

Serial Number Line #: 28  
1983 LUV/Filename : 18/L.TIPD83.9291

Printed on/at : FEB 7, 1990 14:50:51  
For Group/User: O , 1  
On Port No: 5

Print control parameters :	
Printer Class code	0
Form Code/paper type	?
Print Priority (0-9)	5
Starting Page Number	1
This is copy number	1
Keep File (Y/N)	Y
Notify User when done	N
Comments, optional	For RELSE CNTRL

Spool Queue Line # : 28  
PRIS LU/Filename : 1B/L.TIFD83.9291

Printed on/at : FEB 7, 1990 14:51:00  
For Group/User: 0 , 1  
On Port No: 5

Print control parameters :  
Printer Class code : 0  
Form Code/paper type : ?  
Print Priority (0-7) : 5  
Starting Page Number : 1  
This is copy number : 1  
Keep File (Y/N) : Y  
Notify User when done: N  
Comments, optional : For RELSE CNTRL

EOT : "TIPOG" (DISCSUBS GROUP 5) FOR IRIS R9. xx  
EOT : "DSUBDEFS" FOR IRIS  
END

ASM 16/A, TIPDB3, 92911, @18/L, TIPDB3, 92911, B050, -B051, B052  
FEB 7, 1990 11:40:43

Batch File: R95JCL.TIPDB3

A = 9291

-R95DEFSPZ  
-R95DSUBDEFSD  
R92TIPDSB60SA

. EOT : "TIPB3" (DISCSUBS GROUP 5) FOR IRIS R9. xx

<< BT = R92TIPDSB8GSA; BO = 18/A. TIPD83. 9291! >>  
DSUB83, R?>

R9 TYPIST Discsub, written by JPMH, 27-Jly-82

This discsub provides the following functions:-

- 0) It returns its own revision number
  - 1) It permits the insertion of one string into another
  - 2) It provides the PEEK function
  - 3) It provides the POKE function
  - 4) It provides the SOFT function

The PEEK and POKE functions require that an extra parameter, the key, be provided. This parameter is checked against the sum of the PLC and the UVS thus ensuring that the program is unmodified.

To determine the value required as the key, word 57720 should be changed from SUBZ 1,1 to JMP .+1, thus causing an error to be set when the key is wrong. In such a case the correct key will be returned in the parameter A.

Calling sequence:

CALL 83, MODE, <params list>

```
MODE=0, <params list>=returned revision number (numeric)
MODE=1, <params list>=string to insert into, insert string
                  insertion position (numeric)
MODE=2, <params list>=address, value, key
MODE=3, <params list>=address, new value, key
MODE=4, <params list>=dummy, code (string), key
```

REVIS. # 001 ; first released revision, 29-Jul-82

```
1 .TXTM 1 ; set the text mode  
104000 .LDC LTPO3 ; the address in DISCSU
```

104000 162 DSB83: TIPO3 ; it is actuall number 10  
104001 3 START-DSB83 ; address of start  
104002 177422 DSB83-DSBEND ; size

```

104003 54577 START: STA 3, RET ; the return address
104004 50577 STA 2, ACC2 ; will be required by SOFT
104005 44577 STA 1, ACC1 ; as above
104006 40577 STA 0, ACC0 ; as above
104007 50540 STA 2, APT ; the pointer to the arguments
104010 44222 JSR PICK ; go and get the first (MODE)
104011 44007 JSR VECEND ; jump over the vector table, etc

```

```

)
; VECTOR TABLE FOR VARIOUS MODES ;
104012    35 VECTOR: REVNO-VECTOR           ;relative position for mode=0
104013    56 STINS-VECTOR                 ;relative pos of STINS routine
104014    251 PEEK-VECTOR                  ;relative position of PEEK
104015    251 POKE-VECTOR                  ;relative position of POKE
104016    251 SOFT-VECTOR                  ;relative position of SOFT

)
; VSIZE-VECTOR-1 ;
104017    4 VSIZE: VSIZE-VECTOR-1          ;maximum switch value in vector table

104020 101112 VECEND: MOVL# 0,0,SZC      ;??? is the switch negative
104021 2561 JMP @RET                     ;YES, error return
104022 24775 LDA 1,VSIZE                 ;NO, now check for maximum size
104023 106433 SLE 0,1                   ;??? is switch in range
104024 2558 JMP @RET                     ;NO, error return
104025 171000 MOV 3,2                   ;A3 has address of table because of JSR
104026 117020 ADDZ 0,3                 ;add value of switch, so A3 points a vector
104027 35400 LDA 3,0,3                 ;get the vector value (pointer), rel address
104030 173000 ADD 3,2                   ;convert to an absolute address
104031 1000 JMP 0,2                   ;jumpt to routine

104032 PICK: ;
; return value of next numeric parameter in A0 ;
; on entry A2 points to argeument pointer ;
;

104032 25001 LDA 1,1,2                 ;number type of parameter
104033 125112 MOVL# 1,1,SZC          ;??? is it a numeric
104034 2546 JMP @RET                 ;NO, error return
104035 54411 STA 3,RETS              ;the return address for this routine
104036 31000 LDA 2,0,2                 ;address of the parameter
104037 102520 SUBZL 0,0               ;generate AO=1 for DECIMAL LOAD
104040 6120 DECIMAL                ;get the decimal value passed
104041 6121 FIX                      ;Ai gets integer part of DA
104042 2540 JMP @RET                 ;fix failed, error return
104043 121000 MOV 1,0                  ;put parameter in AO
104044 2402 JMP @RETS                ;return from subroutine

)
; RNUM: REVIS ;
104045    5 RNUM: REVIS                 ;the revision number
104046    0 RETS: 0                    ;return address for routines

104047 REVNO: ;
; return the revision number of this discsub ;

```

104047	102420	SUBZ	0, 0	; no error yet	
104050	40577	LDA	0, ERRFLG	; so that we can make appropriate return	
104051	24774	LDA	1, RNUM	; get the revision number	
104052	102420	RETRNUM:	SUBZ	0, 0	; sign of the returned value is +ve for FLOAT
104053	6122	FLOAT		; convert it to decimal in DA	
104054	30473	LDA	2, APT	; the argument pointer	
104055	25003	LDA	1, 3, 2	; the number type of second parameter	
104056	125112	MOVL#	1, 1, SZC	??? is it numeric	
104057	2523	JMP	@RET	; NO, error return	
104060	31002	LDA	2, 2, 2	; get the address of 2nd parameter	
104061	102420	SUBZ	0, 0	; tell DECIMAL to do a store	
104062	6120	DECIMAL		; store the value from DA into argument	
104063	20564	LDA	0, ERRFLG	; get the error flag	
104064	101014	MOV#	0, 0, SZR	??? was it set	
104065	2515	JMP	@RET	; YES, error return	
104066	34514	EXIT:	LDA	3, RET	; the return address
104067	1401	JMP	1, 3	; a non error return	

104070 STINS: ;  
; ; This routine will insert into the first string, P1\$,  
; ; the string P2\$ at position P3.  
;

104070	30457	LDA	2, APT	; the pointer to the arguments	2
104071	24002	LDA	1, C2	; the constant 2	1
104072	133020	ADDZ	1, 2	; step past the mode switch	
104073	50454	STA	2, APT	; update the storage	
104074	25000	LDA	1, O, 2	; address of P1\$	1
104075	44557	STA	1, P1ADD	; store it for future	
104076	25001	LDA	1, 1, 2	; the dimension and type of P1\$	1
104077	125123	MOVZL	1, 1, SNC	; ??? is it a string and move out msb	1
104100	2502	JMP	@RET	; NO, error	
104101	125220	MOVZR	1, 1	; YES, fix the size after the shift	1
104102	44560	STA	1, P1DIM	; note the dimension	
104103	151400	INC	2, 2	; step to P2\$	
104104	151400	INC	2, 2		2
104105	50442	STA	2, APT	; update the stored version	
104106	25000	LDA	1, O, 2	; get the address	1
104107	44547	STA	1, P2ADD	; store the address	
104110	25001	LDA	1, 1, 2	; the size and dimension of P2\$	1
104111	125133	MOVZL#	1, 1, SNC	; ??? is it a string	
104112	2470	JMP	@RET	; NO, error return	
104113	151400	INC	2, 2	; set A2 to point at P3	
104114	151400	INC	2, 2		2
104115	4715	JSR	PICK		0???
104116	126520	SUBZL	1, 1	; YES, get P3, the insert position	
		,		; make a one to reduce pointer since	1
				; we wish to insert before specified	
				; reduce the value as required	
				; where is the insertion	0
				; make a zero	1
104117	122420	SUBZ	1, O		
104120	40541	STA	O, INSPOS		
104121	126420	SUBZ	1, 1		

104122	109032	SGE	0, 1	; ??? is the pointer greater than 0	
104123	2457	JMP	@RET	; NO, error return	1
104124	24536	LDA	1, P1DIM	; get the size of the first param	
104125	106433	SLE	0, 1	; ??? is position greater than size	
104126	2454	JMP	@RET	; YES, error return	
104127	20041	LDA	0, DBA	; used in converting relative/absolute	0
104130	24524	LDA	1, P1ADD	; the address of the destination str	1
104131	106540	SUBOL	0, 1	; convert to a relative address so	1
			;	; that it may be used by XGETBYTE	
104132	44522	STA	1, P1ADD	; update the stored version	
104133	24523	LDA	1, P2ADD	; the address of the new string	1
104134	106540	SUBOL	0, 1	; convert to a relative byte address	1
104135	44521	STA	1, P2ADD	; store it away	
104136	102420	SUBZ	0, 0	; generate a zero	0
104137	40516	STA	0, P1LEN	; clear the length counter of P1\$	
104140	24514	LDA	1, P1ADD	; get the address of P1\$, we are going	1
			;	; to look for the length	
104141	6144 P1LUP:	XGETBYTE		; get the next character	2
104142	151015	SNZ	2, 2	; ??? have we found the terminator	
104143	405	JMP	FNDP1	; YES, we have found and of P1\$	
104144	10511	ISZ	P1LEN	; there is another character	
104145	125400	INC	1, 1	; set up for XGETBYTE	1
104146	773	JMP	P1LUP	; go and look for the next	
104147	0 APT:	O		; argument pointer from RUN	
104150	24505 FNDP1:	LDA	1, P1LEN	; the calculated lenght of P1\$	1
104151	30510	LDA	2, INSPoS	; the insertion position	2
104152	148433	SLE	2, 1	; ??? is insertion in real part	
104153	2427	JMP	@RET	; NO, error returned	
104154	102420	SUBZ	0, 0	; clear counter for lenght of P2\$	0
104155	40502	STA	0, P2LEN	; we are going to find len of P2\$	
104156	24500	LDA	1, P2ADD	; address of P2\$	1
104157	6144 P2LUP:	XGETBYTE		; get the next character	2
104160	151015	SNZ	2, 2	; ??? are we at the null yet	
104161	404	JMP	FNDP2	; YES, we have found the end of P2\$	
104162	10475	ISZ	P2LEN	; No, count the extra character	
104163	125400	INC	1, 1	; step the pointer	1
104164	773	JMP	P2LUP	; deal with the next position	
104165	30472 FNDP2:	LDA	2, P2LEN	; the calculated lenght of P2\$	2
104166	151015	SNZ	2, 2	; ??? is it null string	
104167	677	JMP	EXIT	; YES, we are finished	
104170	24465	LDA	1, P1LEN	; the length of the original	1
104171	147020	ADDZ	2, 1	; the new lenght if performed	1
104172	30470	LDA	2, P1DIM	; the maximum permitted	2
104173	132433	SLE	1, 2	; ??? will the new string fit	
104174	2406	JMP	@RET	; NO, error return	
104175	24457	LDA	1, P1ADD	; get the address of the first	1
104176	20463	LDA	0, INSPoS	; where to perform the insertion	0
104177	123020	ADDZ	1, 0	; the address in the string of the ins	0
104200	30455	LDA	2, P1LEN	; the length of the inserted section	2
104201	405	JMP	BRIDG	; skip over the central data	
104202	0 RET:	O		; initial values of accumulators	
104203	0 ACC2:	O			
104204	0 ACC1:	O			
104205	0 ACC0:	O			

104206	147020	BRIDG:	ADDZ	2, 1			1
104207	30450		LDA	2, P2LEN	; end of old P1\$		2
104210	113020		ADDZ	0, 2	; length of the new section		2
			,		; A2 gets new position of the second		
					segment of the first string		
104211	4414		JSR	MOVE	; perform the move to make space for	????	
104212	20444		LDA	0, P2ADD	; P2\$ in P1\$		0
104213	152000		ADC	2, 2	; where is P2\$		2
104214	24443		LDA	1, P2LEN	; generate -1		
104215	147020		ADDZ	2, 1	; the length of the second		1
104216	107020		ADDZ	0, 1	; make lenght -1		1
104217	30435		LDA	2, P1ADD	; position of end of P2\$		2
104220	34441		LDA	3, INSPOS	; address of the original string		3
104221	173020		ADDZ	3, 2	; where in P1\$ is the insertion		
104222	4403		JSR	MOVE	; set A2 as the actual position		2
104223	643		JMP	EXIT	; move P2\$ into place		????
					; we are finished		
104224	606	JPICK:	JMP	PICK	; stepping stone		
104225	MOVE:			;;;;;;;	AO = where to copy to	0	
				;;;;;;;	AI = old end position (last char)	1	
				;;;;;;;	A2 = new end position (firt char)	2	
104225	40425		STA	0, BEGS			
104226	44425		STA	1, ENDS	; note the source begining		
104227	54617		STA	3, RETS	; note the end of the source		
104230	106400		SUB	0, 1	; note the return address		
104231	133020		ADDZ	1, 2	; the number to move		1
104232	50426		STA	2, DESTC	; where to start in destination		2
104233	24420	MOVLPUP:	LDA	1, ENDS	; note where we will go to		
104234	6144		XGETBYTE		; the end of the source		1
104235	141000		MOV	2, 0	; get the character from the source		2
104236	24422		LDA	1, DESTC	; move the character		0
104237	6145		XPUTBYTE		; current detination character		1
104240	20413		LDA	0, ENDS	; put the character in its new position		
104241	24411		LDA	1, BEGS	; the end of the source		0
104242	106415		SNE	0, 1	; beginning of destination		1
104243	2603		JMP	@RETS	; ??? are they equal, have we copied all		
104244	14414		DSZ	DESTC	; YES, return		
104245	14406		DSZ	ENDS	; ready for next character		
104246	765		JMP	MOVLPUP	; we have moved one more		
					; dela with the next		
104247	0	ERRFLG:	0		; O-no error, 1-error in KEY		
104250	602	JRETNUM:	JMP	RETNUM	; stepping stone to RETNUM		
104251	615	JEXIT:	JMP	EXIT	; stepping stone to exit		
104252	0	BEGS:	0				
104253	0	ENDS:	0		; beginning od source for move		
104254	0	P1ADD:	0		; end of source for move		
104255	0	P1LEN:	0		; address of P1\$		
104256	0	P2ADD:	0		; length of P1\$		
104257	0	P2LEN:	0		; address of P2\$		
104260	0	DESTC:	0		; len of P2\$		
					; pointer to next destination char		

i04261 O INSPOS: 0 ;position to insert at  
104262 O P1DIM: 0 ;dimensioned size of P1\$

104263 PEEK: ;  
; Returns in the third parameter the contents of the ;  
; memory location specified by the second. The third ;  
; parameter is used to specify the key to permit the ;  
; operation.  
;

```

104263 POKE: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; As PEEK except that the third parameter is written
; to rather than read from memory.
; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

104263 SOFT: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Execute the third parameter (string).
; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

104263 24002 LDA 1,C2 ;displacement of mode values 1
104264 122420 SUBZ 1,0 ;make Peek=0, Poke=1, Soft=2 0
104265 40467 STA 0,SWTCH ;the mode we are operating in
104266 4415 JSR CHKKEY ;see if the keys are valid
;arrive back only if so
;increment to point at 2nd param

104267 10660 ISZ APT
104270 10657 ISZ APT
104271 30656 LDA 2,APT ;where are the arguements 2
104272 4732 JSR JPICK ;get the address to peek/poke 0???
104273 111000 MOV 0,2 ;the address specified 2
104274 102420 SUBZ 0,0 ;prepare to clear error flag 0
104275 40752 STA 0,ERRFLG ;clear it
104276 20456 LDA 0,SWTCH ;get the peek/poke flag 0
104277 101034 MOVZ# 0,0,SZR ;??? is this a peek
104300 423 JMP POKE1 ;NO, so go and poke or soft
104301 25000 LDA 1,0,2 ;get the contents
104302 746 JMP JRETNUM ;return the value 1

104303 CHKKEY: ;;;;;;;;;;;;;;;;;;; Check that the entered key is valid
; if not, return correct key in 2nd
; param and make error return.

104303 54450 STA 3,CHKRET ;the return address
104304 32004 LDA 2,@PIB ;the address of the pointer to part 2
104305 25002 LDA 1,2,2 ;the relative address of my PLC 1
104306 31006 LDA 2,6,2 ;the relative address of UVS 2
104307 147020 ADDZ 2,1 ;add them for checking 1
104310 30637 LDA 2,APT ;pointer to arguments 2
104311 20026 LDA 0,C6 ;generate a constant 6 0
104312 113020 ADDZ 0,2 ;to point at fourth param 2
104313 44745 STA 1,DESTC ;remember the value
104314 4710 JSR JPICK ;get the fourth parameter
104315 24743 LDA 1,DESTC ;restore the actual UVS+PLC 1
104316 122415 SNE 1,0 ;??? is the key correct
104317 2434 JMP @CHKRET ;YES, so return
104320 126420 SUBZ 1,1 ;NO, make a zero
104321 44726 STA 1,ERRFLG ;so set the error flag
104322 726 JMP JRETNUM ;NO so return the required value
;in the second parameter

104323 14431 POKE1: DSZ SWTCH ;??? is the switch=1 (poke)
104324 411 JMP SOFT1 ;NO, execute the soft function

```

104325	50430	STA	2, PADDR	; the address to poke	
104326	30621	LDA	2, APT	; get the address of 2nd parameter	2
104327	151400	INC	2, 2	; step to the third, the value	2
104330	151402	INC	2, 2		2
104331	4673	JSR	JPICK	; get the value	0???
104332	30423	LDA	2, PADDR	; get the address to poke	2
104333	41000	STA	0, 0, 2	; put the value into core	
104334	715	JMP	JEXIT	; we are finished	
104335	30612	SOFT1:	LDA	2, APT	
104336	145000	MOV	2, 1	; pointer to the string to execute	2
104337	20002	LDA	0, C2	; ready for passing externally	1
104340	106420	SUBZ	0, 1	; constant for resetting APT	0
104341	21003	LDA	0, 3, 2	; A1 now points at original APTs	1
104342	34640	LDA	3, RET	; the number type of the string	0
104343	101133	MOVZL#	0, 0, SNC	; return address to RUN	3
104344	1400	JMP	0, 3	; ??? is it a string	
104345	31002	LDA	2, 2, 2	; NO, error non-skip return	2
104346	50407	STA	2, PADDR	; address of the string	
104347	30634	LDA	2, ACC2	; store it to jump through	
104350	24634	LDA	1, ACC1	; restore the original values of accs	2
104351	20634	LDA	0, ACC0		1
104352	2403	JMP	@PADDR		0
			;	; go and execute the routine with all	
			;	; ACCS set as though we came from RUN	
104353	0	CHKRET:	0	; return address for the routine CHKKEY	
104354	0	SWTCH:	0	; 0=PEEK: 1=POKE: 2=SOFT	
104355	0	PADDR:	0	; address to poke	

```

104356      DSBEND=.           ; this is the end
      0      . ERR    DSB83+400C. ; OVERFLOW CHECK
      . END

```

ACCO	104205	ACC1	104204	ACC2	104203	APT	104147	BEGS	104252
BINDI	6115	BINMU	6116	BPI	16	BRIDG	104206	BSACF	75
BUMPU	6117	C10	30	C100	51	C1000	67	C11	31
C12	32	C13	33	C14	34	C15	35	C16	36
C160	174	C163	175	C166	176	C17	37	C170K	21
C171	177	C177	52	C1777	70	C2	2	C20	42
C200	53	C2000	71	C205	54	C215	55	C240	56
C244	57	C260	60	C271	61	C3	3	C300	62
C334	63	C37	43	C377	64	C4	24	C40	44
C400	65	C4000	72	C5	25	C6	26	C600	100
C7	27	C77	50	C774C	22	C777	66	CALL	6101
CHANN	6106	CHKKE	104303	CHKRE	104353	CM400	23	DA	160
DAC	164	DAS	165	DATAP	6110	DB	166	DBA	41
DBC	172	DBS	173	DECIM	6120	DESTC	104260	DFTCA	34106
DMCAL	34110	DQUEU	6105	DSB83	104000	DSBEN	104356	ENDS	104253
ERRF	76	ERRFL	104247	ESCF	73	ETSF	74	EXIT	104066
FINDL	6123	FIX	6121	FLAGC	6102	FLOAT	6122	FNDP1	104150
FNDP2	104165	FREEN	6107	GETBY	6124	HALTS	6153	INBYT	6125
INSPO	104261	INSTB	6126	ILOCAL	34103	IOP	6	ISA2D	6127
ISA2L	6130	JEXIT	104251	JFLTO	151	JPICK	104224	JRETN	104250
LACNT	4000	LAFSE	13000	LALCO	47400	LALLO	1400	LATOE	36000
LBAKU	106000	LBILD	5000	LBUIL	4400	LCALL	75000	LCHAN	41000
LCHFL	30000	LCHSU	61000	LCLEA	7400	LCLOS	7000	LCLPY	76000
LCNVA	114000	LCNVD	12000	LCOMM	33400	LDALC	2000	LDALL	1000
LDB7A	114000	LDB7B	114400	LDB7C	115000	LDB7D	115400	LDB7E	116000
LDB7F	116400	LDB7G	117000	LDB7H	117400	LDB7I	120000	LDEKE	52400
LDELE	3400	LDIRE	50400	LDLTP	20400	LDREN	37400	LDSB1	400
LDSB2	22400	LDSB3	47000	LDSB4	65000	LDSB5	77000	LDSB6	106400
LDSB7	113400	LECHO	37000	LEO87	105400	LERO	23000	LFAUL	400
LEFFIL	2400	LFIIXD	57400	LFNDC	112000	LFNDL	20000	LFOFI	17000
LGETR	10000	LGHOP	107400	LGHOS	107000	LGMUX	16000	LHCON	17400
LIBCA	44400	LIBEN	45000	LIBTR	45400	LIDAT	103000	LLINK	35400
LLOAD	34400	LOGI	32000	LLUIN	112400	LMAPB	73000	LMDEO	65000
LMDF1	66000	LMDE5	71400	LMRC3	56400	LMRFH	57000	LMRFI	54000
LMTAP	55400	LMTAS	54400	LMTFP	56000	LMTFY	60400	LMTNX	55000
LMTPL	60000	LOADD	6131	LOPEN	6000	LOPNM	13400	LPATQ	110000
LPEXP	23400	LPFAB	72000	LPFLN	73400	LPFNA	3000	LPFRL	72400
LPPSE	67000	LPPSH	70000	LPFSX	70400	LPLOG	24400	LPPWR	33000
LPRAN	36400	LPRCO	71000	LPSIN	25400	LPSQR	22400	LPTAN	25000
LQIBF	63400	LQICL	63000	LQIOP	62400	LRDFH	26400	LRDIS	31400
LRDSE	110400	LREDC	50000	LREDI	11000	LREDM	14000	LREDP	74000
LRENA	15000	LREOP	53000	LRES0	42000	LRWIT	113000	LRWMB	14400
LRWSX	111400	LS105	77000	LS152	102000	LS153	101000	LS154	100400
LS156	101400	LS157	100000	LSAVE	43000	LSAVP	43400	LSEAB	64000
LSEAR	51000	LSETF	40000	LSHUF	52000	LSIGP	12400	LSING	40400
LSMC05	106400	LSPEC	27000	LSTRI	32400	LSYSC	30400	LTP01	102400
LTP03	104000	LTP04	104400	LTP05	105000	LVMUX	42400	LWRIT	47000
LXMIN	62000	MOVE	104225	MOVLU	104233	OUTBY	6132	OUTTE	6133
P1ADD	104254	P1DIM	104262	P1LEN	104255	P1LUP	104141	P2ADD	104256
P2LEN	104257	P2LUP	104157	PADDR	104355	PEEK	104263	PIB	4
PICK	104032	POKE	104263	POKE1	104323	PUTBY	6134	QCHAR	6103
QUEUE	6104	READB	6135	RELJM	6136	RET	104202	RETRNU	104052
RETS	104046	REVIS	5	REVNO	104047	RJSR	6136	RNUM	104045
RTP	7	RUP	5	SBA	40	SCDCA	34147	SOFT	104263
SOFT1	104335	SPINP	6146	START	104003	STINP	6140	STINS	104070
STINT	6147	STORD	6137	STOUT	6141	SWTCH	104354	TASKQ	15
TRAPP	6142	VECEN	104020	VECTO	104012	VSIZE	104017	WRITB	6143
XGETB	6144	XPUTB	6145	.ABA	14	.BPS	77	.BSA	10

---  
DA 174 DA3 175 DB 176 DB3 177 FLTO 152  
HBA 11 HXA 12 INFO 100 INTR 111 LCM 114  
NRET 112 SRET 113 SSA 13

Spool Queue Line #: 29  
IRIS LU/Filename : 18/X.TIPD83.9291

Printed on/at : FEB 7, 1990 14:52:51  
For Group/User: 0 , 1  
On Port No: 5

Print control parameters	:	
Printer Class code	:	0
Form Code/paper type	:	?
Print Priority (0-9)	:	5
Starting Page Number	:	1
This is copy number	:	1
Keep file (Y/N)	:	Y
Notify User when done	:	N
Comments, optional	:	For RELSE CNTRL

Spool Queue Line #: 29  
IRIS LU/Filename : 18/X.TIPD83.9291

Printed on/at : FEB 7, 1990 14:52:57  
For Group/User: 0 , 1  
On Port No: 5

Print control parameters :

Printer Class code	0
Form Code/paper type	?
Print Priority (0-9)	5
Starting Page Number	1
This is copy number	1
Keep file (Y/N)	Y
Notify User when done	N
Comments, optional	: For RELSE CNTRL



\*\*\*\*\* JOB STATISTICS \*\*\*\*\*

2 TOTAL # DUPLICATE KEYS  
0 TOTAL # DIR. RE-ORG'S  
198 TOTAL # KEYS INSERTED  
0 TOTAL # ASSEMBLY ERR'S

FEB 7, 1990 11:41

18/L TIPD83.9291!

PAGE 1

.ERR	3.098						
ACCO	2.057	2.239:	3.085				
ACC1	2.056	2.238:	3.084				
ACC2	2.055	2.237:	3.083				
APT	2.058	2.131	2.155	2.158	2.168	2.206:	3.025
	3.026	3.027	3.049	3.065	3.073		
BEGS	2.267	2.279	2.293:				
BRIDGE	2.234	2.243:					
C2	2.156	3.020	3.075				
C6	3.050						
CHKKE	3.023	3.040:					
CHKRE	3.044	3.056	3.090:				
DBA	2.187						
DECIM	2.106	2.137					
DESTC	2.272	2.276	2.282	2.299:	3.052	3.054	
DSBB83	2.050:	2.051	2.052	3.098			
DSBEN	2.052	3.096=					
ENDS	2.268	2.273	2.278	2.283	2.294:		
ERRFL	2.127	2.138	2.286:	3.031	3.058		
EXIT	2.141:	2.224	2.259	2.289			
FIX	2.107						
FLOAT	2.130						
FNDP1	2.201	2.208:					
FNDP2	2.217	2.222:					
INSPD	2.180	2.209	2.231	2.256	2.300:		
JEXIT	2.289:	3.071					

FEB 7, 1990 11:41

18/L TIPD83.9291!

PAGE 2

JPICK	2. 261:	3. 028	3. 053	3. 068			
JRETN	2. 288:	3. 036	3. 059				
LTP03	2. 048						
MOVE	2. 248	2. 258	2. 263:				
MOVLU	2. 273:	2. 284					
P1ADD	2. 160	2. 188	2. 191	2. 197	2. 230	2. 255	2. 295:
P1DIM	2. 165	2. 184	2. 227	2. 301:			
P1LEN	2. 196	2. 202	2. 208	2. 225	2. 233	2. 296:	
P1LUP	2. 199:	2. 204					
P2ADD	2. 170	2. 192	2. 194	2. 214	2. 250	2. 297:	
P2LEN	2. 213	2. 218	2. 222	2. 244	2. 252	2. 298:	
P2LUP	2. 215:	2. 220					
PADDR	3. 064	3. 069	3. 082	3. 086	3. 092:		
PEEK	2. 072	2. 306:					
PIB	3. 045						
PICK	2. 059	2. 093:	2. 176	2. 261			
POKE	2. 073	3. 006:					
POKEI	3. 034	3. 062:					
RET	2. 054	2. 083	2. 086	2. 102	2. 108	2. 134	2. 140
	2. 141	2. 163	2. 173	2. 183	2. 186	2. 211	2. 229
	2. 236:	3. 078					
RETRNU	2. 129:	2. 288					
RETS	2. 103	2. 110	2. 117:	2. 269	2. 281		
REVIS	2. 043=	2. 116					
REVND	2. 070	2. 120:					
RNUM	2. 116:	2. 128					
SOFT	2. 074	3. 013:					

FE8 7, 1990 11:41

18/L TIPD83. 9291!

PAGE 3

SOFT1	3. 063	3. 073:				
START	2. 051	2. 054:				
STINS	2. 071	2. 147:				
SWTCH	3. 022	3. 032	3. 062	3. 091:		
TIPO3	2. 050					
VECEN	2. 060	2. 082:				
VECTO	2. 070:	2. 071	2. 072	2. 073	2. 074	2. 079:
VSIZE	2. 079:	2. 084				
XGETB	2. 199	2. 215	2. 274			
XPUTB	2. 277					