SH20-1872-0

Installed User Program

VS APL for TSO: Yale University Terminal User's Guide

Program Number: 5796-ALB
Program Product: 5748-AP1

This manual describes the use of VS APL when it is under control of TSO using Installed User Program 5796-ALB (VS APL for TSO). It contains detailed information on the terminals that can be used and the procedures that must be followed in using VS APL under TSO. This book also describes the commands presented by TSO and VS APL relevant to the needs of the VS APL user, and the auxiliary processors and workspaces distributed with the product. It is assumed that the user is familiar with the APL language, but has limited experience with TSO.

This manual is based upon the "VS APL for CMS: Terminal User's Guide" and has been modified to include instructions for executing VS APL in the TSO environment.

The information contained in this manual is essential to the proper execution of VS APL in the TSO environment.

PROGRAMMING SERVICES PERIOD

During a specified number of months immediately following initial availability of each licensed program, designated as the PROGRAMMING SERVICES PERIOD, the customer may submit documentation to a designated IBM location when he encounters a problem which his diagnosis indicates is caused by a licensed program error. During this period only, IBM through the program sponsor(s), will, without additional charge, respond to an error in the current unaltered release of the licensed program by issuing known error correction information to the customer reporting the problem and/or issuing corrected or notice of availability of corrected code. However, IBM does not guarantee service results or represent or warrant that all errors will be corrected. Any onsite programming services or assistance will be provided at a charge.

WARRANTY

EACH LICENSED PROGRAM IS DISTRIBUTED ON AN 'AS IS' BASIS WITHOUT WARRANTY OF ANY KIND EITHER EXPRESS OR IMPLIED.

First Edition (August 1976)

This edition applies to Release 1.1 of VS APL, Program Product number 5748-AP1, and to Release 1.0 of VSAPL for TSO, IUP number 5796-ALB.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, 2200 Whitney Avenue, P.O. Box 5091, Hamden, Ct 06518, Attention: Dick Dunbar

All comments and suggestions become the property of IBM.

AV[112 107 198,65+0 4 5 14 9 19 5]

PREFACE

This book describes the use of VS APL when it is under control of TSO. It contains detailed information on the terminals that can be used with the product and the procedures that must be followed in communicating with TSO and VS APL. This book also describes the commands presented by TSO and VS APL relevant to the needs of the VS APL user, and the auxiliary processors and workspaces distributed with the product. It is assumed that you are already familiar with the APL language, but that you have no prior experience with TSO.

The information in this publication is organized into seven sections:

- "Introduction," which introduces TSO and illustrates its relationship to the VS APL Program Product.
- "Terminal Procedures," which lists and describes the terminals that you may use in your work, details what conventions you must follow in making entries and correcting entries at these terminals; and discusses how the form of output at your terminal may be controlled.
- "The Work Session," which describes how to start and end a work session with TSO and VS APL.
- "Workspaces and Libraries," which covers the structure and attributes of the VS APL workspace and details the libraries supported by TSO.
- "System Commands," which illustrates the form and use of VS APL system commands. System commands are used to monitor and control the contents of APL workspaces and libraries, as well as to communicate messages to other users.
- "Auxiliary Processing," which describes the use of the auxiliary processors distributed with VS APL.
- "Sample Terminal Session," which shows a hypothetical terminal session with VS APL under TSO. This session shows many of the features presented in the first six sections of the publication. The session is designed so that you can sit at your terminal and work along as each entry and response is described.

Six appendixes are also provided:

- "Appendix A," which lists and summarizes the predefined public library workspaces distributed with VS APL Program Product and VS APL FOR TSO Installed User Program.
- "Appendix B," which introduces the VS APL Conversion Program and describes the VS APL conversion report.
- "Appendix C," which describes some special language considerations for VS APL.
- "Appendix D," which summarizes the operating procedures for all the terminals supported by TSO and discusses a number of APL considerations for these terminals.
- "Appendix E," which discusses the migration considerations for VS APL FOR VSPC.

- "Appendix F," which lists and explains the error messages that you may receive in interacting with TSO and VS APL, and suggests possible corrective action that you might take.

Required Publications

This publication assumes that you are familiar with the APL language as described in the publication *APL Language,* GC26-3847.

Related Publications

In addition to *APL Language,* you might find the following publications helpful:

- *OS/VS2 Terminal User's Guide,* GC28-0645 This publication should be referenced for a description of the operating procedures for all the terminals supported by VS APL under TSO.
- *VS APL for CMS: Writing Auxiliary Processors,* SH20-9068. This publication describes how installation-dependent auxiliary processors may be created to operate with VS APL. This book is directed primarily to system programmers.
- *VS APL Installation Reference Material,* SH20-9065. This publication contains a full description of the APL Conversion Utility Program and should be consulted for a description of conversion procedures. This book is directed primarily to system programmers.
- *OS/VS2 TSO Command Language Reference,* GC28-0646 This publication contains a complete description of the TSO command language. This book should be referenced for detailed information about TSO operating environments and the commands that are acceptable to each environment.
- *OS/VS Message Library: VS2 System Messages,* GC38-1002 This publication describes the messages issued by the Operating System and TSO.

Syntax Notation

Throughout this book an attempt is made to define as clearly as possible the statements and commands that you may enter at your terminal. Where an entry may take several forms, all the forms are indicated. When an entry is shown, a distinction is made between information that you must enter exactly as shown and information that you may supply, as appropriate. Furthermore, a distinction is made between APL entries and TSO entries. If an item is representative and must be replaced with an appropriate value, the item will appear in lowercase italics. If an APL entry is to be made exactly as shown, the entry will appear in APL font (an uppercase italicized font). For instance, in the following VS APL system command:

)CLEAR size

)CLEAR is entered at the terminal as shown, while *size* is only representative and is replaced by a suitable size when the actual entry is made. If a TSO entry is to be made exactly as shown, the entry will appear in lowercase courier font. For instance, in the following TSO command, logon is entered exactly as shown, while *userid* must be replaced:

logon *userid*

This page intentionally left blank

CONTENTS

This page intentionally left blank

FIGURES

This page intentionally left blank

INTRODUCTION

This book describes how you can use VS APL when it is operating under control of the Time Sharing Option (TSO) of the IBM Operating System (OS/MVT or OS/VS2).

Together, the *system* presented by the combination of VS APL and TSO allows you to interact with a computer conversationally through the powerful APL language. This book assumes that you already know the APL language as described in the publication "APL Language."

In executing VS APL under control of TSO you are presented with the following features:

- A variety of different terminals that you can work from, including the IBM 2741 and 3767 terminals, the IBM 5100 operating in communications mode, the IBM 3270 Information Display System Terminals, and full ASCII terminals.
- A set of instructions called *VS APL system commands* that allow you to monitor and control your work as well as send messages to other users.
- A number of programs called *auxiliary processors,* that allow you to perform OS sequential dataset access and other operations that are normally beyond the scope of the APL language.
- A collection of pre-defined or *distributed workspaces* that are helpful in learning APL, in converting from other IBM APL systems, or in using auxiliary processors.

This page intentionally left blank

TERMINAL PROCEDURES

This section describes the terminals you can use with VS APL under TSO, some general conventions about making and correcting your terminal entries, and how output information is displayed at these devices.

Terminals

The terminals that you may use in working with VS APL under TSO are:

- IBM 2741 Communication Terminal
- IBM 3767 Communication Terminal
- IBM 3270 Interactive Display System Terminals (IBM 3275/3277)
- IBM 5100
- IBM 1050 Communication System Terminals (IBM 1052 Printer-Keyboard)
- Full ASCII terminals (Bit pairing or typewriter pairing keyboard arrangements)

All of these terminals have a typewriter-like keyboard for entering information and either a typewriter-like printer or display screen for recording your entries and displaying responses by the system. Most of them can be ordered with a number of different keyboards, and some can be used with different printing elements. As an APL user, you'll probably be working at an APL keyboard, and where printing elements are changeable, using a standard courier printing element to communicate with TSO and a standard APL printing element to communicate with VS APL. The elements required are listed in Figure 1.

Terminal	Keyboard	Standard Printing Element TSO	VS APL
2741	Correspondence	1167043	1167987
2741	PTTC/EBCD	1167963	1167988
2741	PTTC/BCD	1167938	1167988
1050	PTTC/EBCD	1167963	1167988
1050	PTTC/BCD	1167938	1167988

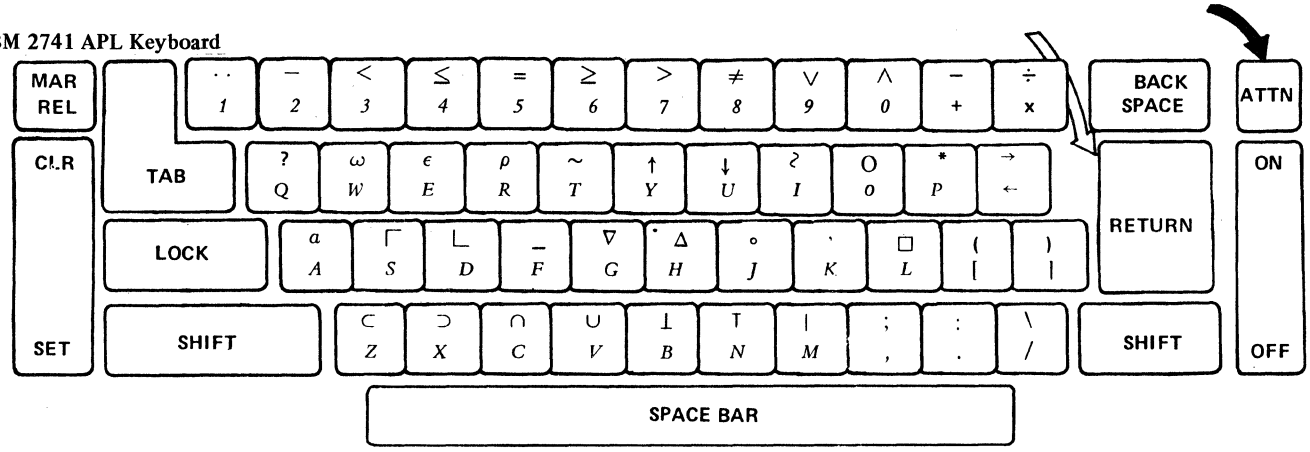
Figure 1. Standard Printing Elements for TSO Terminals

The IBM 3767 terminal uses a non-replaceable printing device whose set of printable characters is controlled by the switch marked EBCDIC (or Correspondence)/APL on the control panel. To communicate with TSO you should depress the EBCDIC part of the switch. To communicate with VS APL you should depress the APL part of the switch. Similar options are available on Full ASCII terminals.

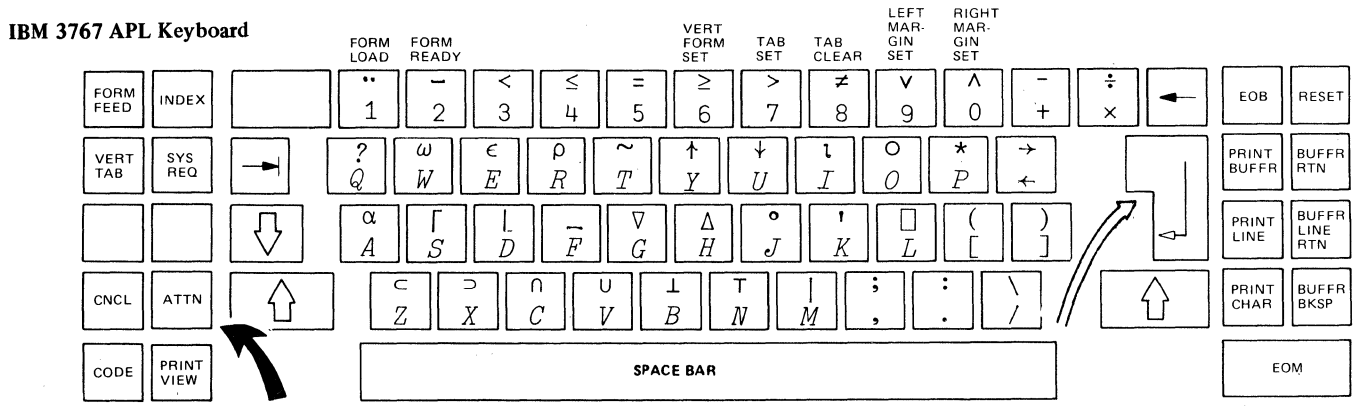
The operating procedures described in this book, unless otherwise indicated, assume that you are using an IBM 2741 terminal or an IBM 3767 terminal, with an APL keyboard. Existing IBM 2741 and IBM 3767 terminal keyboards can be modified by attaching *APL Characters* (order number GX20-1783) to the keys. The APL keyboards for these terminals and the IBM 3270 are illustrated in Figure 2.

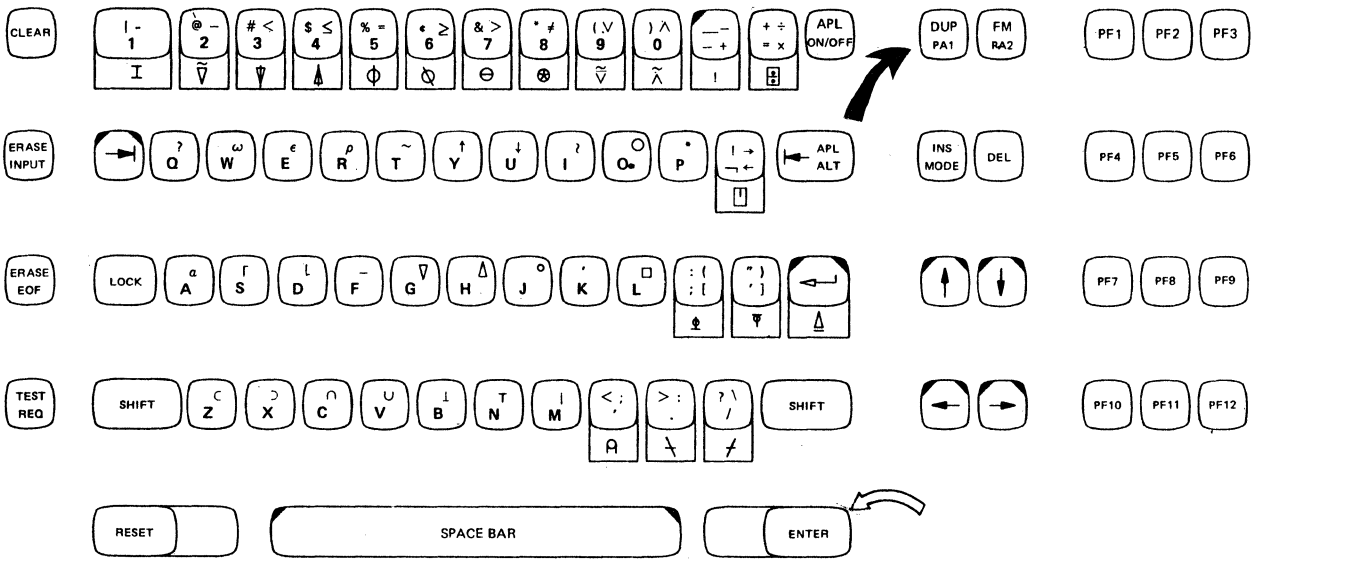
"Appendix D: TSO Terminals" summarizes the operating procedures for all the terminals that can be used with VS APL under TSO and describes restrictions and other items to be considered when using APL at these terminals.

IBM 2741 APL Keyboard



IBM 3767 APL Keyboard





Attention Signaling Key

End-of-line Signaling Key

LEGEND:

TYPAMATIC WHEN THE APL ON/OFF KEY IS OFF

TYPAMATIC WHEN THE APL ON/OFF KEY IS ON

KEY TOP

KEY FACE

APL KEYS:

APL ON/OFF -- Must be on to enter APL characters or the character to the right on double-character keys.

APL ALT -- Press and hold down to enter character on key face or to underscore APL capitals.

APL on and SHIFT key -- Enters APL characters shown on the upper right of key top.

Figure 2. The IBM 2741, 3767, and 3270 APL Keyboards

Character Sets

The characters that you have available to create your entries depend on (1) the terminal you're working from, and (2) the part of the system you're communicating with.

When you communicate with TSO, it assumes the non-APL character set associated with your terminal keyboard. These are the characters that can be printed using the standard TSO printing elements listed in Figure 1 or for the IBM 3767, the characters that can be printed when the EBCDIC (or Correspondence)/APL switch is set to EBCDIC (or Correspondence).

APL Language

When you establish contact with VS APL, the APL character set illustrated in Figure 3 is assumed. These are the characters that can be printed using the standard APL printing elements listed in Figure 1 or for the IBM 3767, the characters that can be printed when the EBCDIC (or Correspondence)/APL switch is set to APL.

Some of the characters shown in Figure 3, such as ⍞ are *compound characters* that are formed by entering a character, pressing the backspace key, and overstriking the first character.

On the 3270, these characters are formed by pressing the "APL ALTERNATE" key and a specially marked key for each of the compound characters. (see figure 2 on page 16).

Auxiliary Processing

In addition to the characters available to you for APL program development described above, VS APL supports additional character translations for communicating with your terminal (depending upon type), communicating directly with programs written in other languages called Auxiliary Processors (eg., APL100, the TSO Command Processor), or indirectly with other programs through OS datasets processed using Auxiliary Processor 111, the OS Sequential Dataset Processor.

The character conversion you select depends upon the final destination of the data, which might be:

- Your APL program (from an OS dataset)
- Your terminal (as output from an APL program)
- Another user (perhaps on a different terminal type)
- A printer for hardcopy display
- A sequential OS dataset for input to aother program written in APL or any other supported OS language.
- An auxiliary processor provided by the system, or written by you.

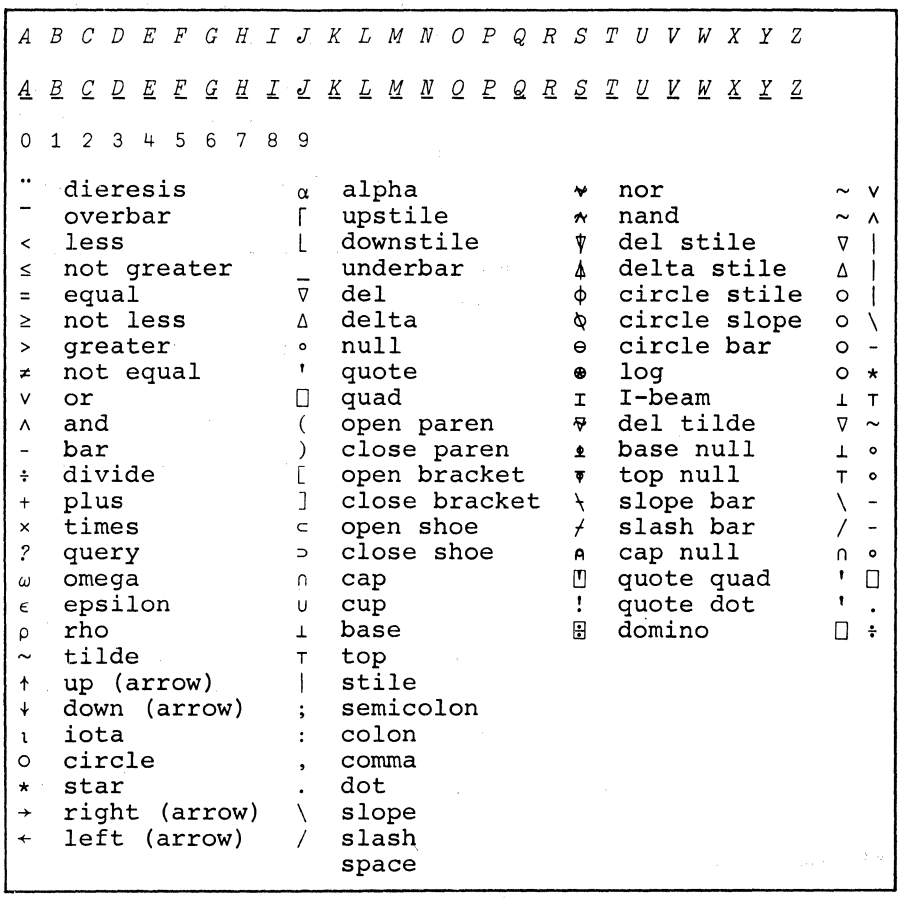


Figure 3. The APL Character Set

Conversion Options

The auxiliary processing conversion options are:

- 370—limited graphic character conversion from APL to EBCDIC. On output, the data part of a character vector or scalar is written one byte per element, characters that cannot be translated are sent as blanks. On input, the record is transmitted as a character vector, one element per byte, characters that cannot be translated are accepted as the APL character (○).
- 192—full graphic character translation from APL to EBCDIC. This includes direct translation of compound APL characters to unique EBCDIC codes. On output, the data part of a character vector or scalar is written one byte per element. On input, the record is transmitted as a character vector, one element per byte.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by TSO. Compound characters are expanded to their constituent parts. For example, A̲ is converted to A, backspace, __ on output and __, backspace, A is converted to A̲ on input. On output, the data part of a character vector or scalar is written. On input, the record is transmitted as a character vector.
- VAR—on output, the entire variable including its size, shape and type information is written with no conversion. On input, the entire record is transmitted.

- BIT—on output, the data is transmitted as a vector or scalar one bit per element; each element of data must have a value of 1 or 0. On input, the record is transmitted as logical vector, one element per bit.
- BYTE—on output, the data is written with no conversion, one byte per element. On input, the record is transmitted as a character vector or scalar, one element per byte. The data conversion functions in the public library workspace *APFNS* can be used to convert this information. (see figure 4).

Function Name	Description
ECI	Converts EBCDIC characters to APL characters.
ECO	Converts APL characters to EBCDIC characters.
FI	Converts System/370 floating-point numbers to APL format.
FO	Converts APL numbers to floating-point format.
II	Converts System/370 binary values to APL format.
IO	Converts APL numbers to System/370 binary format.
LI	Converts System/370 logical data to APL format.
LO	Converts APL numbers with values of 0 or 1 to System/370 logical format.
PDI	Converts System/370 packed-decimal numbers to APL format.
PDO	Converts APL numbers to System/370 packed-decimal format.

Figure 4. Data Conversion Functions in APFNS

Atomic Vector

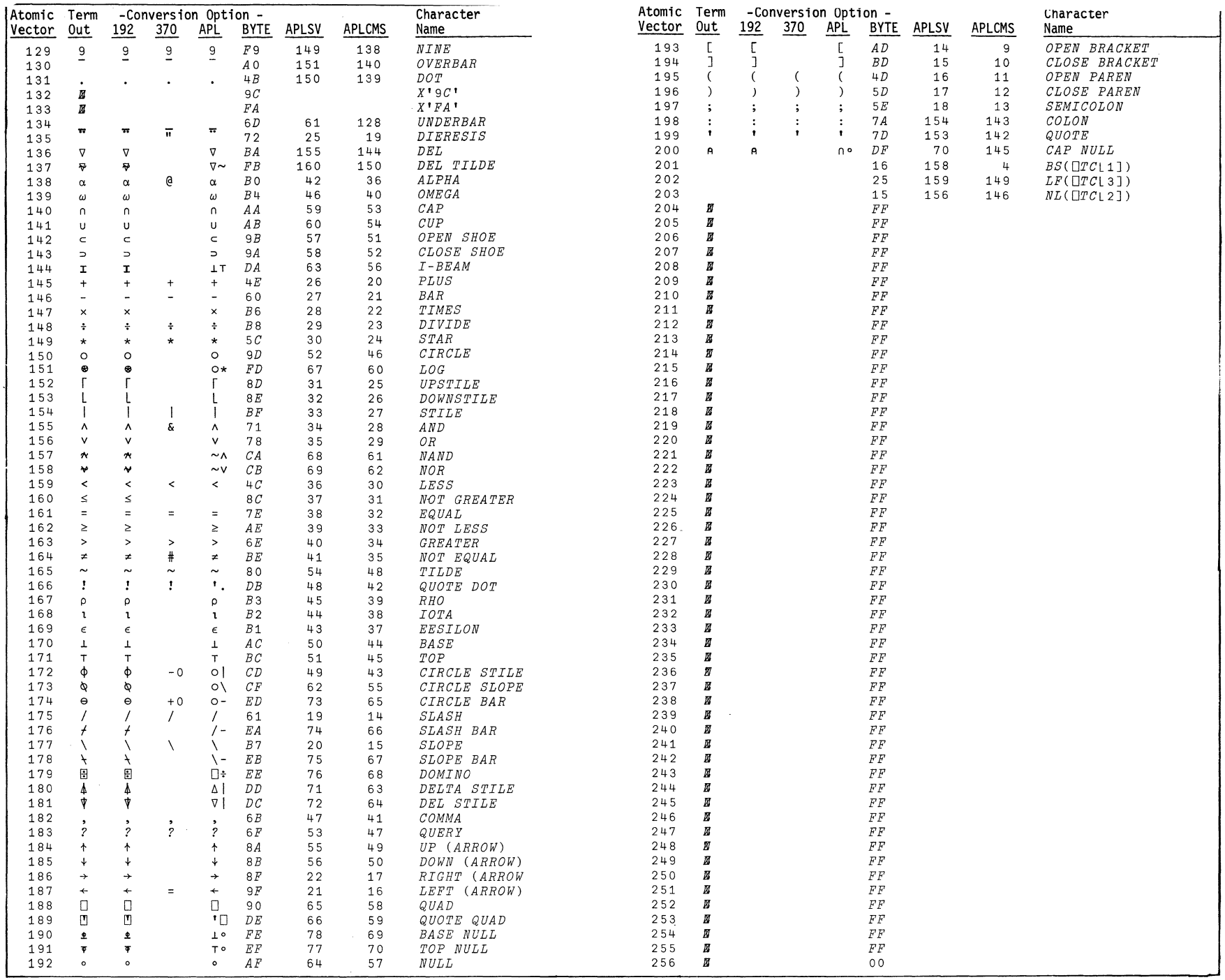
⎕*AV* is the ATOMIC VECTOR. Contained in ⎕*AV* are all of the characters acceptable to the VS APL interpreter.

Important: it should be noted that the use of ⎕*AV* is completely implementation dependent! Because ⎕*AV* is based upon the actual machine encoding of the character strings used to form the characters, it will vary from system to system (though it is constant within VS APL systems). Therefore, calling ⎕*AV* from within functions should be avoided if you wish to remain implementation independent, or plan to transfer your workspace to any other APL system.

⎕*AV* is 256 elements in length, and contains the complete character set acceptable to VS APL. Although all the positions are assigned, many of the characters may not be directly "visible" to you at your terminal because of the translation performed by TCAM (the terminal control program used by TSO). Instead, these characters will be transmitted to your terminal as the character string 'Z', backspace, 'N' (or the blot character " on a 3270).

Atomic Vector	Term Out	-Conversion Option- 192	370	APL	BYTE	APLSV	APLCMS	Character Name
1	▯				FF			
2	▯				FF			
3	▯				FF			
4	▯				FF			
5	▯				FF			
6	▯				FF			
7	▯				FF			
8	▯				FF			
9	▯				FF			EBCDIC FF
10	▯	&			50	206		AMPERSAND
11	▯	%			6C	211		PERCENT
12	▯	"			7F	215		DOUBLE QUOTE
13	▯	±			9E	235		PLUS-OR-MINUS
14	▯				B9			EBCDIC B9
15	▯				70			EBCDIC 70
16	▯				73			EBCDIC 73
17	▯				74			EBCDIC 10
18	▯				75			EBCDIC 75
19	▯				76			EBCDIC 76
20	▯				77			EBCDIC 77
21	▯				B5			EBCDIC B5
22	▯				E1			EBCDIC E1
23	▯	¢			4A	207		CENT
24	▯	}			5A			EBCDIC RIGHT BRACE
25	▯	{			C0	236		LEFT BRACE
26	▯	}			D0	237		RIGHT BRACE
27	▯	┐			EC			EBCDIC EC
28	▯	└			CE			EBCDIC CE
29	▯	┘			CC			EBCDIC CC
30	▯	¦			6A			EBCDIC 6A
31	▯	—			4F	213		EBCDIC VERTICAL
32	▯	¬			5F	214		PL/1 NOT
33	▯	\			E0			EBCDIC BACK-SLASH
34	▯	~			A1			EBCDIC TILDE
35	▯	`			79			ACCENT GRAVE
36	▯	@			7C	212		AT SIGN
37	▯	#			7B	210		POUND SIGN
38	▯	$			5B	209		DOLLAR SIGN
39	▯	a			81	180		A-LOWER CASE
40	▯	b			82	181		B-LOWER CASE
41	▯	c			83	182		C-LOWER CASE
42	▯	d			84	183		D-LOWER CASE
43	▯	e			85	184		E-LOWER CASE
44	▯	f			86	185		F-LOWER CASE
45	▯	g			87	186		G-LOWER CASE
46	▯	h			88	187		H-LOWER CASE
47	▯	i			89	188		I-LOWER CASE
48	▯	j			91	189		J-LOWER CASE
49	▯	k			92	190		K-LOWER CASE
50	▯	l			93	191		L-LOWER CASE
51	▯	m			94	192		M-LOWER CASE
52	▯	n			95	193		N-LOWER CASE
53	▯	o			96	194		O-LOWER CASE
54	▯	p			97	195		P-LOWER CASE
55	▯	q			98	196		Q-LOWER CASE
56	▯	r			99	197		R-LOWER CASE
57	▯	s			A2	198		S-LOWER CASE
58	▯	t			A3	199		T-LOWER CASE
59	▯	u			A4	200		U-LOWER CASE
60	▯	v			A5	201		V-LOWER CASE
61	▯	w			A6	202		W-LOWER CASE
62	▯	x			A7	203		X-LOWER CASE
63	▯	y			A8	204		Y-LOWER CASE
64	▯	z			A9	205		Z-LOWER CASE

Atomic Vector	Term Out	-Conversion Option- 192	370	APL	BYTE	APLSV	APLCMS	Character Name
65					40	152	141	SPACE
66	A	A	A	A	C1	86	74	A
67	B	B	B	B	C2	87	75	B
68	C	C	C	C	C3	88	76	C
69	D	D	D	D	C4	89	77	D
70	E	E	E	E	C5	90	78	E
71	F	F	F	F	C6	91	79	F
72	G	G	G	G	C7	92	80	G
73	H	H	H	H	C8	93	81	H
74	I	I	I	I	C9	94	82	I
75	J	J	J	J	D1	95	83	J
76	K	K	K	K	D2	96	84	K
77	L	L	L	L	D3	97	85	L
78	M	M	M	M	D4	98	86	M
79	N	N	N	N	D5	99	87	N
80	O	O	O	O	D6	100	88	O
81	P	P	P	P	D7	101	89	P
82	Q	Q	Q	Q	D8	102	90	Q
83	R	R	R	R	D9	103	91	R
84	S	S	S	S	E2	104	92	S
85	T	T	T	T	E3	105	93	T
86	U	U	U	U	E4	106	94	U
87	V	V	V	V	E5	107	95	V
88	W	W	W	W	E6	108	96	W
89	X	X	X	X	E7	109	97	X
90	Y	Y	Y	Y	E8	110	98	Y
91	Z	Z	Z	Z	E9	111	99	Z
92	Δ	Δ	$	Δ	BB	112	100	DELTA
93	A	A	a	A_	41	113	101	A-UNDERSCORE
94	B	B	b	B_	42	114	102	B-UNDERSCORE
95	C	C	c	C_	43	115	103	C-UNDERSCORE
96	D	D	d	D_	44	116	104	D-UNDERSCORE
97	E	E	e	E_	45	117	105	E-UNDERSCORE
98	F	F	f	F_	46	118	106	F-UNDERSCORE
99	G	G	g	G_	47	119	107	G-UNDERSCORE
100	H	H	h	H_	48	120	108	H-UNDERSCORE
101	I	I	i	I_	49	121	109	I-UNDERSCORE
102	J	J	j	J_	51	122	110	J-UNDERSCORE
103	K	K	k	K_	52	123	111	K-UNDERSCORE
104	L	L	l	L_	53	124	112	L-UNDERSCORE
105	M	M	m	M_	54	125	113	M-UNDERSCORE
106	N	N	n	N_	55	126	114	N-UNDERSCORE
107	O	O	o	O_	56	127	115	O-UNDERSCORE
108	P	P	p	P_	57	128	116	P-UNDERSCORE
109	Q	Q	q	Q_	58	129	117	Q-UNDERSCORE
110	R	R	r	R_	59	130	118	R-UNDERSCORE
111	S	S	s	S_	62	131	119	S-UNDERSCORE
112	T	T	t	T_	63	132	120	T-UNDERSCORE
113	U	U	u	U_	64	133	121	U-UNDERSCORE
114	V	V	v	V_	65	134	122	V-UNDERSCORE
115	W	W	w	W_	66	135	123	W-UNDERSCORE
116	X	X	x	X_	67	136	124	X-UNDERSCORE
117	Y	Y	y	Y_	68	137	125	Y-UNDERSCORE
118	Z	Z	z	Z_	69	138	126	Z-UNDERSCORE
119	Δ	Δ		Δ_	FC	139	127	Δ-UNDERSCORE
120	0	0	0		F0	140	129	ZERO
121	1	1	1	1	F1	141	130	ONE
122	2	2	2	2	F2	142	131	TWO
123	3	3	3	3	F3	143	132	THREE
124	4	4	4	4	F4	144	133	FOUR
125	5	5	5	5	F5	145	134	FIVE
126	6	6	6	6	F6	146	135	SIX
127	7	7	7	7	F7	147	136	SEVEN
128	8	8	8	8	F8	148	137	EIGHT



Atomic Vector	Term Out	-Conversion Option- 192	370	APL	BYTE	APLSV	APLCMS	Character Name
129	9	9	9	9	F9	149	138	NINE
130	¯	¯	¯	¯	A0	151	140	OVERBAR
131	.	.	.	.	4B	150	139	DOT
132	▯				9C			X'9C'
133	▯				FA			X'FA'
134	¨	¨	¯	¨	6D	61	128	UNDERBAR
135					72	25	19	DIERESIS
136	∇	∇		∇	BA	155	144	DEL
137	⍫	⍫		∇~	FB	160	150	DEL TILDE
138	α	α	@	α	B0	42	36	ALPHA
139	ω	ω		ω	B4	46	40	OMEGA
140	∩	∩		∩	AA	59	53	CAP
141	∪	∪		∪	AB	60	54	CUP
142	⊂	⊂		⊂	9B	57	51	OPEN SHOE
143	⊃	⊃		⊃	9A	58	52	CLOSE SHOE
144	⌶	⌶		⊥⊤	DA	63	56	I-BEAM
145	+	+	+	+	4E	26	20	PLUS
146	-	-	-	-	60	27	21	BAR
147	×	×		×	B6	28	22	TIMES
148	÷	÷	÷	÷	B8	29	23	DIVIDE
149	*	*	*	*	5C	30	24	STAR
150	○	○		○	9D	52	46	CIRCLE
151	⍟	⍟		○*	FD	67	60	LOG
152	⌈	⌈		⌈	8D	31	25	UPSTILE
153	⌊	⌊		⌊	8E	32	26	DOWNSTILE
154	|	|	|	|	BF	33	27	STILE
155	∧	∧	&	∧	71	34	28	AND
156	∨	∨		∨	78	35	29	OR
157	⍲	⍲		~∧	CA	68	61	NAND
158	⍱	⍱		~∨	CB	69	62	NOR
159	<	<	<	<	4C	36	30	LESS
160	≤	≤			8C	37	31	NOT GREATER
161	=	=	=	=	7E	38	32	EQUAL
162	≥	≥		≥	AE	39	33	NOT LESS
163	>	>	>	>	6E	40	34	GREATER
164	≠	≠	#	≠	BE	41	35	NOT EQUAL
165	~	~	~	~	80	54	48	TILDE
166	⍡	⍡	⍡	'.	DB	48	42	QUOTE DOT
167	⍴	⍴		⍴	B3	45	39	RHO
168	⍳	⍳		⍳	B2	44	38	IOTA
169	∈	∈		∈	B1	43	37	EESILON
170	⊥	⊥		⊥	AC	50	44	BASE
171	⊤	⊤		⊤	BC	51	45	TOP
172	⌽	⌽	-0	○|	CD	49	43	CIRCLE STILE
173	⍉	⍉		○\	CF	62	55	CIRCLE SLOPE
174	⊖	⊖	+0	○-	ED	73	65	CIRCLE BAR
175	/	/	/	/	61	19	14	SLASH
176	⌿	⌿		/-	EA	74	66	SLASH BAR
177	\	\	\	\	B7	20	15	SLOPE
178	⍀	⍀		\-	EB	75	67	SLOPE BAR
179	⌹	⌹		□÷	EE	76	68	DOMINO
180	⍙	⍙		Δ|	DD	71	63	DELTA STILE
181	⍒	⍒		∇|	DC	72	64	DEL STILE
182	,	,	,	,	6B	47	41	COMMA
183	?	?	?	?	6F	53	47	QUERY
184	↑	↑		↑	8A	55	49	UP (ARROW)
185	↓	↓		↓	8B	56	50	DOWN (ARROW)
186	→	→		→	8F	22	17	RIGHT (ARROW
187	←	←	=	←	9F	21	16	LEFT (ARROW)
188	□	□		□	90	65	58	QUAD
189	⍞	⍞		'□	DE	66	59	QUOTE QUAD
190	⍬	⍬		⊥○	FE	78	69	BASE NULL
191	⍭	⍭		⊤○	EF	77	70	TOP NULL
192	○	○		○	AF	64	57	NULL

Atomic Vector	Term Out	-Conversion Option- 192	370	APL	BYTE	APLSV	APLCMS	Character Name
193	[	[		[	AD	14	9	OPEN BRACKET
194	]	]		]	BD	15	10	CLOSE BRACKET
195	(	(	(	(	4D	16	11	OPEN PAREN
196	)	)	)	)	5D	17	12	CLOSE PAREN
197	;	;	;	;	5E	18	13	SEMICOLON
198	:	:	:	:	7A	154	143	COLON
199	'	'	'	'	7D	153	142	QUOTE
200	⍾	⍾		∩○	DF	70	145	CAP NULL
201					16	158	4	BS(□TC⌊1])
202					25	159	149	LF(□TC⌊3])
203					15	156	146	NL(□TC⌊2])
204	▯				FF			
205	▯				FF			
206	▯				FF			
207	▯				FF			
208	▯				FF			
209	▯				FF			
210	▯				FF			
211	▯				FF			
212	▯				FF			
213	▯				FF			
214	▯				FF			
215	▯				FF			
216	▯				FF			
217	▯				FF			
218	▯				FF			
219	▯				FF			
220	▯				FF			
221	▯				FF			
222	▯				FF			
223	▯				FF			
224	▯				FF			
225	▯				FF			
226	▯				FF			
227	▯				FF			
228	▯				FF			
229	▯				FF			
230	▯				FF			
231	▯				FF			
232	▯				FF			
233	▯				FF			
234	▯				FF			
235	▯				FF			
236	▯				FF			
237	▯				FF			
238	▯				FF			
239	▯				FF			
240	▯				FF			
241	▯				FF			
242	▯				FF			
243	▯				FF			
244	▯				FF			
245	▯				FF			
246	▯				FF			
247	▯				FF			
248	▯				FF			
249	▯				FF			
250	▯				FF			
251	▯				FF			
252	▯				FF			
253	▯				FF			
254	▯				FF			
255	▯				FF			
256	▯				00			

Figure 5. The Atomic Vector

Figure 5. The Atomic Vector

Terminal Entry

Your dialogue with the system consists of entries made from the terminal and responses generated by the system. In general, an entry consists of a TSO command, a VS APL system command, an APL statement, or data.

You can never type more than one VS APL system command, APL statement, TSO command, or data entry per line.

You may not change more than one line on the 3270 display screen before pressing ENTER.

All lines are completed by a carrier return. On the IBM 2741, the carrier return is the RETURN key, while on the IBM 3767, the carrier return is the key marked ←┘. On the IBM 3270, lines are accepted when you press the ENTER key.

Once you complete a line, the terminal does not accept further information until the system has analyzed and executed the line. If it detects an error in your entry, it prints a message indicating the trouble. Otherwise, the system performs the operation and displays a response when called for. After acting on your requests, the system usually prompts you for your next line of input. It may indent six character positions or print input request characters such as □:. The terminal then permits you to begin your next entry at that point. This method of entry and response continues until you sign-off.

Figure 6 illustrates a typical dialogue between you and VS APL. The first line represents your entry while the second line represents the VS APL's response. The arrow indicates the starting point of the next entry.

```
      )CLEAR
CLEAR WS
      2×17 19
34 38
      ↑
```

Figure 6. A Typical Dialogue with VS APL

If a compound character is entered that is unrecognizable to VS APL, for instance □, the carrier spaces over to the unrecognizable character, prints the inverted caret character (∨ under the character in error, and Line Feeds to the next line; the terminal then awaits further input. All characters to the left of the unrecognized character are accepted; all characters to the right of and including the unrecognized character are ignored. Any further characters that you enter are then substituted for the remainder of the entry.

Correcting Entries

There will be times when you'll have made a mistake before entering a carrier return and will want to make a correction.

If the error occurs in a TSO command, you can request the system to delete the character you just typed or to delete all the preceding characters in the line. You can define character-deletion and line-deletion characters, or you can use the default characters in the system. For example, if the control characters are the quotation mark (") for deleting the preceding character, and the precent sign (%) for deleting the current line, and you type the following message:

```
firstent%Sect"onft""dENR"try
```

The system receives it as:

```
SECOND ENTRY
```

Note that you can use the character-deletion character repetitively (to delete more than one of the preceding characters in the line).

Normally, you will use the default characters in the system, (usually the BACKSPACE and the ATTN key). However, you can use the PROFILE command to establish your own character-deletion and line-deletion characters.

If the error occurs in an entry to VS APL, you can erase all or part of the entry. To do so, backspace to the point of error and create an illegal character by overstriking with any suitable character. The system responds by spacing to the point of error, printing the inverted caret character (∨) under the character in error, advancing to the next line and awaiting further entry. This has the effect of erasing everything on the line from the point of the inverted caret to the end of the line, so that all characters from the point of error to the end must be retyped.

Suppose, for example, you typed the following statement at the IBM 2741 terminal:

```
ADD2←76  24+10.5  1:.1
```

and you realized that you wanted to enter the number 7 instead of the character :. To make this correction, you simply backspace to the position of the character : and type a " / " over the : and press the RETURN key. The system spaces to the point of error, prints the inverted caret and advances to the next line. Now you enter 7.1.

In practice, the exchange would look like this:

```
ADD2←76  24+10.5  1/.1
                  ∨
                  7.1
```

Tabs on Input

In communicating with VS APL, you may find it convenient to use the terminal's tab facility to save yourself the trouble of entering blanks. Tab controls are set locally at the terminal by depressing the SET key on the IBM 2741 or the combination of tab set (numeric 7) and CODE keys on the IBM 3767, or COMMAND 5 on the IBM 5100.

Once tab settings have been made, each time you press the tab key, the carrier will skip to the next tab setting. On the IBM 2741, the tab key is marked TAB, on the IBM 3767, the tab key is marked →|.

When you set tab controls at the terminal, you must also communicate these settings to VS APL. You do this by assigning the settings to the APL system variable $\square HT$. The tab settings you indicate through $\square HT$ are in effect until changed or until you sign off VS APL. When you next establish contact with VS APL, no tab settings are assumed.

For instance, if you physically set tab controls at positions 5, 10, 15, 20, 25, 30, 35 and 40, you must correspondingly set $\square HT$ as follows:

```
      □HT←5 10 15 20 25 30 35 40
```

Care should be taken that the value of $\square HT$ actually reflects the physical tab settings, as misleading or unreadable output may otherwise be produced.

If no tab settings have been communicated to VS APL or if the carrier is beyond the largest tab setting, then pressing the tab key during keyboard entry will result in an entry error. This kind of entry cannot be corrected using the procedures described in "Correcting Entries." Instead, the line must be re-entered correctly.

Terminal Output and Display

One convenience of VS APL under TSO, is that you can always follow the dialogue between it and yourself. The dialogue that is printed at the terminal consists of your printed entries and VS APL's responses. For instance, a typical dialogue between you and VS APL might include the following:

```
      ⍝ SEQUENCING
      NUMS←7 23 11 5 28
      NUMS[⍋NUBS]
VALUE ERROR
      NUMS[⍋NUBS]
             ^
      NUMS[⍋NUMS]
5 7 11 23 28
```

Messages and other printed responses sent to you by VS APL begin at the left-hand margin. This provides a handy way of distinguishing between your entries and VS APL's responses.

There may be occasions when VS APL attempts to print a character that is not representable at your terminal with the currently selected character set. When this happens, you'll see an error character. On the IBM 2741 terminal or IBM 3767, a blot (the character Z overstruck with the character N) will be printed instead of the unprintable character.

The examples illustrated in this book always indicate your entries and the system's responses as they would be printed at the terminal.

Tabs on Output

Terminal tab settings for VS APL output are established in the same way as they are for input. That is:

- Physical tab positions are set with the tab set (or SET) key.
- The physical tab positions must be communicated to VS APL through the system variable (`⎕HT`).

If tab settings have been communicated to VS APL, then it will use these tabs to reduce the time it takes to display information. The appearance of the printed output is not affected by the tab settings, only the time it takes VS APL to print it. For instance, the character string:

```
A  B  C
```

is displayed the same whether the tabs are set for the positions of A, B, and C or not set but the printing time can be reduced if tabs are set for the positions of A, B, and C.

The value of `⎕HT` and the physical tab settings must always correspond. Otherwise, misleading output may be produced.

Terminal Print Controls

Two system variables in VS APL have particular pertinence in controlling the format of output, these are `⎕PW` (printing width) and `⎕TC` (terminal control characters).

Printing Width

When you begin your work with VS APL, one item of information will have already been set by the system. This is the maximum line of output that can be printed, your printing width. The default value assumed by the system is 120 characters. You do however have the option of temporarily overriding this value by assigning a value to the system variable ⎕*PW*. The value you assign can be any number between 30 and 255 (although a value greater than the physical width of the terminal is not advisable).

For example, the following assignment would limit printing to 40 characters per line:

```
      ⎕PW←40
```

so that an expression producing 40 output characters would print on one line:

```
      40⍴'A'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

However, in this example, a printing width of 30 would force printing on two lines, the first line containing the first 30 characters while the remaining 10 characters would be printed on the second line:

```
      ⎕PW←30
      40⍴'A'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAA
```

The value you set for the printing width remains in effect throughout your TSO session, whether or not you end your communication with VS APL. When you next sign on, the printing width reassumes its default value.

In general, ⎕*PW* should be set to 255 when an IBM 5100 used to communicate with VSAPL under TSO to make EDIT/RETRANSMISSION of lines easier.

Terminal Control Characters

The printing of output may also be controlled by the terminal control characters found in the system variable ⎕*TC*. The characters found in ⎕*TC* are set by the system and cannot be overridden by you. There are three terminal controls in ⎕*TC* as follows:

- ⎕*TC*[1]: backspace character
- ⎕*TC*[2]: new line character
- ⎕*TC*[3]: line feed character

When one of these elements is encountered by the system in the printing of output, the indicated terminal action is taken, so that the entry:

```
      'A',□TC[2],'BC',□TC[3],□TC[1],'DEF'
```

is a request to:

Symbol	Meaning
`'A'`	Print *A*
`□TC[2]`	Advance to the next line and reset to position 1
`'BC'`	Print *BC*
`□TC[3]`	Advance to the next line but stay in same relative position
`□TC[1]`	Backspace one position
`'DEF'`	Print *DEF*

In practice, the exchange would look like this:

```
      'A',□TC[2],'BC',□TC[3],□TC[1],'DEF'
A
BC
  DEF
```

Interrupting Output

You can interrupt the action that VS APL takes on your entries, and the possible printed responses it returns, by entering an interrupt signal at your terminal. On both the IBM 2741 and 3767 terminals, the interruption of output is controlled by the ATTN key. Pressing the ATTN key once, a *weak interrupt,* has the effect of terminating output immediately and interrupting execution at the end of the current line. Pressing the ATTN key twice, a *strong interrupt,* terminates output and the execution of the current statement at the point the signal is generated.

The user should be aware, however, that TCAM is processing output messages asynchronous to APL execution. The last printed line probably will not reflect the current execution point of the VSAPL function.

This ability to interrupt VS APL prevents you from being "locked out" while a long-running function is executing and particularly when a great deal of terminal printing is preventing you from efficiently doing your work.

TSO does not support interruptions from a 3270 terminal. During a terminal session, there are three possible states for a 3270:

- The program is executing and the keyboard is locked. The user cannot type any key to interrupt this state. This is characteristic of the evaluation of a complex APL statement, the execution of an APL function, reference to a shared variable which requires substantial auxiliary processor service or operator intervention, or the delay (*□DL*) system variable.
- The program is waiting for input. This is characteristic of normal APL expression input, function editing, □ or ⍞ input. The keyboard is unlocked and anything may be typed in.
- The program is not executing and it is not waiting for input. Instead, TSO has stopped APL execution, and is waiting for a user decision to continue execution. The user signals a desire to continue execution by pressing the ENTER key; the user signals a desire to interrupt execution by pressing the PA1 key. Any other input is ignored. TSO does this automatically when output has filled the screen. The user may also request that this type of interruption decision be provided after the keyboard has been locked for a user specified period of time. The user chooses the amount of time in the SECONDS parameter of the TSO TERMINAL command:

```
READY
TERMINAL SECONDS(90)
```

After 90 seconds in keyboard-locked state, the keyboard is unlocked by TSO, and the APL execution is suspended. Pressing the ENTER key causes a return to keyboard-locked state; pressing the PA1 key interrupts the execution of the APL function.

Transmission Failures

Occasionally, the entry you make at your terminal or the output that the system returns may be garbled by certain failures in the transmission of data. If the failure occurs in your entry, the message *READ ERROR* will be printed and you'll have to repeat the previous entry. If the failure occurs while the system is transmitting output, there may be one or more unexpected characters in the output line. The presence of these unexpected characters should be obvious in most contexts. Additionally, on the IBM 3767 the SYSTEM CHECK panel light should go on. However, on the IBM 2741, there is no absolutely certain way of detecting a transmission failure in output.

THE WORK SESSION

No matter what kind of work you're going to do at the terminal, there are some initial steps that you'll have to take to get things started. Similarly, there will be a number of things that you'll have to do to get things finished. In programming terms, these initial and final steps are said to start and end the *work session.* In this section you'll learn what these procedures are and how they may be affected by the equipment you're using.

Starting the Work Session

There are four steps involved in beginning a work session with VS APL. The first step is to establish a communications link between your terminal and the computer. Once a link has been established, the next step is to make contact with TSO. The last step is to establish contact with VSAPL.

The procedures described in this section are by no means all-inclusive. They only represent the most general ways of beginning a work session with VS APL. Certain requirements unique to your operations may alter the steps outlined in this section. To determine the additional actions you may have to take, see your system administrator.

Step 1: Connection

The way you make a connection between your terminal and the computer depends on the type of terminal equipment you're using. If your connection to the computer is through telephone lines, you will need to dial a telephone number. Some terminals are directly wired to the computer, in which case you'll have to set some switches, but you won't have to dial up.

The connection procedure for the IBM 2741 Communication Terminal and the IBM 3767 Communication Terminal will be covered here. For connection procedures related to other TSO terminals, see the *TSO Terminal User's Guide.*

Connecting the IBM 2741

An IBM 2741 can be permanently connected to a computer system through non-switched (leased) line, or temporarily connected through a switched (dial) line.

If you're working through a switched line, you have the additional option of using a telephone data set or a regular telephone with an acoustic coupler.

Figure 7 shows the procedure for connecting an IBM 2741. Steps 1 and 2 are the only steps that need to be carried out if you're using a non-switched line. Otherwise, steps 1 through 5 must be completed. Steps 3 and 5 additionally depend on whether you're using a telephone data set or an acoustic coupler to maintain the telephone connection.

Switched and Non-Switched Lines

1. Find the COM/LCL switch on the left side of your terminal and switch it to COM.
2. For switched lines (excluding acoustic couplers) and non-switched lines, turn on the ON/OFF switch located at the right of the keyboard. (If the switch is on, turn it off, then on again.) For non-switched lines only, the keyboard unlocks; at this point the terminal is connected to the computer and you are ready to establish contact with TSO.

Switched Lines Only

3. **Telephone Data Set:** Press the TALK button.

   Acoustic Coupler: Make sure that the acoustic coupler is connected to the power supply, turned off, and connected to the terminal.
4. Remove the handset from the cradle and dial TSO's telephone number. This number should be supplied to you by your system administrator.
5. **Telephone Data Set:** Wait for a high-pitched tone. When you hear it, push the DATA button and place the handset in the cradle. The DATA light will go on. Your keyboard will unlock; this means that the terminal is connected with the computer and you're ready to establish contact with TSO. If the data light goes off at any time during the terminal session, start again from step 3.

   Acoustic Coupler: Wait for a high-pitched tone. When you hear it, place the handset face down in the coupler box. Turn on the acoustic coupler within 20 seconds. Turn on the ON/OFF switch located at the right of the keyboard. The keyboard will unlock, meaning the computer is ready to receive input data. If the keyboard does not unlock, start again from step 3.

Figure 7. Connecting the IBM 2741

Connecting the IBM 3767

Before a connection can be made between the IBM 3767 terminal and the computer system, certain switches on the terminal control panel must be set. Figure 8 lists the recommended switch settings for interaction with the system. On certain models of the IBM 3767, a security lock is provided. For these models, the key should be turned to ON before the POWER switch is turned on.

Switch	Setting
COMM/LOCAL	COMM
AUTO/OFF	AUTO
EDIT/OFF	OFF
AUTO VIEW/OFF	AUTO VIEW
DATA/TALK[1]	DATA
DIAL DISC/OFF[2]	OFF
SDLC/SS	SS.
EBCDIC(or Correspondence)/APL	Initially, EBCDIC or Correspondence as appropriate to your terminal keyboard. Subsequently, this switch is set to APL when communication with VS APL is to begin.
CALC/OFF	OFF
TEST/OFF	OFF
POWER/OFF	POWER

1 This switch appears on terminals supplied to World Trade countries except Germany.
2 This switch appears on terminals supplied to Germany only.

Figure 8. IBM 3767 Switch Settings

Like the IBM 2741, the IBM 3767 can be permanently connected to the computer through a non-switched (leased) line, or temporarily connected through a switched (dial) line.

If your terminal is operated through a non-switched line, once the switch settings in Figure 6 have been made, the connection to the computer is complete. The PROCEED light on your terminal console will go on. At this point you are ready to establish contact with TSO. If the light is not on, check to see that the security lock, if any, is in the ON position and the switches have been properly set.

To complete the connection procedure for a switched line, follow the instructions in Figure 9.

Connection Procedure

1. Press the TALK button, remove the handset from the cradle and dial TSO's telephone number. This number should be supplied to you by your system administrator.
2. Wait for a high-pitched tone. When you hear it, switch from TALK to DATA and replace the handset. (This procedure depends on the type of telephone set you're using; check with your system administrator if necessary.)
3. The DATA SET READY light should go on followed by the PROCEED light. You are now ready to establish contact with TSO. If the DATA SET READY light goes off at any time during the terminal session, try again from step 1.

Figure 9. Connecting the IBM 3767 (Switched Line)

Step 2: Contacting TSO (Logging On)

Once a connection has been made between your terminal and the computer, the next step is to contact the TSO system. This step is termed *logging on.* During the logon procedure it is important that you use the proper TSO printing element for the IBM 2741, or have the switch marked EBCDIC (or Correspondence)/APL set in the non-APL position for the IBM 3767.

The PROCEED light on your console will go on, the keyboard will unlock, or on an IBM 3270, "ENTER LOGON —" message will appear.

Now you're ready to enter the logon command. Type:

logon *userid* *si(nnn)*

where:

userid
is the identification assigned by your system administrator.

si(nnn)
is an optional size request to direct your session to a sufficiently large TSO foreground region to run VS APL.

Normally, TSO does not recognize terminals with APL print elements, or a switch to APL character mode during logon. If your installation has made provision for an entire TSO session to be run in APL character mode, there may be a special APL logon entry such as ")aplogon", or some other command provided for your terminal type by the system administrator.

If you make an error, TSO will not accept your logon. Instead, it will display the reason the logon request was unsuccessful. You must then repeat the logon command correcting the indicated problem, if possible.

If you typed the logon command correctly, the system will respond by requesting your entry password:

```
ENTER PASSWORD FOR  userid
```

Like your identification number, your entry password is issued by your system administrator.

On some terminals, display of the password is surpressed to prevent others from learning what your entry password is.

If the password you enter is not acceptable, TSO may repeat its prompt or ask you to re-enter the logon command.

Once your password is accepted, you are logged on to TSO. You may now, if you wish, enter any of the commands acceptable to TSO. For a description of these commands, refer to the *TSO Command Language Reference*. You may now also see a logon message indicating the time and date of logon. For instance:

```
LOGON IN PROGRESS FOR  userid  AT  date  time
```

What you do next depends on how procedures have been established at your place of work.

It is possible for the system administrator to provide an automatic connection to VS APL. If so, you will see the message:

```
v s  a p l
```

At this point, you should change your printing element to a standard APL element or set the switch on the 3767 to APL. The "vs apl" message means that you have established contact with VS APL and that TSO has given you an area of virtual storage to make your APL entries. This area of storage is your *active workspace.* The active workspace you receive is either:

- Continued—a workspace in which information has already been entered. You will receive a continued active workspace if you ended your last session with VS APL with a `)CONTINUE` or `)CONTINUE HOLD` command, or, if your previous terminal session had been abnormally terminated.
- Automatic—a workspace which is automatically loaded for you by the system as a result of the AUTOLOAD (*wsid*) parameter of the VSAPL command. The AUTOLOAD operand will override the loading of the CONTINUE workspace.
- Clear—a workspace where no information has been loaded. You get a clear active workspace if neither of the above options is activated.

You'll know that you have been given a clear workspace if VS APL displays the following message:

```
clear ws
```

If you receive a clear workspace, you'll be free to begin your entries. If you receive a CONTINUE or AUTOLOAD workspace, the following message will be displayed:

```
saved  time  date
```

where:

time date
is the time and date the CONTINUE or AUTOLOAD workspace was saved. Please note that the only way to receive a CONTINUE workspace is by ending the previous session with a *)CONTINUE* or *)CONTINUE HOLD* command, an abnormal termination, or with a forced ending under MVS. The *)OFF* command will DROP the CONTINUE workspace.

Step 3: Contacting VS APL

If you're going to be doing your work with VS APL, it may be that the logon command will automatically establish contact with VS APL and will prepare for APL entries.

If contact with VS APL is not automatically established, enter the following command:

VSAPL *LOADLIB(dsname dsname ...) APNAMES(apname apname ...) WORKSPACE(wssize) SHARESPACE(sharesize) FREESPACE(freesize) AUTOLOAD(wsname) TERMCODE(overbar)*

LOADLIB (dsname dsname ...)

Alias - LIB This operand is optional. If given, then it identifies one or more private load libraries from which auxiliary processors may be loaded. (Auxiliary processors are identified by the APNAMES operand).

Normally, auxiliary processors would be loaded from the system link libraries. However, if you have written and wish to use your own auxiliary processors, then you must place them in one or more of your own private load libraries. Then you must make your libraries known to VSAPL/TSO in either of two ways:

1) prior to invoking VSAPL, use the ALLOCATE command to allocate your libraries with the filename LOADLIB; or

2) use the LOADLIB operand so that VSAPL/TSO will perform the allocations for you.

If you do both, then the LOADLIB operand overrides the pre-allocation.

When the LOADLIB facility is used, the identified load libraries are searched first for the requested auxiliary processors. The system link libraries are searched only for those processors not found in the private load libraries.

Note, the LOADLIB operand has no affect if the APNAMES operand is not also used.

APNAMES (name name ...)

This operand is optional. If given, then it identifies the names of one or more auxiliary processors to be loaded into memory along with VSAPL. An auxiliary processor is a program which provides a communication path between the VSAPL environment and an external environment. For more information, refer to the Shared Variable Option topic (□SVO) in the VSAPL Language Manual.

Currently, the following auxiliary processors are available:

APL100 - allows the issuance of TSO commands from within the VSAPL environment;

APL101 - allows APL functions to STACK system commands to)LOAD,)SAVE,)COPY, or)PCOPY workspaces.

APL111 - allows access to OS sequential data sets from within the VSAPL environment.

If an auxiliary processor is to be used at any time during a VSAPL terminal session, then it must first be identified via the APNAMES operand when the session is first initiated.

Normally, auxiliary processors are loaded from the system link libraries; however, a facility is also provided to allow the loading of auxiliary processors from private load libraries. See the LOADLIB operand description for more information.

WORKSPACE (size)

Aliases - WSSIZE, SIZE This operand is optional. If given, then it specifies the minimum acceptable workspace size. If more memory is available, then the workspace size actually allocated is correspondingly increased.

"size" is a decimal number which, if given by itself or followed by a "K", represents kilo-bytes (one kilo-byte is 1,024 memory locations). If "size" is followed by an "M", then it represents mega-bytes (one mega-byte is 1,048,576 memory locations).

For WORKSPACE, the minimum valid size is 32k. The default size is also 32k.

SHARESPACE (size)

This operand is optional. If given, then it specifies the exact amount of memory to reserve for shared variables. A shared variable is a variable that is shared between VSAPL and an auxiliary processor. Consequently, it functions as a communication path. For more information, refer to the Shared Variable Option topic (□*SVO*) in the VSAPL Language Manual.

"size" is a decimal number which, if given by itself or followed by a "K", represents kilo-bytes (one kilo-byte is 1,024 memory locations). If "size" is followed by an "M", then it represents mega-bytes (one mega-byte is 1,048,576 memory locations).

For SHARESPACE, the default value depends upon whether or not any auxiliary processors are loaded (see the APNAMES operand description). If one or more auxiliary processors are loaded, then the default sharespace size is 4k; otherwise, the default is 0k. The minimum valid sharespace size is the same as the default.

Note, if no auxiliary processors are loaded, then the sharespace operand, if given, is ignored.

FREESPACE (size)

This operand is optional. If given, then it specifies the maximum amount of memory to make available for OS/TSO support functions. (This is especially significant if auxiliary processors are used). Any extra memory that might be available is made a part of the workspace.

If the requested freespace size is not available, then a warning message is issued.

"size" is a decimal number which, if given by itself or followed by a "K", represents kilo-bytes (one kilo-byte is 1,024 memory locations). If "size" is followed by an "M", then it represents mega-bytes (one mega-byte is 1,048,576 memory locations).

For FREESPACE, the default value depends upon whether or not any auxiliary processors are loaded (see the APNAMES operand description). If one or more auxiliary processors are loaded, then the default freespace size is:

MAXIMUM(32k,16k+2*SHARESPACE);

otherwise, the default is 10K.

Warning, making the freespace size too small can result in 804 or 80A system abends.

AUTOLOAD (wsname)

This operand is optional. If given, then it identifies the name of a workspace to be automaticly loaded after initialization. If this operand is omitted, then either a clear workspace or the CONTINUE workspace is loaded. The workspace name is given in the APL format, thus the following forms are valid:

AU(WSN) - loads a workspace named WSN fron the user's private, non-sharable library.

AU(1 NEWS) - loads the workspace named NEWS from the public library number 1.

AU(SECRET:PSWD) - loads a workspace named SECRET which has a password of PSWD.

TERMCODE (overbar)

This operand is required. If it is omitted, then it will be prompted for. This operand is used by VSAPL to determine what kind of character translation is required for proper communication with the user's terminal. For "overbar" use the APL overbar character which is done by holding the "shift" key while hitting the "2" key.

The following command requests contact with VS APL, asks for a shared memory size of 10,240 bytes, indicates two auxiliary processors named APL100 and APL111, and provides a terminal code for an EBCDIC 2741 terminal (upshift 2).

```
vsapl ap(apl100 apl111) sh(10k) te(@)
```

If the APL command is executed successfully, the system will return the message:

```
v s  a p l
```

and follow it with the message CLEAR WS or SAVED as appropriate. At this point, change the printing element to a standard APL printing element or set the switch on the IBM 3767 to APL. You can now begin your session with VS APL. It is also permissable to change your terminal to APL mode before typing the VSAPL command. On a 3767, you must switch your terminal to APL mode before typing the VSAPL command if you use the *termcode* parameter.

Example

Now that you have read through the description of logging on, let's look at an example. Let's assume that you are using a IBM 2741 terminal and that you have the proper courier printing element mounted. In addition, let's assume that the following has been supplied to you by your system administrator:

```
userid=gilbert
password=APLUSER
```

Here's what your logon procedure would look like, your entries are in lowercase characters, the system's responses are in uppercase characters:

```
logon gilbert si(80)
ENTER PASSWORD
apluser
CTF086 LOGON IN PROGRESS AT 14:14:34 ON JULY 4, 1976
READY
vsapl ap(apl100 apl101 apl111) f(30k) te(@)
```

At this point, the system will return the message "vs apl" followed by "clear ws". Mount the APL typing element or change your terminal to the apl character set. Now you're ready to begin your session with VS APL.

Ending the Work Session

When your work with VS APL is finished, you have various options as to what you do next. You can simply end the session, thereby breaking all connection with VS APL and TSO; you can end the session but maintain contact with TSO; or you can do either of these things and have your workspace stored at the same time. The way you direct your intentions to the system is through one of four APL system commands:

```
)OFF
)OFF HOLD
)CONTINUE
)CONTINUE HOLD
```

Ending Contact with VS APL and TSO

The APL system commands)*OFF* and)*CONTINUE* end your contact with VS APL and TSO.

The difference between these two commands is how they handle your active workspace. Once)*OFF* is executed, the information in your active workspace is lost and the CONTINUE workspace is dropped. Once)*CONTINUE* is executed, a copy of your active workspace is saved, making its contents available for your next VS APL session.

In response to the)*OFF* command or the)*CONTINUE* command, the system displays the message:

```
GILBERT LOGGED OFF TSO AT 14:32:18 ON JULY 4, 1976+
```

In this example the workspace is not saved. If)*CONTINUE* were specified, a date and time message for the save operation would precede the sign-off messages.

Ending Contact with VS APL Only

The APL system commands)*OFF HOLD* and)*CONTINUE HOLD* end your contact with VS APL but maintain your contact with TSO. When you specify)*OFF HOLD*the contents of your workspace are lost and the CONTINUE workspace is DROPPED. however, when you specify)*CONTINUE HOLD*, a copy of your workspace is saved making its contents available for your next session with VS APL.

When the)*CONTINUE HOLD* command is executed, the system prints a message showing the time and date the active workspace was saved, followed by the characters ⍴ ∈ ⍺ ⌊ ↑ (these are the characters READY; assuming a courier printing element). Once these messages are displayed, change your typing element or on the IBM 3767, set the switch marked EBCDIC (or Correspondence)/APL to EBCDIC or Correspondence. You are now in contact with TSO.

Breaking the Computer Connection

The step that actually completes your interaction with the computer is a physical one. It may simply be turning the terminal off and, if you're using a telephone, hanging up.

Normally, to finish your work at the terminal, you should enter the APL system command)*OFF* or)*CONTINUE* as appropriate. The system will take the indicated action, including the printing of accounting information. You can then hang the phone up and turn the terminal off.

If you turn the terminal off before entering these commands, the system disconnects you from TSO.

Forced Endings

Sometimes, certain malfunctions in the computer, such as temporary losses of power, or failures in the telephone circuits may endanger the integrity of your workspace. To safeguard against this possibility, the system will break the connection. This is called a *forced ending*. The MVS system handles a forced ending as if you had entered a)CONTINUE command. An attempt is made to SAVE a CONTINUE workspace. This saveguard is not available in MVT or SVS systems.

WORKSPACES AND LIBRARIES

This section describes the content and attributes of VS APL workspaces and details the libraries supported by TSO. Workspaces are areas of direct access or virtual storage that contain your work. Libraries are areas of direct access storage where copies of workspaces are stored.

The Workspace

A workspace is an area of virtual storage that contains all the data, functions, variables, and groups that you define in your VS APL entries. In addition, a workspace contains certain system variables and system functions that monitor or control the nature of your work.

When you establish contact with VS APL, you are automatically given a workspace to use. This is called your *active workspace.* All interaction that you have with VS APL from this point until you end your work is made through the active workspace.

It is possible to request that a duplicate of a workspace be saved in a library. When that is done, a copy of the entire active workspace is saved, including all the data, functions, status and control information within it. When you subsequently ask to retrieve a saved workspace, you receive a duplicate of what was saved. This restores everything to the way it was at the moment the workspace was last used.

Workspace Attributes

Each workspace that you use is qualified by certain *workspace attributes.* These attributes can be changed without affecting the variables and defined functions you have in your workspace, but they may affect the results of APL statements that are being executed.

Unless you're continuing some previous work, the workspace initially presented to you when you establish contact with VS APL is a *clear workspace.* This is a workspace in which no data has been entered, no names defined, and in which the workspace attributes indicate standard initial values. As you make entries, the values of some of these attributes change. This may happen implicitly as functions are being executed or as more of your workspace is being filled up. The values of the attributes may also change explicitly through certain system variables or system functions.

Figure 10 lists the attributes of a VS APL workspace, indicating the possible range of values that they can assume and the standard initial values they have in a clear workspace.

Seven of the attributes (index origin, latent expression, line counter, printing precision, state indicator, comparison tolerance, and random link) are fully described in the publication, *APL Language.* The remaining attributes involve workspace identification, size, and organization, topics that are covered in the following discussions.

Attribute	Possible Values	Value in a Clear WS	Can Be Changed By	Value Returned By
Name	See "Workspace Identification" in this section	*CLEAR WS*	*)WSID* *)SAVE*	*)WSID*
Password	See "Workspace Identification" in this section	None	*)SAVE*	
Size	See "Workspace Size" in this section	Depends on TSO region size or VSAPL invocation parameter	*)LOAD* *)CLEAR*	*)WSSIZE*
Symbol Table Size	See "Workspace Organization" in this section	256	*)SYMBOLS* number	*)SYMBOLS*
Execution Control Area	See "Workspace Organization" in this section	512	*)STACK* number	*)STACK*
Available Work Area	See "Workspace Organization" in this section.	Depends on TSO region size	Changed via user action	*□WA*
State Indicator	See "Workspace Organization" in this section	Empty	Changed via user action	*)SI* *)SINL*
Index Origin	0 or 1	1	*□IO*←number	*□IO*
Latent Expression	Any character vector	Empty	*□LX*←character vector	*□LX*
Line Counter	Integer Vector	Empty	Changed via user action	*□LC*
Printing Precision	1 through 16	10	*□PP*←number	*□PP*
Comparison Tolerance	0 through 2*¯32	*1E¯13*	*□CT*←number	*□CT*
Random Link	integer 1 through (2*31)-1	7*5	*□RL*←number	*□RL*

Figure 10. VS APL Workspace Attributes

Workspace Identification

Each workspace in VS APL is identified by a library number and a name. Library numbers are discussed in "The Library" later in this section.

A workspace name can be any combination of letters and numbers beginning with a letter. It cannot contain blanks, underscored letters, or special characters and should not be more than eight characters long. (Longer names are diagnosed with an error message in the *)LOAD*, *)SAVE*, *)COPY*, *)PCOPY* commands.)

Passwords

In addition to an identification, a workspace may have a password associated with it. A password is associated with a workspace if it is password-protected by the TSO PROTECT command or saved with the password option of)*SAVE*.

If a workspace is protected by a password, any system command that requests reading from the workspace (the commands)*LOAD*,)*COPY*, and)*PCOPY*) must include the proper password. If a workspace is to be saved over an existing password protected workspace, or is to be dropped, the system commands)*SAVE* or)*DROP* must include the proper password; for a save operation, if the workspace already is associated with the proper password, no password is required in the)*SAVE* command.

If an existing workspace password is previously known to VS APL (from a)*LOAD* or)*WSID* command), then it may be changed by specifying a new password on the)*SAVE* command.

Warning: if VS APL either doew not previously know a workspace's password, or if the password known to VS APL is incorrect, then an attempt to change the password via)*SAVE* will fail. You must then)*DROP* the previous workspace copy from disk, and then issue the)*SAVE* command for the active workspace specifying the password you wish to assign.

For example, the following command drops a workspace from a private/shareable library that is protected with the password SECRET:

```
)DROP 98226 TEST:SECRET
```

If you fail to specify the correct password when required, you will be prompted for one by the system.

To determine which workspaces, if any, require a password when referenced, ask your system administrator.

Note: APL password protection of workspaces is only available if your installation allows it. An attempt to password protect a workspace when password protection is not implemented on the system will produce a *SAVE* time-stamp which contains a + as the final character. If you receive such a time-stamp, type a ? (upshift-Q) to obtain more information.

Workspace Size

Unless you change it, the size of your active workspace corresponds to the size of your TSO region minus whatever is taken up by VS APL, auxiliary processors, shared storage, and freespace.

If you want to increase the size of your active workspace above this size, you must increase the size of your TSO region. To do this, you must provide the appropriate SIZE parameter in the LOGON command as described in the *TSO Command Language Reference.* The maximum workspace size that you can receive is 12 million bytes.

You can, however, directly control the size of your active workspace within the limits created by the TSO region size through the system commands:

```
)CLEAR size
```

and

```
)LOAD wsid size
```

The command `)CLEAR size` activates a clear workspace whose size is the value of *size.* The command `)LOAD wsid size` activates a workspace whose identification is represented by *wsid* and whose size is the value of *size.* The system will add some area to your size specification for its own needs so the size allocated is somewhat greater than you specify.

You can determine the size provided by the system for your active workspace by issuing the system command:

```
)WSSIZE
```

You can also issue the command `)QUOTA` to determine the default size of your active workspace. Under TSO this is also the maximum size that you can specify.

Workspace Organization

A workspace is divided into a number of separate areas in which different types of information are contained. Aside from controlling the overall size of your active workspace, you also can control the size allotted to specific areas of the workspace. This is particularly useful when a lack of space in one or another workspace area halts the progress of your work.

Figure 11 illustrates how the active workspace is divided by the system and the means you have available to monitor and control the size of these areas. Three areas are shown: the symbol table with its accompanying command `)SYMBOLS`, the execution control area with its accompanying command `)STACK` and the dynamic storage area with its accompanying system variable `⎕WA`.

The *symbol table* is used by VS APL to keep track of names occurring in the workspace. In a clear workspace, the number of entries permitted in the table is 256.

If you want to change the number of symbols accommodated in the symbol table, you may issue the command:

```
)SYMBOLS   number
```

where *number* is the number of symbols you want accommodated in the table. You can issue the `)SYMBOLS number` command only while your active workspace is clear.

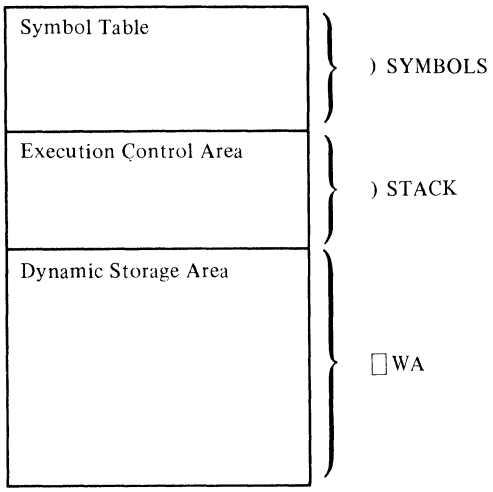


Figure 11. Organization of a Workspace

At any time during your session with VS APL, you can determine both the maximum number of entries that can be included in the symbol table and the current number of names in use, through the system command:

```
)SYMBOLS
```

The *execution control area* is used to contain information that is generated during function execution. In particular, the execution control area contains the *state indicator* which indicates the progress of function execution. (In a clear workspace the state indicator is empty.)

A clear workspace contains an execution control area that can accommodate 512 entries. You can modify the number of entries permitted in the execution control area through the system command:

```
)STACK  number
```

where *number* is the number of entries to be permitted. This command may be issued in any workspace. the stack *number* reduces the size of working storage ⎕*WA* by 4 ×*number*.

If the)*STACK* command is issued without a number, the currently allowable number of entries in the execution control area is displayed.

The remaining area, sometimes called the *dynamic storage area,* is the area in which data, function definitions, and group definitions are kept. Unlike the symbol table and the execution control area, the size of the dynamic storage area cannot be explicitly controlled. However the size of the dynamic storage area can be indirectly increased or decreased by respectively decreasing or increasing the size of the symbol table and execution control areas. Similarly, the size of the dynamic storage area will increase or decrease as the total size of the workspace is increased or decreased. As work progresses, you may find it useful to determine how much dynamic storage area is still unused. At any time during your work, you may determine the amount of dynamic storage remaining available by displaying the system variable ⎕*WA*.

The Library

Inactive workspaces that you or other users of VS APL have saved are stored in *libraries* which are identified by number. Libraries are classified as private, private/shareable, or public depending on their accessibility to users of the system.

Everyone who uses VS APL under TSO owns a private library, may also own one or more shared private libraries, and has access to all public and shared/private libraries in the system.

Private Libraries

As a user of VS APL under TSO, you own a private library which is available only to you. You can store copies of your active workspace there, retrieve copies of stored workspaces from it, display its contents, or drop workspaces from it when you no longer need them.

Any reference you have to a workspace in your private library is made using its workspace name. No library number need be supplied. You can, however, optionally specify the number 1001. For instance, either of the following commands could be used to retrieve a copy of a workspace named *GAMES* stored in your private library:

```
)LOAD GAMES
)LOAD 1001 GAMES
```

Private/Shared Libraries

A private/shareable library is intended for users who want to share workspaces. If you own a private/shareable library, you can perform any of the operations that you can perform on your private library. A private/shareable library may be owned by only one user at a time. A user may own more than one private/shareable library. You may issue a *)LIB*, *)LOAD*, *)COPY*, or *)PCOPY* command for any private/shareable library in the system, whether or not that library is owned by you. You may only issue the *)SAVE* or *)DROP* commands for private/shareable libraries which you own.

Any reference you have to a workspace in a private/shareable library is made using the library number of the private/shareable library and the workspace name. Additionally, the owner may have assigned a password to the workspace. If so, any reference you have to a workspace in the private/shareable library must indicate the proper password. (For further information, see "Passwords" earlier in this section.) For instance, the following command is used to copy the object *ITEM* from the workspace *ACCOUNTS* in project library 2000; ACCOUNTS is protected with the password *PSW*:

```
)COPY 2000 ACCOUNTS:PSW ITEM
```

Your system administrator may restrict the range of private/shareable library numbers available to you. Within the range permitted to your userid, ownership of an unused library number is obtained by issuing a *)SAVE* request specifying the number. Ownership of the library number is relinquished when you *)DROP* the last workspace in the library.

Public Libraries

A public library is available to all users of VS APL under TSO. You may list the contents of a public library or retrieve copies of workspaces stored there. You are not allowed to save or drop workspaces in a public library, however, you can request your system administrator to perform these operations for you.

Any reference you have to a workspace in a public library is made using the library number of the public library and the workspace name.

Workspaces and Libraries in the TSO Environment

Although your interaction with VS APL operates in terms of workspaces and libraries, TSO organizes these collections of information as OS sequential datasets. To TSO, each workspace is an OS sequential dataset, so that a library is simply a collection of datasets with the same high level index.

For the most part, this internal representation will not be apparent to you.

Exporting Workspaces

Workspaces may be transferred to other VS APL FOR TSO systems by using standard OS Utilities (IEBGENER). Private workspaces may only be)*LOAD*ed by the TSO userid who created the workspace. Therefore, if you wish to transfer a PRIVATE workspace to another TSO user using OS Utilities, whether that TSO user is on your same system or not, you must first issue a)*SAVE* command to place the workspace into one of your Private/Shareable libraries.

If you are trying to send or receive a workspace to VSAPL For CMS or VSPC, you may copy the workspace to tape using OS Utilities (IEBGENER), and then use MOVEFILE under CMS, or the VSPC SERVICE Utility to IMPORT the workspace. For these two systems, the PRIVATE workspace restriction cited above for TSO systems does not apply.

This page intentionally left blank

VS APL System Commands

This section describes the system commands available when you use VS APL under control of TSO. Through these commands you can request the system to:

- Monitor and control the content and attributes of the active workspace.
- List the contents of your private library, or any public or private/shareable library.
- Retrieve workspaces from all libraries except other users' private library.
- Copy all or part of a workspace.
- Determine the status of halted functions.
- Send messages to other users of TSO and to the TSO Operator.
- Sign-off VS APL and TSO.

Using System Commands

To use system commands you simply enter them at your terminal just as you enter APL statements. System commands cannot be used in APL expressions and cannot be executed as part of or input to a function definition. Otherwise, a system command can be entered any time you are in contact with VS APL, in which case it is executed immediately.

System Command Types

System commands can be grouped into five types based on the kinds of operations they perform:

- Library Control Commands, which are used to control the contents of libraries.
- Workspace Control Commands, which are used to control the contents and attributes of the active workspace.
- Inquiry Commands, which are used to return information about libraries and the active workspace.
- Communication Commands, which are used to communicate information to other terminals.
- Sign-Off Commands, which are used to sign-off the system.

Library Control Commands

Library control commands are used to control the contents of libraries. There are two system commands that belong in this category: `)SAVE`, which saves a copy of the active workspace in a library, and `)DROP`, which drops a workspace from a library.

Saving the Active Workspace: The)SAVE Command

You can save a copy of your active workspace by issuing the following command:

)SAVE wsid :password

where:

wsid
is a workspace identification to be assigned to the active workspace before saving. If omitted, the workspace identification currently associated with the active workspace is retained. If the workspace does not have a name, *wsid* must be specified.

:
is an optional separator, required only when a password is indicated.

password
is used to issue the PROTECT command to password protect the OS sequential dataset containing the user's workspace. If dataset protection is not implemented on the TSO system, *:password* will have no effect.

The date and time message will contain a '+' as the final character to warn you that the workspace has not been protected. Typing a '*?*' following this message will produce a more descriptive message explaining that password protection is not implemented on your system. If you require this protection, see your systems administrator.

When the *)SAVE* command is issued, a copy of the active workspace is stored in the library indicated by the current workspace identification. If *wsid* is specified, the current workspace identification is changed to the one indicated, provided the new name does not already exist in the library.

When you save a copy of your active workspace, all the attributes of the active workspace are retained by the stored copy. In particular, the userid, the library number and workspace name are stored with the copy, and the present size of the active workspace becomes the size of the copy. Most important, all the objects in the active workspace (functions, variables, and so on) retain their values in the stored copy (any shared variables are saved with their most recent values).

Saving a copy of your active workspace replaces the copy, if any, in your library, but does not destroy your active workspace. While a copy now exists in a library, the original is still available until you activate another workspace or sign-off.

After a successful save operation, the system responds with a message showing the time and date when the save operation was performed.

Figure 12 illustrates two instances of saving. In the first case, a copy of an active workspace named *TABLES* is saved with no change in its workspace identification. Notice that the response returned by the system includes the name of the saved workspace. In the second case, a copy of the active workspace is saved with a new workspace identification, a password is used for the save operation.

```
      )SAVE
11:15:27  07/06/76  TABLES
      )SAVE 123456 NEWNAME:ABC
14:22:01  07/06/76
        -- or --
      )SAVE 123456 NEWNAME:ABC
14:22:01  07/06/76 +
      ?
PASSWORD PROTECTION NOT AVAILABLE
        -- or --
      )SAVE 123456 NEWNAME:ABC
NOT SAVED +
      ?
YOU ARE NOT PERMITTED TO USE THIS LIBRARY
```

Figure 12. Using the)SAVE Command

Saving is not permitted when the workspace identification given in the command matches an identification of an existing saved workspace but does not match the identification of the active workspace. In other words, if you attempt to save a workspace named TEST with the command

```
)SAVE 1975 NEWNAME
```

and *NEWNAME* already exists in library 1975, the save operation will not be executed. This restriction prevents you from inadvertantly overwriting one workspace with another. If you want to overwrite the workspace, you have to first change the identification with the)*WSID* command before you attempt to save it.

You may only save a workspace in a private/shareable library if you own that library; ie., if that library number is assigned to your TSO userid. If the private shareable library number does not exist, and your TSO userid has been authorized by the Systems Administrator to use this number, that library number will be allocated to you with the first workspace saved.

You may not save a workspace in a public library directly. If you want a workspace saved in a public library, see your system administrator.

Dropping a Workspace from a Library: The)DROP Command

You can drop a workspace from a library by issuing the command:

```
)DROP wsid :password
```

where:

wsid
is the workspace identification of the workspace to be dropped.

:
is an optional separator, required only when a password is indicated.

password
is required if the workspace to be dropped was password protected using the TSO PROTECT command, or as a parameter of the)SAVE command.

If the workspace is not password protected, the *:password* has no effect and is ignored.

The system responds with a message indicating the time and date the workspace was dropped. For example, to drop a workspace named FINANCE from your private library, you specify:

```
      )DROP FINANCE
14:22:27 07/10/77
```

If you want to drop the workspace ACCOUNT from private/shareable library 995053 associated with password VS202, specify:

```
      )DROP 995053 ACCOUNT:VS202
14:07:13 07/08/77
```

You may not drop a workspace from a public library directly. To have a workspace dropped from a public library, see your system administrator.

Workspace Control Commands

Workspace control commands are used to change the contents and attributes of the active workspace. The workspace control commands are:

-)LOAD which retrieves a copy of a saved workspace and brings it into the active workspace.
-)COPY which copies global objects from a stored workspace into the active workspace; the active workspace is not protected.
-)PCOPY which copies global objects from a stored workspace into the active workspace; the active workspace is protected.
-)GROUP which gathers global objects in the active workspace into a group or disperses a group.
-)CLEAR which activates a clear workspace.
-)ERASE which erases global objects from the active workspace.
-)WSID which changes the workspace identification of the active workspace.
-)SYMBOLS which changes the number of entries allowed in the symbol table area of the active workspace.
-)STACK which changes the number of entries allowed in the execution control area of the active workspace.

Retrieving a Workspace from a Library: The)LOAD Command

To retrieve a saved workspace from a library, you specify the following system command:

)LOAD wsid :password size

where:

wsid
is the workspace identification of the workspace to be retrieved.

:
is an optional separator, required only when a password is indicated.

password
is required if the workspace was saved using the password option, or the TSO PROTECT command was issued for the OS dataset containing the workspace. The system will prompt for the correct password if it is not specified when required.

size
is an optional size specification, required when you want to change the size of your active workspace.

When you retrieve a saved workspace, a complete copy of the saved workspace is brought into your active workspace. All the workspace attributes have the values they had when the saved workspace was last active. In addition, all the objects in the retrieved workspace (functions, variables, and so on) retain their previous values. Any shared variables in the previously active workspace are retracted.

In retrieving a workspace, you do not destroy the library copy. In effect, two copies now exist, one in the library and one in your active workspace. Unless you specifically overwrite the library copy, or the workspace is dropped from the library, the library copy always remains intact.

If the *)LOAD* command is successful, the system returns a message indicating the time and date the workspace was last saved. The system may also follow this with the message *WSSIZE IS size* if the retrieved workspace changes the size of your active workspace and you did not explicitly define a size in the *)LOAD* command. The new workspace size is indicated by *size*.

Figure 13 illustrates how you can use the *)LOAD* command to retrieve a copy of a stored workspace. The first example in Figure 12 shows how you can retrieve a workspace named *FINANCE* from your own library. In the second example, a password-protected workspace named *GAMES* is retrieved from public library 57. The final example indicates how *GAMES* can be retrieved and an explicit size set for the active workspace.

```
      )LOAD FINANCE
SAVED 11:15:27 07/06/77

      )LOAD 57 GAMES:FUN
SAVED 11:10:20 12/12/77
WSSIZE IS 152412

      )LOAD 57 GAMES:FUN 150000
SAVED 18:16:55 04/08/77
```

Figure 13. Using the)LOAD Command

If you specify a size in the)*LOAD* command that is larger than the maximum workspace size authorized to you, the command will be rejected. If you don't specify a size, but the workspace you specify is larger than the maximum workspace size authorized to you, the system will attempt to pare the workspace down to the allowable limit. If all the data in the workspace cannot fit into this smaller area, the)*LOAD* command will be rejected.

Copying Objects into the Active Workspace: The)COPY and)PCOPY Commands

The system commands)*COPY* and)*PCOPY* are used to copy one or more objects from a stored workspace into your active workspace.

)*COPY* and)*PCOPY* differ in the way they handle a copied object when another object of the same name appears in the active workspace. The)*COPY* command replaces an existing object, while the)*PCOPY* command does not make the replacement.

The)*COPY* and)*PCOPY* commands are specified as follows:

)*COPY* *wsid :password objects*

)*PCOPY* *wsid :password objects*

where:

wsid
is the workspace identification of the workspace to be copied from.

:
is an optional separator, required only when a password is specified.

password
is required if the workspace was saved using the password option, or the TSO PROTECT command was issued for the OS dataset containing the workspace. The system will prompt for the correct password if it is not specified when required.

objects
are the names of objects to be copied. Each name must be separated by at least one space. If no objects are listed, all global objects in the stored workspace are assumed.

Figure 14 shows how you can use the)*COPY* and)*PCOPY* commands to copy various objects from *STORED*, a workspace stored in your library, into *ACTIVE*, your active workspace. The upper portion of Figure 13 shows the contents of *STORED* and *ACTIVE* before copying. The names within braces are members of the group named *GROUP*.

If you want to copy the objects in *STORED* named *A* and *GROUP* without protecting objects in *ACTIVE* from replacement, the command you issue is:

```
)COPY STORED A GROUP
```

The system responds with the date and time *STORED* was last stored.

STORED and *ACTIVE* Before Copying:

STORED			*ACTIVE*		
NAMES		*VALUES*	*NAMES*		*VALUES*
A		5	*A*		10
GROUP	*B*	20.2	*GROUPX*		15
	C	groupname		*Y*	25
	Z	37.7		*Z*	35

ACTIVE After Issuing the Command)*COPY STORED A GROUP*:

ACTIVE		
NAMES		*VALUES*
A		5
X		15
Y		25
GROUP	*Z*	37.7
	B	20.2
	C	groupname

ACTIVE After Issuing the Command)*PCOPY STORED A GROUP*:

ACTIVE		
NAMES		*VALUES*
A		10
GROUP	*X*	15
	Y	25
	Z	35
B		20.2
C		groupname

Figure 14. Using the)COPY and)PCOPY Commands

If you want to copy *A* and *GROUP* into *ACTIVE* but you also want to protect objects in *ACTIVE* from replacement, the command you issue is:

```
)PCOPY STORED A GROUP
```

The middle portion of Figure 13 illustrates the contents of *ACTIVE* after execution of the *)COPY* command. Notice that the definition of *GROUP* in *STORED* has replaced the definition of *GROUP* in *ACTIVE*. Notice, too, that the values of *A* and *Z* are now the values of *A* and *Z* in *STORED* and that the groupname *C* is copied into *ACTIVE*. When a member of a copied group is a groupname, only its definition is brought into the active workspace, not its members.

The lower portion of Figure 13 illustrates the contents of *ACTIVE* after execution of the *)PCOPY* command. Notice that the definition of *GROUP* and the objects *A* and *Z* are protected from replacement. The objects *B* and *C* are copied in *ACTIVE*.

To copy all the objects from a stored workspace into the active workspace, you don't specify any object names. For example, the following command copies without protection all the objects from the workspace *OTHER* in library 6065:

```
)COPY 6065 OTHER
```

The following command copies all the objects in *OTHER* while protecting objects in the active workspace from replacement:

```
)PCOPY 6065 OTHER
```

When you copy objects from a stored workspace you should be aware of the following:

- Shared variables that are replaced by copied objects are retracted.
- If there is not enough room in your workspace, some of the objects may not be copied.
- If copying creates more symbols than can be accommodated in the symbol table, some of the objects may not be copied.
- Any attempt to copy an object with the same name as a halted function (a function that has not completed execution) will cause the message *SI DAMAGE* to be printed.

Grouping Items Together: The)GROUP Command

You may find it convenient at times to group the names of related items together in your workspace. You can form a group (or redefine or extend one you have already created) through the following system command:

```
)GROUP  groupname  names
```

where:

groupname
is the name to be associated with a newly-created group or the name of an existing group to be modified.

names
is an optional list of names to be associated with *groupname*. If specified, each name listed must be separated from a preceding name by at least one space.

The)*GROUP* command can be used in three ways:

- If you want to create a new group, you specify)*GROUP groupname names*. The names in *names* are grouped together under the name *groupname*. If a group named *groupname* already exists, the new group definition replaces any previous group definition.
- If you want to add one or more names to an existing group, you specify)*GROUP groupname groupname names*. The names in *names* are added to the definition of *groupname*.
- If you want to disperse a group, that is disassociate a groupname from a list of names, you specify)*GROUP groupname*.

Suppose, for example, you have three weather forecasting functions *PRECIP*, *BAROM*, and *TEMP*. At your option, you can group these names under a groupname *WEATHER* as follows:

```
)GROUP WEATHER PRECIP BAROM TEMP
```

If you want to add the names *MEAN* and *AVG* to the existing defintion of *WEATHER*, you specify:

```
)GROUP WEATHER WEATHER MEAN AVG
```

When you want to disperse the group *WEATHER*, you simply specify:

```
)GROUP WEATHER
```

The names you choose as a groupname cannot already be in use as a global variable or as a defined function name. If the name is already in use in this way, any attempt to specify the name as a groupname, is rejected.

Clearing the Active Workspace: The)CLEAR Command

The)*CLEAR* command is used to discard all the objects contained in your active workspace. The command is specified as follows:

)*CLEAR* *size*

where:

size
is an optional size specification that is used to control the size of the cleared active workspace.

When you specify)*CLEAR*, all the objects in the active workspace are erased, any shared variables are retracted, and all the attributes of the active workspace are set to the standard values shown in Figure 9 in the section "Workspaces and Libraries." In particular, the size of the active workspace is set to a default value.

When you specify)*CLEAR size,* your active workspace is cleared and its size is changed to a value indicated by *size.* The system adds a certain amount of space to your size specification for its own needs. If the size you specify is larger than the maximum allowable size, the command is rejected. (You can determine this maximum by issuing the)*QUOTA* command.)

In response to the command)*CLEAR*, the system returns the message *CLEAR WS*. The system may also follow this with the message *WSSIZE IS size,* if clearing the active workspace changes its size, the value *size* is the current size of the active workspace. If the)*CLEAR* command is specified with a size operand, the system returns the message *CLEAR WS*, only.

Figure 15 illustrates how the active workspace can be cleared. In the first example shown, no workspace size is explicitly specified. In the second example shown, the workspace size is explicitly specified.

```
      )CLEAR
CLEAR WS
WSSIZE IS 73000
      )CLEAR 100000
CLEAR WS
```

Figure 15. Using the)CLEAR Command

Erasing Objects in the Active Workspace: The)ERASE Command

You can erase any global variable, globally defined function, or group in your active workspace with the command:

```
)ERASE  objects
```

where:

objects
are the names of one or more objects (global variables, defined functions, or groups) to be erased; each indicated object is separated from another by at least one space.

Suppose, for example, your active workspace contained a global variable *A*, a function *MATH*, and a group named *GRP1*, these objects would be erased by the specification:

```
)ERASE A MATH GRP1
```

When a group is erased all the objects named in the group are erased with it. If one of the objects in a group is, in turn, the name of a group, its definition is erased but not its members. To erase a group without erasing its members, use the)*GROUP* command.

When you use the)*ERASE* command, be aware of the following:

- If you erase an object that has been offered for sharing, the share offer for the object is retracted.
- If you erase a suspended function (a function that has not completed execution) the execution of the suspended function or any function awaiting the suspended function's completion cannot be resumed.

Identifying the Active Workspace: The)WSID wsid Command

You can assign an identification to your active workspace as follows:

```
)WSID  wsid :password
```

where:

wsid
is the workspace identification to be assigned to the active workspace.

:
is an optional separator, required only when a password is specified.

password
is an optional password to be associated with the active workspace.

When you specify the)WSID command, any current identification that your active workspace has, is replaced by the new identification. The system acknowledges the identification change by displaying the message:

```
WAS  wsid
```

where *wsid* is CLEAR WS if the workspace had no identification, or is the previous identification that the workspace had.

If *password* is specified, the new password overrides any password associated with the active workspace. Otherwise, any current password is erased.

In the following example, a new identification and password are associated with the active workspace.

```
      )WSID REPORTS:UPDATE
WAS 123456 REPORTS
```

Controlling the Symbol Table: The)SYMBOLS number Command

All the names that occur in your active workspace are maintained in an area of the workspace called the symbol table. In a clear workspace the maximum number of entries permitted in the table is 256. This includes entries for function and variable names, labels, names used in function definition, group names, and names appearing in a group list.

If you want to change the number of symbols permitted in the symbol table, you can do so only while your active workspace is clear. In that case, you can use the following system command:

```
)SYMBOLS  number
```

where:

number
is the number of entries you want accommodated in the symbol table. The system may increase this number slightly for its own convenience.

In response to the)*SYMBOLS number* command, the system returns the message:

WAS number

where *number* is the former number of entries permitted.

For instance, in a clear workspace you can change the number of symbols permitted as follows:

```
      )SYMBOLS 512
WAS 256
```

If your workspace is not clear, any attempt to assign a new size for the symbol table is rejected.

The maximum value acceptable in the)*SYMBOLS* command depends on the amount of unused area in your workspace. The larger you permit the symbol table to be, the smaller you permit the dynamic storage area to be. If you indicate a value in)*SYMBOLS* so large that it doesn't leave enough room for a usable dynamic storage area, the command will be rejected.

Controlling the Execution Control Area: The)STACK number Command

The execution control area is an area in your workspace that contains information that is generated during function execution and contains the state indicator, which tracks the progress of executing functions.

In a clear workspace, the number of entries that can be reserved for temporary names and the state indicator is 512. However, you have the option of directly specifying how many entries you want accommodated in the execution control area as follows:

```
)STACK  number
```

where:

number
is the number of entries you want accommodated in the execution control area.

In response to)*STACK number*, VS APL returns:

WAS number

where *number* is the previous number of entries permitted.

For instance, you can enlarge the execution control area of a workspace to 650 entries from 500 entries by specifying:

```
      )STACK 650
WAS 500
```

The maximum value acceptable in the)*STACK* command depends on the amount of unused area in your workspace. The larger you permit the execution control area to be, the smaller you permit the dynamic storage area to be. If you indicate a value in)*STACK* so large that it doesn't leave enough room for a usable dynamic storage area, the command will be rejected. The useable dynamic storage area indicated by □*WA* is reduced by 4× *number*.

Similarly, if you specify a value for the execution control area too small to contain the names generated by executing functions, the command will be rejected.

Inquiry Commands

Inquiry commands are used to display the current state of the active workspace, display the contents of libraries, and to indicate the availability of space in libraries and in the active workspace. Inquiry commands are also used to indicate the workspace, library, and shared variable quotas. The inquiry commands are:

-)*WSID*, which displays the identification of the active workspace.
-)*SYMBOLS*, which displays the number of entries currently in the symbol table and the maximum number of entries permitted.
-)*STACK*, which displays the number of entries currently permitted in the execution control area.
-)*WSSIZE*, which displays the size of the active workspace.
-)*LIB*, which lists the names of the workspaces in a designated library.
-)*QUOTA*, which lists the values established for maximum and default workspace sizes, number of shared variables and shared variable size. This command also displays the remaining library space available for use.
-)*FNS*, which lists the names of the global defined functions in the active workspace.
-)*VARS*, which lists the names of the global variables in the active workspace.
-)*GRPS*, which lists the names of the groups in the active workspace.
-)*GRP*, which lists the members of a designated group.
-)*SI*, which displays the contents of the state indicator.
-)*SINL*, which displays the contents of the state indicator with local names.

Listing the Identification of the Active Workspace: The)WSID Command

At any time during your session with VS APL, you may request a display of the active workspace identification, by issuing the system command:

```
)WSID
```

In return, VS APL displays the workspace identification currently associated with the active workspace, or the message:

```
IS CLEAR WS
```

if your active workspace has no identification.

Figure 16 indicates how a workspace identification is listed. In the first example, a workspace is given an identification of `1484 SECRET`, which is then displayed via the `)WSID` command. In the second example, a workspace is saved with a new identification. When the workspace identification is that of a private workspace, no library number is displayed.

Assign an identification:

```
      )WSID 1484 SECRET
WAS CLEAR WS
```

What is the identification?

```
      )WSID
IS 1484 SECRET
      )SAVE UPDATE:WRITEPW
13:17:33 07/15/76
      )WSID
IS UPDATE
```

Figure 16. Using the)WSID Command

Monitoring the Symbol Table: The)SYMBOLS Command

If at some time during your work, you're not sure of the status of the symbol table, you can specify the following:

```
      )SYMBOLS
```

In reply, VS APL displays the message:

IS max; number IN USE

where *max* is the current number of entries permitted in the symbol table, and *number* is the number of entries that have been made so far. In a clear workspace *max* is 256 and *number* is 0.

For instance:

```
      )CLEAR
CLEAR WS
      )SYMBOLS
IS 256; 0 IN USE
      A←57
      )SYMBOLS
IS 256; 1 IN USE
      )CLEAR
CLEAR WS
      )SYMBOLS 350
WAS 256
      )GROUP ALPHA A B C D E
      )SYMBOLS
IS 350; 6 IN USE
```

Monitoring the Execution Control Area: The)STACK Command

If you issue the command:

```
)STACK
```

with no number following it, VS APL will display the number of entries currently permitted in the execution control area. The value is specified as follows:

IS number

where *number* is the number of entries currently permitted.

In a clear workspace the value returned is 512:

```
      )CLEAR
CLEAR WS
      )STACK
IS 512
```

Listing the Workspace Size: The)WSSIZE Command

If you want, you can determine the size of your active workspace by issuing the command:

```
)WSSIZE
```

For example:

```
      )WSSIZE
176988
```

In a clear workspace, if not otherwise specified, the value returned by)*WSSIZE* is dependent on the size of your TSO region.

Listing the Workspaces in a Library: The)LIB Command

The)*LIB* command is used to generate an alphabetized list of workspaces in a given library. The format of the command is:

```
)LIB   libnum   letters
```

where:

libnum
is an optional library number, required only if the library is not your private library.

letters
is an optional series of letters, required when the list of workspace names is to alphabetically follow the indicated letters.

Suppose, for instance, the following workspaces were in your library:

```
INDEX
FINANCE
FACTORS
ACCOUNT
```

If you specify)*LIB*, the list generated would be:

```
ACCOUNT  FACTORS  FINANCE  INDEX
```

If however, you specify:

```
)LIB  FI
```

the list would include only those workspaces whose names begin with *FI* or follow alphabetically:

```
)LIB  FI
FINANCE  INDEX
```

Listing Workspace, Library, and Shared Variable Quotas: The)QUOTA Command

The)*QUOTA* command is used to list information about library, workspace, and shared variable availability. The command is specified as follows:

```
)QUOTA
```

In response, the following message is displayed:

```
LIB   libquota    FREE   remaining
WS    default     MAX    max
SV    number      SIZE   size
```

where:

libquota
is 0 and is displayed only for compatibility with other VS APL implementations, since library quotas are not implemented in VSAPL For TSO.

remaining
is 0 and has no use in TSO.

default
is the default size of the active workspace in bytes.

max
is the maximum workspace size you can request in bytes.

number
is the maximum number of variables that can be shared.

size
is the maximum permitted size of a shared variable in bytes.

A typical request for current library, workspace, and shared variable availability, might look like this:

```
      )QUOTA
LIB          0 FREE          0
WS      176988 MAX      176988
SV          10 SIZE       1024
```

Listing the Defined Functions in the Active Workspace: The)FNS Command

You can list the names of the **globally** defined functions appearing in your workspace by issuing the system command:

```
)FNS  letters
```

where:

letters
is a series of letters, required only when the list of function names is to alphabetically follow the indicated letters.

In response, VS APL displays all the functions in the active workspace in alphabetical order.

For example, if four functions, *MATH*, *GEOG*, *HIST*, and *LANG* were defined in your workspace, the command:

```
)FNS
```

would return:

```
GEOG HIST LANG MATH
```

The same information may be retrieved and processed in an APL function (though not in alphabetic order) by using the ⎕*NL* 3 system variable. In this example, ⎕*NL* 3 might return a 4 x 4 character matrix containing:

```
LANG
GEOG
MATH
HIST
```

The variable names may be in a different sequence.

If one of the functions is suspended and contains locally defined function names in the header, these names will be added to the ⎕*NL* output. The system command)*SINL* will display the names of these locally defined functions.

If you follow)*FNS* with one or more letters, VS APL displays only those functions whose names follow those letters in alphabetical order:

```
      FNS L
LANG MATH

      )FNS GI
HIST LANG MATH

      )FNS P
```
(nothing is displayed)

Listing the Variables in the Active Workspace: The)VARS Command

At any time during your terminal session, you can determine the **global** variables that appear in your workspace, as follows:

```
)VARS  letters
```

where:

letters
is a series of letters, required only when the list of variable names is to alphabetically follow the indicated letters.

VS APL responds, by listing in alphabetical order, all the global variables in the active workspace; no local variables are listed.

Suppose, for example, your workspace contained the global variables, *AMT*, *VALUE*, *COST*, and *RATE*, then the command:

```
)VARS
```

would return:

```
AMT  COST  RATE  VALUE
```

⎕*NL* 2 returns 4 x 5 character matrix containing local and global variable names (though not in alphabetic order) which may be processed in APL. In this workspace ⎕*NL* 2 might return:

```
RATE
AMT
VALUE
COST
```

If a function is suspended in the workspace which contain locally defined variables (variable names in the function header), these variable names will not appear in the)*VARS* system command output. The names will appear in the ⎕*NL* 2 output and the display produced by the system command)*SINL*.

If)*VARS* is followed by one or more letters, VS APL displays only the functions whose names follow those letters in alphabetical order:

```
      )VARS V
VALUE
      )VARS RAT
RATE VALUE
      )VARS RAV
VALUE
```

Listing the Groups in the Active Workspace: The)GRPS Command

The system command)*GRPS* lists the names of all the groups defined in your workspace in alphabetical order. The command is specified as follows:

)*GRPS* *letters*

where:

letters
is a series of letters, required only when the list of groupnames is to alphabetically follow the indicated letters.

For example, in a workspace containing the groups *FINFNS*, *ACCTFNS*, and *TEMPVARS*, a request to list the groups could be stated as follows:

```
      )GRPS
ACCTFNS  FINFNS  TEMPVARS
```

Like)*FNS* and)*VARS*, the)*GRPS* command can be optionally followed by one or more letters which control the group names that are printed. VS APL displays only those group names that alphabetically follow the letters listed in the)*GRPS* command:

```
      )GRPS  TEMP
TEMPVARS
      )GRPS  F
FINFNS  TEMPVARS
```

Listing the Members of a Group: The)GRP Command

Suppose you have a group in your workspace, but are not sure of its members. The way you can receive this information is by issuing the system command:

)*GRP* *groupname*

where:

groupname
is the name of the group you are interested in.

In response, the system displays the members of the group you specify. There is no response if the group does not exist in your active workspace:

```
      )GRP  FINANCE
GROSS  NET
```

The group named *FINANCE* is defined with members *GROSS* and *NET*.

```
      )GRP  MASTER
(nothing displayed)
```

The indicated group named *MASTER* does not exist in the workspace.

Displaying the State Indicator: The)SI and)SINL Commands

One of the items maintained in your active workspace is a state indicator which contains information on the progress of defined function execution. If a defined function halts execution, a display of the state indicator lists the names of halted functions in order, starting with the most recently halted first. The list may also include the symbols ⎕ or ⍞ if operations pertaining to these symbols are pending. For each function listed, the state indicator shows the line on which work was halted.

With this information available, you can either:

- Resume your work at the point it halted by entering → *n*, where *n* is a line number in the most recently halted function.
- Terminate all currently halted functions by entering → for each suspended function in the state indicator list.

A request to display the state indicator may be specified in two ways:

```
)SI
)SINL
```

If)*SINL* is specified, VS APL displays the state indicator with any names local to each function.

If no functions are currently halted,)*SI* or)*SINL* returns nothing.

For example, a typical use of the)*SI* command is shown below:

```
      )SI
FN3[4] *
FN2[5]
FN1[2]
```

In this example, the function *FN*1 called the function *FN*2 which called the function *FN*3. The * after *FN*3 indicates that it is *suspended,* that is, it has not completed execution because of some error of its own or because you explicitly halted the function while it was executing. The functions *FN*2 and *FN*1 are *pendent*, that is, they are awaiting the completion of *FN*3 before completing their execution. The bracketed numbers following each function name indicate the statement to be executed in the function.

If the)*SINL* command were issued in the same situation, the results might appear as follows:

```
      )SINL
FN3[4]    *   A
FN2[5]    LABEL
FN1[2]
```

In this example, the name *A* is local to suspended function *FN*3, the name *LABEL* is local to the pendent function *FN*2, and the pendent function *FN*1 has no names local to it.

If the name of a function appears in the state indicator list, you should not attempt to erase it using the)*ERASE* command or attempt to replace it using a)*COPY* command. If you erase or copy a halted function, you make it impossible to resume its execution. A function that cannot be resumed is a *damaged* function.

A damaged function is displayed in the state indicator with a bracketed line number of ¯1:

```
      )SI
FN3[4] *
FN2[5]
FN1[2]
      )ERASE FN3
SI DAMAGE
      )SI
FN3[¯1] *
FN2[5]
FN1[2]
```

Sometimes editing a function in the state indicator list will also damage the function. If the function is suspended, damage will result if:

- The function header is modified.
- The order of its labels is changed.
- Labels are added or deleted.

If the function is pendent, damage to the function will result if any line is modified, moved, deleted, or inserted.

Communication Commands

Communication commands are used to send messages to other users of the system and to the TSO operator. The communication commands are:

-)*MSG*, which sends a message to another user.
-)*OPR*, which sends a message to the TSO operator.
-)*MSG OFF*, which suspends the reception of messages.
-)*MSG ON*, which restores the reception of messages.

Sending Messages: The)MSG and)OPR Commands

During your session at the terminal you may want to communicate with other users of the system or with the TSO operator.

You can send a message to another user by specifying the following command:

```
)MSG   userid  message
```

where:

userid
is the TSO user identification of the user you are directing the message to.

message
is a line of text to be transmitted. As many characters may be entered as will fit on the remainder of the line.

It is not possible for VS APL to determine what type of terminal another user may operating. Therefore, it is unable to send APL graphics to other users. No error message will be produced if APL graphics are included in a message, but the output at the other terminal may be garbled.

You can send a message to the TSO operator by specifying the following command:

```
)OPR   message
```

As soon as you enter the)MSG or)OPR commands, (if your terminal is not a 3270), your terminal will prevent you from making further entries, so that it can print any response that is sent to you.

A 3270 terminal under TSO cannot block input without also inhibiting PA1. Therefore, normal execution continues immediately after sending a message.

The next thing you should see at your terminal is the system response:

```
SENT
```

This means that the system has posted your message for eventual delivery to its intended recipient. It does not necessarily mean that transmission has begun.

Once your terminal prevents you from making entries, it remains so until you enter a weak interrupt signal (by pressing the ATTN key once).

A message can be received by you any time your terminal is set to display output. In fact, you can receive a message during the execution of a function or between the displayed rows of an array.

When a message is displayed, it is followed by the TSO userid.

Example)MSG

Figure 17 illustrates a typical exchange between two users of VS APL under TSO. The first user (*GILBERT*) initiates communication by sending a message to user *COLE*. In response to his *)MSG* command, the sender's keyboard locks and the system displays *SENT*. Meanwhile, user *COLE* receives the message and sends a reply. In response to his reply, his keyboard locks and *SENT* is displayed. User *COLE* now unlocks his keyboard by entering a weak interrupt signal.

After user *GILBERT* receives the reply, he too unlocks his keyboard and continues his work.

User GILBERT

```
      )MSG COLE YOUR SHARED LIBRARY NUMBER?
SENT
I'M USING LIBRARY 2001 COLE
```

User COLE

```
YOUR SHARED LIBRARY NUMBER? GILBERT
      )MSG GILBERT I'M USING LIBRARY 2001
SENT
```

Figure 17. Sending Messages

If for some reason, your message cannot be posted, the *SENT* message will not be displayed. You can then discard the message by entering a weak interrupt signal. You will then receive the report:

```
      MESSAGE LOST
```

You can now try to resend your message if you want.

When you send a message to another user, he must be signed on to the system. If not, you will receive the report:

```
      USER NOT LOGGED ON
```

You should try again later, or, if you have made a mistake in specifying the user's identification, you should try again now, with a corrected user identification.

If a user has blocked messages, and you attempt to send a message to him, you will receive the report:

```
      USER NOT RECEIVING
```

and your message will not be transmitted.

TSO does not supply a method of displaying the userids of logged on users (though many installations have written commands to do so). Unlike previous APL implementations, there is no *)PORTS* or *)USERS* system command in VS APL. So, unless your installation has one of the command processors for displaying logged-on users, sending messages with the *)MSG* command will be a process of trial and error.

An alternative method of sending a message to a user to ensure that he receives it, even if he is logged off or not receiving, is to send the message using the function TSO or MSG found in *LIB 1 TSO* . The user may issue the *TSO SEND* command with the *L* option as follows:

```
TSO 'SE ''message-text'' U(userid ) L'
     -- or --
'userid' MSG 'message text'
```

These functions require auxiliary processor APL100. The *MSG* function will date and time stamp messages, and keep an optional message log in your workspace.

Blocking Messages: The)MSG OFF Command

If you do not want to be disturbed by incoming messages, you can tell the system to block them from reaching your terminal. You can do this by issuing the command:

```
)MSG OFF
```

Once you issue this command, no user may direct a message to your terminal. If a user attempts to send a message your way, the report:

```
USER NOT RECEIVING
```

will be printed at his terminal and the message will not be transmitted.

The system operator may still send high priority messages which will break through your message block.

If you need to communicate briefly with another user or with the TSO operator, you may issue the *)MSG* or *)OPR* commands. These commands will temporarily restore message acceptance at your terminal. While your terminal is locked in response to either command, you can receive messages from other users or the TSO operator. However, once you enter a weak interrupt signal, message blocking will be restored.

It is important to note that message blocking remains in effect for your entire terminal session and not simply for your session with VS APL. If you sign off of VS APL and maintain contact with TSO, message blocking will continue for your terminal.

Restoring Message Acceptance: The)MSG ON Command

If you want to restore message acceptance permanently after message blocking, you can issue the command:

```
)MSG ON
```

No direct reply is made by the system. At this time, you may receive messages which may have been queued for your terminal by issuing the TSO command through APL100:

```
TSO 'LISTBC'
```

assuming your workspace has the APL function 'TSO' defined in LIB 1 TSO, and you have access to auxiliary processor 100.

Sign-Off Commands

Sign-off commands are used to end a session with VS APL and TSO or to end the VS APL session only and remain under control of TSO. There are four commands in this category. Two of the commands save a copy of the active workspace as the session is ended while the remaining two discard the active workspace as the session is ended and DROP the CONTINUE workspace. The sign-off commands are:

- `)OFF`, which ends the session with VS APL, TSO; the active workspace is lost.
- `)OFF HOLD`, which ends the session with VS APL only; the active workspace is lost.
- `)CONTINUE`, which ends the session with VS APL, TSO; the active workspace is saved.
- `)CONTINUE HOLD`, which ends the session with VS APL only; the active workspace is saved.

For a detailed description of each of the sign-off commands, refer to "Ending the Work Session" in the section "The Work Session."

AUXILIARY PROCESSING

An auxiliary processor is a program that enables you to perform operations that are not normally available through APL. For instance, an auxiliary processor gives you the opportunity of creating and referencing OS sequential datasets outside of your workspace.

Three auxiliary processors are available with the VS APL for TSO Installed User Program as follows:

Name	Description
APL100	TSO Command Processor. This processor is used to enter TSO commands.
APL101	Stack Input Processor. This processor is used to store data to be used at the next terminal input request.
APL111	OS Sequential Dataset Processor. This processor is used to read from or write to OS datasets supported by the Queued Sequential Access Method (QSAM).

To use an auxiliary processor, its name must first be specified in the APL command that initially establishes contact with VS APL. For a description of the APL command, see "Step 3: Contacting APL" in the section "The Work Session."

In addition to the auxiliary processors available with VS APL, you can write additional auxiliary processors to handle other operations. For a description of how you can create your own auxiliary processors, refer to "VS APL for CMS: Writing Auxiliary Processors."

This implementation of shared storage management follows the conventions described in that manual.

Communicating With an Auxiliary Processor

There are generally five steps in communicating with one of the auxiliary processors distributed with VS APL:

1. Initialize a variable with information needed by the auxiliary processor.
2. Offer the initialized variable for sharing with the auxiliary processor.
3. Check the variable for a return code.
4. Send and/or retrieve information via the shared variable.
5. End the procedure by retracting the variable.

VS APL includes a public library workspace *APFNS* which contains, among other items, functions that perform one, more, or all of the five steps indicated for communication with an auxiliary processor. For further information on *APFNS*, see "VS APL Functions for Auxiliary Processors" later in this section.

Initializing a Variable

Before an auxiliary processor can begin operating, there are certain items of information it must have. For instance, the type of data conversion you need performed or the ddname allocation of an OS sequential dataset you want to access. You communicate this kind of information through the following initializing assignment:

x ← '*argument* (*options*)'

where:

x
is the variable to be shared with the auxiliary processor.

argument
indicates the source or destination of subsequent operations, for instance, the ddname of an OS sequential dataset to indicate the destination of an output operation.

(
indicates options are to follow; it should be indicated only when options are specified.

options
indicate various processing options, for instance data conversion or stacking order. If an option is specified more than once, the rightmost specification applies.

)
is an optional delimiter that may be omitted if desired.

For example, the following assignment indicates a destination of TSO and that data is to be translated into an internal APL format:

```
STACK←'TSO (APL'
```

Certain input/output operations require two variables to be initialized. The first variable is generally used to transmit data and is called a *record variable.* The second variable is used to control or monitor data transmission and is called a *control variable.*

Offering a Variable for Sharing

An offer to share a variable with an auxiliary processor is made as follows:

```
a  □SVO  b
```

where:

a
is the number of the auxiliary processor. This number is usually 100 for the TSO Command Processor, 101 for the Stack Input Processor, 111 for the OS Sequential Dataset Processor. However, your installation may choose to change them. If you're in doubt, contact your system administrator.

□SVO
is a system function used to issue shared variable offers.

b
is the name, in quotes, of the variable or variables to be shared. When more than one variable is to be shared, each name can be specified as a row in a character array. You can share up to 14 variables with each auxiliary processor.

When a variable is offered for sharing, the system responds with a status indication of the sharing operation. This status indicator is called the *degree of coupling* and consists of the numbers:

- 0 if the share offer has not been made
- 1 if the share offer has been made
- 2 if the share offer has been matched by the auxiliary processor, that is sharing is complete

If more than one variable is offered, the degree of coupling is a vector with one element for each variable offered.

As an example of a shared variable offer, the following expression offers two variables *X* and *CTL* to auxiliary processor 111:

```
      111 □SVO 2 3⍴'X CTL'
2 2
```

The response 2 2 indicates that sharing for both variables has been completed.

Each auxiliary processor sets an *access control vector* for the shared variable that controls the sequence in which the variable can be set or referenced. You can change the setting of the access control vector, but cannot affect communication with the auxiliary processor by doing so except as noted in the discussion headed "The OS Sequential Dataset Processor."

Checking for a Return Code

Once a variable has been offered and matched, the auxiliary processor inspects the argument and options specified in the variable. In response, the auxiliary processor assigns the following *return code* to the variable:

- 0 or a vector whose first element is 0—the initialization values are acceptable
- 1—the initialization values are not acceptable

If the return code is 1, correct the error by assigning a valid initial value to the variable.

Sending or Retrieving Information through the Shared Variable

Once a shared variable has been referenced for a return code, any subsequent reference of the variable retrieves information from the auxiliary processor. Any setting of the variable sends data, commands, or control information to the auxiliary processor.

For example, once a variable *COMMAND* has been shared with the TSO Command Processor, it can be used to issue the TSO command LISTBC by specifying:

```
COMMAND←'LISTBC '
```

Once a record variable, *REC* has been shared with the OS Sequential Dataset Processor and checked for a return code, the next reference to *REC* reads a data record from the OS sequential dataset:

```
    111⎕SVO 'REC'
2 variable sharing complete
    ⎕←REC
0 share was accepted
      REC
(a record is returned)
```

Ending the Procedure

To end communication with an auxiliary processor, the shared variables are retracted. This is done through the following system function:

```
⎕SVR b
```

where:

b
is the name in quotes of the shared variable to be retracted. Once a variable is retracted, it may be reinitialized and reoffered.

The system responds with the degree of coupling that the variable has prior to retraction.

Another method of retracting variables automatically is to declare them LOCAL to the involking function (by placing them in the function header). When the function terminates, the variable is automatically retracted.

A shared variable is also automatically retracted when replaced by variable in the)*COPY* command, or when a new workspace is activated by)*LOAD* or)*CLEAR*.

The TSO Command Processor

The TSO Command Processor, auxiliary processor 100, is used to dynamically establish contact with TSO so that you can enter TSO commands.

Any TSO command can be executed by processor 100 except:

HELP

TEST [1]

TIME

LOGON, LOGOFF, and EXEC commands will not be processed until VSAPL is terminated by)*OFF* or)*CONTINUE*. Sufficient storage must be available in freespace to hold the TSO command and its data.

Note: [1] This limitation does not prohibit running VSAPL FOR TSO using the TEST facility of TSO. This may be required for the purposes of debugging installation written auxiliary processors. The procedure is fully documented in the *VSAPL FOR TSO Systems Guide*. The limitation described here is only for auxiliary processor 100.

Initialization Values: The TSO auxiliary processor ignores the initial value of the shared variable.

Return Codes: Once the initial value is accepted, any subsequent reference of the variable will produce a return code set by the auxiliary processor or TSO. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example:

The following function *MAIL* uses the TSO Command Processor to request messages queued for his terminal while he was logged off, or not receiving.

```
      ∇ MAIL;MODE
[1] 100 ⎕SVO 'MODE'
[2] MODE←'LISTBC MAIL'
      ∇
```

In this example, the first statement offers a variable for sharing with auxiliary processor 100. The second statement requests TSO to display MAIL left in the BROADCAST dataset.

The Stack Input Processor

The stack input processor, auxiliary processor 101, is used to create a stack of data to be used when the system is next ready to accept terminal input.

You build the stack by assigning to a shared variable each line of data as a character vector or scalar. Options specified when the shared variable is initialized indicate how you want the data stacked and what type of conversion you want applied to the stacked data.

If a strong interrupt signal is issued or a character error is detected as the stacked data is being used, the entire contents of the stack is deleted and the terminal opens for input.

Initialization Values: The initialization values for communication with the Stack Input Processor are:

TSO(*order conversion*)

where:

TSO
indicates that data is to be placed on a stack maintained by TSO. If omitted, TSO is assumed.

(
indicates options are to follow; it should be indicated only when options are specified.

order
indicates whether the processor places the data at the beginning of the stack (*BEG* or *LIFO*) or at the end (*END* or *FIFO*). The default is *FIFO*.

conversion
indicates the conversion to be applied. Note that the conversion options apply only to specification of the shared variables. Input is read from the stack as though it had been typed in from the terminal prior to stacking, and the conversion option does not apply to this part of the process.

The conversion options are:

- 192—full graphic character translation from APL to EBCDIC. This includes direct translation of compound APL characters to unique EBCDIC codes. This should be used if APL compound graphics are stacked for a 3270 terminal session.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by TSO. Compound characters are expanded to their constituent parts. For example, A̲ is converted to A, backspace, _. This option should be used if compound APL graphics are stacked for a non-3270 terminal session.
- 370—limited graphic character translation from APL to EBCDIC. Conversion is performed as indicated in Figure 18. Characters that cannot be translated directly are transmitted as blanks.

The default is 370.

If no initial value is specified, or the initial value is null, the defaults are used.

Return Codes: If the initial value is invalid, a code of 1 is returned in the shared variable, otherwise a 0 is returned. Once the initial value is accepted, any subsequent reference of the variable will produce a return code set by the auxiliary processor or TSO. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example: Figure 19 illustrates a function *CHECKPOINT* that uses the Stack Input Processor to save a copy of the active workspace and then returns.

In execution, *CHECKPOINT* builds a two line stack. The first line to be executed is the)*SAVE* command, which saves a copy of the active workspace. The second line to be executed is a branch to *LABEL* which resumes execution.

APL	EBCDIC
A - Z	a - z
~	¬
∧	&
⍞	"
⍺	@
÷	%
≠	$
⍙	#
⍬	-0 (equivalent to })
⍵	+0 (equivalent to {)
←	=
_	only when translating from APL to EBCDIC -

The following characters are common to APL and EBCDIC and are translated directly:

A through Z 0 through 9 blank < = > + - ÷ . : ; , ? !) (/ \ | _ * '

Figure 18. APL/EBCDIC Conversion via 370 Conversion Option

```
      ∇CHECKPOINT;S
[1]   ⍝USE TSO STACK,LAST IN,FIRST OUT
[2]   S←'TSO (APL BEG'
[3]   ⍝SHARE S AND IGNORE RESULT
[4]   S←101 ⎕SVO 'S'
[5]   ⍝RESUME EXECUTION
[6]   S←'→LABEL'
[7]   ⍝SAVE THE ACTIVE WS
[8]    S←')SAVE TEMP'
[9]   ⍝SET STOP VECTOR TO OPEN KEYBOARD
[10]  S⍙CHECKPOINT←LABEL
[11]  LABEL:⍝RETURN TO CALLER
      ∇
```

Figure 19. Stack Input Processor Application

Note:

The primary use of processor 101 is to stack those commands which cannot be issued by the APL execute operator or to convey information from one workspace to another across the)*LOAD* operation. The use of compound characters in the stack processing should be avoided, even though conversion options for them are provided. While simple APL graphics are independent of terminal type, the handling of overstrikes is different for 3270 and non-3270 terminals, and the stack processor cannot guarantee correct execution across terminal types in such cases.

The APL101 stack is 512 characters in length. Any attempt to place more pending commands on the stack will produce a *STACK OVERFLOW* error message. The APL stack is emptied by ATTN, system error, overflow, or APL termination by)*OFF* or)*CONTINUE*.

You cannot stack TSO commands for execution after VS APL terminates using APL101. An alternative approach is to use APL111 to create a CLIST dataset, APL100 to issue the 'EXEC clist' command, and APL101 to issue the)*OFF HOLD* command. The CLIST will start executing immediately upon APL termination.

The OS Sequential Dataset Processor

The OS Sequential Dataset Processor, auxiliary processor 111, is used to sequentially read from or write to any TSO dataset supported by the Queued Sequential Access Method (QSAM). The OS dataset must first be accessed through the TSO command, ALLOCATE. The ALLOCATE command is described in the *TSO Command Language Reference.*

All record processing using the OS Sequential Dataset Processor is done through a record variable. Once sharing is completed and you check for a return code, the first reference of the record variable reads the first record from the OS sequential dataset; each successive reference reads the next record from the OS sequential dataset. The first setting of the record variable writes the first record; each successive setting writes the next record to the OS sequential dataset.

You may optionally share a control variable. If you share one, the control variable can be used to check return codes for each read or write operation. If you set a value for the control variable, it is ignored.

Initialization Values: The initialization values for communication with the OS Sequential Dataset Processor are:

ddname (*CTL conversion*)

where:

ddname
is the ddname of the device to be accessed. It must be the ddname defined by a ALLOCATE command already issued to TSO.

CTL
is specified only if this variable is a control variable. If omitted, the variable is established as a record variable. If CTL is specified, any specified conversion option is ignored. The control variable must be shared after the record variable is shared; it is associated with the record variable for which an identical ddname was specified.

conversion

indicates the conversion to be applied to the processed data. Conversion is performed as indicated in Figure 20. Characters that cannot be translated directly are transmitted as blanks.

The conversion options are:

- 370—limited graphic character conversion from APL to EBCDIC. On output, the data part of a character vector or scalar is written one byte per element, characters that cannot be translated are sent as blanks. On input, the record is transmitted as a character vector, one element per byte, characters that cannot be translated are accepted as the APL character (∘).
- 192—full graphic character translation from APL to EBCDIC. This includes direct translation of compound APL characters to unique EBCDIC codes. On output, the data part of a character vector or scalar is written one byte per element. On input, the record is transmitted as a character vector, one element per byte.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by TSO. Compound characters are expanded to their constituent parts. For example, $\underline{A}$ is converted to A, backspace, _ on output and _, backspace, A is converted to $\underline{A}$ on input. On output, the data part of a character vector or scalar is written. On input, the record is transmitted as a character vector.
- VAR—on output, the entire variable including its size, shape and type information is written with no conversion. On input, the entire record is transmitted.
- BIT—on output, the data is transmitted as a vector or scalar one bit per element; each element of data must have a value of 1 or 0. On input, the record is transmitted as logical vector, one element per bit.
- BYTE—on output, the data is written with no conversion, one byte per element. On input, the record is transmitted as a character vector or scalar, one element per byte. The data conversion functions in the public library workspace *APFNS* can be used to convert this information. (For a description of *APFNS*, see "VS APL Functions for Auxiliary Processing" later in this section.)

The default is VAR.

)

is an optional delimiter that may be omitted if desired.

Return Codes: If the initial value of the record variable is invalid, a code of 1 is returned in the record variable, otherwise, the record variable contains a scalar 0.

If the initial value of the control variable is invalid, a code of 1 is returned in the control variable, otherwise the control variable contains 0.

After each read or write operation, the control variable, if any, contains the return code of the previous operation. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Auxiliary Processor Return Codes

Figure 21 lists and describes the codes that may be returned in response to auxiliary processor operations. With the TSO Command Processor and the Stack Input Processor, you reference the shared variable to obtain the return code. The input/output auxiliary processors return codes in the control variable.

In addition to the codes listed in Figure 20, other codes can be returned by auxiliary processors 100 and 111. With the TSO Command Processor, the return code is generally the one from the command that you previously assigned to the shared variable; these codes are described in "TSO Command Language Reference." If an I/O error occurs when using the OS Sequential Dataset Processor, a code with a decimal value is returned. When this code is converted to its four-byte hexadecimal representation, the first two bytes are the sense bytes and the last two are the status bytes. Sense and status bytes are described in the "Status Information" section of "OS/VS Data Management Macro Instructions."

Code	Description
0	No error exists
1	Attempt to execute an unknown TSO command. unknown TSO command or Stack Overflow.
2	ATTENTION interrupted execution of TSO command. read error.
4+	Return code returned from TSO command.

Figure 21. Auxiliary Processor Return Codes

VS APL Functions for Auxiliary Processing

VS APL includes a public library workspace named *APFNS* which can be used to simplify auxiliary processing. *APFNS* contains various types of functions. Some functions directly communicate with an auxiliary processor. Others can be used to prepare for auxiliary processing. Other functions still, can be used to help in converting data once an auxiliary processing operation has been executed.

Figure 22 lists the functions in *APFNS* summarizing the operations they perform. For detailed information on any one of these functions display the variable *DESCRIBE∆fn* in *APFNS*, where *fn* is the name of the function. For example, display *DESCRIBE∆TSO* for information on the function *TSO*. For the functions *ECI*, *ECO*, *FI*, *FO*, *II*, *IO*, *LI*, *LO*, *PDI* *PDO*, display the variable *DESCRIBE∆DATACV*.

Function Name	Description
TSO	Allows you to issue certain TSO commands.
ECI	Converts EBCDIC characters to APL characters.
ECO	Converts APL characters to EBCDIC characters.
FI	Converts System/370 floating-point numbers to APL format.
FO	Converts APL numbers to floating-point format.
F∆O	Issues a TSO ALLOCATE command and shares two variables with auxiliary processor 111.
II	Converts System/370 binary values to APL format.
IO	Converts APL numbers to System/370 binary format.
LI	Converts System/370 logical data to APL format.
LO	Converts APL numbers with values of 0 or 1 to System/370 logical format.
PARSE∆	Breaks up a character vector into tokens. Used by F∆O.
PDI	Converts System/370 packed-decimal numbers to APL format.
PDO	Converts APL numbers to System/370 packed-decimal format.

Figure 22. Functions in APFNS

S/370 Data Conversion

Functions To Support Conversion of System/370 Internal Data Formats

General Comment on Errors:

If the following functions encounter an invalid argument, they print a message consisting of the function name followed by the words *DOMAIN ERROR* . They then exit without assigning a result. Hence the application function will halt with a *VALUE ERROR* pointing to the data conversion function which was expected to provide a value. The argument can then be checked by the following definitions for each function.

EBCDIC Characters In

```
A←ECI C
```

'C' is a character array.

Result: A character array of the same size and shape as 'C'. It contains the APL characters which are equivalent to the EBCDIC characters represented in ''C''. Translation is performed using the global variable 'ZC'.

EBCDIC Characters Out

```
C←ECO A
```

'A' is a character array

Result: A character array of the same size and shape as 'A'. It contains the APL characters which are equivalent to the APL characters in 'A'. Translation is performed using the global variable 'ZC'. If any character in 'A' does not have an equivalent EBCDIC character then the function exits with no result.

Floating In

```
F←FI C
```

'C' is a character array, the last coordinate of which must have a length of 4 or 8. The last coordinate thus represents either single or double precision System/370 floating point numbers.

Result: An array of numbers equivalent to the floating point representations in 'C'. The rank of 'F' is one less than the rank of 'C'.

```
ρF←→¯1↓ρC
```

Floating Out

```
C←FO F
```

'F' is a numeric array.

Result: A character array whose last coordinate must have a length of 8, and which contains the System/370 double precision floating point representations of the numbers in 'F'. The rank of 'C' is one greater than the rank of 'F'. If single precision is required, then drop the last four columns of the result.

```
ρC←→(ρF),8
```

Integers In

```
I←II C
```

'C' is a character array whose last coordinate has a length of between 1 and 7 inclusive, and which contains the System/370 binary representations of integers. The limit of 7 characters or 56 bits is the maximum precision representable in APL.

Result: An array of integers representing the binary numbers in 'C'. The rank of 'I' is one less than the rank of 'C'.

```
ρI←→¯1↓ρC
```

Integers Out

```
C←N IO I
```

'I' is an array of integers.

'N' is an integer scalar not greater than 7. It gives the number of bytes in which each integer is to be represented. 'N' must be large enough to represent the largest magnitude of the integers in 'I'.

Result: A character array whose last coordinate contains the System/370 representation of the logical data in the last coordinate of 'L'. The rank of 'C' is one greater than the rank of 'I'.

```
ρC←→(ρI),N
```

Logical In

```
L←LI C
```

'C' is a character array whose last coordinate contains System/370 logical data or a string of boolean bits.
Result: A numeric array consisting of zeros or ones representing the logical data in 'C'. The rank of 'L' is the same as the rank of 'C', but the last coordinate of 'L' is 8 times as long as the last coordinate of 'C'. A scalar value for 'C' produces an eight element vector.

```
ρL←→(¯1↓ρC),8×¯1↑ρC)÷8
```

Logical Out

```
C←LO L
```

'L' is a numeric array consisting of only zeros and ones. The length of its last coordinate must be a multiple of 8.

Result: A character array whose last coordinate contains the System/370 representation of the logical data in the last coordinate of 'L'. The rank of 'C' is the same as the rank of 'L' but the length of the last coordinate of 'C' is one-eighth of the length of the last coordinate of 'L'.

```
ρC←→(¯1↓ρL),(¯1↑ρC)÷8
```

Packed Decimal In

```
I←PDI C
```

'C' is a character array whose last coordinate must have a length between 1 and 16 inclusive, and which contains the valid System/370 packed decimal representation of integers.

Result: An array of integers representing the packed decimal numbers in 'C'. The rank of 'I' is one less than the rank of 'C'.

```
ρI←→¯1↓ρC
```

Note that if the length of the packed decimal numbers is greater than 9 bytes, then a loss of precision may result.

Packed Decimal Out

```
C←N PDO I
```

'*I*' is an array of integers.

'*N*' is an integer scalar not greater than 16. It gives the number of bytes in which each integer is to be represented. '*N*' must be large enough to represent the largest magnitude of the integers in '*I*'.

Result: A character array whose last coordinate contains the System/370 packed decimal representations of the integers in '*I*'. The rank of '*C*' is one greater than the rank of '*I*'.

```
ρC←→(ρI),N
```

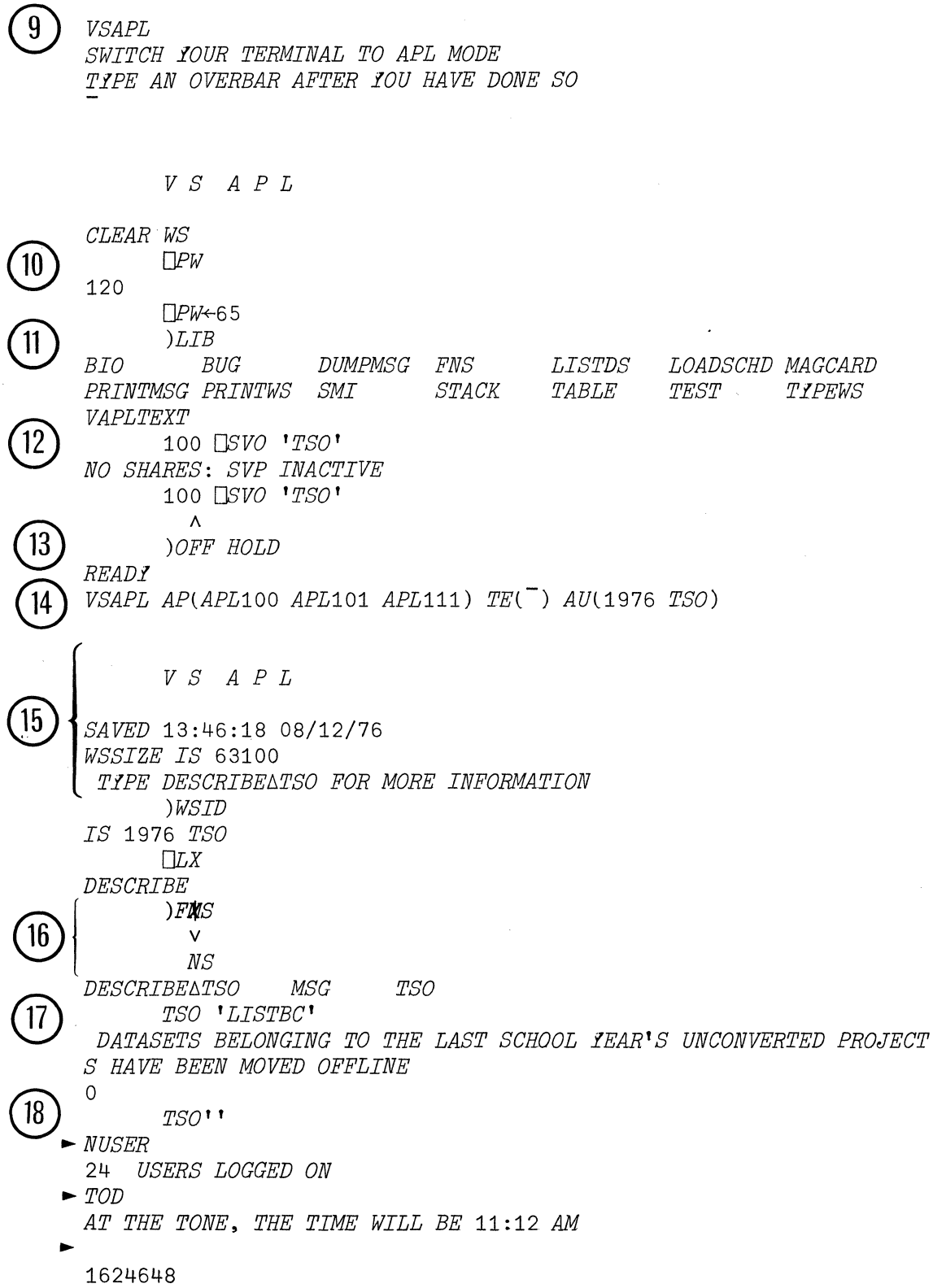
Group for Data Conversion Functions

'*DATACVGP*' is the name of the group containing the objects related to support of conversion of System/370 internal data formats.

SAMPLE TERMINAL SESSION

```
)APLOGON DUNBAR SI(300)
ENTER PASSWORD FOR DUNBAR-
)
DUNBAR LOGON IN PROGRESS AT 11:02:44 ON AUGUST 17, 1976
DATASETS BELONGING TO THE LAST SCHOOL YEAR'S UNCONVERTED PROJECTS HAVE BEEN MOVED OFFLINE
)
28  USERS LOGGED ON
READY
)
LISTC
αPL.BIO
αPL.BUG
αPL.DUMPMSG
αPL.FNS
αPL.LISTDS
αPL.LOADSCHD
αPL.MAGCARD
αPL.PRINTMSG
αPL.PRINTWS
αPL.SMI
αPL.STACK
αPL.TABLE
αPL.TEST
αPL.TYPEWS
αPL.VAPLTEXT

αPL01976.BUGS
αPL01976.LISTDS
αPL01976.NEWS
αPL01976.PFK
αPL01976.PRINTWS
αPL01976.QUADAV
αPL01976.TEST
αPL01976.TSO
αPL01976.TSOFNS
αPL01976.TYPEFNS
αPL01976.TYPEVAR
αPL01976.TYPEWS
αPL01984.NEWS
αPL05100.LINK5100
αPL05100.MOVE5100
αPL05100.RLNK5100
αPL05100.TRANS
APL.CLIST
APLCNVT.JCL.DATA
APLCOYCC.ASM
)
AP101.CLIST
BS.CLIST
COBOL.CNT↓
READY
```


9
VSAPL
SWITCH YOUR TERMINAL TO APL MODE
TYPE AN OVERBAR AFTER YOU HAVE DONE SO
¯
V S A P L
CLEAR WS
10
⎕PW
120
⎕PW←65
11
)LIB
BIO BUG DUMPMSG FNS LISTDS LOADSCHD MAGCARD
PRINTMSG PRINTWS SMI STACK TABLE TEST TYPEWS
VAPLTEXT
12
100 ⎕SVO 'TSO'
NO SHARES: SVP INACTIVE
100 ⎕SVO 'TSO'
^
13
)OFF HOLD
READY
14
VSAPL AP(APL100 APL101 APL111) TE(¯) AU(1976 TSO)
V S A P L
15
SAVED 13:46:18 08/12/76
WSSIZE IS 63100
TYPE DESCRIBE△TSO FOR MORE INFORMATION
)WSID
IS 1976 TSO
⎕LX
DESCRIBE
)FNS
16
v
NS
DESCRIBE△TSO MSG TSO
17
TSO 'LISTBC'
DATASETS BELONGING TO THE LAST SCHOOL YEAR'S UNCONVERTED PROJECT
S HAVE BEEN MOVED OFFLINE
0
18
TSO''
► NUSER
24 USERS LOGGED ON
► TOD
AT THE TONE, THE TIME WILL BE 11:12 AM
►
1624648

```
)LOAD 1976 PFK
SAVED 13:28:28 08/12/76
THE FUNCTIONS IN THIS WORKSPACE ARE DESIGNED TO
DEMONSTRATE THE PFK FUNCTION KEYS OF THE 3270

      ⍝ THIS TERMINAL ENTRY SIMULATES PRESSING THE 3270 PFK KEYS
      PFK 1

      PFK'S 10 11 AND 12 ARE DELAY FUNCTIONS
YOU ARE EXPECTED TO TYPE SOMETHING OR JUST HIT ENTER
      PFK 7
      )FNS
      DESCRIBE        PFK5    PFK6    PFK9    TRYIT   PFK
      PFK 12
     )PCOPY  1976 TSO
      NOT COPIED:  DESCRIBE
SAVED 13:46:18 08/12/76
      X←'(APL'
      101⎕SVO'X'
2
      X←')WSID'
      IS 1976 PFK
      X
0
      2⎕SVQ''
101
      (0≠⎕SVO M)/M←⎕NL 2
X
      100⎕SVO 3 1⍴'ABC'
2 2 2
      A←'SE ''ECHO'' U(DUNBAR) L'
ECHO DUNBAR
      B←'LIST ABC'
DATA SET ABC NOT IN CATALOG
      ⎕SVR (0≠⎕SVO M)/M←⎕NL 2
2 2 2 2
```

(28)

```
      MSGΔLOG
VALUE ERROR
      MSGΔLOG
            ^
      MSGΔLOG←''
      TEAM←4 7⍴'COLE    FREEMANGILBERTOSGOOD '
      TEAM
COLE
FREEMAN
GILBERT
OSGOOD
      TEAM MSG ''
NEW FUNCTIONS IN 1976 TSO, GANG ---
0
            MSGΔLOG

08/17/76  11:34:48 NEW FUNCTIONS IN 1976 TSO, GANG ---  ' U⍳COLE
08/17/76  11:34:48 NEW FUNCTIONS IN 1976 TSO, GANG ---  ' U⍳FREEMAN
08/17/76  11:34:48 NEW FUNCTIONS IN 1976 TSO, GANG ---  ' U⍳GILBERT
08/17/76  11:34:48 NEW FUNCTIONS IN 1976 TSO, GANG ---  ' U⍳OSGOOD
```

(29)

```
      )SAVE MSGLOG:PASSWORD
11:49:20 08/17/76+
      ?
PASSWORD PROTECTION NOT AVAILABLE IN THIS SYSTEM
      )SAVE TEST
NOT SAVED, THIS WS IS MSGLOG+
      ?
YOU ARE ATTEMPTING TO CHANGE THE NAME OF YOUR WORKSPACE, BUT THE NEW NAME ALRE
Y EXISTS IN THE LIBRARY.+
      ?
IF THIS IS REALLY WHAT YOU WANT TO DO, THEN USE THE )WSID COMMAND TO CHANGE TH
WORKSPACE NAME BEFORE ATTEMPTING THE )SAVE
      ?
SYNTAX ERROR
      ?
      ^
```

(30)

```
)LOAD LISTDS
SAVED 12:56:21 08/17/76
```

(31)

```
     ∇LISTDS[□]∇
     ∇ LISTDS;□IO;TSO;STK;REC;X;Y;Z
[1]   □IO←0
[2]   Z←100 □SVO 'TSO'
[3]   STK← 'TSO(FIFO 192)'
[4]   Z←101 □SVO 'STK'
[5]   REC← 'SYSIN(370'
[6]   →OPEN⌈⍳0≠□NC 'UID'
[7]   ⍞←Z← 'ENTER YOUR TSO USERID: '
[8]   UID←(⍴Z)↓⍞
[9]  OPEN:Z←111 □SVO 'REC'
[10]  Z←REC
[11]  TSO← 'FREE ATTRLIST(FB80)'
[12]  TSO← 'FREE ATTRLIST(FB121)'
[13]  TSO← 'FREE ATTRLIST(U121)'
[14]  TSO← 'ATTRIB  FB80  RECFM(F B) LRECL(80) BLKSIZE(800)'
[15]  TSO← 'ATTRIB FB121 RECFM(F B) LRECL(121) BLKSIZE(1210)'
[16]  TSO← 'ATTRIB U121  RECFM(U) LRECL(1210) BLKSIZE(1210)'
[17]  TSO← 'DELETE LISTDS.SYSIN'
[18]  TSO← 'FREE FI(SYSIN)'
[19]  TSO← 'ALLOC FI(SYSIN) DA(LISTDS.SYSIN) NEW SP(10,1) BL(800) USING(FB80)'
[20]  REC←80↑ ' LISTCTLG VOL=3330=USER01,NODE=' ,UID
[21]  REC←80↑ '/*'
[22]  Z←□SVR 'REC'
[23]  TSO← 'FREE FI(SYSPRINT)'
[24]  TSO← 'DELETE LISTDS.SYSPRINT'
[25]  TSO← 'ALLOC FI(SYSPRINT) NEW DA(LISTDS.SYSPRINT) SP(10 1) BL(1210) USING(F
      B121)'
[26]  TSO← 'ALLOC DA(''YCC1.VTOC.USER01'')'
[27]  TSO← 'IEHLIST'
[28]  TSO← 'FREE FI(SYSPRINT)'
[29]  REC← 'LISTING (370'
[30]  TSO← 'ALLOC FI(LISTING) DA(LISTDS.SYSPRINT) OLD USING(U121)'
[31]  Z←111 □SVO 'REC'
[32]  Z←REC
[33] L:X←REC
[34]  →RPT⌈⍳0=⍴X
[35]  Y←((⌊(⍴X)÷121),121)⍴X
[36]  DSN←DSN,[0]((Y[;1+⍳⍴UID]∧.=UID)⌿Y)[;(51+⍳9),1+⍳44]
[37]  →L
[38] RPT: 'ENTER ''VOLUMEΔREPORT'' OR ''LIBΔLIST'' '
     ∇


          LISTDS
ENTER YOUR TSO USERID: DUNBAR
ATTR-LIST-NAME FB80 NOT FOUND
ATTR-LIST-NAME FB121 NOT FOUND
UTILITY DATA SET NOT FREED, IS NOT ALLOCATED
ENTER 'VOLUMEΔREPORT' OR 'LIBΔLIST'
```

(32)

```
      ∇VOLUMEΔREPORT[□]∇
      ∇ VOLUMEΔREPORT;VOLS;N;M;J;I
[1]   VOLS← 0 ¯44 ↓DSN
[2]   N←2+⌊/+/' '=DSN
[3]   M←(1↑ρVOLS)ρI←0
[4]  L:J←VOLS∧.=VOLS[I;]
[5]   VOLS[I;]
[6]   3 COL((+/J),N)↑' ',(0,10+ρUID)↓J⌿DSN
[7]   M←J∨M
[8]   I←M⍳0
[9]   →L⌈⍳I<1↑ρVOLS
      ∇
      ∇COL[□]∇
      ∇ Z←C COL X;I;J
[1]   I←1↑ρX
[2]   I←⌈I÷C
[3]   J←¯1↑ρX
[4]   J←I,J
[5]   Z←(I,0)ρ' '
[6]  L:Z←Z,J↑X
[7]   X←(I,0)↓X
[8]   →(0≠1↑ρX)/L
      ∇
      VOLUMEΔREPORT
USER06
 αPL.BIO                  αPL01976.TSO             SAMPLE.COBOL
 αPL.LISTDS               αPL01976.TSOFNS          TEST.FORT
 αPL.MAGCARD              COBOL.SYSPRINT           X
 αPL.STACK                PRINTWS.CNTL
                          PRTHEX.CLIST
USER02
 αPL.BUG                  αPL01976.TYPEWS          FORT.APL
 αPL.TYPEWS               APLCOYCC.ASM             P
 αPL01976.TEST            COBOL.CNTL
USER01
 αPL.DUMPMSG              αPL.MSGLOG               αPL01976.QUADAV
 αPL.LISTDSX              αPL01976.BUGS            LISTDS.SYSPRINT
USER04
 αPL.FNS                  APLCNVT.JCL.DATA         OFFLINE.CNTL
 αPL.SMI                  AP101.CLIST              SYSPRINT
USER05
 αPL.LOADSCHD             αPL01976.TYPEFNS         αPL05100.TRANS
 αPL.PRINTMSG             αPL01976.TYPEVAR         LISTDS.SYSIN
 αPL.PRINTWS              αPL05100.LINK5100        TEST.OBJ
 αPL.TABLE                αPL05100.MOVE5100
 αPL01976.PRINTWS         αPL05100.RLNK5100
USER03
 αPL.TEST                 αPL01976.PFK             TEST.CLIST
 αPL.VAPLTEXT             αPL01984.NEWS            VSAPL.CLIST
 αPL01976.LISTDS          SAMPLE.FILE1
 αPL01976.NEWS            SYSDMP
STOR05
 APL.CLIST
UN6071
 BS.CLIST                 SMISAVE                  VSAPLOLD.APLSCINI
 COPY.CNTL                SYSUDUMP                 YALEUPD.CNTL
 SMI                      VSAPLNEW.APLSCINI        YALEUPD.PLI
```

```
      ∇LIB∆LIST[□]∇
      ∇ LIB∆LIST;PRIVATE;PUBLIC;PUBNUM;APL
[1]   APL←(0,10+⍴UID)↓('⍺'=DSN[;16])⌿DSN
[2]   PRIVATE←((J←'.'=APL[;3])⌿APL)[;4+⍳9]
[3]   PUBLIC←(~J)⌿APL
[4]   PUBNUM←⍎,' ',( 0 3 ↓PUBLIC)[;⍳5]
[5]   'PRIVATE APL LIBRARY'
[6]   ' ',6 COL PRIVATE
[7]  L: 'PRIVATE/SHAREABLE ' ,⍕1↑PUBNUM
[8]   ' ',6 COL((+/J),10)↑ 0 9 ↓(J←PUBNUM=1↑PUBNUM)⌿PUBLIC
[9]   PUBLIC←(J←~J)⌿PUBLIC
[10]  →L⌈⍳10≠⍴PUBNUM←J/PUBNUM
      ∇

      LIB∆LIST
PRIVATE APL LIBRARY
 BIO       FNS       LOADSCHD  PRINTMSG  STACK     TYPEWS
 BUG       LISTDS    MAGCARD   PRINTWS   TABLE     VAPLTEXT
 DUMPMSG   LISTDSX   MSGLOG    SMI       TEST

PRIVATE/SHAREABLE 1976
 BUGS      NEWS      PRINTWS   TEST      TSOFNS    TYPEVAR
 LISTDS    PFK       QUADAV    TSO       TYPEFNS   TYPEWS
PRIVATE/SHAREABLE 1984
 NEWS
PRIVATE/SHAREABLE 5100
 LINK5100  MOVE5100  RLNK5100  TRANS

      )OFF
CONNECT 2.176 HR
CPU .467 MIN
DUNBAR LOGGED OFF TSO AT 14:56:13 ON AUGUST 17, 1976+
```

This page intentionally left blank

(1) This sample terminal session was executed on an IBM Communicating Magnetic Card Selectric Typewriter. Since this terminal appears to TSO as a Correspondence code 2741, a special TSO logon command was used to select a translate table which would allow the entire TSO session to be conducted using the APL type element (Part Number 1167987). The terminal session was edited before publication to remove sensitive data.

(2) TSO requests the password for the userid specified in the)APLOGON command. The security mask printed on the terminal is a modification to TSO made by Yale University, and may not appear on all systems. After the password is entered, the logon procedure continues with the display of BROADCAST messages.

(3) The logon procedure is complete, and the READY message indicates that the user is now in contact with TSO. Any valid TSO command may be entered at this point.

(4) The first TSO command in this terminal session is LISTC, which requests TSO to display the cataloged OS datasets belonging to this user.

(5) The first set of datasets in the OS catalog are a list of workspaces belonging to the user's PRIVATE APL library. Only the owner may issue an APL)LOAD,)LIB,)SAVE,)DROP or)COPY for these workspaces. Non-APL access to these datasets conforms to the security constraints of the other TSO datasets. They may be copied with any OS utility (IEBGENER). If a user copies another user's private workspace into any other library (in effect, gives it another name), he will be able to find it in that library with the)LIB command, but will be unable to)LOAD or)COPY the workspace under APL. Only workspaces in PUBLIC or PRIVATE/SHAREABLE libraries may be exported to other users.

(6) The second grouping of datasets is the APL workspaces belonging to the user's PRIVATE/SHARED library number 1976. No other APL user may have a PRIVATE/SHARED library number of 1976. These workspaces may be accessed by any APL user through the)LOAD,)COPY, and)LIB commands, but they may only this user may issue the APL)SAVE or)DROP command for any of these workspaces. If any of these datasets have been password protected, the user must supply the password in the)LOAD or)COPY command, but does not require a password for)LIB.

(7) This user owns two additional PRIVATE/SHARED libraries, 1984 and 5100. If the user issues a)DROP for workspace '1984 NEWS', then ownership for PRIVATE/SHARED library 1984 is relinquished, and any other user who has authority to access that range of numbers, may gain control of 1984 as a PRIVATE/SHARED library by issuing a)SAVE in APL for a workspace.

(8) Following the PRIVATE/SHARED APL workspaces is the list of other TSO datasets. The user supresses the remaining listing by pressing the ATTN key. Output is shown to be suppressed on the final line by the TCAM transmission of the character ↓. The TSO READY message indicates additional user commands are required.

(9) The VSAPL command is typed nextwith no invocation parameters, to enter the APL environment from TSO. Since VSAPL has no reliable way for detecting what type of terminal is being used (other than 3270), the user is prompted to type a distinguishing character on his terminal. The character typed is always an upshift-2, or APL overbar character. On a 3767 terminal, you must shift the EBCDIC/APL keyboard switch to APL before typing the character.

This user has been authorized to use APL, indicated by the VSAPL message and a CLEAR APL workspace.

(10) The printing width for a 2741 terminal is 120 characters, by default. For the purposes of this session, it is reset to 65 so that data may be displayed on an 8 1/2 x 11 paper. Please note, that the printing width is a characteristic of the terminal *session*, and not of the *workspace*, as in previous implementations. VSAPL coordinates this printing width with TSO. If other workspaces are LOADed that have created their output for 120 character display, incorrect output will occur as each line is wrapped to the suceeding line.

(11) The first APL system command issued is a request to list the workspaces in the PRIVATE Library. These workspaces correspond to those listed by TSO at point number 5. The output of this request differs from all other implementations of APL. The workspaces are listed horizontally, in alphabetic order like the output of)FNS and)VARS in order to save paper, and to try to keep as much output as possible on a 3270 screen before screen overflow.

(12) At this point, the user attempts to share a variable with the TSO command processor 100. The APL command is rejected with an APL error message indicating that no Shared Variable Processors are active with this invocation. The caret indicates the point of error in the APL expression.

(13) To correct this situation, the user exits from APL with the)OFF HOLD command, to return to the TSO environment.

(14) When the TSO READY message is received, the user types the VSAPL command again, this time specifying the auxiliary processors he wishes to communicate with. The system defaults are accepted for SHARESPACE and FREESPACE. The TE (TERMCODE) command indicates to APL the terminal type being used, so that no prompt will be issued.

The user also indicates a workspace which he wants to be automatically LOADed: 1976 TSO.

This command also exists in a TSO dataset called 'APL.CLIST', which appeared in the LISTC output. An alternate TSO command could be: 'EX APL', which would Execute the command list. **Warning**: If you wish to use this form of APL invocation, you may only invoke the CLIST from the terminal type on which you created the character 'overbar', or a 3270. If you attempt to execute this CLIST from an EBCDIC 2741, for example, an error condition will exist, and you may have to call the system operator to cancel your TSO session.

(15) Upon entry to VSAPL, the workspace 1976 TSO is automatically LOADed, and the latent expression (DESCRIBE) is executed, to produce an informational message to the user. The latent expression could also have been an application function name. The user application program could then read and evaluate all user input for those users who are not familiar with the APL language.

(16) The APL system command)FNS was mis-typed at the terminal. The user backspaces to the point of error, and creates an illegal overstrike (an X over the invalid M character), and presses RETURN. The system responds by spacing to the incorrect error, typing an inverted cared, and waiting for input on the second line under the character in error. Logically, all characters from the typewriter position to the right have been deleted, so that the user may properly complete the command by typing the characters 'NS'.

If the user had pressed the ATTN key, TCAM would delete the entire input line, and require the user to retype it.

On a 3270, if the user had discovered his error before pressing the ENTER key, he could have used the cursor control keys to backspace, and replace the single incorrect character.

(17) The function TSO found in this workspace is used to issue the TSO command LISTBC which retrieves the broadcast messages. The message exceeds the 65 characters alloted for output, and TCAM splits the message into two lines. The 0 on the thirdline is the RETURN CODE from the LISTBC command, which was successful.

The TSO command is documented in Appendix A of this publication.

(18) The TSO command is again issued, but this time with a null command argument. The TSO function reads commands from the terminal, and passes them to TSO, just as if the user had typed them in the TSO environment. Two Yale commands are typed: NUSER retrieves the number of logged on users, and TOD displays the time-of-day. A null line (RETURN only) terminates the TSO function, and the user receives the RETURN CODE of the last TSO command executed.

(19) The workspace 1976 PFK is next loaded to demonstrate the use of program function keys on the IBM 3270 Display System Terminal. This workspace also has a latent expression which instructs the user. (On your system, this workspace is located in Public Library 1).

(20) This terminal entry demonstrates how to type comments on your terminal listing. The 'lamp' indicates that what follows is for illumination.

If the 3270 Terminal user presses PFK 1, an internal message is passed to APL which is PFK-underscore, and the PFK number 1. In the workspace PFK is a function which takes as its single argument, the program function key pressed. The entry does not appear on the screen. The functions in this workspace are set up to display an informational message to the user when PFK 1 is pressed.

(21) PFK's 7, 8, and 9 are set up to execute immediately the system commands)FNS,)VARS, and)SINL. The results are indicated. The function uses auxiliary processor 101 to STACK the commands for APL input.

(22) PFK's 11, 12, and 13 are set up as DELAY functions. They display a portion of the commands)SAVE,)LOAD, and)COPY on the screen, and wait until the user completes the command with a workspace identifer. After adding the workspace-identifer, the user presses ENTER, and the command is transmitted to VSAPL through auxiliary processor 101.

The protective-copy command fails for the DESCRIBE entity in the workspace 1976 TSO, because there is already a global in this workspace named DESCRIBE.

(23) The next entries demonstrate how the user may exercise the auxiliary processors in desk-calculator mode. The variable X is first initialized defaulting the processor name to TSO. The conversion option of APL was selected, since the terminal used is a typewriter terminal. 192 conversion should be used for a 3270.

The initialized variable is offered to processor 101 for sharing, and the coupling response is 2, indicating a valid share has taken place.

The user then assigns a value to the variable X, which is an APL command to be transmitted to APL the next time terminal input is requested.

(24) The response *IS* 1976 *PFK* does not appear at the left margin. When the APL statement *X*←')*WSID* ' had completed (ie., it transmitted the variable assignment to processor 101), the keyboard next opened for input with the APL prompt (6 spaces), and opened the keyboard for input. The system command)WSID was transmitted to APL, and the response is returned assuming the typewriter is at the left margin as the result of a carriage return. Notice that the system command)WSID is not displayed. If you wish the user to know that the response IS 1976 PFK is as a result of issuing the)WSID command from inside an APL function, you will have to display that yourself (eg., *X*←⎕←')*WSID* ')

The response from processor 101 from the last assignment of X is queried by displaying the variable X. Notice, that it does not have the last assigned character variable)*WSID*; it has the results of the Shared Variable communication with processor 101.

(25) If you have variables in your workspace whose values seem to disappear, you may find out if there are shared variable processors involved by typing the SVQ query command specifying a coupling of 2 and a null processor list. The response is the processor number that is active.

You may find out which variables in the workspace have outstanding shares by typing in the next command (extracted from the APL Language manual) which indicates that the variable X is being shared with a processor.

(26) More than one variable may be offered to a processor at one time. Here, the variables A, B, and C are offered to the TSO Command Processor 100. The three 2's in response indicate that all have been accepted.

Variable A is used to issue the TSO SEND command to echo a message back to the terminal. This is sometimes useful when you wish to test the character conversions that are taking place for the auxiliary processors.

Variable B is used to issue the TSO LIST command against a non-existent dataset. The DATA SET ABC NOT IN CATALOG message is generated by TSO, and is not subject to APL character translations. Unless you have a 3270, or have used a special APL logon command to TSO, this message will be in upper-case characters, which are APL operators, and generally unreadable.

(27) This command retracts all of the variables which are being shared with any auxiliary processor. This statement makes a handy RESET function when you are working with shared variables which are GLOBAL in the workspace. For instance, any attempt to go back to point 23 in this terminal session to assign an initial value to a variable X before sharing it with processor 101 would be met with frustration, because the initial value would be passed directly to processor 101, and then on to APL. The processor would then produce a *SYNTAX ERROR* because *'(APL'* is an un-executable APL statement.

A more difficult problem exists when the variable is being shared with processor 111 creating an output dataset. All inital assignments of the variable are simply written to the output dataset, and they seem to disappear, since the value in the variable will indicate the response of the auxiliary processor.

(28) This portion of the Sample Terminal Session demonstrates how the MSG function found in LIB 1 TSO may be used as a broadcast facility and tracking mechanism for team members working on a common project. The messages are always placed in the TSO MAIL facility for the user if he is not logged on, or not receiving messages. The sender may keep a log of errors, or announcements, and who he sent them to by SAVEing the workspace.

(29) This section demonstrates an exclusive that VSAPL FOR TSO has over other implementations. The secondary message facility is implemented so that the knowledgable user is not overwhelmed by diagnostic messages, and the confused user can obtain more information regarding errors that may occur in a TSO environment.

The first SAVE attempt fails to place a password on the MSGLOG workspace, because Password Protection has not been implemented at Yale. If Password Protection is available, the)SAVE command would have placed an OS Password on the sequential dataset which is the workspace. The workspace is saved with no password. The philosophy here is that an indication of no password is better than to use an internal password on an OS dataset which may be *stolen* by a knowledgable user through non-APL means.

The second SAVE command has two levels of secondary message chained. The third question-mark is transmitted to APL, and is reported as an APL language syntax error, because no messages were queued for the terminal.

(30) The final section of the terminal section demonstrates a sample application using the TSO Command Processor, and OS Sequential Dataset Processor. This APL code may not execute on your system due to any number of situations: The OS Utility IEHLIST may be unavailable to you as a TSO user; the output of the IEHLIST listing may not appear in the same columns; the IEHLIST utility may not be resident in your system, and therefore require FREESPACE in you TSO region to invoke it from APL.

What is important about this application is the way in which processor 100 is used from within APL functions to create a 'Super-CLIST' facility within the TSO environment. Under APL function control, any program that can be executed in a TSO foreground region, may be executed under control of an APL function, provided that the resources are available in the region for that program. (Usually storage space for the program and any dataset buffers it may use., This is controlled through the VSAPL FREESPACE option).

Specifically, any batch program that may create reports that can be directed to sequential disk datasets, DL/1 batch programs that access an IMS database and create sequential disk transfer files, or reports on disk; or any other non-standard access method not supported by auxiliary processor 111 (QSAM) may be invoked from APL functions, and the output read into an APL workspace for further manipulation.

The OS catalog, for example, is a non-standard OS dataset organization that is not generally available to application coders. IEHLIST is used to convert this non-standard data to a form that can be handled by APL (QSAM dataset).

One function is written to read the data into the APL workspace using auxiliary processor 111, and the data is then manipulated using APL functions only, without further need for auxiliary processors or OS datasets. This application principle is what makes APL so powerful for handling ad hoc queries and one-time reports. Data from standard application reports, for example, may be read into an APL workspace and plotted using the PLOT functions available in PUBLIC LIBRARY 1.

(31) The LISTDS function declares all of its variables to be shared as LOCAL. This means that when the function terminates, all of the variables will be retracted automatically.

Statement

[6] Tests for the existence of a global variable containing the userid. This function is meant to be placed in a public library with no *UID* variable.userid. On first invocation by a user, the function will prompt him for his userid. Thereafter, if he saves and reusesresues the workspace, the prompt will not be issued.

[10] Retrieves the return value from the inital offer to processor 111, but does not check it. More coding could be added to check the return value for 0.

[11-19] Initialize the attribute-lists (DCB attributes) for datasets which will be used.

[19] Allocates a NEW dataset to place the IEHLIST SYSIN control card into.

[20] The first assignment in the variable REC which was offered for sharing at statement 9 OPENs the dataset LISTDS.SYSIN. Even though the dataset was ALLOCATED in statement 19, it is not opened until the first assignment. If the statement at 20 had been an APL reference to variable X, the dataset would have been opened as INPUT.

[22] The variable retraction also CLOSEs the dataset LISTDS.SYSIN.

[23-26] Allocate the datasets required by the OS Utility IEHLIST.

[27] The system utility IEHLIST is executed as a TSO command. The return-code of IEHLIST has not been checked. More thorough checking could be done.

[28-30] Clean-up of dataset allocations following IEHLIST execution.

[31] Offer variable REC for sharing with processor 111 using default conversion (370).

[32] Read out the return code as the result of the share.

[33-37] The APL Read Loop for the SYSPRINT output listing from IEHLIST.

[33] - reads the record from disk. The Attribute lists for the dataset that created the dataset (IEHLIST) are Fixed Blocked Records, but the Attribute list for APL reading is Undefined Length records equal to block length. The APL program does its own deblocking at statement 35 in order to reduce the overhead of reading the dataset. Any size record could have been created, subject to FREESPACE constraints.

[34] Branches to statement RPT if a null entry is received from processor 111. This is presumed to be end of file, but it could also be an I/O error. To be sure, a record variable should have been shared with 111, and the return code checked.

[35] Deblock Undefined length 1210 records to an APL array of 121 columns, and rows equal to the number of logical records.

[36] Select out only those records which have the userid in the second position of the listing. An APL array is created with the VOLSER of the dataset in position 1-6 (from 51-59 of the listing), and the Datasetname in

positions 10-53.

[38] At this point we have an array of datasetnames in DSN. The user may now use this array of data to create any reports he wishes. The data required for reports now exists entirely within the APL workspace, and no external I/O needs to be performed (by the user) to create reports. This is where APL gets its power from; the ability to manipulate arrays of data (files) with a single APL statement, without external I/O requests.

(32) *VOLUMEΔREPORT* creates a list of user datasets by volume. If a DASD volume suddenly becomes unavailable for some reason, the user may go to this report to find the impact.

Statement

[1]This statement creates an array of just the volume serial numbers. It does so by dropping the last 44 positions of the DSN array, and thus it need not know how many rows (datasets) there are in DSN.

[2] Calculates the width of the largest dataset name.

[3] Creates a vector of 0's to keep track of the volumes processed from the list.

[4] Finds all volumes equal to the next unprocessed volume in the list (starts with 0).

[5] Displays the volume

[6] Creates a 3-up listing of the datasets with matching volume serials

[7] Masks those volumes from further processing

[8] Calculates the next unprocessed volume number

[9] Loops until all volumes have been processed.

This page intentionally left blank

APPENDIX A: DISTRIBUTED WORKSPACES

Certain predefined workspaces are distributed with VS APL.

Additional workspaces are distributed with the Yale IUP, VS APL FOR TSO.

These workspaces are very helpful if you're trying to learn APL, converting from some other IBM APL system, or intending to use auxiliary processors.

Figure 23 lists and summarizes the workspaces distributed with VS APL. Generally, distributed workspaces are found in public libraries 1 and 2, but it is up to your installation personnel to determine which public libraries will actually be used.

Workspace Name	Description
Library 1 workspaces:	
NEWS	Indicates general items of information about VS APL, posts bulletins and lists operating schedules.
CONVERT	Assists in workspace conversion from previous IBM APL systems to VS APL.
WSFNS	Provides comparable defined functions for the APL system variables ⎕*IO*, ⎕*PP*, ⎕*PW*, ⎕*CT* and ⎕*RL*, and for the APL system function ⎕*DL*.
APLCOURS	Illustrates the use of APL.
TYPEDRIL	Contains a timed typing test.
EXAMPLES	Contains various examples of APL programming that demonstrate good APL coding techniques.
PLOT	Prints graphs.
FORMAT	Contains functions designed to aid in the formatting of output.
TSO	Contains functions to issue messages to users and execute TSO commands.
PFK	Contains functions which demonstrate how application programs may use the 3270 Program Function Keys.
Library 2 workspace:	
APFNS	Contains functions that can be used with the VS APL auxiliary processors under TSO. For further details, refer to the section "Auxiliary Processing."

Figure 23. VS APL Distributed Workspaces

Like other public library workspaces, the workspaces in Figure 23 can be retrieved at will. For instance, the distributed workspace *NEWS* can be activated by entering:

```
      )LOAD 1 NEWS
SAVED 11:15:27 01/15/76
```

Similarly, if you are only interested in a particular function in one of the distributed workspaces, you can copy it into your active workspace:

```
      )COPY 1 NEWS SCHEDULE
SAVED 11:15:27 01/15/76
```

All of the workspaces distributed with VS APL are self-documenting. Specifically, each workspace contains a group named *DESCGP*, which can be used like a computer-stored textbook to describe the contents and use of the workspace. *DESCGP* includes the following variables:

Because distributed workspaces are self-documenting, no further discussion of their contents and use are provided here. You should reference *ABSTRACT*, *DESCRIBE* and *HOW* as appropriate for the information you need.

Variable Name	Contents
ABSTRACT	Gives the purpose of the workspace in one or two sentences.
DESCRIBE	Gives the purpose of the workspace in detail. It also lists and details the names, syntax, and argument requirements of functions in the workspace.
HOW	Describes how the workspace is used. It also details the use of functions in the workspace.

LIB 1 TSO Public Workspace

```
      DESCRIBE∆TSO

THE FUNCTIONS TSO AND MSG EXERCISE THE AUXILIARY PROCESSOR 100

TSO    - TAKES A COMMAND AS ITS ONLY ARGUMENT AND SUBMITS IT TO TSO
         - A NULL ARGUMENT WILL READ AND EXECUTE TSO COMMANDS UNTIL
           A NULL LINE IS READ FROM THE TERMINAL

MSG    -  TAKES A LIST OF USERS (ON THE LEFT) AND A LIST OF MESSAGES
          (ON THE RIGHT) AND TRANSMITTS MESSAGES TO THOSE USERS,
          WHETHER OR NOT THEY ARE LOGGED ONTO TSO AND/OR RECEIVING
          MESSAGES.
          - A NULL ARGUMENT FOR MESSAGE WILL PROMPT THE USER FOR
            THE MESSAGE FROM THE TERMINAL.
```

```
      ∇ Z←TSO X;Y;⎕IO
[1]    ⎕IO←0
[2]    Z←100 ⎕SVO 'Y'
[3]    →CMD⌈⍳0≠1↑⍴X
[4]   LOOP:→0⌈⍳0=⍴X←⍞
[5]    Y←X
[6]    Z←Y
[7]    →LOOP
[8]   CMD:Y←X
[9]    Z←Y
       ∇
```

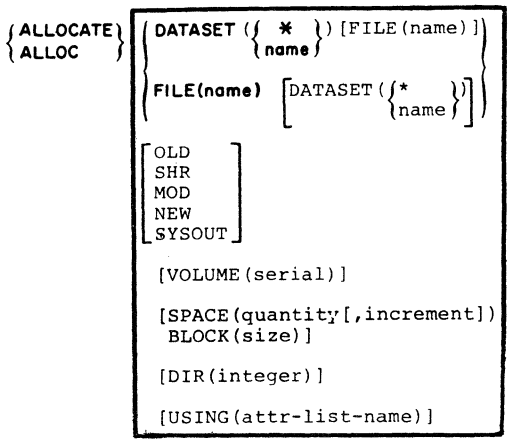
```
      ∇ Z←USER MSG TEXT;TSO;⎕IO;U;T;X;B
[1]   ⍝ SENDS MULTI-LINE MESSAGES TO 1 OR MORE USERS
[2]   ⍝ -- NULL MESSAGE PROMPTS FOR MESSAGE UNTIL NULL LINE
[3]    ⎕IO←0
[4]    B← 5 6 2 7 8 2 3 4 0 0 9 10 1 11 12 1 13 14 0
[5]    B←( ' :/' , 1 0 ⍕(14⍴10)⊤100⊥100|⎕TS)[B]
[6]    Z←100 ⎕SVO 'TSO'
[7]    TEXT←(¯2↑ 0 1 ,⍴TEXT)⍴TEXT
[8]    →IN⌈⍳~∧/×⍴TEXT
[9]   SC:X←(⍳⍴T),('''='=T)/⍳⍴T←TEXT⌊0;]
[10]   U←(¯2↑1,⍴USER)⍴USER
[11]  SE:TSO← 'SE ''' ,B,T←T[X[⍋X]], ''' U(' ,U[0;], ') L'
[12]   →⎕LC+ 2 1 [×⎕NC 'MSG∆LOG' ]
[13]   MSG∆LOG←MSG∆LOG,⎕TC[1],B,¯3↓T
[14]   Z←TSO
[15]   →SE⌈⍳0≠1↑⍴U← 1 0 ↓U
[16]   →(SC,0)[0=1↑⍴TEXT← 1 0 ↓TEXT]
[17]  IN:TEXT←(1,⍴TEXT)⍴TEXT←⍞
[18]   →SC
       ∇
```

TSO Command Summary

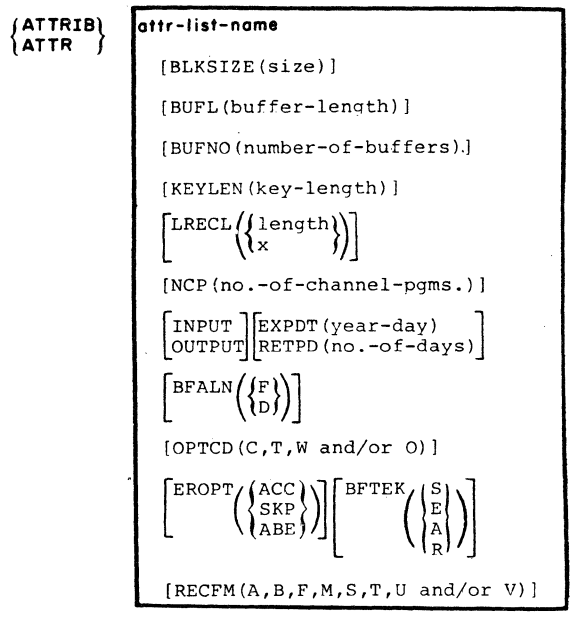
The following information was extracted from *OS/VS2 Command Language Reference Summary* (GX28-0647-0). The commands selected are the ones most often referenced in an APL environment. The TSO Edit command is displayed on a separate page following the alphabetical list of commands.

Caution: Not all commands are listed. Some commands in your installation may differ from the information listed here due to Operating System differences and modifications made by your installation. If an error occurs, display the command syntax for your installation by typing the TSO HELP command. If errors persist, see your Systems Administrator.

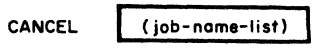
ALLOCATE - define and allocate a new or old data set.



ATTRIB - build an attribute list for data sets you intend to allocate dynamically.



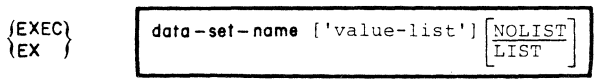
CANCEL - cancel a job or jobs previously submitted for background execution.



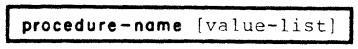
DELETE - delete and uncatalog one or more data sets or members.

EXEC - execute a command procedure.

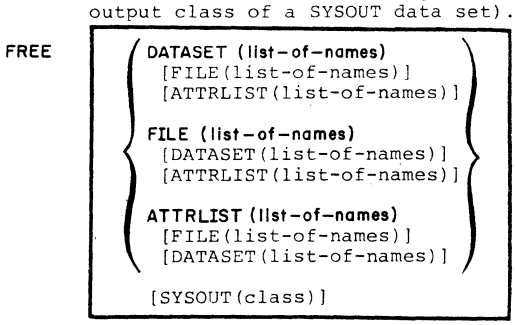
(Explicit form)



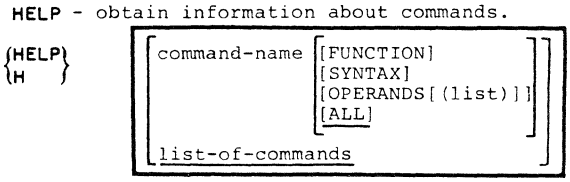
(Implict form, no Command name needed.)



FREE - release a previously allocated data set, file, or attr-list (or change the output class of a SYSOUT data set).



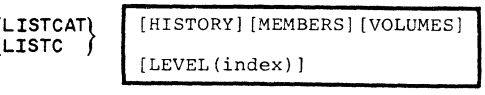
HELP - obtain information about commands.



LISTALC - list the names and characteristics of allocated data sets.

LISTBC - list messages from other users or system notices.

LISTCAT - list the names and characteristics of data sets cataloged under your userid.



LISTDS - display the attributes of specified data sets.

{LISTDS / LISTD}

```
(data-set-list)    [STATUS] [HISTORY]
[MEMBERS] [LABEL]
```

LOGOFF - end a terminal session.

LOGOFF

LOGON - start a terminal session.

LOGON

```
userid [/password]
[ACCT(account)] [PROC(procedure)]
[SIZE(integer)] [NOTICES / NONOTICES] [MAIL / NOMAIL]
```

OUTPUT- direct output (SYSOUT data sets and system messages) from submitted jobs to the terminal or a specified data set.

{OUTPUT / OUT}

```
(job-name-list)
[CLASS(class-name-list)]
[PRINT [* / data-set-name] [NEXT / HERE / BEGIN] [PAUSE / NOPAUSE]
 NOPRINT [(class-name)]]
```

CONTINUE - resume output operations that have been interrupted.

CONTINUE

```
[NEXT / HERE / BEGIN] [PAUSE / NOPAUSE]
```

END - terminate OUTPUT command processing

END

HELP - obtain information about OUTPUT subcommands.

{HELP / H}

```
[subcommand-name [FUNCTION]
                 [SYNTAX]
                 [OPERANDS[(list)]]
                 [ALL]
 list-of-subcommands]
```

SAVE - rename and catalog SYSOUT data set.

{SAVE / S} **data-set-name**

PROFILE - establish a user profile.

{PROFILE / PROF}

```
[CHAR({character / BS}) / NOCHAR] [LINE({ATTN / character / CTLX}) / NOLINE]
[PROMPT / NOPROMPT] [INTERCOM / NOINTERCOM]
[PAUSE / NOPAUSE] [MSGID / NOMSGID]
```

PROTECT - assign, delete, or modify a password for a data set.

{PROTECT / PROT} **data-set-name**

```
[ADD(password2)
 REPLACE(password1 password2)
 DELETE(password1)
 LIST(password1)]
[PWREAD / NOPWREAD] [PWWRITE / NOWRITE]
[DATA('string')]
```

RENAME - change the name of a data set or assign an alias.

{RENAME / REN}

```
old-name  new-name  [ALIAS]
```

SEND - send a message to the system operator or another user.

{SEND / SE}

```
'text' [USER(userid...)                [NOW / LOGON]
        OPERATOR[(integer)]]
```

STATUS - display status information at the terminal for a job or jobs entered through the SUBMIT command.

{STATUS / ST}

```
[(jobname-list)]
```

SUBMIT - submit a data set containing job control language for one or more jobs for interpretation and execution in the background.

{SUBMIT / SUB}

```
(data-set-list)    [NOTIFY / NONOTIFY]
```

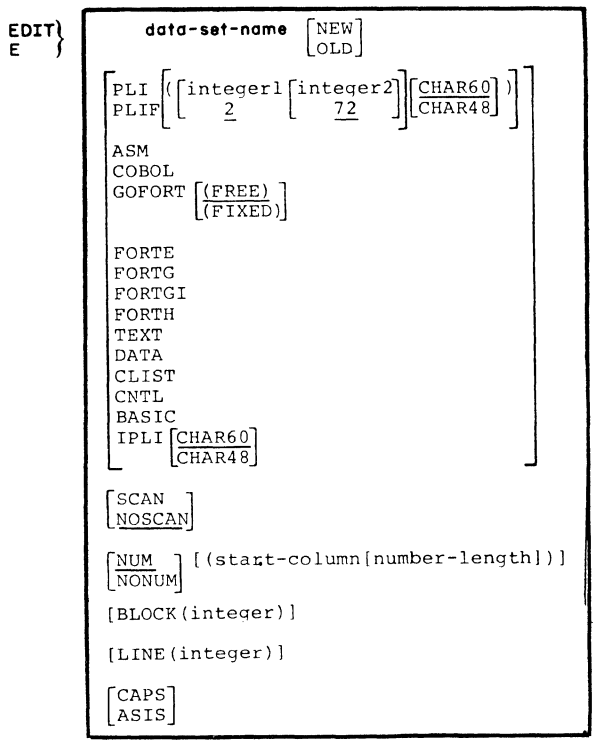
TERMINAL - define terminal operating characteristics.

{TERMINAL / TERM}

```
[LINES (integer) / NOLINES] [SECONDS (integer) / NOSECONDS] [INPUT (string) / NOINPUT]
[BREAK / NOBREAK] [TIMEOUT / NOTIMEOUT] [LINESIZE(integer)]
[CLEAR(string) / NOCLEAR] [SCRSIZE(rows,length)]
```

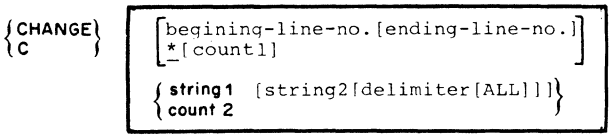
NOTES:

EDIT - create or modify a data set.

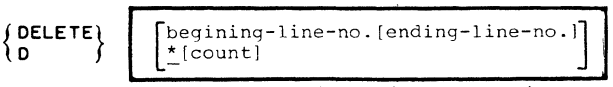


BOTTOM - set the current line pointer to point to the end of the data set.

CHANGE - alter the contents of a data set.



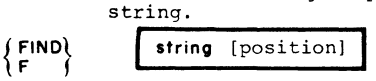
DELETE - delete specified portions of a data set.



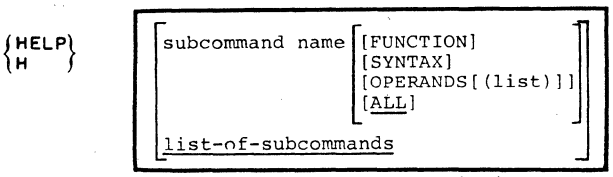
DOWN - set the current line pointer to point closer to the end of a data set.

END - stop operation of the EDIT command.

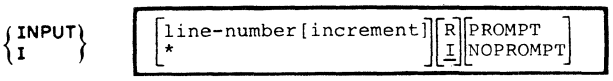
FIND - set the current line pointer to point to a line containing a specified character string.



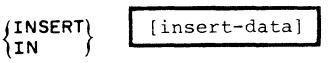
HELP - obtain information about **EDIT** subcommands.



INPUT - add or replace data in an existing data set.

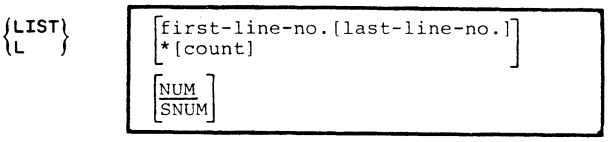


INSERT - place one or more new lines of data into an existing data set.

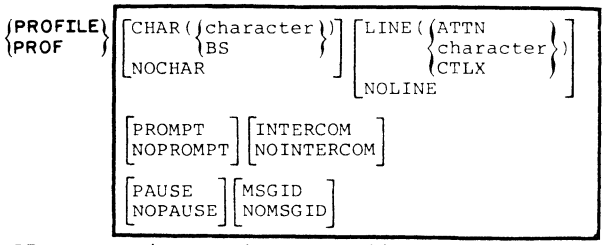


insert/replace/delete functions of EDIT - do not need a subcommand name. Specify either a line number or * (current line pointer).

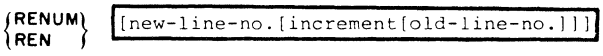
LIST - display one or more lines of a data set.



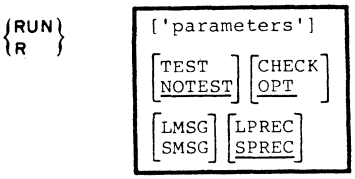
PROFILE - specify character deletion and/or line deletion characters.



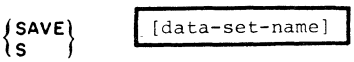
RENUM - assign or change the line numbers of an existing data set.



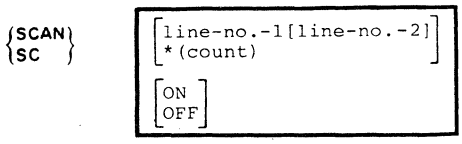
RUN - compile, load, and execute source statements in the data set. Used with program products.



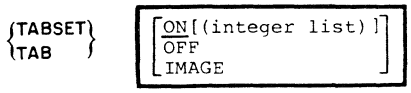
SAVE - retain an edited data set as a permanent data set.



SCAN - request syntax checking for source statements. Used with program products.

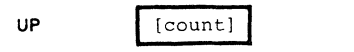


TABSET - establish or change tab settings.



TOP - set value of current line pointer to zero.

UP - set the current line pointer to point closer to the beginning of the data set.



VERIFY - display the new line whenever the value of the current line pointer changes.

LIB 1 PFK
Public
Workspace

```
THE FUNCTIONS IN THIS WORKSPACE ARE DESIGNED TO
DEMONSTRATE THE PFK FUNCTION KEYS OF THE 3270
  PFK1      - VARIABLE DISPLAY OF INFORMATION
  PFK2      - VARIABLE DISPLAY OF INFORMATION
  PFK3      - VARIABLE
  PFK4      - VARIABLE
  PFK5      - FUNCTION PFK5
  PFK6      - FUNCTION PFK6
  PFK7      - EXECUTES )FNS
  PFK8      - EXECUTES )VARS
  PFK9      - EXECUTES )SINL
  PFK10     - LEAVES )SAVE ON THE SCREEN
              - HIT ENTER, OR TYPE WSID AND ENTER
  PFK11     - LEAVES )LOAD ON THE SCREEN
              -FILL IN WSID AND HIT ENTER
  PFK12     - LEAVES )PCOPY ON THE SCREEN
              -FILL IN WSID AND ELEMENTS AND HIT ENTER

YOU MAY REPLACE ANY OF THE FUNCTIONS OR VARIABLES
LABELED PFK TO INVOKE YOUR OWN FUNCTIONS WITHOUT
MODIFYING THE MASTER FUNCTION PFK
OR YOU MAY MODIFY FUNCTION PFK TO PROVIDE
YOUR OWN SERVICE FUNCTIONS.
```

```
      ∇ PFK N;X;C;S
[1]   S← 'TSO (FIFO 192'
[2]   C←101 ⎕SVO 'S'
[3]   X←( 12 5 ↑ 9 5 ⍴ 'PFK1 PFK2 PFK3 PFK4 PFK5 PFK6 )FNS )VARS)SINL' )
[4]   →(N≥10)/DELAY
[5]   ( 0 ¯11 )[1+')'=1↑X]↑X
[6]   S←X
[7]   →0
[8]  DELAY:X←¯12↑( 4 7 ⍴ ')SAVE  )LOAD  )PCOPY  ''ERROR''' )[ 10 11 12 ⍳
[9]   ⍞←X
[10]  S←X,(⍴X)↓⍞
      ∇
```

APPENDIX B: WORKSPACE CONVERSION

This appendix introduces the VS APL Conversion Program and describes the report generated by the program.

The VS APL Conversion Program

The VS APL Conversion Program converts APL\360, APLSV and APL/CMS workspaces to VS APL workspaces. The conversion program is designed to be executed by the personnel responsible for installing your system. Once the program has been run, you'll be given a conversion report which lists those items in each workspace which must be evaluated or modified to ensure proper execution of the workspace under VS APL.

Pre-Conversion Considerations

When a workspace is submitted for conversion, the conversion program uses the library number and workspace name of the input workspace to form a workspace identification for the converted workspace. Not all library numbers and workspace names are acceptable to VS APL. Specifically, the following are unacceptable:

- A library number greater than seven digits
- A workspace name containing the symbol delta (Δ) or underscored characters

These unacceptable items should be modified before the conversion program is run.

Types of Conversions

There are two types of conversion that the VS APL Conversion Program performs: format and content.

In format conversion, all global objects (variables, functions, groups) and their names are converted into VS APL internal format.

In content conversion, all global objects are converted into VS APL internal format and all function statements are examined for items which require conversion in order to execute properly under VS APL. Such items are converted to their VS APL equivalent, if possible, and noted on the conversion report. Items with no VS APL equivalent are just reported.

The Conversion Report

The conversion report lists exceptional conditions found during format or content conversion. When format conversion is done, the report produced contains any exceptions found in workspace parameters or variables and any conversion errors. When content conversion is performed, the report additionally contains any exceptional functions. Figure 24 illustrates a sample conversion report.

```
LIBRARY        897574          CONVERSION SUMMARY REPORT
WORKSPACE: VIP

WS PARAMETER:
      PRINT PRECISION          5
      RANDOM LINK              1097971256
      COMPARISON TOLERANCE 1.136729599338082E-13

FUNCTION:HAM
*     SPECIFICATION            5    8    11   13

FUNCTION:DELAY
      REPLACED

FUNCTION:BET
*     TRANSPOSE                1

FUNCTION:KEYWORD
*     RESIDUE                  3
**    DYADIC IBEAM             1    2    3

FUNCTION:HAD
      (LOCKED)

FUNCTION:TIME
*     MIXED OUTPUT             1
*     ENCODE                   1
*     MONADIC IBEAM            1

FUNCTION:IBE
      IBEAM SIMULATOR FUNCTION ADDED TO WORKSPACE

**    WARNING. WILL NOT EXECUTE UNDER VS APL
*     CAUTION. MAY NOT EXECUTE AS INTENDED UNDER VS APL
```

Figure 24. Sample Conversion Report

Workspace Parameters Reported

For each workspace to be converted, the name and value of the following workspace parameters are reported if the value in the input workspace is other than the VS APL default:

- Comparison tolerance
- Index origin
- Printing precision
- Random link
- Symbol table size

Variables Reported

If a variable is invalid, its name and one of the following exceptions are reported. Invalid variables are deleted.

** REJECTED, INVALID DIMENSIONS(S)

The element of a dimension is negative or exceeds the VS APL maximum, or the element count exceeds the VS APL maximum.

** REJECTED, INVALID RANK

The rank is less than zero or exceeds the VS APL maximum.

Functions Reported

Functions are examined and exceptional conditions are reported only when content conversion is done. For each function, the function name and a list of exceptions are reported. The format of each item in the list is:

flag exception line-number(s)

The line numbers are those in the function in which the exception occurs. The flag indicates the level of severity:

Flag	Meaning	Description
blank	Information	The item has been converted to its VS APL equivalent and will execute properly under VS APL.
*	Caution	The item may not execute as intended under VS APL.
**	Warning	The item will not execute under VS APL.

The exceptions are as follows. The information in parentheses indicates the source APL systems for which the exception should be evaluated and corrected where needed. The information in parentheses does not appear in the conversion report.

** DYADIC IBEAM (APL\360, APL/CMS, APLSV)

There is no VS APL equivalent for dyadic Ibeams. They are left unchanged. Execution results in a syntax error.

** LINE TOO LONG (APL\360, APL/CMS, APLSV)

If conversion causes a line to be expanded to more than 4093 bytes, the line is deleted and replaced with:

"THIS FUNCTION LINE WAS TOO LONG AND WAS DELETED BY CONVERSION" *item*

The term *item* will cause a syntax error in the converted statement.

** SYSTEM VARIABLE (APL/CMS, APLSV)

There is no VS APL equivalent for system variables $\square TT$ and $\square UL$. They are left unchanged. Execution results in a syntax error.

** UNCONVERTABLE (APL\360, APL/CMS, APLSV)

A function which localizes $\square TT$ or $\square UL$, or a function whose header contains a syntax error cannot be converted. Such functions are deleted.

* AMBIGUOUS IBEAM (APL\360, APL/CMS, APLSV)

An ambiguous Ibeam is assumed to be monadic converted as described in "Monadic Ibeam" below.

* ELIDED SEMICOLS (APL\360, APLSV)

When mixed output is converted, contiguous semicolons are discarded (see "Mixed Output" below).

* ENCODE (APL\360)

The definition of encode for a left argument having one or more negative elements is different from the definition used in APL/360. Execution might have a result different from the one expected.

* INCOMPLETE LIST (APL\360, APL/CMS, APLSV)

The function contains more exceptions than the conversion program can record. Additional exceptions exist, but are not reported.

* MIXED OUTPUT (APL\360, APL/CMS, APLSV)

Mixed output is converted to an equivalent expression that uses format primitive function. Execution in rare cases may cause a rank or length error.

* QUAD AV (APL/CMS, APLSV)

The correspondence of particular symbols to elements of $\square AV$ is implementation dependent. Reference to $\square AV$ might have a different result from the one expected.

* RESIDUE (APL\360)

The definition of residue for a left argument having one or more negative elements is different from the definition used in APL\360. Dyadic and ambiguous residue are reported. Execution might have a different result from the one expected.

* SPECIFICATION (APL\360, APLSV)

Conformability requirements for indexed specification are more restrictive for VS APL than for APL\360 or APLSV. All occurrences of indexed specification are reported. Execution might result in a rank or length error.

* TRANSPOSE (APL\360)

The definition of monadic transpose for an argument of rank greater than two is different from the definition used in APL\360. Monadic and ambiguous transpose are reported. Execution might result in a different result than expected.

CARRIER RETURN (APL\360, APL/CMS, APLSV)

A character array that contains carrier returns is converted to an equivalent expression that uses the second element of the system variable $\square TC$.

IBEAM SIMULATOR FUNCTION ADDED TO WORKSPACE (APL\360, APL/CMS, APLSV)

See "Monadic Ibeam" below.

MONADIC IBEAM (APL\360, APL/CMS, APLSV)

A monadic or ambiguous Ibeam is replaced with a call to a monadic function; the argument of the function is that of the Ibeam. The function simulates all monadic Ibeams except I23 and I28, which have no equivalents in VS APL. Execution of the function with such arguments results in an error. The simulator function that is added to the workspace has a unique name; the name is IBE unless that name already exists in the workspace, in which case it is IBF. The function is locked so it will behave as a primitive function.

REPLACED (APL\360, APLSV)

A WSFNS function (DELAY, DIGITS, ORIGIN, SETFUZZ, SETLINK, WIDTH) is replaced if the function is locked and its definition is exactly the same as that in APL\360 distributed library 1. The new function is unlocked; it performs the equivalent of the replaced function using the appropriate system variable.

(LOCKED)

The function is locked. Content conversion has been done, but exceptions are not reported. The flag indicates the highest severity level.

Conversion Errors Reported

The following error messages are displayed in the VS APL Conversion Report and should be reported to the personnel responsible for installing your system:

WORKSPACE FULL

The available space in the workspace has been exhausted. As a result, the workspace is not completely converted. Conversion might be possible if the maximum workspace size defined under TSO is increased.

*THIS LIBRARY DEFINED WITH DEFAULTS DUE TO SYSTEM ERROR

A system error has been encountered during conversion. In response, certain items in the workspace have been respecified using default values. The items for which default values are used are maximum library size, maximum and default workspace size, maximum shared variable size, and maximum number of shared variables.

SYSTEM ERROR WHILE CONVERTING WORKSPACE/DIRECTORY *number name* WORKSPACE/DIRECTORY UNCONVERTED AND PRESUMED DAMAGED

A system error has been encountered during conversion. This message is followed by a printout of the time of day and the contents of the program check old PSW and general registers.

ERROR UNRECOVERABLE. CONVERSION ABORTED

An unrecoverable error has been encountered during conversion. As a result, conversion is terminated. This message is followed by a storage dump.

***WSID INVALID FOR VSPC WORKSPACE (OR LIBRARY) REJECTED

The library number of the workspace submitted for conversion is greater than seven digits or the workspace name contains invalid characters.

*** WORKSPACE REJECTED, NOT CONVERTED, DUE TO I/O ERROR

An input/output error has been encountered during conversion. As a result, the workspace is not converted.

*** HDR LABEL I/O ERROR. CONVERSION CANCELLED

An internal label error has terminated conversion.

*** TRLR LABEL I/O ERROR. CONVERSION CANCELLED

An internal label error has terminated conversion.

WRITE ERROR nn WHILE SAVING libnum name. CONVERSION CANCELLED.

NOT A CMS DUMP TAPE. CONVERSION CANCELLED.

filename filetype filemode NOT AN APL/CMS (PRPQ) WORKSPACE.

INADEQUATE SPACE TO RUN CONVERSION. CONVERSION CANCELLED.

filename filetype WORKSPACE TOO LARGE FOR VIRTUAL MACHINE SIZE. RERUN WITH LARGER MACHINE. WSSIZE IS xxxx NEED ABOUT yyyy BYTES TO CONVERT.

filename filetype WORKSPACE DAMAGED OR INVALID. NOT CONVERTED.

Unreported Items

The following items may cause a converted workspace to execute differently under VS APL that it did under the source APL system. They are not reported by the conversion program.

Imbedded respecification may cause a result different than expected. In VS APL, the following statement yields a result of 15.

```
(A←3)×A←5
```

For APL/CMS and APLSV workspaces, system variables are given the corresponding workspace parameter value. The value of a system variable with an implicit error is not retained. The value of $\square LX$ is not retained; it is given the default value of null.

For APL/CMS workspaces, an underscore character _ in the name of an object is converted to a delta (Δ) character. If the underscored character appears in a character array, the character is transmitted as is.

All APL/TSO workspaces have a datasetname of the following format:

userid.@pl.wsid

See your system administrator.

APPENDIX C: LANGUAGE CONSIDERATIONS

This appendix covers some special language considerations for VS APL.

Outgoing Offer Query

VS APL supports the dyadic form of the system function $\square SVQ$. This function gives you the ability to list shared variable offers made by a given processor.

The dyadic $\square SVQ$ function is specified as follows:

```
a  □SVQ  b
```

where *a* represents a degree of coupling of 1 or 2 and *b* represents one or more processor identifications.

A specification of *a* $\square SVQ$ *b* asks the system to display a list of variable names offered by processor *b* having a degree of coupling of *a*.

For example, suppose you wanted to determine which variables are being shared with a processor whose identification is 1234, the expression you would issue in that case would be:

```
2 □SVQ 1234
```

In response, the system returns a character matrix of shared variable names having a degree of coupling of 2. The surrogate names of the shared variables are also included if they differ from the shared variable name. For the expression:

```
2 □SVQ 1234
```

a typical response might be:

```
ALPHA
APAR
```

If you specify a right argument that is empty, then the result is a list of processors. Each processor identified is a share partner of a shared variable having the degree of coupling specified by the left argument.

System Variables TT, UL, HT, and TC

VS APL does not support the terminal type ($\square TT$) or user load ($\square UL$) system variables. VS APL includes two new system variables: tab settings ($\square HT$) and terminal control characters ($\square TC$). The system variables $\square HT$ and $\square TC$ are described in the section "Terminals."

Duplicate Names in a Defined Function

In VS APL, the name of a defined function, local names indicated in the function header, and labels contained within the body of the function are established in the following order:

1. Function name.
2. Other names in the function header taken from left to right.
3. Labels in the function body taken from top to bottom.

If duplicate names appear, then all but the first occurrence of each name taken in the above sequence are ignored.

Line Deletion in a Function Definition

During function definition, any entered statement *N* can be deleted by typing [Δ*N*], where *N* is a single statement number. After deleting the statement, the system awaits entry of statement *N*.

For example, the following illustrates the deletion of statement 1 in the definition of *FUNC*:

```
      ∇FUNC
[1] ALPHA←2 3⍴⍳6
[2] BETA←⍞
[3] [Δ1]
[1]
```

A bracketed statement number immediately followed by both an attention and a carrier return has no effect on the definition of the function, that is the same bracketed statement number is returned.

Interrupting Input

VS APL permits you to interrupt requests for both evaluated (⍞) and character (⍠) input during the execution of a defined function by entering the sequence *O* backspace *U* backspace *T*.

As certain terminals supported by TSO do not have a backspace character, for example the IBM 3270, this method of interrupting input requests cannot be used in all cases. For further information, see "Appendix D: TSO Terminals."

Indexing on the Left of an Assignment

If *A* and *X* are arrays and *P* is a subscript list, then an expression of the form *A*[*P*]←*X* is executable only if (1) *X* is a scalar or one-element array, or (2) the dimension vector of *X* is the same as the catenation of the dimension vectors of the list elements of *P*. So that if an array *A* were defined as follows:

```
      A
1 2 3 4
5 6 7 8
```

the following expressions are executable:

```
      A[1;3]←9
      A
1 2 9 4
5 6 7 8

      A[2;]←8 7 6 5
      A
1 2 9 4
8 7 6 5
```

However, the following expression will produce an error report because it doesn't meet requirement (1) or (2):

```
      A[;1]←2 1 4
LENGTH ERROR
```

Some *RANK ERRORS* are more subtle, but common in converting from other APL implementations, or in commercial APL programming. Using the matrix *A* defined above, the following 2 examples will each produce a *RANK ERROR* report:

```
A[,2;]←8 7 6 5

A[2;]←1 4⍴8 7 6 5
```

A vector subscript list produces an array; a scalar subscript produces a vector. Reversing the assignments in the above examples, will cause them to execute correctly. An alternative way, which may be applied no matter how complex the expression is that produces the index, is to convert the subscript to the appropriate form to accept the data being assigned.

```
A[''⍴,2;]←8 7 6 5

A[,2;]←1 4⍴8 7 6 5
```

Null-restructure of a variable produces a scalar; ravel of a scalar produces a vector.

This page intentionally left blank

APPENDIX D: TSO TERMINALS

This appendix summarizes the sign-on/sign-off procedures and various operating characteristics for all the terminals supported by TSO and discusses a number of items to be considered when using APL at IBM 3270 terminals. Terminal operating characteristics are summarized in Figure 25. For further information concerning each of the terminal supported by TSO, refer to the *TSO: Terminal User's Guide.*

Action	TERMINALS				CPT-TWX Mod 33/35
	3767	2741	3270	1050	
Set switches for computer connection	Keylock=ON COMM/LOCAL=COMM AUTO/OFF=AUTO EDIT/OFF=OFF AUTO VIEW/OFF=as desired DOUBLE/SINGLE SPACE=as desired DATA/TALK=DATA DIAL DISC/OFF=OFF SDLC/SS=SS EBCDIC (or Corr.)/APL=EBCDIC (or Corr.) CALC/OFF=OFF TEST/OFF=OFF POWER/OFF=POWER	COM/LCL=COM ON/OFF=ON	Pull out OFF/PUSH	MAIN-LINE=ON SYSTEM=ATTEND MASTER=ON PRINTER 1=SEND/REC PRINTER 2=HOME KEYBOARD=SEND PUNCH=NORMAL SYSTEM=PROGRAM EOB=AUTO SYSTEM=UP All others=OFF	Press ORIG button Model 35: Press K button On high-pitched tone, press CTRL, WRU
Terminal ready to accept logon	DATA SET READY, PROCEED lights on	Keyboard unlocks	SYSTEM AVAILABLE Cursor appears	POWER, PROCEED lights on	Key mode: paper advance optional ? printed
Logon	logon userid m	logon userid m	logon userid m	logon userid m	logon userid m
Enter a line	↵ (return)	RETURN	ENTER	RETURN ALTN CODING + EOT	RETURN
Terminal ready to accept input	ON LINE and PROCEED lights on	Unlocks keyboard	Cursor appears	Unlocks keyboard PROCEED light on	Paper advance Line number Stops noise Bell
Correct character in current APL line	← (backspace), retype, ↵(return)	BKSP, retype, RETURN	Position cursor, type correct character	BACKSPACE, retype, RETURN	Press CTRL and H keys, retype, RETURN
Send signal to system	ATTN	ATTN	PA1	ATTENTION	BREAK
Interrupt input	O←U←T	O BKSP U BKSP T	PA2* ENTER	O BACKSPACE U BACKSPACE T	O CTRL H U CTRL H T
Cancel output	ATTN	ATTN	PA2*	ATTENTION	BREAK
Sign off	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE
Turn off terminal	POWER=OFF Keylock=OFF	ON/OFF=OFF	Push OFF/PUSH	MAIN-LINE=OFF	CLR
Default printing width (□PW)	132	130	80	130	72

Figure 25. TSO Terminal Summary

IBM 3270 Display System Terminal Considerations for APL

The IBM 3270 terminals that can be used with VS APL are the IBM 3275 or the IBM 3277. The IBM 3277 may be installed with an APL feature.

If the IBM 3270 is operated without an APL feature, existing workspaces can be loaded and names which do not contain underlined alphabetic characters can be entered and functions executed. There is no backspace character on the terminal, so that any of the compound characters cannot be entered (though they may be displayed as a 3 character representation, in general as upper case alphabetics).

The IBM 3270 APL Feature

The IBM 3270 APL feature makes it possible to enter and display the full EBCDIC and APL character set from the keyboard illustrated in Figure 26.

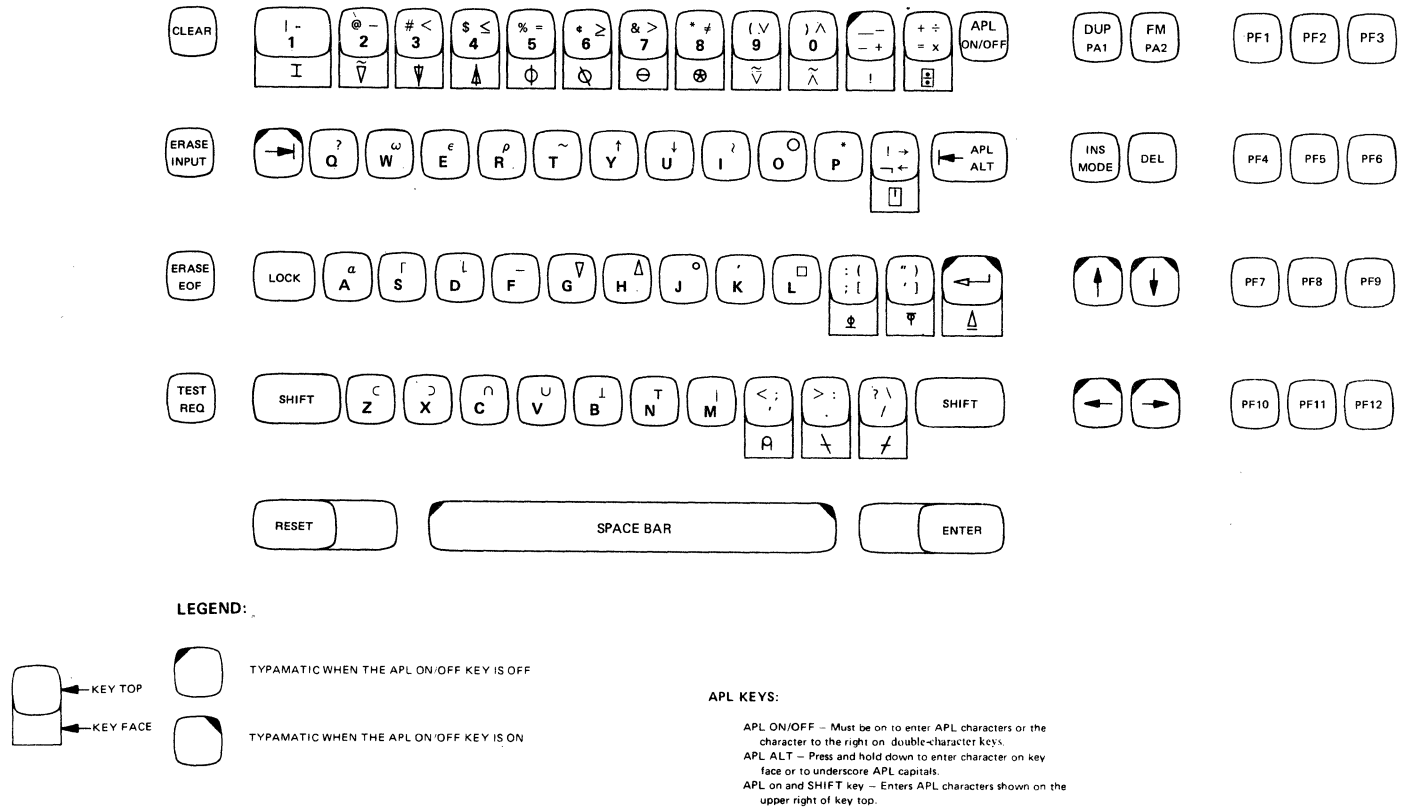


Figure 26. The IBM 3270 APL Keyboard

When power is initially turned on, the keyboard acts as a regular 3270 keyboard. Pressing the APL ON/OFF key once, turns on the APL character set. All the APL graphic characters can be produced by pressing the appropriate key while compound characters can be produced by pressing the APL ALT key and the key that has the appropriate compound character on its front. Underscored alphabetic characters are produced by pressing the APL ALT key and the alphabetic character that is underscored. The APL ON/OFF key does not affect character display. A 3270 terminal equipped with the APL feature can display all EBCDIC and APL characters regardless of whether the APL ON/OFF key is on or off. When VS APL asks for input it displays a prompt (such as six blanks) on the screen. It then positions the cursor in the same relative position that an IBM 2741 printing element would appear when the keyboard unlocks. You can then enter one or more APL characters. To correct a character, the cursor is moved back to the incorrect character with one of four cursor control keys at the lower righthand position of the keyboard. A correct character can then be typed or the INS MODE or DEL keys can be used to otherwise alter the line. The INS MODE key allows insertion of one or more characters at the position of the cursor. The DEL key deletes the character at the position of the cursor without leaving a blank space. When the input line is complete, press the ENTER key to send the input line to VS APL.

The default printing width (⎕*PW*) is 76. Bare (⎕) output greater than 76 characters is folded, that is characters beginning with the eightieth appear on the following line at the first character position on the line. If ⎕*PW* is set to greater than 76, output lines greater than 77 characters are similarly folded.

Special Keys

In addition to the ENTER, INS MODE, and DEL keys, other keys have important uses:

Key	Use
CLEAR	Blanks screen to allow display of additional output when *** appears in the lower left corner of the screen. The CLEAR key can also be used to clear the screen before entering additional input. Be aware though, that this also erases any current prompt and moves the cursor to the first location in the input area.
ERASE INPUT	Blanks or erases the input area of the screen, erasing any prompt, and moves the cursor to the first location in the input area.
ERASE EOF	Replaces all characters from the cursor location to the end of the input area with nulls. (Nulls are displayed as blanks but are not transmitted as characters.)
RESET	Resets the terminal after an invalid key is pressed or terminates character insertion (via the INS MODE key).
PA1	Used to signal attention. Pressing the PA1 key transmits a weak interrupt signal. Pressing the PA1 key transmits a strong interrupt signal. If the PA1 key is pressed during input, it is ignored.
PA2	If the terminal is in READ state, pressing PA2 signals the character *O* backspace *U* backspace *T*.

Key	Use
PFK 1-12	Used to transmit the line *PFK n* to VS APL, where *n* is the number of the program function key pressed. A sample workspace is provided in LIB 1 PFK which demonstrates how this facility may be used.

CURSOR MOVEMENT CONTROLS

The cursor is a symbol displayed on the screen that indicates to you the placement of the next input character. The cursor keys control the vertical and horizontal movement of the cursor.

Remember that when you are editing data on the 3270, the cursor keys do not enter blanks into the input line. False blanks (or null characters) entered on the screen by the cursor keys disappear when you press ENTER, and all entered characters are compressed to the left.

Use the SPACE BAR to enter blanks on the screen.

The cursor keys and their functions are as follows:

This key, a Typamatic key, moves the cursor up and off the screen. The cursor reappears at the bottommost line of the screen and continues upward until you stop pressing the key.

This key, a Typamatic key, moves the cursor down and off the screen. The cursor reappears at the topmost line of the screen and continues downward until you stop pressing the key.

This key, a Typamatic key, moves the cursor to the left and off the screen. The cursor reappears one line higher at the rightmost edge of the screen and continues moving to the left until you stop pressing the key.

This key, a Typamatic key, moves the cursor to the right and off the screen. The cursor reappears one line lower at the leftmost position of the screen and continues moving until you stop pressing the key.

This key, a Typamatic key, moves the cursor to the leftmost position of the next line. If the cursor is on the bottommost line, the cursor will reappear at the top of the screen.

This key has two functions, depending on whether the APL ALT key has been pressed. If APL ALT key is ON, this key is used in conjunction with other keyboard text keys to generate the compound APL characters.

Pressing this key with the APL ALT key ON generates the APL compound character DELTA-UNDERSCORE.

TEST REQ

Ignored; treated as null-line input to VS APL.

LIGHT PEN DETECT

Ignored; treated as null-line input to VS APL.

Backspacing

The backspace terminal control character (⎕TC[1]) is effected on the IBM 3270 only in bare output situations (⍞ output followed by ⍞ input) and only when it follows ⎕TC[3]. In all other situations, ⎕TC[1] produces a blot character (") on output. For example, the expression 'S',⎕TC[1],'/' is displayed on a 3270 as:

```
S"/
```

Function Editing

If you have the APL feature installed at your terminal, a simple way you can edit function lines is to enter:

[*line number* ⎕]

This will cause the function line with the specified number to be displayed on the screen. You can then:

- Delete characters using the DEL key
- Insert characters using the INS MODE key. (Be sure to press RESET when you are finished.)
- Replace characters by positioning the cursor and entering replacement characters.

It is important in function editing to determine (reset INS MODE first). whether the printing width is greater than or equal to the length of the line to be edited. If the line is larger than the printing width, it will span two adjacent lines, but may be edited as one entity.

You are permitted to correct any function line displayed on the 3270 screen by using the 3270 local editing function keys to position the cursor, overlay characters, insert and delete characters. You should, hovever, change only one function statement before pressing ENTER. A single statement may span two physical lines on the screen, but may be properly edited. When a statement spans lines, inserting characters on the first line forces characters to move from the last position on the first line to the first position on the second line. Deleting characters on the first line does not affect the display of the second line.

You are permitted to correct (using the 3270 local edit functions) any line which is displayed on the 3270 screen. This can be important when you are executing variations of single line APL statements in a desk calculator mode. Desk calculator mode on a 3270 is an effective alternative to function definition mode for the development of APL function statements because:

- you see the results of execution immediately,
- the development process requires fewer interactions with the system, since the function lines are edited in local mode on the screen,
- functions may be tested one line at a time, and the lines added to a function without rekeying.

Because of these characteristics, the desk calculator mode may be more useful than it would be on a typewriter terminal. The following suggestions will help you work more effectively in this environment:

- Start with a CLEAR screen, so that your entry appears at the top of the screen. This permits you to use the remainder of the screen for display of output. Additionally, you may position the 3270 cursor to the first character of your statement following the last output response from APL by pressing the field-jump key marked → | to quickly position the cursor at the top of the screen for modification and re-transmission.
- Try to display your data in such a way as to fill the screen from left to right, rather than top to bottom. Display data has vectors, rather than arrays in order to take maximum advantage of the 1920 characters available to you.

  When VSAPL (or more correctly, TSO) displays data on one line from the bottom of the screen, the characters '***' will appear on the screen (from TSO), and you will have to press ENTER or CLEAR to continue the output display, which, or course, clears the line at the top of the screen that you were working with.
- Data that is displayed on the screen as the result of the execution of a function, or the display of a variable, may be edited on the 3270 screen and used as input data to a function or variable assignment.

  This applies only to data which is displayed on a single 3270 line, or data transmitted as a vector whose length is less than the value of $\square PW$. For example, if the matrix defined by `X←3 1 ρ 'ABC'` is displayed on the 3270 by typing X, the following lines result:

```
      X
A
B
C
```

  Any of the lines may be edited and transmitted (one at a time), but they may not all be changed. If you had chosen to display the value of X by raveling it to a vector (, X), then the data would have been displayed as:

```
ABC
```

  on the screen. The value of the second row of X, which is B, could be changed to a D by positioning the cursor of the 3270 under the A, pressing the INSERT key, and typing: `X←3 1ρ '`

```
X←3 1ρ 'ABC
```

then position the cursor to the character *B*, press the RESET key, and type the character *D*;

then position the cursor to the character beyond the *C* (it actually doesn't make any difference where since all the characters to the right of *C* are null), and type '. Your line on the screen will look like:

```
X←3 1⍴'ADC'
```

when you press the ENTER key. Obviously, the second row of *X* could have been changed in a more straight-forward way using conventional APL statements. The principle of change is what was illustrated. Numeric data may be displayed and re-entered as input data as well (without the surrounding character data quotes).

- Functions which terminate with execution errors will display the function line which terminated, and an inverted caret indicating the point of error. If you recognize the cause of the error, you may position the cursor to the character before the function name, type a function-definition del ∇ (with or without INSERT mode). You may then correct the statement, and terminate the line with a function-edit termination del Δ, and press ENTER. Execution is then resumed in the standard way →⎕*LC*, or →*linenumber*.

These actions may seem to take more effort than to type all of the entries each time. You will have to use your own best judgment as to the appropriateness of this mode of 3270 operation. One additional advantage that the function editing procedure described above has over conventional function editing performed on a typewriter terminal, is to reduce the number of times that the system must respond to terminal input requests. Normal function editing of a single line in an APL function takes one terminal entry (if you type the entire line just as we changed it on the 3270, but this is prone to errors, unless the line is very short), a minimum of 3 terminal entries must be used for context editing, and as many as 5 or 6 may be used by an inexperienced user. All of these terminal entries put an additional load on the TSO system, and increase the time you must wait for each request before proceeding.

The 3270, on the other hand, allows you to visually verify each entry before transmission to TSO. Error corrections in typing may be handled by you at the terminal, and not by TSO. On a typewriter terminal, there is no character delete option when you are running APL, and each mis-typed character must be handled by APL/TSO intervention.

Error Situations

If you do not have the APL feature installed, you will not be able to use APL characters to communicate with VS APL. APL characters directed to the screen will be garbled and the screen format could be destroyed if compound characters are generated. In this event, press the CLEAR key to reformat the screen. This will allow you to use the PA2 key to cancel output and signal *O* backspace *U* backspace *T*.

APPENDIX E: VSPC MIGRATION CONSIDERATIONS

It is anticipated that some installations will install VSPC in order to take advantage of the additional facilities and performance features of that subsystem.

If your installation installs VSPC and requests that you move your workspaces to that system, you must be aware of the differences between these two subsystems.

LANGUAGE:

There are no differences in the interpreter portions of VS APL running under TSO and VSPC. From a language standpoint, then, VSAPL is transparent to the subsystem it is running on. The primary difference between VSAPL programs running under TSO and running under VSPC is the nature of the services offered by the subsystem (TSO or VSPC) to the APL user through Shared Variables.

Broadly speaking, migration considerations in VS APL fall into three main categories: differences in features offered, changes in terminal habits of the APL user, and workspace conversion. Of these three, the workspace conversion is the most critical. Since the APL language requires no changes in syntax, users who require workspaces to be transferred between systems should map out in advance the services they wish to be performed by the auxiliary processors. The services then may be provided by APL functions written to interface to each of the set of auxiliary processors supplied. Workspace migration would then involve replacement of these functions in the workspace.

- Sharing variables among users

  The facility to offer variables to other users running under VSPC is not available under TSO. Therefore, there is no conversion in migrating to VSPC. The user may wish to explore the application possibilities offered by this facility in justifying a move to VSPC (eg., sensitive applications may require the security facilities offered by this mode of operation, or data base applications may be offered using this facility which will enhance data integrity and provide faster response).

- Auxiliary Processors

  There is no overlap in the processors offered under TSO and those offered by VSPC. In particular, the facilities offered by processor 100 under TSO (the TSO Command Processor) are achieved in a different way under VSPC.

  Also missing from VSPC is the facilities of the STACK processor (Auxiliary Processor 101) which allows an APL program to issue APL System Commands like)LOAD,)SAVE,)COPY.

  Users who write applications which depend upon linking workspaces together via these system commands must achieve the results in another way.

  It should be noted that auxiliary processor 101 under TSO was written entirely by Yale University, and was not converted from the processor supplied by CMS (which used the STACK facilities of VM). It may be possible to write a STACK processor for VSPC using the auxiliary processor supplied with this IUP. This auxiliary processor differs from most other AP's because it interfaces directly with the VS APL executor.

  Sequential files written by other APL users or other language processing programs are read under TSO using AP111. Similar function may be achieved in VSPC using auxiliary processors 121 or 122.

  TSO files must first be IMPORTed to VSAM using standard utilities before access under VSPC.

- User Written Auxiliary Processors

  The VSAPL user under TSO may write an auxiliary processor to provide function not achieved with the supplied program (access to VSAM or ISAM files, for example). These auxiliary processors must meet the Shared Storage Manager interface described in the manual Writing Auxiliary Processors for VSAPL Under CMS. The interface described in that manual is different from that provided by VSPC described in a similar manual. The user must convert his private Auxiliary Processors to VSPC standards and functions provided by VSPC. (note, the considerations are the same as those for users converting from CMS VSAPL).

Terminal Support

The same set of terminals is supported under VSPC as TSO, but the operational characteristics may be different. The way in which VSPC handles the 3270, for example, is entirely different from TSO/TCAM.

VSPC also uses VTAM for its terminal control. When a 3767 is operated in SDLC mode, the operational characteristics are different. These differences are discussed in the VSPC literature.

Line Editing on 2741 and 3767

APL requires a line-feed to be in an input message indicating that all characters to the right are to be deleted. If your terminal has line-feed, you may use it in APL entry editing. Since the 2741 and 3767 do not have these keys, the ATTN key is used to indicate the point of line correction.

Under TSO, an ATTN is handled by TCAM and the entire line is dropped (instead of just the characters to the right). VSPC will handle such an attention properly, but the APL user under TSO must use other methods to indicate "logical end-of-line delete".

This is handled under TSO by creating an illegal overstrike at the point of correction and pressing RETURN. VSAPL/TSO handles an illegal overstrike by spacing to the point of the illegal character, typing a caret, and performing a line-feed/backspace.

Visually, the result is the same as if the user had pressed ATTN. Under VSPC, this action would have resulted in the entire line being echoed back to your terminal with the print mechanism positioned at the point of the illegal character.

VSAPL/TSO did not choose to implement the*echo* method of error reporting because, if the user has horizontal tabs set (⍞*HT*) the VSAPL response will be much quicker by tabbing to the point of error.

Users who switch from TSO to VSPC will achieve the same functional result by using the over-strike method of line correction, though they may be annoyed at the inconvenience under VSPC. Under TSO, if they use the ATTN method, they may be annoyed by the fact that they have to retype the entire line.

VSPC Workspace QUOTA

Users who migrate to VSPC should be aware that the Systems Administrator has much more control over the resources consumed. In particular, the)QUOTA command will return values which place a restriction on the number of workspaces that may be created.

Under TSO, this control mechanism is handled outside of the VSAPL environment.

PRIVATE/SHAREABLE LIBRARIES

This mechanism attempts to provide a facility similar to the PROJECT library function of VSPC and CMS. The PROJECT library concept was not implemented because of integrity and/or performance considerations in TSO.

In particular, no user may)SAVE into a private/shareable library except the owner, though anyone may)LOAD or)COPY it. Users who wish to implement a PROJECT library facility in TSO, must agree on an "OWNER" for the project. They must notify the "OWNER" of their wish to replace the PROJECT workspace by requiring the "OWNER" to)LOAD the new copy from a separate private/shareable library and then to)SAVE the copy into the PROJECT library.

PRIVATE LIBRARIES

Private libraries are protected from unauthorized browsing in a TSO environment by the following mechanisms:

-)LIB command cannot access another userid since every user is 1001 in TSO for the private library.
- Private workspaces are internally locked so that they may not be copied using OS utilities to another userid dataset and)LOADed under VSAPL.

  This means that the only way workspaces may be made portable to other TSO/VSAPL systems is to make the workspace a private/shareable or public workspace under VSAPL before exporting it.

PASSWORD PROTECTION

VSAPL For TSO attempts to use OS Password Protection if the user attempts to password protect the workspace. If password protection is not implemented in the system, (ie., no SYS1.PASSWORD dataset) the)SAVE or)WSID command will issue a diagnostic (a + on the date/time stamp for)SAVE).

If password protection is not implemented, the workspace is not LOCKed, since any lock would be meaningless when workspaces are available through other TSO facilities.

In VSPC, workspaces are VSAM datasets, and therefore may be password protected properly. VSAM is not used for workspace storage in TSO in order to allow its use on OS/MVT.

VSAPL INVOCATION PARAMETERS

VSAPL under TSO allows for additional functions to be performed which are not available to the CMS or VSPC user. In particular, this implementation provides a mechanism to discover the terminal type the user is running on (in order to select proper translate tables) TERM; to automatically load a workspace upon invocation (to provide a mechanism for an application that should completely shield the user from TSO and APL syntax and conventions).

A user may be automatically placed into VSAPL by a TSO logon procedure; VSAPL may then automatically load a workspace; the workspace may contain a latent expression; the latent expression may invoke an APL function which will read and analyze all user terminal requests. Thus an application user need not know TSO syntax, or APL syntax to communicate with an application program following logon.

WORKSPACE MIGRATION TO VSPC

The same utilities are used to communicate with VSPC from the TSO environment as those defined in the APL Installation Reference Manual for communicating with CMS.

Workspaces are entered into the VSPC environment by executing the VSPC Service Program with IMPORT control statements using the TSO workspace datasets as input, or a tape dataset created by using standard OS Utilities (IEBGENER) to copy the workspace dataset to tape.

The internal locks created by VSAPL/TSO have no effect in a VSPC environment. Private workspaces may be loaded into any library in VSPC and)LOADed by VSAPL regardless of their origin.

Workspaces that are to be transferred from VSPC to VSAPL for TSO may be EXPORTed using the VSPC Service Program as described in the Installation Reference Manual. Instead of using MOVEFILE under CMS, the user may use IEBGENER to create the DASD workspace, or it may be EXPORTED directly to a DASD OS Sequential dataset with a DSNAME conformable to the VSAPL naming conventions.

Workspaces coming from a non-TSO VSAPL environment are not subject to the private library restrictions described above. The first)SAVE from VSAPL for that workspace will encrypt it to prevent workspace "stealing" from private libraries.

SECONDARY MESSAGES

Yale University has added a second level message facility equivalent to that for other TSO components. This allows optional display of diagnostic information in increasing levels of detail. This facility does not interfere with normal APL behavior, but is accessed using the familiar TSO protocol. Any message which is displayed on the APL terminal which has a + as a final character, indicates that additional message detail may be retrieved by typing a '?' (upshift-Q). If this is the first terminal response following the message, an additional second level message will be displayed. If any other line is typed following the message, typing a ? will produce the familiar *SYNTAX ERROR* message from APL. These messages can give valuable insight to the APL programmer as to the nature of an APL Error message which is not gained in any of the other VS APL environments.

)*LIB* Output

The normal APL response to a)*LIB* command is a list of the APL workspaces (non-alphabetic in a CMS environment) in a columnar display. VS APL For TSO produces the display in row fashion, alphabetically. This format was chosen over other APL implementations because

- it is consistent with the)*FNS* and)*VARS* display,
- it saves paper and time on a typewriter terminal, and
- it will be displayed for full view on a 3270 Display without causing a screen overflow (and possible loss from view of the output).

APPENDIX F: ERROR MESSAGES

In your work at the terminal there will be occasions when certain conditions will prevent the system from successfully executing your entry and force the production of an error message instead. This appendix lists and discusses the error messages produced by VS APL that you may encounter at your terminal. For a description of the error messages produced by TSO, refer to the TSO Messages Publication listed in the preface. *TSO System Messages* publication listed in the preface.

Error Messages Issued by VS APL

VS APL generates three kinds of error messages, each dealing with a specific type of error situation:

- Error reports, which are produced when VS APL encounters an unexecutable statement.
- Trouble reports, which are produced when VS APL encounters difficulty with a system command.
- Executor messages, which are produced for a variety of error conditions.

VS APL also issues a specific message when an error is encountered by a distributed auxiliary processor.

Error Reports

Error reports are produced by VS APL when it encounters an APL statement that it cannot execute. This may be because you've used incorrect syntax in an entry, specified inappropriate arguments to a primitive function, or otherwise misused names or symbols in a statement. Error reports are also produced when the execution of your statement requires certain resources of the system that the system does not have, like additional space for names.

When an error report is displayed, VS APL indicates the point at which it stopped execution by displaying your erroneous statement and printing the caret symbol (^) below the point in the statement where the error was detected. You then have the option of correcting the statement or entering something new. For example:

```
      A←2×Z
VALUE ERROR
      A←2×Z
        ^
```

Here a *VALUE ERROR* was detected because an attempt was made to use a name (*Z*) that had not been assigned a value. You now have the opportunity of assigning a value to *Z* and reentering the statement.

VS APL Error Reports

The errors in APL language syntax reported by the VS APL Interpreter are:

```
DEFN ERROR
VALUE ERROR
RANK ERROR
LENGTH ERROR
SYNTAX ERROR
DOMAIN ERROR
INTERRUPT
NO SHARES
INTERFACE QUOTA EXHAUSTED
INDEX ERROR
```

For a description of the error reports generated by VS APL, refer to *APL Language*.

Trouble Reports

Trouble reports are produced by VS APL when it encounters a system command it cannot execute. When trouble is detected in a system command, the appropriate trouble report is immediately displayed. You then have the option of correcting your system command, changing some of the attributes of your workspace so that your command will work, or disregarding the offending entry and going on to something new.

The following lists and describes the trouble reports issued by VS APL. For each listed message, a cause for the message is indicated as well as a suggested action you might take to correct the error.

If any message you receive at the terminal ends in the character '+', typing a '?' (upshift-Q) as the next entry from the terminal will produce a message providing additional detail regarding the first message.

IMPROPER LIBRARY REFERENCE +

Cause: You are not authorized to use the referenced library, or you referenced a non-existent library.

Suggested Response: Make the necessary change in your specification. Make sure that you are authorized to use the referenced library.

Type a '?' (upshift-Q) to obtain more information.

INCORRECT COMMAND

Cause: The command you issued was specified incorrectly.

Suggested Response: Check the proper form of the command in the section "System Commands" and make the appropriate changes.

INCORRECT PASSWORD

Cause: You specified a password incorrect for the referenced workspace or you forgot to supply one.

Suggested Response: Correct the erroneous password or supply the missing password.

LIBRARY NOT AVAILABLE +

Cause: The referenced library is unavailable.

Suggested Response: Type a '*?*' (upshift-*Q*) to obtain more information.

MESSAGE LOST

Cause: You sent a message, but sent a weak interrupt signal immediately.

Suggested Response: Resend the message.

NO OBJECTS COPIED

Cause: A)*COPY* or)*PCOPY* command cannot be executed because there is insufficient disk space for temporary OS sequential datasets.

Suggested Response: Drop any unneeded workspaces and reissue the command or ask your system administrator to increase the size of your primary disk.

NOT A CLEAR WS

Cause: You issued the command)*SYMBOLS number,* but your active workspace was not clear.

Suggested Response: Save the workspace, load a clear workspace, issue the *SYMBOLS number* command, and copy the saved workspace.

NOT COPIED:list

Cause: The objects represented in list could not be copied either because they could not fit in your workspace or because you issued a protected copy command and the names already exist in the workspace.

Suggested Response: If appropriate, erase objects that are no longer needed and reissue the)*COPY* or)*PCOPY* command for the objects in list.

NOT ERASED:list

Cause: The items represented in list have not been erased as you requested; they do not exist as global objects in the workspace.

Suggested Response: Change the names of the listed objects as appropriate.

NOT FOUND:list

Cause: You attempted to copy objects that do not exist in the referenced workspace. The term list represents the names of the unfound objects.

Suggested Response: Change the names of the listed objects as appropriate. The objects needed may exist in another workspace or are named differently.

NOT GROUPED:list

Cause: The items represented in list have not been grouped as you requested; their names are not valid or are names of system functions or variables.

Suggested Response: Change the names of the listed objects as appropriate. If desired, recreate or extend the group including the new objects.

NOT GROUPED, NAME IN USE

Cause: The groupname you specified in the)*GROUP* command is already in use for a function or variable.

Suggested Response: Change groupname or erase object that already has that name.

NOT SAVED, THIS WS IS CLEAR WS

Cause: You attempted to save a clear workspace (that is a workspace whose name is *CLEAR WS*).

Suggested Response: Assign a name to the workspace via a)*WSID wsid* or)*SAVE wsid* command.

NOT SAVED, THIS WS IS WSID

Cause: Your active workspace does not conform to the name specified in the)*SAVE wsid* command and the name already exists in the library. The term WSID is the name of your active workspace.

Suggested Response: Change the name of your active workspace via the)*WSID wsid* command, change the name specified in the)*SAVE wsid* command, or drop the existing saved workspace.

SI DAMAGE

Cause: You edited or erased a pendent or suspended function, or you copied an object appearing in the state indicator list.

Suggested Response: Clear the state indicator list.

SPACE NOT AVAILABLE

Cause: The space indicated in the)*LOAD* or)*CLEAR* command cannot be allotted by the system.

Suggested Response: Change the size indication in the)*LOAD* or)*CLEAR* command.

STACK FULL

Cause: You specified a command whose execution would exhaust the available space in the execution control area.

Suggested Response: Enlarge the execution control area via the)*STACK number* command.

SYMBOL TABLE FULL

Cause: Sufficient room does not exist in the symbol table area of your active workspace for all the new names created by the command.

Suggested Response: Save the workspace, load a clear workspace, define the required number of symbols, and copy the saved workspace.

SYSTEM RESOURCE PROBLEM +

Cause: A machine malfunction was encountered in attempting to read or write to a OS sequential dataset.

Suggested Response: Reissue your previous command. If the report is still produced, see your system administrator.

Type a '*?*' (upshift-*Q*) to obtain more information.

USER NOT LOGGED ON

Cause: The user you indicated in the *)MSG* command is not signed-on to the system.

Suggested Response: Check to see that the account number you specified in the command is correct. Otherwise, contact your system administrator.

USER NOT RECEIVING

Cause: The user you designated in the *)MSG* command is blocking messages.

Suggested Response: If the message is important, contact the user directly. Otherwise, try again later.

WS FULL

Cause: The action you requested would overfill the available area in your active workspace.

Suggested Response: Erase unneeded objects in the workspace. If necessary, save the workspace and reload it with a larger size.

WS LOCKED

Cause: You failed to supply a password for the workspace.

Suggested Response: See your system administrator.

WS NOT FOUND

Cause: You specified a workspace name that could not be found.

Suggested Response: If the workspace name was incorrectly specified, change it. Otherwise, the workspace you need may not exist.

WS TOO LARGE

Cause: The size you specified in a *)LOAD* command is too small to accommodate the contents of the referenced workspace, or your maximum allowable workspace size cannot accommodate the contents of the referenced workspace.

Suggested Response: If possible, enlarge the size in your *)LOAD* command or issue the command without a size specification.

Yale Executor Messages

System command error reports in VS APL may be caused by many conditions in a TSO environment. Yale has chosen to implement a secondary message concept available thoughout the rest of TSO to allow the confused user to obtain more detailed information regarding each of the messages listed above which contain a + as the final character. Although not proper APL syntax, a question mark '?' entered as the only character following an error message will produce additional information regarding that message.

The following messages are fully described in LIB 3 MESSAGES workspace. You should obtain a listing of the messages in that workspace as the final authority for the meaning of messages in your environment. You are encouraged to read through the following messages to see the scope of situational errors you can create for yourself in the TSO environment.

In a number of cases, the error condition described will have no specific meaning to you. For these cases, the user action will direct you to see your system administrator. He will know how to interpret the error situation with the information supplied.

APL002I ERROR 1 INITIALIZING APL ASSIST. ASSIST NOT IN USE.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: HARDWARE ERROR.

EXPLANATION: THE 'APL ASSIST' FEATURE (A MICRO-CODED HARDWARE FEATURE AVAILABLE ON CERTAIN CPU MODELS) IS INSTALLED BUT NOT FUNCTIONING.

SYSTEM RESPONSE: INITIALIZATION CONTINUES. THE APL ASSIST FEATURE WILL NOT BE USED.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

APL020I ERROR CODE XX FROM SSM DURING INITIALIZATION. NO SHARED VARS.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: AN ERROR HAS OCCURRED DURING INITIALIZATION OF SHARED VARIABLE PROCESSING (I.E. AUXILIARY PROCESSERS). THE ERROR CODE DISPLAYED IN THE MESSAGE IS IN DECIMAL.

SYSTEM RESPONSE: INITIALIZATION CONTINUES. SHARED VARIABLE PROCESSING (AUXILIARY PROCESSORS) WILL NOT BE AVAILABLE.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

APL021I APL ASSIST INCOMPATIBLE WITH APL PROCESSOR. ASSIST NOT USED.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE VERSION LEVELS OF THE 'APL ASSIST' FEATURE (A MICRO-CODED HARDWARE FEATURE AVAILABLE ON CERTAIN CPU MODELS) AND OF THE VSAPL PROGRAM ARE BOTH DIFFERENT AND INCOMPATIBLE.

SYSTEM RESPONSE: INITIALIZATION CONTINUES. THE APL ASSIST FEATURE WILL NOT BE USED.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

APL107I SAVE AREA OVERFLOW. CALLEE=XXXXXX, CALLER=XXXXXX.

ENVIRONMENT: VSAPL.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: A LOGIC ERROR INTERNAL TO VSAPL HAS OCCURRED.

SYSTEM RESPONSE: VSAPL/TSO ABENDS THEREBY PASSING CONTROL (EVENTUALLY) TO THE ABEND RECOVERY ROUTINE. SEE ERR010 FOR MORE INFORMATION.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP OF THE ERROR.

APL108I SYSTEM ERROR IN APL PROCESSOR.

ENVIRONMENT: VSAPL.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: A LOGIC ERROR INTERNAL TO VSAPL HAS OCCURRED.

SYSTEM RESPONSE: A CLEAR WORKSPACE IS LOADED.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP OF THE ERROR.

ERR010S ABEND - SYSTEM CODE - XXX, USER CODE - XXXX.

ENVIRONMENT: ABEND ERROR RECOVERY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE VSAPL/TSO PROGRAM HAS ABNORMALLY TERMINATED WITH THE INDICATED ERROR CODE.

SYSTEM RESPONSE: FOR RECOVERY, THE SYSTEM ATTEMPTS THE FOLLOWING:

- A CONTINUE WORKSPACE IS SAVED.

- A COMMAND IS ISSUED TO TSO TO CAUSE IT TO REINVOKE A FRESH COPY OF VSAPL/TSO UPON TERMINATION OF THE CURRENT COPY.

- THE CURRENT COPY OF VSAPL/TSO TERMINATES SO THAT THE NEW COPY CAN GAIN CONTROL.

- THE NEW COPY THEN RELOADS THE CONTINUE WORKSPACE.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP OF THE ABEND.

ERR020S ERROR OCCURRED IN AN AUXILIARY PROCESSOR.

ENVIRONMENT: ABEND ERROR RECOVERY.

PROBABLE CAUSE: INSTALLATION ERROR.

EXPLANATION: THIS MESSAGE, WHEN DISPLAYED, ALWAYS FOLLOWS ERR010S. IT INDICATES THAT THE ABEND DESCRIBED BY ERR010S OCCURRED WITHIN AN AUXILIARY PROCESSOR. SEE ERR010 FOR MORE INFORMATION.

SYSTEM RESPONSE: SEE ERR010.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING STAFF OF THE ABEND.

ERR030S THE ATTEMPT TO SAVE A CONTINUE WORKSPACE HAS FAILED. SORRY.

ENVIRONMENT: ABEND ERROR RECOVERY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: A NON-RECOVERABLE ABEND RECURSION HAS OCCURRED DURING THE ATTEMPT TO SAVE A CONTINUE WORKSPACE AS A RESULT OF A PREVIOUS ABEND.

SYSTEM RESPONSE: THE VSAPL/TSO PROGRAM TERMINATES. THERE IS NO RECOVERY.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING STAFF OF THE ABEND.

INI010S YOU ARE NOT AUTHORIZED TO USE VSAPL/TSO.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: INSTALLATION RESTRICTION.

EXPLANATION: YOUR INSTALLATION HAS PROHIBITED YOU FROM USING THE VSAPL/TSO SYSTEM.

SYSTEM RESPONSE: VSAPL TERMINATES THEREBY RETURNING YOU TO THE TSO COMMAND LEVEL.

USER RESPONSE: OBTAIN FROM YOUR INSTALLATION AUTHORIZATION TO USE VSAPL/TSO.

INI020I SWITCH YOUR TERMINAL TO APL MODE.
INI021I TYPE AN OVERBAR AFTER YOU HAVE DONE SO.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: INFORMATION REQUEST.

EXPLANATION: DIFFERENT TERMINALS REQUIRE DIFFERENT CHARACTER TRANSLATIONS,
SO VSAPL/TSO ISSUES THESE MESSAGES IN ORDER TO DETERMINE WHICH TRANSLATION TO USE FOR YOU. BEFORE VSAPL/TSO INITIALIZATION WILL PROCEED, YOU MUST RESPOND TO THESE MESSAGES BY TYPING AN 'OVERBAR' WHICH IS DONE BY HOLDING DOWN THE SHIFT KEY WHILE SIMULTANEOUSLY PRESSING THE DIGIT 2 KEY.

ALSO AT THIS TIME, YOU MUST CHANGE YOUR TERMINAL FROM ITS NORMAL MODE TO APL MODE. FOR MANY TERMINALS, THIS IS DONE BY PRESSING AN 'APL' SWITCH LOCATED SOMEWHERE ON THE TERMINAL. FOR OTHER TERMINALS (E.G. IBM'S 2741), THIS IS DONE BY PHYSICALLY CHANGING A TYPING ELEMENT.

SYSTEM RESPONSE: INITIALIZATION WAITS UNTIL IT RECEIVES A PROPER RESPONSE. INITIALIZATION THEN CONTINUES. IF AN IMPROPER RESPONSE IS RECEIVED, THEN THE MESSAGES ARE REISSUED AND INITIALIZATION AGAIN WAITS FOR A PROPER RESPONSE.

USER RESPONSE: SWITCH YOUR TERMINAL TO APL MODE AND THEN TYPE THE OVERBAR AS DESCRIBED ABOVE. IF, HOWEVER, YOU DETERMINE AT THIS POINT THAT YOUR TERMINAL WILL NOT SUPPORT APL, YOU MAY ABORT BY PRESSING THE ATTENTION (BREAK, INTERRUPT) KEY.

NOTE, AN ALTERNATIVE WAY OF SUPPLYING TO VSAPL/TSO THE NECESSARY TRANSLATION INFORMATION IS BY USING THE TERMCODE OPERAND WHEN YOU INVOKE VSAPL.

INI030E UNRECOGNIZED TERMINAL CODE - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: EITHER THE VALUE OF THE 'TERMCODE' OPERAND OR THE RESPONSE TO THE INI020I MESSAGE IS INVALID. SEE INI020 FOR MORE INFORMATION.

SYSTEM RESPONSE: MESSAGES INI020I AND INI021I ARE DISPLAYED AND THEN VSAPL WAITS FOR A VALID RESPONSE.

USER RESPONSE: RESPOND BY TYPING AN 'OVERBAR' . SEE INI020 FOR MORE INFORMATION.

INI100S UNRECOGNIZED OPERAND - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: DURING SYNTAX ANALYSIS OF THE INVOCATION OPERANDS, AN UNRECOGNIZABLE CHARACTER STRING HAS BEEN ENCOUNTERED.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH CORRECT OPERANDS.

INI110S AMBIGUOUS OPERAND - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: AN INADEQUATELY IDENTIFIED INVOCATION OPERAND HAS BEEN GIVEN.

FOR EXAMPLE, 'A(...)' MAY BE EITHER 'APNAMES(...)' OR 'AUTOLOAD(...)'.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: MORE FULLY IDENTIFY THE AMBIGUOUS OPERAND. FOR EXAMPLE, 'AP(...)' AND 'AU(...)' ARE BOTH UNAMBIGUOUS.

INI120S REDUNDANT OPERAND - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: AN INVOCATION OPERAND HAS BEEN ENCOUNTERED MORE THAN ONCE (EITHER ITSELF OR ITS ALIAS) DURING OPERAND SYNTAX ANALYSIS. FOR EXAMPLE, THE FOLLOWING ARE ALL MUTUALLY REDUNDANT: WORKSPACE(...) WSSIZE(...) SIZE(...) W(...) WS(...) ETC.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND AVOIDING REDUNDANCIES.

INI130S INVALID DSNAME - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A DATA SET NAME GIVEN IN THE LOADLIB OPERAND CONTAINS INVALID SYNTAX. NOTE, MEMBER NAMES MAY NOT BE SPECIFIED. PASSWORDS MAY BE SPECIFIED AS FOLLOWS:

DSN/PSWD

IF THE NAME IS FULLY QUALIFIED, THEN IT MUST BE ENCLOSED WITHIN SINGLE QUOTES. IF IT IS NOT, THEN YOUR USERID WILL AUTOMATICALLY BE PREFIXED TO IT.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH THE ERROR CORRECTED.

INI140S INVALID PASSWORD - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A PASSWORD GIVEN WITH A DATA SET NAME IN THE LOADLIB OPERAND CONTAINS INVALID SYNTAX. SPECIFICLY, IT IS LONGER THAN EIGHT CHARACTERS.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH THE ERROR CORRECTED.

INI150S ALLOCATION ERROR - RC=XX, DARC=XXXX, CTRC=XXXX - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: A DATA SET GIVEN BY THE LOADLIB OPERAND COULD NOT BE ALLOCATED. 'RC' IS THE ERROR CODE FROM THE DAIR (DYNAMIC ALLOCATION INTERFACE ROUTINE) ROUTINE. 'DARC' IS THE ERROR CODE FROM THE DYNAMIC ALLOCATION ROUTINE ITSELF. 'CTRC' IS THE ERROR CODE FROM THE CATALOG MANAGEMENT ROUTINE. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: RE-ATTEMPT THE VSAPL COMMAND. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

INI160S CONCATENATION FAILURE - RC=XX, DARC=XXXX - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE LOADLIB OPERAND GAVE MORE THAN ONE DATA SET NAME AND THE ATTEMPT BY VSAPL TO CONCATENATE THEM TOGETHER FAILED. 'RC' IS THE ERROR CODE FROM THE DAIR (DYNAMIC ALLOCATION INTERFACE ROUTINE) ROUTINE. 'DARC' IS THE ERROR CODE FROM THE DYNAMIC ALLOCATION ROUTINE ITSELF. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: VERIFY THAT THE LOADLIB DATA SETS ARE ALL PARTITIONED LOAD LIBRARIES.

INI170S OPEN FAILURE - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S OPEN ROUTINES FAILED TO OPEN THE LOADLIB DATA SETS.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: RE-ATTEMPT THE VSAPL COMMAND. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

INI180S INVALID AUXILIARY PROCESSOR NAME - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A NAME GIVEN BY THE APNAMES OPERAND HAS INVALID SYNTAX.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH CORRECT OPERANDS.

INI190S AUXILIARY PROCESSOR NOT FOUND - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: AN AUXILIARY PROCESSOR NAME GIVEN BY THE APNAMES OPERAND WAS NOT FOUND IN EITHER THE LOADLIB DATA SETS (IF ANY) OR IN YOUR INSTALLATION'S LINK-LIBRARY CONCATENATION.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: VERIFY BOTH THE APNAMES OPERAND AND THE LOADLIB OPERAND.

INI200S INVALID SPACE SIZE - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A MEMORY SIZE VALUE GIVEN BY THE WORKSPACE (WSSIZE, SIZE), FREESPACE, OR SHARESPACE OPERAND EITHER CONTAINS INVALID SYNTAX OR RESOLVES TO MORE THAN 16 MEGA-BYTES.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH CORRECT DATA.

INI210W NO AUXILIARY PROCESSORS LOADED. VALUE IGNORED - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A SHARESPACE OPERAND HAS BEEN GIVEN, BUT NO AUXILIARY PROCESSORS HAVE BEEN LOADED. SHARESPACE IS USED ONLY BY AUXILIARY PROCESSOR COMMUNICATIONS.

SYSTEM RESPONSE: VSAPL CONTINUES. THE SHARESPACE OPERAND IS IGNORED.

USER RESPONSE: NONE.

INI220W MINIMUM SHARESPACE IS 4K. VALUE IGNORED - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A SHARESPACE VALUE LESS THAN 4K (4096) BYTES HAS BEEN GIVEN.

SYSTEM RESPONSE: VSAPL CONTINUES. A VALUE OF 4K IS USED FOR SHARESPACE.

USER RESPONSE: NONE.

INI230W MINIMUM WORKSPACE IS 32K. VALUE IGNORED - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A WORKSPACE VALUE LESS THAN 32K (32768) BYTES HAS BEEN GIVEN.

SYSTEM RESPONSE: VSAPL CONTINUES. A VALUE OF 32K OR MORE IS USED FOR WORKSPACE.

USER RESPONSE: NONE.

INI240S MEMORY ALLOCATION ERROR.
INI241S XXXXXXXX BYTES AVAILABLE.
INI242S XXXXXXXX BYTES NEEDED.
INI243S (XXXXXXXX BYTES NEEDED FOR THE WORKSPACE).
INI244S (XXXXXXXX BYTES NEEDED FOR THE SHARESPACE).
INI245S (XXXXXXXX BYTES NEEDED FOR THE AUXILIARY PROCESSOR WORK AREAS).

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THERE IS INSUFFICIENT MEMORY TO RUN VSAPL AS REQUESTED. INI241S DISPLAYS THE AMOUNT OF MEMORY AVAILABLE AFTER VSAPL ITSELF AND ALL AUXILIARY PROCESSORS (IF ANY) HAVE BEEN LOADED. MESSAGES INI243S, INI244S, AND INI245S ARE DISPLAYED ONLY IF AUXILIARY PROCESSORS HAVE BEEN LOADED SINCE THE VALUES DISPLAYED BY INI244S AND INI245S ARE ZERO OTHERWISE.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: EITHER CHANGE YOUR APNAMES, WORKSPACE (WSSIZE, SIZE), AND SHARESPACE SO AS TO REDUCE YOUR CORE NEEDS, OR RELOGON TO TSO WITH A LARGER REGION.

INI250W ONLY XXXXXXXX BYTES OF FREESPACE LEFT IN THE REGION.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: AFTER VSAPL AND ALL OF THE REQUESTED AUXILIARY PROCESSORS HAVE BEEN LOADED, AND AFTER THE WORKSPACE, SHARESPACE, AND AUXILIARY PROCESSOR WORK AREAS HAVE BEEN ALLOCATED, THERE IS INSUFFICIENT FREESPACE LEFT TO SATISFY THE AMOUNT REQUESTED BY THE FREESPACE OPERAND.

SYSTEM RESPONSE: VSAPL CONTINUES WITH WHATEVER FREESPACE IS AVAILABLE.

USER RESPONSE: NONE.

INI260S INVALID WORKSPACE NAME - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A SYNTAX ERROR HAS BEEN FOUND IN THE WORKSPACE NAME GIVEN BY THE AUTOLOAD OPERAND. THE FOLLOWING FORMS ARE VALID:

AU(WSID)

AU(WSID:PSWD)

AU(LIBNO WSID)

AU(LIBNO WSID:PSWD).

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: REISSUE THE VSAPL COMMAND WITH CORRECT DATA.

INI270S AUTOLOAD WORKSPACE NOT FOUND - X.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE WORKSPACE DATA SET TO BE AUTOMATICALLY LOADED WAS NOT CATALOGED.

SYSTEM RESPONSE: VSAPL TERMINATES AFTER COMPLETING THE SYNTAX ANALYSIS.

USER RESPONSE: VERIFY THAT YOU HAVE GIVEN THE CORRECT WORKSPACE NAME.

INI280W WARNING, YOUR CONTINUE WORKSPACE WILL BE DROPPED WHEN YOU ISSUE)OFF.

ENVIRONMENT: INITIALIZATION.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A CONTINUE WORKSPACE EXISTS, BUT YOU HAVE REQUESTED (VIA THE AUTOLOAD OPERAND) THAT ANOTHER WORKSPACE BE AUTOMATICALLY LOADED INSTEAD. HOWEVER, THE OFF COMMAND AS PART OF ITS NORMAL PROCESSING ALWAYS DROPS THE CONTINUE WORKSPACE IF IT EXISTS.

SYSTEM RESPONSE: VSAPL INITIALIZATION CONTINUES.

USER RESPONSE: IF YOU WANT TO PRESERVE THE CONTENTS OF THE CONTINUE WORKSPACE, THEN BEFORE YOU ISSUE THE OFF COMMAND, YOU MUST LOAD IT AND THEN SAVE IT INTO A DIFFERENT WORKSPACE.

LIB000E AN I/O ERROR HAS OCCURRED WHILE READING(WRITING) THE WORKSPACE(WORK DATA SET).

LIB001E ERROR DATA - CCHHR XXXXXXXXXX, CCW XX-XXXXXX-XXXX-XXXX, CSWSTAT/COUNT XXXXXXXX, SENSE XXXX.

ENVIRONMENT: LOAD, SAVE, COPY, OR PCOPY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S HARDWARE OR SOFTWARE HAS DETECTED INVALID OR INCONSISTANT DATA WHILE TRANSFERING DATA BETWEEN MEMORY AND DIRECT ACCESS STORAGE.

- 'CCHHR' IS THE ABSOLUTE DISK ADDRESS BEING READ OR WRITTEN AT THE TIME OF THE ERROR.

- 'CCW' IS THE CHANNEL COMMAND WORD DOING THE READING OR WRITING.

- 'CSWSTAT/COUNT' IS THE STATUS FLAGS AND RESIDUAL COUNT ASSOCIATED WITH THE TERMINATION OF THE READ OR WRITE OPERATION.

- 'SENSE' IS, IF APPROPRIATE, THE FIRST TWO SENSE BYTES RETURNED FROM THE DIRECT ACCESS DEVICE CONCERNING THE ERROR.

NOTE, ALL DATA IS DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: THE COMMAND IS TERMINATED. IF THE PREVIOUSLY LOADED WORKSPACE HAS BEEN OVERWRITTEN IN MEMORY, THEN A CLEAR WORKSPACE IS LOADED.

USER RESPONSE: RE-ATTEMPT THE COMMAND, IF THE ERROR IS PERSISTENT, THEN NOTIFY YOUR INSTALLATION'S OPERATIONS STAFF.

LIB010I YOU MAY DROP FROM OR SAVE INTO ONLY YOUR OWN LIBRARIES

ENVIRONMENT: DROP OR SAVE.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE LIBRARY NUMBER THAT YOU USED ON THE DROP OR SAVE COMMAND ALREADY BELONGS TO ANOTHER USER.

SYSTEM RESPONSE: THE DROP OR SAVE FUNCTION IS NOT EXECUTED.

USER RESPONSE: FOR THE SAVE COMMAND, TRY USING ANOTHER LIBRARY NUMBER.

LIB020I LIBRARY NOT FOUND.

ENVIRONMENT: LIB.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE INDICATED LIBRARY DOES NOT CURRENTLY EXIST ON THE SYSTEM (I.E. THE SYSTEM CATALOG ENTRY FOR THE LIBRARY WAS NOT FOUND).

SYSTEM RESPONSE: THE LIB COMMAND IS ENDED.

USER RESPONSE: RE-ISSUE THE LIB COMMAND USING A CORRECT LIBRARY NUMBER.

LIB030I THE LIBRARY IS EMPTY.

ENVIRONMENT: LIB.

PROBABLE CAUSE: SEE BELOW.

EXPLANATION: A.) FOR BOTH PUBLIC LIBRARIES AND NON-SHAREABLE PRIVATE LIBRARIES, THERE ARE CURRENTLY NO WORKSPACES SAVED IN THE LIBRARY.

B.) FOR SHAREABLE PRIVATE LIBRARIES, THE LIBRARY EXISTS (I.E. THE SYSTEM CATALOG ENTRY FOR THE LIBRARY WAS FOUND), BUT THERE ARE NO WORKSPACES SAVED IN THE LIBRARY. NOTE, THIS CONDITION CANNOT EXIST UNLESS THE WORKSPACE DATA SETS WERE SCRATCH INDEPENDENTLY OF VSAPL.

SYSTEM RESPONSE: THE LIB COMMAND IS ENDED.

USER RESPONSE: N/A.

LIB040W MORE WORKSPACE DATA SETS EXIST IN THIS LIBRARY THAN CAN BE LISTED.

ENVIRONMENT: LIB.

PROBABLE CAUSE: SYSTEM LIMITATION.

EXPLANATION: DUE TO INTERNAL MEMORY LIMITATIONS, LIBRARIES CONTAINING MORE THAN APPROXIMATELY 113 WORKSPACES MAY NOT BE FULLY LISTED (ALTHOUGH THEY MAY STILL BE LOADED, SAVED, OR DROPPED).

SYSTEM RESPONSE: ONLY THE ALPHABETICALLY FIRST 113 (APPROXIMATELY) WORKSPACES ARE LISTED.

USER RESPONSE: A FULL LISTING OF THE LIBRARY MAY BE OBTAINED USING THE TSO 'LISTCAT' COMMAND. AN EXPLANATION OF THE DATA SET NAMING CONVENTIONS FOR VSAPL WORKSPACES SHOULD BE OBTAINABLE FROM YOUR INSTALLATION.

LIB050E SYSTEM CATALOG SEARCH ERROR - DISPLAY ABORTED.

ENVIRONMENT: LIB.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: A FAILURE IN THE SYSTEM'S CATALOG SEARCH FUNCTION HAS OCCURRED.

SYSTEM RESPONSE: THE LIB COMMAND IS TERMINATED.

USER RESPONSE: RE-ATTEMPT THE LIB COMMAND. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB060E THE WORKSPACE DATA SET DOES NOT CONTAIN A VALID APL WORKSPACE.

ENVIRONMENT: LOAD, COPY, PCOPY.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: AN OS DATA SET HAS A VSAPL WORKSPACE TYPE DATA SET NAME BUT DOES NOT ACTUALLY CONTAIN A VSAPL WORKSPACE (I.E. THE CONTENTS OF THE DATA SET HAVE NOT BEEN CREATED BY THE SAVE COMMAND).

SYSTEM RESPONSE: THE LOAD, COPY, OR PCOPY FUNCTION IS NOT EXECUTED.

USER RESPONSE: AVOID NAMING NON-VSAPL WORKSPACE DATA SETS WITH VSAPL TYPE DATA SET NAMES.

LIB070E WORKSPACE DATA SET OPEN FAILURE.

ENVIRONMENT: LOAD, COPY, PCOPY, SAVE.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S DATA SET OPEN FUNCTION HASF FAILED FOR THE WORKSPACE DATA SET.

SYSTEM RESPONSE: THE LOAD, SAVE, COPY, OR PCOPY COMMAND IS NOT EXECUTED.

USER RESPONSE: RE-ATTEMPT THE COMMAND. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB080E BLKSIZE TOO LARGE.

ENVIRONMENT: LOAD, COPY, PCOPY.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE NORMAL BLKSIZE FOR ALL VSAPL WORKSPACE DATA SETS IS 800. IN FACT, THE SAVE COMMAND WILL ALWAYS USE AN 800 BYTE BLKSIZE WHENEVER IT WRITES A WORKSPACE TO A DATA SET. HOWEVER, THE LOAD COMMAND (AND, THEREFORE, THE COPY AND PCOPY COMMANDS) WILL READ WORKSPACE DATA SETS HAVING ANY VALID (I.E. MULTIPLE OF 80) BLKSIZE UP TO A MAXIMUM IMPOSED BY THE AVAILABILITY OF FREE MEMORY.

SYSTEM RESPONSE: THE LOAD, COPY, OR PCOPY COMMAND IS NOT EXECUTED.

USER RESPONSE: REVERT TO TSO AND EITHER COPY THE WORKSPACE TO A DATA SET HAVING SMALLER BLKSIZE OR REINVOKE VSAPL USING THE FREESPACE OPERAND TO INCREASE AVAILABLE FREE MEMORY.

LIB090I THE WORKSPACE HAS BEEN IMPROPERLY MOVED FROM ANOTHER LIBRARY.

ENVIRONMENT: LOAD, COPY, OR PCOPY.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: VSAPL/TSO ATTEMPTS TO DETECT WHEN A WORKSPACE HAS BEEN IMPROPERLY COPIED FROM ONE LIBRARY TO ANOTHER OUTSIDE OF ITS CONTROL (I.E. VIA TSO COMMANDS OR BATCH PROGRAMS). THE FOLLOWING COPY SITUATIONS ARE POSSIBLE:

- ANY WORKSPACE MAY BE COPIED INTO ANOTHER WORKSPACE IN THE SAME LIBRARY.

- ANY WORKSPACE FROM A SHAREABLE LIBRARY (PUBLIC OR PRIVATE) MAY BE COPIED INTO ANY OTHER SHAREABLE LIBRARY (PUBLIC OR PRIVATE).

- NO WORKSPACE FROM A SHAREABLE LIBRARY (PUBLIC OR PRIVATE) MAY BE COPIED INTO A NON-SHAREABLE LIBRARY. USE THE VSAPL LOAD AND SAVE COMMANDS TO ACCOMPLISH THIS.

-NO WORKSPACE FROM A NON-SHAREABLE LIBRARY MAY BE COPIED INTO ANY OTHER LIBRARY.

- IF WORKSPACES ARE TO BE EXPORTED TO OTHER VSAPL/TSO INSTALLATIONS, THEN THEY SHOULD BE SAVED INTO SHAREABLE LIBRARIES (PUBLIC OR PRIVATE) BEFORE BEING UNLOADED TO TAPE. SUBSEQUENTLY, THEY SHOULD BE LOADED INTO SHAREABLE LIBRARIES AT THE RECEIVING INSTALLATION.

- THERE NO RESTRICTIONS ON WORKSPACES BEING EXPORTED TO OR IMPORTED FROM OTHER VSAPL/VSPC OR VSAPL/CMS INSTALLATIONS.

WARNING, THESE RESTRICTIONS DO NOT PROVIDE ABSOLUTE SECURITY SINCE THEY CAN BE DEFEATED BY A SOPHISTICATED USER.

SYSTEM RESPONSE: THE LOAD, COPY, OR PCOPY FUNCTION IS NOT EXECUTED.

USER RESPONSE: RETURN THE WORKSPACE TO ITS ORIGINAL LIBRARY.

LIB100I THE WORKSPACE SIZE THAT YOU REQUESTED IS LARGER THAN THE AVAILABLE SPACE IN YOUR REGION.

ENVIRONMENT: LOAD.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A WORKSPACE SIZE HAS BEEN GIVEN WITH THE LOAD COMMAND, BUT IT IS LARGER THAN THE WORKSPACE MEMORY AVAILABLE IN YOUR REGION.

SYSTEM RESPONSE: THE LOAD COMMAND IS NOT EXECUTED.

USER RESPONSE: EITHER RE-ISSUE THE LOAD COMMAND WITH A SMALLER WORKSPACE SIZE, OR REVERT TO TSO AND REINVOKE VSAPL USING THE WORKSPACE OPERAND TO PROVIDE A LARGER WORKSPACE MEMORY, OR REVERT TO TSO AND RE-LOGON INTO A LARGER REGION AND THEN REINVOKE VSAPL.

LIB110E THE SIZE OF THE WORKSPACE TO BE LOADED IS SMALLER THAN THE SYSTEM DEFINED MINIMUM (32K).

ENVIRONMENT: LOAD.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A WORKSPACE SIZE THAT IS LESS THAN 32K HAS BEEN GIVEN ON A LOAD COMMAND. VSAPL DOES NOT ACCEPT A SIZE LESS THAN 32K.

SYSTEM RESPONSE: THE LOAD COMMAND IS NOT EXECUTED.

USER RESPONSE: RE-ISSUE THE LOAD COMMAND EITHER WITHOUT A SIZE VALUE OR WIT A SIZE THAT IS 32K OR LARGER.

LIB120I THE USED PORTION OF THE WORKSPACE TO BE LOADED IS TOO LARGE.

ENVIRONMENT: DROP, COPY, OR PCOPY.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE STORAGE OCCUPIED BY TABLES, VARIABLES, AND FUNCTIONS IN THE WORKSPACE TO BE LOADED EXCEEDS THE AVAILABLE WORKSPACE MEMORY.

SYSTEM RESPONSE: THE LOAD, COPY, OR PCOPY COMMAND IS NOT EXECUTED.

USER RESPONSE: EITHER REVERT TO TSO AND REINVOKE VSAPL USING THE WORKSPACE OPERAND TO INCREASE AVAILABLE WORKSPACE MEMORY, OR REVERT TO TSO AND RELOGON INTO A LARGER REGION AND THEN REINVOKE VSAPL.

LIB130E THE WORKSPACE DATA SET IS EMPTY.

ENVIRONMENT LOAD, COPY, OR PCOPY.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: EITHER AN EMPTY OS DATA SET HAS BEEN GIVEN A WORKSPACE TYPE DATA SET NAME, OR A WORKSPACE DATA SET HAS BEEN EMPTIED VIA NON-VSAPL MEANS.

SYSTEM RESPONSE: THE LOAD, COPY, OR PCOPY COMMAND IS NOT EXECUTED.

USER RESPONSE: A BACKUP COPY OF THE WORKSPACE MUST BE RESTORED.

LIB140E THE RECORDED SIZE OF THE WORKSPACE TO BE LOADED IS GREATER THAN ITS ACTUAL SIZE.

ENVIRONMENT: LOAD, COPY, OR PCOPY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE WORKSPACE DATA SET CONTAINS ONLY AN INCOMPLETE (AND, THEREFORE, AN UNUSABLE) COPY OF THE WORKSPACE BEING LOADED.

SYSTEM RESPONSE: IF THE ERROR IS DETECTED BEFORE THE CURRENTLY LOADED WORKSPACE IS OVERWRITTEN, THEN THE COMMAND (LOAD, COPY, OR PCOPY) IS NOT EXECUTED@ OTHERWISE, A CLEAR WORKSPACE IS LOADED.

USER RESPONSE: A BACKUP COPY OF THE WORKSPACE MUST BE RESTORED.

LIB160I YOU ARE NOT AUTHORIZED TO SAVE WORKSACES IN THIS LIBRARY.

ENVIRONMENT: SAVE, OR WSID.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: YOUR INSTALLATION HAS PROHIBITED YOU FROM ALLOCATING THIS PUBLIC LIBRARY OR SHAREABLE PRIVATE LIBRARY TO YOURSELF. YOU MAY NOT SAVE YOUR WORKSPACES INTO IT REGARDLESS OF WHETHER OR NOT IT ALREADY EXISTS. NOTE, IF THE LIBRARY DOES EXIST, THEN YOU MAY, OF COURSE, LOAD WORKSPACES FROM IT.

SYSTEM RESPONSE: THE SAVE, OR WSID COMMAND IS NOT EXECUTED.

USER RESPONSE: EITHER SAVE THE WORKSPACE INTO YOUR NON-SHAREABLE PRIVATE LIBRARY (I.E. UNNUMBERED LIBRARY) OR DETERMINE FROM YOUR INSTALLATION WHAT LIBRARY NUMBERS YOU ARE AUTHORIZED TO USE.

LIB170I THE WORKSPACE MUST BE NAMED BEFORE IT CAN BE SAVED.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE WORKSPACE BEING SAVED IS STILL NAMED 'CLEAR WS'.

SYSTEM RESPONSE: THE SAVE COMMAND IS NOT EXECUTED.

USER RESPONSE: EITHER NAME THE WORKSPACE WITH THE WSID COMMAND BEFORE SAVING IT, OR SUPPLY THE WORKSPACE NAME ON THE SAVE COMMAND ITSELF.

LIB180I A CONTINUE WORKSPACE CAN ONLY BE SAVED VIA THE)CONTINUE COMMAND.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE WORKSPACE NAME 'CONTINUE' IS RESERVED FOR USE BY THE CONTINUE COMMAND.

SYSTEM RESPONSE: THE SAVE COMMAND IS NOT EXECUTED.

USER RESPONSE: CHOOSE A DIFFERENT WORKSPACE NAME.

LIB190I YOU ARE ATTEMPTING TO CHANGE THE NAME OF YOUR WORKSPACE, BUT THE NEW NAME ALREADY EXISTS IN THE LIBRARY.

LIB191I IF THIS IS REALLY WHAT YOU WANT TO DO, THEN USE THE)WSID COMMAND TO CHANGE THE WORKSPACE NAME BEFORE ATTEMPTING THE)SAVE.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: YOU HAVE SPECIFIED A WORKSPACE NAME ON THE SAVE COMMAND THAT IS DIFFERENT FROM WHAT THE NAME OF THE CURRENTLY LOADED WORKSPACE HAS BEEN. THIS IS VALID ONLY IF THE NEW NAME DOES NOT ALREADY EXIST IN THE LIBRARY.

SYSTEM RESPONSE: THE SAVE COMMAND IS NOT EXECUTED.

USER RESPONSE: EITHER OMIT THE NAME FROM THE SAVE COMMAND THEREBY USING THE ORIGINAL NAME, OR CHOOSE ANOTHER NAME THAT DOES NOT ALREADY EXIST IN THE LIBRARY, OR USE THE WSID COMMAND TO RENAME THE CURRENTLY LOADED WORKSPACE AND THEN REISSUE THE SAVE COMMAND.

LIB200E WORK DATA SET ALLOCATION FAILURE.

ENVIRONMENT: COPY OR PCOPY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: POSSIBLY THERE IS INSUFFICIENT DIRECT ACCESS SCRATCH SPACE IN THE HOST SYSTEM OR THERE ARE AN INSUFFICIENT NUMBER OF 'DD DYNAM' DD CARDS IN YOUR TSO LOGON PROCEDURE.

SYSTEM RESPONSE: THE COPY OR PCOPY FUNCTION IS NOT EXECUTED.

USER RESPONSE: RE-ATTEMPT THE COMMAND. IF THE FAILURE PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB210E WORK DATA SET OPEN FAILURE.

ENVIRONMENT: COPY OR PCOPY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S OPEN FUNCTION HAS FAILED FOR A TEMPORARY WORK DATA SET REQUIRED BY THE COPY OR PCOPY COMMAND.

SYSTEM RESPONSE: THE COPY OR PCOPY FUNCTION IS NOT EXECUTED.

USER RESPONSE: RE-ATTEMPT THE COMMAND. IF THE FAILURE PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB220E WORK DATA SET TOO SMALL.

ENVIRONMENT: COPY OR PCOPY.

PROBABLE CAUSE: SYSTEM LIMITATION.

EXPLANATION: THE SOURCE WORKSPACE IS VASTLY LARGER THAN THE SINK.

SYSTEM RESPONSE: THE COPY OR PCOPY COMMAND IS NOT EXECUTED.

USER RESPONSE: RE-ATTEMPT THE COPY USING THE SINK AS THE SOURCE AND THE SOURCE AS THE SINK. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB230I THE MAXIMUM SUPPORTED LIBRARY NUMBER IS 9...9.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, DROP, OR LIB.

PROBABLE CAUSE: INSTALLATION LIMITATION.

EXPLANATION: YOU HAVE USED A LIBRARY NUMBER THAT IS TOO LARGE.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: CHOOSE A SMALLER LIBRARY NUMBER.

LIB240W MULTIPLE VERSIONS OF THIS LIBRARY EXIST.
LIB241W ACCESS IS LIMITED TO THE FIRST, AND THAT IS CATALOGED UNDER XXXXXXXX.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, LIB, OR DROP.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: MULTIPLE LIBRARY IDENTIFIERS FOR THIS LIBRARY EXIST IN THE SYSTEM CATALOG.

SYSTEM RESPONSE: THE COMMAND PROCEEDS USING THE ALPHABETICALLY FIRST IDENTIFIER FOUND. LIBRARIES ASSOCIATED WITH THE REMAINING IDENTIFIERS ARE INACCESSIBLE TO VSAPL.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

A.) LIB250E A SECONDARY ERROR HAS OCCURRED DURING LIBRARY CREATION PROCESSING.

-OR-

B.) LIB250E A SECONDARY ERROR HAS OCCURRED DURING LIBRARY DELETION PROCESSING.

ENVIRONMENT: SAVE OR DROP.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: EITHER THE SYSTEM'S CATALOG MANAGEMENT FUNCTION OR TSO'S DYNAMIC ALLOCATION INTERFACE ROUTINE HAS FAILED DURING AN ATTEMPT TO EITHER (A) ADD OR (B) DELETE A LIBRARY IDENTIFIER IN THE SYSTEM CATALOG. THIS MESSAGE IS FOLLOWED BY EITHER LIB260E OR LIB350E TO FURTHER EXPLAIN THE ERROR.

SYSTEM RESPONSE: THE SAVE OR DROP FUNCTION CONTINUES.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

A.) LIB260E CREATION OF CATALOGED LIBRARY IDENTIFIER FAILURE - CATLG RC XX, SECONDARY RC XX.

-OR-

B.) LIB260E DELETION OF CATALOGED LIBRARY IDENTIFIER FAILURE - CATLG RC XX, SECONDARY RC XX.

ENVIRONMENT: DROP OR SAVE.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S CATALOG MANAGEMENT FUNCTION HAS FAILED WITH THE INDICATED ERROR CODES DURING AN ATTEMPT TO EITHER (A) ADD OR (B) DELETE A LIBRARY IDENTIFIER IN THE SYSTEM CATALOG. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: THE SAVE OR DROP COMMAND CONTINUES.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB270I DATA SET NOT FOUND.

ENVIRONMENT: LOAD, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: A CATALOG ENTRY FOR THE WORKSPACE DATA SET WAS NOT FOUND; THEREFORE, PRESUMEDLY THE WORKSPACE DOES NOT EXIST.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: VERIFY THAT YOU ARE USING THE CORRECT LIBRARY NUMBER AND WORKSPACE NAME.

LIB280I DATA SET NAMING CONFLICT.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE SYSTEM CATALOG CONTAINS A CONFLICTING DATA SET NAME STRUCTURE SO THAT IT WOULD BE IMPOSSIBLE FOR THE NAME OF THE REQUESTED WORKSPACE ALSO TO BE CATALOGED.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: IF APPROPRIATE, CHOOSE A DIFFERENT WORKSPACE NAME AND/OR LIBRARY NUMBER. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB290I INVALID WORKSPACE NAME.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: VSAPL/TSO WORKSPACE NAMES CANNOT CONTAIN SPECIAL CHARACTERS, OR UNDERSTRIKES, OR BE LONGER THAN 8 CHARACTERS.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: CHOOSE A VALID WORKSPACE NAME.

LIB310I THE WORKSPACE DATA SET IS CURRENTLY IN USE BY SOMEONE ELSE. TRY AGAIN LATER.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: EITHER ANOTHER TSO USER OR A BATCH PROGRAM IS CURRENTLY USING THE WORKSPACE DATA SET. YOU MUST WAIT UNTIL THE DATA SET IS RELEASED BY THAT USER OR BATCH PROGRAM BEFORE YOU WILL BE ALLOWED ACCESS TO IT.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: WAIT A WHILE AND THEN TRY THE COMMAND AGAIN.

LIB320I THE WORKSPACE DATA SET RESIDES ON A CURRENTLY UNAVAILABLE VOLUME.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: THE DIRECT ACCESS STORAGE VOLUME (DRUM OR DISK) CONTAINING THE REQUIRED WORKSPACE DATA SET IS NOT CURRENTLY AVAILABLE TO THE SYSTEM.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: ARRANGE WITH YOUR INSTALLATION'S OPERATIONS STAFF FOR THE VOLUME TO BE MOUNTED, AND THE REISSUE THE COMMAND.

LIB330E THERE IS INSUFFICIENT DIRECT ACCESS STORAGE SPACE AVAILABLE FOR SAVING THIS WORKSPACE.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: ALL DIRECT ACCESS VOLUMES ONTO WHICH THE WORKSPACE MIGHT BE ALLOCATED ARE ALREADY FILLED TO CAPACITY WITH OTHER DATA SETS.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S OPERATIONS STAFF SO THAT THEY MIGHT TAKE STEPS TO CORRECT THE SITUATION.

LIB340E THE WORKSPACE DATA SET IS CATALOGED BUT NON-EXISTANT.

ENVIRONMENT: LOAD, SAVE, COPY, OR PCOPY.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM CATALOG CONTAINS INCORRECT INFORMATION. IT INDICATES THAT THE DATA SET EXISTS ON A PARTICULAR DIRECT ACCESS VOLUME WHEN IN FACT IT DOES NOT.

SYSTEM RESPONSE: THE COMMAND IS NOT EXECUTED.

USER RESPONSE: IF THE DATA SET HAS BEEN SCRATCHED WITHOUT BEING UNCATALOGED, THEN USE THE DROP COMMAND TO ALSO UNCATALOG THE DATA SET. IF, HOWEVER, THE DATA SET HAS BEEN CATALOGED INCORRECTLY, THEN YOU MUST CORRECT THE CATALOG INFORMATION INDEPENDENTLY OF VSAPL.

LIB350E DYNAMIC ALLOCATION FAILURE - FUNCTION CODE XX, DAIR RC XX, CATLG RC XX, DYNAM RC XXXX.

ENVIRONMENT: LOAD, SAVE, COPY, PCOPY, OR DROP.

PROBABLE CAUSE: USER ERROR OR SYSTEM ERROR.

EXPLANATION: WORKSPACE DATA SETS ARE MADE ACCESSABLE TO VSAPL VIA THE SYSTEM'S DYNAMIC ALLOCATION ROUTINES. THESE ROUTINES MAY FAIL IN ANY PARTICULAR INSTANCE FOR ANY OF A LARGE NUMBER OF REASONS. THE MORE COMMON CAUSES OF FAILURE ARE DOCUMENTED BY OTHER MORE DESCRIPTIVE MESSAGES. THIS MESSAGE IS USED FOR ALL OF THE OTHER LESS COMMON CAUSES.

- 'FUNCTION CODE' INDICATES THE FUNCTION THAT VSAPL HAS

REQUESTED OF DAIR ('DYNAMIC ALLOCATION INTERFACE ROUTINE').

- 'DAIR RC' IS THE ERROR CODE RETURNED FROM DAIR.

- 'CATLG RC' IS THE ERROR CODE RETURNED TO DAIR FROM THE SYSTEM'S CATALOG MANAGEMENT ROUTINES.

- 'DYNAM RC' IS THE ERROR CODE RETURNED TO DAIR FROM THE SYSTEM'S DYNAMIC ALLOCATION ROUTINES.

NOTE, ALL ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

WHETHER OR NOT THE ERROR OCCURRED DURING DATA SET ALLOCATION OR DEALLOCATION.

USER RESPONSE: IF APPROPRIATE, BRING THE ERROR TO THE ATTENTION OF YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB370I PASSWORD PROTECTION NOT AVAILABLE IN THIS SYSTEM.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: USER ERROR.

EXPLANATION: YOUR INSTALLATION HAS NOT PROVIDED A 'PASSWORD' DATA SET; THEREFORE, USAGE OF PASSWORDS TO LOCK WORKSPACES CANNOT BE SUPPORTED.

SYSTEM RESPONSE: THE SAVE FUNCTION CONTINUES.

USER RESPONSE: NONE.

LIB380E PASSWORD PROTECTION FAILURE - CODE XX.

ENVIRONMENT: SAVE.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S PASSWORD PROTECTION FUNCTION HAS FAILED WITH THE INDICATED ERROR CODE. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: THE SAVE FUNCTION CONTINUES.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB410W ERROR OCCURRED DURING ATTEMPT TO DROP THE CONTINUE WORKSPACE

ENVIRONMENT: DROP OR OFF.

PROBABLE CAUSE: REFER TO OTHER MESSAGES DISPLAYED ALONG WITH THIS MESSAGE.

EXPLANATION: THIS MESSAGE IS DISPLAYED WHENEVER ANY ERROR OCCURS WHILE DROPPING THE CONTINUE WORKSPACE (WHICH IS DONE AUTOMATICALLY BY THE OFF COMMAND). THIS MESSAGE IS DISPLAYED TO REDUCE POSSIBLE CONFUSION SHOULD SUCH AN ERROR OCCUR DURING THE OFF COMMAND.

SYSTEM RESPONSE: REFER TO OTHER MESSAGES DISPLAYED ALONG WITH THIS MESSAGE.

USER RESPONSE: REFER TO OTHER MESSAGES DISPLAYED ALONG WITH THIS MESSAGE.

LIB420E PASSWORD DELETION FAILURE - CODE XX

ENVIRONMENT: DROP.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S PASSWORD PROTECTION FUNCTION HAS FAILED WITH THE INDICATED ERROR CODE. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: DROP PROCESSING CONTINUES BUT WILL PROBABLY FAIL.

USER RESPONSE: NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB430E SCRATCH FAILURE - CODE XX

ENVIRONMENT: DROP.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S DATA SET SCRATCH FUNCTION HAS FAILED WITH THE INDICATED ERROR CODE. NOTE, ERROR CODES ARE DISPLAYED

IN HEXADECIMAL.

SYSTEM RESPONSE: DROP PROCESSING TERMINATES.

USER RESPONSE: RE-ATTEMPT THE DROP. IF THE ERROR PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEM'S PROGRAMMING GROUP.

A.) LIB431E SCRATCH FAILURE - VOLUME XXXXXX, INCORRECT PASSWORD

-OR-

B.) LIB431E SCRATCH FAILURE - VOLUME XXXXXX, CODE 08-XX.

ENVIRONMENT: DROP.

PROBABLE CAUSE: A.) USER ERROR; B.) SYSTEM ERROR.

EXPLANATION: A.) PROBABLY AN ATTEMPT HAS BEEN MADE TO SCRATCH A PASSWORD PROTECTED WORKSPACE WITHOUT SUPPLYING THE PASSWORD.

B.) THE SYSTEM'S SCRATCH FUNCTION HAS FAILED ON THE INDICATED VOLUME WITH THE INDICATED ERROR CODE. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: DROP PROCESSING IS TERMINATED.

USER RESPONSE: A.) SUPPLY THE CORRECT PASSWORD@ B.) RE-ATTEMPT THE SCRATCH. IF THE FAILURE PERSISTS, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

LIB440E UNCATALOG FAILURE - CODE XX.

ENVIRONMENT: DROP.

PROBABLE CAUSE: SYSTEM ERROR.

EXPLANATION: THE SYSTEM'S UNCATALOG FUNCTION HAS FAILED WITH THE INDICATED ERROR CODE. NOTE, ERROR CODES ARE DISPLAYED IN HEXADECIMAL.

SYSTEM RESPONSE: SINCE THE WORKSPACE HAS BY THIS TIME ALREADY BEEN SCRATCHED, DROP PROCEEDS AS IF NO ERROR HAD OCCURRED.

USER RESPONSE: IF THE ERROR IS PERSISTENT, THEN NOTIFY YOUR INSTALLATION'S SYSTEMS PROGRAMMING GROUP.

Auxiliary Processor Messages

The following message is reported when an auxiliary processor distributed with VS APL receives an unexpected return or reason code as the result of a shared variable operation:

id ABENDED AT address, RETURN CODE IS $code_1$, *REASON CODE IS* $code_2$

where:

id
identifies the auxiliary processor that encountered the unexpected code (AP100, AP101, and AP111 for auxiliary processor numbers 100, 101, and 111 respectively.)

address
is the address at which the unexpected return or reason code was discovered.

$code_1$
is the return code as described in the macro ASUSHSUP.

$code_2$
is the reason code as described in the macro ASUSHSUP.

In response to the unexpected code, VS APL processing is terminated. You should immediately communicate the displayed information to your system administrator.

System Errors

A system error is reported any time during your terminal session an unrecoverable internal error is encountered by VS APL. The format of a system error report is:

time date SYSTEM ERROR code

where *code* indicates the reason for the error. The code displayed will have no specific meaning to you but is useful if communication with your system administrator is necessary.

GLOSSARY

This glossary defines words and phrases used in the text. The definitions are primarily in terms of VS APL and its use; the definitions are not intended to apply generally to all of data processing.

auxiliary processor: A program that communicates with another program through shared variables.

clear workspace: A workspace in which no data has been entered, no names defined, and in which the workspace attributes indicate standard initial values.

communication command: A system command used to communicate information to other terminals.

continued workspace: A workspace that is stored in a library via the)CONTINUE or)CONTINUE HOLD commands or as a result of abnormal termination. A continued workspace has the name CONTINUE.

A CONTINUE workspace is DROPPED by the)OFF or)OFF HOLD command, and may not be explicitly SAVEd by the user.

control variable: A shared variable used in auxiliary processing operations to monitor and in some instances control data transmission.

distributed workspace: A public library workspace distributed with VS APL. Generally distributed workspaces are installed as part of libraries 1 or 2.

dynamic storage area: An area of the workspace in which data, function definitions, and group definitions are kept.

error report: A message produced by VS APL when it encounters an unexecutable statement.

execution control area: An area of the workspace used by VS APL to contain information generated during function execution and to contain the state indicator.

executor message: A message produced by the executor portion of VS APL. Executor messages are caused by a variety of error conditions such as erroneous library designations or initialization errors.

forced ending: An abrupt break of your link to the computer caused by machine malfunctions such as failures in the telephone circuits.

inquiry command: A system command that returns information about the contents and attributes of the active workspace.

library control command: A system command used to control the contents of libraries.

logoff: The procedure by which you end contact with TSO.

logon: The procedure by which you establish contact with TSO.

message: A communication sent by the system in response to an entry, or an indication of an encountered or impending problem. A line of text sent from one terminal to another. (see also *error report, trouble report,* and *executor message*)

message blocking: The act of halting incoming terminal messages. You can instruct the system to block messages through the system command)MSG OFF.

OS Sequential Dataset Processor: An auxiliary processor distributed with VS APL and the YALE IUP that can be used to read from or write to any OS Sequential Dataset supported by QSAM.

print suppression: A feature of certain terminals that suppresses the printing of sensitive information such as your logon password.

private library: A set of stored workspaces available to a specific VS APL user.

private/shared library: A set of stored workspaces that can be shared among VS APL users, but controlled by a single user.

public library: A set of stored workspaces available to all VS APL users, but controlled by the Systems Administrator.

record variable: A shared variable used in auxiliary processor operations to transmit data.

return code: A value placed in the record or control variable that indicates the acceptance or rejection of an auxiliary processing request.

sign-off: The procedure by which you end contact with VS APL.

sign-off command: A system command used to sign off the system.

sign-on: The procedure by which you establish contact with VS APL.

Stack Input Processor: An auxiliary processor distributed with VS APL that can be used to create a stack of data for subsequent input to VS APL.

state indicator: An object in the workspace that tracks the progress of executing defined functions. If function execution is halted, the state indicator shows all the halted functions in order, the most recently active function first, and the line in each function that is in the process of being, or the next to be, executed.

symbol table: An area of the workspace used by VS APL to keep track of names.

system administrator: The person who is responsible for the system and who assists you in your use of the system.

system command: (see *VS APL system command*)

system error: An internal error condition detected by VS APL.

system error report: A message indicating that a system error has been encountered by VS APL. The message indicates the time and date the error was encountered and a code representing the nature of the error.

trouble report: A message produced by VS APL when it encounters an unexecutable statement.

TSO Command Processor: An auxiliary processor distributed with VS APL that can be used to dynamically enter a TSO command.

TSO region: A region of the IBM Operating System used by the Time Sharing Option to swap multiple concurrent user programs in and out of main storage.

TSO Operator: The TSO user who is designated to receive TSO operator messages.

VS APL system command: An instruction to VS APL that allows you to monitor and control the contents of workspaces and libraries.

work session: The time elapsed between your TSO logon and logoff.

workspace control command: A system command used to control the content and attributes of the active workspace.

workspace identification: The library number and workspace name that uniquely identify a workspace.

SH20-1872-0

VS APL for TSO: Yale University Terminal User's Guide Printed in U.S.A. SH20-1872-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

READER'S COMMENT FORM

VS APL for TSO:

Yale University

Terminal User's Guide

SH20-1872-0

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessay if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Industry – Dept. 825

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

VS APL for TSO: Yale University Terminal User's Guide Printed in U.S.A. SH20-1872-0