Volume 1 No. 6 March/April 1978

Journal
of the HP General Systems Users Group

Executive Board

Published By-Monthly

Contributions: Address the Journal Editor

Journal Editor Elias Zabor
HP General Systems Users Group, c/o Hewlett-Packard Company, 5303 Stevens Creek Blvd., Santa Clara, CA 95050, (408) 249-7020

Featured Articles

Tips and Techniques

Contributed Library Corner

The Clearing House

All About Us

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Featured articles

Asynchronous Communications Protocols

by Charles J. Villa, Jr.*
Alter•Ability
154 Laidley Street
San Francisco, CA 94131
(415) 641-1692

Introduction

Asynchronous communications is the means by which the HP 3000 "talks" to all of its interactive terminal devices. Data is passed back and forth from the CPU, through the Asynchronous Terminal Controller (ATC), down a communications line, to the terminal, and back in reverse order.

This presentation concerns one link in this communication chain: the communication line, or data link. The HP 3000 recognizes three asynchronous line disciplines:

1) Hardwired, using no modem;
2) 103 protocol, full-duplex;
3) 202 protocol, half-duplex, reverse-channel.

Hardwired Communications

Hardwired communications is the simplest link between an HP 3000 and a terminal. Only three signals from the RS-232 interface are used: pin 2 (Transmitted Data, or "BA"); pin 3 (Received Data, or "BB"); and pin 7 (Signal Ground, or "AB"). No other signals are required or used. Particular care should be taken to remove all connections to pins 9, 13, 14, 15, 16 and 18, since these signals may affect system or terminal performance. A complete table of the RS-232 interface signals and mnemonics is presented in Appendix A.

The ATC clocks data bits into and out of the HP 3000. All characters are represented by a ten-bit word: one start bit, seven data bits, a parity bit, and a stop bit. Parity on the HP 3000 is "None" (no checking), with the parity bit forced to a 1.

When you configure a logical device on the HP 3000, you are defining a port on the ATC, and *not* a terminal itself. The characteristics given to that port then define the way in which the HP 3000 handles any terminal plugged into that port. Two configuration parameters are critical in defining the data link: the terminal sub-type; and the port speed (baud rate).

Hardwired terminals should be configured as sub-type 0, which provides only the three interface signals described earlier. Sub-type 0 also provides for speed-sensing of the port, whereby the speed of the incoming port is determined when the initial carriage-return is received by the ATC. Speed-sensing is not necessary if the speed of a given port is fixed, or if a leased-line full-duplex modem is used.

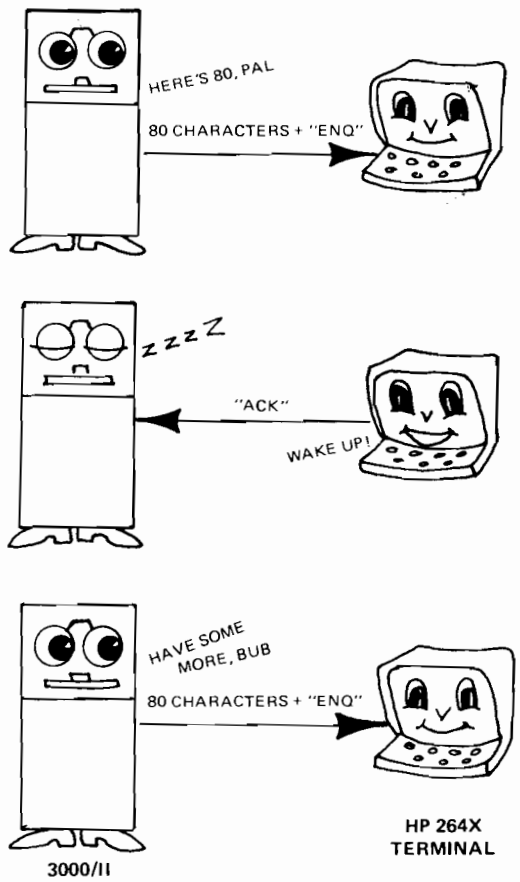
Some sites have reported that speed-sensing of certain terminals causes problems which force the ATC to set *all* ports to operate at 2400 baud. In either case, you may set a fixed port speed by configuring a port as sub-type 4 and then entering the desired baud rate under the "Default Terminal Speed" configuration question.

For hard-wired terminals, then, the HP 3000 provides no control signals whatsoever. Data is sent and received without regard to the "readiness" of the terminal on the other end of the line. A possibility of data overruns exist when you have many high speed hardwired terminals (i.e., characters in/out of the ATC exceed the ATC bandwidth).

Terminal Type

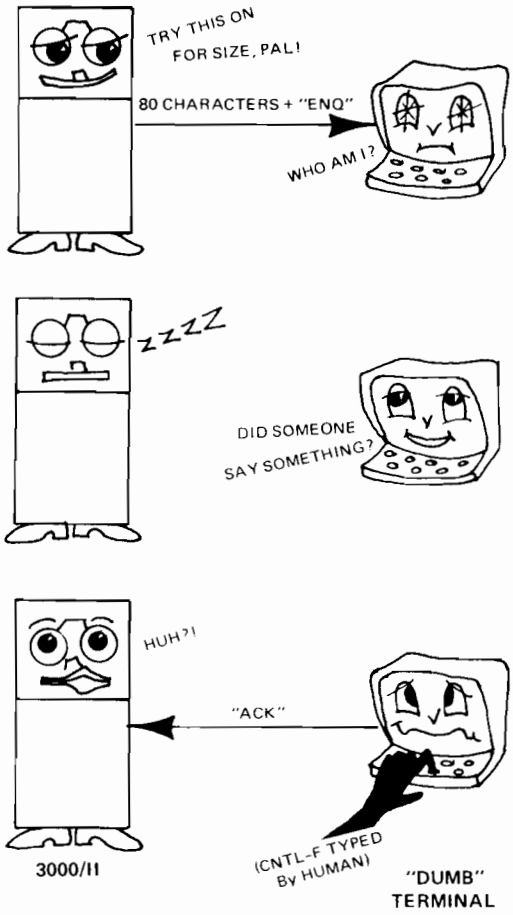
Every terminal requires a certain amount of time to perform a function like carriage-return, line-feed, horizontal tab, etc. During the time the terminal executes the function, any other data coming to it will be discarded. Therefore, pad characters — "nulls" — are added to the data stream immediately after the characters which cause such functions to be performed. The result is that "nulls" are discarded instead of your good data.

Figure 1



Editor's Note: Charles Villa is Chairman of the Data Communications Technical Advisory Committee.

Figure 2



The Terminal Type parameter of the configuration or log-on dialogue tells the HP 3000 how many such "nulls" to add after certain functions. Since there are only ten terminal-types, and hundreds of different terminals, there will arise situations in which you have a terminal that does not appear in the terminal-type table (see MPE Commands Reference Manual). In such a case, you should check the manufacturer's specifications regarding proper delays in order to determine the best Terminal Type to use (contact your local SE for assistance).

Terminal Type also determines how a backspace is handled (whether a "®" is printed, etc.), and how the password at log-on is masked.

Term-type 13 is a new designation implemented for users of value-added networks like Telenet and Tymnet. Since these VAN's provide their own terminal handling via their own terminal ID (which is part of their "added value"); term-type 13 provides no padding or terminal control from the HP 3000. Moreover, since these VAN's normally provide echoplex operation from their closest central office, term-type 13 also turns off the echo from the HP 3000. Usually, echo from a central office two-hundred miles distant is much faster than that from the HP 3000, which may be a two-*thousand* mile distance.

Term-type 10 has a special effect of which you should be aware. When used with the Series II it causes the 3000/II to send an "ENQ" character to the terminal after every 80 characters. The 3000/II then waits to receive an "ACK" before sending any more data.

This "ENQ-ACK" sequence is a simple line protocol implemented specifically for HP 264X terminals. Without it, data overrun could occur on the terminal when running at 2400 baud. The reason the protocol works is that the HP 264X terminal recognizes an "ENQ" as the signal to fire back an "ACK" (see Figure 1). With other than HP terminals, the following could occur:

1) Very few terminals aside from the HP 264X can talk in "ENQ-ACK" language. If you plug such a terminal into a port configured as term-type 10, that terminal will "hang-up" after 80 characters are sent to it. The only way out is for you to type a CNTL-F (which is actually an "ACK" character) after every 80 characters, until the write terminates (see Figure 2). Then, either plug the terminal in elsewhere, log-on with a term-type other than 10 (which overrides the default), or take the machine down and reconfigure the port.

2) Sometimes the HP terminal doesn't send an "ACK". Your terminal will seem to hang up. Type a CNTL-F to get going again.

3) The HP 3000/Series I (or CX) does not speak the "ENQ-ACK" language. As a result, on a lightly-loaded machine, HP 264X terminals running at 2400 baud may be affected by data overrun. The symptom is a garbled screen form, a little gray box printed on the screen, a line missing from a report, and so on. The only remedy is to run the terminal at 1200 baud, either at log-on time, or via the ":SPEED" command once logged-on.

Telecommunications

Transmitting data across phone lines requires converting the computer's digital signal (basically a square wave) into the phone circuit's analog signal (sine wave). This is the job of a *modem.* The modem modulates digital signals into analog voltages in transmission, and demodulates analog to digital in reception.

The modem modulates a carrier wave; thus, changes in the carrier wave are what determines whether a 0 or a 1 has been sent or received. Though there are many ways of modulating a signal, that most commonly seen with the low speeds used on the 3000 is *frequency shift keying* (FSK), where one frequency represents a 1, and another frequency a 0. Modems at both ends of the line must use the same kind of modulation in order for them to understand each other.

There are several line disciplines used in asynchronous telecommunications. Since a local modem usually "knows" only one discipline, the modem at the other end of the line must also know that discipline. The safest approach to ensuring modem compatibility is to use the same kind of device at each end.

The Data Access Arrangement (DAA)

The DAA is a little gray box placed between your modem and the phone lines. It's there to protect your equipment and telephone company lines from power surges and the like, as well as to simulate the effect of a ringing phone being answered for your modem. The operation of the DAA is depicted in Appendix B.

Before June 1, 1977, everyone had to rent DAA's from the Telephone Company. The type most people used (CBS-1001A or 1001F) rented for about $6 or $7 a month. Then, some enterprising modem manufacturers decided they'd like to make their own DAA's, and even put them inside the modem. Now, you may purchase a modem with a built-in DAA, and plug it into a Data Jack as provided by your telephone company.

103 Protocol

The 103 telecommunications protocol is a simple discipline which supports 300 baud operation with many modems, or 1200 baud operation with the Vadic 3400, the Bell 212, or the Prentice P-1200-PLUS. Any port on the ATC which is to use the 103 protocol must be configured as either sub-type 1 or sub-type 5 (for no speed-sensing).

The purpose of the 103 protocol is simply to establish a communications link. The protocol is full-duplex (data sent in both directions at once), so that "turning the line around," that is, going from the transmit to receive state, or vice-versa, is not necessary.

The sequence of events in line connection is shown in Appendix C. The HP 3000, looking like just another piece of DTE (Data Terminal Equipment), varies from this sequence in that Data Terminal Ready (CD) is always on, or HIGH, except when it is dropped to disconnect the modem from the line.

The sequence of events for connection is:

1) The originator places a call to the remote device (in our case, a remote HP 3000).
2) At the answering modem, Ring Indicator is turned on to indicate an incoming call. In normal operation, the terminal would then set Data Terminal Ready HIGH; however, the HP 3000 already has this signal up.
3) After answering a call (CE HIGH) and seeing CD HIGH, the modem sets Data Set Ready (CC) HIGH. The HP 3000 now starts its log-on timer.
4) After 1.5 seconds, the answering modem transmits an F2 Mark (F2M) signal to disable echo suppressors.
5) Back at the originating modem, the operator hears the F2M tone as a high-pitched squeal, and puts the modem in Data mode (either with a DATA button, putting the handset in the coupler, or whatever). Data Set Ready (CC) is set HIGH by the originating modem.
6) 1.5 seconds after the originating modem sets CC HIGH, it sends an F1M tone to the answering modem. Then, 265 milliseconds later, it sets Clear to Send (CB) and Carrier Detector (CF) HIGH. This signals to the attached terminal that all is "go."
7) When the answering modem "hears" the F1M tone, it sets CB and CF HIGH. The HP 3000 must see CF go HIGH within 30 seconds after CC is set HIGH. Otherwise, it will drop CD for 5 seconds to cause the modem to disconnect from the line.

The HP 3000 will drop CD for 5 seconds and terminate the session under the following conditions:

1) The log-on timer expires.
2) CF does not go HIGH within 30 seconds after CC is set HIGH.
3) CC drops.

Normal session termination occurs after a ":BYE" command. The HP 3000 drops CD for 5 seconds, which causes its modem to drop CF. Loss of carrier detected by the originating modem causes it to disconnect from the line, also.

202 Protocol

The 202 protocol is a half-duplex protocol which normally is used with FSK-type modulation. The nature of this modulation is such that data can only be sent in one direction at a time at speeds of 1200 baud with adequate low-error rates. Thus, a more complicated protocol must be used to tell each end of the link when to transmit, and which to "turn-around" to the receive state. Every time this "turn-around" occurs, a delay of about 200 milliseconds is introduced, because the modem must retransmit the tone necessary to disable echo suppressors. That kind of overhead is costly, especially in an interactive inquiry-response environment, where the blocks of data sent back and forth between terminal and computer are very small.

Modems like the Vadic 3400 are able to run at 1200 baud, but with a *103, full-duplex protocol.* They are able to do so because they use a modulation technique called *Dibit Phase Shift Keying* (DPSK). This kind of modulation represents two bits of data for every 90 degree shift in the phase of the modulated carrier wave. Thus, twice as much information can be represented per cycle of sine wave.

If possible, use such modems for 1200 baud operation. Not only is the 103 protocol simpler, it is inherently more efficient than the 202 half-duplex protocol.

If you use Bell modems for 202 operation, be sure to order them with reverse channel. The HP 3000 only supports reverse-channel 202 operation (no main channel, please).

Sub-type 2 or sub-type 6 (for no speed-sensing) is used for ports configured for 202 operation. *Note that this sub-type is used only if you are using 202-protocol modems.* As previously mentioned, some 1200 baud modems operate under the 103 protocol, in which case sub-type 1 or 5 is to be used.

Cabling and Modems

Appendix D is a reprint from the April, 1976 issue of the *Communicator*. It is a definitive listing of what options must be ordered with what modems, how cables must be fabricated, etc. You should refer to this any time you have questions about modems or cables. You should also be aware that most of the modem "Recommendations" are *requirements* in normal operation. You should not deviate from those recommendations unless you really know better.

A recent development in the modem arena is the introduction of a device which "speaks" at either 300 or 1200 baud. The Bell version is model 212A; it'll "talk" to a 103 or another 212. The Vadic version (out in June 1978) is the 3467; it'll "talk" to a 103 *or* a 212 *or* a Vadic 3400 *or* another Vadic 3467. Presumably, other vendors will be out with these devices soon. There are tremendous benefits to having one modem which will talk both to your high-speed, locked-down modems *and* to your low-speed couplers and portable terminals: one line instead of two; one port used on your ATC; one manufacturer, and so on. And both the Bell 212A and the Vadic 3467 use a 103-type protocol, thanks to DPSK modulation. So configure them as sub-type 1 (but *not* 5; your port has to be able to operate at either 300 baud or 1200 baud).

Further Information

The area of data communications on the HP 3000 is beginning to grow exponentially. Recently HP produced a guide to communications on the HP 3000. If you do any data communications at all, you should get this little handbook. It's called "Communications Pocket Guide," and the part number is 30000-90105.

A useful introduction to the area is HP's "Guidebook to Data Communications," part number 5955-1715.

The definitive statement concerning the HP 3000, line disciplines, sub-types, etc. is an article by Don Van Pernis in issue number 15 of the *Communicator* (I'm indebted to Don and his article for much of the information presented herein).

Data Communications magazine is very valuable if your datacomm needs are growing, if you're trying to keep on top of trends, or if you're just interested. There's also a good book by the same publishers called "Basics of Data Communications."

Vadic Corporation publishes a free brochure called "Telecommunications from the Terminal User's Guide," which provides some very useful information in a very compact and well-illustrated form.

Write:

The Vadic Corporation
222 Caspian Drive
Sunnyvale, CA 94086

Minicom Systems also publishes a comprehensive brochure called "Multiplexors, Concentrators and Communications Controllers." Write:

Minicom Systems
9551 Irondale Avenue
Chatsworth, CA 91311

● ● ●

Appendix A: RS-232 Interface Signals

EIA RS232C Interchange Circuits by Category

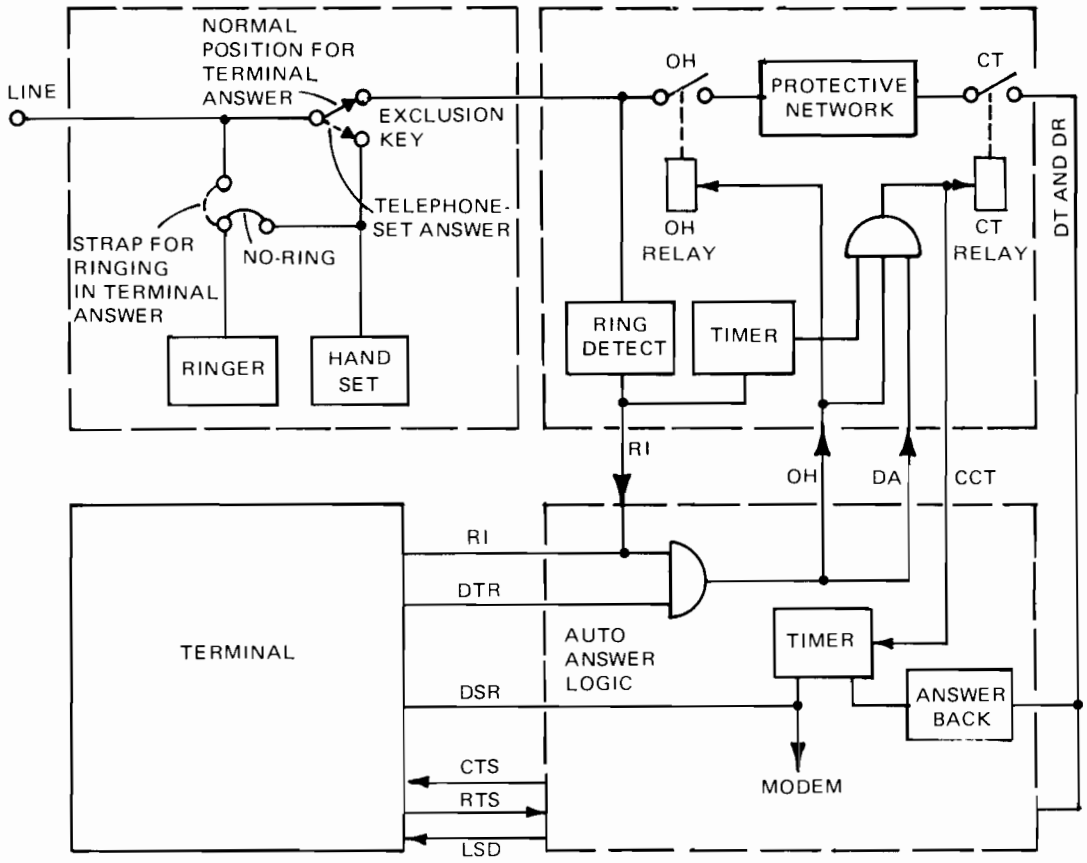
INTERCHANGE CIRCUIT	CCITT V24 EQUIVALENT	DESCRIPTION	GND	DATA		CONTROL		TIMING	
				FROM DCE	TO DCE	FROM DCE	TO DCE	FROM DCE	TO DCE
AA	101	Protective Ground	X						
AB	102	Signal Ground Common Return	X						
BA	103	Transmitted Data			X				
BB	104	Received Data		X					
CA	105	Request to Send					X		
CB	106	Clear to Send				X			
CC	107	Data Set Ready				X			
CD	108.2	Data Terminal Ready					X		
CE	125	Ring Indicator				X			
CF	109	Received Line Signal Detector				X			
CG	110	Signal Quality Detector				X			
CH	111	Data Signal Rate Selector (DTE)					X		
CI	112	Data Signal Rate Selector (DCE)				X			
DA	113	Transmitter Signal Element Timing (DTE)							X
DB	114	Transmitter Signal Element Timing (DCE)						X	
DD	115	Receiver Signal Element Timing (DCE)						X	
SBA	118	Secondary Transmitted Data			X				
SBB	119	Secondary Received Data		X					
SCA	120	Secondary Request to Send					X		
SCB	121	Secondary Clear to Send				X			
SCF	122	Secondary Received Line Signal Detector				X			

EIA RS232C Interface Connector Pin Assignments

Signal Direction Terminal	Signal Direction Modem	Pin Number	Circuit	Description
		1	AA	Protective Ground
	——►	2	BA	Transmitted Data
◄ --	--	3	BB	Received Data
	►	4	CA	Request to Send
◄ -		5	CB	Clear to Send
◄ -		6	CC	Data Set Ready
		7	AB	Signal Ground/Common Return
◄ --	-	8	CF	Received Line Signal Detector
		9		(Reserved for Data Set Testing)
		10		(Reserved for Data Set Testing)
		11		Unassigned
◄	---	12	SCF	Secondary Received Line Signal Detector
◄		13	SCB	Secondary Clear to Send
-	►	14	SBA	Secondary Transmitted Data
◄	-	15	DB	Transmission Signal Element Timing (DCE Source)
◄		16	SBB	Secondary Received Data
◄		17	DD	Receiver Signal Element Timing (DCE Source)
		18		Unassigned
	►	19	SCA	Secondary Request to Send
	►	20	CD	Data Terminal Ready
◄	-	21	CG	Signal Quality Detector
◄	-	22	CE	Ring Indicator
-	►	23	CH/CI	Data Signal Rate Selector (DTE/DCE Source)
--	►	24	DA	Transmitter Signal Element Timing (DTE Source)
		25		Unassigned

From "Guidebook to Data Communications"
HP 5955-1715

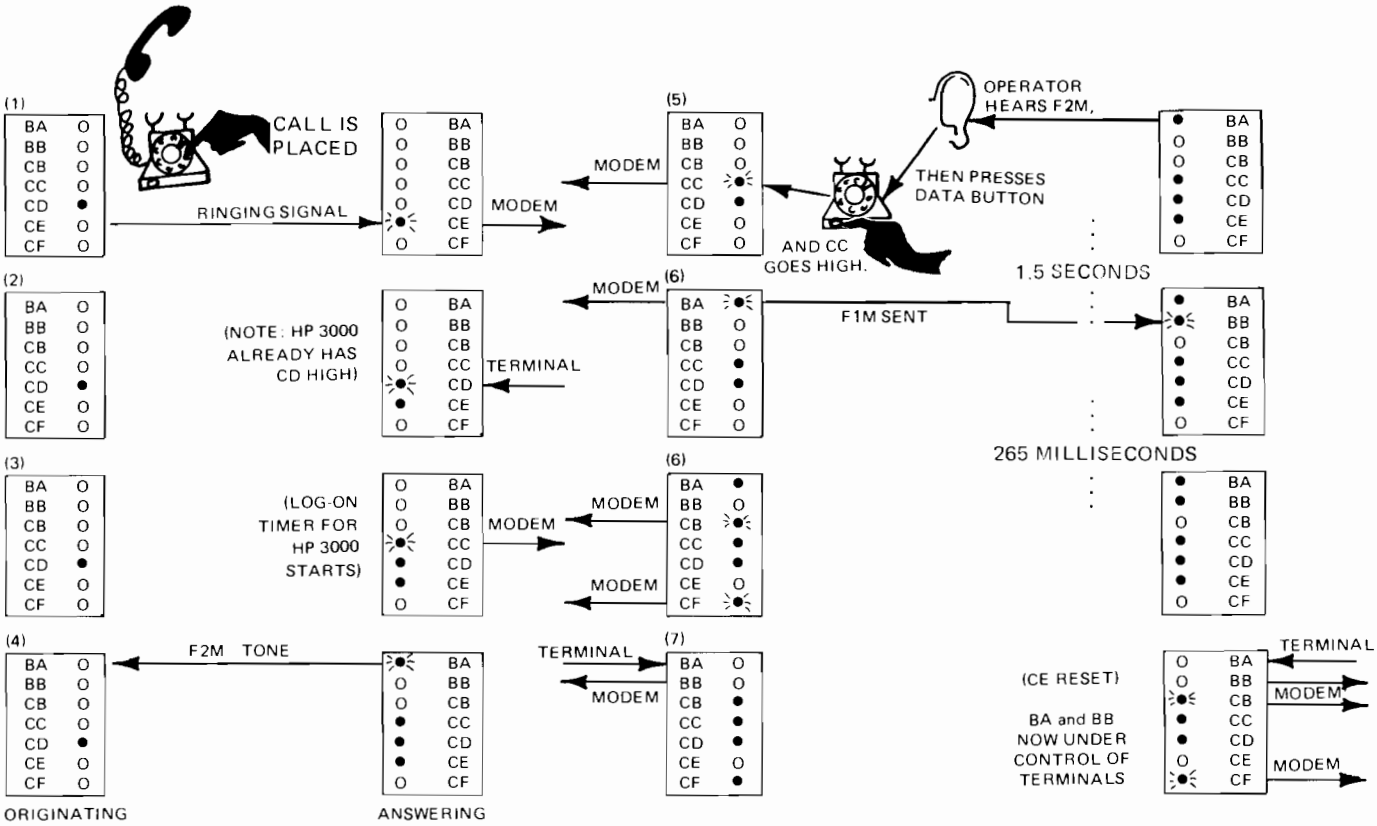
Appendix B CONTROL LINES



1) DAA detects ringing signal, sets RI HIGH.
2) Modem sets OH on if Data Terminal Ready (CD) and Ring Indicator (from DAA) are HIGH. Setting OH on simulates the answering phone going off-hook (receiver picked up).
3) Data Available (DA) signal from modem is turned on to indicate to DAA that modem is ready.
4) Coupler Cut Through (CCT) set on by DAA 2 to 5 seconds after DAA sees DA go on. CCT signals to modem that it has been put on line, and should send answerback tone (F2M).

(Diagram from "Basics of Data Communications," McGraw-Hill, 1976, p. 247).

Appendix C


(1)
CALL IS PLACED
RINGING SIGNAL
MODEM
(5)
OPERATOR HEARS F2M,
THEN PRESSES DATA BUTTON
AND CC GOES HIGH.
1.5 SECONDS
(2)
(NOTE: HP 3000 ALREADY HAS CD HIGH)
TERMINAL
(6)
F1M SENT
265 MILLISECONDS
(3)
(LOG-ON TIMER FOR HP 3000 STARTS)
(6)
(4)
F2M TONE
TERMINAL
(7)
(CE RESET)
BA and BB NOW UNDER CONTROL OF TERMINALS
ORIGINATING
ANSWERING

Appendix D

MODEMS AND OPTIONS

Following are the recommended asynchronous modems and options to be used in conjunction with the 30032B sub-system. Note that these options are those required *at the CPU end* (local). To insure successful communication with the distant end, different options in the distant data set may be required.

103 OPTIONS AND RECOMMENDATIONS

OPTION	DESCRIPTION	RECOMMENDATIONS	
		COMPUTER	TERMINAL
A	1. Rotary Dial 2. Touch Tone Dial	Area Optional	Area Optional
B	3. With Card Dialer 4. Without Card Dialer	Customer Decisions	
C (Note 1)	5. Loss of CXR on Disconnect 6. No Loss of CXR Disconnect	C5	C5
D (Note 2)	7. Send Space Disconnect 8. Send No Space Disconnect	D7	D7
E	9. Receive Space Disconnect 10. No Receive Space Disconnect	E9	E9
F (Note 3)	11. Auto Answer Permanent 12. Auto Answer Select	F11	F12

Note 1: **CXR Disconnect. Data set disconnects with carrier loss for more than 160 msec.**

Note 2: **Space Disconnect. A Space Disconnect is initiated by removal of DTR – the associated CPU space will result in modem disconnect.**

Note 3: **Auto Answer. Permanent on CPU side but selectable on terminal side – this means the call to the terminal must be manually answered if F12 is selected.**

For further definition of these options/modem capabilities refer to the relevant "Bell System Technical Reference" publication that is available from your local Bell System Representative.

Bell 103A3, 0-300 bits/sec, Asynchronous
(Replaces 103A2 which may still be available in some instances.)

Full Duplex on 2 wire operation – no reverse channel is available. The 103 works with another 103 or 113 modem.

Bell 113B, 0-300 Bits/sec, Asynchronous
An "answer only" modem.
Auto Dialing and calling is not available.
Full duplex on 2 wire operation – no reverse channel is available.
The 113B works with other 100 series data sets that have originate capabilities.

113B OPTIONS AND RECOMMENDATIONS

OPTION	DESCRIPTION	RECOMMENDATIONS
U	Dummy Load	Required for 10
V	Common	Not installed
W	Common CB/CF	In
X	Ignore data terminal make busy	In
Y	Make busy implementation	In
Z	Data terminal controls disconnect	In

The 113B-L1 is an answer only modem which is rack mounted with *auto seek* for open port.

Appendix D (Continued)

202C OPTIONS AND RECOMMENDATIONS

OPTION	DESCRIPTION	RECOMMENDATIONS		COMMENTS
		COMPUTER	TERMINAL	
A	1. EIA Interface 2. Contact Interface	A1	A1	Required
B	3. With 801 ACU 4. Without 801 ACU	B4	B4	
C	5. With Auto Answer 6. Without Auto Answer	C5	C6	If C6 installed no D option apply
D	7. Auto Answer Only 8. Auto Answer Key Control	D7	–	D options depend on C options
E	9. Carrier Soft Turn Off (In) 10. Carrier Soft Turn Off (Out)	E9	E9	*

***Note: E9 normally used – when request to send is turned off, carrier frequency drops from 1200 Hz to 900 Hz. This can cause noise or transmission problems in a polling environment or on private lines; therefore, in these conditions, "E10" should be specified.**

Bell 202C Modem, 0 to 1200 bits/sec, Asynchronous

THE HP 3000 REQUIRES REVERSE CHANNEL IN 2 WIRE HALF-DUPLEX TRANSMISSION FOR SWITCHED NETWORK SERVICE.

10

Bell 202S and 202T Modems, Asynchronous

202S – provides speeds up to 1200 bits/sec simplex or half-duplex transmission for Switched Network service.

202T provides speeds up to 1800 bits/sec simplex, half-duplex or full duplex in private line service as follows.

1. 2 Wire Private Line with Reverse Channel
   - up to 1200 bits/sec on Basic 3002 channel.
   - between 1200 and 1800 bits/sec on 3002 channel with C2 conditioning.
2. 4 Wire Private Line and 2 Wire Private Line without Reverse Channel.
   - up to 1400 bits/sec on Basic 3002 Channel.
   - between 1400 and 1800 bits/sec on 3002 Channel with C2 conditioning.

202S OPTIONS AND RECOMMENDATIONS

OPTION	RECOMMENDATION
Received Data Squelch	156 m/Sec.
Clear-to-Send Delay	180 m/Sec.
Fast Carrier Detection	Out (Normal Mode – 23 m/Sec.)
Soft Turn-Off	In (24 m/Sec.)
Local Copy of Primary/ Reverse Channel	Out-On CPU Side *Note 1
Auto Answer (202S only)	Computer Side – Application Dependent
Received Data Clamp	In

***Note 1: If terminal has internal echo capability use OUT, else use IN.**

THE 202S IS UTILIZED BY THE HP 3000 IN HALF-DUPLEX WITH REVERSE CHANNEL IN SWITCHED NETWORK SERVICE.

The 202T is utilized by the HP 3000 in half-duplex with Reverse Channel or full duplex, 4 Wire with "subtype" specified as "0" in Private Line service.

SUBSYSTEM INTERFACE CABLE ASSEMBLIES

CABLE DESCRIPTION	PART NUMBER
25 foot cable for connecting a data set to a connector panel.	30062-60004
25 foot cable for connecting a terminal directly to the connector panel.	30062-60006
50 foot cable for connecting a data set to the connector panel.	30062-60007
50 foot cable for connecting a terminal directly to the connector panel.	30062-60009
100 foot cable for connecting a terminal directly to the connector panel.	30062-60012

Modem/Hardwire Extension Cable Manufacturing Specifications (Cable, Unshielded). This specification establishes requirements for a 25 conductor external low voltage computer cable with overall jacket U.L. style 2560.

Appendix D (Continued)

ELECTRICAL

Voltage Rating: 30V for Class 2 wiring systems only (220V rms test between conductors).

MECHANICAL

Construction Details

Singles: Twenty-five 26 (7 x 34) AWG tinned copper; tinned after stranding.

Insulation: PVC, seven-mil minimum wall thickness; rated at +60°C

Colors:

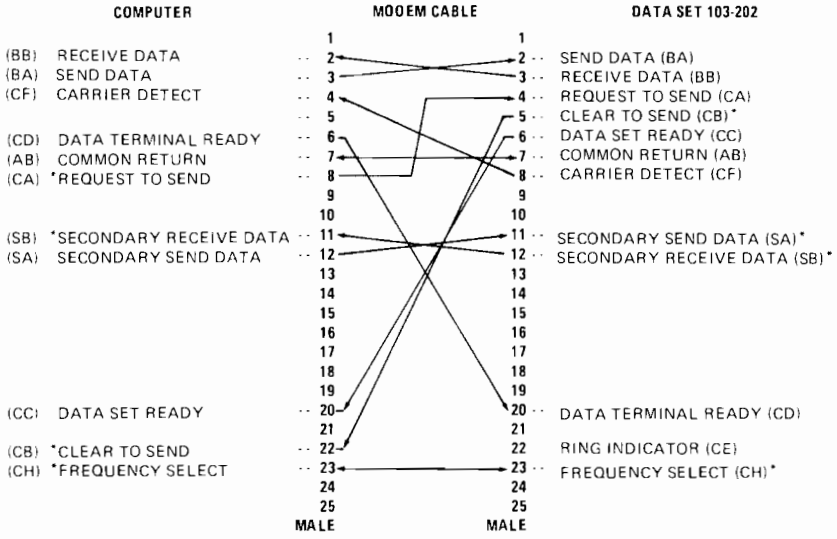
BLK	WHT/BLK	WHT/BLK/BRN
BRN	WHT/BRN	WHT/BLK/RED
RED	WHT/RED	WHT/BLK/ORN
ORN	WHT/ORN	WHT/BLK/YEL
YEL	WHT/YEL	WHT/BLK/GRN
GRN	WHT/GRN	WHT/BLK/BLU
BLU	WHT/BLU	
VIO	WHT/VIO	
GRA	WHT/GRA	
WHT		

Cable Lay: Twist singles for flexibility and in the above color sequence. Fillers, cloth or nylon binding may be used for a smooth, round construction.

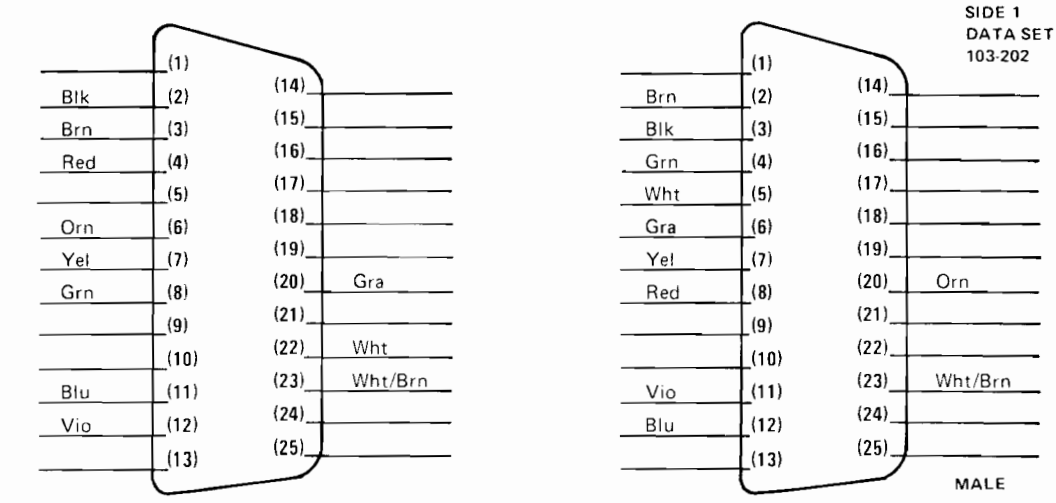
Jacket: PVC, 35-mil minimum wall thickness; rated at +60°C.

Color: Jade Gray per Visual Color Std., HP Part No. 6009-0021.

DATA SET CABLE SIGNAL INDEX
HP 30032B ASYNCHRONOUS TERMINAL CONTROLLER



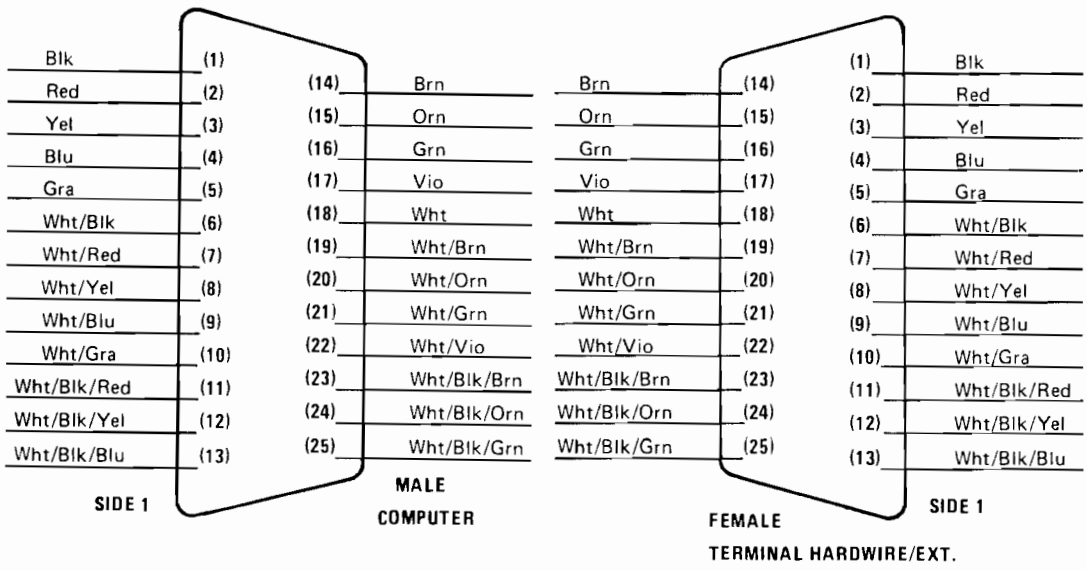
The asterisk indicates signals required for 202 data sets only.



Appendix D (Continued)

HARDWIRE/EXTENSION CABLE
HP 30032B ASYNCHRONOUS TERMINAL CONTROLLER

COMPUTER		MALE	FEMALE	EIA COMPATIBLE DEVICE	
		1	1		
(BB)	RECEIVE DATA	2	2	RECEIVE DATA (BB)	
(BA)	SEND DATA	3	3	SEND DATA (BA)	
(CF)	CARRIER DETECT	4	4	CARRIER DETECT (CF)	
		5	5		
(CD)	DATA TERMINAL READY	6	6	DATA TERMINAL READY (CD)	
(AB)	COMMON RETURN	7	7	COMMON RETURN (AB)	
(CA)	REQUEST TO SEND	8	8	REQUEST TO SEND (CA)	
		9	9		
		10	10		
(SB)	SECONDARY RECEIVE DATA	11	11	SECONDARY RECEIVE DATA (SB)	
(SA)	SECONDARY SEND DATA	12	12	SECONDARY SEND DATA (SA)	
		13	13		
		14	14		
		15	15		
		16	16		
		17	17		
		18	18		
		19	19		
(CC)	DATA SET READY	20	20	DATA SET READY (CC)	
		21	21		
(CB)	CLEAR TO SEND	22	22	CLEAR TO SEND (CB)	
(CH)	FREQUENCY SELECT	23	23	FREQUENCY SELECT (CH)	
		24	24		
		25	25		



Recursive Programming Using BNF

by Thomas R. Harbron
Anderson College
Computing Center
Anderson, IN 46011

Certain very difficult problems can be programmed correctly and efficiently by using a combination of two familiar tools: Backus-Naur Form (BNF) notation and recursive programming. The resulting code is guaranteed correct if the corresponding BNF is correct. Further, the code is both concise and efficient.

Much of the power of BNF lies in its inherently recursive nature. This attribute allows otherwise unwieldy definitions to be stated concisely. For example:

<NUMBER>::=<DIGIT>|<DIGIT><NUMBER>

<DIGIT>::=0|1|2|3|4|5|6|7|8|9

defines decimal integers of any size. The recursion in the first definition allows numbers with any number of digits greater than zero. Without the recursive property, a new definition would be needed for each different length of number or else a different convention would need be invented.

If we allow BNF to represent semantics as well as its usual role of syntax, it provides a notation for defining the structure of a program very precisely and concisely, provided that a few basic rules are followed.

The first rule is that definitions must never be circular, either explicitly or implicitly. The corresponding program structure is some form of endless loop. Non-circular definitions by contrast guarantee that the program will make progress toward termination.

The second rule is not as rigid as the first, for it affects program efficiency and not ultimate computability. The rule is that a term should not appear more than once in a definition if it is recursive with the item being defined.

To illustrate the problem that may arise, consider a mathematical definition of Fibonacci numbers:

$$L(N) = L(N-1) + L(N-2) \quad (N>2)$$

$$L(2) = 1$$

$$L(1) = 1$$

This recursive definition (for N>2) requires the procedure to reference itself at each level. In general, this results in $\sum_{i=1}^{N} M^i$ calls to the procedure where N is the number of levels and M is the number of calls per level. For the Fibonacci definition, M is two, and for an N of twenty the number of calls is approximately * 2^{21}, which is certainly excessive to calculate L(20).

The actual number is smaller since the second recursive call in the definition drops N by two levels rather than one.

This rule requires judgment in its application. If it is known that N will always be relatively small, the number of calls when M>1 may still be reasonable and a fair price to pay for a recursive definition.

To illustrate the application of these principles, consider the following problem. A program is to be developed which will examine a large number of records and select certain of these records for further processing. The criterion for selection is a logical expression to be entered at run time. The logical expression contains propositions relating to the data items in each record and is constructed according to the rules of Boolean Algebra.

The processing of the logical expression is done at three levels. Upon being entered, the expression is checked for syntactical correctness and, if correct, translated into a tabular form.

The second phase, which is repeated for each record, is to perform each test required by the logical expression and to produce a string of characters containing the logic values for each test and the operations which relate them. This string of characters always begins with an opening parenthesis and ends with a closing parenthesis for convenience. Besides parenthesis, it may contain the operators A (and), O (or), and N (not), and the values T (true), and F (false). This string of characters is henceforth referred to as "the string."

In the third phase, the string is analyzed to determine whether or not the original proposition is true for the given record. It is this third phase which will be the subject of our example; it is also the most difficult and error prone of the three phases.

Here is where we begin our BNF definitions. The object is to distinguish between a string which is true and one which is false; therefore the first definition:

1. <STRING>::=<T STRING>|<F STRING>

Thus objects of the class <STRING> are divided into two mutually exclusive classes.

Strings are required to be bounded by parenthesis. It is the value of the proposition within the parenthesis which determines the value of the string. Thus:

2. <T STRING>::=(<T PROP>)

3. <F STRING>::=(<F PROP>)

Further, matters will be much simpler if we state that all propositions are either true or false:

4. <PROP>::=<T PROP>|<F PROP>

It follows directly from equations 1 to 4 that all strings are propositions bounded by parenthesis, or:

<STRING>::=(<PROP>)

which need not be included in the formal definitions, but may be used.

As it has already been determined that all propositions are either true or false, we may safely define only those which are true and consign the rest to the class of false propositions. This cuts the work in half. Thus we begin the task of defining true propositions.

The simplest true proposition is a simple variable which has the value "true."

5a. <T PROP>:: = <T VAL>

Another easy one is the negation of a false proposition:

5b. |N<F PROP>

A true value followed by the operator "or" followed by any proposition is likewise true:

5c. |<T VAL> 0 <PROP>

as is a false value followed by "or" followed by a true proposition:

5d. |<F VAL> 0 <T PROP>

A true value followed by the operator "and" must be followed by a true proposition:

5e. |<T VAL> A <T PROP>

A proposition may begin with a string, and since a string is evaluated differently than a value, those definitions beginning with a value must be repeated for strings. Thus:

5f. |<T STRING>

5g. |<T STRING> 0 <PROP>

5h. |<F STRING> 0 <T PROP>

5i. |<T STRING> A <T PROP>

Finally the definitions for true and false values must be supplied:

6. <T VAL>:: = T

7. <F VAL>:: = F

The BNF definitions are now complete. It may be noted that the second rule has apparently been ignored in that the term <T PROP> appears four times on the right side of definition 5 and other terms whose definition ultimately depends on the definition of <T PROP> are even more common. While this does not present a serious problem it cannot be ignored.

Its resolution requires some knowledge of the mapping of the BNF onto the code. Definitions 5d, 5e, 5h and 5i, in which <T PROP> appear, all contain uniquely different precedents before <T PROP>. Thus the code need only use the definition of <T PROP> once, if at all.

Likewise the term <T STRING> occurs three times and <F STRING> once. Both ultimately depend upon the definition of <T PROP> and are thus recursive. Moreover they have no precedents and cannot be treated as <T PROP> was in the preceding paragraph.

However, if the proposition begins with <STRING>, it is possible to determine which kind of string (<T STRING> or <F STRING>) with a single call and apply the result of that call to definitions 5f, 5g, 5h and 5i without making a separate call for each definition.

In mapping the BNF to code, there are many options. In this case it is convenient to divide the definitions between two procedures, one to handle strings, the other propositions. While other implementations are feasible, this is the simplest one, since strings are defined in terms of propositions, while propositions are defined in terms of strings and propositions. A third procedure could be used for the definitions of <T VAL> and <F VAL>, but these are trivial and non-recursive.

Let us name the procedure to handle strings GEORGE and the procedure to handle propositions BOOLE. The resultant code in the SPL version of ALGOL is shown in Figure 1.

It may be seen that GEORGE does little more than strip away the parenthesis and call BOOLE to evaluate the resulting proposition. This is not surprising in view of definitions 1-3. BOOLE, by contrast, maps definitions 4-7 and is correspondingly longer. This does not imply, however, that GEORGE is trivial. Without it, we should find it much more difficult to organize the procedures.

An examination of Figure 1 reveals the close relationship between the definitions and the code. Likewise, a true, complete, and consistent set of definitions will guarantee that the code has the same attributes.

This property is very valuable for all problems that are amenable to definition in BNF or similar forms. Since these same problems are likely to involve recursive definitions, they are among the less tractable problems when approached with conventional methods.

• • •

Figure 1.

```
LOGICAL PROCEDURE GEORGE(S);  <<EVALUATES STRINGS PER DEFINITIONS 1-3>>
   BYTE ARRAY S;  <<CONTAINS STRING>>
BEGIN

   IF S(0)="(" THEN GEORGE:=BOOLE(S(1))  <<DEF 1,2,3>>
      ELSE QUIT(1);  <<ABORT, OBJECT IS NOT A STRING>>
END;  <<PROCEDURE GEORGE>>


LOGICAL PROCEDURE BOOLE(S);  <<EVALUATES PROPOSITIONS PER DEF 4-7>>
   BYTE ARRAY S;  <<CONTAINS PROPOSITION>>
BEGIN
   INTEGER X:=0;  <<POINTER FOR BYTE ARRAY>>
   INTEGER C:=1;  <<COUNTER FOR MATCHING PARENTHESIS>>
   LOGICAL B;     <<TEMPORARY STORE FOR VALUE OF STRING>>

IF S(X) <> "(" THEN  <<THIS DEFERS CONSIDERATION OF DEF 5F-5I>>
   IF S="T" AND S(1)=")"                              <<DEF 5A,6>>
   OR S="N" AND BOOLE(S(1)) = FALSE                   <<DEF 5B>>
   OR S="I" AND S(1) = "O"                            <<DEF 5C,6>>
   OR S="F" AND S(1) = "O" AND BOOLE(S(2))=TRUE       <<DEF 5D,7>>
   OR S="T" AND S(1) = "A" AND BOOLE(S(2))=TRUE       <<DEF 5E,6>>
   THEN BOOLE:=TRUE ELSE BOOLE:=FALSE                 <<DEF 4>>
ELSE BEGIN    <<DEFINITIONS 5F-5I>>
   DO IF S(X:=X+1)=")" THEN C:=C-1
      ELSE IF S(X)="(" THEN C:=C+1
   UNTIL C=0;  <<FINDS THE OTHER END OF THE STRING>>
   B:=GEORGE(S);  <<LET GEORGE DO IT ONE TIME, SAVE RESULT>>
   IF B=TRUE AND S(X+1)=")"                                <<DEF 5F>>
   OR B=TRUE AND S(X+1)="O"                                <<DEF 5G>>
   OR B=FALSE AND S(X+1)="O" AND BOOLE(S(X+2))=TRUE        <<DEF 5H>>
   OR B=TRUE AND S(X+1)="A" AND BOOLE(S(X+2))=TRUE         <<DEF 5I>>
   THEN BOOLE:=TRUE ELSE BOOLE:=FALSE                      <<DEF 4>>
END;
END;  <<PROCEDURE BOOLE>>
```

Featured articles

KSAM vs. IMAGE

by Steve Kaminski
San Bernardino Valley Municipal Water Dept.
San Bernardino, CA 92408

IMAGE (INTRODUCTION)

The IMAGE Information management sub-system is a two level hierarchical data base management system composed of master file data sets at one level and detail file data sets at the other, along with procedures for accessing the data sets. Each data set entry contains one or more data items with the capability of having the data items define a relationship between its data set and another. The data sets, data items and passwords (which are associated with the sets and items) are defined through a SCHEMA (an ASCII file) processed by a system program. The data sets are PRIVILEGED files, accessed through system procedures. Data sets and data items are also protected by the data base administrator and account manager through appropriate assignation of passwords and through file access restriction at the group and account levels. The building of the data base is accomplished through a system utility program which also allows the data base to be purged or erased. At the time the data base is built, all space defined for the data sets is allocated. Once the data base is allocated, modifications can be made by using a system program to unload and purge the data base, redefine the schema, rebuild and load the data base – however these modifications are limited. A user contributed program (DBREBILD) exists in the USER group library which extends the scope of modifications that can be made to an existing data base.

IMAGE (STRUCTURE)

The two level hierarchy of IMAGE is implemented by defining two data set types, master data sets and detail data sets and by relating entries in a detail data set to entries in one or more master data sets through search items. All items in a detail data set which have the same search item value are chained together (optionally in sorted order by another data item). The detail chain is a doubly linked chain permitting data to be accessed in forward or reverse direction.

If data in a chain of a detail data set is to be retrieved, the chain must be located which requires the storage of the address of some point of the chain. IMAGE stores the address of the detail chain head in the master file entry of the associated master file. A hashing algorithm is performed on the search item value to establish the address for the master file entry. In the case where multiple search item values hash to the same address, the secondary entry is assigned an address in the vicinity of the hashed address and the secondary entry is chained to the first. In this manner collision of master file entries is rectified.

So far the discussion has been limited to detail data sets and the relationship between detail data sets and master data sets. Master data sets also merit discussion. IMAGE permits two types of master data sets, the automatic master and manual master. For both types of masters, only one data item in the data set may be designated as a search item with that item being related to a maximum of 16 detail data sets. It is also possible to have an item in a master data set multiply index the same detail data set. There are several differences between automatic and manual masters. Manual masters may contain more than one data item and may exist without referencing any detail data sets, whereas automatic masters may contain only the search item and cannot exist without a related detail. Manual masters are denoted "manual" because they require entries to be manually added to the master before data may be added to related details, whereas automatic masters automatically add entries (if necessary) to masters when entries are added to related details.

IMAGE (ACCESS METHODS)

Four methods are used for accessing data of IMAGE data sets: serial, directed, chained and calculated read. The serial read retrieves the next entry in a data set, the directed read retrieves an entry using a record number, the chain read retrieves the next entry of a chain and the calculated read retrieves a master data set entry according to the value of a search item.

KSAM (INTRODUCTION)

KSAM is an acronym for Keyed Sequential Access Method and is an information management sub-system of type indexed sequential. The KSAM sub-system is composed of a key file, data file, temporary file (for internal maintenance) and procedures for accessing the data in key order or chronological order. Each KSAM file requires at least one key with a maximum of sixteen. KSAM permits all keys to be modified in FORTRAN, BASIC, SPL; for COBOL, one key, the primary key, may not be modified. The record structure of a KSAM file is defined using a system program.

KSAM (STRUCTURE)

In order to implement an indexed sequential access method, entries in the file must be retrieved in key sequence beginning at some key value. A B-tree structure is used to implement KSAM. KSAM requires all keys to be duplicated in the key file with appropriate arrangement of key values according to definition of B-trees. A B-tree is a collection of nodes with more than one key per node (KSAM requires at least four) linked to form a tree. The upper most level of the tree is the root and the lower most level is the leaf. At any level (except the leaf) if there are N keys defined, then the structure requires N+1 pointers to lower levels. For each key (in one of the nodes between root and leaf) there is a pointer to keys less than itself and a pointer to keys greater than itself. Also associated with each key is a

value. In this manner, the key file is used to locate the first element in the data file having a specified key value and retrieves elements sequentially by key from that point.

In order to access data by key, the key file is accessed to locate the first entry in key sequence with matching key or approximate matching of keys. The data file is then accessed to retrieve the data entry. For retrieval of entries in key sequence after the first entry is located, both the key and data files are accessed.

KSAM (ACCESS METHODS)

KSAM permits two types of access methods for retrieving data. These are keyed access and chronological access. For keyed access, both the key and data files must be accessed which requires the tree structure to be traversed. For chronological access, only the data file is accessed which in the general case retrieves data in a random order.

IMAGE FEATURES

Provides logical relationships between data items of detail data sets and manual data sets. This feature allows easy definition of complex data element relationships.

Permits an element to be retrieved when key value is known or all values in the data set with the same key value.

Data inquiry and update sub-system QUERY can be used to generate a limited scope of reports and can be used to add, delete and update entries.

Requires all data set space to be initially allocated.

Restricts functions which can be simultaneously performed on the data base. Permits modify with concurrent modify access of data base.

Provides security at the data item level, which can be used to restrict access to critical items while permitting access to non-critical items of a data set.

Modifications require the data base to be unloaded, purged, redefined, built and reloaded using the DBUTIL and DBSCHEMA system programs.

All data items except search and sort items may be updated on the current entry.

KSAM FEATURES

Permits definition of sixteen keys of which one is primary and up fifteen are alternates.

Access can be in key or chronological order.

Entries in the data set are not deleted but are flagged as deleted, which permits deleted data to be retrieved. Since deleted records are not reused, it is possible to have the data portion of the KSAM file full when few entries are in the key file. This is an extreme case where many deletions have been made.

An option of FCOPY permits the KSAM file to be copied using any key or permits the file to be copied in chronological order. When records are copied in key order, records which have been deleted are not copied.

Key file is fully allocated but data file need only be partially allocated.

Keys must occupy contiguous set of bytes with overlapping of keys permitted.

Permits modification of all keys (except in COBOL).

Keys can be defined to have duplicate entries or may be defined to be unique.

Permits temporary file status and opening with all data deleted.

SUMMARY

The preceding section lists some of the features of KSAM and IMAGE. IMAGE and KSAM lie at opposite ends of the spectrum as far as data set relationships and ability to retrieve data in sorted order are concerned. IMAGE provides an easy way to handle data sets which contain many interrelationships whereas it does not efficiently support storage and retrieval of data in sorted order. For IMAGE, if data is to be retrieved in sorted order, the data should be copied from the data base to a file and sorted. KSAM on the other hand requires all interrelationships between data sets to be handled at the application software level but supports retrieval of data in sorted order efficiently. When updating entries in an IMAGE or KSAM system, the amount of time required for the update is dependent on the number of relations associated with the data set (for IMAGE) and the number of keys (for KSAM). The choice of KSAM or IMAGE is primarily dependent on the manner in which data is to be retrieved. By including both information management systems in a system configuration, user defined information management software is virtually eliminated.

COMMON ERRORS (IMAGE): rf1

The current position in an IMAGE data set is important in most intrinsics: DBUPDATE and DBDELETE operate exclusively on the current position; DBGET (modes 2,3, 5,6) reads the "next" record relative to the current position. Many operations change the "current" position and to overlook one of the cases is very easy; examples follow:

You have done a DBGET on a manual master and are holding the data to update it later. You then do a DBFIND (with a different key value) on a detail data set which is indexed by the same master. This changes the current position in the master data set so that when you DBUPDATE the manual master you will get an ERROR 41 (critical item).

You are reading sequentially through a manual master (DBGET, mode 2). If you do any DBPUTs or DBDELETEs as you are scanning, you may move your position and miss entries or read them twice. This is because records in the manual masters are often moved as the result of a DBPUT or DBDELETE. The proper way to do this is to copy the key values into a file, then process from the file.

You are scanning a detail data set (DBGET, mode 2). This works fine the first time, because the "current" position after DBOPEN is at the beginning of each data set. However, the second time the program scans, it must rewind the data set first (DBCLOSE, mode 2 or 3).

You have done a DBFIND on key value A. After a mode 5 DBGET, you DBDELETE the record, change key to B and DBPUT it. How do you get the next record on the chain of A? You cannot just do a DBGET mode 5; you must first do another DBFIND with A. That is because DBPUT changes the current position.

COMMON ERRORS (KSAM):

When updating a key which cannot be duplicated, changing to a value which already exists will not update the key. You should first do a read by key to determine if the key already exists.

ERRORS (IMAGE AND KSAM)

There arises a problem when an IMAGE data base of KSAM file is being locked. The basic problem is forgetting to unlock the data base or file; this can lock up all terminals. There are several common ways to get into this kind of a "deadlock"; examples follow.

You lock the data base or file, but don't unlock it in every case. The most frequent place to overlook is an error exit. This example is a good argument for structured procedures with only one entry and one exit. You lock upon entry and unlock upon exit.

Another source of lockups is the "BREAK" key. If the user hits this key while the program has the data base or file locked, it will remain locked until someone types "RESUME." The "BREAK" key can be disabled through FCONTROL and should be.

If the program opens the data bases or files with locking, it must be PREPed with CAP=MR (multiple RINs). Unless the programmers are careful, it is easy to get into a deadlock where terminals are externally waiting for each other. This can be avoided by always locking the data bases or files in a certain order and unlocking them in the reverse order.

All programs which lock a data base or file should normally be run at the same priority level. This is especially important on Pre-Series II machines, where a low priority process can lock out high priority processes for an unlimited time.

References:
rf1 - Robert M. Green
President, Robelle Consulting, Ltd.
#130-5421 10th Avenue
Canada V4M 3T9

● ● ●

HP 3000 With Front-End Processor

On-line, Real-time: Keeps up-to-the-minute track of Correctional Institution Activity

by Elias Zabor, Editor

With a daily population of 4,000 individuals, approximately 400 of whom go to court daily (and sometimes as many as 1,000 after the delays of a long weekend) and with about 55,000 persons annually entering the Cook County Department of Corrections, a sophisticated management system is a necessity.

The system, called CIMIS (Correctional Institution Management Information System), utilizes MCS 3000 developed by Systems Research, Inc. of East Lansing, Michigan. It meets the requirements stipulated by the staff of Cook County Department of Corrections, under the direction of Sheriff Richard J. Elrod, and by J. David Coldren of the Illinois Law Enforcement Commission and director of the CIMIS project.

An earlier manual system augmented by monthly batch reports had been of little value in helping the line decision-makers (e.g., Captains, Sergeants, Lieutenants) in their daily work. Additionally, much was lacking in communication for a great deal of information which needed to be accessed at a moment's notice.

The goal of the institution was to have 64 terminals and eventually as many as 120. In addition to the need for reliability, the system desired would support transaction processing, include a data base system, have a proven operating system, employ high-level languages close to ANSI Standards and support a large terminal network with a high degree of data communication flexibility. Moreover, expansion of the system was expected to include the business office, food services, etc.

Systems Resarch, Inc.[1] primarily engaged in the development of software systems oriented to on-line processing, had already developed MCS 3000, a transaction processing and terminal network control system software that considerably expands the terminal capability of the HP 3000 System; in CIMIS, Systems Research, Inc. has developed an expanded capabilities system, without mixing hardware vendors and without modification of any HP software.

The CIMIS system at Cook County consists of an HP 3000, with an HP 21MX computer as a front-end communications processor. The 21MX with proprietary software and firmware insulates application programs on the HP 3000 from the physical terminal environment, handles message queuing and routes messages to terminal devices or user programs (these can even run on as many as four HP 3000 systems) and includes the required interfaces.

[1] *Note - Systems Research, Inc., 241 Saginaw, East Lansing, MI 48823*

The system permits up to 208 point-to-point terminals or 40 multi-drop data lines to be interconnected throughout the facility (presently 41 HP 2640 terminals are in use). The SRI transaction processing software system enables CIMIS to handle up to 60 transactions per minute and provides a modular structure for application programming, enabling application software to be written as if for a single user and terminal of unspecified type.

Hardware includes the HP 3000 Computer System with 512 Kbytes of internal memory, an HP 2888 50 Mb disc drive, and two HP 7905 15 Mb disc drives. The HP 21 MX employed as a front-end processor contains 64 Kbytes of internal memory. 41 HP 2640 Video Display Terminals are connected through dedicated telephone lines to the HP 21 MX front-end processor. Additionally, HP 2644 Video Display Terminals are used for program development.

The CIMIS project (Correctional Institutional Management Information System) was designed by seven programmers from the Illinois Law Enforcement Commission (ILEC) over a period of four years, and the actual coding of the Cook County Department of Corrections (CCDOC) facility entailed about a year's work. The current staff consists of the CIMIS manager, one systems analyst, one user trainer, four others for field support, and a computer operator.

USE OF THE SYSTEM

Four separate divisions located on a 56-acre site are served: one division deals with felons whose bond exceeds $10,000; another deals with those inmates whose bond assignments are lower, and also includes younger and older offenders. A third division deals with women only; Division IV is for those in the work release program. Part of the facility incorporates a 1,000 man central reception building, a gymnasium, a hospital, and a criminal court building.

As can easily be seen, to deal with the complexity of humane management, and the safety and control of such a large and transient population involved in so many facilities, requires a comprehensive and extensive tracking and control system.

Each entering inmate is assigned an ID number, and all records and medical history relating to the individual are entered. Upon arrival at the tiering area, the officer in charge makes a cell assignment decision based on the information called up at the officer's CRT; among the factors that affect assignment are health status, sexual offenses, gang membership, etc. At several locations along the way, there are check-points where terminals are used to enter the inmate's arrival there, or to update a file. This information automatically ripples through the data base, updating all pertinent files.

File Security

Authorized users are given access to particular files, and/or the ability to alter them is assigned and controlled by MCS 3000, on a transaction-by-transaction basis, at both the user and terminal level. Individuals can be tracked to the moment, and any aspect from their personal history can be recalled.

Generating Forms

One of the major uses of the system is for court attendance records. As many as 500 inmates may need to attend court in a single day, and some may have been assigned to several court appointments on the same day; in this event, the courts must be notified of conflicts. A CRT terminal in the records office is used to enter the new information on a formatted screen. The system then dynamically updates all records and produces listings of the court calls for each individual court branch; the sheets are distributed to tier officers and block sergeants. Additionally, the system generates dispatch sheets for use by the transport office.

Other summaries are also produced. A population analysis, which previously consumed four work days for eight staff members, is now produced in one-half-hour of system time. Additionally, each morning an inventory of the population is produced to determine the food and sundry needs of the facility. Weekly summaries sorting the inmates according to various categories, e.g., length of sentence, indictments, etc., are used by several State Agencies.

The ability to rapidly locate individuals and their historical information (whether for prompt access to visiting legal representatives or for medical attention) decreases tension and makes it possible to provide for just and humane treatment. Information is available within three to five seconds on active terminals which simultaneously access a large IMAGE data base, adding significantly to security and overall efficiency of operation.

• • •

Tips and techniques

SPL Supports the Guarded Construct IF-FI*

by Jung Pyo Hong
University of California
Los Alamos Scientific Laboratory
Los Alamos, NM 87545

Many programming steps are most easily expressed using a construct that allows the programmer to code repeated *else if*. Some languages have implemented such a structure, i.e., *elif* in ALGOL 68, (Ref. 1). SPL supports it in a subtle way. This article presents how one can make and use the repeated *else if* construct in standard SPL.

The repeated *else if* construct with special semantics appears in the literature. Dijkstra (Ref. 2) uses construct (1).

(1) *if* $B_1 \rightarrow S_1 \square | B_2 \rightarrow S_2 \square \ldots B_n \rightarrow S_n$ *fi*

B_i is a boolean expression and S_i is a statement sequence. S_i is said to be guarded by B_i and $\square$ is used as a delimiter. The meaning of a sentence that has the form of (1) is (a) perform the first statement sequence S_i whose boolean guard, B_i, is *true*; (b) if no boolean guard is *true* than abort.

*Work performed under the auspices of the U.S. Department of Energy.

An example of the use of *if-fi* can be written with "perform" taking the place of " → " and "elseif" taking the place of " □ ":

```
(2)  If      p<=u perform c:=0; d:=5; e:=3
     elseif  p<=v perform c:=1; d:=2; e:=4
     elseif  p<=w perform e:=2;
             If    q<=x perform c:=2; d:=47
             elseif q<=y perform c:=3; d:=12
             fi
     fi;
```

Program steps in (2) will compile directly in standard SPL with the following definitions:

```
(3)  Define perform = then begin=,
             elseif  = end else if=,
             fi      = end else quit(0)=;
```

The above program steps (2) can be written in direct SPL as in (4), but would require prolixity of *begin end* pairs obscuring the flow of the program and interfering with readability.

```
(4)  If      p<=u then
             begin
             c:=0; d:=5; e:=3
             end
     else if p<=v then
             begin
             c:=1; d:=2; e:=4
             end
     else if p<=w then
             begin
             e:=2;
             If      q<=x then
                     begin
                     c:=2; d:=47
                     end
             else if q<=y then
                     begin
                     c:=3; d:=12
                     end
             else    Quit(0)
             end

     else    Quit(0);
```

Termination of the program when all guards are *false* encourages the programmer to consider carefully all possible guard conditions. However, it is often convenient to have a default statement sequence. That is, when all the guards, B_i, are *false* then perform the statement sequence S_t, as in (5).

(5) *If* $B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \square \ldots B_n \rightarrow S_n$ *true* $\rightarrow S_t$ *fi*

Because this construct occurs so frequently, another version (6) of (5) that increases the readability of programs is considered.

(6) *If* $B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \square \ldots B_n \rightarrow S_n$ *otherwise* S_t *endif*

The ending *endif* is used to distinguish the construct from the *if-fi* construct: *endif* is not abortive; it only finishes off the *if* statement.

An example of the form with default statement sequence follows:

```
(7)  If        a<0 perform c:=1; d:=2; e:=4
     elseif    a=0 perform c:=3; d:=12; e:=2
     otherwise             c:=7; d:=2; e:=14
     endif
```

This will compile directly in standard SPL with the definitions of (8).

```
(8)  Define perform   = then begin=,
             elseif   = end else if=,
             otherwise = end else begin=,
             endif    = end=;
```

The syntactic description of the constructs are in the style recommended by N. Wirth (Ref. 3), with deletion of annoying quote marks. The braces { } indicate repetition, i.e., { a } stands for ∅ | a | aa | aaa | . . . (∅ stands for null).

```
( 9) if-fi     = If      condition-clause perform state-seq
                 elseif  condition-clause perform state-seq
                 fi.

(10) if-endif  = If      condition-clause perform state-seq
                 { elseif  condition-clause perform state-seq }
                 { otherwise state-seq }
                 endif.

(11) State-seq = { statement { ;statement } } .
```

(12) *Statement* and *condition-clause* are described in the SPL Reference Manual.

Perform, elseif, otherwise, fi and *endif* are defined above in (3) and (8).

References:

1. F. G. Pagan, *A Practical Guide to Algol 68*, (John Wiley & Sons, New York, 1976), p. 83.
2. E. W. Dijkstra, *A Discipline of Programming*, (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976), pp. 33-34.
3. N. Wirth, *"What Can We Do About the Unnecessary Diversity of Notations for Syntactic Definitions?"* - Federal Institute of Technology (ETH), Zurich.

● ● ●

FORTRAN OPTIMIZATION

by Christine Morris, Hewlett-Packard
General Systems Division
5303 Stevens Creek Blvd.
Santa Clara, CA 95050

Optimizing the use of a compiler involves, in general, minimizing time and memory usage at both compile-time and run-time. The amount of time used at compile-time is generally a function of the design of the compiler, i.e., how much optimization, number of passes, etc., and the user does not have much control over the matter. However, in FORTRAN/3000 the compile-time memory usage can be influenced by the user.

The issue of compile-time stack usage is important due to the fact that FORTRAN/3000 uses 13 bits to link entries in the symbol table. Thus, excessive use of memory in the symbol table can result in a compiler error-Symbol Table Overflow since the entire symbol table cannot exceed 13 bits or 4095 words. Each entry in the symbol tables takes four words plus the name (= of characters /2 = = of words needed to store the name). Each subroutine, system intrinsic and function, whether in the program or declared external, also requires a symbol table entry. All variables and arrays, whether local or in common, require an entry. There is also a symbol table entry for each Fortran intrinsic which takes ~450 words and is always present.

In addition to the symbol table entry, there is extra information kept within the symbol table, which also must be linked to the proper symbol table entry with 13 bits of pointer. For an array there is one word each for the = of dimensions, array size and each of the bounds. For a statement function, system intrinsic, or external procedure there are (= of arguments /2 +1, words required).

The symbol table also contains control entries. Each DO-LOOP creates a three-word entry, format labels require 10 words, other labels require six words, block data requires three words + (= of characters in name /2 +1), and constant entries require two words + length of constant (1 to 4 words). The symbol table contains a program unit entry. A main program uses 14 + (size of name /2) words and a function or subroutine uses 16 + (size of name /2) + = of parameters in its entry.

Other features will use varying amounts of space in the symbol table. CROSSREF uses one extra word for each symbol table entry for linking purposes. Data statements use symbol table space much as constant entries, approximately a symbol table entry for each variable equivalent along with a constant entry. Assigning values to the variable in the program will use less symbol table space but will use extra time while running.

The symbol table is reinitialized after each program unit (a program unit is a main program, a subroutine, or a function). Thus, each symbol table contains the local entries for the program unit and any common declared in that program unit.

To reduce the symbol table requirements, some or all of the following can be employed:

- Minimize the length of variable and array names
- Change variable references to array element references
- Modify large program units into more than one program unit, since the symbol table is initialized after each program unit
- Do not CROSSREF large program units unless necessary

All of this minimizing of stack usage is best employed to avoid the compiler error of Symbol Table Overflow. One must be careful however, about the restriction of 254 variables in COMMON. More than this number of COMMON variables will produce a SEGMENTER error. If this occurs, either remove some variables from COMMON or use the new MORECOM' enhancement. This will allow more than 254 variable in COMMON at an expense of a 10 to 30% increase in code segment size and approximately a 0 to 4% increase in execution time.

Stack usage in a FORTRAN program can be optimized in several ways. Storage for this simple program is shown in Figure 1.

Changing the order of declarations to COMMON C(3), A, B produces storage as shown in Figure 2. Thus, the order of declarations of both common and local variables matters in the amount of space used. Also, note that local variables are Q-relative and are active only when that program unit is active, i.e., the stack gets reused. However, common variables are allocated DB-relative and always are present. Thus, moving large blocks of storage from common to local storage better utilizes the user's stack.

Figure 1

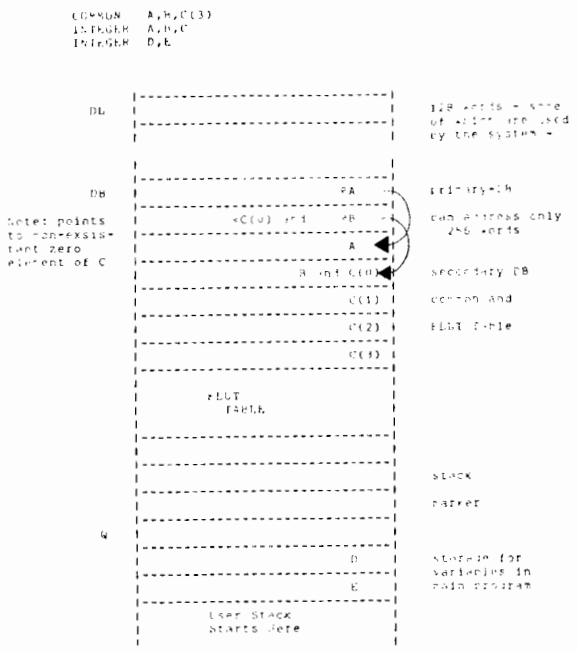
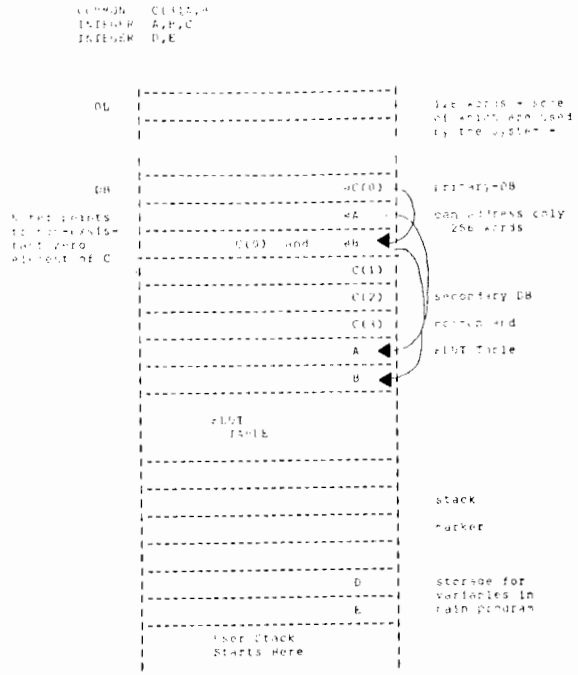


Figure 2



Run-time efficiencies in a FORTRAN/3000 program can be gained in many ways by the avoidance of certain constructs. In particular, avoid external PCAL's as much as possible. Many such calls are generated by using FORTRAN intrinsics or certain language constructs. An example is:

```
Character *20 Y
Character *  2 Y
Y = "YY"
X =  Y
X [1:2] = Y
:
:
```

The first assignment X=Y will generate an external call to BLANKFILL' to fill in the rest of X with blanks. In contrast the second construct X [1:2] = Y will leave the rest of X undefined, but not generate the PCAL.

Also, the Formatter in Compiler Library is used for I/O. Using Format statements may cause more calls than necessary to the Formatter. In general, an I/O statement causes various calls to different entry points in the Formatter. When in doubt decompile the program and look at the number of calls to the Formatter.

Mixed-mode expressions are inefficient because the conversion generates code which can cost surprisingly large amounts of time particularly when done inside a loop.

Proper segmentation, so as to avoid as far as possible, PCAL's to an other segment can also decrease execution-time. Segmentation can also decrease the amount of memory used by the code portion of a program.

Other languages can also improve run-time efficiency by making better use of the hardware instructions. An example is the move instruction in SPL which very efficiently initializes an array.

In general, knowledge of how the Fortran Compiler works at run-time can influence the structure of a Fortran program and greatly increase its run-time efficiency.

• • •

Stack Overflow in 'LIBRN'

by John Page, Hewlett Packard
General Systems Division
5303 Stevens Creek Blvd.
Santa Clara, CA 95050

In some newer versions of MPE on the Series II, the librarian program 'LIBRN' may get a stack overflow abort when doing a '=PUT'. This is not a bug per se; it is caused by MPE using slightly more stack during an FOPEN than before, and you only have just enough stack in LIBRN. To fix it, get the source from the contributed library, recompile it and prep it with a larger MAXDATA:

```
:PURGE LIBRN
:SPL LIBRN.SOURCE.LIBX,,$NULL <<X=LIB
                                VERSION>>
:PREP $OLDPASS,LIBRN;MAXDATA=8000
:SAVE LIBRN
```

8000 ought to be enough. If not, increase MAXDATA until the problem goes away.

• • •

USING COBOL: A Guide for New Users of HP 3000 Computer Systems

by Sandy Martensen, Hewlett-Packard
General Systems Division
5303 Stevens Creek Blvd.
Santa Clara, CA 95050

A new two-color, spiral bound manual similar to the *Using Files* guide has just been printed. It leads COBOL programmers step-by-step through the process of developing a COBOL program on the HP 3000. The guide begins by describing how to create and modify source code, providing examples such as the following:

- creating and editing a source file
- creating and using an Editor USE file to format a 264x terminal screen with COBOL column headings
- creating a copy library (in an MPE or KSAM file) and creating the source programs that use it.

A section on compiling includes examples of compiling to check syntax and compiling while using a copy library and a maintenance file.

USLs, RLs, SLs, and other Segmenter mysteries are described with examples showing you how to examine and manipulate their contents. The guide illustrates methods of preparing and running programs using these libraries. Other examples show you how to

- request and interpret a load map
- adjust the size of data areas
- perform file operations such as using variable length records, passing a file number to an SPL program, and using special forms on the line printer.

Examples in the last section illustrate debugging. They show you how to examine and modify locations in the data stack using PMAPs, LMAPs, and either the :SETDUMP command or interactive DEBUG commands. MPE operating system, file system, and COBOL terms are defined in a glossary.

The manual part number is 32213-90003 and the price is $6.50. Order from your HP Sales Office.

• • •

Mark Sense Cards Applications in Manufacturing

by Alic Rakhmanoff, Hewlett-Packard
Boise Division
P. O. Box 15
Boise, Idaho 83707

You have probably noticed that many manufacturing companies are very reluctant to dispose of their card-oriented applications. In fact, most of them consider those the most reliable and economical way of recording data for specific applications like stock control, production tracking, inventory control, parts requests, etc.

That is why many companies are implementing new applications with pencil-marked cards combined with computer printing (turn-around document application). Several Hewlett-Packard divisions have been using this system within the last year. Marked cards read by a 7260A Optical Mark Reader connected to the HP 3000 are used for pull decks, stock control and cyclic inventory control.

When production lines need parts from the stockroom for work-orders issued on the HP 3000, a program in the HP 3000 generates a printed continuous fan-fold form. Information such as part number, order number, location to deliver parts, quantity, units of measure, part description, color code, higher level assembly part number and date are printed for each work-order. All this information is useful to the person who will do the pull and deliver the parts to the production line. It is also useful for the people on the production line. Also, a reference number is printed with black marks (OMR characters). This unique number associates the work-order number and the part reference in the HP 3000 IMAGE data base.

After being printed, the continuous form is separated into 11½ inch cards by a burster (bursting machine). Each card has two parts: a standard tab card 7 3/8 inches long and a tear-off part. One card represents one work order to pull a specific number of parts from the stock.

The person in charge of the stock takes each card and according to the printed instructions, pulls the required quantity of the needed part, and then marks a personal code (two digits), the quantity pulled (which can be less than the required quantity) and a check digit (which is marked on each box containing the parts).

The individual then separates the tear-off from the card and delivers the parts with this tear-off which is used as a hard-copy document. The card is then read by the HP 7260A Optical Mark Reader connected to the HP 3000. If the card has been incorrectly marked (wrong check digit or invalid quantity) the card is rejected in the "select" hopper of the 7260A. The operator can immediately check the pull, and by carefully erasing the card, correct the marked data. Cards mismarked by operators are usually less than 5 per thousand.

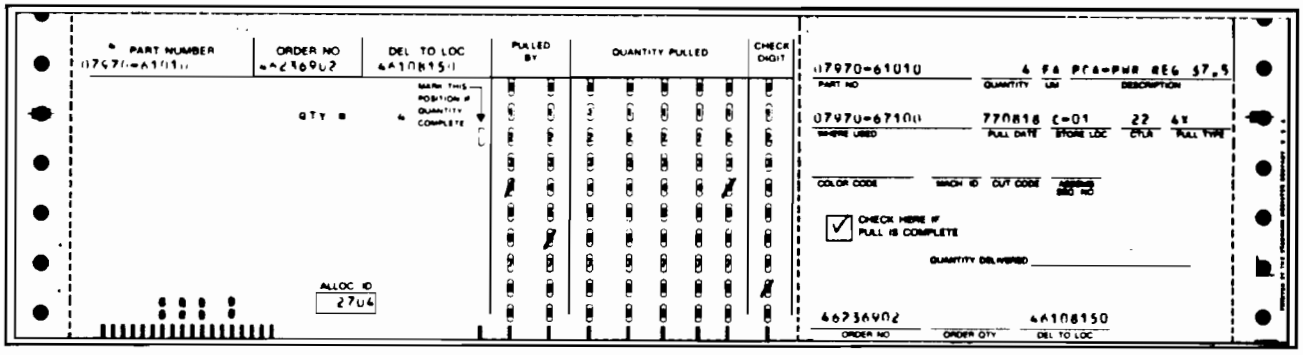
The same forms are also used to do the cyclic physical inventory of the stock as the 7260A reads the marked data and also the printed data (OMR characters). The printed data enables the computer to retrieve the work order from the database and to update the stock, generate back orders if necessary and even to print order forms or invoices.

Continuous fan-fold forms must be correctly aligned on the HP 2613/17/18A printer (using alignment boxes where those are provided). A good quality ribbon must be used, (although mylar ribbon is not necessary) in order to print a dark mark for the OMR character.

HP printers with free option 002/003 having the OMR (or "slug") character give a good black mark for this character. Otherwise overprinting is recommended. Many HP divisions overprint three times using "left bracket, 1, right bracket" characters with excellent results.

● ● ●

SAMPLE CARD



Stand-Alone HP 7221A Plotter Operation

by Jason M. Goertz
Whitman College Computer Center
Walla Walla, WA 99362

Introduction

The HP 7221A Graphics Plotter is one of the newest and most versatile plotting devices developed by HP. Programmatic control of every function, including pen color, macroinstructions, and built-in character sets give the plotter its versatility. These features allow greater ease in programming, and thus the HP 7221A has found its way into a larger number of disciplines, particularly those outside of the so-called "hard" sciences, such as economics, sociology, and other social sciences. Programmatic use of the plotter is made easier by the use of PLOT21, a software package that allows the 7221A to be controlled by a series of subroutines written in FORTRAN.

By configuring the HP 7221A as an output device such as a line printer, any user can access the plotter by calling the appropriate subroutines and intrinsics. Access to the HP 7221A as such an output device is easily accomplished, while still leaving the plotter configured as a terminal, so that use in series with a terminal is still possible if desired. This is done by opening the plotter port as a file, allocating the port as a terminal, then directing output and receiving input from the plotter. The port is finally closed, as any file would be.

Configuration and Program Example

The plotter port should be configured with the default termtype of the installation, using IOTERMØ as the driver. Programmatically, the port is opened by using the FOPEN intrinsic, and allocated as a terminal with FCONTROL. Plotting routines are then performed normally, with or without PLOT21 software. The file is closed with the FCLOSE intrinsic.

Example:

```
$CONTROL USLINIT, FILE=7-8
        PROGRAM EXAMPLE
        CHARACTER*14 ARRAY
        DIMENSION LARRAY(7)
        EQUIVALENCE (ARRAY,LARRAY)
      CALL PLOTOPEN(IFILENUM)
      CALL PLOTS(1,7,8)
      CALL SETIN
      CALL CSIZEA(1.,.667,Ø.)
      ARRAY="THIS IS A TEST"
      CALL SYMBOL
                (1.,3.,1.,LARRAY,Ø.,14)
      CALL NEWPEN(Ø)
      CALL PLOTOF
      CALL PLOTCLOSE(IFILENUM)
      STOP
      END
```

```
      SUBROUTINE PLOTOPEN(IFILENUM)
      SYSTEM INTRINSIC FOPEN,FCONTROL
      LOGICAL PARM
      PARM=%ØL
C
C  CALL FOPEN TO OPEN PLOTTER PORT
C
      IFILENUM=FOPEN(,%4Ø4L,%1Ø4L,
                                  -2ØØ,"31")
      IF(.CC.)1ØØ,2ØØ,1ØØ
  1ØØ DISPLAY "PLOTTER NOT AVAILABLE"
      STOP
C
C  CALL FCONTROL WITH 37 AS CONTROLCODE
                        TO ALLOCATED PLOTTER
C  AS A TERMINAL
C
  2ØØ CALL FCONTROL (IFILENUM,37,PARM)
      IF(.CC.)3ØØ,4ØØ,3ØØ
  3ØØ DISPLAY "FCONTROL FAILED"
      STOP
  4ØØ RETURN
      END

      SUBROUTINE PLOTCLOSE(IFILENUM)
      SYSTEM INTRINSIC FCLOSE
      CALL FCLOSE (IFILENUM,%Ø,%Ø
      RETURN
      END
```

The FOPEN intrinsic is called with the following parameters:

FOPEN(,foptions,aoptions,recsize,dev)

	OPTION	BITS
foptions	New file	14:2 = ØØ
	ASCII	13:1 = 1
	Default Filename	1Ø:3 = ØØØ
	Fixed reclength	8:2 = ØØ
	CCTL	7:1 = 1
	Allow :FILE	5:1 = Ø
aoptions	Read/Write	12:4 = Ø1ØØ
	No Multi-record	11:1 = Ø
	No FLOCK	1Ø:1 = Ø
	Exclusive access	8:2 = Ø1
recsize	2ØØ bytes	
dev	Logical Device Number of Plotter for particular configuration	

I/O Methods

After opening, two methods may be used to access the plotter. The first is assignment of file equations, via the COMMAND intrinsic:

```
:FILE FTN07;DEV=31;REC=-200,1,F,ASCII;CCTL
:FILE FTN08;DEV=31;REC=-200,1,F,ASCII;CCTL
```

(All parameters in these equations are duplicated in the FOPEN call. Specification of the proper Logical Device Number for the particular installation is the only one necessary, i.e., "DEV=devnumber"). These assign the FLUT files 7 and 8 to the plotter. The standard files referenced by the PLOT21 software are 5 and 6, which use SSTDIN and SSTDLIST, respectively. These should be avoided, since these files are used for communication to the user's terminal. If file equations are used to reference the plotter, the IFILENUM parameter from the calling statement may be eliminated, and passed instead by a LABELED COMMON statement.

A second way is to return the file number from the FOPEN intrinsic, and use the negative value as the FLUT file:

```
CALL PLOTOPEN(IFILENUM)
CALL PLOTS(1,-IFILENUM,-IFILENUM)
.
.
.
(ETC)
```

The problem with both of the above methods is that labeling the graph using:

```
.
.
CALL LABON (14)
DISPLAY "THIS IS A TEST"
CALL LABOFF
.
.
(ETC)
```

is not possible unless a :FILE equation is assigned for FTN06, since the Fortran DISPLAY statement uses this file. Without the :FILE command, DISPLAYS would appear on the user's screen, not the plotter.

By use of the PRINTOPREPLY intrinsic, a reply from the operator may be solicited to make sure the plotter has paper loaded and is ready to receive output. This would be useful if the plotter were being accessed by a number of different people physically distant from the plotter. This could also be done by using the formsmessage parameter of the FOPEN intrinsic, not shown in the above program example.

Conclusion

The possibilities that stand-alone operation offer are almost limitless. Batch access is now possible, allowing the plotter to become the final output of a large statistical analysis job stream or similar application. In general the plotter can now be made available by any user, and thus increase its utility.

The examples shown above outline the basic techniques involved. A full, documented version of these subroutines will be submitted to the next version of the User's library. Any questions may be directed to:

Jason Goertz
Whitman College
Walla Walla, WA 99362
(509) 527-5417

● ● ●

HP 3000 System Backup Techniques

by Dr. John F. True
Director, Computer Center
The University of Tennessee at Chattanooga
Chattanooga, Tennessee 37401

Visualize this scene. The CPU on your HP 3000 series II suddenly halts with a hardware problem just hours before a major production job must be run. An attempted coolstart survives only long enough for you to realize that your system directory tables are scrambled. You say to yourself: "No problem, we'll just reload from our backup tapes."

You mount reel one of your most recent set of backup tapes and punch enable/load. The tape moves a fraction of an inch and rewinds. The loader is unable to read the boot strap on the front end of the tape. Again you say: "No big deal, let's try reel one of the set prior to this one." The same thing happens. After an hour or so you realize that MPE has allowed six sets of backup tapes to be written with bad boot straps.

Only hours before the production job must begin the system is loaded with an old MIT tape and you create one account with one user and restore only the necessary files. Somehow you get the production run out on time. It may sound like a far out story, but it happened to us. The 800 bpi tape drives that we use do not have read-after-write capability and MPE does not offer a verify option (like the HP 2000 Access HIB). Since that time we have developed some techniques for backups that we would like to share.

We execute a full SYSDUMP (date code 0) every night of the week. The job stream that submits the SYSDUMP also gives us a REPORT, LISTF @.@.@,2, MEMLOGAN and a FREE2 report. The last thing that the job stream does is resubmit a deferred job. The purpose of the deferred job is to change the JOBPRI allowing SYSDUMP to run at CS and then to actually stream the real SYSDUMP job. The real SYSDUMP file is lockword protected. Our night operators

are students and this technique allows us to keep the passwords to Manager.Sys private from them. The night operator simply alters the input priority of the deferred job and away it goes.

Users at various remote sites on campus enter a SHOWJOB command the next morning when they sign on to see if the backup job was introduced and deferred the previous night. If they see that the backup worked, they know they may file the previous day's work. Transactions from the previous day are paper-clipped but not filed until there is a tape backup.

We backup one 7920 drive and two 2888 drives to six or seven 2400 foot reels of tape each night. The job takes two hours and we alternate each of our two 800 bpi drives allowing one to run while the other is rewinding. It should be pointed out that we run approximately one and one-half shifts per day and the period between 8 and 10 p.m. is devoted to these backups. Even if we were to run twenty-four hours a day I would still allocate that much time for a full SYSDUMP . . . maybe I'm shell shocked. The two backup job streams mentioned above are listed here for your information.

```
>FROM=BACKJOB.BACKUP.ASSIST;TO=
!JOB BACKUP,MANAGER/LONE.ASSIST/RANGER;
                                    INPRI=1;PRI=ES
! JOBPRI  CS,ES
!STREAM BACKSYS.BACKUP
!EOJ
>FROM=BACKSYS.BACKUP.ASSIST;TO=
!JOB SYSBACK,MANAGER/RUBBER.SYS/DUCKY;
                                    HIPRI;PRI=CS
!JOBPRI  DS,ES
!REPORT  @.@
!FILE SYSDUMP;DEV=TAPE
!SYSDUMP *SYSDUMP
NO
0
YES
!LISTF  @.@.@,2
!RUN FREE2
!RUN MEMLOGAN
!STREAM BACKJOB.BACKUP.ASSIST
!EOJ
```

A partial or dated SYSDUMP would not be of much use to us since many of the data sets in our large student records data base do not get opened every day. A partial backed up data base with pointers going in directions from which there are not corresponding pointers would not be of much value. During our heavy on-line registration periods we do a STORE of the data base during the noon lunch hour when registration is shut down. We have found that although we handle hundreds of transactions daily, it is not necessary for us to bog down the system further by writing transactions to a log file. We simply use the "paper clip" approach to keeping the data ready to reenter if it should ever become necessary. During busy times the most we would ever lose would be four hours of work and on a normal day we could always restore the system as of the previous evening. As a final precaution we rotate one of our nine sets of backup tapes to an off-site vault once a week. In case of a total disaster, we could restore from a week ago. A tape analysis report program reads this set before it is sent to the vault to make sure all tapes are readable. About once a month, at a relatively safe time, we coldstart the system from a backup tape to make sure the boot strap is good.

It should be noted that the ASSIST (shown in the job stream listings above) account was created (after the horror story) to assist in starting and backing up the system and, if necessary, completely regenerating the system. The name was chosen so that these files would be on the first reel of a SYSDUMP tape set. Files in this account include:

1) "LISTGROUP @" and "LISTUSER @" files
2) Backup jobstreams
3) A STARTSYS program to initialize the operating system after an IPL

Any time accounting structure changes are made, a LISTGROUP @.@ and a LISTUSER @.@ commands are written to disc and then massaged with EDITOR. In the event of another catastrophic failure, the plan is to:

1) Find *any* MIT tape that will load, SYSDUMP a properly configured operating system to tape, and reload from it.
2) Create the ASSIST account and restore all files in it.
3) Print the group and user files and begin manually rebuilding the accounting structure.
4) Use the RESTORE command to recover files. The software error recovery would probably handle read errors that the microcode could not tolerate for the initial load sequence.

The STARTSYS program operates only from the console, and is a compiled Basic program. It reminds the operator of the correct OUTFENCE, STREAMS, and other commands. Then it streams a job that sets the QUANTUM and JOBPRI commands, ALLOCATE'S desired programs, streams the deferred backup jobs and a DPAN2 job. When the mount request for DPAN2 comes up, the system is ready for use. If no dump tape is available, a reply of "0" is given which aborts DPAN2.

We think that these techniques are worthwhile and would be pleased to discuss them further with interested parties. The STARTSYS program mentioned above is available upon request.

• • •

Lightning and Your Communication Lines

by Larry Hartge, Hewlett-Packard
General Systems Division
5303 Stevens Creek Blvd.
Santa Clara, CA 95050

Are you having problems with lightning induced noise on communication lines? If you are in the Colorado, New Mexico, Arizona or Northern Florida areas (the lightning intensity peaks in the U.S.), you undoubtedly are!

Regardless of where you are located, Lightning Elimination Associates of Downey, California (Phone: (213) 923-1268) may have the solution to your problem. In addition to providing a consulting service, they produce the following products:

Transient Eliminators	Prevent passage of unwanted electrical disturbances on signal lines (e.g., phone lines).
Surge Eliminators	Prevent passage of unwanted electrical disturbances on power mains (e.g., to power supply of 3000).
Lightning Warning Systems	Give warnings that lightning may strike.
Dissipation Array Systems	Prevent lightning strikes in any protected area.

General Systems Lab personnel attended an IEEE presentation by the President of Lightning Elimination Associates and felt that those users having communications line problems with lightning might consider purchasing the Transient Eliminators; these typically range from $35 to $70, and would be useful for protecting terminal (ATC) and data communication (SSLC, HSI) lines on site.

• • •

New Computer Security Booklet

Available Free From Computer Security Institute

by Elias Zabor, Editor

A twelve page, 4"x7½" booklet – *Guidelines for Establishing a Computer Security Program* – has just been published by Compuer Security Institute and is available (from them) on a complimentary basis.

Recognizing that most organizations have not had the *time* nor the *resources* to implement an effective computer security program, this booklet offers useful information and a practical framework for getting started.

The material served as the basis for a workshop presented at the Institute's 1977 Annual Conference.

Contact: John C. O'Mara
Computer Security Institute
Five Kane Industrial Drive
Hudson, MA 01749
(617) 562-7311

• • •

Contributed Library Corner

The New Contributed Library

by Wayne E. Holt
Whitman College
Walla Walla, WA 99362

In June a new version of the Contributed Library will be released. At first glance, it will seem to be similar to the previous versions, but closer inspection will reveal quite a few improvements. The name will be different - LIB instead of LIB4. There will be multiple accounts on the tape. You will find several new software tools to assist in using the Library. All this, and more, is the result of an evaluation process started at the Seattle meeting in September of last year.

At that time, a Library Committee was formed to evaluate the then current structure of the Library and submit recommendations to the Executive Board on how to improve the Library. Such a report was filed in December; in January, the Executive Board met and approved an implementation plan that Whitman College submitted with the Committee report. The major points of the report were as follows:

1. The Library is one of the keys to the future growth and success of the Users Group. In its current state the Library does not adequately provide for the needs of all Users, and the time for a restructuring had come.
2. The Library needed care and attention that could only be provided by a regular staff.
3. Better documentation and general information was crucial to making the Library more usable. Especially important was some sort of grading scheme.
4. Library distribution had to be done on a timely and dependable schedule.

Whitman College volunteered its site for a Library Cleanup Project. The restructuring of the Library and increased level of documentation constitutes the first major goal. First release of the revised and improved library is scheduled for late June.

What is the new library structure? Basically, all of the previous LIB's will be consolidated into a single LIB. Only the most current version of a given program is retained. A definitive policy for library programs will be issued with the June release and will cover naming submittal conventions. Other new accounts have been established as well, to provide better services to you:

INFOBASE This account will be used to store all "information" about Software available on the HP 3000 series systems, as well as information about the User Group. In addition, the programs necessary to maintain and retrieve this information will be included. Some of the information will be:

a. An IMAGE data base: containing a comprehensive set of data about all software available-contributed (free), clearing house (S), and HP supplied. It will be structured for easy QUERY access.
b. A KSAM data file – Similar to A.
c. A sequential data list file – Similar to A. Print ready, with three to four abstracts per page.
d. Roster: User Group membership data.
e. Release message: A print file with note about the current library release.
f. Forms: A standard submittal form ready for printing.

LIB — This account will contain the contributed software programs and routines, using a group structure similar to the current Library. A major change: there will only be one LIB (with revision numbers) instead of "new" LIB's every six months. A modified or enhanced contribution will actually replace the old contribution, unless the changes are significant. In this case, a new name would be assigned.

ACCOUNT "PACKAGES" — These accounts would consist of major contributions at the "system" level that do not readily fit into the LIB structure. The group structure for these accounts would be up to the submitter.

This new type of structure provides much more flexibility for future growth, and allows you to tailor the Library to meet the needs of your shop. Along with the new structure, new software will also be provided to support use of the library. This will include sophisticated "Restore" programs as well as variety of informational reports from INFOBASE.

This sounds like a pretty ambitious project, and frankly, it is. It is also a very realistic one, in that it addresses the problem in a very flexible manner, allowing for future enhancements dictated by your needs. It will not be finished overnight, or in fact, ever. It constitutes an on-going project that will be necessary as long as there is a Users Group.

Ultimately, the project (as well as the Group) will fail unless the members stand behind it and support it. In terms of the Library, this means sending in contributions, sending in "fixes" to things that don't work, taking the time to review items in your specialty area and telling us how they perform. It also means informig us when we make mistakes and allowing us to rectify them.

As I write this article, I can't honestly say how much will be accomplished by what date. Our goals are as follows:

Establish INFOBASE	June '78
Restructure Library	June '78
Develop new supporting software	Dec. '78
Grade programs	Dec. '78
"Fix-up" Ailing programs	Mar. '79

It is hoped that this article will serve to whet your appetite for the June release!

● ● ●

Computer Publications

by Wayne E. Holt

There are two publications produced at Government expense that might be of some interest to members of the User's Group:

Computer Science & Technology Guide to Computer Program Directories is a master listing of computer user organizations and computer program libraries. It is listed by organization name and indexed by subject.

NBS Special Publication 500-22
U.S. Department of Commerce
Information Technology Division
Institute for Computer Sciences Tech.
NBS
Washington, D.C. 20234

COSMIC (Computer Software Management and Information Center) Guide to Available Programs is a partial listing of software abstracts (with prices) of various programs developed for NASA and now available to the public at distribution cost. The bulk of the programs is in IBM Fortran and covers a wide range of scientific applications.

COSMIC
NASA
112 Barrow Hall
University of Georgia
Athens, Georgia 30602

● ● ●

The Clearing House

Laboratory for Computer Graphics and Spatial Analysis

The Laboratory is part of the Graduate School of Design at Harvard University. We are very interested in establishing a working relationship with a current HP 3000 user to function as a conversion center for the Laboratory's computer mapping programs. We currently have conversion centers for CDC, DEC, and Burroughs. Our experience with these centers has been excellent to date.

Because of our non-profit status, we cannot offer any great amount of compensation; however, all conversion centers receive free of charge the following software packages which usually have a price tag of $7,060.00 for universities and governmental agencies and $10,610.00 for commercial clients:

SYMAP	– Choropleth or isopleth mapping program, line printer output
SYMVU	– Perspective mapping program, pen plotter output
GRID	– Choropleth mapping program (grid based), line printer output
CALFORM	– Choropleth mapping program, pen plotter or CRT output
ASPEX	– Perspective mapping program (interactive), CRT output
POLYVRT	– Cartographic data base conversion program, pen plotter output
DOT.MAP	– Choropleth mapping program, CRT output
PRISMAP	– Cartogram mapping program, CRT output

For your information, "International Users Conference on Computer Mapping Software and Data Bases: Application and Dissemination," will be conducted from July 23rd through July 28th at the Hyatt-Regency Hotel in Cambridge, Mass. For additional conference information, you may contact Allen H. Schmidt, the Conference Chairman at 520 Gund Hall, 48 Quincy Street, Cambridge, Mass. 02138. Telephone: (617) 495-2526, Telex: 92-1496. (Mr. Schmidt is Executive Director of the Laboratory.)

If you wish to respond to our inquiry requesting an HP 3000 site to act as a conversion center for the Laboratory's programs, please respond to:

William G. Nisen
Director of Distribution Services
Laboratory for Computer Graphics & Spatial Analysis
520 Gund Hall
48 Quincy Street, Cambridge, Mass. 02138

• • •

Collaborator Sought

"The book *Software Tools* (Brian W. Kernighan and P. J. Plauger, Addison-Wesley, 1976) contains many programs which would be extremely useful if adapted to the HP 3000. The programs are written in Rational Fortran (Ratfor) and are available on magnetic tape from Addison-Wesley; the tape also contains a Ratfor-to-Fortran preprocessor, written in Fortran.

"I'd be interested in hearing from anyone who has suggestions to offer or is interested in collaborating on this project."

Jamie E. Hanrahan
Data Processing Manager
The San Gabriel Valley Tribune
1210 N. Azusa Canyon Road
West Covina, CA 91723
(213) 962-8811, Ext. 408

• • •

Finite Differences and Finite Element Codes

Has anyone converted a 2-D Lagrangian finite difference code such as TOODY or STEALTH or a 2-D static and dynamic, linear and non-linear finite element code such as ADINA for an HP 3000? Our fields of interest are dynamic fracture and fluid dynamics calculations.

Any information welcome.

Contact: Anne Vallotton
Institut CERAC S.A.
Chemin des Larges Pieces
CH-1024 *Ecublens*
Switzerland

• • •

Wanted: List Processing Languages

We have an HP 3000 II Model 8 with BASIC, FORTRAN, COBOL, and APL. As we begin a formal Computer Science curriculum in the fall of 1978, we would like to have an even broader range of languages available. In particular, we would like to add a list processing language such as LISP or SNOBOL and a highly structured language such as PASCAL; have compilers for these languages been developed for the 3000 II?

Contact: R. S. Cunningham
Associate Professor of Mathematics
Birmingham-Southern College
Birmingham, Alabama 35204
Tel. (205) 328-5250

• • •

Printer Interface Controller

BUSINESS COMPUTER CONCEPTS, Inc. at RD 1, Box 131-B, Burgettstown, PA 15021, is now marketing the printer-interface controller. This is an automatic switch which allows two CPU's to share one printer. The device was described in the November/December, 1977, issue of the User's Group Journal, ("Get the Most Out of Your Printer" - p. 5). Business Computer Concepts is offering a 15-day free trial on the unit.

If interested, contact Richard E. Starck at (412) 729-3510.

• • •

Data Concentrator for Use With HP Computer Systems

MICOM's Micro800 Data Concentrator now includes an option for use with Hewlett-Packard computer systems.

The Data Concentrator uses statistical multiplexing techniques to allow up to eight asynchronous terminals to share a single telephone line, with channel capacity at least double that provided by a conventional time-division multiplexor. The Micro800 also provides automatic retransmission on error for all terminals using the system.

Several thousand characters of dynamically allocated buffer storage allow data to be backed up when poor line conditions necessitate retransmissions.

The 'HP' option for the Micro800 eliminates the risk of buffer overflow in these conditions by sending the DC1/DC2 control characters to control transmission from the attached computer port or terminals. DC1 causes transmission to be suspended; DC3 causes transmission to resume.

For more information contact Roger Evans, Vice President, Marketing, MICOM SYSTEMS, INC., 9551 Irondale Avenue, Chatsworth, CA 91311, 213/882-6890, TWX 910/494-4910.

Package: 2645 DATA ENTRY AND EXTRACT SYSTEM

Description:

1. Copy Files from 2645 Cartridge to HP 3000.
2. BUILD Detail Files based on record ID's. (EXTRACT). Record types may be mixed on cartridge. Several added functions such as zero fill, right and left justify, are performed on data records. HP 3000 EXTRACT files can be appended.

PRICE: $600.00 – one-time charge.

CONTACT: Mike Cimball
c/o Willamette Valley Co.
660 McKinley St.
Eugene, Oregon 94702
Tel. (503) 484-9621

• • •

Arrays Too Big?

Are your HP 3000 engineering application programs hampered by the 32K data stack size? Steven Richfield is marketing a program designed to circumvent this limitation. While use of this program does require slight modifications to the user's programs, they are indeed minor and easily effected. For further information:

Steve Richfield
10032 - 31st Avenue N.E.
Seattle, WA 98125
(206) 527-0114

• • •

All About Us

Report on CCRUG (Central Canada)

Ninety-six persons attended the Central Canada Regional Users Group at the HP Toronto Office, on February 14, 1978.

President Doug Wilson outlined new directions the group will follow in 1978.

The HP 263X family of Printer/Terminals was introduced and demonstrated.

New "Foreign" Software (MCS 3000) was described by Doug Wilson (Conestoga College).

IMAC, an interface dictionary package for IMAGE, was described by Joel Robinson (Wimpey).

Performance Measurement and Tuning – Two presentations were made on this topic by Don Gilchrist (McMaster University) and by Joel Robinson (Wimpey), followed by a panel discussion.

Peter McNaughton of The Royal Canadian Mounted Police, was the scheduled speaker; his presentation dealt with EDP Security.

For additional information, contact Doug Wilson, Conestoga College, 299 Doon Valley Drive, Kitchener, Ontario, Canada N2G 3W5. (Telephone) (519) 653-2511.

• • •

SCRUG Held Two-Day Meeting at Sea

The Southern California Regional Users Group (SCRUG) held a meeting March 1st and 2nd, on the Queen Mary, a former Cunard Line ship–now securely cradled on underwater concrete piers at wharfside, at Long Beach, California.

The meeting was very well attended by over 200 persons, according to Doug Mecham, SCRUG's Meeting Manager. The variety and quality of the presentations made the two-day experience very worthwhile to all present. Topics covered included the following:

TECHNICAL PAPERS

HP 3000 Operator Tasks and Procedures, by Kenneth R. Konold and Dennis S. Chernoff, Hughes Aircraft Company, El Segundo, California.

Extra Data Segments and Process Handling, by Jack Howard - HP, Lawndale, California.

Writing A User's Guide, by Douglas J. Mecham, Hughes Aircraft Co., Fullerton, California.

Operator Tasks and Procedures, by Debi Bragg, Hughes Aircraft Co., Fullerton, California

Editing and Publishing a Company Technical Newsletter, by Richard J. Nelson, Statek Corporation, Orange, Calif.

Operator Utilities, by Lloyd Gulick and Dan Coats, Hughes Aircraft Co., Fullerton, California.

Simple Application Modeling that Can Cut System Design in Half, by Emmet Rixford, The Toro Company, Riverside, California.

Software Considerations for Networks on the HP 3000, by Rick Ehrhart, Hughes Aircraft Co., El Segundo, California.

An Introduction to the Segmenter, by David J. Watson, Armament Systems, Inc., Anaheim, California.

Segmentation for Software Conversion, by Doug Mecham, Hughes Aircraft Co., Fullerton, California.

Systems Management, by Doug Mecham, Hughes Aircraft Co., Fullerton, California.

Accounting Procedures, by Randy Cummings, Hughes Aircraft Co., Fullerton, California.

User and Operator Training, by Randy Cummings, Hughes Aircraft Co., Fullerton, California.

Security, by Randy Cummings, Hughes Aircraft Co. Fullerton, California.

KSAM vs. IMAGE, by Steve Kaminski, San Bernardino Valley Municipal Water District, San Bernardino, CA.

Asynchronous Communication Protocols, by Charles J. Villa, Alter*Ability, San Francisco, California.

For additional information, contact Jack Savage, Comarco, Inc., 227 W. Hueneme Road, Oxnard, California 93030. Tel. (805) 488-6441.

• • •

GNUG Is Greater New York

The Greater New York Users Group met February 28, 1978, at Vydec Corporation, Florham Park, New Jersey; the meeting was hosted by Sharad Hedda, GNUG President. Topics of discussion included:

General Data Communications Concepts, by Jack O'Neill, General Data Communications

HP Paramus Update, by Marc Hoff, Hewlett-Packard

Terminal Support by MPE, by Jeff Wolfson, Hewlett-Packard

Additional group information is available from Bill Glanzman, Airco Industrial Gases, P. O. Box 1601, 2720 U.S. Highway 22, East, Union, New Jersey 07083. Tel.: (201) 464-8100.

• • •

BARUG Attendees Treated to Canadian Member's Presentation

The Bay Area Regional Users Group met at HP Neely Santa Clara Sales Office on April 5, 1978. The one-day session was well attended by more than 100 people. Technical sessions were presented by Robert M. Green, President of Robelle Consulting, Ltd., in Delta, British Columbia, Canada, and by HP Personnel. Technical sessions included the following topics:

Optimizing On-Line Programs, by Robert M. Green, Robelle Consulting, Ltd.

Binary Synchronous Protocol, by Don Van Pernis, HP, General Systems Division

Electrical Power Considerations for an HP 3000 Installation, by Frank Steiner, Hewlett-Packard

The RJE 2780/3780 Emulator, by Ed Turner, HP General Systems Division

MPE-IIB Update, by Ken Spalding, HP, General Systems Division

For more information, contact Bill Gates, Longs Drug Stores, Inc., 141 North Civic Drive, Walnut Creek, CA 94556. Tel.: (415) 937-1170.

• • •

European Users Group Meeting in June '78!

by William Bryden
Chairman, Executive Board
HP General Systems Users Group
c/o Inland Systems Engineering
424 Beverly Drive
Redlands, California 92373

HP 3000 users will meet at the London Graduate School of Business Studies June 28th through 30th. The meeting will be hosted by John Eaton, Director of Computing Services for the school.

HP will have ample representation, and it is expected that attendance will exceed 100. Please plan to be there; presentations are also desired.

For additional details, please contact John Eaton at London Graduate School of Business Studies, Sussex Place, Regents Park, London NW1 4SA, England.

● ● ●

HP General Systems Users Group Update:

7th INTERNATIONAL MEETING
OCTOBER 30TH THROUGH NOVEMBER 3RD, 1978
AT THE DENVER HILTON
DENVER, COLORADO

First, mark your calendar-*right now*-with "ATTEND USERS GROUP MEETING, DENVER" in the week of October 30th to November 3rd. Second, think back over the last few months . . . . . hasn't there been an application, program, technique, or piece of knowledge that is worth passing on to other users? You bet! So now is the time to fill out the enclosed "CALL FOR PRESENTATIONS & HOSTS" pamphlet, and send it off to the Program Chairman. If you have inquiries or suggestions relative to other areas, feel free to contact the appropriate Chair Person listed below.

Planning for the 7th International Meeting is underway. Now is the time to get your inputs into the Meeting Committee:

Joyce B. Pleasants
Int'l. Meeting Host
Aurora, Colorado
(303) 344-8060, X253

C. R. Van Ausdall
(Program)
Denver, Colorado
(303) 373-4320

Bob Rice
(Registration)
Commerce City, CO
(303) 289-5966

Chuck Green
(Exhibits)
Commerce City, CO
(303) 289-5966

Don Phelps (Activities)
Denver, Colorado
(303) 757-9011

Send your inputs to the appropriate person, addressed to:

HP General Systems Users Group
1978 International Meeting
P. O. Box 722
Aurora, Colorado 80040

MAJOR PROGRAM AREAS:

- *Applications*
- *Data Management*
- *System Utilization/Optimization*
- *Installation Management*
- *Languages*
- *Data Communications/Networking*
- *System Development*

Additional information on program areas is provided in the enclosed "CALL FOR PRESENTATIONS & HOSTS." The current program structure calls for each program area to be covered twice - Monday through Wednesday A.M., with a "repeat" from Wednesday P.M. to Friday. Program areas such as Languages and Applications will be treated in concurrent (horizontal) sessions, while the remaining program areas will be treated in serial (vertical) sessions. (See accompanying article on program structure by Tom Harbron). For example you will be able to follow a data management vertical strand and attend one of the language and one of the applications sessions in the Monday to Wednesday A.M. time-frame, then follow a communications/networking vertical strand and attend another of the language and applications sessions in the Wednesday P.M. to Friday time-frame. In either time frame, intermixing of vertical strands will be possible, as common time blocks will be used.

Wednesday will feature exhibits of complementary products for your HP 3000 system. Registration packages will be available for the full week, or for three days, i.e., Mon.-Tues.-Wed. or a Wed.-Thurs.-Fri. package. This program arrangement provides the flexibility for users who can spend a week at the meeting to spend their time productively. It also provides for users who cannot spend up to a full week at the meeting, to participate in a meaningful way, and permits sites that find it difficult to have two people away at the same time, to have two people participate.

We will keep you posted via special mailings and updates in the Journal. One final point - if you didn't mark your calendar, do it right now . . . the date again is October 30th through November 3rd, the place is Denver, Colorado. That way you won't schedule something that will prevent you from enhancing your professional knowledge, meeting old friends, making new ones, and generally having A ROCKY MOUNTAIN HIGH!

Please help plan the meeting. Fill out the card on the rear cover of this issue, and mail. *Your input is important!*

● ● ●

HOW DO YOU LIKE YOUR MEETING ORGANIZED?

by Tom Harbron
Director, Computing Center
Anderson College
Anderson, IN 46011

At a recent meeting of the Board of Directors, the question of how to best structure the Denver meeting was raised. In a meeting where there are concurrent sessions and multiple sessions on related topics, there are two basic forms of organization: horizontal and vertical. Each has advantages and disadvantages.

Horizontal organization means that all related sessions are grouped together in the same time-slot. For example, the first time slot might contain sessions relating to data management with individual sessions on IMAGE fundamentals, using QUERY, KSAM applications, and MPE file systems. The second time-slot might be oriented toward applications with sessions on manufacturing, student scheduling, structural engineering, and accounting. The advantage of horizontal structuring is that each person can sample from a wide range of topics. The disadvantage is that it is impossible to attend all sessions in a particular area of interest.

Vertical organization means that most sessions on related topics are in different time-slots. For example, all data management sessions might meet in room #1 with IMAGE fundamentals in the first time-slot, QUERY in the second, KSAM in the third, and MPE files in the fourth. Concurrently, applications sessions would be held in room #2 with manufacturing in the first time-slot, student scheduling in the second, etc. The advantage of vertical structuring is that a person can usually attend all sessions in a given area of interest. The disadvantage is that it is more difficult to attend sessions in many different fields.

Both methods work, and have their supporters. Last year's meeting in Seattle was horizontally structured; the National Computer Conferences are usually vertically structured. Currently,* the Denver meeting is being organized using both horizontal and vertical scheduling - applications and language sessions will be treated horizontally, with the remaining sessions being treated vertically.

If you have an opinion, use the enclosed reply card to let us know your preference.

---

*We would like your opinion.