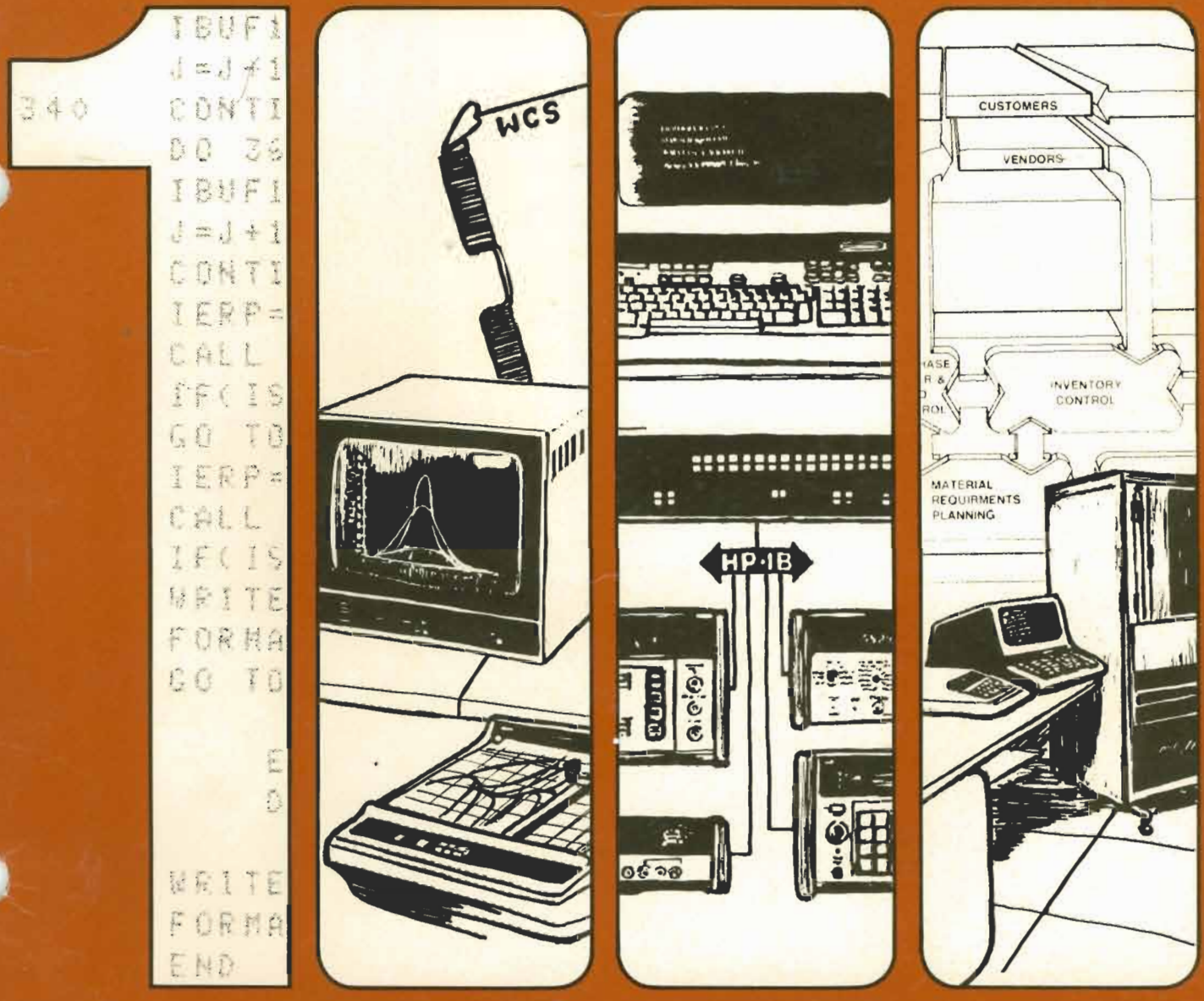
HEWLETT PACKARD

Hewlett-Packard Computer Systems

COMMUNICATOR



1979 Volume III Issue 6

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HEWLETT-PACKARD
COMPUTER SYSTEMS

Volume III
Issue 6

COMMUNICATOR/1000

Feature Articles

Departments

EDITOR'S DESK

IN THE NEXT ISSUE

Look for . . .

- A restructuring of our format
- Announcements of 1980 products
- 1980 Training schedules
- Chapter 3 of Bob Niland's subroutine linkage manual
- An article on using the 264X as a data recorder
- An introduction to PASCAL

and much more . . .

Remember, deadlines for articles for the 1980 Communicator/1000 are:

Issue 1: January 4
Issue 2: March 21
Issue 3: May 30
Issue 4: July 18
Issue 5: September 12
Issue 6: October 31

We are especially looking for articles on RTE-IVB products, new IMAGE, new DATACAP, and all 1980 products. However, we are always interested in ways to make our older products work better.

The Editor

ABOUT THIS ISSUE

Issue 6 of the Communicator/1000, being the last of 1979, once again features an index of all articles published in the past year. Also published again is the Julian calender for the new year. There was such good feedback from last year's calender that this will be a continued feature.

In the OPERATIONS MANAGEMENT section, we have two Data Systems contributors. Darrell Krulce of Sales Development wrote an article describing Steve Witten's TMATT program. Steve worked on the development of DATACAP and saw the need for this program. Darrell added a few enhancements, and then wrote his explanation of the program. Martin Phillips of Technical Marketing also wrote an article about DATACAP, but this one is a description of how DATACAP can be put to use retrieving data. He stresses the security of DATACAP over that of IMAGE or the operating system.

In the OPERATING SYSTEMS section, I have published the first two chapters of Bob Niland's Links/1000 manual. This is the first article in a series which will comprise the entire manual. Bob is at HP's Lexington, Massachusetts sales and service office. His manual is a long-needed, clear explanation of HP's subroutine linkage conventions.

In the COMPUTATION section, Phil Walden of Data Systems Applications Development wrote a terrific article describing Jim Long's PENNY program. PENNY is a drawing program for computer aided design, and Phil's article shows ways it can be put to use.

All the articles were good in this issue and I hope each will be useful in its own way. There was some competition for the HP32E calculator even though Martin Phillips from Technical Marketing is ineligible. Three judges from Data Systems Technical Marketing chose the following articles to be the winners:

Best Feature Article by an HP Division Employee not in Data Systems Technical Marketing	**PENNY: Computer Aided Drawing on the HP 1000** Phil Walden
Best Feature Article by an HP field person	**HP Subroutine Linkage Conventions** Robert Niland

There were no entrants in the customer category for this issue. We hope to have one for the first issue of next year.

The Editor

EDITOR'S DESK

BECOME A PUBLISHED AUTHOR IN THE COMMUNICATOR/1000 . . .

The COMMUNICATOR is a technical publication designed for HP 1000 computer users. Through technical articles, the direct answering of customers' technical questions, cataloging of contributed user programs, and publication of new product announcements and product training schedules, the COMMUNICATOR strives to help each reader utililize their HP 1000's more effectively.

The Feature Articles are clearly the most important part of the COMMUNICATOR. Feature Articles are intended to promote a significant cross-fertilization of ideas, to provide in-depth technical descriptions of application programs that could be useful to a wide range of users, and to increase user understanding of the most sophisticated capabilities designed into HP software. You might think of the COMMUNICATOR as a publication which can extend your awareness of HP 1000's to include that of thousands of users worldwide as well as that of many HP engineers in Data Systems factories at Cupertino, California and Grenoble, France.

To accomplish these goals, editors of the COMMUNICATOR actively seek technical articles from HP 1000 customers, HP Systems Engineers in the Field, and Marketing and R&D Engineers in the factories. Technical articles from customers are most highly valued because it is customers who are closest to real-world applications.

WIN AN HP-32E CALCULATOR!

Authoring a published article provides a uniquely satisfying and visible feeling of accomplishment. To provide a more tangible benefit, however, HP gives away three free HP-32E hand-held calculators to Feature Article authors in each COMMUNICATOR/1000 issue! Authors are divided into three categories. A calculator is awarded to the author of the best Feature Article in each of the author categories. The three author categories are:

1. HP 1000 Customers;

2. HP field employees;

3. HP division employees not in the Data Systems Division Technical Marketing Dept.

Each author category is judged separately. A calculator prize will be awarded even if there is only one entry in an author category.

Feature Articles are judged on the following bases: (1) quality of technical content; (2) level of interest to a wide spectrum of COMMUNICATOR/1000 readers; (3) thoroughness with which subject is covered; and, (4) clarity of presentation.

What is a Feature Article? A Feature Article meets the following criteria:

1. Its topic is of general technical interest to COMMUNICATOR/1000 readers;

2. The topic falls into one of the following categories —

   OPERATING SYSTEMS
   DATA COMMUNICATIONS
   INSTRUMENTATION
   COMPUTATION
   OPERATIONS MANAGEMENT

3. The article covers at least two pages of the COMMUNICATOR/1000, exclusive of listings and illustrations (i.e., at least 1650 words).

There is a little fine print with regard to eligibility for receiving a calculator; it follows. No individual author will be awarded more than one calculator in a calendar year. In the case of multiple authors, the calculator will be awarded to the first listed author of the winning article. An article which is part of a series will compete on its own merits with other articles in the issue. The total of all articles in the series will not compete against the total of all articles in another series. Employees of Technical Marketing at HP's Data Systems Division factory in Cupertino are not eligible to win a calculator.

All winners of calculators will be announced in the issue of the COMMUNICATOR/1000 in which their articles appear. Again, all Feature Articles are judged by an impartial panel of three DSD Technical Marketing Engineers.

A SPECIAL DEAL IN THE OEM CORNER

When an HP 1000 OEM writes a Feature Article that is not only technically detailed and insightful but also application-oriented as opposed to theoretical, then that OEM may ask that the article be included in THE OEM CORNER. A Feature Article included in THE OEM CORNER may contain up to 150 words of pure product description as well as a picture or illustration of the OEM'S product or its unique contribution. HP's objective is twofold: (1) to promote awareness of the capabilities HP 1000 OEMs' products among all HP 1000 users; and, (2) to publish an article of technical interest and depth.

IF YOU'RE PRESSED FOR TIME . . .

If you are short of time, but still have that urge to express yourself technically, don't forget the COMMUNICATOR/1000 BIT BUCKET. It's the perfect place for a short description of a routine you've written or an insight you've had.

THE MECHANICS OF SUBMITTING AN ARTICLE

If at all possible please submit an RTE File containing the text of your article recorded on a Minicartridge (preferrably) or on a paper tape along with the line printer or typed copy of your article. This will help all of us to be more efficient. The Minicartridge will be returned to you promptly. Please include your address and phone number along with your article.

All articles are subject to editorship and minor revisions. The author will be contacted if there is any question of changing the information content. Articles requiring a major revision will be returned to the author with an explanatory note and suggestions for change. We hope not to return any articles at all; if we do, we would like to work closely with the author to improve the article. HP does, however, reserve the right to reject articles that are not technical or that are not of general interest to COMMUNICATOR/1000 readers.

Please submit your COMMUNICATOR/1000 article to the following address:

Editor, COMMUNICATOR/1000
Data Systems Division
Hewlett-Packard Company
11000 Wolfe Road
Cupertino, California 95014
USA

The Editor looks forward to an exciting year of articles in the COMMUNICATOR/1000.

With best regards,

The Editor

LETTERS TO THE EDITOR

Dear Editor,

Would you please explain the reason for having two subprogram linkage conventions (i.e., .ENTC and .ENTR) and under what circumstances each should be used.

Why should one convert calls using .ENTR to .ENTC with the .RCNG subroutine?

Sincerely,

Richard B. Gilbert
Gas Dynamics Laboratory

Dear Richard,

.ENTR is used only with subroutines which, immediately following the JSB instruction, have a DEF to the return point. Routines which use .ENTC do not have this DEF instruction and can only pass a fixed number of parameters, the number of which the caller and callee must agree upon.

.ENTR is preferable to .ENTC because of its parameter flexibility. .ENTC is retained in the relocatable library for backward compatability; there are many routines which were written using .ENTC before .ENTR was developed.

.ENTC can be used for re-entrant and privileged subroutines, whereas .ENTR cannot. The equivalent to .ENTC for re-entrant and privileged routines is .ENTP. However, before .ENTP was developed, any routines using the .ENTR convention which the user occasionally desired to be re-entrant or privileged (put in the memory resident library, for example) needed to have the ability to be easily converted from .ENTR to .ENTC conventions. .RNCG handled this conversion.

You will be interested to know that with this issue the Communicator is beginning a series of articles on HP subroutine linkage which I hope will clear up any confusion in this area.

With Best Regards,

The Editor

Dear Editor,

Thanks for the list of microcode entry points. It's great to finally see these all in one place.

However, I do have a complaint with the bulletin section in Vol. III, Issue 3. I found a description of a workshop that sounded very interesting except that I received my issue of the Communicator on October 17 and the workshop was held on October 11,12. At least I know what I have missed.

Otherwise, keep up the good work.

B. Allen Harbin
R&D Computer Science Services
R.J. Reynolds Tobacco Company

Dear Allen,
I'm sorry about the obsolete date. It takes four weeks from the time each issue goes into typesetting to the time it gets distributed. Sometimes this delay is longer, and we publish late bulletins. I'm glad the entry points were useful. Hopefully, this issue will contain lots of information you can use.

With best regards,

The Editor

LOCUS ANNOUNCEMENT

As you may have noticed new user contributed programs have ceased appearing in the User's Queue section of the Communicator/1000. This is because Locus has been undergoing a change. First of all, it is no longer going to be called Locus. The User Contributed Library is now called PLUS/1000. Secondly, it is no longer associated with the Communicator; it is going to be maintained by the HP 1000 International Users Group. For more information about the PLUS/1000 library see the Bulletins section of this Communicator.

Software Samantha
HP 1000 Communicator
Hewlett-Packard Data Systems Division
11000 Wolfe Road, Cupertino, California 95014

Software Samantha
HP 1000 Communicator
Hewlett Packard Data Systems Division
11000 Wolfe Road, Cupertino, California 95014

Dear Samantha,

In Volume II, Issue 6 of the Communicator, Richard Gilbert inquired as to the meaning of "blocked input/output in the description of MAGTP", you replied that it was an error in the DOS/RTE Relocatable Library Reference Manual. On page 4-25, of that same manual, there is an entire page devoted to the I/O techniques being described are used by making "direct calls to .IOC. (through the MAGTP subroutine)".

This reference to MAGTP is obviously related to, and more confusing than, the one pointed out by Mr. Gilbert. It states that you can get at .IOC. via MAGTP, but there is no entry point for blocked I/O described for MAGTP; and indeed you have told us there is none. Can you shed some light on what the Manual is 'talking about' on page (4-25)? If you can't use MAGTP what do you use? If you do use MAGTP how do you get at it? Does the technique referenced offer an alternative to Class I/O EXEC calls?

Sincerely,

Jeff Wynne
Code 3021 Bldg 759
US Naval Ordnance Station
Indian Head, Maryland 20640

Dear Jeff,

Unfortunately, the DOS/RTE Relocatable Library Reference Manual has lots of carryovers from the old BCS. Many items were overlooked when the manual was updated for use with RTE. This is another case of such forgetfulness. When the reference to .IOC. that Richard Gilbert pointed out was corrected, page 4-25 should have been changed as well.

.IOC. is essentially the EXEC call of BCS. All references to .IOC. should be replaced by references to EXEC calls. The manual will be corrected; please notify me if you find more outdated references.

Sincerely,

Samantha

DRIVER 05 EQT DEFINITIONS

Glenn Talbott/Data Systems Division

In recent months I have noticed an increasing trend in the questions I receive concerning the 264X terminal driver, DVR05. Many of these questions relate to the use of modems and the modem version of this driver, DVA05. Often problems are encountered while attempting out of the ordinary communications such as special terminal control functions, or using new, unsupported modems. Having the contents of each word in the Equipment Table defined would enable users to better understand and diagnose these problems. To aid both our customers and our Systems Engineer force to better understand this driver I have compiled a complete definition of the Equipment Table for DVA05. DVR05 versions for terminals with or without Cartridge Tape Units are subsets of DVA05 so the included table is valid for these versions of the driver.

DRIVER 05 EQT

WORD	USE
1	REQUEST LIST POINTER
2	IA05 INITIATOR ENTRY POINT ADDRESS
3	CA05 CONTINUATOR ENTRY POINT ADDRESS
4	BIT 15 IS 0 (NO DMA) BIT 14 IS BUFFERED/UNBUFFERED (0/1) BIT 13 IS 0 (NO POWER FAIL PROCESSING) BIT 12 IS 1 (ENTER DRIVER ON TIME OUT) BIT 11 IS SET IF TIME OUT OCCURED BITS 10-6 INDICATE LAST SUBCHANNEL ADDRESSED BITS 5-0 DEVICE SELECT CODE
5	BITS 15-14 AVAILABILITY 00 AVAILABLE 01 DOWN 10 BUSY 11 DMA WAIT (NOT USED) BITS 13-8 DRIVER TYPE CODE (05) BITS 7-0 STATUS AFTER LAST REQUEST FOR CTU STATUS: 7 END OF FILE SENSED. A FILE MARK HAS BEEN DETECTED DURING A PRIOR READ OPERATION OR A FILE MARK HAS JUST BEEN RECORDED. 6 LOAD POINT SENSED. CARTRIDGE TAPE IS AT OR BEFORE LOAD POINT MARKER. MEANINGFUL ONLY IF CARTRIDGE IS INSERTED. 5 END OF TAPE SENSED. THE CARTRIDGE TAPE HAS PASSED OVER EARLY WARNING MARKER IN THE TAPE AND AN END-OF-VALID DATA MARK HAS BEEN RECORDED AUTOMATICALLY. COMMANDS DIRECTING FORWARD MOTION OF TAPE WILL BE REJECTED. THIS STATUS ONLY HAS MEANING IF A CARTRIDGE IS INSERTED. 4 READ\WRITE ERROR. WRITE 2645 ONLY A READ ERROR EXISTS IF THREE SUCCESSIVE ATTEMPTS FAILED TO READ THE DATA IN THE RECORD. THE TAPE IS POSITIONED AFTER THE BAD RECORD. 3 LAST COMMAND ABORTED. THE LAST COMMAND INITIATED FROM THE CPU OR KEYBOARD WAS UNSUCCESSFULLY PERFORMED. OTHER STATUS CONDITIONS MAY BE CHECKED FOR CAUSE. 2 WRITE PROTECTED. THE FILE PROTECT TAB ON THE CARTRIDGE IS IN THE POSITION TO PROHIBIT RECORDING OF DATA. THIS STATUS ONLY HAS MEANING IF A CARTRIDGE IS INSERTED. 1 END OF VALID DATA. THE CARTRIDGE TAPE DETECTED AN END-OF-VALID DATA MARK DURING A PRIOR READ OR SEARCH OPERATION, OR HAS JUST COMPLETED RECORDING AN END-OF-VALID DATA MARK. IN EITHER CASE, THE

	TAPE IS POSITIONED BEFORE THE END-OF-VALID DATA MARK. RECORDING OPERATIONS MAY BE EXECUTED TO OVERWRITE THIS MARK WITH DATA OR A FILE MARK, UNLESS THE TAPE IS AT END OF TAPE. 0 CARTRIDGE NOT INSERTED OR UNIT BUSY. NOTE THAT CTU STATUS IN BITS 7 - 1 IS ONLY VALID IF THIS BIT IS A ZERO. FOR CRT STATUS: 7 BUFFER FLUSH IN PROGRESS. 5 CONTROL D ENTERED. THE USER HAS HIT THIS KEY. 4 DATA SET LINE DOWN 3 PARITY ERROR 1 TERMINAL ENABLED. IF THE STATUS WORD IS FOR A TERMINAL (NOT A SYSTEM CONSOLE) TYPING ANY KEY WILL SCHEDULE THE TERMINAL'S PROGRAM (IF IT HAS ONE).
6	CURRENT REQUEST WORD BITS 15-14 CONTAIN REQUEST TYPE LOCATION CODE: 00 USER UNBUFFERED 01 USER BUFFERED 10 SYSTEM REQUEST 11 CLASS I/O REQUEST BITS 10-6 CONTAIN FUNCTION CODE FROM REQUEST FOR CTU CONTROL REQUEST: 01 WRITE END OF FILE 02 BACKSPACE 1 RECORD 03 FORWARD SPACE 1 RECORD 4,5 REWIND 06 DYNAMIC STATUS 10 WRITE EOF IF NOT JUST WRITTEN OR NOT AT BOT 14 BACKSPACE 1 FILE 13 FORWARD SPACE 1 FILE 26 WRITE END OF VALID DATA (EOV) 27 LOCATE FILE. THE ABSOLUTE FILE NUMBER IS A REQUEST PARAMETER. FOR CRT CONTROL REQUEST: 11 SPACE LINES. THE NUMBER OF LINES IS THE REQUEST PARAMETER. IF NEGATIVE, PAGE EJECT FOR 9871 ONLY. 20 ENABLE TERMINAL. ALLOWS TERMINAL TO SCHEDULE PROGRAM WHEN KEY IS STRUCK. 21 DISABLE TERMINAL. INHIBIT SCHEDULING OF TERMINAL'S PROGRAM. 22 SET NEW TIME OUT (IN OPTIONAL PARAMETER) 23 CLEAR THE OUTPUT QUEUE (BUFFER FLUSH). 24 RESTORE OUTPUT PROCESSING. REQUIRED ONLY IF SOME OF THE BUFFER IS TO BE SAVED. 25 UPDATE TERMINAL STATUS 30 TERMINAL MODEM CONTROL, SET UP BAUD RATE AND PARITY. 31 LINE CONTROL FOR MODEMS 32 AUTO-ANSWER FOR MODEMS FOR READ AND WRITE REQUESTS: BIT 10 0/1 TRANSPARENT MODE OFF/ON BIT 8 0/1 ECHO OFF/ON BIT 6 0/1 ASCII/BINARY BITS 10,9 = 11 INDICATES A PROGRAM ENABLED BLOCK READ BITS 1-0 REQUEST TYPE 01 READ 10 WRITE 11 CONTROL

7	REQUEST BUFFER ADDRESS OR OPTIONAL PARAMETER ON CONTROL REQUEST. MAY ALSO CONTAIN A 10004B WHILE WAITING FOR RING AFTER AUTO-ANSWER MODEM ENABLE
8	REQUEST BUFFER LENGTH (+ WORDS, – CHARACTERS). MAY BE SET TO ZERO WHEN ZERO TRANSMISSION LOG IS REQUIRED ON COMPLETION EXIT. ALSO USED FOR RETURN ADDRESS STORAGE FOR INTERNAL DRIVER SUBROUTINES <BSR1>, <BSR2> (BACKSPACE ONE OR TWO RECORDS), <FRS1> (FOWARD SPACE 1 RECORD), AND <BSF1> (BACKSPACE ONE FILE)
9	RUNNING CHAR. ADDRESS OR NUMBER OF RECORDS TO BACKSPACE (ASCII 1 OR 2) FOR <BSR(ASCII 1> OR <BSR2> (BACKSPACE ONE OR TWO RECORDS)
10	LAST CHARACTER ADDRESS; OR FOR CTU AND PRINTER CONTROL REQUESTS BITS 4 - 0 CONTAIN THE FUNCTION CODE (EQT6 BITS 10 - 6)
11	ADDRESS TO GO ON INTERRUPT TO THE CONTINUATION SECTION. SEVERAL INTERNAL SUBROUTINES KEEP THEIR RETURN ADDRESSES HERE AND EXIT TO WAIT FOR THE DEVICE TO RESPOND. THE NEXT INTERRUPT FROM THE DEVICE (IF IT IS NOT A TIME-OUT) WILL CAUSE A SUBROUTINE RETURN (JMP ,I) TO THE ADDRESS KEPT HERE.
12	NO. OF EQT EXTENSION WORDS OR LAST REQUEST WORD (EQT6) FOR ABORT PROCESSING.
13	EQT EXTENSION STARTING ADDRESS.
14	TIME OUT RESET VALUE FOR CRT READ TIME OUT. OTHER RESET VALUES ARE FIXED INTERNALLY FOR: CRT WRITE 4 SEC. NON CRT FUNCTIONS 60 SEC. MODEM LINE OPEN/CLOSE 2 SEC.
15	CURRENT TIME OUT CLOCK
16	TERMINAL STRAPPING AND CTU INFO (BITS 13 5 ARE ONLY USED BY THE MODEM DRIVER, DVA05): BIT 15 0\1 =LINE\PAGE BIT 14 0\1 =CHAR.\BLOCK BIT 13 LINE 0\1 =HARD\MODEM BIT 12 "CA" (RTS) SET BIT 11 "CD" (DTR) SET BIT 10 PARITY ON\OFF = 1\0 BIT 9 PARITY EVEN\ODD = 1\0 BIT 8-5 BAUD RATE BIT 4 KEYBOARD LOCKED BIT 3 <CN50C> FLAG (INDICATES INTERNAL SUBROUTINE <CN50C> IS PROCESSING A CTU POSITIONING REQUEST, AND SPECIAL PROCESSING IS REQUIRED) BIT 2 RCTU EOF FLAG BIT 1 LCTU EOF FLAG BIT 0 TERMINAL STRAPPING ALREADY READ
17	ID ADDRESS OF TERMINAL PROGRAM. 0 BEFORE SETUP, –1 IF NO PROGRAM TO SCHEDULE, OR ID SEGMENT ADDRESS OF PROGRAM (USUALLY PRMPT) TO SCHEDULE ON UNEXPECTED INTERRUPT IF EQT28 BIT 1 IS SET (TERMINAL ENABLED).
18	NOT USED
19	RETURN ADDRESS FOR INTERNAL SUBROUTINES <TRAN1> OR <OUT4>. A BUFFER ADDRESS POINTER FOR THE SUBROUTINE <BINAS>. VALUE FOR A REGISTER ON COMPLETION EXIT.

20	BINARY RECORD LENTGH COUNTER. $UPIO ENTRY FLAG, SET TO 10032B BY MODEM LINE UP DETECTION CODE. WHEN DEVICE UP IS REQUIRED ON NEXT CONTINUATOR ENTRY. USED AS A COUNTER BY INTERNAL SUBROUTINE <BINAS>.
21	NOT USED
22	RETURN ADDRESS FOR INTERNAL SUBROUTINE <BINAS>. A REGISTER SAVE LOCATION FOR INTERNAL SUBROUTINE <CTPRP>.
23	RETURN ADDRESS FOR INTERNAL SUBROUTINE <ENAK>
24	RETURN ADDRESS FOR INTERNAL SUBROUTINE <CTUST>. LINE CONTROL REF. (MODEM); COMPARED TO INTERFACE STATUS WORD TO TEST FOR MODEM LINE OPEN/CLOSE COMPLETION.
25	RETURN ADDRESS FOR INTERNAL SUBROUTINE <CTPRP>. LINE CONTROL FLAG; IF 32B ON TIME OUT ENTRY CONTROL 31 OR 32 IS IN PROGRESS.
26	NOT USED
27	RETURN ADDRESS FOR INTERNAL SUBROUTINES <USINT> AND <TERST>
28	TERMINAL STATUS BIT 1 TERMINAL ENABLED BIT 3 PARITY ERROR BIT 4 DATA SET ERROR (LINE DOWN) BIT 5 CNTRL D ENTERED (EOT) BIT 7 BUFFER FLUSH IN PROGRESS

DEFINITIONS OF SUBROUTINE NAMES USED:

<BINAS>	Converts binary values to ASCII saving the ASCII in an internal buffer for transmission.
<BSF1>	Backspace one file on CTU
<BSR1>	Backspace one record on CTU
<BSR2>	Backspace two records on CTU
<CN50C>	Tape positioning subroutine that does the special processing required when a tape is positioned after an EOF mark.
<CTPRP>	Prepares terminal to accept data by sending the control sequences for CTU's if required, and locking the keyboard.
<CTUST>	Reads the CTU status
<ENAK>	Handles the ENQ/ACK handshake
<FRS1>	Forward space one record for CTU
<OUT4>	Output routine used in CTU control processing
<TERST>	Reads the terminal status and updates EQT16
<TRAN1>	Handles user writes to the display, CTU, printer
<USINT>	Tests for user keyboard interrupt

JANUARY 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1 (1)	2 (2)	3 (3)	4 (4)	5 (5)
6 (6)	7 (7)	8 (8)	9 (9)	10 (10)	11 (11)	12 (12)
13 (13)	14 (14)	15 (15)	16 (16)	17 (17)	18 (18)	19 (19)
20 (20)	21 (21)	22 (22)	23 (23)	24 (24)	25 (25)	26 (26)
27 (27)	28 (28)	29 (29)	30 (30)	31 (31)		

FEBRUARY 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1 (32)	2 (33)
3 (34)	4 (35)	5 (36)	6 (37)	7 (38)	8 (39)	9 (40)
10 (41)	11 (42)	12 (43)	13 (44)	14 (45)	15 (46)	16 (47)
17 (48)	18 (49)	19 (50)	20 (51)	21 (52)	22 (53)	23 (54)
24 (55)	25 (56)	26 (57)	27 (58)	28 (59)	29 (60)	

MARCH 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
2 (62)	3 (63)	4 (64)	5 (65)	6 (66)	7 (67)	8 (68)
9 (69)	10 (70)	11 (71)	12 (72)	13 (73)	14 (74)	15 (75)
16 (76)	17 (77)	18 (78)	19 (79)	20 (80)	21 (81)	22 (82)
23 (83)	24 (84)	25 (85)	26 (86)	27 (87)	28 (88)	29 (89)
30 (90)	31 (91)	30 (90)	31 (91)			

APRIL 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1 (92)	2 (93)	3 (94)	4 (95)	5 (96)
6 (97)	7 (98)	8 (99)	9 (100)	10 (101)	11 (102)	12 (103)
13 (104)	14 (105)	15 (106)	16 (107)	17 (108)	18 (109)	19 (110)
20 (111)	21 (112)	22 (113)	23 (114)	24 (115)	25 (116)	26 (117)
27 (118)	28 (119)	29 (120)	30 (121)			

MAY 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1 (122)	2 (123)	3 (124)
4 (125)	5 (126)	6 (127)	7 (128)	8 (129)	9 (130)	10 (131)
11 (132)	12 (133)	13 (134)	14 (135)	15 (136)	16 (137)	17 (138)
18 (139)	19 (140)	20 (141)	21 (142)	22 (143)	23 (144)	24 (145)
25 (146)	26 (147)	27 (148)	28 (149)	29 (150)	30 (151)	31 (152)

JUNE 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1 (153)	2 (154)	3 (155)	4 (156)	5 (157)	6 (158)	7 (159)
8 (160)	9 (161)	10 (162)	11 (163)	12 (164)	13 (165)	14 (166)
15 (167)	16 (168)	17 (169)	18 (170)	19 (171)	20 (172)	21 (173)
22 (174)	23 (175)	24 (176)	25 (177)	26 (178)	27 (179)	28 (180)
29 (181)	30 (182)					

JULY 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1 (183)	2 (184)	3 (185)	4 (186)	5 (187)
6 (188)	7 (189)	8 (190)	9 (191)	10 (192)	11 (193)	12 (194)
13 (195)	14 (196)	15 (197)	16 (198)	17 (199)	18 (200)	19 (201)
20 (202)	21 (203)	22 (204)	23 (205)	24 (206)	25 (207)	26 (208)
27 (209)	28 (210)	29 (211)	30 (212)	31 (213)		

AUGUST 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1 (214)	2 (215)
3 (216)	4 (217)	5 (218)	6 (219)	7 (220)	8 (221)	9 (222)
10 (223)	11 (224)	12 (225)	13 (226)	14 (227)	15 (228)	16 (229)
17 (230)	18 (231)	19 (232)	20 (233)	21 (234)	22 (235)	23 (236)
24 (237)	25 (238)	26 (239)	27 (240)	28 (241)	29 (242)	30 (243)
31 (244)						

SEPTEMBER 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
	1 (245)	2 (246)	3 (247)	4 (248)	5 (249)	6 (250)
7 (251)	8 (252)	9 (253)	10 (254)	11 (255)	12 (256)	13 (257)
14 (258)	15 (259)	16 (260)	17 (261)	18 (262)	19 (263)	20 (264)
21 (265)	22 (266)	23 (267)	24 (268)	25 (269)	26 (270)	27 (271)
28 (272)	29 (273)	30 (274)				

OCTOBER 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
			1 (275)	2 (276)	3 (277)	4 (278)
5 (279)	6 (280)	7 (281)	8 (282)	9 (283)	10 (284)	11 (285)
12 (286)	13 (287)	14 (288)	15 (289)	16 (290)	17 (291)	18 (292)
19 (293)	20 (294)	21 (295)	22 (296)	23 (297)	24 (298)	25 (299)
26 (300)	27 (301)	28 (302)	29 (303)	30 (304)	31 (305)	

NOVEMBER 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1 (306)
2 (307)	3 (308)	4 (309)	5 (310)	6 (311)	7 (312)	8 (313)
9 (314)	10 (315)	11 (316)	12 (317)	13 (318)	14 (319)	15 (320)
16 (321)	17 (322)	18 (323)	19 (324)	20 (325)	21 (326)	22 (327)
23 (328)	24 (329)	25 (330)	26 (331)	27 (332)	28 (333)	29 (334)
30 (335)						

DECEMBER 1980

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
	1 (336)	2 (337)	3 (338)	4 (339)	5 (340)	6 (341)
7 (342)	8 (343)	9 (344)	10 (345)	11 (346)	12 (347)	13 (348)
14 (349)	15 (350)	16 (351)	17 (352)	18 (353)	19 (354)	20 (355)
21 (356)	22 (357)	23 (358)	24 (359)	25 (360)	26 (361)	27 (362)
28 (363)	29 (364)	30 (365)	31 (366)			

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY ISSUE

COMMUNICATOR INDEX BY CATEGORY

COMMUNICATOR INDEX BY CATEGORY

HP SUBROUTINE LINKAGE CONVENTIONS

Robert Niland/HP Lexington

Editor's note: This article is the first in a series of articles taken from the Links/1000 manual written by Robert Niland. This is a user contributed manual on HP subroutine linkage conventions which has not yet been published.

INTRODUCTION

1-1. HP 1000 Linkage System

The HP 1000 subroutine linkage system consists of three components:

- Program coding conventions.
- A set of utility subroutines.
- Supporting capabilities in the operating system programs.

The primary intent of this manual is to document the coding conventions. Discussion of the utility subroutines and examples of their use are included, but the final authority on calling sequences and returns for those utilities are the system and library manuals.

Beyond explaining the conventions, a further purpose of this manual is to help the user select the most appropriate programming approach for each application.

1-2. Source Code Conventions

For clarity and consistency, some conventions are used in this manual. They are:

P — Program location counter. A 15 bit register which holds the address of the next machine instruction to be executed. It is important for the beginning Assembly language programmer to recognize that the hardware and firmware of the HP 1000 have no way to distinguish between memory locations which contain instructions and those which contain data. If program control is transferred (i.e. the P register gets sets to . .) memory location 034163B, and that location contains a data value of 10282, the computer will execute a JMP to location 034052B. For 10282 = 024052B, which is a JMP to location 0052B of the current page. The result of such an execution error is unpredictable, and this manual will document coding techniques which minimize or eliminate the chances of such bugs.

< > — Enclosures indicate a required syntax element which is to be replaced in its entirety (including < >) by a symbol or expression appropriate to the language being used.

[ ] — Braces indicate an optional syntax element which may be deleted entirely. However, if included, the syntax denoted within (but not including) the braces must be observed.

••• — Three dots in an op-code or statement field implies that there are one or more additional lines of unspecified code which are not shown, but which do not modify any variables or destroy the contents of any registers being used by the sample code.

MPX — This is an 'MIC' micro-code replacement op-code which will appear frequently in the sample listings, often where more experienced programmers would expect to see a NOP. It is generated by the following Assembler pseudo-instruction:

```
MIC MPX,102000B,0    MP abend if location eXecutes.
```

It is used in place of the NOP, BSS, DEC 0, and OCT 0 in in every program location which must be written into before it is executed (or read from).

An octal 102000 is a HLT (HaLT) instruction. It has a decimal value of −31744. The HLT is an illegal opcode in an RTE program. That is, if the HLT is executed, RTE will abort the program, and log a Memory Protect (MP) error on the console.

This is particularly useful for beginning ASMB programmers who inadvertently type in JMP SUBR where a JSB SUBR was intended. The error is trapped instantly at subroutine entry instead of resulting in a subroutine's attempting to execute any address in the entry point as if it were an instruction and exiting to the previous caller's return point. Coding errors of this type can otherwise be very difficult to isolate.

SYMB All locations in assembly listings in which the data is an ADDRESS or LINK (i.e. are not ASC, DEC, OCT, or BYT constants or variables), will be given symbolic names which have a leading "@" character, denoting "Address". Examples:

```
        STA  INDEX      Typical STore A direct.
        •••
        LDB  @NDEX,I    Typical LoaD B Indirect.
        •••
        JMP  ENTRY,I    Subroutine returns are an exception.
INDEX   MPX             A variable.
        •••
@NDEX   DEF  INDEX      DEFining address of INDEX.
```

bytes Lower case symbolic names denote a reference to BYTE or CHARACTER oriented locations, e.g.

```
        •••
        LDB  ascQ       LoaD B with byte address of source.
        •••
        LBT             Load a ByTe.
        •••
ascQA   ASC  01,QA      ASCii constant "QA".
ascQ    DBL  ascQA      Define Byte address, Left.
```

1-3. Terminology

Program & Subroutine: For the sake of clarity, this manual will always refer to the calling code as the program and the called code as the subroutine. Of course, subroutines may call other subroutines. And in fact, if the main program has multiple entry points, one of that program's subroutines might call another subroutine which is actually part of the main program. The terms program and subroutine are used only to denote who is subordinate to whom with respect to the code currently executing.

THE JSB INSTRUCTIONS

2-1. The Foundation of HP 1000 Subroutines

The implementation of a subroutine scheme on any computer relies on two concepts:

- Transfer of control, i.e. program execution is to resume at other than the next sequential location (P+1).
- A link, trace, thread, or some other means by which the new (subordinate) task can return to that next sequential location in the caller when the task is complete.

In HP 1000 computers this is accomplished via the JSB (Jump SuBroutine) instruction. This instruction is used in conjunction with the JMP xxxxx,I (JuMP Indirect) in subroutines. It meets both requirements by performing two functions when executed:

1. It stores the contents of the P register +1 (address of the next sequential instruction) into the 16 bit memory word at the operand (destination). This allows the subroutine, by examination of its own entry point, to determine where it is to return upon completion.

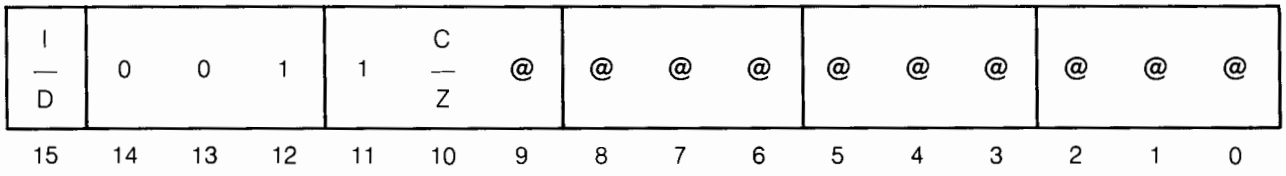
2. It loads the P register with the address of the next memory location after the destination, thus transfering execution to the subroutine, i.e. it jumps to operand (destination) +1. The requirements for subroutine implementation can also be met by the HP 1000 instructions:
   JLY (Jump and Load Y)
   DJS (Disable mapping and Jump to Subroutine)
   SJS (enable System map and Jump to Subroutine)
   UJS (enable User map and Jump to Subroutine)

The JLY is only available on HP 1000 M,E and F-series computers, and the DJS, SJS, and UJS are only available on these same computers when the Dynamic Mapping System (DMS) is installed. These instructions, even when present, have only limited application in user programs, and will not be discussed in this manual.

2-2. The Machine Instruction

This section will discuss the bit structure of the JSB machine instruction. Understanding the JSB at this level of detail is not essential even for the Assembly programmer, but is suggested both to enhance general understanding of subroutines and to guide the user in decoding DEBUG/DBUGR output.

JSB — JUMP TO SUBROUTINE



Where:

I = Indirect if set (1).
D = Direct if clear (0).
0 = Bit always clear.
1 = Bit always set.
C = Current page if set.
Z = Zero (base) page if clear.
@ = Address within page (0000B -to- 1777B).

Bits 11 through 14 are always set to 0011 binary, and tell the computer that this is a JSB instruction.

Bits 00 through 09 are the operand or destination address (or link) and are used by the computer to determine where to store the return address, and where to transfer control. The address denoted by these bits is not necessarily the final operand address. Their significance is a function of the state of bits 10 and 15. Note that the use of only 10 bits restricts us to a destination address range of 1777 octal words (1 page) even though we are probably in a logical address space of 32 pages (77777B locations).

Bit 10 (or Z/C bit) denotes whether the address in bits 00 thru 09 is in the base page (page 0) or the current page. The current page is the page on which the instruction resides. For example, if the sample instructions are in memory location 043635B, then:

Instruction = 014061B = JSB,Z 0061B. Which means JSB to base page location 0061B, which is memory location 000061B.

Instruction = 016061B = JSB,C 0061B. Which means JSB to current page location 0061B, which is memory location 042061B.

Note that this additional page bit has added only one page to the number of locations we can access with the JSB.

To avoid confusion in the following paragraph, remember that the 1st through 16th bits in a word are named bit 00 through bit 15.

Bit 15 (or D/I bit) denotes whether the address (denoted by bits 00 -to- 09, plus 10) is the final address, or the address of the address. When bit 15 is clear (0), our destination address must be in either the base or current pages, as shown in the previous example. However, if bit 15 is set (1), then the destination is **not** assumed to be at the location denoted by bits 00-10. Instead, the destination is assumed to be denoted by bits 00-14 in the memory word at that location. The **indirect** use of these 15 bits now gives us the capability to JSB to any of 77777 octal words in memory (all 32 pages). Moreover, if bit 15 is set in this 15 bit address, we will not use bits 00-14 as the destination address, but as an indirect pointer to yet another memory location. This chain of indirection will continue until the firmware of the JSB instruction points to a memory location containing an address in which bit 15 is **clear**, which location will thus point to the true subroutine address.

In practice, there are seldom more than one or two levels of indirect.

Examples of indirection are:

Instruction = 114061B, and location 000061B contains 052177B: This is an I/JSB/Z 0061B (or simply JSB 61B,I), and since 000061B contains a direct address, the effective instruction is a JSB 52177B.

Instruction = 116061B, and location 042061B contains 142061B: This is an I/JSB/C 0061B (or JSB 42061B,I), and since 042061B contains an INDIRECT address, we have to obtain yet another address. However, in this case the address is **again** 42061B! This condition is referred to as an "infinite indirect", and is a coding error. This instruction would never terminate, and the program would have to be aborted.

2-3. The Symbolic Instruction

The format of the symbolic (Assembly level) instruction is:

[lab] JSB <sub> [,I] [ASCII comment preceded by at least 1 space.]

Where: [lab] is an optional label which will allow symbolic reference to the memory location containing the JSB.

<sub> must be a symbolic name which is defined elsewhere in the calling routine. This means that the same symbol must appear either in the label field of a statement within the caller, or must appear in a pseudo-instruction, such as EXT, MIC, RPL, ABS, EQU, etc.

[,I] indicates that indirection may be explicitly invoked by the user. Note: the absence of ,I does **not** insure that the assembled and loaded JSB instruction will **not** have bit 15 set. For example, if the operand <sub> does not reside on the current page, the RTE LOADR will always generate a link, resulting in bit 15 being set in the JSB. Note also that in such a case, user invocation of indirect would result in bit 15 of the link being set as well.

Note

At the assembly level the user does not have explicit control over the Z/C bit. Control of this bit is reserved for use by the loading program, i.e. RTxGN or LOADR.

2-4. The Effect of the JSB Instruction

This section will display the contents of memory during the execution of a typical JSB instruction. Symbolically the code is:

```
            •••
            JSB SUBRU   Invoke the routine
            •••         Return point
            •••
SUBRU       MPX         Subroutine entry point/return address.
            •••         First instruction in subroutine.
```

Let us assume that the JSB SUBRU is loaded in memory at location 34067B, and that routine SUBRU is at 44521B. Thus, prior to execution of the JSB, the P register is set to 34067B, and memory appears as follows (all in octal):

Loc.	Contents		Comment
00135	044521		Base-page link created by LOADR, and pointing at the destination address.
00136	•••		
	•••		
34066	•••		
34067	114135	<P	The JSB. Note that a link was required, so the instruction is indirect through page 0.
34070	•••		
	•••		
44521	102000		Subroutine entry point (and trapping halt).
44526	•••		First instruction of subroutine.

When the JSB has been executed, the P register will be set to the destination address +1, and the entry point will contain the return address. Memory will appear as follows-

00135	044521		Base-page link.
00136	•••		
	•••		
34066	•••		
34067	114135		The JSB.
34070	•••		
	•••		
44521	034070		Note! Entry point now set to return address.
44526	•••	<P	Execution resumes here.

The 102000B halt which had been assembled into the entry point has been over-written by the JSB. This memory location now contains the address to which we must return on subroutine completion.

Note that the contents of this location will remain unchanged until the subroutine is called again from somewhere else in the program. Of course, the programmer may choose to explicitly reset the entry point contents.

IMAGE, DATACAP AND SECURE DATA RETRIEVAL

Martin Phillips/DSD Technical Marketing

Both DATACAP/1000 and IMAGE/1000 are useful data tools. DATACAP offers real-time datacapture in an easily configured and managed subsystem; IMAGE provides information storage and retrieval in an equally convenient package. Together they comprise the foundation of a powerful Information System.

To play a useful role in decision-making, data must be captured, stored, and retrieved while still relevant. Additionally, the integrity of the data must be maintained. DATACAP and IMAGE may help achieve these goals, but the interface between them must be carefully made. If it is not, capabilities of each subsystem may be lost. Not only can this loss be avoided, but, properly used, DATACAP can provide improved on-line data retrieval and enhanced database security.

In a typical application, datacapture terminals are located throughout a plant at locations where data is generated. During the day, this data is stored directly into an IMAGE database via DATACAP. At the end of the day, operations management reports are generated with QUERY, the IMAGE interactive query language. In this context DATACAP functions effectively as a datacollection tool, while IMAGE performs the data storage and retrieval functions.

A need for local on-line data retrieval may exist. That is, specific information may be needed at particular plant locations to aid in decision-making. As an example, consider a Rework Engineer who must know whether a Write-Only Memory chip (WOM) has passed a previous test. In a non-DATACAP installation, QUERY might let the engineer examine the database interactively via an HP 2621 Display Terminal. However, this would imply that each location at which on-line data retrieval was necessary would require access to a 2621 terminal. Additionally, to effectively query the database, the rework engineer would have to know a good deal about its logical structure and naming conventions. At the very least, he would have to know which item and set names to specify. The expense of additional display terminals, together with the time and expense of providing each user with a personal view of the database, are significant limitations.

Given that DATACAP is already being used for data entry, and that datacapture terminals are located throughout the plant, DATACAP offers an effective data retrieval alternative. Why not design a transaction that displays the required information from an existing datacapture terminal? Not only would the need for an additional 2621 terminal be removed, but the engineer need have no knowledge of database structure — or even that a database exists. The interaction might look like this (user input underlined):

```
***** T.      1 *****
          11
  791031     15:42

WOM NUMBER?
                45488391

TEST =
                      FAIL
```

Alright so far, but suppose that, after learning that the WOM has failed, the engineer re-tests and, surprisingly, it passes. In other words, the data in the database for this WOM is in error (of course, the second test may actually be in error, but let us assume that the WOM is indeed good). The engineer would now like to modify the database by changing the "FAIL" entry to "PASS" for that WOM. This is easily done by adding update capability to the transaction. After the current test result is displayed, the engineer could key in a new result; or he could not key in anything at all, in which case the current value would remain unchanged.

Alternatively, could this be done outside of DATACAP? That is, using QUERY and a 2621 terminal, could the database be updated? The answer is NO, not while datacollection is being done. DATACAP opens the database exclusively: it may be read from but not modified while DATACAP is running. Thus DATACAP is not only a valid data retrieval choice, it is the only one in which database information can be entered, displayed, and modified concurrently.

If, on the other hand, data is to be retrieved when datacollection is not being done, QUERY may be preferable to DATACAP transactions. Since a specific transaction must be designed for each database query or group of queries, if the same query is not made frequently, transaction development effort will be wasted. In this case QUERY would be a better choice for its greater flexibility. DATACAP transactions should be used when database accessing is of a repetitive nature or when operator training is a limitation. In the manufacturing environment typically requiring DATACAP, this is often the case.

There are other benefits to using DATACAP transactions rather than QUERY for interactive information recall. As mentioned, a plant will likely have many more datacapture terminals than display terminals —- hence, data will be available at more locations. This alone may be sufficient to recommend DATACAP.

A less obvious advantage is increased database security. IMAGE security falls into two major categories: security within legitimate accessing techniques and security outside those techniques. The former we shall call logical security; it is the protection found either in QUERY or in embedded programming language calls to IMAGE which prevent an unqualified user from accessing the database. To perform a query, a user must supply the FMP security code for the database file and a valid level code word (a measure of his read and write capabilities). Assuming that these do offer sufficient logical security, what of physical security, that is, the security of the database in its stored form? Since IMAGE databases are stored only in FMP files, our question reduces to: how secure is a FMP file under RTE IVB?

Consider the following scenario. The Rework Engineer at the WOM plant (who also happens to know a great deal about RTE) goes on vacation to Poughkeepsie. There he is kidnapped, brainwashed, and becomes an agent for THE OTHER SIDE. Returning to work, he is ordered to discover the date and carrier name for the next shipment of WOM's. THE OTHER SIDE plans to hijack and steal the WOM's, since they do not have the technology to manufacture WOM's themselves.

During lunch the engineer uses his 2621 Display Terminal — normally used for data retrieval via QUERY — to find the names of the database root and set files. Using his knowledge of RTE, he determines the security codes of the individual files (a feasible operation, though we shall not detail it here), lists them on his terminal, reads the system-supplied Ascii translations and discovers the shipping date and carrier.

Although this is a dramatization, the fact remains that IMAGE security is no greater than RTE security — and an RTE-knowledgeable user can obtain access to FMP files despite the security codes assigned to them. Session Monitor might improve this situation, but DATACAP may be run under Multi-Terminal Monitor, which does not have the logon and command capability level limitations of Session. So the problem is this: how do we guarantee database security using MTM to support IMAGE and DATACAP?

The answer, as you may have guessed, lies in proper use of DATACAP. Throughout the plant all data entry and retrieval should be performed via datacapture terminals. The system console and any 2621 Display Terminals should be made physically secure: for example, by keeping them in a locked room. Since the Working Set of transactions may be carefully controlled, illicit database access is avoided and, since communication with the system is limited to transaction replies, the operating system and database are further isolated from database users. Thus we add a new layer of protection to our system while still offering controlled database access.

To further improve logical security, we can include security-oriented User Subroutines in the DATACAP transactions. The possibilities here are limited only by the imagination and ingenuity of transaction designers — a wide variety of security techniques may be implemented. For example, an answer validation subroutine could check the system time to be sure that access is occurring during legal hours. Or a combination of a display subroutine and a validation subroutine could be used to

output a random Ascii string and make the operator permute it according to some prearranged algorithm. The advantage here is that even if another user saw both the displayed string and the permuted result, he still would not know the operating algorithm. For example, the rule could be "select only odd digits, and reverse their order." A valid transaction would be:

```
***** T.      1 *****
          11
  791031      15:43

SECURITY:       374591
                   943

WOM NUMBER?
            .
            .
            .
```

Is the rule readily identifiable? If so, one more complex could be implemented.

A listing of a transaction and its associated User Subroutine pair accompany this article. They generate a quasi-random string (actually the most rapidly changing components of the system time), display it on the datacapture terminal, verify that the operator's answer is indeed the correct permutation, and check that access is occurring between 8 am and 5 pm. The relative strengths of these methods, as well as others, are documented in a wealth of database security literature.

In our search for data that is both current and accurate, we see that DATACAP/1000 and IMAGE/1000 are basic components of a successful Information System. By expanding the role of DATACAP beyond datacapture into data retrival, reduced peripheral cost and increased database security may be realized.

```
0001  FTN4,L
0002        SUBROUTINE VALD
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C  DATACAP SUBROUTINE THAT, WITH DISPL, GIVES ADDED SECURITY.     C
0005  C  COMPARES OPERATOR ANSWER IN IBUF TO A PERMUTATION OF THE       C
0006  C  DISPLAYED STRING FOUND IN ITEMP. ALSO CHECKS THAT TIME IS      C
0007  C  BETWEEN 8 AM AND 5 PM. IGETB & CMPW ARE UTILITY ROUTINES       C
0008  C  FOUND IN %UTLB4, A LIBRARY OF PLUS/1000, THE USER              C
0009  C  CONTRIBUTED LIBRARY.
0010  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0011  C
0012  C  STANDARD USER SUBROUTINE DECLARATIONS AND CALL TO TMDFN
0013  C
0014        LOGICAL BKSFL
0015        COMMON KEEP1(5),LUQ,LMQ,IBUF(512)
0016        COMMON ITSNU,INDEX,IQNUM,ITMTP,ITMLN,IBUPT,BKSFL,INBKS,IQBKS
0017        COMMON IER,NSTAT,ITEMP(10),ICOMEN
0018  C
0019        DIMENSION ITRUE(3),IOPER(3),ITIME(5)
0020        DATA ISTART/8/,IFIN/17/
0021  C
0022        CALL TMDFN(KEEP1,KEEP1,KEEP1,ITSNU,ITSNU,ICOMEN)
0023  C
0024  C  PERMUTE THE STRING PASSED FROM DISPL AND REFORMAT
0025  C   (ANY OTHER PERMUTATION COULD BE USED)
0026  C
0027        ITRUE(1) = IGETB(ITEMP,5)
0028        ITRUE(2) = IGETB(ITEMP,3)
0029        ITRUE(3) = IGETB(ITEMP,1)
0030  C
0031  C  GET OPERATOR ANSWER IN COMPARABLE FORMAT
0032  C
0033        IOPER(1) = IGETB(IBUF(IBUPT),1)
0034        IOPER(2) = IGETB(IBUF(IBUPT),2)
0035        IOPER(3) = IGETB(IBUF(IBUPT+1),1)
0036  C
0037  C  COMPARE USER ANSWER TO STRING PASSED FROM DISPL
0038  C
0039        IF ( CMPW(ITRUE,IOPER,3) ) 25,50
0040  C
0041  C  CHECK TIME OF DAY HERE
0042  C
0043     25 CALL EXEC(11,ITIME)
0044        IF (ITIME(4) .LE. ISTART) GO TO 50
0045        IF (ITIME(4) .GE. IFIN) GO TO 50
0046  C
0047  C  VALIDATION PASSED. RETURN TO TRANSACTION WITH NO ERROR
0048  C
0049        IER=0
0050        RETURN
0051  C
0052  C  VALIDATION HAS NOT OCCURRED. RETURN WITH ERROR
0053  C
0054     50 IER=1
0055        NSTAT=-1
0056        RETURN
0057        END
  FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)
  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00136      COMMON = 00541
```

```
0001  FTN4,L
0002        SUBROUTINE DISPL
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C  DATACAP SUBROUTINE THAT,WITH VALD,INCREASES DATACAP SECURITY.  C
0005  C  A QUASI-RANDOM STRING (WHICH IS ACTUALLY THE ASCII-FORMATTED   C
0006  C  SYSTEM TIME) IS DISPLAYED ON THE DATACAPTURE TERMINAL          C
0007  C  AND PASSED TO THE VALIDATION SUBROUTINE VALD VIA BUFFER TEMP.  C
0008  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009  C
0010  C  STANDARD USER SUBROUTINE DECLARATIONS AND CALL TO TMDFN
0011  C
0012        LOGICAL BKSFL
0013        COMMON KEEP(5),LUQ,LMQ,IBUF(512)
0014        COMMON ITSNU,INDEX,IQNUM,ITMTP,ITMLN,IBUPT,BKSFL,INBKS,IQBKS
0015        COMMON IER,NSTAT,ITEMP(10),ICOMEN
0016  C
0017        DIMENSION ITIME(5)
0018  C
0019        CALL TMDFN(KEEP,KEEP,KEEP,ITSNU,ITSNU,ICOMEN)
0020  C
0021  C  GET SYSTEM TIME IN ITIME ARRAY
0022  C
0023         CALL EXEC(11,ITIME)
0024  C
0025  C  ARE MINUTES ONLY ONE DIGIT? IF SO, MULTIPLY BY 10
0026  C
0027         IF (ITIME(3) .LE. 10) ITIME(3) = ITIME(3) * 10
0028  C
0029  C  PUT ASCII-FORMATTED TICKS,SECONDS AND MINUTES IN ITEMP
0030  C
0031         CALL CODE
0032         WRITE(ITEMP,5) (ITIME(I), I=1,3)
0033      5  FORMAT(3I2)
0034  C
0035  C  PUT THIS QUASI-RANDOM STRING IN THE DISPLAY BUFFER
0036  C
0037        CALL MOVEW (ITEMP,IBUF(IBUPT),3)
0038        RETURN
0039        END
  FTN4 COMPILER: HP92060-16092 REV. 1926 (790430)
  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00080      COMMON = 00541
```

```
                TRANSACTION SPECIFICATION GENERATOR LIST
               ******************************************
FROM LIBRARY    :  MARLBY (CR =1918  )
NAME            :  COMUN6
NUMBER          :     6
SECURITY CODE   :       0
IMAGE DATA BASE: WOM
SPECIAL FUNCTION KEYS ASSIGNMENT :
**********************************
KEY#    NORMAL VALUE/FUNCTION     PREFIXED VALUE/FUNCTION    TERMINATOR ?
 01      "PASS              "                                     YES
 02      "FAIL              "                                     YES
 03                                                               NO
 04                                                               NO
 05                                                               NO
 06         ABORT/SELECT                                          YES
 07           CONTINUE                                            YES
 08          SAME VALUE                                           YES
 09            RECALL                                             YES
 10         CMPLETE/ABRT                                          YES


                TRANSACTION SPECIFICATION GENERATOR LIST
               ******************************************
 2  U QUESTIONS :
*****************
  QUESTION LABEL :  SECURITY
  ---------------------------
      - DISPLAYED INFORMATION :
                         LIGHT # : NONE
                            TYPE : STRING  (LENGTH =   6)
                  DISPLAY MODULE : DISPL
                     PRINT VALUE : NO
          DATA OFFSET IN BUFFER :    9
      - ANSWER DEFINITION :
                           INPUT : KEYBOARD
                         LIGHT # : NONE
                            TYPE : STRING  (LENGTH =   3)
                    POSITIONING : L
                    EDIT MODULE : VALD
                  DEFAULT VALUE :
                   PRINT ANSWER : NO
          DATA OFFSET IN BUFFER :   12
  QUESTION LABEL :  ROM NUMBER
  ----------------------------
      - ANSWER DEFINITION :
                           INPUT : KEYBOARD
                         LIGHT # : 1
                            TYPE : STRING  (LENGTH =   6)
                IMAGE ITEM NAME : WOM#    (FUNCTION : F)
                        DATA SET : DETAIL
            IMAGE EDIT GENERATED : CHECK CHAIN NON EMPTY
                    POSITIONING : L
                           MASK : 99999A
                  DEFAULT VALUE :
                   PRINT ANSWER : NO
          DATA OFFSET IN BUFFER :   14
  * LENGTH OF STORAGE FOR U QUESTIONS SEQUENCE :   16
```

```
                TRANSACTION SPECIFICATION GENERATOR LIST
               ******************************************
 1  M QUESTIONS :
*****************
  QUESTION LABEL :  TEST RESULT
  -----------------------------
      - DISPLAYED INFORMATION :
                         LIGHT # : NONE
                            TYPE : STRING  (LENGTH =    4)
                 IMAGE ITEM NAME : TEST
                        DATA SET : DETAIL
                    PRINT VALUE : NO
           DATA OFFSET IN BUFFER :   17
      - ANSWER DEFINITION :
                           INPUT : KEYBOARD
                         LIGHT # : 2
                            TYPE : STRING  (LENGTH =    4)
                 IMAGE ITEM NAME : TEST    (FUNCTION : U)
                        DATA SET : DETAIL
                     POSITIONING : L
                            MASK : AAAA
                   DEFAULT VALUE : DISPLAYED VALUE
                    PRINT ANSWER : NO
           DATA OFFSET IN BUFFER :   19
  * LENGTH OF STORAGE FOR M QUESTIONS SEQUENCE :    4
               ******************************************
```

TIME AND ATTENDANCE DATACAPTURE WITH THE HP 1000

Darrell Krulce/DSD Sales Development

INTRODUCTION

DATACAP/1000 is an extensive software package for the HP 1000 computer that allows the user easy control over data collection from the HP 3077A data collection terminal as well as the other Hewlett-Packard data collection terminals, the 3075A, and 3076A.

DATACAP/1000 is an excellent tool for data collection when the data collection rate is in the range of up to 600 to 1000 transactions per hour (rate depends on the number of data base accesses, whether or not magnetic tape logging is utilized, etc.). However, a typical time and attendance application might consist of 600 people clocking in and out at shift change in a matter of 15 minutes. This translates to a transaction rate of 2400 transactions per hour! Clearly, something other than DATACAP/1000 would need to be utilized in such a situation.

The rest of this article describes a computer program, TMATT, that has been written for the HP 1000 to facilitate high speed data input from HP 3077A time reporting data collection terminals. Steve Witten of Data Systems Lab wrote the original version of TMATT as a guide for the user to show how a time and attendance application program might be written. It is expected that the user would want to make modifications to the program for his specific time and attendance applications. The version in the User Contributed Library already contains several changes that I made to the original.

PROGRAM PHILOSOPHY

In order for the time and attendance data collection system to be very fast and responsive, the program that runs the system needs to be able to read data from several terminals simultaneously and very quickly. This is done in TMATT by the use of class I/O. Class reads, utilizing a single class number, are placed on all terminals. The program then executes a class get with wait and waits for a terminal to respond. In this way, as soon as badge is inserted into any 3077A terminal controlled by TMATT, the system completes the read of that terminal, and TMATT is awakened from its class wait to process the information. The program priority of TMATT is set to 25 to ensure that when input is received from a terminal, it can be immediately processed and a class read reissued to that terminal.

The processing of the information by TMATT is kept very simple to insure high speed data throughput. The data is simply logged out to disc for processing later, after everybody has clocked in or out. It is intended that the storage routine be modified for the specific need of the user. Suggestions for data storage are contained in the section "Enhancements to TMATT."

TMATT runs as a program separate from the DATACAP/1000 system. For this reason, the 3077A terminals under the control of TMATT cannot simultaneously be under the control of DATACAP/1000. However, terminals under the control of both systems can be placed on the same data link.

PROGRAM INTERNALS

Refer to the flowchart of TMATT (figure 1) for the following discussion.

Three configuration strings are created to set up the HP 3077A's. The "full" configuration string consists of a complete configuration of the 3077A including writing out a user-defined welcome message, the proper system time, and setting the terminal to read an input. This configuration string is used when first setting up the terminal, when the terminal is powered up during TMATT operation, and whenever a data validation error occurs. The "simple read" configuration string resets the terminal to accept another badge input. It is used after previous data has been entered and stored (normal conditions) and

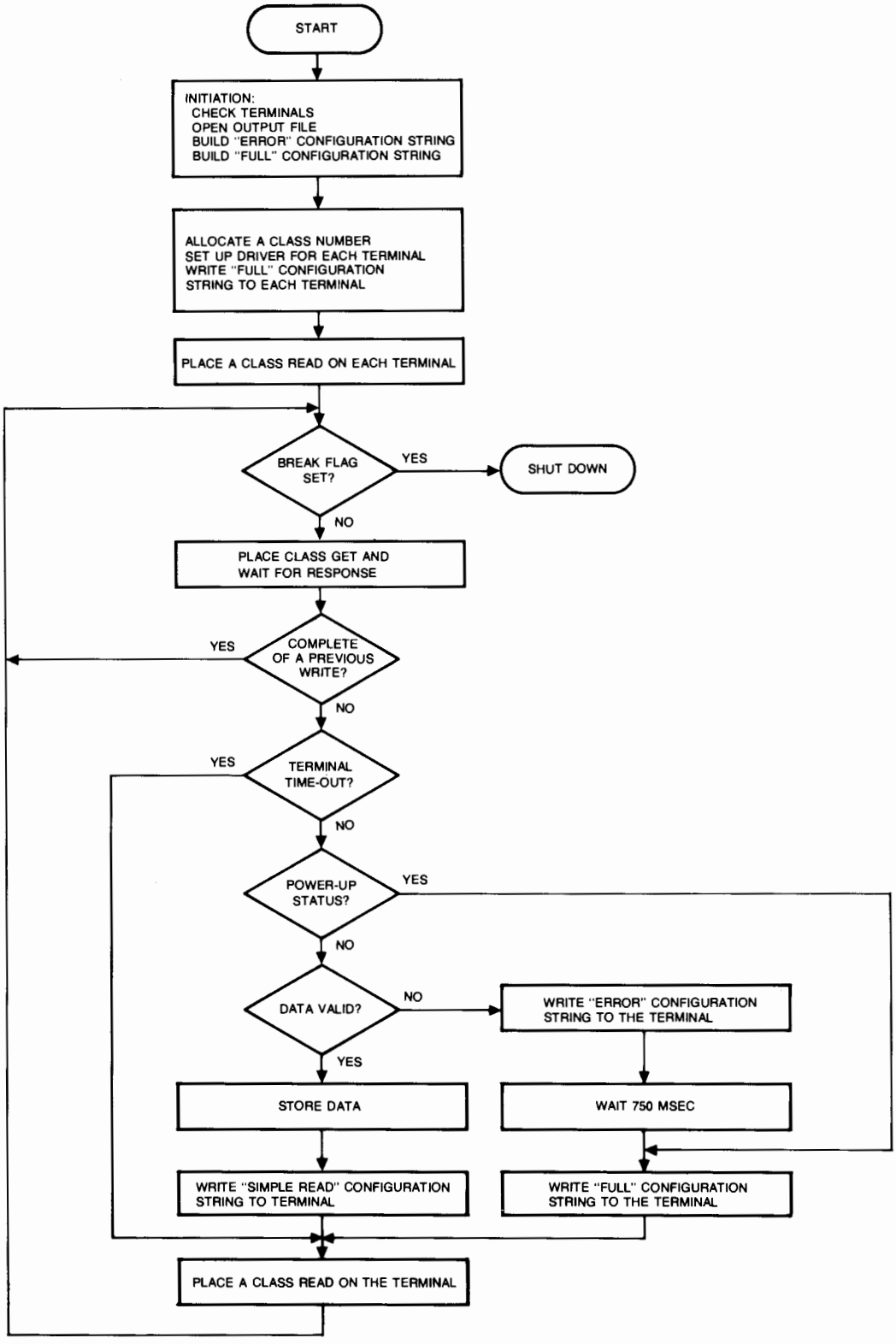

START
INITIATION:
CHECK TERMINALS
OPEN OUTPUT FILE
BUILD "ERROR" CONFIGURATION STRING
BUILD "FULL" CONFIGURATION STRING
ALLOCATE A CLASS NUMBER
SET UP DRIVER FOR EACH TERMINAL
WRITE "FULL" CONFIGURATION
STRING TO EACH TERMINAL
PLACE A CLASS READ ON EACH TERMINAL
BREAK FLAG SET?
YES
SHUT DOWN
NO
PLACE CLASS GET AND
WAIT FOR RESPONSE
YES
COMPLETE OF A PREVIOUS WRITE?
NO
YES
TERMINAL TIME-OUT?
NO
POWER-UP STATUS?
YES
NO
DATA VALID?
NO
WRITE "ERROR" CONFIGURATION
STRING TO THE TERMINAL
YES
STORE DATA
WAIT 750 MSEC
WRITE "SIMPLE READ" CONFIGURATION
STRING TO TERMINAL
WRITE "FULL" CONFIGURATION
STRING TO THE TERMINAL
PLACE A CLASS READ ON THE TERMINAL

Figure 1

saves any overhead involved in transmitting unneeded data to a terminal. The "error" configuration string is used to inform the employee of data validation errors. It consists of an error message that is displayed along with lighting the red light and sounding the buzzer.

When TMATT is first started, it checks all terminals that it is to collect data from, and makes sure that they are indeed 3077A's and that they are up and running. TMATT then interactively requests information from the operator regarding where the data is to be stored (file name, security code, cartridge number) and opens that file. After this, the "full" configuration string is written to each terminal and a class read is placed on the terminals. All of this is done using a single class number through which all I/O is channeled. This completes TMATT's initiation phase after which TMATT goes into its data collection loop.

The first step of the loop is to check to see if the break flag is set. A set break flag indicates that the operator wants to shut down the system so TMATT will execute clean up operations such as flushing class I/O and closing the output file.

After checking the break flag, TMATT then goes into a class get wait state to wait for some I/O activity to occur with the terminals. Whenever a class get is completed on the class number, TMATT is awakened to process it. If the class get is the complete of a class write, TMATT simply returns back to the top of the loop. If the class get is the complete of a class read, TMATT examines how much data was returned to determine whether the terminal timed out or not. A zero length data record returned indicates a time-out so TMATT reissues the class read on that terminal and goes back to the top of the loop. If data was indeed returned from the terminal, TMATT checks to see if the returned data indicates that the terminal was just powered-up (assumed if reverse interrupt is received). A power-up status is responded to by writing out the "full" configuration string to the terminal, reissuing a class read to it and returning to the top of the loop.

If none of these checks are met, the data returned is processed. An optional user written validation routine is executed to determine the validity of the received data. If the data is not valid, the "error" configuration string is written to the terminal followed 750 milliseconds later with the write at the "full" configuration string. The 750 millisecond wait gives the employee time to read the error message. If the data is valid, the storage routine is executed to store the data and then the "simple read" configuration string is issued to the terminal followed by a return to the top of the loop.

PERFORMANCE

The performance of TMATT under different load conditions was measured using a program that simulates 307X terminal transactions. The measurements were made under the following assumed worst case conditions:

1. All terminals attached to a single multipoint line.

2. As soon as one transaction was completed at a terminal, the next one was started immediately with no pause. This is to simulate a worst case queue at a time clock since realistically there would be some time pause between transactions as an employee inserts his badge.

3. All terminals active simultaneously. This was to simulate everybody in a facility clocking out at the same time through all of the 3077A's.

Under these conditions, TMATT performed quite well. Referring to figure 2, we see that the maximum transaction rate that can be expected from the system approaches 600 transactions per minute with 20 terminals on a single multipoint line. Figure 3 derives the transaction rate per terminal from figure 2 and as expected, it decreases as the number of terminals increases.

Figure 4 is the worst case response time that could be expected from a terminal under the conditions of all terminals simultaneously and continuously processing transactions (4:30 PM on Friday!). With 10 terminals each processing 42 transactions per minute we would expect a worst case response time of only 1 second — adequate performance considering it takes about that long just to insert/remove a badge and get out of the way for the next person.

Figure 5 shows that worst-case TMATT CPU utilization. At most, TMATT will only consume 10% of the CPU, excluding the time the system uses for the multipoint card — a maximum of 10%.

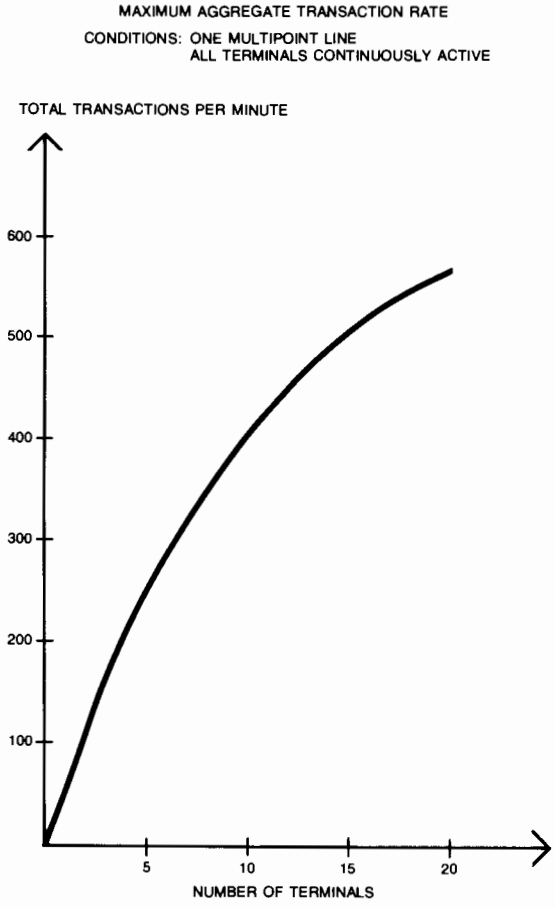


Figure 2

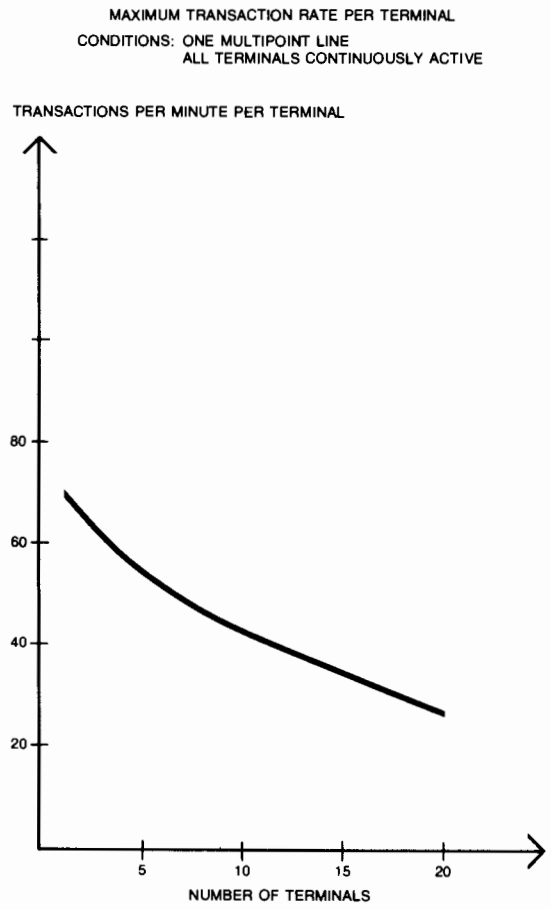


Figure 3

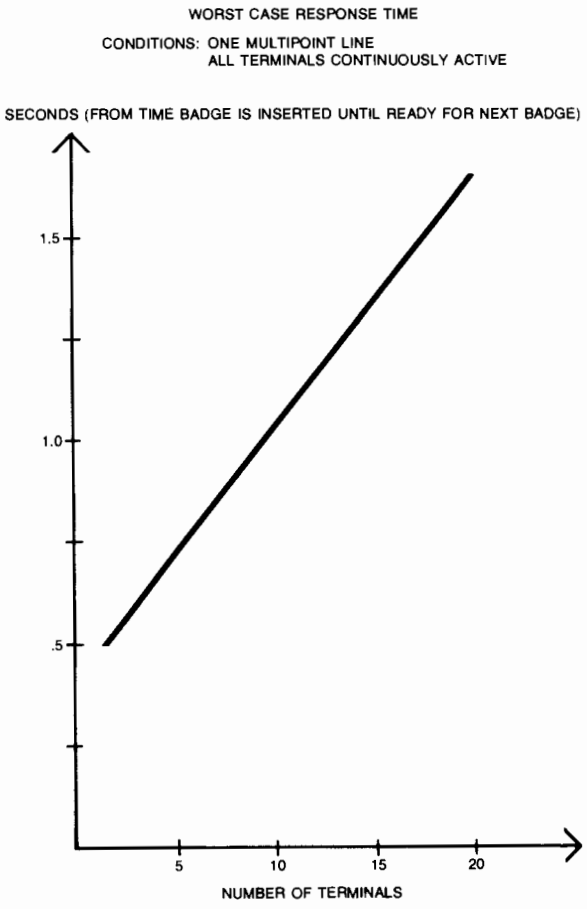


Figure 4

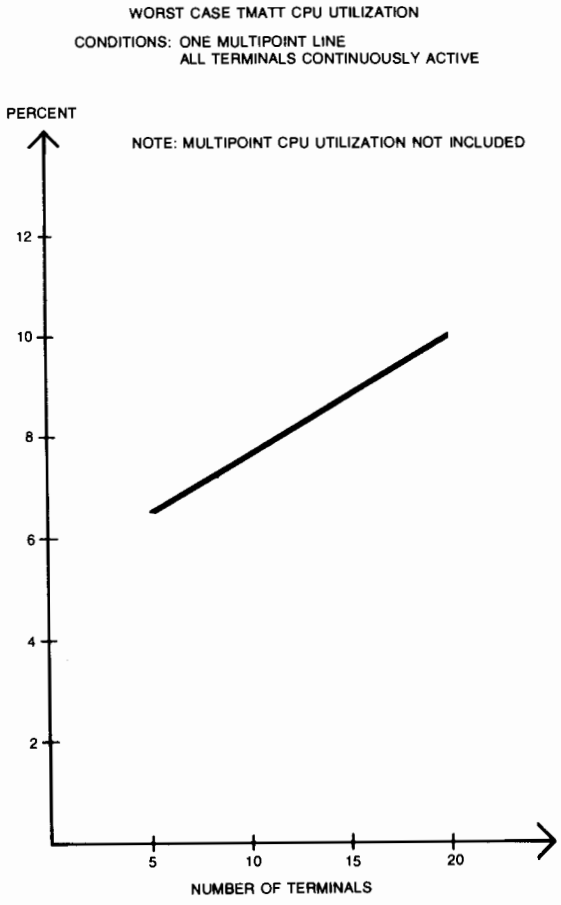


Figure 5

ENHANCEMENTS TO TMATT

The following are suggestions on how the user could customize TMATT to his specific application:

1. If no data validation needs to be done, the internal buffering capability of the 3077A can be utilized. The 3077A has the capability of storing up to 20 data records before it needs to transmit them to the HP 1000. By using this feature, instantaneous terminal response could be achieved even if there were many terminals in the system. The drawbacks are that no validation could be done on the incoming data, as well as the danger of possible loss of data.

2. The program, as written, does nothing with the data except store it in a serial file. Another method of storing the information would be to keep it in an IMAGE/1000 data base. However, the overhead of data base manipulations would probably decrease the responsiveness of the program. This problem could be easily circumvented by writing another program to do the IMAGE/1000 manipulation. TMATT could pass the time and attendance information to the other program through the use of class I/O and by setting the priority of the other program lower than TMATT, the data base manipulation could be done when TMATT is not running thereby making the most efficient use of the system's resources. This other program could also do some simple data reduction such as calculating the number of hours each employee has worked. To eliminate fears of data "lost" in SAM as the result of a system failure, the second program could be written to read the disc file created by TMATT instead of using class I/O. Another possibility would be to have the program pass the data on to an HP 3000 via DS/1000 — 3000 or to a mainframe computer via RJE/1000.

   In cases where the data integrity is highly important, the data could be logged to a mag tape as the data is entered. Again, this would slow down the response of TMATT but the performance degradation might be insignificant for the specific application.

3. TMATT could also be modified to accurately synchronize the 3077A terminals with the system clock. Since the 3077A can only be set to hours and minutes, this could be done by modifying TMATT to wait until system clock reaches 0 seconds before setting the time on the terminals. In addition, TMATT could periodically check and, if required, change the time displayed by the 3077A's.

4. TMATT has the capability to accept a data validation routine. A possible data validation would be to verify that the time received from the 3077A terminal matches the system time. To do extensive IMAGE/1000 data base validation checking would probably warrant the use of DATACAP/1000.

CONCLUSION

TMATT is an example program to help you implement a time and attendance datacapture system. The terminal handling, data validation interface and data storage interface has been done. It is now up to the user to customize the program for his specific application with regard to data validation, data reduction and data storage. TMATT is available in the PLUS/1000 contributed library.

PENNY: COMPUTER AIDED DRAWING ON THE HP 1000

Phil Walden/DSD Applications Development

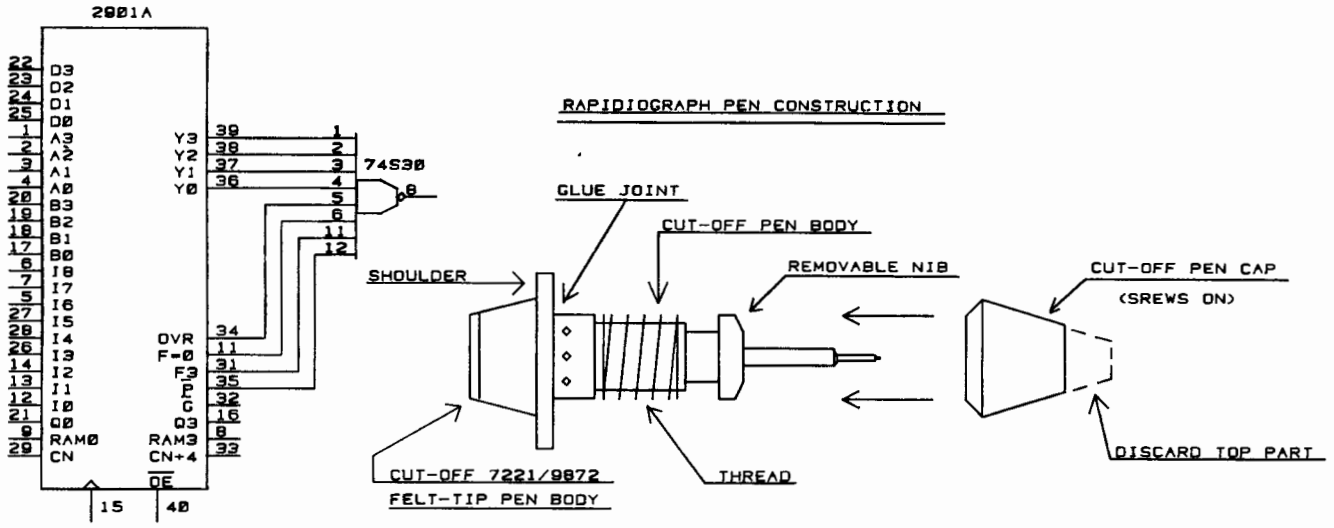


Figure 1. A PENNY Drawing

INTRODUCTION

In recent years, there has been a general recognition of the need to increase the productivity of the engineering design process. Millions of dollars could be saved on large projects by shortening the R&D phase by a few months. Doubling the productivity of engineers and designers has an immediate impact on reducing R&D investments.

PENNY is a computer aided drawing (CAD) program that can increase the productivity of engineers and designers. This article will outline the features of PENNY and how it works.

Several features of PENNY improve the productivity of designers. PENNY drawings (see Figure 1) are inherently cleaner than pencil sketches, reducing confusion and enhancing communications between designers. Its powerful hierarchical data base structure allows the designer to use predefined standardized symbols, eliminating drafting and design errors. Mistakes may be corrected without re-drafting. Furthermore, PENNY allows the user to interactively create drawings. With the addition of a drawing tablet to the graphics terminal, the designer can actually draw faster with PENNY than by hand. Most of the figures in this article were drawn with PENNY.

PENNY was originally developed at HP Data Systems Division (DSD) to assist engineers in developing electrical schematic diagrams. It is basically a line drawing program and thus, has been used for other applications such as floor layouts, piping diagrams and flow charting. PENNY uses GRAPHICS/1000, IMAGE/1000 and RTE-IVB's EMA capability to accomplish its tasks. PENNY requires a 101 page mother partition and an HP2648A graphics terminal for interactive design. An F-series processor and a fast disc (7906, 7920 or 7925) should be used for good response. At least one hardcopy plotting device should also be made available to users. Currently, PENNY will operate the 9872A and 7221A four color X-Y plotters. Optionally, a drawing tablet may be connected to the 2648A to increase the drawing speed and ease of using PENNY.

PENNY DRAWINGS: THEORY

PENNY internally models a drawing as a bounded plane with various objects located on it. The bounded plane is called the "world coordinate space". The objects in the world coordinate space may consist of MACRO's, LINES and TEXT. A MACRO is itself a drawing, and may also consist of macro's, lines and text. A DRAWING and a MACRO are therefore semantically equivalent. Using this concept, PENNY internally describes a drawing/macro as a tree data structure, as shown in Figure 2.

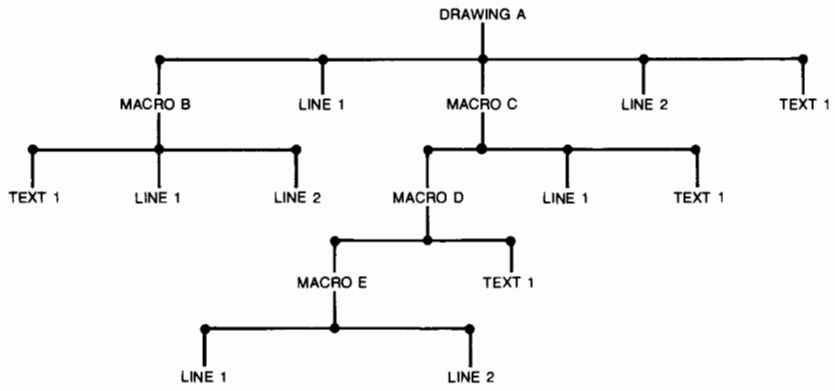


Figure 2. The Hierarchical Structure of a Macro

Thus a macro is a hierarchy of other macros, lines and text entries. This data structure provides PENNY with its most powerful interactive design feature. To create a drawing, the user simply indicates where in the world coordinate space the macro's are to be located and PENNY will automatically link the appropriate tree structure in its IMAGE/1000 data base. If the data base is properly prepared with standard symbols (macros) such as NAND gates or flow chart symbols used by the designer, interactive drawing creation will be very quick because only the upper level of the tree structure is generated. All the lower levels were defined when the data base was prepared, and can be used over and over again by designers.

PENNY uses a linked tree structure in its data base. In other words, each macro node of a tree is unique within the data base and may be an element of one or more trees. When PENNY generates a drawing only pointers to macros are generated, not copies of macros. Therefore, standard symbols (macros) may easily be added, purged or modified because only one copy of any symbol exists in the data base.

A picture is worth a thousand words! Let's examine the structure of a simple macro that a logic designer would use, a NAND gate shown in Figure 3. The macro is given a unique name such as "74S00.AND.1". As seen in Figure 4, five levels are used to describe this macro. Each level may branch off to another macro which also has its own unique name. For example, the macro, "PINS.1,2:3", provides the macro that labels the pinouts on the NAND gate. The macro, "74S00.AND", draws the gate outline with the gate label "74S00". The gate outline is itself a macro called "TWO.NAND.AND", and so on.

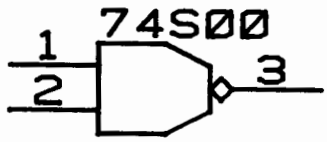


Figure 3. A PENNY Drawn NAND Gate

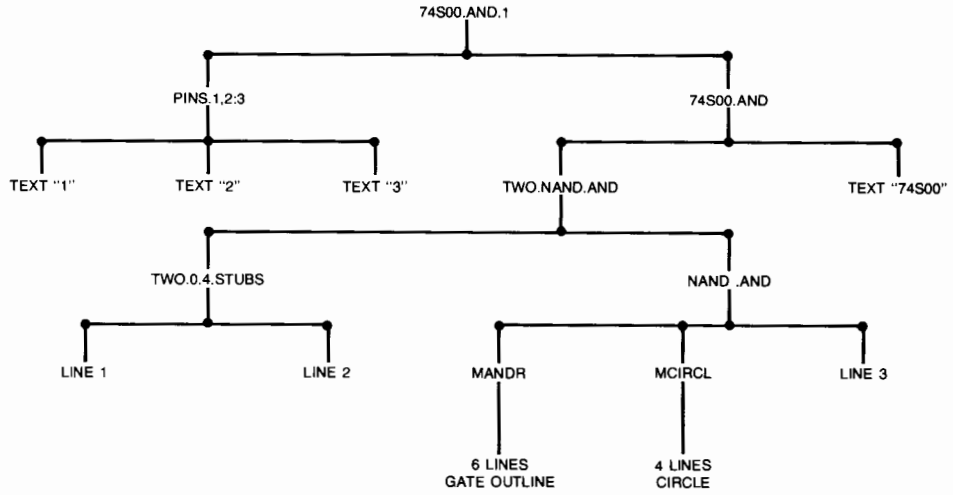


Figure 4. 74S00.AND.1 Macro Example

There is a reason for resorting to many macro levels to describe a simple NAND gate. A much more flexible data base can be developed. For example, if different pinouts are needed, then a new macro called "74S00.AND.2" could be created that points to macros "PINS.4,5:6" and "74S00.AND". Because the macro "74S00.AND" already exists, it need not be re-created, only "PINS.4,5:6" needs to be created in order to develop the new macro "74S00.AND.2".

The macro nodes of a drawing are only intermediate data nodes that do not contain any graphics information. Only the leaves of the tree or terminal nodes containing lines and text hold graphics information. This means that all drawings are ultimately described as so many lines and text. A line is simply two X-Y coordinate pairs and a pen color number. Text is a character string, an X-Y coordinate location and a pen color number. To draw a drawing, PENNY executes a post-order traversal of the drawing tree, visiting all the leaves of a drawing tree and displaying the graphics information. To understand this process a more detailed look at the actual IMAGE data base is needed.

THE PENNY IMAGE/1000 DATA BASE

The IMAGE/1000 data base used by PENNY implements the hierarchical tree structure with one automatic master data set called "DRWGS" and one detailed data set called "LINES". Figure 5, shows the data base structure graphically and the corresponding schema is listed in Table 1.

Table 1. PENNY IMAGE/1000 Data Base Schema

```
$CONTROL: ERRORS=5,ROOT,SET,TABLE,FIELD;
BEGIN DATA BASE: {root namr}
LEVELS:
        5               NOONE;          <<READ LEVEL: NOT USED>>
        15              HELLO;          <<WRITE LEVEL>>
ITEMS:
        DNAME,          X32(5,15);      <<MACRO NAME: 32 CHARACTERS>>
                                        << WITH NO LEADING BLANKS>>
        TYPE,           I1(5,15);       <<TYPE OF INFO IN RECORD:  >>
                                        << -1=MACRO RECORD>>
                                        <<  0=LINE RECORD>>
                                        << +1=TEXT RECORD>>
        INFO,           X32(5,15);      <<INFO:       >>
                                        << 32 CHARACTERS FOR TEXT OR>>
                                        << MACRO NAME; TWO X-Y COORD>>
                                        << PAIRS STORED AS REAL NUMBERS>>
                                        << (8 WORDS TOTAL) FOR LINES>>
        COLOR,          I1(5,15);       <<PEN COLOR: INTEGER 0 TO  4>>
        POSX,           R2(5,15);       <<LOCAL ORIGIN X>>
        POSY,           R2(5,15);       <<LOCAL ORIGIN Y>>
SETS:
      NAME:DRWGS::{crn},A;          <<AUTOMATIC MASTER SET>>
      ENTRY:        DNAME(1);       <<KEY: MACRO NAME>>
      CAPACITY:{prime number};
      NAME:LINES::{crn},D;          <<DETAIL DATA SET>>
      ENTRY:        DNAME(DRWGS),   <<LINK: MACRO NAME>>
                    TYPE,
                    INFO,
                    COLOR,
                    POSX,
                    POSY;
        CAPACITY:{large prime number};
END.
```

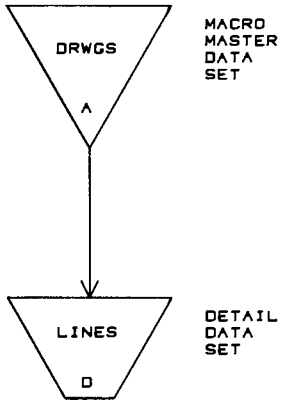


Figure 5. PENNY Data Base Diagram

A drawing/macro is accessed by its name, item "DNAME", which can be up to 32 characters as shown in Table 1. DNAME is the key to the master data set, DRWGS. The hashing algorithm of IMAGE can quickly access the appropriate master record by drawing/macro name. The drawing/macro master record provides PENNY with access to a chain of LINES detail records linked to a specific drawing/macro name.

Each detail data set record in the chain is a son of the corresponding macro node of a drawing tree structure. If the record item TYPE is 0 or +1, then the items INFO, COLOR, POSX and POSY contain graphics information for lines and text as commented in Table 1. However, if TYPE equals −1, then the detail record represents the occurrence of another macro node. INFO will then contain the name of the macro which is used as the key to next level of sons or detail records whose father is this new macro. Figure 6 illustrates the correspondence between the data base and tree structure.

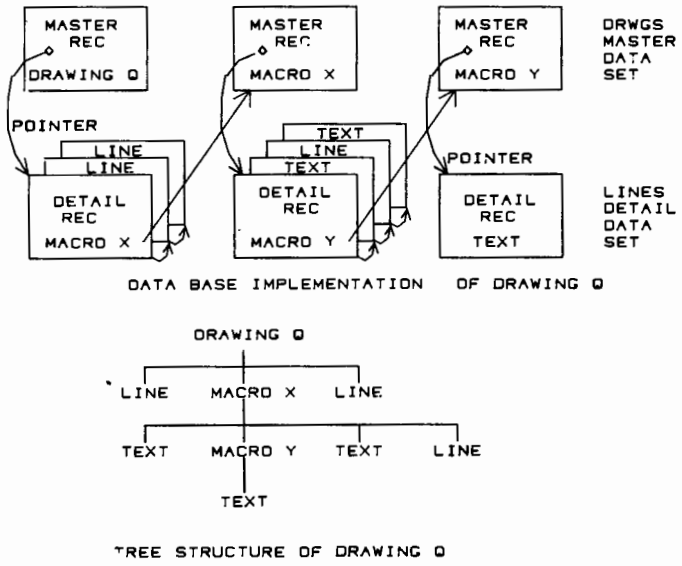


Figure 6. IMAGE/1000 Implementation of a Macro

Let's consider the NAND gate macro "74S00.AND.1" again as an example. Referring to Figures 3, 4 and 7, the macro 74S00.AND.1 has a DRWGS master record that points to a chain of two LINES detail records. These are macro detail records where TYPE = −1. The INFO fields for these records contain the names of the macros which are the sons of 74S00.AND.1. These are the macros "74S00.AND" and "PINS.1,2:3" and they also have their own DRWGS master records. The PINS.1,2:3 master has three detail records chained to it. These are all TYPE +1 records containing the text for the NAND gate pin outs. The 74S00.AND macro master record has only two detail records chained to it. One is another TYPE −1 record for the macro TWO.NAND.AND, and the other is a TYPE +1 text record for the "74S00" label. And so this description could continue on down the remaining three levels of this drawing. Figure 7 illustrates the details of this drawing.

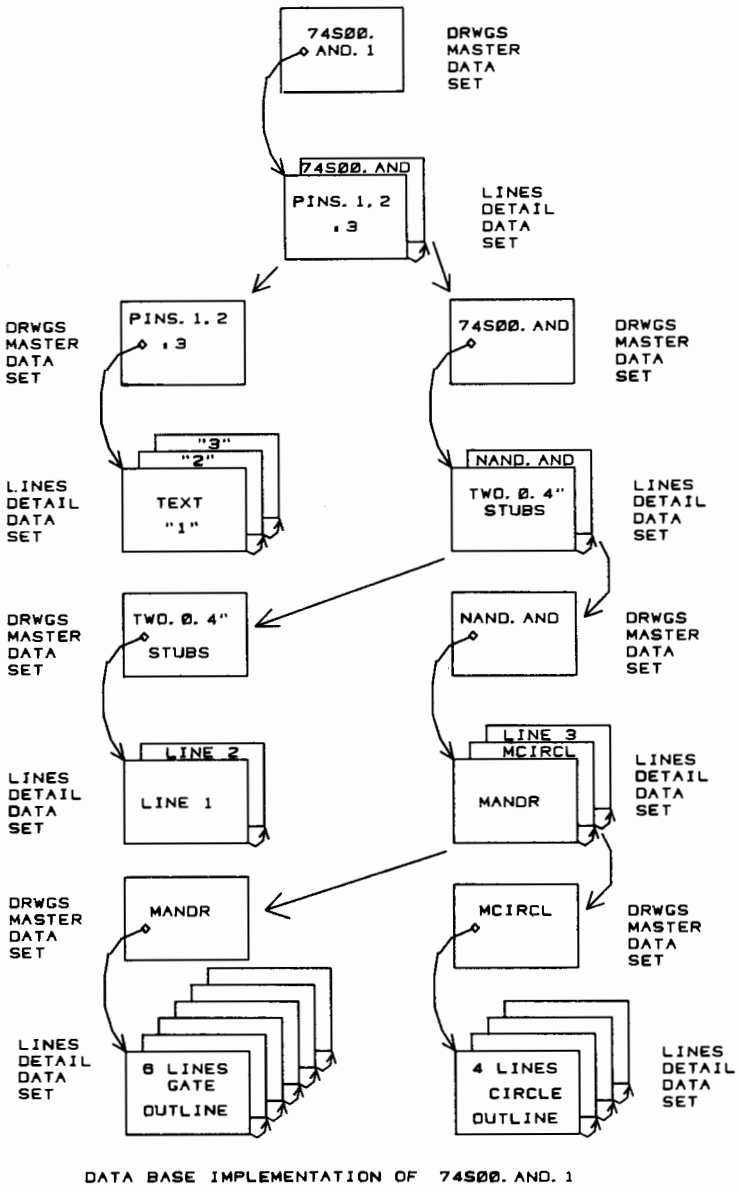


Figure 7. IMAGE/1000 Implementation of the Macro 74S00.AND.1

DRAWING A MACRO

To draw a macro, PENNY executes a subroutine MDRAW. The only parameter MDRAW needs is the macro name (the data base key) for the macro to be drawn. Basically, MDRAW executes a post-order traversal of the drawing tree whose root is the specified macro. The algorithm used by MDRAW is shown in Table 2.

Table 2. Algorithm Used to Draw a Macro

Definitions:

MACRO = name of a macro (data base key)
TYPE = type field in LINES record
INFO = info field in LINES record
STACK = LIFO stack for temporary storage

1. Find the first detail LINES record chained to MACRO

2. Check the TYPE of the LINES record

   If TYPE = +1 Then WRITE TEXT
   GO TO 3.

   If TYPE = 0 Then DRAW LINE
   GO TO 3.

   If TYPE = −1 Then PUSH MACRO IN STACK
   MACRO := INFO
   GO TO 1.

3. Get next LINES record on MACRO chain

4. Check for the end of the chain

   If NOT END OF CHAIN Then GO TO 2.

   If END OF CHAIN Then POP MACRO OFF STACK
   GO TO 3.

The algorithm in Table 2 ensures that every node in the drawing/macro tree is visited and its graphics information is drawn on the graphics device. To implement this algorithm in MDRAW, several GRAPHICS/1000 and IMAGE/1000 calls are required.

For the first step in the algorithm, the first LINES detail record chained to a macro must be found. MDRAW implements this procedure as follows:

```
        CALL DBFND(IBASE,LINES,1,ISTAT,DNAME,MACNAM)
        CHNLEN = ISTAT(6)
10      CALL DBGET(IBASE,LINES,5,ISTAT,LIST,LIBUF,DUMMY)
        CHNLEN = CHNLEN - 1
```

where variables used are defined as follows:

MACNAM	macro name array (16 words)
DNAME	ascii array containing "DNAME " the key item
ISTAT	status array returned by IMAGE, ISTAT(5) and ISTAT(6) contain the double word integer chain length (only least significant half used)
LINES	ascii array containing "LINES " data set name
IBASE	array containing the data base name when opened
LIST	ascii array containing all the item names in the detail record, "DNAME,TYPE,INFO,COLOR, POSX,POSY;"
LIBUF	integer array to receive LINES record
DUMMY	dummy argument for chained read DBGET calls
CHNLEN	integer chain length

To get the next record in the chain, as is done in step 3 of the algorithm, the above procedure is executed from statement 10 on.

Step 2 of the algorithm requires lines and text to be drawn, or macro information to be saved in a stack, depending on the LINES record TYPE value. First, to access the items in LIBUF, the following declarations may be used.

```
      INTEGER NAME(16),TYPE,INFO(16),COLOR
      INTEGER IPOSX(2),IPOSY(2),IXORG(2),IYORG(2)
      REAL    POSX,POSY
      EQUIVALENCE (LIBUF,NAME),(LIBUF(17),TYPE),(LIBUF(18),INFO),
     -       (LIBUF(24),COLOR),(LIBUF(25),POSX),(LIBUF(27),POSY)
      EQUIVALENCE (IPOSX,POSX),(IPOSY,POSY),
     -       (IXORG,XORG),(IYORG,YORG)
```

If the TYPE is +1 then INFO contains text which is drawn with GRAPHICS/1000 calls.

```
    POSX = POSX + XORG              ;calculate the position
    POSY = POSY + YORG              ; of the text.
    CALL PEN(IGCB,COLOR)            ;select the pen color
    CALL MOVE(IGCB,POSX,POSY)       ;position the pen
    CALL LABEL(IGCB)                ;set up for graphics text
    CALL EXEC(2,PLOTLU,INFO,I)      ;write text on display
```

where:

XORG	=	current value of the X origin
YORG	=	current value of the Y origin
IGCB	=	graphics control block for GRAPHICS/1000
PLOTLU	=	logical unit of graphics device
I	=	number of characters in text string

If the TYPE was 0, then a line is drawn using the real X,Y coordinate pairs in INFO.

```
      REAL P1X,P1Y,P2X,P2Y
      EQUIVALENCE (INFO,P1X),(INFO(3),P1Y),(INFO(5),P2X),
     -       (INFO(7),P2Y)

      P1X = P1X + XORG              ;calculate the positions
      P1Y = P1Y + YORG              ; of the to points.
      P2X = P2X + XORG
      P2Y = P2Y + YORG
      CALL PEN(IGCB,COLOR)          ;select the pen color
      CALL MOVE(IGCB,P1X,P1Y)       ;move pen to first point
      CALL DRAW(IGCB,P2X,P2Y)       ;draw line to second point
```

where:

P1X,P1Y	=	first point on line
P2X,P2Y	=	second point on line

If TYPE was −1, then the current IMAGE chain information, origin and chain length must be saved in the STACK before the next macro chain can be followed. The IMAGE DBINF call is used for this purpose.

```
      INTEGER STACK(12,n)          ;stack of n levels
C
C STACK FORMAT: WORD  1-2 CURRENT X ORIGIN
C               WORD  3-4 CURRENT Y ORIGIN
C               WORD 5-11 CURRENT CHAIN INFORMATION
C               WORD   12 REMAINING CHAIN LENGTH
C
      DO 20 I=1,2
      STACK(I,IPTR) = IXORG(I)     ;store the origin
20    STACK(I+2,IPTR) = IYORG(I)
      CALL DBINF(IBASE,LINES,401,ISTAT,STACK(5,IPTR))
      STACK(12,IPTR) = CHNLEN      ;store the chain info and length
      IPTR = IPTR + 1              ;bump the stack pointer
      DO 30 I=1,16
30    MACNAM(I) = INFO(I)          ;set up the new macro name
      XORG = XORG + POSX           ; and origin.
      YORG = YORG + POSY
```

When the next LINES record is fetched with the DBGET call, the chain length counter CHNLEN is checked for zero. If it is zero, then the STACK must be popped to restore the previous chain information, origin and chain length. This procedure also uses the DBINF call.

```
      IPTR = IPTR - 1              ;lower the stack pointer
      DO 40 I=1,2
      IXORG(I) = STACK(I,IPTR)     ;restore origin
40    IYORG(I) = STACK(I+2,IPTR)
      CALL DBINF(IBASE,LINES,402,ISTAT,MSTACK(5,IPTR))
      CHNLEN = STACK(12,IPTR)      ;get chain info and length
```

When the next record on the restored chain is read the macro name is restored from NAME in LIBUF.

That concludes the discussion of the algorithm. However, there are many techniques not discussed which can enhance the speed and power of this algorithm. For example, skipping a pen MOVE call if the pen is already in the correct position, and skipping PEN color calls when the correct color has been previously selected.

TARGETING

Targeting In order to provide an interactive graphics environment for the user, a CAD application program not only must display a drawing, but also must provide the ability to edit drawings. This function implicitly requires the power to delete and move macros, lines and text. To do these functions, the CAD program must remember what and where objects have been drawn on an interactive graphics terminal. Otherwise the program has no clue as to what objects users may be pointing to on their graphics display. Targeting is the process which a CAD program uses to determine what object a user is selecting to edit.

PENNY uses a Target Array to provide the targeting function. The array is a 346 by 220 matrix of integers requiring over 76 pages of memory. Because this array is all data, the EMA capability of RTE-IV makes addressing this large array possible. The use of the Target Array is simple.

First the user positions the graphics cursor at some location on the graphics terminal, hopefully over an object. PENNY then digitizes the cursor's position and reads the cursor's X-Y coordinates. The X and the Y values are used for the index values of the Target Array. The value stored in the array at (X,Y) contains the relative LINES record number of the object in the IMAGE data base. The record number is placed in the array when PENNY draws the object. If no object exists at the cursor's location, a zero record is returned from the Target Array. To delete an object, the data base record number is used to access the data base and undraw the object. The target array is then filled with zeroes at the locations corresponding to the position of the deleted object. A move consists of undrawing the object and then redrawing it at the new location. The relative record number is another part of the target array corresponding to the new position of the macro.

CONCLUSION

Hopefully, this article has given the reader a flavor for the Computer Aided Drawing concepts that PENNY uses. It demonstrates that GRAPHIC/1000, IMAGE/1000 and EMA capabilities can be combined in one application package to enhance the productivity of engineers and designers.

ORDERING

PENNY will be available to all users through the User Contributed Library of the new HP 1000 International Users Group. Contact the:

HP 1000 International Users Group
P.O. Box 1000
Norwood, Massachusetts 02062

ANNOUNCING PLUS/1000: THE NEW CONTRIBUTED LIBRARY FOR HP 1000 SYSTEMS

PLUS/1000 is the new Program Library of User Software for HP 1000 computer systems. PLUS/1000 has been under development within HP's Data Systems Division over the past six months. The new library has resulted from a complete reorganization of the old LOCUS library. The reorganization includes:

— new program classification and extended indexing scheme
— new catalog
— automated submission and update procedures
— machine readable documentation

Many new programs have been added since the last LOCUS catalog was printed.

Starting January 1, 1980, PLUS/1000 will be the responsibility of th new **HP 1000 International User Group**, which was created by the HP 10 user community:

> ". . . . to provide a forum for sharing information among users to enhance their HP 1000 systems, ... and to help their development effort for software . . . ." (Joe Getkin, Startup Committee Chairman, writing in the first issue of INTERFACE/1000, the newsletter of the users group.)

This responsibility involves all phases of library operation, from initial program collection to distribution of the library. Thus, all questions regarding library availability, contributed program existence and/or operation should be directed to the users group.

WHY IS EVERYONE EXCITED ABOUT PLUS/1000?

It Is Easy To Use:

Fast and easy identification of any program in the library is now available. The PLUS/1000 software is classified in **packages**, oriented towards the main application areas of the HP 1000 systems: manufacturing and business management, research and development, measurement and control, data communication. A systems programming package and a demonstrations and games package are also available. A non-HP 1000 compatible software package supplements the library. For quick identification of any program in the library a **catalog** is provided. PLUS/1000 catalog gives, by package, the abstract of each program, the considerations related to hardware and software dependency, the grade and date code of the program, and the author's name. The catalog also contains various **indexes** created around **keywords** over the entire library and around **categories** inside each package. The **software** for package and the **corresponding documentation** are stored **on magnetic tape** .

It Is Easy To Order:

The HP 1000 International Users Group will distribute PLUS/1000 by subscription. The **library subscription** consists of:

1. Complete contributed library on magnetic tape (800 or 1600 b.p.i.)
2. Periodic library updates

It Is Inexpensive:

1. For general membership in the Users Group the library subscription is included as a benefit. Annual dues for general membership is $250.
2. For non-users group members, the annual fee for the library subscription is $250.
3. Price of an additional contributed library catalog is $25 each.

It Is Easy to Submit Programs to PLUS/1000:

The contributor **now has available** for use the program SUBMT. This program interacts conversationally with the contributor to provide, in a standardized format, all information needed to document a software contribution. The validated answers to SUBMT questions are recorded on a file and this file is stored by the user on the tape containing his contributed software.

It Is Easy to Maintain PLUS/1000:

The new library is structured around the PLUS/1000 data base. By using the data base and the programs written to access it, the acceptance of contributed software and the catalog are automated. (See the bulletin in this issue, "PLUS/1000 Organization Automates the Operation of the Library".)

New Significant Programs

JGL — Jim's Graphics Library is a set of Fortran routines that interface graphics applications programs to the HP 2648 graphics terminal.It was designed for the purpose of creating demos of the HP 1000, and has been used very successfully in that endeavor. Nothing seems to make a better impression than watching a computer smoothly and quickly step through a complex series of graphics gymnastics.

PENNY — A brand-new library addition is the program PENNY. This is an exciting program that should gain a lot of popularity very quickly. Basically, PENNY allows the user to create graphics drawings from conversing interactively at the terminal. When he has the drawing as he wants it, the user can store it away in a data base maintained by PENNY. Drawings in the data base can be retrieved, modified, combined with other drawings, and then stored away again. Final results can be output to a plotter to create a high quality hardcopy. PENNY can be used for digital schematics design, floor layouts, piping diagrams, and a host of other graphics applications. (See the feature article, "PENNY: Computer Aided Drawing on the HP 1000" in this issue.)

TMATT — Time and Attendance Datacapture program. This program has been written for the HP 1000 to facilitate high speed data input from 3077A time reporting data collection terminals. For the complete description of the program see the feature article in this issue, titled "Time and Attendance Datacapture with the HP 1000".

OBTAIN YOUR COPY NOW!

PLUS/1000 has already been turned over to the newly established HP 1000 International Users Group for distribution and maintenance. If you have not joined the users group as of yet, you can join it now. Send in your application form today and obtain the new library without delay.

The HP 1000 International Users Group address is:

HP 1000 International Users Group
P.O. Box 1000
Norwood, Massachusetts 02062

LOCUS DISAPPEARS MARCH 31,1980

With the advent of the new PLUS/1000 contributed library the need to continue to supply LOCUS no longer exists. The most popular LOCUS programs have been placed in PLUS/1000 and LOCUS will be discontinued.

Therefore, until March 31, Hewlett-Packard Data Systems continues to accept orders for:

— the programs contained in LOCUS as they were announced in the Communicator/1000
— the LOCUS History Master Tapes — contributed programs collected until November 1977

After March 31, 1980 Data Systems will discontinue the distribution of any contributed library related product. LOCUS will disappear and PLUS/1000 will be handled by the new HP 1000 International Users Group. See the accompanying article entitled "Announcing PLUS/1000: The New Contributed Library for HP 1000 Systems" for more information about PLUS/1000.

PLUS/1000 ORGANIZATION AUTOMATES THE OPERATION OF THE LIBRARY

PLUS/1000 is the new Contributed Library for HP 1000 systems. The library is structured around an IMAGE/1000 data base to allow easy and fast identification of any program in the library and to produce the PLUS/1000 catalog and updates.

The PLUS/1000 contributed software is classified in packages oriented towards the main application areas of the HP 1000 systems:

A. manufacturing and business management
B. research and development
C. measurement and control
D. data communication
E. systems programming
F. demonstrations and games
G. non-HP 1000 compatible software.

LIBRARY MAINTENANCE

Various application programs and QUERY can access the PLUS/1000 data base and permit the creation of the PLUS/1000 catalog.

The main application program updates the data base when new software has been contributed. It reads the submission file, supplied by the contributor for documenting his software, and checks the validity of the information. For correct information, it creates a new entry in the data base; for improper information, it rejects the contributed software.

The catalog-writing programs work in conjunction with QUERY for accessing the data base and create the catalog, which includes abstracts and indexes.

Two programs are used to simplify the distribution of the library. Program PLUSW stores the contributed library from disc to magnetic tape in a packed format. Program PLUSR, supplied within the library, reads the programs from the magnetic tape and stores them on disc.

SUBMISSION BY USERS

The initial submission of contributed software by the user is based upon the utilization of program SUBMT. This program is run by the contributor to provide, in a standardized format, all information needed for documenting the contributed software. SUBMT asks questions and records answers; it checks the validity and format of most answers; it creates a disc, cassette or magnetic tape file containing all the answers; and, on request, it gives a listing of the file. This file must be stored by the user as the first file on the tape containing the contributed software. This file is the submission file used to create a new entry in the PLUS/1000 data base. Each software contribution to PLUS/1000 must contain:

— the submission file created by answering the questions SUBMT asks,
— the source(s) of the contributed program,
— optionally, an extended documentation file, if needed.

Following is a summary of PLUS/1000 operation. This summary shows the path followed by the contributed software from submission to acceptance and utilization, the maintenance and distribution of PLUS/1000 being the responsibility of the new HP 1000 International Users Group (see previous bulletin).

INTERNATIONAL TRAINING CENTER ADDRESSES

AUSTRIA
(Vienna)
Handelskai 52
Postfach 7
A 1205 Wien
Tel: (0222) 35 16 21-32
Telex: 75923
Cable: Hewpack Wien

AUSTRALIA
(Blackburn) B
CUSTOMER TRAINING CENTER
31-41 Joseph Street
Blackburn, Victoria, Australia
(Pymble) P
CUSTOMER TRAINING CENTER
31 Bridge Street
Pymble, New South Wales, Australia

BELGIUM
(Brussels)
Avenue du Col Vert, 1
Groenkraaglaan
B-1170
Brussels, Belgium
Tel: (02) 672 22 40

ENGLAND
(Altrincham) A
Navigation Road
Altrincham
Cheshire WA14 1NU
(Winnersh) W
King Street Lane
Winnersh, Workingham
Berkshire RG11 5 AR
Tel: Workingham 784774
Cable: Hewpie London
Telex: 8471789

FINLAND
(Helsinki)
Nahkahousuntie 5
00211 Helsinki 21
Tel: 90-692 30 31

FRANCE
(Grenoble) G
5, avenue Raymond-Chanas
38320 Eybens
Tel: (76) 25-81-41
Telex: 980124
(Orsay) O
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401-Orsay
Tel: (01) 907 7825

GERMANY
(Boeblingen)
Kundenschulung
Herrenbergerstrasse 110
D-7030 Boeblingen, Wurttemberg
Tel: (07031) 667-1
Telex: 07265739
Cable: HEPAG

ITALY
(Milan)
Via Amerigo Vespucci, 2
20124 Milan
Tel: (2) 62 51
Cable: HEWPACKIT Milano
Telex: 32046

JAPAN
(Osaka) O
Chuo Building
5-4-20 Nishinakajima
Yodogawa-Ku, Osaki-shi
Osaka, 532 Japan
Tel: 06-304-6021
Telex: 523-3624 YHP OSA
(Tokyo) T
2205 Takaido Higashi 3-chome
Suginami-Ku, Tokyo 168
Tel: 03-33-8111
Telex: 232-2024 YHP MARKET TOK

NETHERLANDS
(Amsterdam)
Van Heuven Goedhartlaan 121
Amstelveen 1134
Netherlands
Tel: 020 472021

SPAIN
(Madrid)
Jerez No. 3
E-Madrid 16
Tel: (1) 458 26 00
Telex: 23515 hpe

SWEDEN
(Stockholm)
Enighetsvagen 1-3, Fack
S-161 20 Bromma 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Telex: 10721

SWITZERLAND
(Zurich)
Zurcherstrasse 20
8952 Schlieren
Tel: 01/730 52 40
(Geneva)
47 Avenue Blanc
1202 Geneva
Tel: 022/32 48 00

For course prerequisites and registration information contact one of the HP training centers listed above.

Although every effort is made to ensure the accuracy of the data presented in the **Communicator,** Hewlett-Packard cannot assume liability for the information contained herein.

Prices quoted apply only in U.S.A. If outside the U.S., contact your local sales and service office for prices in your country.

Printed in U.S.A. 12/79

Part No. 5951-6111