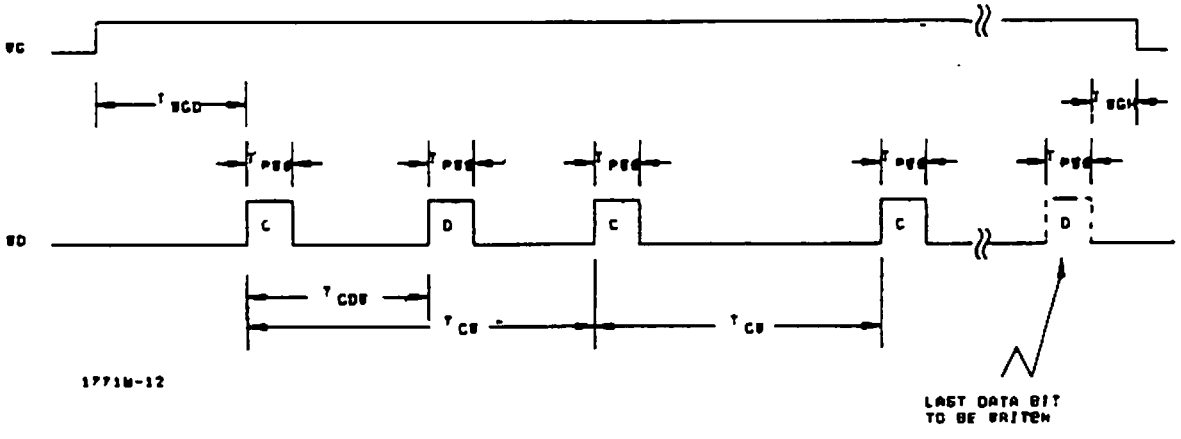


NOTE: INTERNAL DATA SEPERATION MAY WORK FOR SOME APPLICATIONS. HOWEVER, FOR APPLICATIONS REQUIRING HIGH DATA RECOVERY RELIABILITY, SDC RECOMMENDS EXTERNAL DATA SEPERATION BE USED.

Write Data Timing:

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TWGD	Write Gate to Data		1200		nsec	300 nsec ± CLK tolerance
TPWW	Pulse Width Write Data	500		600	nsec	
TCDW	Clock to Data		2000		nsec	±0.5%± CLK tolerance
TCW	Clock Cycle Write		4000		nsec	±0.5%± CLK tolerance
TWGH	Write Gate Hold to Data	0		100	nsec	

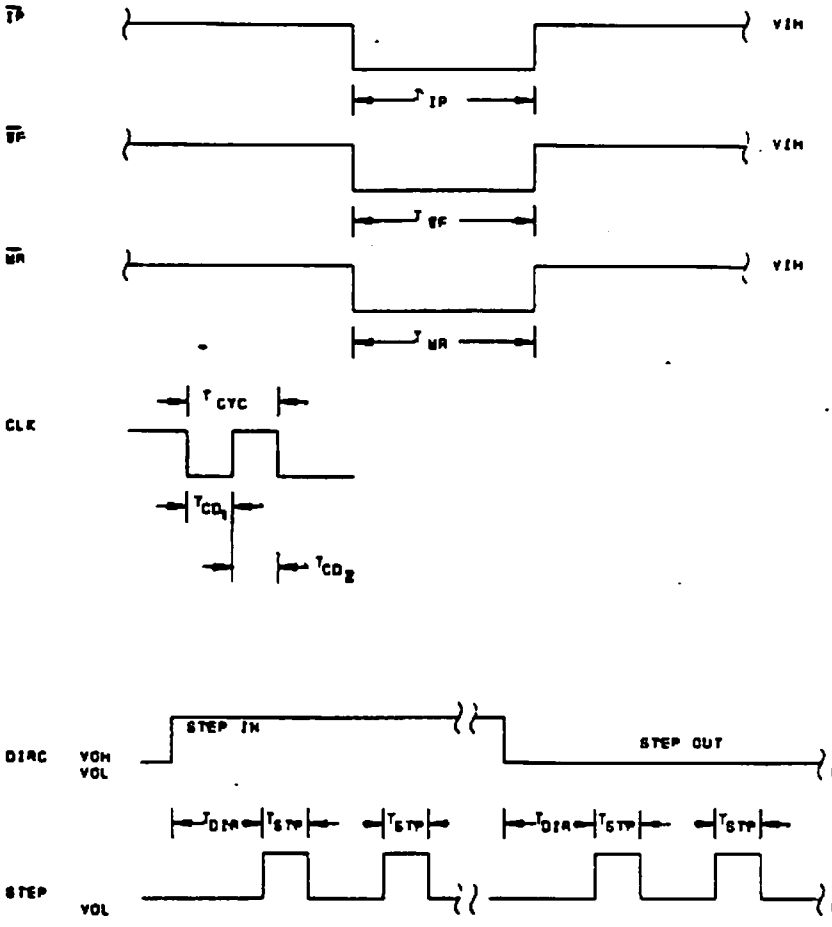
WRITE DATA TIMING



Miscellaneous Timing:

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
TCD_1	Clock Duty	175			nsec	2MHZ ± 1% See Note
TCD_2	Clock Duty	210			nsec	
TSTP	Step Pulse Output	3800		4200	nsec	
TDIR	Dir Setup to Step	24			usec	These times doubled when CLK = 1 MHZ
TMR	Master Reset Pulse Width	10			usec	
TIP	Index Pulse Width	10			usec	
TWF	Write Fault Pulse Width	10			usec	

MISCELLANEOUS TIMING



PIN OUTS

PIN NO.	PIN NAME	SYMBOL	FUNCTION
1	Power Supplies	V_{BB}	−5V
20		V_{SS}	Ground
21		V_{CC}	+5V
40		V_{DD}	+12V
19	MASTER RESET	$\overline{MR}$	• A logic low on this input resets the device and clears the command register. The Not Ready (Status Bit 7) is reset during $\overline{MR}$ ACTIVE. When $\overline{MR}$ is brought to a logic high a Restore Command is executed, regardless of the state of the Ready signal from the drive.

PIN NO	PIN NAME	SYMBOL	FUNCTION
Computer Interface:			
7-14	DATA ACCESS LINES	$\overline{DAL0}$-$\overline{DAL7}$	• Eight bit inverted Bidirectional bus used for transfer of data, control, and status. This bus is a receiver enabled by $\overline{WE}$ or a transmitter enabled by $\overline{RE}$.
3	CHIP SELECT	$\overline{CS}$	• A logic low on this input selects the chip and enables computer communication with the device.
5,6	REGISTER SELECT LINES	A0, A1	•These inputs select the register to receive/transfer data on the DAL lines under $\overline{RE}$ and $\overline{WE}$ control: A1 A0 $\overline{RE}$ $\overline{WE}$ 0 0 Status Reg Command Reg 0 1 Track Reg Track Reg 1 0 Sector Reg Sector Reg 1 1 Data Reg Data Reg
4	READ ENABLE	$\overline{RE}$	•A logic low on this input controls the placement of data from a selected register on the DAL when $\overline{CS}$ is low.
2	WRITE ENABLE	$\overline{WE}$	•A logic low on this input gates data on the DAL into the selected register when $\overline{CS}$ is low.
38	DATA REQUEST	DRQ	•This open drain output indicates that the DR contains assembled data in Read operations, or the DR is empty in Write operations. This signal is reset when serviced by the computer through reading or loading the DR in Read or Write operation, respectively. Use 10K pull-up resistor to +5.
39	INTERRUPT REQUEST	INTRQ	•This open drain output is set at the completion or termination of any operation and is reset when a new command is loaded into the command register. Use 10K pull-up resistor to +5.
24	CLOCK	CLK	•This input requires a free-running 2 MHz + 1% square wave clock for internal timing reference.
Floppy Disk Interface:			
25	EXTERNAL DATA SEPERATION	XTDS	•A logic low on this input selects external data separation. A logic high or open selects the internal data separator.
26	FLOPPY DISK CLOCK (External Separation)	FDCLOCK	•This input receives the externally separated clock when XTDS = 0. If XTDS = 1, this input should be tied to a logic high.
27	FLOPPY DISK DATA	FDDATA	•This input receives the raw read disk data if XTDS = 1, or the externally separated data if XTDS = 0.
31	WRITE DATA	WD	•This output contains both clock and data bits of 500 ns duration.
28	HEAD LOAD	HLD	•The HLD output controls the loading of the Read-Write head against the media the HLT input is sampled after 10 ms. When a logic high is sampled on the HLT input the head is assumed to be engaged.
23	HEAD LOAD TIMING	HLT	
15	Phase 1/Step	PH1/STEP	•If the 3PM input is a logic low the three phase motor control is selected and PH1, PH2, and $\overline{PH3}$ outputs form a one active low signal out of three. PH1 is active low after MR. If the 3PM input is a logic high the step and direction motor control is selected. The step output contains a 4usec high signal for each step and the direction output is active high when stepping; active low when stepping out.
16	Phase 2/Direction	PH2/DIRC	
17	Phase 3	$\overline{PH3}$	
18	3 Phase Motor Select	3PM	

PIN NO.:	PIN NAME:	SYMBOL:	FUNCTION
29	Track Greater Than 43	TG43	•This output informs the drive that the Read-Write head is positioned between track 44-76. This output is valid only during Read and Write Commands.
30	WRITE GATE	WG	•This output is made valid when writing is to be performed on the diskette.
32	Ready	READY	•This input indicates disk readiness and is sampled for a logic high before Read or Write commands are performed. If Ready is low the Read or Write operation is not performed and an interrupt is generated. A Seek operation is performed regardless of the state of Ready. The Ready input appears in inverted format as Status Register bit 7.
33	$\overline{\text{WRITE FAULT}}$	$\overline{\text{WF}}$	•This input detects writing faults indications from the drive. When WG = 1 and WF goes low the current Write command is terminated and the Write Fault status bit is set. The WF input should be made inactive (high) when WG becomes inactive.
34	$\overline{\text{TRACK 00}}$	$\overline{\text{TR00}}$	•This input informs the FD1771 that the Read-Write head is positioned over Track 00 when a logic low.
35	$\overline{\text{INDEX PULSE}}$	$\overline{\text{IP}}$	•Input, when low for a minimum of 10 usec, informs the FD1771 when an index mark is encountered on the diskette.
36	$\overline{\text{WRITE PROTECT}}$	$\overline{\text{WPRT}}$	•This input is sampled whenever a Write Command is received. A logic low terminated the command and sets the Write Protect Status bit.
37	$\overline{\text{DISK INTIALIZATION}}$	$\overline{\text{DINT}}$	•The input is sampled whenever a Write Track command is received. If $\overline{\text{DINT}}$ = 0, the operation is terminated and the Write Protect Status bit is set.
22	$\overline{\text{TEST}}$	$\overline{\text{TEST}}$	•This input is used for testing purposes only and should be tied to +5V or left open by the user.

INSTALLATION NOTES

APPENDIX H:

NOTES ON INSTALLATION AND POWER UP

Unpacking and Inspecting a New Terminal

A 3600 Series Terminal or Desktop Computer is shipped from the factory in a reinforced cardboard carton which is packed with plastic foam. When received, the carton should be inspected for damage **before** the unit is unpacked. If damage is discovered, the shipper and Intelligent Systems Corp. should be notified.

After unpacking and before installation, the unit itself should be subjected to a careful visual inspection. It is a good idea to remove the cabinet cover and to verify that all cables within the unit are properly connected and that the circuit boards and CRT are secure. (The cover is held by two screws situated at the front edge of the cabinet below the keyboard. The cover is held by two screws situated below the front edge of the keyboard. After these screws are removed, the front of the cover can be lifted straight up. The cover will pivot on a flange at its rear edge until this flange is disengaged from the unit frame. The cover can then be pulled forward and off the unit.) Any damage should be reported to Intelligent Systems Corp.

While the cover is off, the setting of SW2, the line voltage selector switch mounted on the main circuit board, should be checked. This switch must be set for either 115V or 220V, in accordance with the line voltage at the site of installation.

> **WARNING:** Work within the cabinet should be performed only by persons who are ware of the possible hazards and are trained and qualified for this type of work.
>
> Hazardous voltages may exist within the terminal when power has been applied.

The Operating Environment

When installing a 3600 Series unit, it is important to take account of environmental factors at the site which could affect the terminal's performance.

The unit's **ambient temperature** should be maintained between 0^{o} and 40^{o} Centigrade, and it should be situated in such a way that its ventillating ports remain unobstructed. Relatively cool ambient air must be able to enter these ports and to be pulled through the cabinet by the

unit's fan in order to keep the unit's electronic components cool.

The unit should not be situated within strong **magnetic fields**, such as are present around large transformers. Magnetic fields can cause distortion of the display on the CRT screen.

Airborne **dust and smoke** particles can cause problems since they are attracted by the unit's high voltage components and can collect at ventillating holes.

In especially **dry environments**, where electrostatic charges are easily created, it may be necessary for the operator to take precautions lest his handling of the equipment result in a discharge which could damage an integrated circuit. In such situations the operator can generally prevent damage by touching an earth ground before touching the keyboard. The unit is equipped with a three-conductor power cord which, when properly installed, ties the unit's chassis to earth ground. Accordingly, the operator may conveniently "ground" himself by touching some exposed metal point on the unit.

The 3600's display is visible under normal indoor **lighting** conditions, and its brightness control compensates for some variation from such conditions. Accordingly, the area in which the 3600 is installed normally need not be darkened. However, the screen should be kept out of the glare of especially bright artificial light or sunlight.

IMPORTANT NOTE: Radio frequency emissions from the 3600 do not exceed the limits set by the FCC (in its Rules, Part 15, Subpart J) for class A computing devices when it is properly installed. With regard to the minimization of RF emissions, proper installation involves

a) the connection of the unit's three-conductor power cord to a three-hole AC outlet, without use of a three-prong to two-prong plug adapter,

b) the use of the unit's cabinet cover, and

c) the installation of interconnecting cables according to the instructions given below.

The inside surfaces of the cabinet are coated with a conductive paint which serves as a shield for the equipment. This shield is important not only to contain RF emissions by the terminal but also to contain X-ray emissions and to protect the terminal itself from external interference.

It should be kept in mind that the unit radiates a small amount of radio frequency energy even when the cover is installed. This radiation can sometimes interfere with sensitive communications equipment (including but not limited to television receivers) and with certain types of electronic instruments operating in the immediate vicinity. Precautions should be taken where appropriate.

Power-Up

WARNING: Before applying power to the unit, verify that the line voltage selector switch, SW2, on the main circuit board is properly set and that the proper fuse is installed in the fuse holder on the rear panel. The fuse should be

1.5 Ampere 250V "Slo-Blo", type 3AG, for 115V operation, or

.75 Ampere 250V "Slo-Blo", type 3AG, for 220V operation.

For continued protection against fire and equipment damage the fuse should be replaced oly with a fuse of the proper type and rating.

WARNING: Operate the unit from a single-phase AC source only. The power circuits are not designed for two-phase sources. Do not defeat the purpose of the three-prong power plug (safety) by cutting or other wise disabling the ground prong.

WARNING: Connection to a source which provides more than 250V RMS could result in serious damage to the equipment.

After the 3600 has been inspected it may be connected to an AC source, switched on and tested. Initial tests should be made with a host and any peripheral devices disconnected. The keyboard should be connected (at rear panel edge connector J1). See Chapter One in this manual for procedures.

Before attaching the power cord, verify that the lower edge of the white power switch on the rear panel is depressed (in the off position). Insert the power cord into its socket on the rear panel and then connect it to a source of signle-phase AC power.

Turn the power switch to the on position (upper edge depressed).

Note: After applying power, it may be necessary to adjust the brightness control, a potentiometer mounted on the rear panel, to make the display visible.

Interconnection

Three connectors are provided at the rear panel of the 3600 for connection to other equipment.

The **keyboard** connector, J1, is the 26 pin edge connector which appears at the left when one faces the rear panel. This connector is slotted to accept a key in its mating connector, an arrangement which prevents accidental reversal of the keyboard cable. Since the keyboard and its interconnecting cable are part of the 3600, the technician is not ordinarily concerned with J1's pin assignments when he is installing the unit. The pin assignments may be obtained from the schematic drawing of the logic circuitry if needed.

The **RS-232C** connector, J2, is a 25 pin female AMP type at the center of the rear panel. This connector can be used for various purposes; a serial printer, terminal, or host may be connected here.

The signals available at this connector include MODEM control signals (RS-232 levels) and serial data transmit and receive signals (RS-232 levels). The need for MODEM control line connections varies from installation to installation; in many cases some or all of these lines can be left disconnected.

The pin assignments for J2 are given on page 160 of this manual. The cable between the 3600 and an external device should be kept as short as possible for best results. Use of a cable made up of twisted conductor pairs is recommended. When such a cable is available, one conductor of a twisted pair should be used for each signal line and the other conductor should be tied to signal ground. In some installations, shielding of the entire cable may be required. In such cases, the cable shield should be tied to chassis ground at either the host or the terminal end.

Internal buss (X-Buss) connector J3 is a 50-pin edge connector which appears to the right as one faces the rear panel. This connector provides the user with access to the unit's address, data and control busses. It may be used for test purposes or for the expansion of the unit's I/O or memory system. Note that the utilization of the signals at this connector generally requires modification of the system utilities by a programmer familiar with 8080 assembly language. The pin assignments for this connector and a discussion of restrictions on line loading are given on page 159. In general, signal lines will handle one TTL load each.

Cable lines connected at J3 should be no longer than eight feet. The cable should be a ribbon type, with signal lines separated by ground lines, or a bunch of twisted pairs, each pair consisting of one signal line and one signal ground line.

ADJUSTMENT AND ALIGNMENT

APPENDIX I: ADJUSTMENT

All 3600 Series units are thoroughly tested and adjusted at the factory before they are shipped. They generally require complete readjustment only after extensive repairs have been made. However, minor adjustments may be required when a terminal has been subjected to physical shocks in the course of transportation or when environmental conditions have been changed significantly. Minor adjustments may also be required from time to time to compensate for the effects of component aging.

The instructions given in this section of the manual are designed to enable the technician to keep the 3600 units in optimal operating condition throughout their courses of service. While not specifically intended for trouble-shooting, the procedures can be useful, too, as diagnostic tools. It is recommended that a periodic preventive maintenance routine include a touch-up of adjustments as described below.

The instructions are given in step-by-step form and describe a complete adjustment procedure such as is done at the factory to establish terminal functions for the first time. When the terminal requires only a routine touch-up of its controls, a short form of the procedure may be used. However, it is recommended that whatever steps are performed be performed in the order given. The order reflects the interdependencies among circuit functions; its observance will save time and protect the equipment from damage.

A shorter touch-up procedure is constituted by those steps which are marked by a double asterisk. Unmarked steps may generally be omitted unless there is a need to check for malfunctions or to guard against damage which could result from severe misadjustments of controls.

WARNING: Because of the dangers, both to personal safety and to equipment, associated with work on the 3600 units, only properly qualified persons should perform service.

Tools and Equipment Needed

#2 Phillips screwdriver

Screwdriver with 1/4" flat blade

Plastic alignment tool (universal type)

Multirange DC voltmeter (20K ohms/V or better, digital recommended)

High voltage probe for meter

Oscilloscope (DC to 15 MHz)

0-1 DC milliammeter, insulated for at least 35KV (optional, for adjustment of CRT beam current)

Cabinet Cover Removal and Other Preliminary Steps

The controls to be adjusted are situated on circuit boards within the unit's cabinet. All the controls are accessible upon removal of the cabinet cover. Accordingly, no disassembly beyond cabinet cover removal is required for normal adjustment.

The cabinet cover is held by two screws at the front edge of the unit below the keyboard. Once the screws have been removed, the cover must be grasped at its sides and lifted up at the front. A flange at its rear edge pivots on the chassis frame. When the front of the cover has been lifted 12" to 18", the cover can be pulled forward slightly to release its flange from the chassis support. It may then be set aside.

When adjustments are to be made following trouble-shooting and repair work and circuit boards have been removed from the unit, it will generally be most convenient to make some initial adjustments before the unit is fully reassembled. The tests and adjustments of power supply circuits described in steps 1 - 6 may be performed with the analog board removed from the chassis and disconnected from other boards. A sheet of cardboard or other suitable material should be placed under the board to insulate it from the work table. Subsequent steps require that the analog board be connected to other circuits. Interconnection requires that the board be installed in the chassis unless the standard cables are replaced with extended cables, which can be made up in the shop.

WARNING: Once the cabinet cover has been removed, do the following before applying AC power:

a) Make sure that the line voltage selector switch, SW2 on the analog board, is properly set for either 115V or 220V (single phase only).

b) Verify that the proper fuse (1.5A, 250V, "Slo-Blo" for 115V or .75A, 250V "Slo-Blo" for 220V) is installed in the line fuse holder on the rear panel.

WARNING: Under no circumstances should the unit be powered up with circuit boards improperly secured and resting on metal frame parts or on the inner surfaces of the cabinet. Short circuits and/or the appearance of dangerous voltages on the frame could result. (The inside surfaces of the unit's plastic base pan and cover are sprayed with a conductive paint.)

1) The complete alignment of a 3600 Series unit begins with checks and adjustment of the low voltage power supplies on the analog board. These initial steps are taken with the horizontal and vertical deflection coils, video, logic and high voltage circuits disconnected. Disconnection prevents damage to these components due to improper supply voltages or control settings.

   Before applying AC power to the analog board, make certain that the cables terminating at the following points are disconnected:

   J6 (vertical deflection coil)
   J7 (horizontal deflection coil and HV interlock)
   J8 (video board power)
   J10 (logic board power)
   J11 (sync)
   J12 (mini disk drive power - 3650)

   In addition, the black wires to the CRT ground strapping, the CRT anode lead, the brightness control cable (terminating at J9), the fan leads (terminating at J13), and the degaussing coil leads (terminating at J2) may be disconnected.

   Double-check the position of SW2, the line voltage selector switch.

2) If there is reason to believe that substantial misadjustment of any or all of the controls has occurred, they should be set initially as follows:

Analog Board

R13	5V Adjust	Mid-range
R19	Power Supply Frequecy	Mid-range
R22	Overcurrent Sensitivity	Full CW
R31	High Voltage Adj.	Mid-range
——	Focus	Mid-range
——	Screen (CRT G2)	1/8 CW See step 15
R115	Brightness	Full CW
R96	Hor. Freq. (Hold)	3/4 CW
R106	Hor. Centering	Mid-range
L4	Hor. Linearity	Arbitrary
L5	Hor. Size	Slug in
R67	Vert. Size	Mid-range
R68	Vert. Linearity	Mid-range
R89	Vert. Centering	Mid-range
R84	Pincushion	Full CCW

Video Board

R1 or R17	Red Gain	3/4 CW
R3 or R6	Green Gain	3/4 CW
R5 or R3	Blue Gain	3/4 CW

Low Voltage Power Circuitry

3) After applying power, use an accurate DC voltmeter to verify the presence of

+12V, +/- .5V, at pin 1, J10 (GND is pins 7 and 8)
-12V, +/- .5V, at pin 2, J10
-5V, +/- .25V, at pin 6, J10

4) With a DC voltmeter, check for +70V, +/- 5V, (in reference to chassis ground) at the ungrounded end of R9.

5) With a DC voltmeter, check for +5V at pins 4 and 5, J10 (pins 7 and 8 are ground). Adjust R13 (5V ADJ) for exactly 5.00V.

6) Remove AC power. Make the appropriate connections of the cables terminating at J2, J6, J7, J8, J9, J10, J11, J12 (3650 Series only) and J13 on the analog board. The CRT anode lead, focus (G3) lead, the black wire from the CRT ground strapping to the analog and video boards, and the video cable between the video and logic boards must also be connected. These connections activate the horizontal, vertical, high voltage, logic and video circuits and the degaussing coil.

For 3650 units, refer to schematic drawing 102428 for details about interconnections.

Insure that the keyboard cable is connected.

WARNING: J5 on the analog board is must under no circumstances be used. This connector is used on 2400 Series units only.

** 7) Apply AC power. Adjust R13 (5V ADJ) for exactly +5.00V (+.05V/-.00V) as measured at pins 4 and 5, J10 (pins 7 and 8 are ground).

NOTE: The power supply frequency control, R19, is adjusted later, if necessary, after the horizontal and high voltage circuitry has been adjusted.

High Voltage Circuitry

** 8) With AC power removed, connect a reliable DC voltmeter with high voltage probe to the unit. This is done by first fixing the ground clip of the probe to a suitable point on the chassis and then slipping the tip of the probe under the anode cap of the CRT, making certain that a good connection is maintained.

** **NOTE:** Unless otherwise indicated, all instructions for giving keyboard commands suppose that the terminal is operating in the Local mode. Most 3600 units default to Local Mode operation at power up. However, some are equipped for other types of initialization. A unit may always be placed in Local Mode by striking the following keys in sequence:

(ATTN BREAK) (ESC) L

** 9) After applying AC power and insuring that the unit is in Local Mode, obtain a black screen with the following keystrokes:

(HOME) (ERASE PAGE)

The cursor will remain visible in the upper left corner of the screen.

A black screen is coincident with minimum CRT beam current, at which the high voltage should be set.

** 10) Adjust R31 for a reading of 25KV on the meter.

NOTE ON X-RADIATION: At an anode voltage of 25KV, X-ray emissions from the CRT will average about .05 mRh/hr. Factory tests of the tube at 30KV and higher indicate a radiation level of .18 mRh/hr maximum. Thus, throughout the range of adjustment of the high voltage supply, levels are well below the .5 mRh/hr maximum prescribed by the Dept. of HEW. Nevertheless, it is recommended that the tube not be operated at higher anode voltages.

** 11) Adjustment of the focus control is facilitated by placing a pattern of white dots over a black background on the screen. The following sequence of keyboard operations causes this display to appear:

(PG ON) (CONTROL W)

(ESC) Y .

The focus control (adjacent to the flyback transformer on the side of the analog board) is adjusted to maximize the convergence of the CRTs beams as indicated by the sharpness and color of the dot pattern.

12) The CRT control grid (G2) bias is adjusted with the control labeled "screen", situated immediately below the focus control. The "screen" control is carefully set at the factory and an adhesive is placed on the shaft to hold it in place. It does not normally require readjustment; however, it may be readjusted provided a properly insulated DC milliameter is available for measuring the CRT anode current. If such a meter is not available this control should be readjusted only under emergency conditions.

Meter method (preferred)

a) Remove AC power. Use a screwdriver to break the adhesive seal holding the "screen" control shaft in place. Set the control at the full CCW position. Make sure that the brightness control is in the full CW (maximum) position. Connect a 0-1 DC milliameter, insulated for at least 35KV, in series with the CRT anode lead.

b) Restore AC power. Allow a minute for the unitto warm up. Insure that the unit is in Local Mode and obtain a black screen (see above for commands). Gradually increase the screen control setting until a dim raster with retrace lines appears on the screen. Decrease the control setting until the raster just disappears.

c) Decrease the brightness setting to about 3/4 CW. Use the following sequence of keystrokes to place a white raster on the screen:

(BG ON) (CONTROL W) (ERASE PAGE)

While observing the milliameter, adjust the brightness control and screen control alternately, a little at a time, until the brightness control is full CW (maximum) and a beam current of 500 microamperes is obtained. (The screen control should be at about 1/8 turn clockwise from full CCW.) At no time should the current be permitted to rise much above 500 microamperes.

d) Without changing the setting of the screen control, turn the brightness control all the way down. It should be possible to completely darken the screen. Gradually restore full brightness. The beam current should come up smoothly to 500 uA.

Note: The setting of the screen control obtained here is provisional, since beam current is affected by the video gain controls which are to be adjusted later. Therefore, the milliameter should not be disconnected and the screen control shaft should not be resealed untile video adjustments have been made.

Emergency Method

a) Remove AC power. Use a screwdriver to break the adhesive seal on the screen control shaft. Turn the shaft to the full CCW position. Leave the brightness control at the full CW position. Make certain that the Overcurrent control is at its full CW position.

b) Restore AC power. Allow a minute for the unit to warm up. Insure that the unit is in Loca Mode and obtain a black screen (see above for commands). Gradually increase the screen control setting until a dim raster with retrace lines appears on the screen. Decrease the control setting until the raster just disappears, then decrease it an additional 5^{o}. The shaft should be about 1/8 turn CW from full CCW.

c) Reduce the brightness control setting to about 3/4 CW. Use the following sequence of keystrokes to place a white raster on the screen:

(BG ON) (CONTROL W) (ERASE PAGE)

Gradually increase the brightness to maximum. A satisfactory maximum brightness should be obtained without tripping the overcurrent protection circuitry. If the raster will not brighten sufficiently, the screen control is set too low. Gradually decrease the brightness. It should be possible to completely darken the screen. If complete darkening is not obtainable, the screen control is set too high. Repeat steps a and b if necessary.

d) Set the brightness control to maximum. Use the following sequence of keystrokes to obtain a black screen:

(BG ON) (CONTROL P) (ERASE PAGE)

There should be no raster ghost.

** 13) A check of the high voltage regulating circuitry is made with a voltmeter and high voltage probe connected as for steps 10-12. While observing the meter reading, switch from a black screen to a white screen using the following keystrokes:

(HOME) (ERASE PAGE)

(BG ON) (CONTROL P) (ERASE PAGE)

(BG ON) (CONTROL W) (ERASE PAGE)

The anode voltage indicated by the meter should drop to around 24.2KV with a white screen (maximum supply loading) and rise to

25.0KV with a black screen (minimum loading). When switching between a black raster and a white raster, the voltage swing at the anode should not exceed 1KV.

Horizontal and Vertical Deflection Circuitry

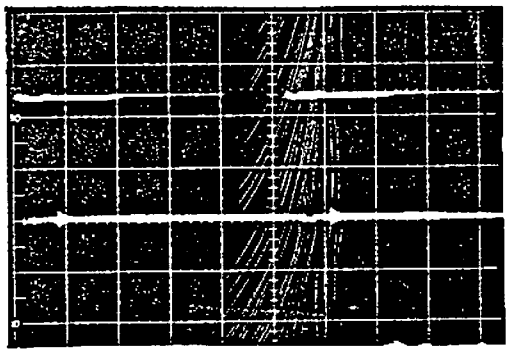
14) The horizontal oscillator frequency can be adjusted in either of two ways. With both methods the aim is to make the oscillator frequency somewhat lower than the frequency of the synchronizing signal from the logic circuitry.

a) A DC voltmeter can be used to monitor the voltage at pin 9 of U6 while R96, the horizontal frequency or "hold" control, is adjusted for a reading of +6 to +6.5V.

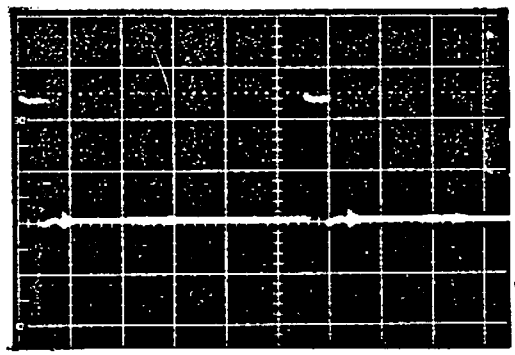
b) The alternative is to adjust R96 while monitoring the waveform of the signal at pin 2 of U6 with an oscilloscope. On an oscilloscope, the signal will appear as shown in one of the photographs below. Turn R96 counter-clockwise, if necessary, to obtain a signal similar to the one at left. Turn R96 clockwise until the edges of the signal appear sharply defined, as in the photograph at the right. Continue turning the control clockwise another 30 degrees.

Oscilloscope
Sync - Int Coupling - DC
Sweep - 10 microsec/div Vert - 10 V/div

Oscillator out of sync



Oscillator in sync



15) Using an oscilloscope, measure the DC voltage at the collector of Q16 (tied to R74). It should be about +45V.

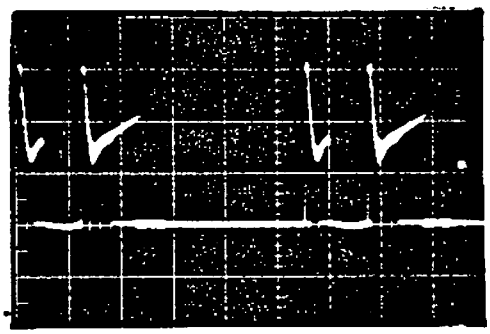
Oscilloscope
Sync - Int Coupling - DC
Sweep - Arbitrary Vert - 10 V/div

16) The power frequency control, R19, should now be adjusted to bring the power supply oscillator into sync with the horizontal oscillator. The aim of the adjustment is to make the power supply frequency somewhat lower than that of the horizontal oscillator. Place an oscilloscope probe on the collector (tab) of Q6 and monitor the signal. With the oscilloscope in sync, it will appear either as a series of overlapping waves, as illustrated in the photograph on the left below, or as a single well defined wave, as shown in the photograph on the right. Turn R19 clockwise, if necessary, to obtain the waveform shown in the photograph on the left. Then turn the control counter-clockwise until the waveform shown in the photograph on the right is obtained. Continue turning the control counter-clockwise another 30 degrees.

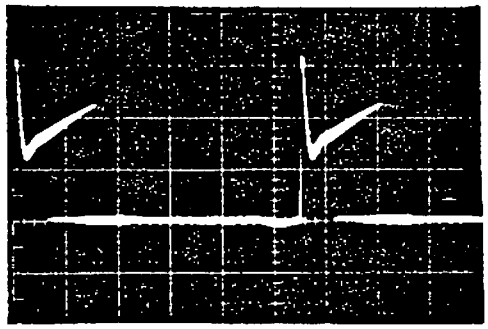
Oscilloscope

Sync - Int Sweep - 10 microsec/div
Coupling - DC Vert- 5 V/div

Oscillator out of sync



Oscillator in sync



** 17) Obtain a green raster by the following keyboard operations:

(BG ON) (CONTROL R) (HOME) (ERASE PAGE)

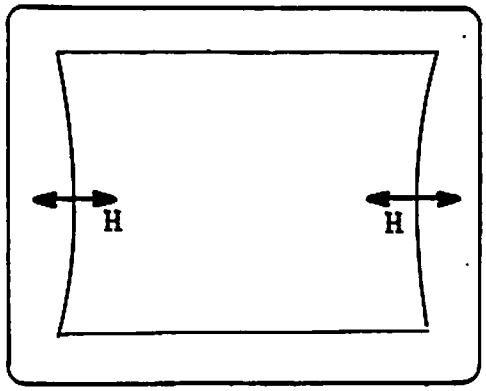
** 18) Adjust R67, the vertical size control, for a display height of 7.25 inches.

** 19) Center the display on the vertical axis by adjusting R89.

** 20) Adjust L5, the horizontal size control, for a display width of 9.5 inches.

** 21) Center the display on the horizontal axis by adjusting R1Ø6.

** 22) Adjust the pincushion control, R84,to obtain a display with straight sides. The effects of this adjustment are illustrated in the drawing below.



** 23) Obtain a white screen using the following sequence of keyboard operations:

(BG ON) (CONTROL W) (HOME) (ERASE PAGE)

Readjust R13 for exactly +5.ØØV (+.Ø5/-.ØØ), measured at pin 4 or 5, J1Ø.

** 24) Readjust the high voltage and focus controls. See steps 1Ø through 13.

** 25) Obtain a display of green horizontal lines on a black background with the following sequence of keyboard operations:

(BG ON) (CONTROL P) (FG ON) (CONTROL R) (HOME) (ERASE PAGE)

(ESC) Y _ (underline)

Adjust R68, the vertical linearity control, for equal spacing of the lines.

Erase the screen with the following keystroke sequence:

(HOME) (ERASE PAGE)

** 26) With the following sequence of commands, obtain a green diagonal line on a black background:

(CONTROL B) (COMMAND @) (COMMAND @)

(CONTROL 2) (SHIFT _) (SHIFT _) (COMMAND ?)

Adjust L4, the horizontal linearity control, for the straightest possible line. Note that the control has a cam action. The full range of adjustment is covered in a single revolution, and continued turning of the control in either direction results in the repeated traversing of the range.

Video Circuitry

** **NOTE:** The three video amplifiers are situated on the small board at the base of the CRT. Each amplifier has its own gain control. The nomenclature for the gain controls varies depending on which version of the video assembly is installed. Some units are equipped with a "one-bit" video assembly which is readily identifiable by the six-pin header connector at J2. Some units are equipped with a "two-bit/one-bit" assembly which is set up for one-bit operation (one input per amplifier); this assembly has an eight-pin connector at J2. The nomenclature of the controls is as follows:

Function	One-bit Assy	Two-bit/One-bit Assy
Red Gain	R1	R17
Green Gain	R3	R6
Blue Gain	R5	R3

Adjustment is the same for both video assemblies.

** 27) Obtain a white raster by the keyboard operations:

(BG ON) (CONTROL W) (HOME) (ERASE PAGE)

Dim the screen with the brightness control, R115. With the control set about 3/4 of full clockwise the raster should remain visible and be of about the right intensity for the gain adjustments.

Set the red gain and blue gain controls to full CW. Set the green gain control about 15 to 20 degrees from full CW. Bring red and blue down in turn from full clockwise to the points at which the screen is whitest. Red should be set so that the screen is neither too blue nor too red. Blue should be set so that it is neither too blue nor too yellow.

After setting all three gain controls, turn up the brightness. There should be no loss of whiteness.

A final check is made by obtaining a raster in each of the colors in turn. Use the keystroke sequence

(BG ON) (CONTROL P, Q, R, S, T, U, V, or W) (HOME) (ERASE PAGE)

to obtain the rasters. Each color should be pure and distinct.

** 28) If the CRT screen control was adjusted before the video gain was adjusted it should be readjusted at this point. See step 14.

Software Problem Report

Use this form to report errors or problems in Intecolor Corporation's software products. Please report only one problem per form.

Mail to:

Intecolor Corporation
Software Development
225 Technology Park/Atlanta
Norcross, Georgia 30092

Date _____

Your name and address:

_____

_____

_____

_____

Program Name _____

Program Version _____

Purchase Date _____

ISC unit serial number _____

Phone number where you can be reached during

the working day _____

Description of the problem and the circumstances surrounding its occurrence. It is best if the problem can be reduced to a simple test case. Include a diskette or write on the back of this form if you wish.

_____

_____

_____

_____

_____

_____

_____

_____

_____

(Please do not write below this line)

_____

Disposition

_____

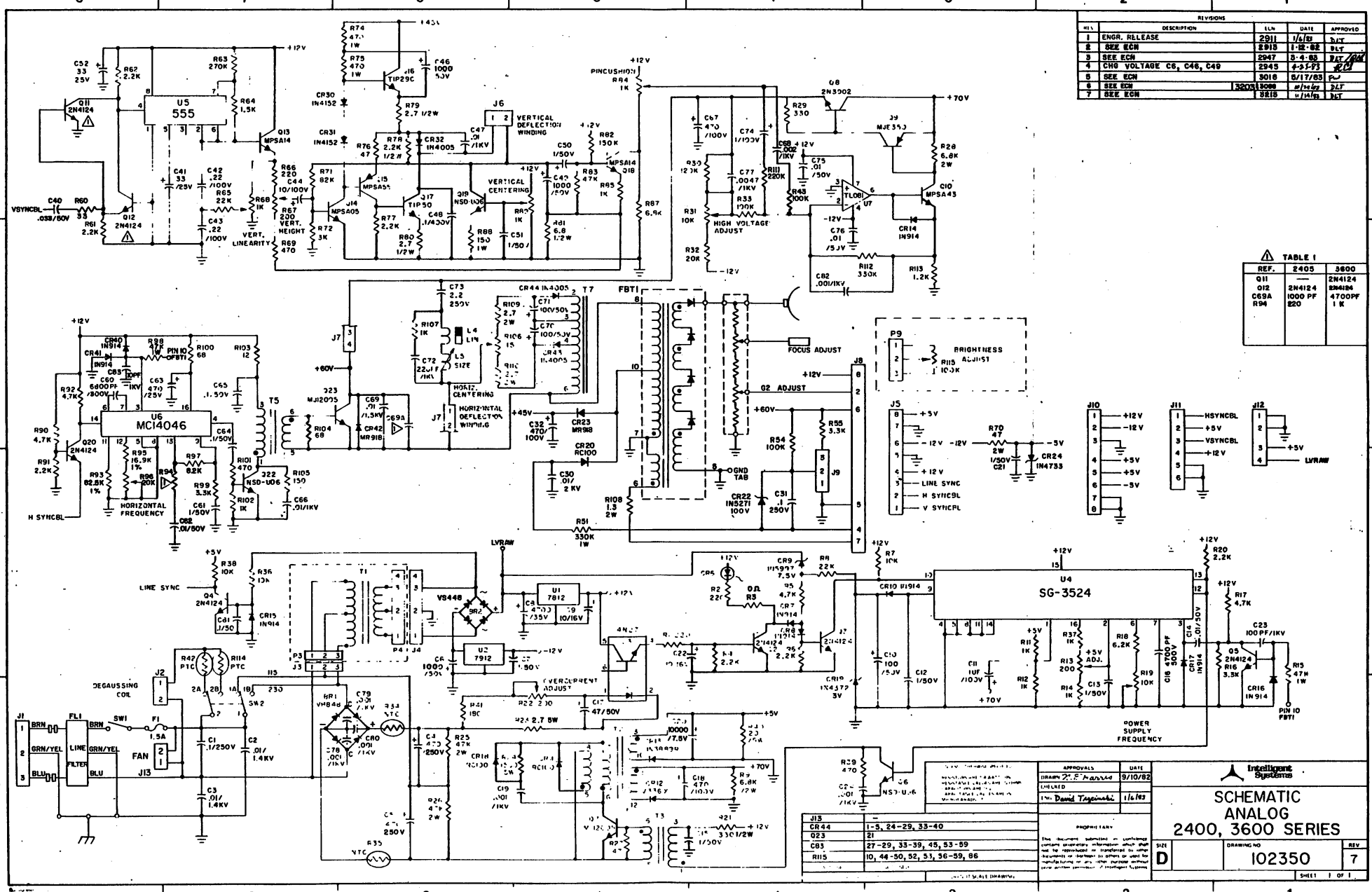
_____

_____

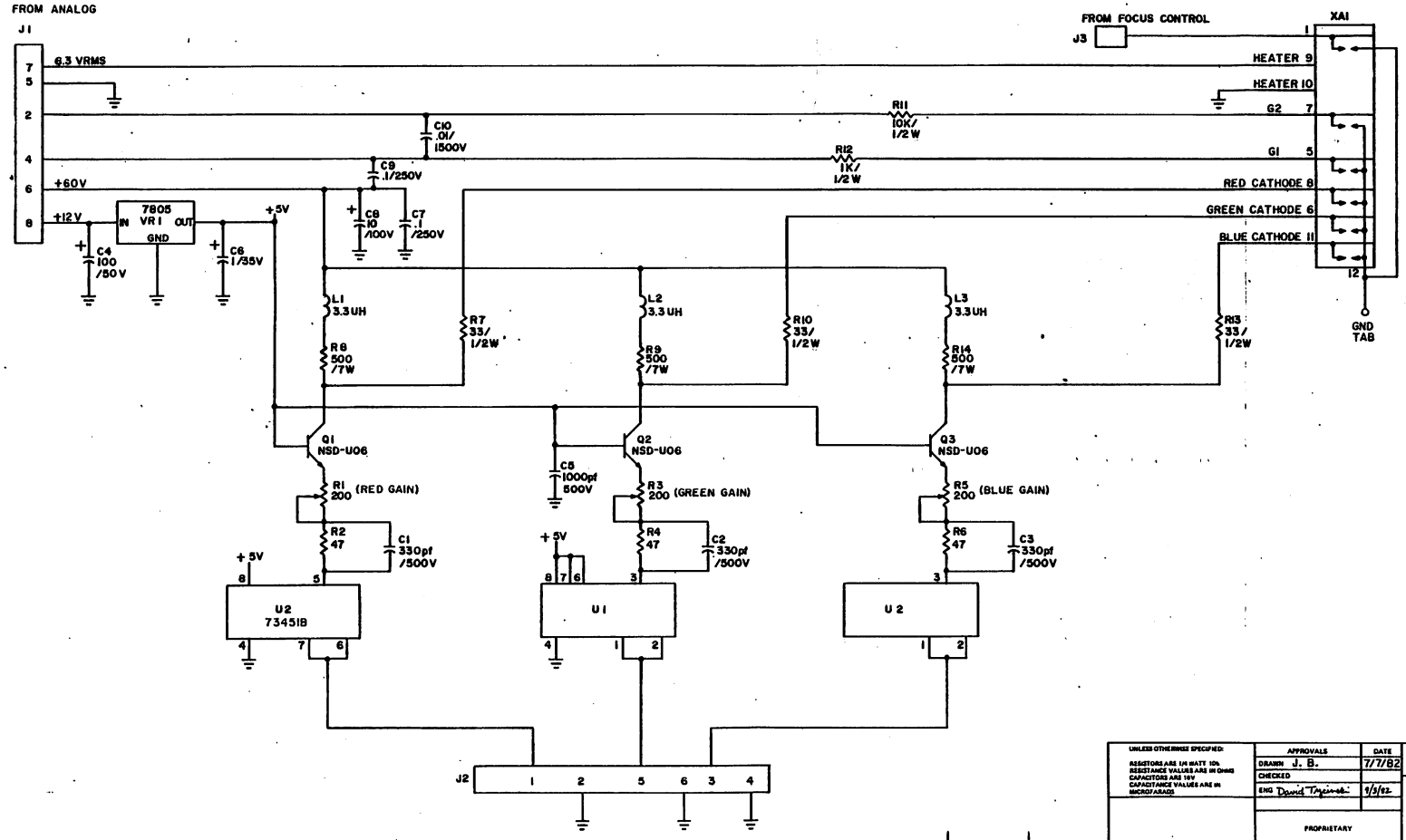
Sequence _____ Date _____

ECN _____ Programmer _____ ISC 10/83

SCHEMATIC DIAGRAMS


REVISIONS
ENGR. RELEASE
SEE ECN
SEE ECN
CHG VOLTAGE C6, C46, C49
SEE ECN
SEE ECN
SEE ECN
TABLE 1
REF. 2405 3600
Q11 — 2N4124
Q12 2N4124 2N4124
C69A 1000 PF 4700PF
R94 220 1 K
U5 555
U6 MC14046
U4 SG-3524
HORIZONTAL FREQUENCY
VERTICAL DEFLECTION WINDING
VERTICAL CENTERING
HIGH VOLTAGE ADJUST
HORIZ. CENTERING
HORIZONTAL DEFLECTION WINDING
FOCUS ADJUST
BRIGHTNESS ADJUST
G2 ADJUST
LINE SYNC
H SYNCBL
V SYNCPL
HSYNCBL
VSYNCBL
LVRAW
PIN 10 FBTI
DEGAUSSING COIL
FAN
OVERCURRENT ADJUST
POWER SUPPLY FREQUENCY
J13 —
CR44 1-5, 24-29, 33-40
Q23 21
C83 27-29, 33-39, 45, 53-59
RII5 10, 44-50, 52, 53, 56-59, 86
APPROVALS DATE
DRAWN P. E. Harris 9/10/82
CHECKED
ENG. David Tsyenishi 1/6/83
PROPRIETARY
Intelligent Systems
SCHEMATIC
ANALOG
2400, 3600 SERIES
SIZE D
DRAWING NO 102350
REV 7
SHEET 1 OF 1

REVISIONS				
REV	DESCRIPTION	ECN	DATE	APPROVED
1	ENG. RELEASE	2782	92-82	DAT



UNLESS OTHERWISE SPECIFIED:	APPROVALS	DATE
RESISTORS ARE 1/4 WATT 10%	DRAWN J. B.	7/7/82
RESISTANCE VALUES ARE IN OHMS	CHECKED	
CAPACITORS ARE 16V	ENG David Tupman	9/3/82
CAPACITANCE VALUES ARE IN MICROFARADS		

PROPRIETARY

This document, submitted in confidence, contains proprietary information which shall not be reproduced or transferred to other documents or disclosed to others or used for manufacturing or any other purpose without prior written permission of Intelligent Systems.

LAST USED | NOT USED

REFERENCE DESIGNATORS

DO NOT SCALE DRAWING

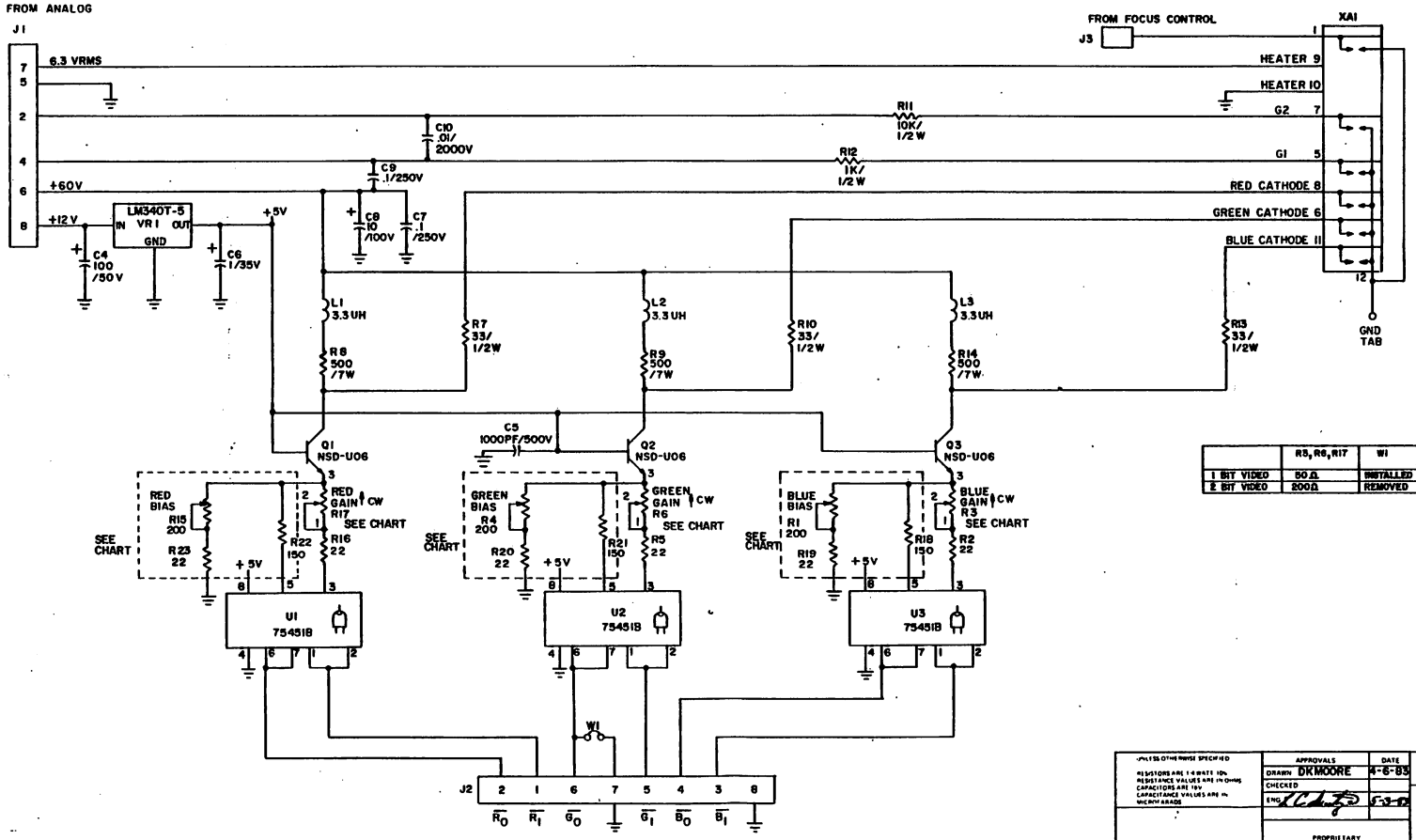
Intelligent Systems

SCHEMATIC LCCT VIDEO 2400 SERIES

SIZE	DRAWING NO.	REV
D	101976	1

SHEET 1 OF 1

REVISIONS				
REV	DESCRIPTION	ECN	DATE	APPROVED
1	ENG. RELEASE	3004		



	R5, R6, R17	W1	CIRCUIT IN OUTLINE
1 BIT VIDEO	50Ω	INSTALLED	REMOVED
2 BIT VIDEO	200Ω	REMOVED	INSTALLED

UNLESS OTHERWISE SPECIFIED
RESISTORS ARE 1/4 WATT 10%
RESISTANCE VALUES ARE IN OHMS
CAPACITORS ARE 10V
CAPACITANCE VALUES ARE IN MICRO FARADS

APPROVALS		DATE
DRAWN	DR MOORE	4-6-83
CHECKED		
ENG		

PROPRIETARY

This document, submitted in confidence, contains proprietary information which shall not be reproduced or transferred to other documents or disclosed to others or used for manufacturing or any other purpose without prior written permission of Intelligent Systems

Intelligent Systems

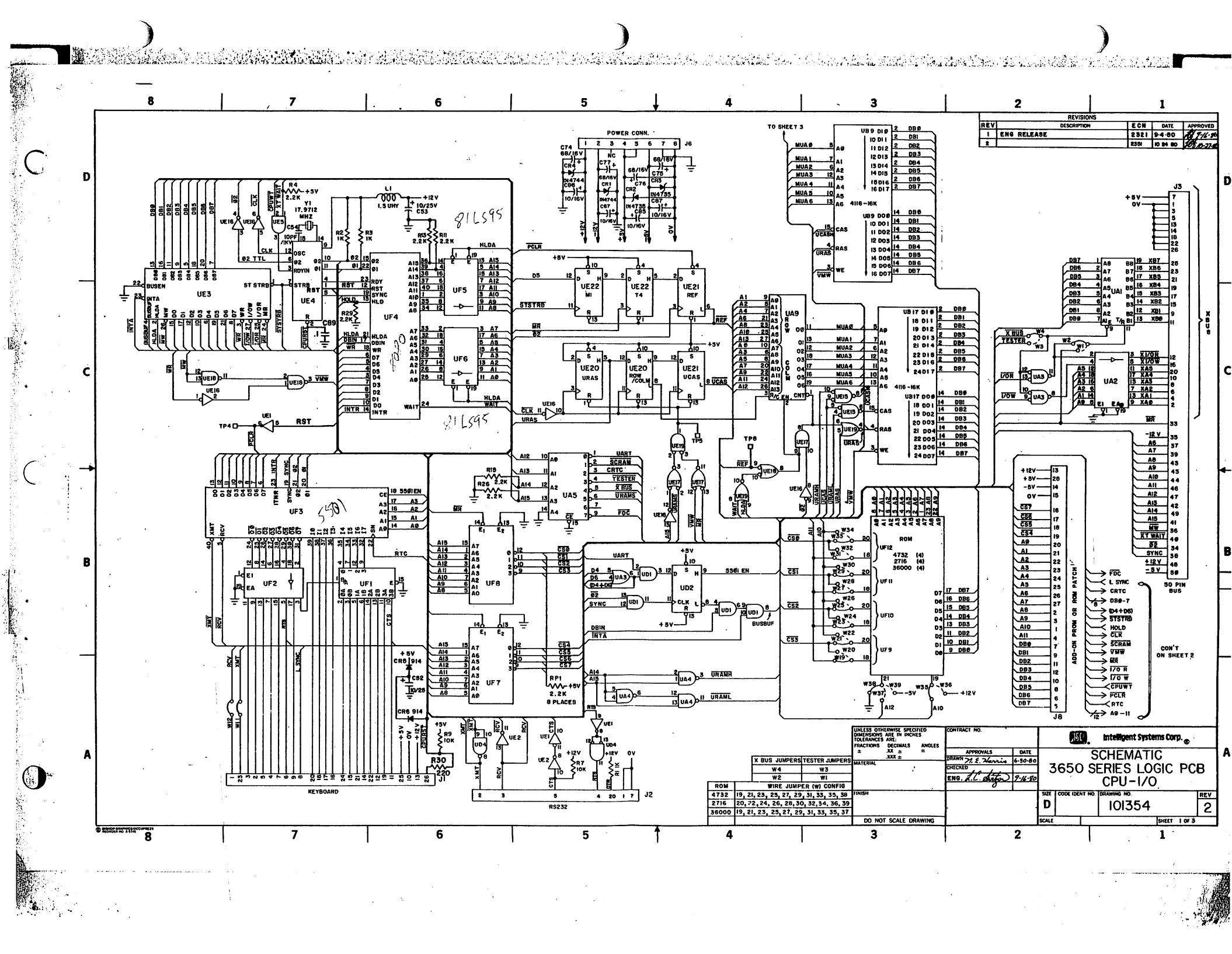
SCHEMATIC 2/1 BIT VIDEO ILVA

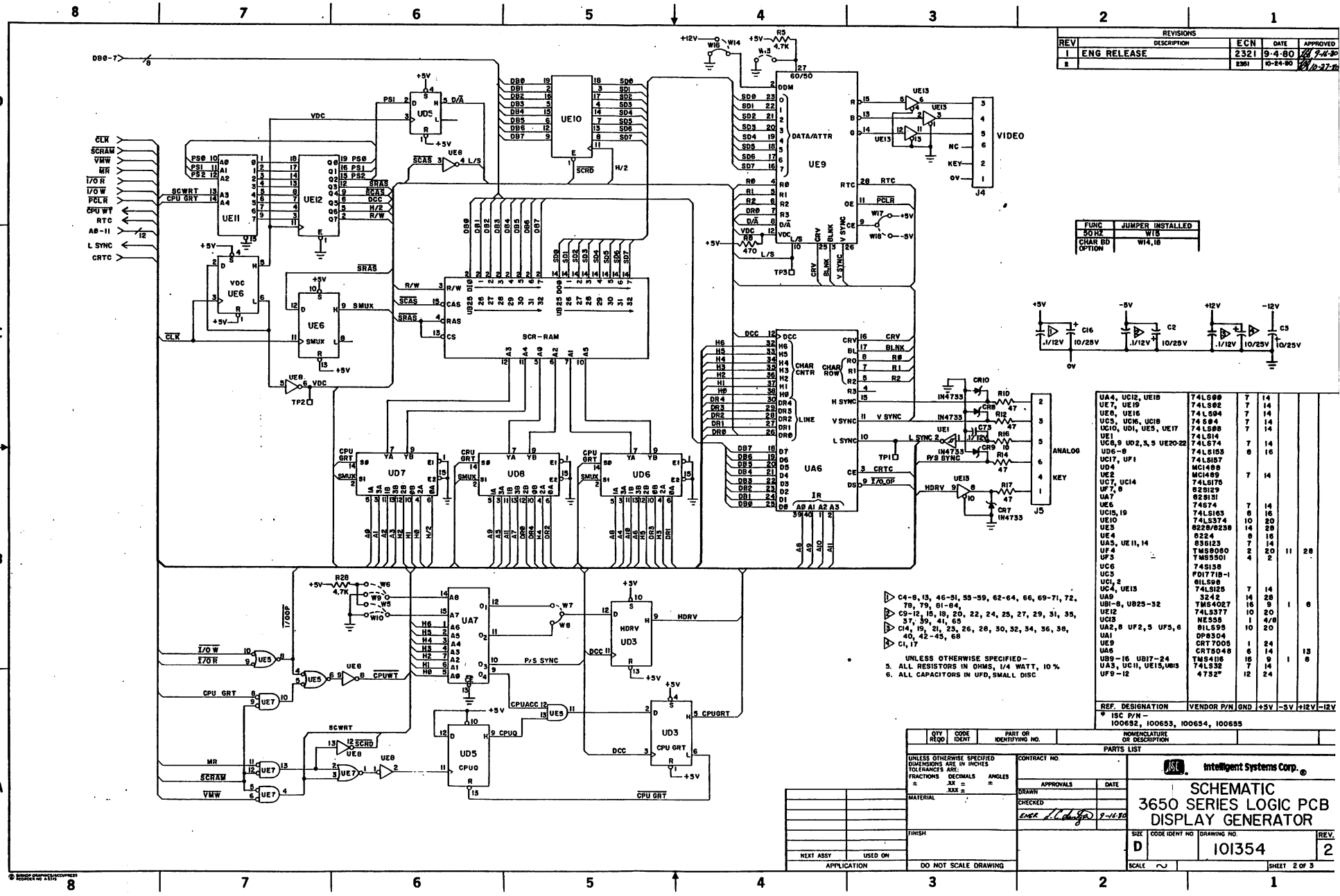
SIZE	DRAWING NO	REV
D	102700	1

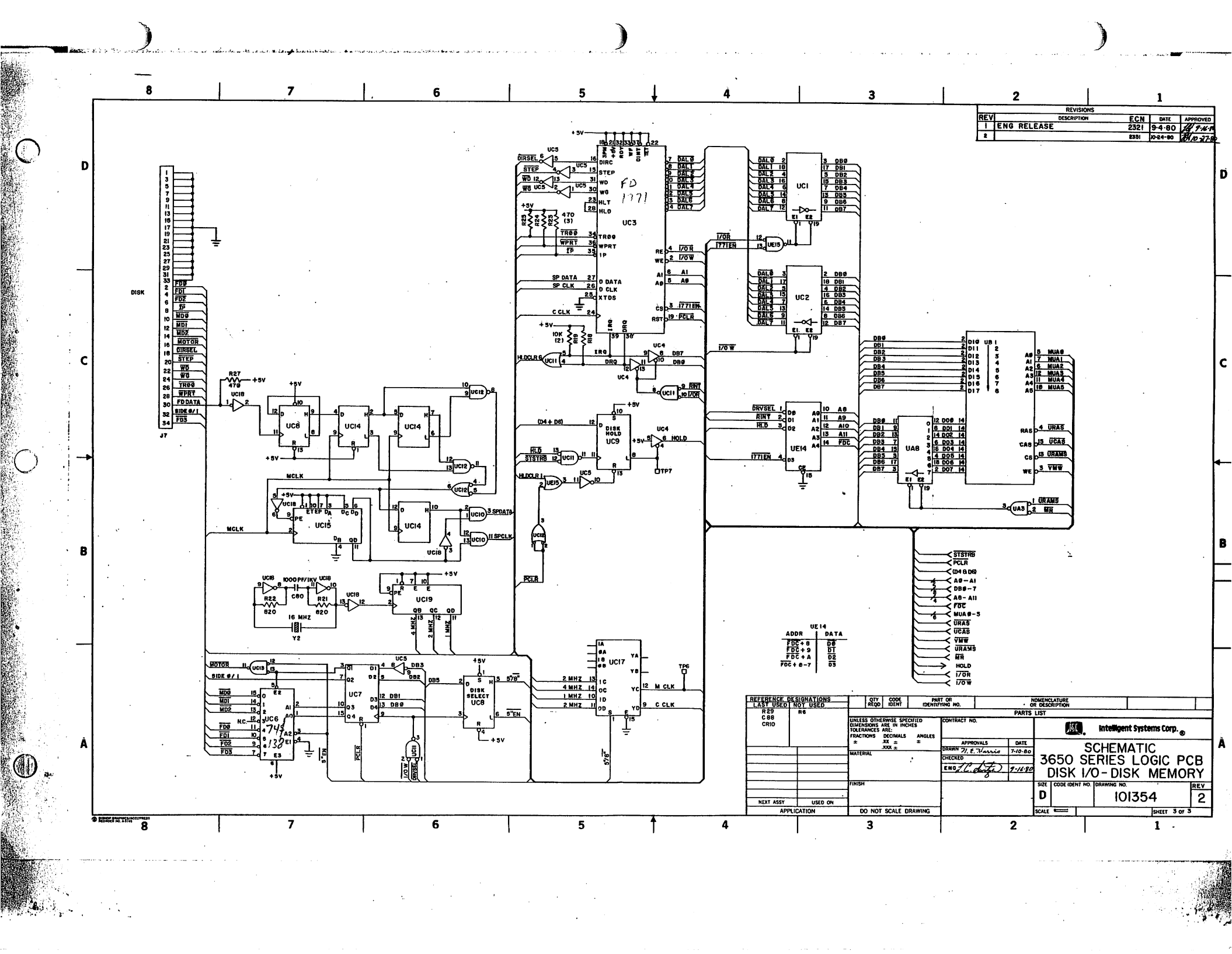
SHEET 1 OF 1

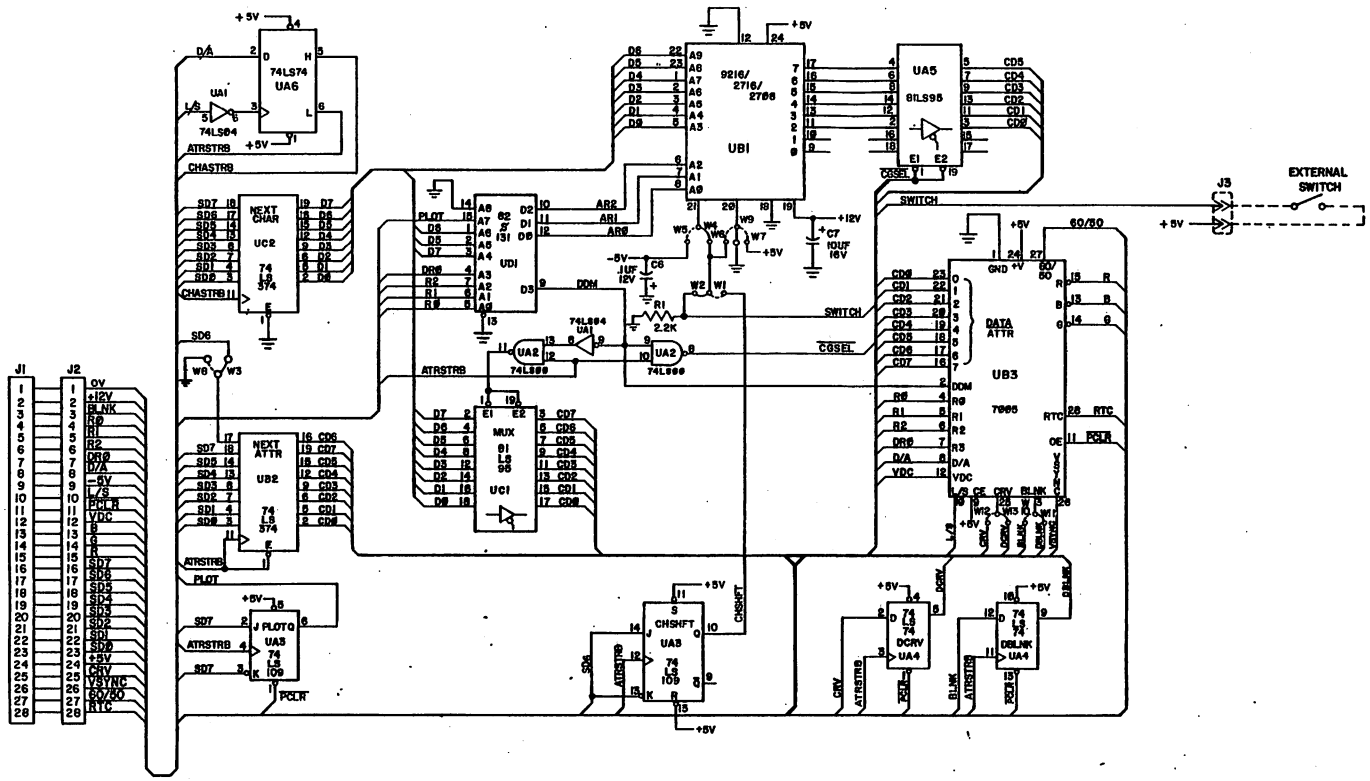
LAST USED	NOT USED
REFERENCE DESIGNATORS	

DO NOT SCALE DRAWING


REVISIONS
REV | DESCRIPTION | ECN | DATE | APPROVED
1 | ENG RELEASE | 2321 | 9-4-80
2 | | 2334 | 10 24 80
POWER CONN.
TO SHEET 3
UE3
UE4
UF4
UF5
UF6
UF3
UF2
UF1
UF8
UF7
UA5
UA9
UE22
UE21
UE20
UA3
UA4
UD2
UD1
UF12
UF11
UF10
UF9
UA1
UA2
ROM
4732 (4)
2716 (4)
36000 (4)
4116-16K
50 PIN BUS
CON'T ON SHEET 2
ADD-ON PROM OR ROM PATCH
KEYBOARD
RS232
RP1
2.2K
8 PLACES
X BUS JUMPERS | TESTER JUMPERS
W4 | W3
W2 | W1
ROM | WIRE JUMPER (W) CONFIG
4732 | 19, 21, 23, 25, 27, 29, 31, 33, 35, 38
2716 | 20, 22, 24, 26, 28, 30, 32, 34, 36, 39
36000 | 19, 21, 23, 25, 27, 29, 31, 33, 35, 37
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±
MATERIAL
FINISH
DO NOT SCALE DRAWING
CONTRACT NO.
APPROVALS | DATE
DRAWN T. E. Harris 6-30-80
CHECKED
ENG. R.C. Ortiz 9-16-80
Intelligent Systems Corp.
SCHEMATIC
3650 SERIES LOGIC PCB
CPU-I/O
SIZE D | CODE IDENT NO. | DRAWING NO. 101354 | REV 2
SCALE | SHEET 1 OF 3


REVISIONS
REV | DESCRIPTION | ECN | DATE | APPROVED
1 | ENG RELEASE | 2321 | 9-4-80 | 9-4-80
2 | | 2351 | 10-24-80 | 10-27-80
FUNC | JUMPER INSTALLED
50HZ | W15
CHAR BD OPTION | W14, 16
REF. DESIGNATION | VENDOR P/N | GND | +5V | -5V | +12V | -12V
* ISC P/N - 100652, 100653, 100654, 100655
PARTS LIST
QTY REQD | CODE IDENT | PART OR IDENTIFYING NO. | NOMENCLATURE OR DESCRIPTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES ± .XX ± .XXX ±
MATERIAL
FINISH
NEXT ASSY | USED ON
APPLICATION
DO NOT SCALE DRAWING
CONTRACT NO.
APPROVALS | DATE
DRAWN
CHECKED
ENGR
Intelligent Systems Corp.
SCHEMATIC
3650 SERIES LOGIC PCB
DISPLAY GENERATOR
SIZE D | CODE IDENT NO | DRAWING NO. 101354 | REV. 2
SCALE | SHEET 2 OF 3
DB0-7
CLK
SCRAM
VMW
MR
I/O R
I/O W
PCLR
CPU WT
RTC
A0-11
L SYNC
CRTC
SCWRT
CPU GRT
PS0
PS1
PS2
VDC
UD5
UE10
UE11
UE12
UE6
UE9
UA6
UA7
UD3
UD6
UD7
UD8
SCR-RAM
CHAR CNTR
CHAR ROW
LINE
DATA/ATTR
VIDEO
J4
J5
ANALOG
KEY
P/S SYNC
HDRV
CPUACC
CPUQ
CPUGRT
CPUWT
SCRD
SRAS
SCAS
SMUX
H/2
R/W
CRV
BLNK
V SYNC
H SYNC
CRTC
I/OOP
IR
TP1
TP2
TP3
W5
W6
W7
W8
W9
W10
W13
W14
W15
W16
W17
W18
R5 4.7K
R10 47
R12 47
R14 47
R16 47
R17 47
R28 4.7K
R8 470
C16 .1/12V 10/25V
C2 .1/12V 10/25V
C3 .1/12V 10/25V
C73 .1/12V
CR7 IN4733
CR8 IN4733
CR9 IN4733
CR10 IN4733
+5V
+12V
-5V
-12V
0V
C4-8, 13, 46-51, 55-59, 62-64, 66, 69-71, 72, 76, 79, 81-84,
C9-12, 15, 18, 20, 22, 24, 25, 27, 29, 31, 35, 37, 39, 41, 65
C14, 19, 21, 23, 26, 28, 30, 32, 34, 36, 38, 40, 42-45, 68
C1, 17
UNLESS OTHERWISE SPECIFIED-
5. ALL RESISTORS IN OHMS, 1/4 WATT, 10%
6. ALL CAPACITORS IN UFD, SMALL DISC
UA4, UC12, UE18 | 74LS00 | 7 | 14
UE7, UE19 | 74LS02 | 7 | 14
UE8, UE16 | 74LS04 | 7 | 14
UC5, UC16, UC18 | 74S04 | 7 | 14
UC10, UD1, UE5, UE17 | 74LS08 | 7 | 14
UE1 | 74LS14
UC8,9 UD2,3,5 UE20-22 | 74LS74 | 7 | 14
UD6-8 | 74LS153 | 8 | 16
UC17, UF1 | 74LS157
UD4 | MCI4006
UE2 | MCI4013 | 7 | 14
UC7, UC14 | 74LS175
UF7, 8 | 823129
UA7 | 6281S1
UE6 | 74S74 | 7 | 14
UC15, 19 | 74LS163 | 8 | 16
UE10 | 74LS374 | 10 | 20
UE3 | 6228/8238 | 14 | 28
UE4 | 8224 | 8 | 16
UA5, UE11, 14 | 836123 | 7 | 14
UF4 | TMS8080 | 2 | 20 | 11 | 28
UF5 | TMS3501 | 4 | 2
UC6 | 74S136
UC5 | FDI771B-1
UC1, 2 | 6IL598
UC4, UE13 | 74LS125 | 7 | 14
UA9 | 3242 | 14 | 28
UB1-8, UB25-32 | TMS4027 | 16 | 9 | 1 | 8
UE12 | 74LS377 | 10 | 20
UC13 | NE555 | 1 | 4/8
UA2,8 UF2,5 UF5,6 | 8ILS95 | 10 | 20
UA1 | DP8304
UE9 | CRT7005 | 1 | 24
UA6 | CRT5048 | 6 | 14 | 13
UB9-16 UB17-24 | TMS4116 | 16 | 9 | 1 | 8
UA3, UC11, UE15, UB13 | 74LS32 | 7 | 14
UF9-12 | 4732* | 12 | 24


REVISIONS
REV | DESCRIPTION | ECN | DATE | APPROVED
1 | ENG RELEASE | 2321 | 9-4-80 | 9-14-80
2 | | 2351 | 10-24-80 | 10-27-80
DISK
J7
UC1
UC2
UC3
UC4
UC5
UC6
UC7
UC8
UC9
UC10
UC11
UC12
UC13
UC14
UC15
UC16
UC17
UC18
UC19
UE14
UE15
UA8
UA9
FD 1771
MCLK
SPDATA
SPCLK
PCLR
HLDCLR
HLD
STSTRB
I/OR
I/OW
1771EN
DRVSEL
RINT
HOLD
M CLK
C CLK
5"EN
5'8"
TP6
TP7
Y2
16 MHZ
1000PF/1KV
R22 620
R21 620
C80
R27 470
R23 470 (3)
10K (2)
UE14
ADDR | DATA
FDC+8 | D0
FDC+9 | D1
FDC+A | D2
FDC+B-7 | D3
REFERENCE DESIGNATIONS LAST USED R29 C88 CR10 | NOT USED R6
PARTS LIST
QTY REQD | CODE IDENT | PART OR IDENTIFYING NO. | NOMENCLATURE OR DESCRIPTION
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±
MATERIAL
FINISH
NEXT ASSY | USED ON
APPLICATION
DO NOT SCALE DRAWING
CONTRACT NO.
APPROVALS | DATE
DRAWN T. E. Harris 7-10-80
CHECKED
ENG T. C. Hartje 9-14-80
Intelligent Systems Corp.
SCHEMATIC
3650 SERIES LOGIC PCB
DISK I/O - DISK MEMORY
SIZE D | CODE IDENT NO. | DRAWING NO. 101354 | REV 2
SCALE | SHEET 3 OF 3



TABLE

DEVICE	JUMPER
UB1 9216	W4, W9
UB1 2716	W5, W6
UB1 2708	W5, W9
UB1 9216P	W4, W7

DEVICE	JUMPER
HARDWARE SWITCH	W2, W3
SOFTWARE SWITCH	W1, W8

REVISIONS

REV	ZONE	LTR	DESCRIPTION	ECN	DATE	APPROVED
1			ENG. RELEASE	2416	1-6-81	
2			CORRECT JUMPER TRACES	2430	1-23-81	

PARTS LIST

QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN TAYLOR	1/7/80
CHECKED ENGR. T.C. Chang	1-6-81

FOR REV 1 & 2 PCB

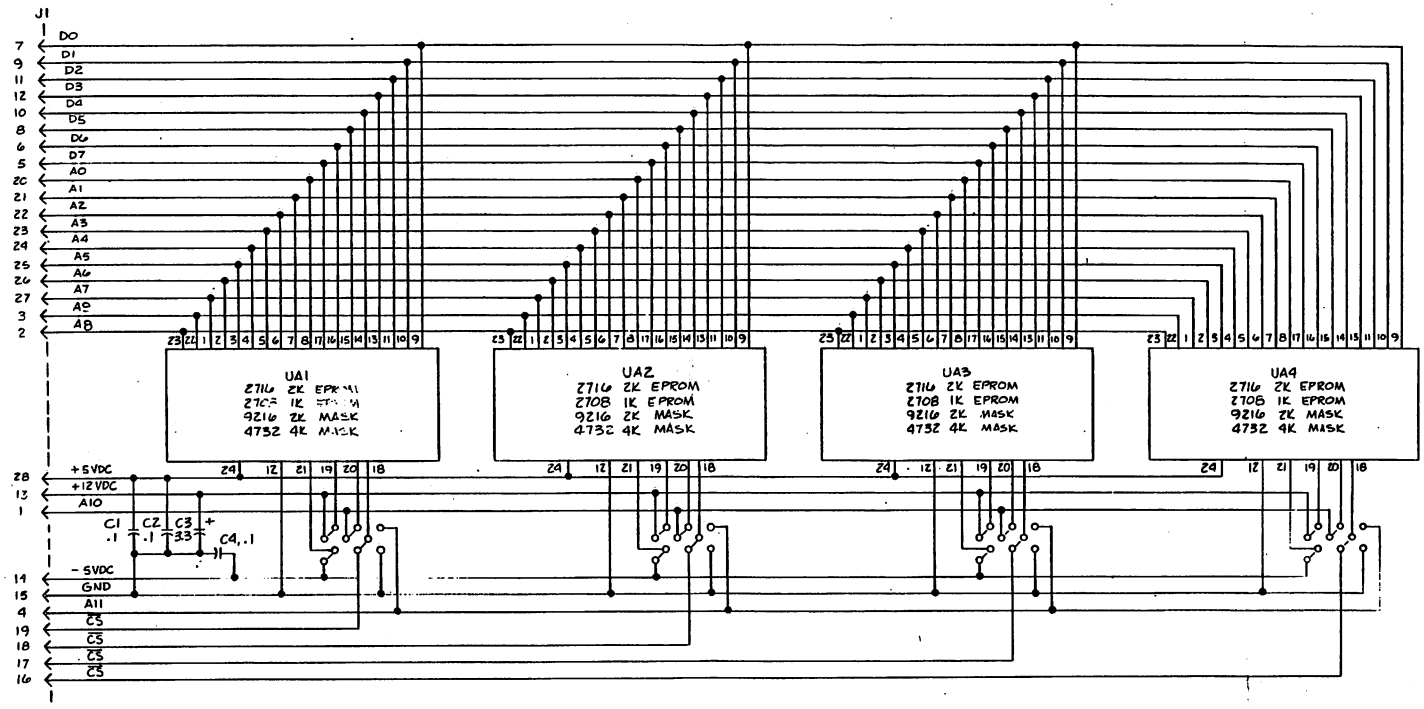
Intelligent Systems Corp.

SCHEMATIC
OPTIONAL CHARACTER GENERATOR

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D		101410	2

SCALE NONE SHEET 1 OF 1

REVISIONS					
ZONE	LTR	DESCRIPTION	ECN	DATE	APPROVED
1		ENGINEERING RELEASE	1224	3/22/82	



QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION
			PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± | DECIMALS .XX ± .XXX ± | ANGLES ±

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

CONTRACT NO

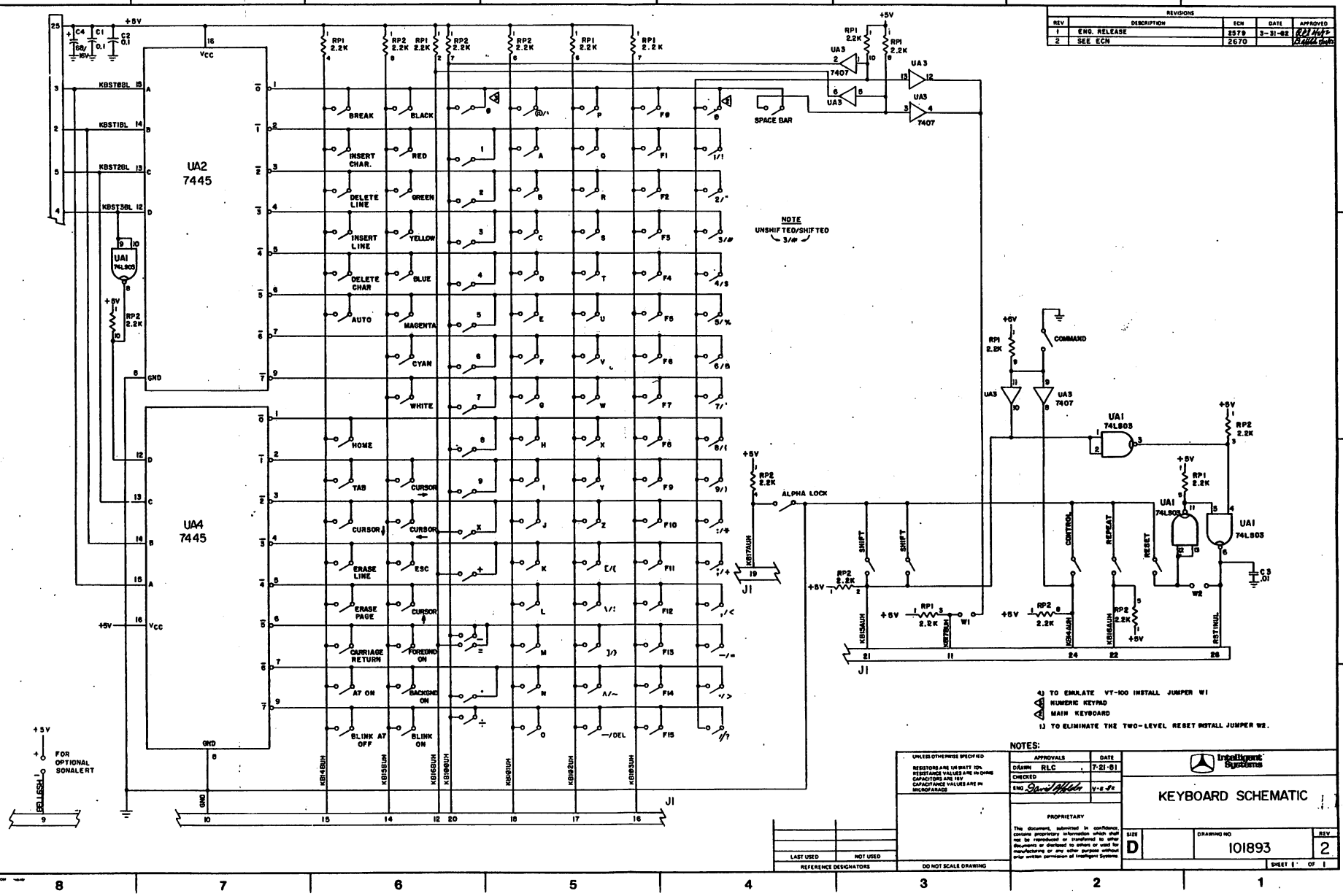
APPROVALS	DATE
DRAWN	8-26-80
CHECKED	9-12-80
ENGR:	1-22-80

Intelligent Systems Corp.

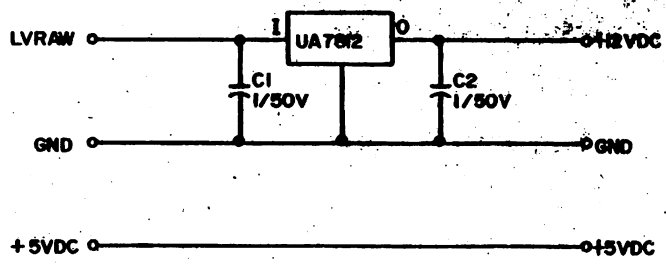
COMPUCOLOR II ADD ON PROM SCHEMATIC DIAGRAM

SIZE D | CODE IDENT NO. | DRAWING NO. 100979 | REV 1

SCALE | SHEET 1 OF 1


UA2 7445
UA4 7445
UA1 74LS03
UA3 7407
BREAK
INSERT CHAR.
DELETE LINE
INSERT LINE
DELETE CHAR
AUTO
HOME
TAB
CURSOR →
CURSOR ↓
ERASE LINE
ERASE PAGE
CARRIAGE RETURN
A7 ON
BLINK A7 OFF
BLACK
RED
GREEN
YELLOW
BLUE
MAGENTA
CYAN
WHITE
CURSOR ←
CURSOR ↑
ESC
FOREGND ON
BACKGND ON
BLINK ON
SPACE BAR
ALPHA LOCK
SHIFT
CONTROL
REPEAT
RESET
COMMAND
FOR OPTIONAL SONALERT
NOTE
UNSHIFTED/SHIFTED
J1
W1
W2
NOTES:
1) TO EMULATE VT-100 INSTALL JUMPER W1
NUMERIC KEYPAD
MAIN KEYBOARD
1) TO ELIMINATE THE TWO-LEVEL RESET INSTALL JUMPER W2.
PROPRIETARY
KEYBOARD SCHEMATIC
DRAWING NO 101893
SIZE D
REV 2
SHEET 1 OF 1

REVISIONS				
REV	DESCRIPTION	ECN	DATE	APPROVED
1	ENG. RELEASE	2947	2-25-83	DLT



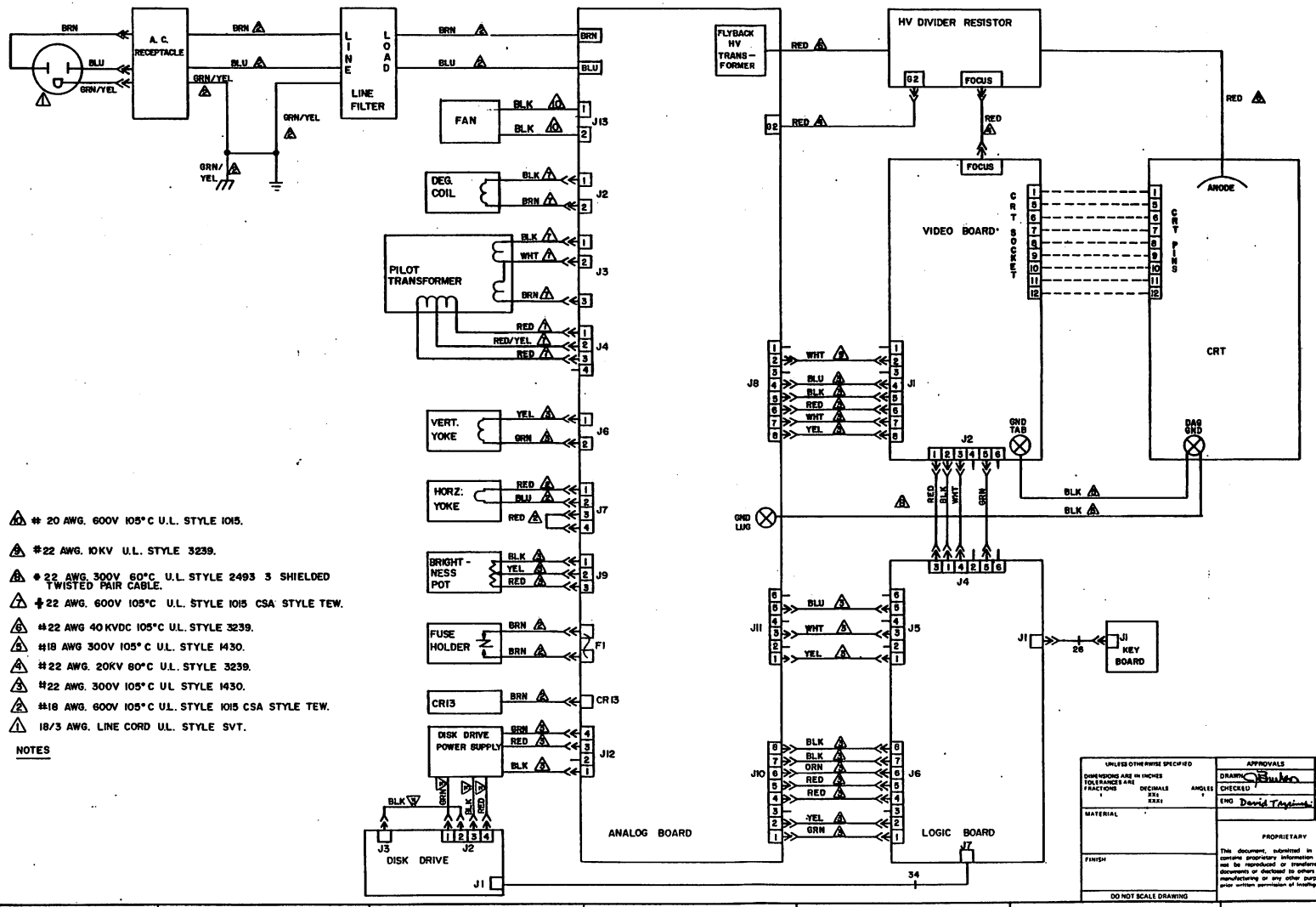
UNLESS OTHERWISE SPECIFIED	APPROVALS	DATE
DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS ± DECIMALS .XX± .XXX± ANGLES ±	DI.AWG Ruth Smith	10/27/82
	CHECKED	
	I.NG David Trzpinski	1/13/83
MATERIAL		
FINISH	PROPRIETARY This document, submitted in confidence, contains proprietary information which shall not be reproduced or transferred to other documents or disclosed to others or used for manufacturing or any other purpose without prior written permission of Intelligent Systems.	
DO NOT SCALE DRAWING		

Intelligent Systems

SCHEMATIC
DISK DRIVE POWER SUPPLY

SIZE	DRAWING NO.	REV
B	102362	1

SCALE NONE SHEET 1 OF 1



⚠ # 20 AWG. 600V 105°C U.L. STYLE 1015.

⚠ #22 AWG. 10KV U.L. STYLE 3239.

⚠ # 22 AWG. 300V 60°C U.L. STYLE 2493 3 SHIELDED TWISTED PAIR CABLE.

⚠ # 22 AWG. 600V 105°C U.L. STYLE 1015 CSA STYLE TEW.

⚠ #22 AWG 40 KVDC 105°C U.L. STYLE 3239.

⚠ #18 AWG 300V 105°C U.L. STYLE 1430.

⚠ #22 AWG. 20KV 60°C U.L. STYLE 3239.

⚠ #22 AWG. 300V 105°C U.L. STYLE 1430.

⚠ #18 AWG. 600V 105°C U.L. STYLE 1015 CSA STYLE TEW.

⚠ 18/3 AWG. LINE CORD U.L. STYLE SVT.

NOTES

REVISIONS				
REV	DESCRIPTION	ECN	DATE	APPROVED
1	ENG. RELEASE	2947	2-25-85	DLT

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES ARE
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±
MATERIAL
FINISH
DO NOT SCALE DRAWING

APPROVALS	DATE
DRAWN	2/14/85
CHECKED	
ENG David Tayeimi	1/13/87

PROPRIETARY

This document, submitted in confidence, contains proprietary information which shall not be reproduced or transferred to other documents or disclosed to others or used for manufacturing or any other purpose without prior written permission of Intelligent Systems.

Intelligent Systems

WIRING DIAGRAM 3651 SERIES

SIZE D | DRAWING NO 102428 | REV 1 | SHEET | OF