AN INTERFACE FOR A REAL-TIME STORAGE OSCILLOSCOPE DISPLAY

by

Robert Arnold Berntson

MIT CSR T-69-3 June, 1969

CENTER FOR SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

AN INTERFACE FOR A REAL-TIME STORAGE OSCILLOSCOPE DISPLAY

by

Robert Arnold Berntson

MIT CSR T-69-3 June, 1969

AN INTERFACE FOR A REAL-TIME

STORAGE OSCILLOSCOPE DISPLAY

by

ROBERT ARNOLD BERNTSON

Submitted to the Department of Electrical Engineering on May 23, 1969, in partial fulfillment of the requirements for the degree of Bachelor of Science.

ABSTRACT

A small general purpose digital computer (HP-2115A) is to be used in a laboratory environment for the monitoring, calibration, and checkout of a satellite experiment package. Test results are to be presented via printout and graphics (Calcomp Model 565 plotter).

The existing real-time graphics system consists of a small general-purpose computer and an incremental X-Y plotter, with an appropriate software and hardware interface. A new interface has been designed to provide for the addition of a storage oscilloscope display to this system.

It is shown that scope writing using unresolved dots is operationally equivalent to plotter writing using incremental steps. Functional graphical equivalence of the two devices is then established through the use of adapted plotter software for the operation of either device.

THESIS SUPERVISOR: Suhas S. Patil
TITLE: Instructor of Electrical Engineering

ACKNOWLEDGMENT

The thesis work was done at the Center for Space Research at M.I.T. and the people there have generously given me valuable assistance. Dr. E. F. Lyon suggested the thesis project, helped me throughout the course of the work, and gave much advice that I recognized too late as wise. J. Binsack and S. Patil provided ideas during the work and much useful criticism in the preparation of the final draft of the text.

This work was supported in part by NASA grant NGL 22-009-019 and NASA contract NAS 5-11062.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

The use of a small general-purpose digital computer as a participating analytical component in a laboratory environment is becoming increasingly popular and productive. With appropriate interfaces, consisting of input/output hardware and software, the computer can rapidly and accurately monitor data from a wide variety of instruments or generate digitally coded signals for input to other information-handling devices or test instruments. In most cases, the computer accepts data from external sources, processes them, and conveys the results in coded form to other devices which frequently display the information for inspection by a human user.

This thesis will deal with the graphical display capabilities of such a real-time system for preflight testing of a satellite-borne plasma instrument. All graphical display is presently done using an X-Y drum plotter which is interfaced with the computer using standard software and hardware. (See Figure 1.)

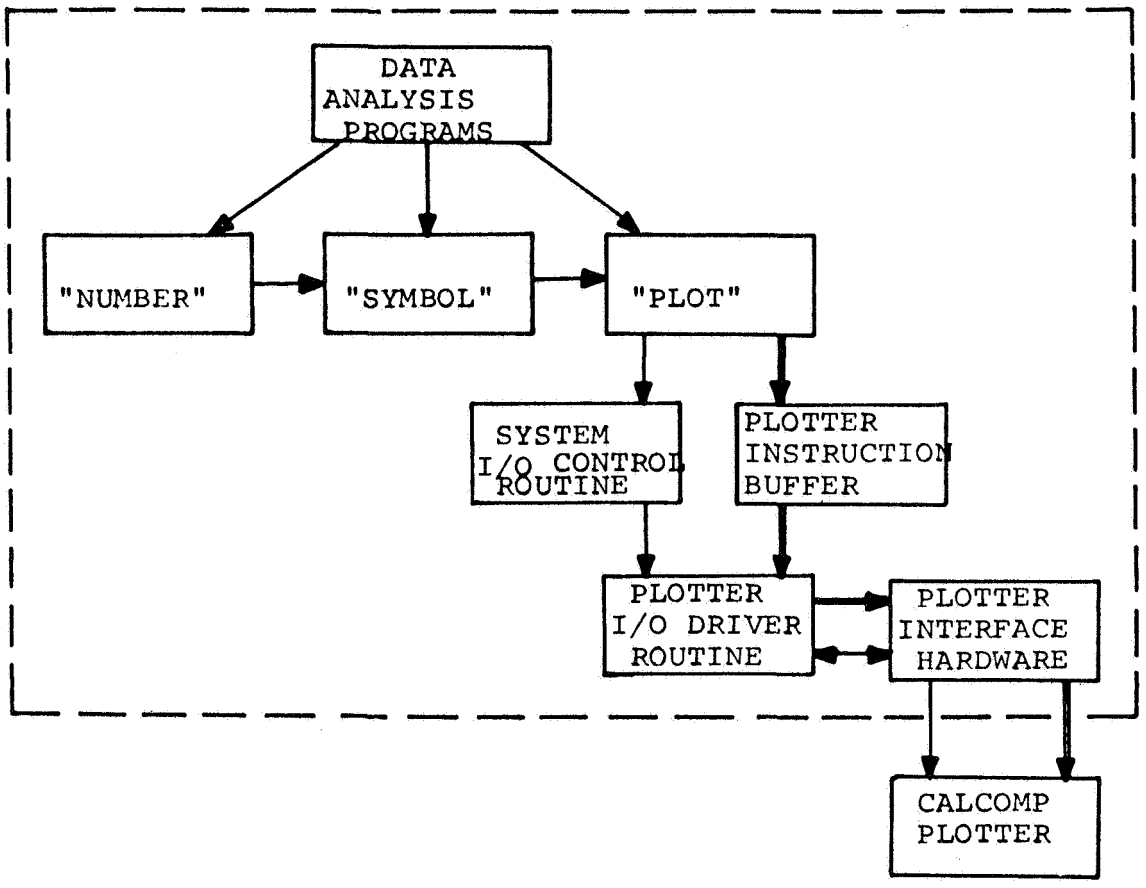


Figure 1.

Black diagram of the existing graphics system. The dotted line represents the boundary of the computer. The rectangles represent either software or hardware modules. NUMBER, SYMBOL, and PLOT are standard Calcomp graphics routines. The plotter instruction buffer is a contiguous area in core memory. Single arrows represent subroutine calls or transfer of control. Double arrows indicate flow of plotter commands. The system is described in more detail in Chapter 3.

To gain considerably greater display speed and flexibility at reasonable cost, the users of this system have decided to add a storage oscilloscope ("scope") to the system.

An interface for the scope will be proposed in this thesis. The overall objectives of the interface design are: (1) to provide the same graphical functions as the plotter does presently, (2) to ease implementation by minimizing modifications to the existing system and making additions conceptually simple, (3) to keep core-storage requirements low, and (4) to take definite advantage of the speed of the scope relative to that of plotter.

Towards these ends, the thesis work was divided into several parts. Chapter 2 contains a comparison of the operating characteristics of the two graphical display devices. In Chapter 3, the operation of the plotter interface is described and relevant features of the computer output system are presented. The objectives of the new interface and their practical consequences are made specific in Chapter 4. Chapter 5 contains the actual interface proposal, followed by conclusions and

comments in the last chapter. The appendices contain details of the existing and proposed systems which will be useful to a programmer involved in implementing the proposed system.

A Hewlett-Packard 2115A general-purpose computer (3) is used in these systems. For brevity and a certain amount of generality, details of its operation are included in this thesis only where necessary.

Chapter 2

COMPARISON OF PLOTTER AND STORAGE OSCILLOSCOPE CHARACTERISTICS

The Plotter

The Calcomp plotter (1) is an electromechanical device consisting principally of input electronics, incremental motors, drive system controls, a pen, and a drum over which runs a roll of paper. The pen can be moved across the drum in the "Y-direction", while the drum can be rotated to effectively create pen motion (relative to the paper) in the "X-direction". The Y-axis is limited to 10 inches by the drum length (i.e. the "height" of the drum cylinder). The X-axis can be as long as the roll of paper used. The pen may be raised from the drum or lowered to the drum surface under electronic control.

There are ten specific plotter movements. Two of them are raising and lowering the pen. Two more are 0.01-in. movements of the pen in either direction along the Y-axis. The drum rotates incrementally, its two

specific movements resulting in 0.01-in. pen displacements in either direction along the X-axis. Simultaneous pen and drum movements will produce four other possible vector movements, along the diagonal in each quadrant. Lowering the pen requires 60 ms for completion. The other nine movements require 3.3 ms. Each plotter movement is initiated by a unique six-bit digital input or command to the plotter electronics. More details of plotter system operation will be found in Chapter 3.

The storage Oscilloscope

The Tektronix 611 Storage Display Unit (2) is a device in which an electron beam excites a phosphor-coated screen, producing a visible spot called the trace. Under certain conditions an electronic process maintains a visible dot on the screen after the trace has moved elsewhere. This is the storage capability of the device. Since the scope is an electronic device, it is much faster than the plotter and comparable to the computer in speed.

The scope requires analog voltages as inputs to horizontal and vertical deflection circuits. These inputs are called the X-axis and Y-axis inputs, respectively. The scope also has a Z-axis input which makes the trace visible (unblanked) or invisible (blanked). If this input is held above a certain voltage (to unblank the trace) and the X-axis and Y-axis inputs remain at the same voltage for at least 20μs, a visible dot is maintained (stored) at the location of the trace. This is how the scope "writes".

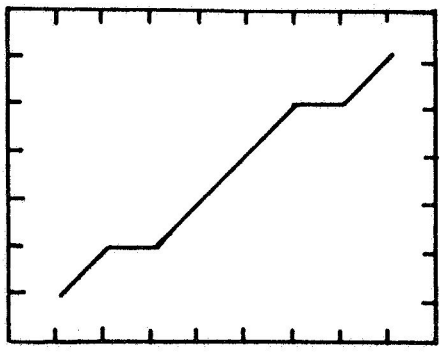
Four other digital scope inputs control special features of scope operation. The view input intensifies stored information on the scope screen. The non-store input allows the scope to function as a normal oscilloscope, without storage capability. The erase input deletes all previously stored information from the screen. The write-through feature allows the trace to be visible without storing any new information or affecting stored information.

Line-Drawing Techniques

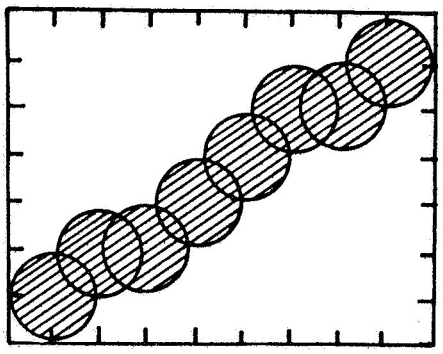
The first two objectives of the interface are to create functional equivalence between scope and plotter

graphics and to simplify changes to the existing system. A first step towards realizing these objectives would be to make the line-drawing technique of the scope analogous to that of the plotter. On the plotter, sequences of small straight line segments can be used to approximate any desired line. On the scope, because of the finite size of a stored dot, it is possible to form a line from a series of unresolved dots. These two techniques are illustrated in Figure 2.

The increment of pen or drum movement by the plotter is 0.01 in. For the scope, the manufacturer claims resolution of up to 50 line pairs per inch, i.e. two lines (or dots) must be 0.02 in. apart to be separate and distinct. If the increment of horizontal and vertical trace movement on the scope is made 0.01 in., then two adjacent (along an axis or diagonally) dots will be unresolved. The scope display size is 8.2 in. (vertical) and 6.4 in. (horizontal). If digital-to-analog (D/A) converters provide the X-axis and Y-axis voltages, then the D/A inputs should be ten bits to obtain the desired increment size.



(a)



(b)

Figure 2.

Line-drawing techniques on the plotter and the scope. Diagram (a) is a greatly enlarged illustration of an approximation to a straight line drawn by the existing system using incremental plotter commands. Diagram (b) shows an approximation to the same straight line drawn by a new system using unresolved dots on a storage scope screen. The rectangular boundaries are marked in units of 0.01 in.

Speed

The plotter can execute a maximum of 300 commands per second. It moves incrementally whether the pen is up (not writing) or down (writing). On the other hand, writing on the scope is limited in speed by the fact that the trace must remain unblanked at a given location for a certain minimum time (20μs) in order to generate a stored dot. When not writing, the scope trace may move as fast as the scope deflection electronics allows. The specified settling time for trace movement within one dot diameter of the final position is 3.5μs/cm + 5μs.

Thus the scope is faster than the plotter by two orders of magnitude for writing operations and by three orders of magnitude for nonwriting operations.

Chapter 3

THE EXISTING INTERFACE BETWEEN PLOTTER AND COMPUTER

In Chapter 2, a possible approach to the functional equivalence of scope and plotter graphics was suggested. In the new system, this equivalence must be attained not only at the level of device operation but also through computer and interface behavior. The first step towards the design of the new system is a study of the computer and of the existing plotter interface. This study, in the light of the comparison of scope and plotter characteristics, should provide useful ideas for the new interface and its relation to the old.

Plotter Graphical Capabilities

Each of the ten plotter commands described previously corresponds to some elementary action. However, the user of the plotter graphics system generally does not want to specify each command necessary to construct any line or character. He should be able to make requests in terms

of the coordinates of points, lines between any two points, and characters or character strings to be drawn. The function of the three software routines (PLOT, SYMBOL, and NUMBER) in Figure 1 is to take such user-oriented requests and break them down into sequences of plotter commands.

The key software routine is PLOT. Its arguments specify the X and Y coordinates of the point to which the pen is to be moved, either in the up or down position, and whether the new point is to be considered as the origin for subsequent pen operations (otherwise the existing frame of reference continues to be used). PLOT generates the appropriate sequence of plotter commands to move the pen in approximately a straight line to the new coordinate location. It is important to recall that the pen actually changes location by a series of incremental steps whether writing or not.

ASCII[1] codes and some special character codes are related to sequences of straight line segments through

---

1 American Standards Code for Information Interchange

extensive tables contained in the SYMBOL routine. To draw the desired charcters, SYMBOL calls PLOT once for each of the straight line segments. The NUMBER routine converts floating-point numbers into ASCII-coded character strings, with optional signs and decimal point, then calls SYMBOL to draw the numerical symbols. In both of these routines the size of the characters and the angle and starting-point of the resulting print are specified by the user.

Operational Characteristics

Once a user's graphical request has been reduced to individual plotter commands, these commands must be presented to the plotter. Since the computer can generate instructions much faster than the plotter can use them, an output buffer is a practical necessity. In conjunction with the computer interrupt system, the buffering scheme will allow the computer to perform other computations rather than waiting idly for the plotter to act on a command. For instance, if the plotter has just received a command, the computer sets the plotter "control bit" and clears the "flag bit". (See Figure 3.) The computer can continue to another set of instructions.

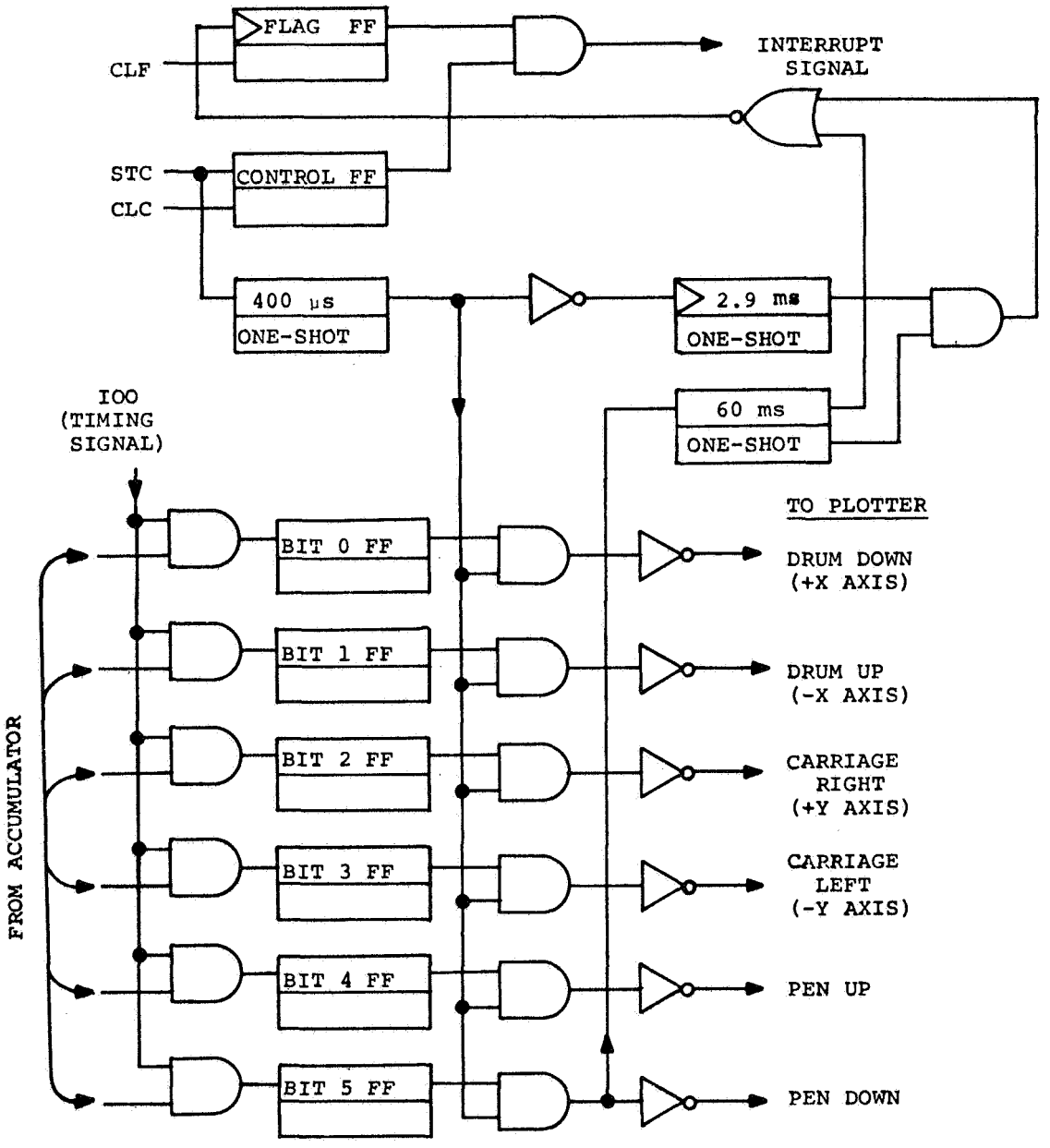


Figure 3.

Simplified logic diagram for existing plotter interface hardware.

After 3.3 ms (or longer for a "pen down" command), during which the computer has gone through 1650 timing cycles (on the order of 1000 typical instructions), the plotter flag bit is set by the interface and an interrupt occurs.

The interrupt stops execution of the present program on the computer and transfers control to a reserved location (the "interrupt location"). An instruction at that location should transfer control to the continuator section entry point of the I/O driver routine (see Appendix A). This routine removes the next command from the buffer and gives it to the plotter interface. The flag bit is cleared to ready the interrupt system again, and then control returns to the interrupted program. A command has just been received by the plotter; the system is back where it started.

To start the above cycle, PLOT stores plotter commands in a buffer in core memory. When PLOT has completed the breakdown of a requested "move", or when it has filled the buffer, it makes an output (write) request to the I/O control system routine, specifying the unit-reference number of the plotter along with the buffer size and location. The unit reference number determines the equipment table entry address for the plotter. (The equipment table is part of

The system software containing information about all I/O facilities used by a given system.) Here the I/O control program finds the address of the I/O driver initiator section and the channel number of the plotter. Control is transferred to the driver, which initializes buffer pointers, output channel number, and other output bookkeeping. Then the driver gives the plotter its first command through the output hardware and sets the interrupt system as described above. The plotter has just received a command, so the interrupt cycle has started again.

The cycle ends when the buffer is empty. If PLOT had finished breaking down its original request, then while the buffer was being emptied other computations may have been performed. However, new calls to PLOT are not acknowledged until the buffer is empty. If PLOT had not finished its original request because the buffer became full, control remained in a small loop within PLOT waiting for the buffer to empty.

Chapter 4

STORAGE OSCILLOSCOPE INTERFACE DESIGN CONSIDERATIONS

Restatement of Objectives

The objectives of the new system are:

1. To give the scope the same functional graphical capabilities as the plotter;

2. To develop operational analogies between scope and plotter functions as suggested in Chapter 2.

3. To use common software wherever possible, as might follow from (1), and (2) thereby simplifying implementation of the new system and reducing core storage requirements;

4. To take advantage of the writing speed of the scope;

5. To take advantage of the potential speed of movement of the scope trace when not writing;

6. To design simple interface hardware for optimal interaction with existing computer hardware and software output facilities.

Software

It has been shown that an approximation to any straight line segment may be produced on the scope by the appropriate series of dots. By analogy with the plotter system, the scope software should have a basic routine for decomposing users' straight-line requests into sequences of output commands to the scope interface. To help satisfy the first three objectives, the existing SYMBOL and NUMBER routines may be used unchanged in the new system. The user, both directly and indirectly through SYMBOL and NUMBER, will call a new PLOT routine which must be able to break down straight-line requests into either plotter or scope commands.

In realizing the first five general objectives, several particular considerations must be made. First, PLOT should distinguish between plotter and scope operation in such a way that calls to PLOT are made just as they are in the present system, i.e. with the same number and interpretation of arguments. Thus no changes need be made in existing programs that use only the plotter. Second, in scope operation, PLOT must respond differently to writing and nonwriting requests. Because of the great speed of trace deflection when not writing, the software should place new coordinate values in the X and Y scope input registers as quickly as possible. In dot writing operations, the registers will be updated by a new command at most every 20μs. Third, since scope and plotter requests may be intermingled in time, current pen (plotter) and trace (scope) location coordinates and status information must be maintained in separate memory locations. Closely related to this issue is the definition of the origins of the coordinate systems. Ultimate plotter commands are incremental; any plotter coordinate system must be maintained by software. On the other hand scope trace coordinates are maintained in the D/A converter input registers, and merely recorded by the software. There

is one "fixed" coordinate system for the scope, unlike the plotter.

The plotter can accept new six-bit commands every 3.3 ms except for a "pen-down" command which requires 60 ms. The computer can generate plotter commands approximately one hundred times faster. Therefore, the plotter software places commands in a buffer; then whenever the plotter is ready an interrupt occurs, transferring control to an output driver which outputs the next command in the buffer to the plotter interface and activates the plotter. On the other hand, the scope writes a dot in 20μs and requires at most 100μs to move the trace when not writing. Since the computer takes approximately 50μs to prepare any new command coordinates for output to the scope interface, the need for buffering, interrupts, and an output driver is obviated. It would be wasteful to put commands in a buffer while the scope is idle and then interrupt subsequent programs every ten machine cycles for output to the scope.

Hardware

The purpose of the hardware for the scope interface is to transfer commands from the computer output hardware to the scope input terminals in such a way that they will result in proper display on the scope. Most important are the ten-bit registers providing input for the D/A converters, which in turn provide input voltages to the X-axis and Y-axis scope inputs. These registers hold the current coordinate values at all times. It will also be remembered that the plotter interface hardware handled the timing of plotter commands and interrupts. The principal design decisions for the scope interface hardware are how to handle timing without interrupts and what information to transfer from the two 16-bit computer accumulators provided with output capability to the D/A input registers. Timing is discussed in Chapter 5. Also to be considered are several digital control inputs to the scope. The final hardware design will affect software output instruction formats.

Three approaches were considered for the content of the computer output data which are to be presented to the interface hardware. The first involved incremental commands to two up-down counters, used as the D/A converter

input registers. The second approach added the possibility of multiple increments particularly for fast incrementing or decrementing of the up-down counters while the scope is not writing. The advantage of these approaches is the small number of bits needed as input to the up-down counters, and thus the small amount and simplicity of the hardware needed. The disadvantage of both is that the computer must calculate incremental commands for the non-writing requests, therefore, not taking advantage of the faster scope trace movement.

A third approach is to load the coordinates for each new command into all ten bits of each D/A input register, whether writing takes place or not. The concept is simple and so is the hardware. The two coordinate values would be kept in memory anyway. This approach is used in the final design. The one complication is that more than twenty bits of output are needed to specify the coordinates and scope function; one computer output instruction transfers only 16 bits.

Chapter 5

AN INTERFACE PROPOSAL

The New PLOT Routine

PLOT is the only software routine that is changed for the new system. The choice between plotter and scope operation is made by calling a new entry (CHOOZ) in the PLOT routine. The one argument passed by a call to CHOOZ determines whether subsequent calls to PLOT will reference the plotter or the scope. Each device therefore operates independently of the other, and, in particular, calls to PLOT for plotter operation are exactly the same as they are in the existing system. A detailed flowchart of the modifications to the PLOT routine may be found in Appendix B.

If scope operation has been specified, then the modified PLOT routine carries out three procedures. First, certain bits in storage are set depending on which of the scope functions are to be used. Second, if the write, write-through, or non-store scope functions are specified by the third argument of PLOT, the software decomposes the PLOT request into individual dot writing commands. The algorithm used is similar to the one for

plotter operation in the existing PLOT routine. Thus an approximation to a straight line will be drawn from the current trace location to the specified new location, as described in more detail below. Third, the X and Y coordinates of a new command are placed in the two accumulator registers of the computer. The function bits are merged into the unused high-order bits of the accumulator containing the Y coordinate. The high-order bits of the other accumulator are set to zero. When the PLOT routine detects that the flag bit of the interface hardware is set, two output instructions are executed to give the accumulator contents to the interface hardware. For a write or erase operation, the flag bit then is cleared.

In those operations using the second procedure described above, each dot writing command is transferred to the interface hardware as soon as it is generated. That is, the second and third procedures form a loop which iterates until the requested line is completely drawn. Nonwrite and erase requests skip the second procedure and perform the output procedure only once before control returns to the program that called PLOT.

Storage Oscilloscope Interface Hardware

A computer output instruction makes available to the specified interface hardware the contents of a 16-bit accumulator. Figure 4 shows the essential logic of the scope interface hardware. Bits 0 to 14 of the computer output are used. The computer output instruction also provides the IO0 timing signal, which strobes bits 0 to 9 into one of the D/A converter input registers. If bit 10 is set to zero, the X-coordinate is loaded; otherwise, the Y-coordinate. Bits 11 through 14 control the non-store, erase, Z-axis, and write-through scope inputs, respectively.

The new PLOT routine transfers the X-coordinate with one output instruction and the Y-coordinate and function control bits with another. The X-coordinate must be transferred first in the scheme employed here. If bit 10 is set to zero, the IO0 timing signal causes the triggering of a 2-μs one-shot. The trailing edge of this 2-μs pulse sets the flag bit. When the PLOT routine detects that the flag bit has been set, it transfers the Y-coordinate, with bit 10 set to one and control functions determined by bits 11 through 14. Bits 11 and 14 enable the non-store and write-through scope inputs, respectively, for 30μs. Bit 11, 13 or 14

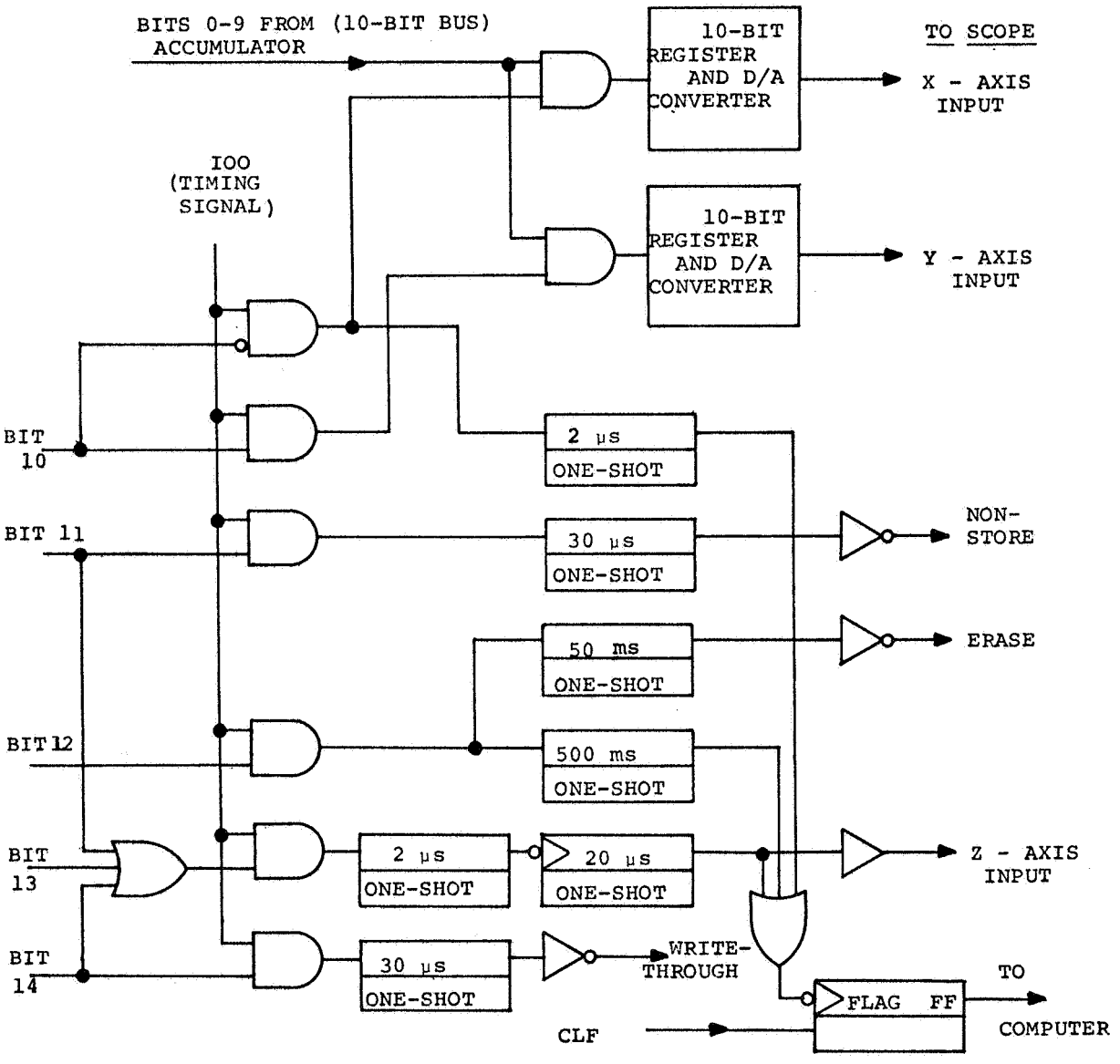


Figure 4.

Simplified logic diagram for proposed scope interface hardware.

triggers a 2-μs delay one-shot, to allow for settling of the D/A converters. After this delay, a 20-μs pulse is sent to the Z-axis input in order to unblank the scope trace. If only bit 13 is set, this results in storage of a dot by the scope. The trailing edge of the 20-μs pulse sets the flag bit. Bit 12 enables the scope erase input for 50 ms, and sets the flag bit after 500 ms.

If none of the bits 11 to 14 is set, a nonwriting command has been given. Since this will have been the only command given by a particular call to PLOT, the trace will have settled in its new position by the time PLOT can be reentered and another command issued.

Chapter 6

SUMMARY AND CONCLUSIONS

By establishing an analogy between scope and plotter writing techniques, it has been possible to adapt the design of the plotter interface software for use by the scope and the plotter. The new software contains additional coding for scope operation, and requires approximately ten percent more storage than the present software. The new scope interface hardware is independent of the plotter hardware, but the function and operation of the two are similar. All of the objectives of the interface specified in Chapter 4 have essentially been attained.

Implementation of the proposed system will involve coding of the algorithm presented in Appendix B and construction of the interface hardware. It is felt that this proposal presents the best possible scope interface using limited hardware and the given writing technique, with the plotter remaining in the system.

APPENDIX A

DETAILS OF EXISTING PLOTTER SOFTWARE

Figures A-1 and A-2 are detailed flowcharts of the algorithms used in the existing PLOT and I/O Driver routines of the Calcomp plotter graphics software. The flowcharts are derived from Assembler coding for the Hewlett-Packard 2115A computer. The labels above some of the boxes are symbolic addresses to assist in cross-reference with the original coding. All other symbolic names from the coding are underlined; those in the I/O Driver routine are explained there (in the flowchart itself) while those from the PLOT routine are listed and defined below:

PADY - subroutine which searches for the plotter entry in the system equipment table and places the plotter select code in call sequences to the system I/O control routine (.IOC.).

PSTAT - subroutine which calls .IOC. to check plotter status; returns when plotter is available.

X, Y - requested final pen coordinates; first two arguments in call to PLOT.

XPEN, YPEN - current pen doordinates in hundredths of inches.

IDX, IDY - number of incremental commands along each axis for this request.

PREST - subroutine which initializes PBUFI to full buffer length (negative), PXB to point to top of buffer, and PDATA and PWDO to zero.

XYPMC, XPMC - four bits of plotter commands needed to construct combined (vector) and one-axis, respectively, incremental commands for this request.

IC - specifies pen up or down and whether new coordinate system origin is requested; third argument for PLOT.

IP - absolute value of previous IC.

XPLT - intermediate storage used to load buffer.

A - denotes A-register (accumulator).

TRA - operation counter: -2means pen must be raised lowered before incremental commands; -1means incremental commands only; 0means finished.

NC - number of incremental moves yet to be broken down for a particular request.

NR, NT, NA - dummy variables used in the algorithm to break down a request into commands.

PBUFI - unused-buffer-word counter (negative).

PXB - points to active buffer word.

PDATA - data flag: 0 means the buffer is full and output has started; -2 means the buffer is not full but the request is finished.

PWDO - number of words of buffer used for commands during present request.

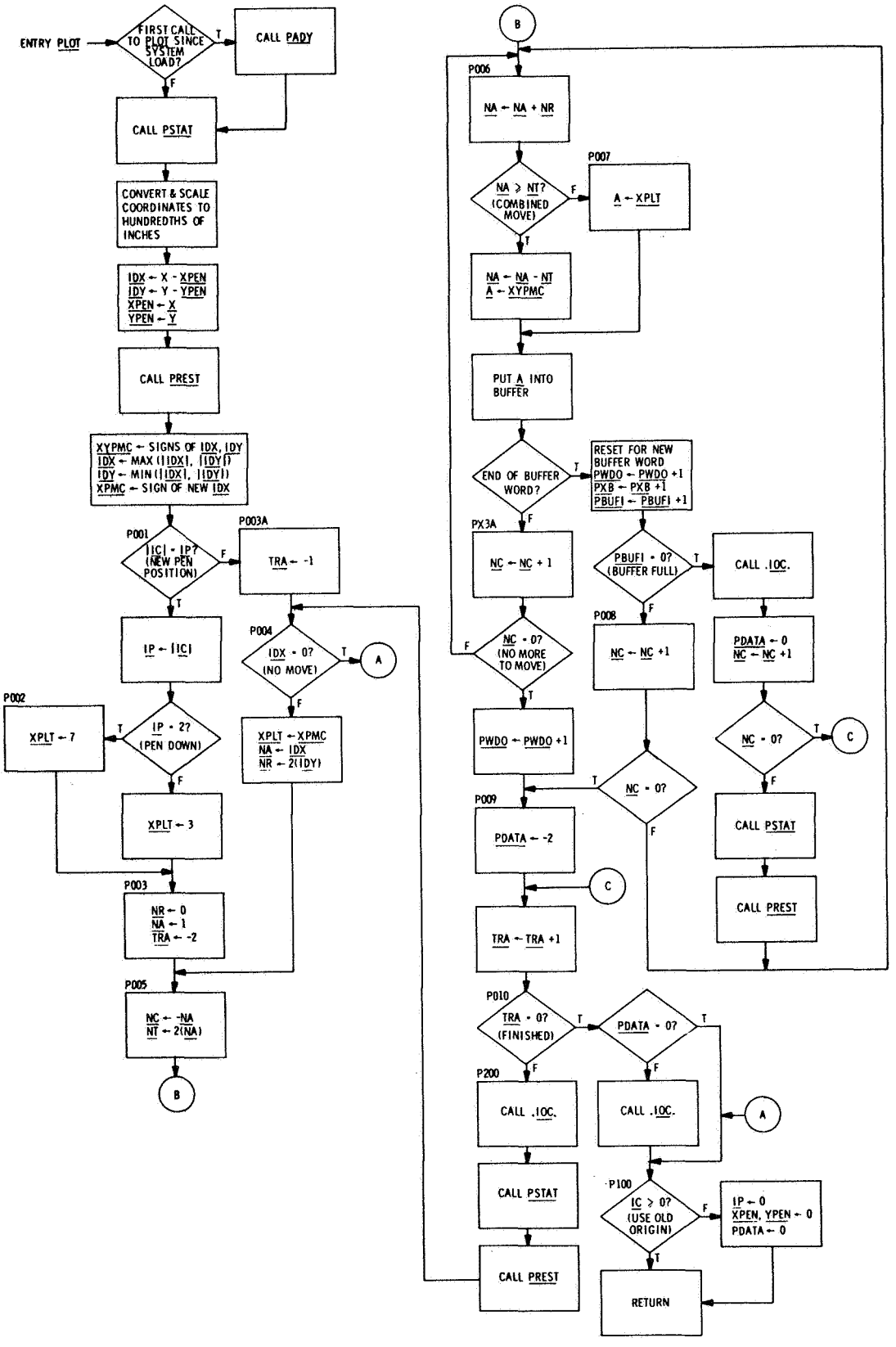


Fig. A-1. The existing PLOT routine.

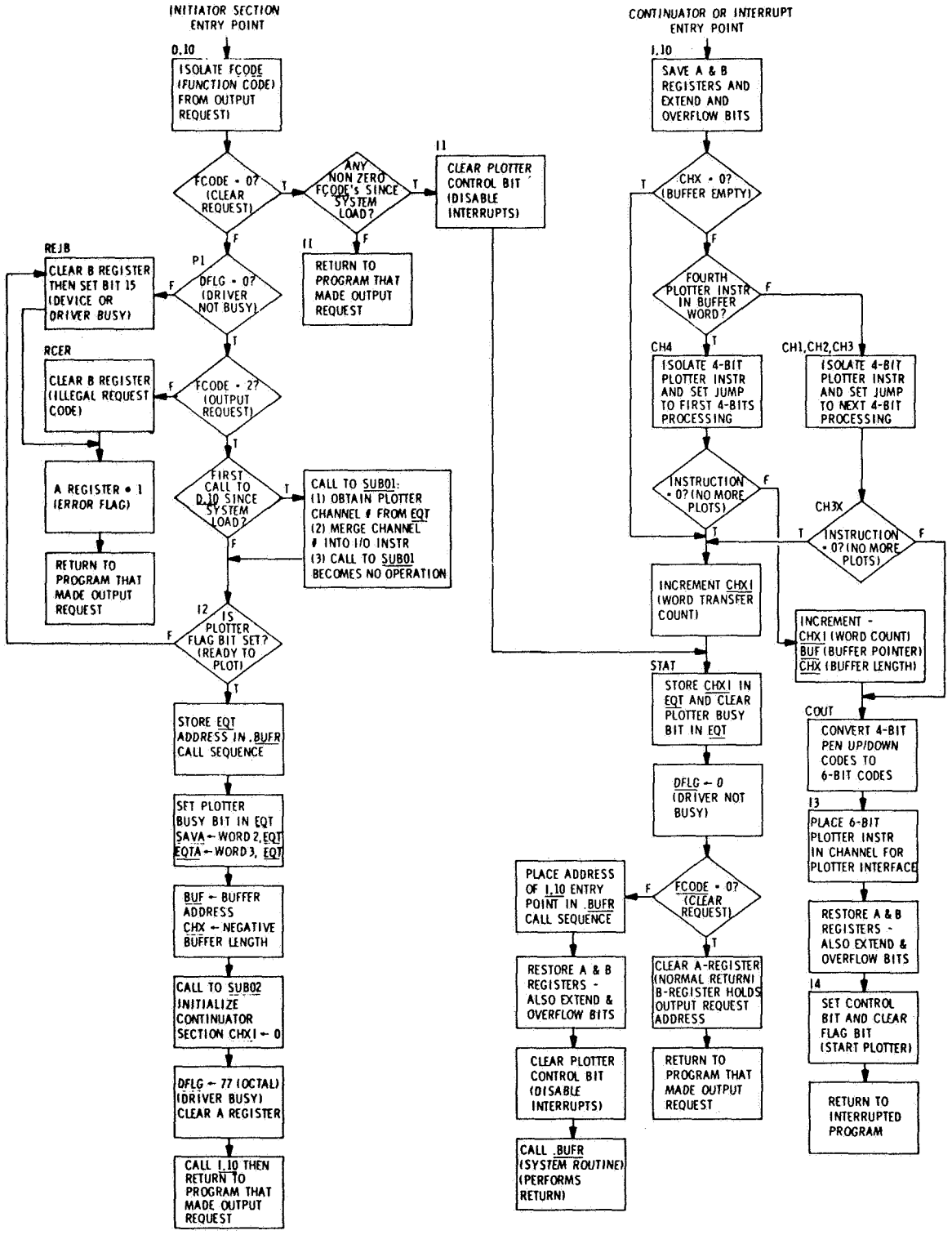


Fig. A-2. The plotter I/O Driver routine.

APPENDIX B

DETAILS OF PROPOSED SOFTWARE MODIFICATION

Figure B-1 is a detailed flowchart of the proposed addition to the PLOT routine. For plotter operation, SCOPE is set to zero and the flowchart continues as in Appendix A (indicated by the dotted arrow and the asterisk). A glossary of symbolic names for the coding is supplied below:

PADY, PSTAT, X, Y, XPEN, YPEN, PREST - See Appendix A.

IC - as in Appendix A, but with extended range of values to represent special scope functions.

SCOPE - set by entry CHOOZ (see Chapter 5) to distinguish between scope and plotter operation.

CODE - contains requested scope function bits to be merged into unused part of Y-coordinate output.

XSCP, YSCP- current trace coordinates.

SDX, SDY - change in scope coordinates for current request.

SA, SC, SR, ST - correspond to NA, NC, NR, NT of Appendix A.

XCOM, YCOM- increments for both coordinates for a combined (vector) command.

XPRT, YPRT -increment (one is zero) for command to move along only one axis.

A, B -denote A and B registers (accumulators).

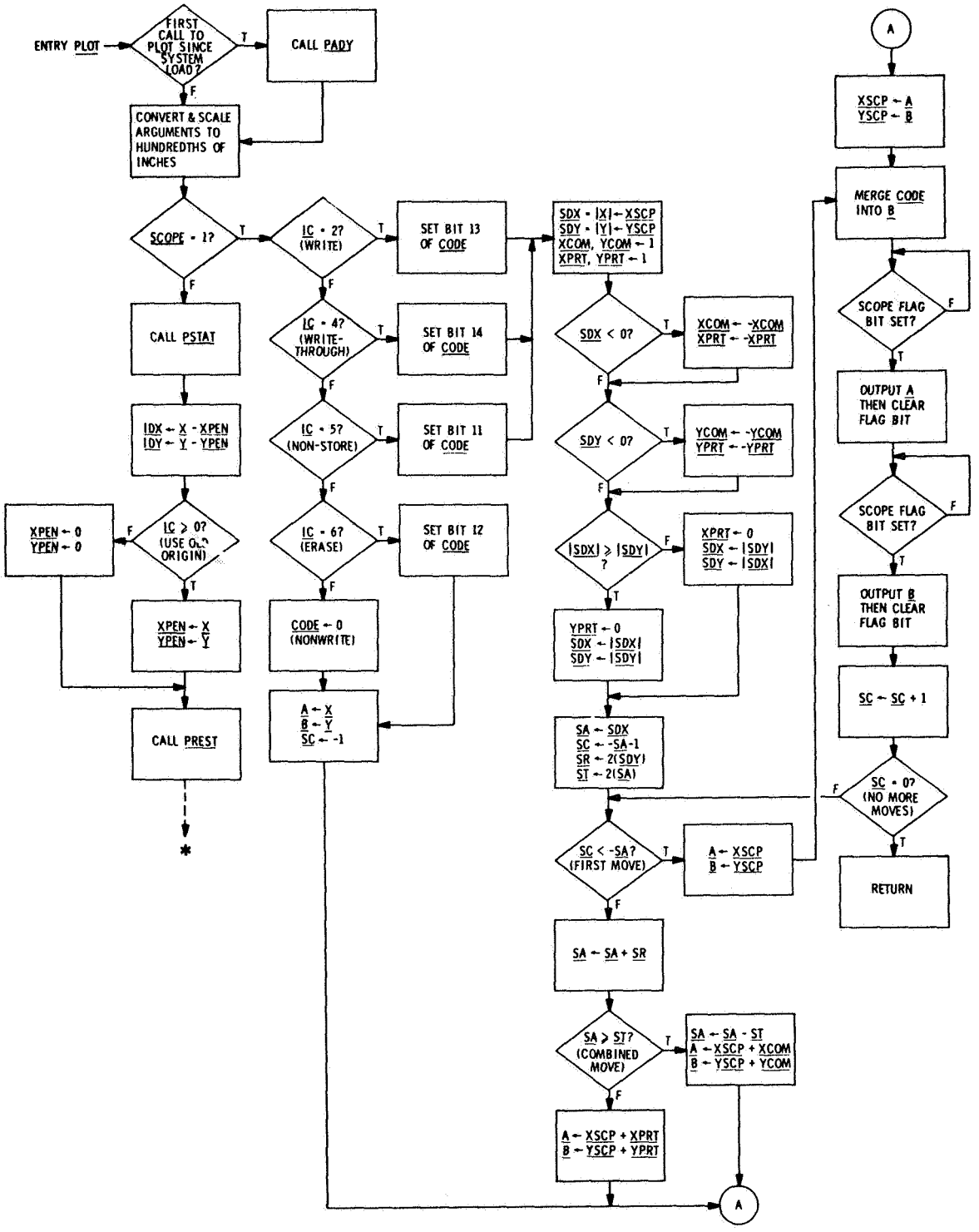


Fig. B-1. The proposed addition to the PLOT routine.

REFERENCES

(1) The following are available from Hewlett-Packard:

-- "Manual for Model 12560A Digital Incremental Plotter Interface" (#12560-9001, Rev. 9/68)

-- "BCS Plotter Driver Source Listing" (#HP20014AL)

-- "Plotter Library Source Listing" (#HP20201BL)

(2) Tektronix "11-inch Storage Display Unit Type 611" see also:

-- Storage Cathode-Ray Tubes and Circuits

-- Information Display Concepts

(available from Tektronix)

(3) Hewlett-Packard Model 2115A Computer manuals:

-- Volume 1: Specifications and Basic Operation (#02471-1, Feb. 1968)

-- Assembler Programmer's Reference Manual

-- Volume 3: Preliminary Manual for Input/ Output System Operation (#02115-9013, Feb. 1968)