
*
* FILE: 00IL3YS-FILE*XAMOS*MD*XAMOS.D*AMU.D*US0500.D*AMU_XAMOS.P
*
* DATE: 01:12:14
*
* TIME: 08:42:24
*

Kompatibel med PROM 2054-1 → 2056-1

XAMU

CR80 XAMOS ASSEMBLER VERSION 811110

ASSEMBLY OF FILE: AMU.M

AT 81:12:13 11:45:35

```
00000001 0 0000 LIST
00000002 0 0000 ;
00000003 0 0000 ;
00000004 0 0000 ; PROJECT: PDS
00000005 0 0000 ;
00000006 0 0000 ; MODULE NAME: AMOS MASTERCLEAR UTILITIES (AMU)
00000007 0 0000 ; MODULE ID NMB: PD/1000
00000008 0 0000 ; MODULE VERSION: 0400
00000009 0 0000 ; MODULE TYPE: SYSTEM
00000010 0 0000 ; MODULE FILES: REFER TO INFO.T
00000011 0 0000 ; PREFIX FILES: NONE
00000012 0 0000 ;
00000013 0 0000 ; SPECIFICATIONS: PDS/1000/USM/XXXX
00000014 0 0000 ; AUTHOR/DATE: FR/811104
00000015 0 0000 ;
00000016 0 0000 ; DELIVERABLE: YES
00000017 0 0000 ; SOURCE LANGUAGE: CR80 ASSEMBLER
00000018 0 0000 ; COMPILE COMPUTER: PDS DEVELOPMENT SYSTEM
00000019 0 0000 ; TARGET COMPUTER: CR80S
00000020 0 0000 ; OPER. SYSTEM: NONE
00000021 0 0000 ;
00000022 0 0000 ;
00000023 0 0000 ;
00000024 0 0000 ; CHANGE RECORD
00000025 0 0000 ;
00000026 0 0000 ; VERSION AUTHOR/DATE DESCRIPTION OF CHANGE
00000027 0 0000 ; -----
00000028 0 0000 ; 0400 FR/811104 INITIAL SUBMITTAL TO PDS-SCL
00000029 0 0000 ;
00000030 0 0000 ;
00000031 0 0000 ;PAGE
```



```

00000032 0 0000 ; AMOS MASTER CLEAR UTILITIES FR/811101
00000033 0 0000 ;
00000034 0 0000 ; IDENTIFICATION: CSS/395/PGM
00000035 0 0000 ; VERSION: 04
00000036 0 0000 ; DATED: 811101/FR
00000037 0 0000 ;
00000038 0 0000 ; THE AMOS_MASTERCLEAR_UTILITIES PROGRAM (AMU) RESIDING IN THE UPPER
00000039 0 0000 ; 12K WORDS PROM OF MEMORY SECTION 0 WILL SERVE THE USER OF THE CR80 WITH
00000040 0 0000 ; A RANGE OF UTILITIES. THIS AIM NOT ONLY AT PROGRAMMERS BUT ALSO AT
00000041 0 0000 ; MAINTENANCE PERSONNEL.
00000042 0 0000 ;
00000043 0 0000 ; THE START-UP SEQUENCE AFTER MASTER CLEAR IS AS FOLLOWS:
00000044 0 0000 ; *-----1----->2:-----3-----:4:-----5-----:MASTER
00000045 0 0000 ; *-----1----->2:-----3-----:4:6:LOOSER
00000046 0 0000 ; *-1->2:-----3-----:4:6:LOOSER
00000047 0 0000 ;
00000048 0 0000 ; * - MASTER CLEAR
00000049 0 0000 ; 1 - INITIALIZATION IN CPU (CPU/CACHE CPU TEST)
00000050 0 0000 ; 2 - COPY BUILT-IN (PROM-)PROCESS DESCRIPTOR TO NEW BASE
00000051 0 0000 ; 3 - WAIT SPECIFIED TIME = REG 2 CONTENT (DEFAULT 256-350 MSEC)
00000052 0 0000 ; 4 - RESS
00000053 0 0000 ; 5 - WINNER: WAIT TWO TIMES SPECIFIED TIME
00000054 0 0000 ; RE-ESTABLISH BASE/MOD-REGISTERS
00000055 0 0000 ; TAKE OWNERSHIP OF PROCESSOR UNIT.
00000056 0 0000 ; 6 - LOOSER: PREPARE DUMMY PROCESS
00000057 0 0000 ; AWAKE SLEEPING DUMMY PROCESS
00000058 0 0000 ; WAIT FOR CPU-INTERRUPT FROM WINNER.
00000059 0 0000 ;
00000060 0 0000 ; ONLY MEMORY LOCATIONS 200 THRU 223 INCL WILL BE DESTROYED UPON EITHER A
00000061 0 0000 ; MASTER CLEAR OR EMERGENCY ACTION. THIS APPLIES AS LONG AS THE USER
00000062 0 0000 ; RESTRICTS HIMSELF TO THE FOLLOWING UTILITIES:
00000063 0 0000 ; COPY, DUMP, ID_UTILITIES, MEMORY_CHECK, PATCH, SEARCH
00000064 0 0000 ; EXECUTE, SET_PARITY (,AND OPTIONS).
00000065 0 0000 ;
00000066 0 0000 ; HOWEVER THE AMU_PROGRAM WILL NEVER CLAIM BUT THE LOWER #247 WORDS OF
00000067 0 0000 ; MEMORY EXCEPT FOR THE CPU_TEST WHICH REQUIRES AN ADDITIONAL 4K RAM
00000068 0 0000 ; STARTING AT #1000.
00000069 0 0000 ; THIS IMPLIES THAT FD_BOOT_FILES CAN HAVE THEIR LOADING ADDRESS (START)
00000070 0 0000 ; FROM #0248 AND UPWARDS. (CROSSING PAGE_BOUNDARIES DURING FD_LOAD WILL
00000071 0 0000 ; RESULT IN A PAGE SHIFT AND CONTINUATION FROM #0000 THAT PAGE).
00000072 0 0000 ;
00000073 0 0000 ; DEFAULT DEVICE ADDRESSES:
00000074 0 0000 ; DC - 1
00000075 0 0000 ; SCM - 1
00000076 0 0000 ; FD - 2 (MAY BE OVERRIDED)
00000077 0 0000 ; DMA - 7 (X - INDIRECT LOAD/BOOT)
00000078 0 0000 ; FF

```

```

00000079 0 0000 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000080 0 0000 ; U S E R ' S   M A N U A L
00000081 0 0000 ; -----
00000082 0 0000 ;
00000083 0 0000 ; UTILITIES
00000084 0 0000 ; -----
00000085 0 0000 ;
00000086 0 0000 ;   BOOT_LOAD:
00000087 0 0000 ;       B ?F,X? @<DEVICE_ADDRESS>.A<DRIVE_NO>           (F= FD, X=DMA)
00000088 0 0000 ;   COPY_MEMORY:
00000089 0 0000 ;       C<PAGE>A   <SOURCE>@+<SIZE>A   <OBJECT>@/<PAGE>A
00000090 0 0000 ;               <SOURCE>/<OBJECT>:!=<BASE_ADDRESS>@.<OFFSET>A
00000091 0 0000 ;   DUMP:
00000092 0 0000 ;       D<PAGE>A   <BASE_ADDRESS>@.<OFFSET>A           <END_ADDRESS>
00000093 0 0000 ;       D<PAGE>A   <BASE_ADDRESS>@.<OFFSET>@+<SIZE>A
00000094 0 0000 ;       D<PAGE>A   <BASE_ADDRESS>@.<OFFSET>A*<RE-READS>
00000095 0 0000 ;   COPY_AND_EXECUTE:
00000096 0 0000 ;       E<PAGE>A <PROG_START_ADDRESS>   @<NEW_BASE>@/<NEW_PROG>AA
00000097 0 0000 ;   IO:
00000098 0 0000 ;       I ?S,R?   <MOD_BY>   <DEV_ADDRESS>@+<REPEATS>A
00000099 0 0000 ;       I ?C,W?   <MOD_BY>   <DEV_ADDRESS>@+<REPEATS>A   <OUTPUT_WORD>
00000100 0 0000 ;       I ?K,V?   <MOD_BY>   <DEV_ADDRESS>@+<REPEATS>A   <OUTPUT_WORD>...
00000101 0 0000 ;   LOAD_FILE:
00000102 0 0000 ;       L ?F,X? @<DEVICE_ADDRESS>.A<DRIVE_NO>           (F= FD, X=DMA)
00000103 0 0000 ;   MEMORY_CHECK:
00000104 0 0000 ;       M<PAGE>A <BASE_ADDRESS>@.<OFFSET>@+<SIZE>A   @<PATTERN>A
00000105 0 0000 ;   PATCH:
00000106 0 0000 ;       P<PAGE>A <BASE_ADDRESS>@.<OFFSET>A           @<PATCH_DATA>A
00000107 0 0000 ;   MULTIPATCH:
00000108 0 0000 ;       P<PAGE>A <BASE_ADDRESS>@.<OFFSET>@+<SIZE>A   <PATTERN>
00000109 0 0000 ;       P<PAGE>A <BASE_ADDRESS>@.<OFFSET>A*<RE-WRITES>   <PATTERN>
00000110 0 0000 ;   SWITCH_CPU:
00000111 0 0000 ;       Q<CPU_NO>
00000112 0 0000 ;   RETURN_CALLING_PROCESS:
00000113 0 0000 ;       R
00000114 0 0000 ;   SEARCH:
00000115 0 0000 ;       S<PAGE>A <BASE_ADDRESS>@.<OFFSET>@+<SIZE>A...
00000116 0 0000 ;               ...<PATTERN>@/<MASK_PATTERN>A
00000117 0 0000 ;   NEGATED_SEARCH:
00000118 0 0000 ;       S<PAGE>A <BASE_ADDRESS>@.<OFFSET>@+<SIZE>A   <PATTERN>/
00000119 0 0000 ;   TEST_CPU:
00000120 0 0000 ;       T<PAGE>A <CPU_PROM_ADDRESS>
00000121 0 0000 ;   UNIT_MAPPER:
00000122 0 0000 ;       U
00000123 0 0000 ;   WAIT_INTERRUPT:
00000124 0 0000 ;       W @<DEVICE_ADDRESS>A
00000125 0 0000 ;   XECUTE:
00000126 0 0000 ;       X <PROCESS_BASE_ADDRESS>
00000127 0 0000 ;   SET_PARITY:
00000128 0 0000 ;       Z

```

```

00000127 0 0000 ;
00000128 0 0000 ;
00000129 0 0000 ; OPTIONS:
00000130 0 0000 ; 0
00000131 0 0000 ; PRINT_CONTROL (IO AND CPU TEST)
00000132 0 0000 ; LOOP_CONTROL (CPU TEST)
00000133 0 0000 ;
00000134 0 0000 ; INSTRUCTION_DEBUGGER:
00000135 0 0000 ; *?F,<CR>?
00000136 0 0000 ;
00000137 0 0000 ;
00000138 0 0000 ;
00000139 0 0000 ; LEGEND:
00000140 0 0000 ; -----
00000141 0 0000 ; BREAK
00000142 0 0000 ; BREAK FROM THE OPERATOR'S CONSOLE WILL IMMEDIATELY STOP ANY
00000143 0 0000 ; PRINT-OUT ON THE OC AND RETURN TO THE MAIN LINE SELECTION MODE.
00000144 0 0000 ; ALL UTILITIES, EXCEPT MEMORY TEST, INCLUDES A BREAK TEST.
00000145 0 0000 ;
00000146 0 0000 ; <PAGE>::= ?0/1/2/3?
00000147 0 0000 ;
00000148 0 0000 ; <BASE_ADDRESS>::= <HEXA>
00000149 0 0000 ; <SIZE>::= <HEXA>
00000150 0 0000 ; <PATTERN>::= <HEXA>
00000151 0 0000 ; <---> HEXADECIMAL DATA_WORD:
00000152 0 0000 ; AN ENTERED HEXADECIMAL DATA WORD CAN BE ANY COMBINATION OF
00000153 0 0000 ; HEXA-DECIMAL CHARACTERS. THOUGH ONLY THE LAST ENTERED FOUR ARE
00000154 0 0000 ; ACCEPTED. ZEROES WILL AUTOMATICALLY BE FILLED IN IF LESS THAN
00000155 0 0000 ; FOUR HEX CHARACTERS HAVE BEEN ENTERED.
00000156 0 0000 ; HEX DATA_WORDS ARE SEPARATED BY ANYTHING BUT '0123456789ABCDEF'
00000157 0 0000 ; AND CARRIAGE RETURN <CR>.
00000158 0 0000 ; <CR> FLAGS END_OF_DATA.
00000159 0 0000 ; <---><DEL> IMPLIES THAT THE ENTERED HEXADECIMAL DATA_WORD IS
00000160 0 0000 ; REJECTED.
00000161 0 0000 ; NOTE THAT +0 ALWAYS IS INTERPRETED AS 64K (ONE PAGE).
00000162 0 0000 ;
00000163 0 0000 ; <OFFSET>::=<HEXA>#. <OFFSET>A
00000164 0 0000 ;
00000165 0 0000 ; A----A OPTIONAL FIELD.
00000166 0 0000 ;
00000167 0 0000 ;

```

```

00000168 0 0000 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000169 0 0000 ;PROGRAM STRUCTURE:
00000170 0 0000 ;-----
00000171 0 0000 ;0000 ALLOCATED CPU_TEST_PROGRAM REFERENCES
00000172 0 0000 ;
00000173 0 0000 ;0004 OFFSET TO BUILT-IN PROCESS DESCRIPTOR
00000174 0 0000 ;0005 PROGRAM_START_ADDRESS (MOD 16=0) + PAGE OF ALTERNATIVE PROGRAM
00000175 0 0000 ;0006 SIZE OF ALTERNATIVE PROGRAM
00000176 0 0000 ;0007 SPARE
00000177 0 0000 ;
00000178 0 0000 ;0010 AUTOCONTINUATION:
00000179 0 0000 ; LOCATIONS FROM WHICH CONTINUATION MAY TAKE PLACE
00000180 0 0000 ; DEFAULTS:
00000181 0 0000 ; DC
00000182 0 0000 ; FD
00000183 0 0000 ; DMA
00000184 0 0000 ; SB16PAGE:
00000185 0 0000 ; PSW#0:3A
00000186 0 0000 ; MSEC
00000187 0 0000 ; JMPL_TABLES:
00000188 0 0000 ; - SUBROUTINES
00000189 0 0000 ; - AMU_UTILITIES
00000190 0 0000 ; - FDSUPPORT_UTILITIES
00000191 0 0000 ; LOCALACT:
00000192 0 0000 ; SERVICE WHENEVER A LOCAL ACTION IS DETECTED WITHIN AMU
00000193 0 0000 ; EMERGENCY:
00000194 0 0000 ; DOUBLE ERROR - EMERGENCY ACTION - DETECTED IN OTHER PROCESS
00000195 0 0000 ; PROCEDURES:
00000196 0 0000 ; * IN_CHAR SB1100
00000197 0 0000 ; * OUT_NL SB1200
00000198 0 0000 ; * OUT_CHAR SB1250
00000199 0 0000 ; * OUT_STRING SB1300
00000200 0 0000 ; * BREAK_CHECK SB1350
00000201 0 0000 ; * GET_HEXA SB1400
00000202 0 0000 ; * OUT_HEXA SB1500
00000203 0 0000 ; * GET_ADDRESS SB1600
00000204 0 0000 ; * SET_PARITY SB1700
00000205 0 0000 ; UTILITIES:
00000206 0 0000 ; - MEMORY_CHECK UT1300 M
00000207 0 0000 ; - COPY_AND_EXECUTE UT1400 E
00000208 0 0000 ; - DUMP UT1500 D
00000209 0 0000 ; - PATCH UT1600 P
00000210 0 0000 ; - SEARCH UT1700 S
00000211 0 0000 ; - EXECUTE UT1800 X
00000212 0 0000 ; - COPY_MEMORY UT1900 C
00000213 0 0000 ; * COPY SB1900 PROCEDURE
00000214 0 0000 ; - IO_UTILITIES UT2000 I
00000215 0 0000 ; - LOOP UT2300 *
00000216 0 0000 ; - WAIT_FOR_INTERRUPT UT2500 W
00000217 0 0000 ; - SWITCH_CPU UT2800 R
00000218 0 0000 ; * SEND_CPU_MESSAGE SB2800 PROCEDURE

```

Address	Value	Label	Offset	Size	Attributes	Procedure
00000217	0 0000	- SWITCH_CPU	UT2800	R		
00000218	0 0000	* SEND_CPU_MESSAGE	SB2800			PROCEDURE
00000219	0 0000	- UNIT_MAPPER	UT2900	U		
00000220	0 0000	- OPTIONS	OPT00	0		
00000221	0 0000	- BOOT_DISKETTE	UT3000	B		
00000222	0 0000	* DEFAULT_DISKETTE	UT3050			PROCEDURE
00000223	0 0000	- LOAD_UTILITIES	UT3100	L		
00000224	0 0000	AMU_COMMAND_CONTROL				
00000225	0 0000	FD_SUPPORT_UTILITIES				
00000226	0 0000	- LOAD_BOOT_FILE	FD1000			
00000227	0 0000	- FD_READ_BLOCK	FD2000			
00000228	0 0000	- FD_TRANSFER	FD3000			
00000229	0 0000	- FD_MEM_COPY	FD3100			
00000230	0 0000	- FD_CONTROL	FD3500			
00000231	0 0000	- DO_DMA	DMA00			
00000232	0 0000	CPU_TEST_UTILITIES				
00000233	0 0000	- INITIALIZE_CPU_TEST				
00000234	0 0000	- EXECUTE_CPU_TEST				
00000235	0 0000	AUTO_BOOT_LOADERS				
00000236	0 0000	- AUTODISKETTE				
00000237	0 0000	- AUTODMA				
00000238	0 0000	- AUTOEXTERNAL				
00000239	0 0000	* SET_UP_DEFAULT_DISKETTE_DRIVE				
00000240	0 0000	----> MCENTRYPOINT:				
00000241	0 0000	- WAIT_MSECS				
00000242	0 0000	- CLAIM_MEMORY				
00000243	0 0000	- WAIT_2*MSECS				
00000244	0 0000	- WRITE_IDENTIFICATION				
00000245	0 0000	- SELECT_CONTINUATION				
00000246	0 0000					
00000247	0 0000					
00000248	0 0000					
00000249	0 0000	IFF				

```

00000250 0 0000 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000251 0 0000 ; CPU TEST ;
00000252 0 0000 ; INSTRUCTION SET KERNEL ;
00000253 0 0000 ; ----- ;
00000254 0 0000 ; 0 - UNTIL TEXT TO BE WRITTEN ON OC ;
00000255 0 0000 ; 1 - DURING WRITE-OUT ;
00000256 0 0000 ; 2 - REST OF INITIALIZATION BEFORE ';>' ;
00000257 0 0000 ; 3 ESSENTIAL TO CPU-TEST PROGRAMS ;
00000258 0 0000 ; ;
00000259 0 0000 ;MOD ;
00000260 0 0000 ; 0 MODC &MA CX ;
00000261 0 0000 ; 0 MOD BX ;
00000262 0 0000 ;MOV ;
00000263 0 0000 ; 0 MOVC &MA C&NA8 R3 ;
00000264 0 0000 ; 0 MOV R3 R3 ;
00000265 0 0000 ; 2 MOV B8 R3 ;
00000266 0 0000 ; 0 MOV R3 B8 ;
00000267 0 0000 ; 3 MOV B6. X2 R3 ;
00000268 0 0000 ; 3 MOV R2 B6. X2 ;
00000269 0 0000 ; 0 MOVL B6. X2 R22 ;
00000270 0 0000 ; 0 MOVL R22 B6. X2 ;
00000271 0 0000 ; 0 MOV P8 R3 ;
00000272 0 0000 ; 1 MVP P6. X2 R2 ;
00000273 0 0000 ;JUMP ;
00000274 0 0000 ; 1 JMP S2 L&NA8 ;
00000275 0 0000 ; 0 JMP L&NA10 ;
00000276 0 0000 ; 1 JMP B6.X2 ;
00000277 0 0000 ; 1 JMPI B6 ;
00000278 0 0000 ; 0 JMPI &S2A P6 ;
00000279 0 0000 ; 0 SOB R3 LNB ;
00000280 0 0000 ;ARIT/LOG ;
00000281 0 0000 ; 0 ADDC C&NA8 R3 ;
00000282 0 0000 ; 0 ADD R3 R3 ;
00000283 0 0000 ; 0 SUB R3 R3 ;
00000284 0 0000 ; 1 SBN/Z R3 C4 ;
00000285 0 0000 ; 1 SBZF R3 C4 ;
00000286 0 0000 ; 3 SLC R3 C4 ;
00000287 0 0000 ; 1 XTR R3 C4 ;
00000288 0 0000 ; 1 SWP R3 R3 ;
00000289 0 0000 ; 3 INE R3 R3 ;
00000290 0 0000 ; 3 INE X3 X3 ;
00000291 0 0000 ; 1 IER/INE R3 &MA C4 ;
00000292 0 0000 ; 3 SHS/SLO R2 C4 ;
00000293 0 0000 ;I/O ;
00000294 0 0000 ; 0 CIO R3 R3 ;
00000295 0 0000 ; 0 SID R3 R3 ;
00000296 0 0000 ; 0 RID R3 R3 ;
00000297 0 0000 ; 1 WIO R3 R3 ;
00000298 0 0000 ;SPECIAL ;
00000299 0 0000 ; 0 SVP L4 ;
00000300 0 0000 ; 2 LDP R3 ;

```

[illegible]

```

00000318 0 0000 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;MEMORY LAYOUT;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000319 0 0000 BEGIN  MODULE CPUROM
00000320 0 0000 AREASWITCH= 1
00000321 0 0000
00000322 0 0000 FDSUPPORT= TRUE
00000323 0 0000 SCNMODE= TRUE
00000324 0 0000 SCNMODE= TRUE
00000325 0 0000 AMOS= FALSE
00000326 0 0000 XAMOS= NT AMOS
00000327 0 0000
00000328 0 0000 XXAMOS=TRUE
00000329 0 0000 CPUROM= #F800
00000330 0 0000
00000331 0 0000 USE BASE
00000332 0 0000 BIT0=0, BIT1=1, BIT2=2, BIT3=3, BIT4=4, BIT5=5, BIT6=6
00000333 0 0000 BIT7=7, BIT8=8, BIT9=9, BIT10=10, BIT11=11, BIT12=12,
00000334 0 0000 BIT13=13, BIT14=14, BIT15=15
00000335 0 0000
00000336 0 0000 ; MEMORY LAY-OUT (ABS)
00000337 0 0000 LOC= 0 ; -----
00000338 0 0000 CPUMEN:
00000339 0 0000 CPUIVT: LOC= LOC+4 ; INTERRUPT VECTOR TABLE INDEX - PRIORITY
00000340 0 0004 CPUIVR: LOC= LOC+1 ; RESET BASE, INTERRUPT SERVE PROCESS
00000341 0 0005 LOC= LOC+2 ;
00000342 0 0007 CPUMSG: LOC= LOC+1 ; USED FOR "OTHER'S" MESSAGE (ADDREESS DUMMY)
00000343 0 0008 CPUCPU: LOC= LOC+8 ; CPU MESSAGE AREA
00000344 0 0010 IF AMOS THEN
00000345 0 0010 LOC= LOC+#10 ;
00000346 0 0010 ELSE
00000347 0 0010 LOC= LOC+#E ;
00000348 0 001E CPUDUMMYLEVEL: LOC= LOC+1;
00000349 0 001F CPUDUMMYBOUND: LOC= LOC+1;
00000350 0 0020 FI
00000351 0 0020 CPUDUMMY: LOC= LOC+#1F ; BASE LOOSER'S DUMMY PROCESS.
00000352 0 003F ; USED FOR SVP FROM OTHER CPUS UPON FIRST
00000353 0 003F ; CPU INTERRUPT;
00000354 0 003F LOC= LOC+1 ; SPECIAL MONCALL CASE (0-63)
00000355 0 0040 CPUMON: LOC= LOC+192 ; MONITOR TABLE (ABS PROGRAM ADDRESS
00000356 0 0100 LOC= LOC+128 ; RESERVED PROGRAM AREA (MON 0-63)
00000357 0 0180 CPUIVB: LOC= LOC+64 ; INTERRUPT VECTOR TABLE (PROCESS BASE)
00000358 0 01C0 IF AMOS THEN
00000359 0 01C0 LOC= LOC+32 ; BASE DUMMY INTERRUPT PROCESS
00000360 0 01C0 LOC= LOC+32 ;
00000361 0 01C0 ELSE
00000362 0 01C0 LOC= LOC+14 ;
00000363 0 01CE CPUDLEVEL: LOC= LOC+1 ;
00000364 0 01CF CPUDBOUND: LOC= LOC+1 ;
00000365 0 01D0 CPUDBASE: LOC= LOC+32 ;
00000366 0 01F0 LOC= LOC+16 ;
00000367 0 0200 FI
00000368 0 0200 CPUBASE: LOC= LOC+256 ; PROCESS AREA CPU TEST PROGRAM

```



```

00000367 0 0200 FI ;
00000368 0 0200 CPUBASE: LOC= LOC+256 ; PROCESS AREA CPU TEST PROGRAM
00000369 0 0300 CPUNWORK: ; WORK AREA
00000370 0 0300 ;
00000371 0 0300 IF XXAMOS THEN ;
00000372 0 0300 MASTERCLEARLEVEL:=#FFDE ;
00000373 0 0300 FI ;
00000374 0 0300 MASTERCLEARBASE:=#FFE0 ;
00000375 0 0300 ;
00000376 0 0300 ; ASCII CHARACTERS
00000377 0 0300 ; -----
00000378 0 0300 NIL= 0 ; NIL CHARACTER
00000379 0 0300 LF= <:<10>:> ; LINE_FEED
00000380 0 0300 CR= <:<13>:> ; CARRIAGE_RETURN
00000381 0 0300 YES= <:Y:> ;
00000382 0 0300 DEL= #7F ; DELETE
00000383 0 0300 EDT= #FF ; END_OF_TEXT (PROCEDURE OUT_STRING)
00000384 0 0300 JFF ;

```

```

00000385 0 0300 ;-----
00000386 0 0300 ;
00000387 0 0300 ;NOTE THAT THE FDLUTILITIES USES 256/272 WORDS BELOW THE AMU PROCESS
00000388 0 0300 ;   DESCRIPTOR AS DEDICATED MEMORY BUFFER DURING INPUT.
00000389 0 0300 ;
00000390 0 0300 ;----- P R O C E S S   D E S C R I P T O R -----
00000391 0 0300 IF AMOS THEN
00000392 0 0300     LOC= 0      ;
00000393 0 0300 ELSE
00000394 0 0300     LOC= -2
00000395 0 FFFE XLEVEL: LOC= LOC+1 ;
00000396 0 FFFF XBOUND: LOC= LOC+1 ;
00000397 0 0000 FI
00000398 0 0000 XR0:   ABSMCE50 ;
00000399 0 0001 XR1:   CPUDUMMY ;
00000400 0 0002 XR2:   256 ; MSEC BEFORE FASTEST CPU CLAIMS MEMORY
00000401 0 0003 XR3:   CPUBASE ;
00000402 0 0004 XR4:   0 ; OFFSET FROM AMUCONTINUATION TAKES PLACE
00000403 0 0005 XR5:   PROGRAMSTART ;
00000404 0 0006 XR6:   XR0 ;
00000405 0 0007 XR7:   XABS ;
00000406 0 0008 XBASE: CPUBASE ; PROCESS BASE CPU TEST PROGRAM
00000407 0 0009 XMOD:  CPUBASE ; MODIFY REGISTER NORMAL STATE
00000408 0 000A XPROG: PROGRAMSTART ;
00000409 0 000B XPRPC: PROGRAMGO ;
00000410 0 000C XTIMER: #7FFF ; TIMER REGISTER
00000411 0 000D XPSW:  #E000 ; PROGRAM STATUS WORD (PSW)
00000412 0 000E ;***** END OF PROCESS DESCRIPTOR *****
00000413 0 000E
00000414 0 000E XOLDPRC:MASTERCLEARBASE ; PROCESS LINK ADDRESS (ABS)
00000415 0 000F XLOCACT:LOCALACT ; LOCAL INTERRUPT ACTION PROCEDURE ADDRESS
00000416 0 0010 XLOCRET:0-0 ; LOCAL INTER'T RETURN ADDRESS
00000417 0 0011 XCAUSE: #FFFF ; CAUSE:
00000418 0 0012 ; 0 - LOCAL TIMER
00000419 0 0012 ; 1 - TRAP
00000420 0 0012 ; 2 - PARITY ERROR
00000421 0 0012 ; 3 - TIME OUT
00000422 0 0012 ; 4 - BOUND ERROR
00000423 0 0012 XDEVICE:0-0 ; IF INTERRUPT THEN DEV_ADDR ELSE #FFFF
00000424 0 0013 XTIMRS: #8000 ; FAST TIMER PRESET VALUE
00000425 0 0014
00000426 0 0014
00000427 0 0014 SEMAPHORE:0 ; DEDICATED AS CPU_SEMAPHORE
00000428 0 0015
00000429 0 0015
00000430 0 0015 ; WORK AREA
00000431 0 0015 ; -----
00000432 0 0015 PGMEVALUATE: ; TRUE => EVALUATE TEST RESULTS AND PRINT
00000433 0 0015 ; MISMATCHES ON THE OC.
00000434 0 0015 PRINTOUT: TRUE
00000435 0 0016

```

```

00000434 0 0015 PRINTOUT: TRUE
00000435 0 0016
00000436 0 0016 PGML00P: 0, 0 ; NO_OF_LOOPS_LEFT-1 (LONG_INTEGER);
00000437 0 0018 ;
00000438 0 0018 TCBASE: #1000 ; START CPU_TEST PROGRAM (RAM_COPY)
00000439 0 0019 ;NOTE HAS TO BE WITHIN 32 K FROM "OWN" BASE.
00000440 0 0019 IF AMOS THEN
00000441 0 0019 XABS= (LOC+1-XR0)/2
00000442 0 0019 ELSE
00000443 0 0019 XABS= (LOC+1-XLEVEL)/2 ;
00000444 0 0019 FI
00000445 0 0019
00000446 0 0019 ENDOFDATA:
00000447 0 0019
00000448 0 0019
00000449 0 0019
00000450 0 0019
00000451 0 0019 SB12SV: LOC= LOC+1 ; DEDICATED SB1200 (WRITE_NEWLINE)
00000452 0 001A SB14SV: LOC= LOC+1 ; DEDICATED SB1400 (GET_HEX)
00000453 0 001B SB15SV: LOC= LOC+1 ; DEDICATED SB1500 (WRITE_HEX)
00000454 0 001C SB16SV: LOC= LOC+1 ; DEDICATED SB1600 (GET_ADDRESS)
00000455 0 001D SB17SV: LOC= LOC+2 ; DEDICATED SB1700 (SET_PARITY)
00000456 0 001F
00000457 0 001F UT50SV: LOC= LOC+1 ;
00000458 0 0020 ;
00000459 0 0020 UT13PATTERN: ; 3 (MEMORY_CHECK)
00000460 0 0020 UT14SV: ; 1 (COPY_AND_EXECUTE)
00000461 0 0020 UT16SV: ; 1 (PATCH - REWRITE SECTION ONLY)
00000462 0 0020 UT17SV: ; 1 (SEARCH)
00000463 0 0020 UT20SV: ; 3 (IO_UTILITIES)
00000464 0 0020 UT30SV: ; 2 (BOOT/LOAD FROM FLOPPY)
00000465 0 0020 UT60SV: ; 1 (DOWNLOAD LOG'S)
00000466 0 0020 LOC= LOC+3
00000467 0 0023 UT99SV:
00000468 0 0023 ;***** END OF DUMP/PATCH/SEARCH UTILITIES *****
00000469 0 0023 JFF

```

```

00000470 0 0023 ;*****

```

```

00000470 0 0023 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000471 0 0023 ;CPU_TEST PROGRAM REFERENCES
00000472 0 0023 ;-----
00000473 0 0023 LOC= LOC+(#24-LOC)
00000474 0 0024 TCREFWORK:
00000475 0 0024 REFWORK: LOC= LOC+1 ; REFERENCE CPU_TEST PROGRAM WORK_AREA.
00000476 0 0025 MAISV: LOC= LOC+2 ; DEDICATED MAIN LINE
00000477 0 0027
00000478 0 0027 B5555: LOC= LOC+1 ;
00000479 0 0028 BAAAA: LOC= LOC+1 ;
00000480 0 0029
00000481 0 0029 CP06:
00000482 0 0029 CP20INS:
00000483 0 0029 CPWORK: LOC= LOC+4 ; WORK AREA FOR CPU_TEST PROGRAMS
00000484 0 002D
00000485 0 002D CP46P: LOC= LOC+1 ;
00000486 0 002E CP46: LOC= LOC+1 ;
00000487 0 002F
00000488 0 002F
00000489 0 002F PGMNEXT: LOC= LOC+1 ; REF TABLE PGMPOOL - CURRENT TEST PGM REF
00000490 0 0030 ;
00000491 0 0030 TCP00L: ; REF(TEST_PROGRAM), REF(DATA_AREA)
00000492 0 0030 LOC= 0
00000493 0 0000 TCPFROG: LOC= LOC+1
00000494 0 0001 TCPDATA: LOC= LOC+1
00000495 0 0002 LOC= LOC+TCP00L
00000496 0 0032 ;
00000497 0 0032 IFF

```

```

00000498 0 0032 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000499 0 0032 ;FD UTILITIES WORK ;
00000500 0 0032 ;----- ;
00000501 0 0032 IF AMOS THEN ;
00000502 0 0032 FDMEMBUF:= XRO-256 ; I/O BUFFER 256 POSITIONS BELOW OWN BASE ;
00000503 0 0032 ELSE ;
00000504 0 0032 FDMEMBUF:= XRO-272 ; I/O BUFFER 272 POSITIONS BELOW OWN BASE ;
00000505 0 0032 FI ;
00000506 0 0032 ; ;
00000507 0 0032 LOC= UT99SV ;
00000508 0 0023 FDADR: LOC= LOC+1 ; FLOPPY DISK DEVICE ADDRESS ;
00000509 0 0024 FDMODE: LOC= LOC+1 ; MODE OF OPERATION: ;
00000510 0 0025 FD10SV: LOC= LOC+1 ; DEDICATED FD1000 (FD_BOOT) ;
00000511 0 0026 ; 'F'-'D' - FD ;
00000512 0 0026 ; 'X'-'D' REMOTE FD VIA DMA ;
00000513 0 0026 FD10DRV: LOC= LOC+1 ; FD DRIVE NO (0-3) ;
00000514 0 0027 FDZERO: ; ;
00000515 0 0027 FDTRACK: LOC= LOC+1 ; CURRENT TRACK POSITION OF FD_HEAD ;
00000516 0 0028 FD20SV: LOC= LOC+1 ; DEDICATED FD2000 (FD_READ_BLOCK) ;
00000517 0 0029 FD20CTL: LOC= LOC+1 ; ;
00000518 0 002A FD30SV: LOC= LOC+1 ; DEDICATED FD3000 (FD_TRANSFER) ;
00000519 0 002B FDRECOUNT: LOC= LOC+1 ; NO OF RESTORE ATTEMPTS ON FD (<16). ;
00000520 0 002C ; ;
00000521 0 002C FDCLOAD: LOC= LOC+1 ; CURRENT LOAD ADDRESS (OUTPUT_AREA) ;
00000522 0 002D FDCPAGE: LOC= LOC+1 ; CURRENT MEMORY SECTION ;
00000523 0 002E ; ;
00000524 0 002E FDBLOCKNO: LOC= LOC+1 ; CURRENT BLOCK NO ;
00000525 0 002F ; ;
00000526 0 002F FDAREAS= 10 ; MAX AREAS PER FILE ;
00000527 0 002F FDBENTRY: ; FD BOOT FILE DIR ENTRY ;
00000528 0 002F LOC= 0 ;
00000529 0 0000 FDAREASIZE: LOC= LOC+1 ; AREA_SIZE ;
00000530 0 0001 FDSECTORS: LOC= LOC+1 ; SECTORS PER BLOCK ;
00000531 0 0002 FDHIBLOCK: LOC= LOC+1 ; HIGHEST BLOCK ALLOCATED ;
00000532 0 0003 FDBLOCKS: LOC= LOC+1 ; BLOCKING_FACTOR: BLOCKS PER AREA ;
00000533 0 0004 FDAREA: LOC= LOC+FDAREAS; 1ST AREA ADDRESS (-1 IF NON-ALLOCATED) ;
00000534 0 000E FDL0AD: LOC= LOC+1 ; FIRST ADDRESS (LOAD ADDRESS) ;
00000535 0 000F FDPAGE: LOC= LOC+1 ; MEMORY SECTION (PAGE) ;
00000536 0 0010 FDBASE: LOC= LOC+1 ; BASE ADDRESS ;
00000537 0 0011 FDCOUNT: LOC= LOC+1 ; WORD COUNT OF FILE ;
00000538 0 0012 FDCHECKSUM: LOC= LOC+1 ; CHECKSUM OF BOOT FILE ;
00000539 0 0013 FDCHECK: LOC= LOC+1 ; CHECKSUM OF DIRECTORY ENTRY ;
00000540 0 0014 FDENTRYLENGTH: ; ;
00000541 0 0014 LOC= FDBENTRY+FDAREA+1 ;
00000542 0 0034 FDDIRBUF: LOC= LOC+FDENTRYLENGTH ;
00000543 0 0048 FDEND: ; ;
00000544 0 0048 ;***** END OF DESTROYED AREA ***** ;
00000545 0 0048 ;FF ;

```

```

00000546 0 0048 USE PROG
00000547 1 0000 ;----- P R O G R A M   A R E A -----;
00000548 1 0000 ;
00000549 1 0000 ;NOTE THE FIRST LOCATION OF THE PROGRAM AREA MUST BE A "MODE -1"
00000550 1 0000 ; FOLLOWED BY THREE MORE LOCATIONS DEDICATED THE CPU TEST.
00000551 1 0000 ; WHILE THE 5TH LOCATION OF THE PROGRAM AREA MUST BE A
00000552 1 0000 ; PROGRAM RELATIVE POINTER TO THE PROCESS DESCRIPTOR TO
00000553 1 0000 ; USED IN ORDER TO INITIATE THE PROGRAM BY THE UTILITY
00000554 1 0000 ; "EXECUTE".
00000555 1 0000 ;
00000556 1 0000 ;
00000557 1 0000 PROGSTART:
00000558 1 0000 ;CPU TEST PROGRAM CONSTANTS
00000559 1 0000 ;-----
00000560 1 0000 MODC -1 ; THIS AND SUBSEQUENT THREE LOCATIONS IS
00000561 1 0001 ; USED BY THE CPU TEST PROGRAM AND MUST THUS
00000562 1 0001 ; NOT BE CHANGED.
00000563 1 0001 ;
00000564 1 0001 P5555: #5555 ;
00000564 1 0002 ***** A ***** MODE-WRN 302 ;
00000565 1 0002 PAAAA: #AAAA ;
00000566 1 0003 ;
00000567 1 0003 MAINLINE: MAINLOOP ; RETURN LINK FROM CPU TEST SEQUENCES.
00000568 1 0004 ;
00000569 1 0004 ;COPY AND EXECUTE BUILT-IN DEFAULTS
00000570 1 0004 ;-----
00000571 1 0004 ; THIS AND SUBSEQUENT TWO LOCATIONS USED
00000572 1 0004 ; FOR EXECUTION OF EXTERNAL PROGRAM FROM
00000573 1 0004 ; AMU PROCESS. MUST NOT HAVE LOCATION CHANGED.
00000574 1 0004 ; AMU. THEREFORE, THE LOCATIONS MUST NOT BE
00000575 1 0004 ; CHANGED IN FUTURE AMU VERSIONS.
00000576 1 0004 AMUPROCESS: RELPROMBASE ;***FOR USAGE BY THE EXECUTE UTILITY, I.E.
00000577 1 0005 ; MUST BE POINTER TO PROCESS DESCRIPTOR.
00000578 1 0005 #0800 ; SIZE OF PROGRAM MODULE;
00000579 1 0006 ;
00000580 1 0006 #0000, REPEAT (#08-LOC)
00000581 1 0008 ;
00000582 1 0008 #0003 ;*** IDENTIFICATION OF EXTERNAL MODULE:
00000583 1 0009 ; PROGRAM_START_ADDRESS + MEMORY_SECTION
00000584 1 0009 ; PROGRAM_START_ADDRESS MULTIPLY #10
00000585 1 0009 ;
00000586 1 0009 #0000, REPEAT (#10-LOC)
00000587 1 0010 ;
00000588 1 0010 AMUCONTINUATION:
00000589 1 0010 MCE50 ; UTILITIES
00000590 1 0011 AUTODISKETTE ; FD 0 ELSE FD 1 ELSE UTILITIES
00000591 1 0012 AUTODMA ; XD 1 ELSE XD 0 ELSE UTILITIES
00000592 1 0013 AUTO60 ; EXTERNAL MODULE
00000593 1 0014 AUTOEXTERNAL ; FD 0 ELSE FD 1 ELSE EXTERNAL MODULE
00000594 1 0015 0, REPEAT (#08-(LOC-AMUCONTINUATION))
00000595 1 0018 JFF

```

00000594 1 0015
00000595 1 0018 ;FF

0, REPEAT (#08-(LOC-AMUCONTINUATION))

00000596 1 0018 ;DEFAULTS

```

00000596 1 0018 ;DEFAULTS
00000597 1 0018 ;-----
00000598 1 0018          OC= 1
00000599 1 0018          SCM= OC
00000600 1 0018 SCMAADR:
00000601 1 0018 OCAADR:      OC      ; ADDRESS OF OPERATOR'S CONSOLE (OC)
00000602 1 0019
00000603 1 0019          FD= 2
00000604 1 0019 FDDEFAULT: FD      ; DEFAULT DISKETTE CONTROLLER ADDRESS
00000605 1 001A
00000606 1 001A DMAADR:      7      ; DMA IO ADDRESS
00000607 1 001B
00000608 1 001B
00000609 1 001B SB16PAGE: #E000, #E004, #E008, #E00C
00000610 1 001F
00000611 1 001F MSEC:      727      ; SOB-COUNTER (APPR 1-1.3 MSEC)
00000612 1 0020
00000613 1 0020
00000614 1 0020          ;SUBROUTINES:
00000615 1 0020          ;-----
00000616 1 0020 BREAKCHECK: SB1350      ; BREAK_CHECK      R:(LINK/LINK+1);
00000617 1 0021
00000618 1 0021 COPY:      SB1900      ; COPY      I:(SOURCE, OBJECT);
00000619 1 0022
00000620 1 0022 GETADDRESS: SB1600      ; GET_ADDRESS      O:(REF(LOWER, UPPER));
00000621 1 0023 GETHEXA:      SB1400      ; GET_HEXADECIMAL  O:(DELIMITER, BINARY);
00000622 1 0024
00000623 1 0024 INCHAR:      SB1100      ; READ_SINGLE_CHAR O:(CHAR);
00000624 1 0025
00000625 1 0025 OUTCHAR:      SB1250      ; WRITE_SINGLE_CHAR I:(CHAR);
00000626 1 0026 OUTHEXA:      SB1500      ; WRITE_HEXADECIMAL U:(BINARY);
00000627 1 0027 OUTNL:      SB1200      ; WRITE_NEWLINE;
00000628 1 0028 OUTSTRING: SB1300      ; OUT_STRING      I:(REF(STRING));
00000629 1 0029
00000630 1 0029 SETPARITY: SB1700      ;
00000631 1 002A
00000632 1 002A UTILITIES: UT5000      ;UTILITIES:
00000633 1 002B BOOTLOAD: UT3000      ; B
00000634 1 002C COPYMEMORY: UT1900      ; C
00000635 1 002D DUMP: UT1500      ; D
00000636 1 002E EXECUTE: UT1400      ; E
00000637 1 002F ID: UT2000      ; I
00000638 1 0030 INSTRUCTIONLOOP:UT2300      ; *
00000639 1 0031 LOADFILE: UT3100      ; L
00000640 1 0032 MEMCHECK: UT1300      ; M
00000641 1 0033 OPTIONS: OPT00      ; O
00000642 1 0034 PATCH: UT1600      ; P
00000643 1 0035 SWITCHCPU: UT2800      ; Q
00000644 1 0036 RETURNCALLER:MCE90      ; R
00000645 1 0037 SEARCH: UT1700      ; S
00000646 1 0038 TESTCPU: TEC00      ; T

```



```

00000645 1 0037 SEARCH: UT1700 ; S ;
00000646 1 0038 TESTCPU: TEC00 ; T ;
00000647 1 0039 UNITS: UT2900 ; U ;
00000648 1 003A WAITINTERRUPT:UT2500 ; W ;
00000649 1 003B XECUTE: UT1800 ; X ;
00000650 1 003C ; ;
00000651 1 003C ERROR: UT5090 ; ;
00000652 1 003D ; ;
00000653 1 003D NOLIST IF FDSUPPORT THEN LIST ; ;
00000654 1 003D FDBLOAD: FD1000 ; LOAD_FD_BOOT_MODULE ;
00000655 1 003E FDBREADBLOCK:FD2000 ; FD_READ_BLOCK I:(REF(DIR_ENTRY)); ;
00000656 1 003F FDWAITCMDCOMPLETE: ; ;
00000657 1 003F FDTRANSFER: FD3000 ; FD_TRANSFER I:(REF(BUFFER)); ;
00000658 1 0040 FDCOPY: FD3100 ; FD_MEM_COPY ;
00000659 1 0041 FDCONTROL: FD3500 ; FD_CONTROL_IO I:(COMMAND); ;
00000660 1 0042 NOLIST ELSE LIST ; ;
00000662 1 0042 NOLIST FI LIST ; ;
00000663 1 0042 JFF ; ;

```

```

00000664 1 0042 ; COMMON LOCAL ACTION
00000665 1 0042 ;
00000666 1 0042 ; PROCEDURE SERVE_LOCAL_INTERRUPT;
00000667 1 0042 ;
00000668 1 0042 ;
00000669 1 0042 LOCALACT: ;NORMAL_LOCALACTION:
00000670 1 0042 SUP ;
00000671 1 0043 IF XXAMOS THEN
00000672 1 0043 MOVC XLEVEL R7 ;
00000673 1 0044 CLR 0. X7 ; RESET LEVEL
00000674 1 0045 FI
00000675 1 0045 MOV XLOCRET R0 ; GET CAUSING POSITION;
00000676 1 0046 //2 MOVC $FFFF R2 ; PRESET CAUSE_VALUE;
00000677 1 0047 MOVC XCAUSE R7 ;
00000678 1 0048 XCH X7 R2 ; GET CAUSE AND RESET FIELD;
00000679 1 0049 MOVC XPSW R7 ; POINT PROCESS_PSW;
00000680 1 004A RELS X7 BIT11 ; RESET LOCAL_INTERRUPT_FLAG;
00000681 1 004B ;
00000682 1 004B ;----- CPU-TEST (NOP) -----;
00000683 1 004B JMP LOC10 ;***BY-PASS LOCAL_TIMER_INTERRUPT***;
00000684 1 004C SNE R2 0 ; IT .NOT. TIMER_INTERRUPT
00000685 1 004D JMP LOC40 ; THEN
00000686 1 004E LOC10: MOV XPSW R7 ;
00000687 1 004F SETS R7 BIT15 ; DISABLE TIMER_INTERRUPTS;
00000688 1 0050 LDS R7 ; RE-ESTABLISH PSW;
00000689 1 0051 JMPI S4 OUTSTRING, <(<13>LOCAL_INTERRUPT, CAUSE=:), EOT
00000690 1 005F JMPI S6 OUTHEXA ;
00000691 1 0060 JMPI S6 UTILITIES ;
00000692 1 0061 ;RETURN:
00000693 1 0061 JMPI S4 OUTSTRING, <(<13>LOCAL_RETURN_ADDRESS :), EOT
00000694 1 006E MOV XLOCRET R2 ; GET SAVED LOCATION;
00000695 1 006F JMPI S6 OUTHEXA ;
00000696 1 0070 JMPI S4 OUTSTRING, <:, ENTER_OFFSET :), EOT
00000697 1 007A JMPI S6 GETHEXA ; GET LOCAL_RETURN_OFFSET;
00000698 1 007B MOV XLOCRET R0 ;
00000699 1 007C ADD R2 R0 ; COMPUTE NEW START LOCATION
00000700 1 007D LOC40: ;RETURN_CAUSING_LOCATION:
00000701 1 007D MOVC XBASE R4 ; POINT "OWN" BASE;
00000702 1 007E ADD XPRG-XBASE.X4 R0 ; ADJUST WAKE-UP LOCATION;
00000703 1 007F MOV R0 XPRPC ;
00000704 1 0080 LDN X4 ; RETURN INTERRUPTED LOCATION;
00000705 1 0081 ;
00000706 1 0081 ;FF

```

```

00000707 1 0081 ; EMERGENCY ACTION ENTRY POINT ;
00000708 1 0081 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000709 1 0081 IF ANDS THEN ;
00000710 1 0081 EMERGENCY: ;
00000711 1 0081 SVP ;
00000712 1 0081 MODN XBASE ;
00000713 1 0081 MOVC MASTERCLEARBASE R4 ; POINT PROMMED AMULBASE;
00000714 1 0081 MOV XBASE R0 ;
00000715 1 0081 MOVC #E000 R7 ; FORCE
00000716 1 0081 LDS R7 ; MEMORY SECTION 0;
00000717 1 0081 MOV XBASE. X4 R6 ; GET BASE THAT PROCESS;
00000718 1 0081 SUB R0 R6 ; SET-UP POINTER TO RAMBASE;
00000719 1 0081 MOVN X4 X6 ; CREATE
00000720 1 0081 MOVN X4 X6 ; AMULPROCESS;
00000721 1 0081 ADDC -2*16 R6 ; POINT START PROCESS_DESCRIPTOR;
00000722 1 0081 MOVC LOC50 R0 ;
00000723 1 0081 ADD XPROG. X6 R0 ; COMPUTE START_ADDRESS;
00000724 1 0081 MOV R0 XPRPC. X6 ;
00000725 1 0081 MOV XBASE. X6 R7 ; GET BASE RAM-RESIDENT AMULPROC;
00000726 1 0081 LDP R7 ; LEAVE CAUSING PROCESS
00000727 1 0081 ;*** BY ACTIVATING AMULPROCESS;
00000728 1 0081 ELSE XANDs DO INSERT ;
00000729 1 0081 EMERGENCY: ;
00000730 1 0081 SSP ;
00000731 1 0082 MODN XBASE ;
00000732 1 0083 MOVC MASTERCLEARLEVEL R4 ; POINT PROMMED AMULBASE;
00000733 1 0084 MOV XBASE R0 ;
00000734 1 0085 MOVC #E000 R7 ; FORCE
00000735 1 0087 LDS R7 ; MEMORY SECTION 0;
00000736 1 0088 MOV XBASE-XLEVEL % ;
00000737 1 0088 X4 R6 ; GET BASE THAT PROCESS;
00000738 1 0089 SUB R0 R6 ; SET-UP POINTER TO RAMBASE;
00000739 1 008A ADDC XLEVEL R6 ;
00000740 1 008B MOVN X4 X6 ; CREATE
00000741 1 008C MOVN X4 X6 ; AMULPROCESS;
00000742 1 008D ADDC -2*16-XLEVEL R6 ; POINT START PROCESS_DESCRIPTOR;
00000743 1 008E MOVC LOC50 R0 ;
00000744 1 008F ADD XPROG. X6 R0 ; COMPUTE START_ADDRESS;
00000745 1 0090 MOV R0 XPRPC. X6 ;
00000746 1 0091 MOV XBASE. X6 R7 ; GET BASE RAM-RESIDENT AMULPROC;
00000747 1 0092 SLP R7 ; LEAVE CAUSING PROCESS
00000748 1 0093 ;*** BY ACTIVATING AMULPROCESS;
00000749 1 0093 FI ;
00000750 1 0093 ;***** ENTRY POINT UPON EMERGENCY WAKE UP *****;
00000751 1 0093 LOC50: ;ENTRY_POINT FROM EMERGENCY:
00000752 1 0093 JMPI S4 OUTSTRING, <(<13>)EMERGENCY ACTION BY :>, EOT ;
00000753 1 00A0 MOV XOLDPRC R2 ; GET CAUSING PROCESS_BASE;
00000754 1 00A1 JMPI S6 OUTHEXA ;
00000755 1 00A2 LOC90: JMPI S6 UTILITIES ; ACTIVATE UTILITIES;
00000756 1 00A3 JMP LOC90 ; ENSURE NO RETURN;
00000757 1 00A4 JFF ;

```

00000754 1 00A1 JMP1
00000755 1 00A2 LOC90: JMP1
00000756 1 00A3 JMP
00000757 1 00A4 JFF

56 OUTHEXA ;
56 UTILITIES ; ACTIVATE UTILITIES;
LOC90 ; ENSURE NO RETURN;
;

```

00000758 1 00A4 NOLIST IF SCMODE THEN LIST
00000759 1 00A4 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000760 1 00A4 ; DC SCM CHARACTERISTICS
00000761 1 00A4 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000762 1 00A4 ;COMMAND_REGISTERS:
00000763 1 00A4 SCMMR1= 2#010<BIT8 ;MODE_REG_1:
00000764 1 00A4 SCMASYNC= X)
00000765 1 00A4 2#11<BIT0 OR X) ASYNCHRONOUS
00000766 1 00A4 2#10<BIT2 OR X) 7 BITS
00000767 1 00A4 1<BIT4 OR X) PARITY ENABLED
00000768 1 00A4 1<BIT5 OR X) EVEN PARITY
00000769 1 00A4 2#01<BIT6 ; 1 STOP BIT
00000770 1 00A4 SCMMR2= 2#110<BIT8 ;MODE_REG_2:
00000771 1 00A4 SCMBAUD= X)
00000772 1 00A4 #0<BIT0 OR X) BAUD RATE (ASYNC)
00000773 1 00A4 2#11<BIT4 OR X) INTERNAL CLOCK
00000774 1 00A4 0<BIT6 ; SLOT 1 (FRONT-PANEL)
00000775 1 00A4 SCMCMD= 2#011<BIT8 ;COMMANDS:
00000776 1 00A4 SCMENB= X)
00000777 1 00A4 1<BIT0 OR X) ENABLE TX CONTROL
00000778 1 00A4 1<BIT1 OR X) FORCE DTR LOW
00000779 1 00A4 1<BIT2 OR X) ENABLE RX
00000780 1 00A4 0<BIT3 OR X) NORMAL
00000781 1 00A4 0<BIT4 OR X) KEEP ERROR FLAGS
00000782 1 00A4 1<BIT5 OR X) FORCE RTS LOW
00000783 1 00A4 2#00<BIT6 ; NORMAL OPERATION
00000784 1 00A4 SCMRESET= X)
00000785 1 00A4 1<BIT0 OR X) ENABLE TX CONTROL
00000786 1 00A4 1<BIT1 OR X) FORCE DTR LOW
00000787 1 00A4 1<BIT2 OR X) ENABLE RX
00000788 1 00A4 0<BIT3 OR X) NORMAL
00000789 1 00A4 1<BIT4 OR X) RESET ERROR FLAGS IN STATUS
00000790 1 00A4 1<BIT5 OR X) FORCE RTS LOW
00000791 1 00A4 2#00<BIT6 ; NORMAL OPERATION
00000792 1 00A4
00000793 1 00A4 SCMSTATUS= 2#001<BIT8 ;STATUS_BITS:
00000794 1 00A4 SCHORDY= BIT0 ; TRANSMIT HOLDING REG EMPTY (HIGH)
00000795 1 00A4 SCHIRDY= BIT1 ; RECEIVE HOLDING REG HAS DATA (INPUT READY)
00000796 1 00A4 SCHPARITY= BIT3 ; PARITY ERROR (HIGH)
00000797 1 00A4 SCHOVERRUN= BIT4 ; OVERRUN ERROR (HIGH)
00000798 1 00A4 SCHFRAME= BIT5 ; FRAMING ERROR (HIGH)
00000799 1 00A4 SCHDSR= BIT7 ; DATASET_READY (HIGH - RECEIVER READY)
00000800 1 00A4 ;FF

```

```

00000801 1 00A4 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000802 1 00A4 ; PROCEDURE IN_CHAR 0:(CHAR);
00000803 1 00A4 ; R0 - CHAR (LEAST SIGN 7 BITS)
00000804 1 00A4 ; R6 LINK DESTROYED
00000805 1 00A4 ; R7 - DESTROYED
00000806 1 00A4 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000807 1 00A4 ;IN_CHAR:
00000808 1 00A4 SB1100: ; REPEAT
00000809 1 00A4 MOV SCHADR R7 ; GET DEVICE ADDRESS;
00000810 1 00A5 MOVC SCHENB R0 ;
00000811 1 00A6 MODC SCHCMD R7 ;
00000812 1 00A7 CIO R0 R7 ; ENABLE OC;
00000813 1 00A8 MODC SCHSTATUS R7 ;
00000814 1 00A9 SIO R0 R7 ; SENSE(STATUS, DEVICE);
00000815 1 00AA SBN R0 SCHIRDY ; UNTIL INPUT_READY HIGH;
00000816 1 00AB JMP SB1100 ;
00000817 1 00AC RIO R0 R7 ; GET CHAR FROM OC;
00000818 1 00AD XTR R0 7 ; STRIP OFF PARITY;
00000819 1 00AE JMP SB1250 ; ECHOE CHARACTER;
00000820 1 00AF
00000821 1 00AF
00000822 1 00AF ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000823 1 00AF ; PROCEDURE OUT_NEWLINE;
00000824 1 00AF ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000825 1 00AF ;OUT_NEWLINE:
00000826 1 00AF SB1200: ;
00000827 1 00AF MOVC CR R0 ;
00000828 1 00B0 ;
00000829 1 00B0 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000830 1 00B0 ; PROCEDURE OUT_SINGLE_CHAR 1:(CHAR);
00000831 1 00B0 ; R0 CHAR KEPT
00000832 1 00B0 ; R6 LINK DESTROYED
00000833 1 00B0 ; R7 - DESTROYED
00000834 1 00B0 ;;;;;;;;;;;;;;;;;
00000835 1 00B0 ;OUT_CHAR:
00000836 1 00B0 SB1250: ; REPEAT
00000837 1 00B0 MOV SCHADR R7 ; GET DEVICE ADDRESS;
00000838 1 00B1 MODC SCHSTATUS R7 ;
00000839 1 00B2 SIO R7 R7 ; SENSE(STATUS, DEVICE);
00000840 1 00B3 IBNP 6B2 R7 SCHFRAME ; IF FRAMING ERROR
00000841 1 00B4 IBN & R7 SCHIRDY ; .AND. INPUT_READY
00000842 1 00B5 JMP & SB1290 ;*** THEN BREAK-POINT;
00000843 1 00B6 SBN R7 SCHORDY ; UNTIL OUTPUT_READY HIGH;
00000844 1 00B7 JMP SB1250 ;
00000845 1 00B8 ;
00000846 1 00B8 MOV SCHADR R7 ; RE-ESTABLISH DEVICE ADDRESS;
00000847 1 00B9 WIO R0 R7 ; WRITE TO OC;
00000848 1 00BA MOV R0 R7 ;
00000849 1 00BB XTR R7 8 ; ISOLATE CHAR;
00000850 1 00BC TNE R7 CR ; IF NOT, CARRIAGE_RETURN
00000851 1 00BD JMP X6 ; RETURN

```

```

00000850 1 00B0 TNE CR ;
00000851 1 00B1 JMP X6 ; RETURN
00000852 1 00B2 ; ELSE
00000853 1 00B3 MOV R6 SB12SV ; ESTABLISH RETURN LINKAGE;
00000854 1 00B4 ADDC LF-CR R0 ;
00000855 1 00C0 JMP S6 SB1250 ; OUTCHAR (<LF>);
00000856 1 00C1 ADDC NIL-LF R0 ;
00000857 1 00C2 JMP S6 SB1250 ; OUTCHAR (<NIL>);
00000858 1 00C3 JMP S6 SB1250 ; OUTCHAR (<NIL>);
00000859 1 00C4 JMP S6 SB1250 ; OUTCHAR (<NIL>);
00000860 1 00C5 JMP S6 SB1250 ; OUTCHAR (<NIL>);
00000861 1 00C6 JMP S6 SB1250 ; OUTCHAR (<NIL>);
00000862 1 00C7 ADDC CR-NIL R0 ;
00000863 1 00C8 JMPI SB12SV ; RETURN CALLER;
00000864 1 00C9
00000865 1 00C9
00000866 1 00C9 SB1290: ;BREAK_POINT:
00000867 1 00C9 MOV SCMDR R7 ; GET DEVICE ADDRESS;
00000868 1 00CA MOVC SCMRSET R0 ;
00000869 1 00CB MODC SCMCMD
00000870 1 00CC CIO R0 R7 ; RESET ERROR FLAG;
00000871 1 00CD RIO R0 R7 ;
00000872 1 00CE JMP UT5010 ;
00000873 1 00CF NOLIST ELSE LIST
00000948 1 00CF NOLIST FI LIST
00000949 1 00CF JFF

```

```

00000950 1 00CF ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000951 1 00CF ; PROCEDURE OUT_STRING I:(REF(STRING));
00000952 1 00CF ; R1-3,5 - KEPT
00000953 1 00CF ; R4 LINK DESTROYED
00000954 1 00CF ; R0,6,7 - DESTROYED
00000955 1 00CF ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000956 1 00CF ;OUT_STRING:
00000957 1 00CF SB1300: ; REPEAT
00000958 1 00CF MVP 0. X4 R0 ; GET NEXT SET OF CHARS
00000959 1 00D0 IEQ R0 EOT ; IF END_OF_TEXT
00000960 1 00D2 JMP 1. X4 ; THEN RETURN CALLER;
00000961 1 00D3 JMPI S6 OUTCHAR ; WRITE_CHAR (CHAR);
00000962 1 00D4 SWP R0 ; POSITION 2ND CHAR;
00000963 1 00D5 JMPI S6 OUTCHAR ; WRITE_CHAR (CHAR);
00000964 1 00D6 ADDC 1 R4 ; ADJUST POINTER;
00000965 1 00D7 JMP SB1300 ; UNTIL END_OF_TEXT;
00000966 1 00D8
00000967 1 00D8
00000968 1 00D8 NOLIST IF SCMODE THEN LIST
00000969 1 00D8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000970 1 00D8 ; PROCEDURE BREAK_POINT R:(LINK/LINK+1);
00000971 1 00D8 ; R0 - DESTROYED (IF BREAK ACTIVE)
00000972 1 00D8 ; R1-5 - KEPT
00000973 1 00D8 ; R6 LINK KEPT
00000974 1 00D8 ; R7 - DESTROYED
00000975 1 00D8 ; RETURN:
00000976 1 00D8 ; LINK BREAK DETECTED
00000977 1 00D8 ; LINK+1 NO BREAK DETECTED
00000978 1 00D8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00000979 1 00D8
00000980 1 00D8 SB1350: ;BREAK_POINT:
00000981 1 00D8 MOV SCMAADR R7 ; GET DEVICE_ADDRESS;
00000982 1 00D9 MODE SCMAADR SCMASTATUS
00000983 1 00DA SIO R7 R7 ; SENSE(STATUS, DEVICE);
00000984 1 00DB SBZ R7 SCMAFRAME ; IF FRAMING ERROR
00000985 1 00DC SEN R7 SCMAIRDY ; .AND. INPUT_READY HIGH
00000986 1 00DD JMP 1. X6 ; THEN
00000987 1 00DE MOV SCMAADR R7 ; RE-ESTABLISH DEVICE ADDRESS;
00000988 1 00DF MOVE SCMARESET R0 ;
00000989 1 00E0 MODE SCMAADR SCMACMD
00000990 1 00E1 CIO R0 R7 ; RESET ERROR FLAGS;
00000991 1 00E2 RIO R0 R7 ; EMPTY RECEIVE BUUFER;
00000992 1 00E3 JMP 0. X6 ; RETURN CALLER (ERROR);
00000993 1 00E4 NOLIST ELSE LIST
00001015 1 00E4 NOLIST FI LIST
00001016 1 00E4 JFF

```



```

00001063 1 0100 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001064 1 0100 ; PROCEDURE OUT_HEXADECIMAL I:(BINARY) ;
00001065 1 0100 ; ;
00001066 1 0100 ; R0-1,7 - DESTROYED ;
00001067 1 0100 ; R2 BINARY KEPT ;
00001068 1 0100 ; R3-5 - KEPT ;
00001069 1 0100 ; R6 LINK DESTROYED ;
00001070 1 0100 ; ;
00001071 1 0100 ;CALLS: OUTCHAR ;
00001072 1 0100 ;USING: SB155V ;
00001073 1 0100 ;RETURN: LINK ;
00001074 1 0100 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001075 1 0100 ; ;
00001076 1 0100 SB1500: ;OUT_HEX: ;
00001077 1 0100 MOV R6 SB155V ; ESTABLISH RETURN LINK; ;
00001078 1 0101 MOVC 4 R1 ; FOR I:=1 UNTIL 4 ;
00001079 1 0102 SB1510: ; DO ;
00001080 1 0102 SLC R2 4 ; POSITION LEFT-MOST 4-BIT; ;
00001081 1 0103 MOV R2 R0 ; ;
00001082 1 0104 XTR R0 4 ; ISOLATE PATTERN; ;
00001083 1 0105 IGE R0 10 ; IF ABOVE 10 ;
00001084 1 0106 ADDC <:A:>-<:0:>-10 R0 ; THEN ADJUST; ;
00001085 1 0107 ADDC <:0:> R0 ; MAKE HEXADECIMAL ;
00001086 1 0108 JMPI S6 OUTCHAR ; WRITE_SINGLE_CHAR (CHAR); ;
00001087 1 0109 SOB R1 SB1510 ; END FOR_UNTIL; ;
00001088 1 010A MOVC <: :> R0 ; FORCE SPACE BETWEEN WORDS; ;
00001089 1 010B JMPI S6 OUTCHAR ; ;
00001090 1 010C JMPI SB155V ; RETURN CALLER; ;
00001091 1 010D JFF ;

```

```

00001092 1 010D ;
00001093 1 010D ;
00001094 1 010D ; PROCEDURE GET_ADDRESS 0:(START_ADDRESS, COUNTER);
00001095 1 010D ;
00001096 1 010D ; READS DATA ENTERED FROM KEYBOARD OF FOLLOWING FORMAT:
00001097 1 010D ;      A<PAGE>A  A<BASE_ADDRESS>A.<REL_OFFSET>AA<COUNTER>A
00001098 1 010D ; WHERE
00001099 1 010D ;      <REL_OFFSET>:=<OFFSET>A.<REL_OFFSET>A
00001100 1 010D ; AND INTERPRETES AS FOLLOWS:
00001101 1 010D ;      <START_ADDRESS>:=<ADDRESS>+<REL_OFFSET>
00001102 1 010D ;
00001103 1 010D ; R0 - DELIMITER FOLLOWING ADDRESS_FIELD
00001104 1 010D ; R1,2 - DESTROYED
00001105 1 010D ; R3 - COUNTER (DEFAULT 1)
00001106 1 010D ; R4 - PSW FOR SPECIFIED PAGE (DEFAULT 0)
00001107 1 010D ; R5 - START_ADDRESS
00001108 1 010D ; R6 LINK DESTROYED
00001109 1 010D ; R7 - DESTROYED
00001110 1 010D ;
00001111 1 010D ;CALLS: INCHAR, GETHEXA
00001112 1 010D ;USING: SB16SV
00001113 1 010D ;RETURN: LINK
00001114 1 010D ;
00001115 1 010D ;
00001116 1 010D SB1600: ;GET_ADDRESS:
00001117 1 010D      MOV      R6  SB16SV ;
00001118 1 010E      JMPI     S6  INCHAR ; GET PAGE (IF ANY)
00001119 1 010F      ADDC     0-<:0> R0 ; MAKE DECIMAL
00001120 1 0110      SHS      R0  4 ; IF -1 < PAGE < 4
00001121 1 0111      MOD      R0 ; THEN ADJUST FOR PAGE:
00001122 1 0112      MOV      SB16PAGE R4 ; GET MASKED PSW WITH PAGE BITS;
00001123 1 0113 ;
00001124 1 0113      JMPI     S6  GETHEXA ; GET START ADDRESS;
00001125 1 0114      MOV      R2      R5 ;
00001126 1 0115      SUB      XBASE R5 ; COMPENSATE OWN BASE;
00001127 1 0117 SB1605:
00001128 1 0117      SEQ      R0  <:1> ; DO WHILE DELIMITER="."
00001129 1 0119      JMP      SB1610 ; BEGIN
00001130 1 011A      JMPI     S6  GETHEXA ; READ(HEXA);
00001131 1 011B      ADD      R2      R5 ; ADJUST START_ADDRESS;
00001132 1 011C      JMP      SB1605 ; END DO_WHILE;
00001133 1 011D SB1610: ; END DO_WHILE;
00001134 1 011D      MOVC     1 R2 ; PRESET OPTIONAL '+'-FIELD;
00001135 1 011E      IEQ      R0  <+:> ; IF '+'-FIELD
00001136 1 0120      JMPI     S6  GETHEXA ; THEN GET VALUE;
00001137 1 0121      MOV      R2      R3 ; SET UP LOOP_CONTROL;
00001138 1 0122      JMPI     SB16SV ; RETURN CALLER
00001139 1 0123 ;FF

```

000001140 1 0123 ~~~~~~

```

00001140 1 0123 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001141 1 0123 ; PROCEDURE SET_PARITY
00001142 1 0123 ;
00001143 1 0123 ; R0-5,7 - DESTROYED
00001144 1 0123 ; R6 LINK DESTROYED
00001145 1 0123 ;
00001146 1 0123 ; NOTE - OWN BASE HAS TO BE MULTIPLA 512.
00001147 1 0123 ;
00001148 1 0123 ;CALLS: OUTSTRING [ 3
00001149 1 0123 ;USING: XLOACT, XCAUSE, SB1700:1A
00001150 1 0123 ;RETURN: LINK
00001151 1 0123 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001152 1 0123 ;
00001153 1 0123 SB1700: ;SET_PARITY:
00001154 1 0123 MOV R6 SB1750 ;
00001155 1 0124 MOV R6 SB1750+1 ; RESET PARITY ERROR FLAG;
00001156 1 0125 JMP S4 SB1710 ; BY-PASS LOCAL INTERRUPT ROUTINE;
00001157 1 0126 ;-----;LOCAL_INTERRUPT:
00001158 1 0126 LDS R7 ; RESET BIT_11 (LOCAL INTERRUPT)
00001159 1 0127 IF XXANDS THEN
00001160 1 0127 MOVC 0 CHK=31 00 R1 ;
00001161 1 0128 LDL R1 ; RESET LEVEL
00001162 1 0129 FI
00001163 1 0129 MOV XCAUSE R1 ; GET INTERRUPTING CAUSE;
00001164 1 012A SEQ R1 2 ; IF PARITY ERROR
00001165 1 012B JMP SB1705 ; THEN
00001166 1 012C MOV R1 SB1750+1 ;
00001167 1 012D LDS R6 ;
00001168 1 012E JMP SB1750 ; RETURN MAIN LINE FROM INTER'T;
00001169 1 012F SB1705: ; END SERVING PARITY_ERROR;
00001170 1 012F IER R1 3 ; IF TIMEOUT
00001171 1 0130 JMP SB1760 ; THEN CONTINUE NEXT SECTION;
00001172 1 0131 JMP LOCALACT ; CAUSE NEITHER 2/3
00001173 1 0132 ;
00001174 1 0132 ;-----;
00001175 1 0132 SB1710: ;
00001176 1 0132 MOV R4 XLOACT ; SET-UP OWN LOCAL_ACTION;
00001177 1 0133 ;
00001178 1 0133 MOVC 4 R2 ; FOR I:=0 UNTIL PAGE 3
00001179 1 0134 SB1720: ; DO
00001180 1 0134 MOD. MOD=4 R2 ;
00001181 1 0135 MOV SB16PAGE-1 R6 ; GET CURRENT PSW;
00001182 1 0136 MOVC 128 R3 ; FOR I:=1 UNTIL 128 BLOCKS
00001183 1 0137 SVS R7 ; DO
00001184 1 0138 SB1730: MOV R3 R5 ;
00001185 1 0139 SLL R5 9 ; ESTABLISH CORRECT START;
00001186 1 013A MOVC 512 R4 ; FOR I:=1 UNTIL 512 WORDS
00001187 1 013C LDS R6 412 ; DO
00001188 1 013D SB1740: MOV 0. X5 R0 ; RE-ESTABLISH
00001189 1 013E SB1750: MOV R0 0. X5 ; PARITY;
00001190 1 013F ABDC 1 R5 ; ADJUST POINTER;

```

R2 ← Page pointer

R6 ← ORIG PSW

R3 ← 0-128 blocks

R4 ← 0-512 blocks

R5 ← MEM Pointer

00001182	1	013E	SB1750:	MOV	R0	0.	X5	;	PARITY;	;
00001190	1	013F		ABDC	1		R5	;	ADJUST POINTER;	;
00001191	1	0140		S0B		R4	SB1740	;	UNTIL ALL 1024 WORDS DONE;	;
00001192	1	0141	SB1760:	S0B		R3	SB1730	;	UNTIL ALL 64 MODULES DONE;	;
00001193	1	0142		LDS		R7		;	RETURN "OWN" PAGE;	;
00001194	1	0143		S0B		R2	SB1720	;	UNTIL ALL PAGES DONE;	;
00001195	1	0144		MOV	LOCALACT		R0	;		;
00001196	1	0145		MOV		R0	XLOCACT	;	RE-ESTABLISH;	;
00001197	1	0146		MOV	SB17SV+1		R0	;		;
00001198	1	0147		SEQ		R0	2	;	IF PARITY ERROR	;
00001199	1	0148		JMPI			SB17SV	;	THEN	;
00001200	1	0149		JMPI	S4		OUTSTRING,	;	<:, MEM PARITY ERROR:;>, EOT;	;
00001201	1	0154		JMPI			SB17SV	;	RETURN CALLER;	;
00001202	1	0155						;		;
00001203	1	0155	/FF							

```

00001204 1 0155 ;
00001205 1 0155 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001206 1 0155 ;MEMORY_CHECK UTILITY
00001207 1 0155 ;-----
00001208 1 0155 ; M <PAGE>A <BASE_ADDRESS>A.<OFFSET>AA+<SIZE>A <PATTERN>A <PATTERN2>AA;
00001209 1 0155 ;
00001210 1 0155 ;CALLS: GETADDRESS, GETHEXA, OUTHEXA, OUTNL, OUTSTRING
00001211 1 0155 ;USING: UT13PATTERN#0:2A, PROCESS_DESCRIPTION#0:CONTEXT_SIZE4
00001212 1 0155 ;RETURN: UT5020
00001213 1 0155 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001214 1 0155 UT1300: ;MEMORY:
00001215 1 0155 MOV P5555 R0 ; INITIALIZE TEST PATTERN
00001216 1 0156 MOV R0 UT13PATTERN ;
00001217 1 0157 MOV PAAAA R0 ;
00001218 1 0158 MOV R0 UT13PATTERN+1
00001219 1 0159 JMPI S6 GETADDRESS ; GET BASE_ADDRESS;
00001220 1 015A SNE R0 CR ; IF PATTERN
00001221 1 015B JMP UT1310 ; THEN
00001222 1 015C JMPI S6 GETHEXA ; GET PATTERN
00001223 1 015D MOV R2 UT13PATTERN
00001224 1 015E INE R0 CR ; IF 2ND PATTERN_PARAM
00001225 1 015F JMPI S6 GETHEXA ; GET 2ND PATTERN;
00001226 1 0160 MOV R2 UT13PATTERN+1
00001227 1 0161 UT1310: ; END GETTING PARAMETERS;
00001228 1 0161 JMP S6 UT1315 ;
; 1 0162 UT1350 ; ADDRESS INTERN LOCAL ACTION;
00001229 1 0163 UT1330 ; ADDRESS UPPER PART TEST;
00001230 1 0164 UT1315: MVP 0. X6 R0 ; ESTABLISH INTERNAL
00001231 1 0165 MOV R0 XLOACT ; LOCAL ACTION PROCEDURE;
00001232 1 0166 MVP 1. X6 R0 ; GET LOCATION OF LOWER TEST;
00001233 1 0167 ADD XPROG R0 ; COMPUTE ABS LOCATION;
00001234 1 0169 SUB XBASE R0 ; MAKE REL OWN BASE;
00001235 1 016B MOV R0 UT13PATTERN+2 ;
00001236 1 016C MOV R3 R6 * ; ESTABLISH AREA_SIZE;
00001237 1 016D UT1320: ; REPEAT
00001238 1 016D MOV UT13PATTERN R0 ; GET TEST PATTERN;
00001239 1 016E MOV UT13PATTERN+1 R1 ;
00001240 1 016F MOV UT13PATTERN+2 R2 ; GET OFFSET TO UPPER SERVING;
00001241 1 0170 SVS R7 ; SAVE OWN PSW;
00001242 1 0171 LDS R4 ; SWITCH TO OTHER PAGE
00001243 1 0172 SHS R5 R2 ; IF UPPER PART OF MEMORY
00001244 1 0173 JMP UT1335 ; THEN
00001245 1 0174 MOVL 0. X5 R23 ; SAVE CONTENT DURING TEST;
00001246 1 0175 MOVL R01 0. X5 ; INSERT TEST PATTERN;
00001247 1 0176 MOVC 1024/2 R0 ; REPEAT
00001248 1 0178 SOB R0 LOC ; UNTIL .5-1 MSEC ELAPSED;
00001249 1 0179 MOVL 0. X5 R01 ; GET PATTERN FROM RAM;
00001250 1 017A MOVL R23 0. X5 ; RE-ESTABLISH CONTENT;
00001251 1 017B JMP UT1340 ; ELSE
00001252 1 017C UT1330: NOP ;
00001253 1 017D UT1335: MOVL 0. X5 R23 ; SAVE CONTENT DURING TEST;

```

[illegible]

00001301	1	01BA	UNS	6			RE-ESTABLISH REGS CONTENT;
00001302	1	01BB	UT1390				END SERVING MISMATCH;
00001303	1	01BB	ADDC 2		R5		ADJUST POINTER NEXT SEQUENCE;
00001304	1	01BC	ADDC -1		R6		
00001305	1	01BD	INE	R6	0		
00001306	1	01BE	SOB	R6	UT1320		UNTIL WHOLE MEMORY AREA CHECKED;
00001307	1	01BF	JMP		UT5020		RETURN UTILITIES MAIN LINE;
00001308	1	01C0	FF				


```

00001309 1 01C0 ;-----
00001310 1 01C0 ;
00001311 1 01C0 ; PROJECT: PDS
00001312 1 01C0 ;
00001313 1 01C0 ; MODULE NAME: AMU UTILITIES
00001314 1 01C0 ; MODULE ID NMB: PDS/1000
00001315 1 01C0 ; MODULE VERSION: 0400
00001316 1 01C0 ; MODULE TYPE: SYSTEM
00001317 1 01C0 ; MODULE FILES: UTILITIES.S
00001318 1 01C0 ;
00001319 1 01C0 ;-----
00001320 1 01C0 ; M E R G E   F I L E   U T I L I T I E S
00001321 1 01C0 ; FR/810915
00001322 1 01C0 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001323 1 01C0 ;COPY AND EXECUTE
00001324 1 01C0 ;-----
00001325 1 01C0 ; EA<PAGE>A <PROGRAM_START_ADDRESS> @<NEW_BASE>@<NEW_PROG>AA
00001326 1 01C0 ;
00001327 1 01C0 ;DEFAULTS NEW_BASE=200, NEW_PROG=F000 (BELOW AMU)
00001328 1 01C0 ;
00001329 1 01C0 ;CALLS: COPY(SB1900), GETADDRESS, GETHEXA
00001330 1 01C0 ;USING: UT14SV, PROCESS_DESCRIPTOR#0:CONTEXT_SIZE
00001331 1 01C0 ;RETURN: NO RETURN (ACTIVATES AS OTHER PROCESS)
00001332 1 01C0 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001333 1 01C0 ;
00001334 1 01C0 UT1400: ;COPY AND EXECUTE:
00001335 1 01C0 JMPI S6 GETADDRESS ; GET PROGRAM?_START_ADDRESS;
00001336 1 01C1 MOV XBASE R1 ; SETUP DEFAULT BASE;
00001337 1 01C2 MOVC #F000 R2 ; PROG;
00001338 1 01C4 IEQ R0 CR ; IF DELIMITER=CR
00001339 1 01C5 JMP UT1410 ; THEN DEFAULTS
00001340 1 01C6 JMPI S6 GETHEXA ; ELSE
00001341 1 01C7 MOV R2 UT14SV ; GET NEW_BASE;
00001342 1 01C8 MOVC #F000 R2 ; SETUP DEFAULT PROG;
00001343 1 01CA SEQ R0 CR ; IF REQUEST FOR NEW_PROG
00001344 1 01CB JMPI S6 GETHEXA ; THEN GET ENTERED ADDRESS;
00001345 1 01CC MOV UT14SV R1 ; END IF_THEN_ELSE;
00001346 1 01CD UT1410:
00001347 1 01CD MOV R2 R6 ; NEW_PROGRAM ADDRESS;
00001348 1 01CE SUB XBASE R2 ; REL OWN BASE;
00001349 1 01D0 SVP
00001350 1 01D1 MOVC 0 R0 ; SET UP FOR MEMORY_SECTION 0;
00001351 1 01D2 SVS R7 ; SAVE OWN PAGE;
00001352 1 01D3 LDS R4 ; SWITCH TO EXTERNAL MODULE
00001353 1 01D4 MOV AMUPROCESS+1.X5 R3 ; GET SIZE OF MODULE;
00001354 1 01D5 LDS R7 ; RETURN OWN PAGE;
00001355 1 01D6 JMPI S6 COPY ; COPY TO NEW AREA;
00001356 1 01D7 MOVC XRO R7 ;
00001357 1 01D8 UNS 6 ; RE-ESTABLISH REG CONTENT;
00001358 1 01D9
00001359 1 01D9 MOV R2 R5 ;
00001360 1 01DA ADD AMUPROCESS.X5 R5 ; POINT PROCESS DESCRIPTOR;

```

00001356	1	01D7	MOV	XR0	R7	;	
00001357	1	01D8	UN		6	;	RE-ESTABLISH REG CONTENT;
00001358	1	01D9				;	
00001359	1	01DA	MOV	R2	R5	;	
00001360	1	01DB	ADD	AMUPROCESS, X5	R5	;	POINT PROCESS DESCRIPTOR;
00001361	1	01DC	MOV	R1 XR3.	X5	;	SETUP REG3 WITH NEW BASE;
00001362	1	01DD	MOV	R6 XR5.	X5	;	SETUP REG5 WITH NEW PROG;
00001363	1	01DE	MOV	R1 XBASE.	X5	;	SETUP BASE
00001364	1	01DF	MOV	R1 XMOD.	X5	;	AND MOD_REGISTER;
00001365	1	01E0	MOVL	XPROG, X5	R01	;	XPRPC:= XPRPC
00001366	1	01E1	SUB	R0	R1	;	-DEFAULT_PROG
00001367	1	01E2	ADD	R6	R1	;	+NEW_PROG;
00001368	1	01E3	MOV	R6	R0	;	XPROG:= NEW_PROG;
00001369	1	01E4	MOVL	R01 XPROG.	X5	;	
00001370	1	01E5				;	
00001371	1	01E6	ADD	XBASE	R5	;	POINT PROCESS DESCRIPTOR;
00001372	1	01E7	IF AMOS THEN			;	
00001373	1	01E8	LDP		R5	;	ACTIVATE PROCESS;
00001374	1	01E9	ELSE			;	
00001375	1	01EA	SLP		R5	;	ACTIVATE PROCESS;
00001376	1	01EB				;	
00001377	1	01EC				;	
00001378	1	01ED	IF			;	

```

00001379 1 01E7 ; FR/B11021 ;
00001380 1 01E7 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001381 1 01E7 ;DUMP;
00001382 1 01E7 ;----;
00001383 1 01E7 ; DA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A <END_ADDRESS>;
00001384 1 01E7 ; DA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A+<SIZE>;
00001385 1 01E7 ; DA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A*(<RE-READS>;
00001386 1 01E7 ;
00001387 1 01E7 ;CALLS: GETADDRESS, GETHEXA, OUTCHAR, OUTHEXA, OUTNL;
00001388 1 01E7 ;USING: PROCESS_DESCRIPTOR@0:CONTEXT_SIZEA <ONLY REREAD PART>;
00001389 1 01E7 ;RETURN: UT5010;
00001390 1 01E7 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001391 1 01E7 ;
00001392 1 01E7 UT1500: ;DUMP:
00001393 1 01E7 JMPI S6 GETADDRESS ; GET START_ADDRESS, AREA_SIZE;
00001394 1 01E8 IEQ R0 <:~> ; IF REQUEST FOR REREAD;
00001395 1 01EA JMP UT1575 ; THEN SERVE AS SUCH;
00001396 1 01EB ADD R5 R3 ; COMPUTE ENDING ADDRESS;
00001397 1 01EC IEQ R0 CR ; IF END_ADDRESS;
00001398 1 01ED JMP UT1520 ; THEN;
00001399 1 01EE JMPI S6 GETHEXA ; GET ENDING ADDRESS;
00001400 1 01EF MOV R2 R3 ;
00001401 1 01F0 MODN XBASE;
00001402 1 01F1 ADDC 1 R3 ; MAKE REL "OWN" BASE;
00001403 1 01F2 UT1520: ; END SERVING END_ADDRESS;
00001404 1 01F2 SRL R5 4 ;
00001405 1 01F3 SLL R5 4 ; ROUND START ADDRESS;
00001406 1 01F4 SUB R5 R3 ; FOR I:=1 UNTIL NO_OF_WORDS;
00001407 1 01F5 UT1540: ; DO;
00001408 1 01F5 MOV R5 R0 ;
00001409 1 01F6 XTR R0 3 ; COMPUTE MODULO 8;
00001410 1 01F7 IEQP R0 8 ; IF (ADDRESS MOD 8) .EQ. 8;
00001411 1 01F8 MOVC <:~> R0 ; THEN;
00001412 1 01F9 JMPI S6 OUTCHAR ; INSERT XTRA SPACE;
00001413 1 01FA JON R0 UT1560 ; IF (ADDRESS MOD 16) .EQ. 0;
00001414 1 01FB JMPI S6 OUTNL ; WRITE_NEWLINE;
00001415 1 01FC MOV XBASE R2 ;
00001416 1 01FD ADD R5 R2 ; COMPUTE ADDRESS NEXT DUMP;
00001417 1 01FE JMPI S6 OUTHEXA ; WRITE_HEXADECIMAL (ADDRESS);
00001418 1 01FF UT1560: ; END IF NEW LINE;
00001419 1 01FF SVS R7 ; SAVE OWN PSW;
00001420 1 0200 LDS R4 ; SWITCH TO RELEVANT PAGE;
00001421 1 0201 MOV 0. X5 R2 ; GET NEXT WORD;
00001422 1 0202 LDS R7 ; RE-ESTABLISH OWN PSW;
00001423 1 0203 JMPI S6 OUTHEXA ; WRITEN;
00001424 1 0204 ADDC 1 R5 ; ADJUST POINTER;
00001425 1 0205 SOB R3 UT1540 ; END FOR_UNTIL;
00001426 1 0206 JMP UT1590 ; RETURN UTILITIES_MAINLINE;
00001427 1 0207 ;
00001428 1 0207 UT1575: ;REREAD:;
00001429 1 0207 JMPL S6 GETHEXA ; GET REREAD_COUNT;
00001430 1 0208 MOVC 0 R3 ; INITIALIZE MOST SIGNIFICANT PART;

```

00001426	1	0206	JMP		UT1590		RETURN UTILITY1590:MAINLINE;
00001427	1	0207					
00001428	1	0207			UT1575:		;/REREAD;
00001429	1	0207	JMPI	S6	GETHEXA		GET REREAD_COUNT;
	1	0208	MOV	0		R3	INITIALIZE MOST SIGNIFICANT PART;
00001430	1	0209	IEQP		<:L:>		IF REQUEST FOR LONG INTEGER
00001431	1	0208	MOV	&		R3	MAKE LEAST TO MOST SIGNIFICANT;
	1	020C	JMPI	&	GETHEXA		AND GET LEAST SIGNIFICANT PART;
00001432	1	020D	MOV			R0	SET UP LEAST SIGNIFICANT COUNT;
00001433	1	020E	ADDC	1		R3	
00001434	1	020F	MOV	PRINTOUT		R1	GET PRINT-OUT FLAG;
00001435	1	0210	SUS			R7	SAVE OWN STATUS (PAGE);
00001436	1	0211	LDS				SWITCH TO REQUESTED PAGE;
00001437	1	0212					FOR I:=1 UNTIL REREAD_COUNT
00001438	1	0212	MOV			R2	READ FROM MEMORY
00001439	1	0213	SEQ		TRUE		IF PRINT-OUT REQUESTED
00001440	1	0214	JMP		UT1585		THEN
00001441	1	0215	LDS			R7	RETURN OWN PAGE;
00001442	1	0216	SVP				
00001443	1	0217	JMPI		OUTHEXA		
	1	0218	MOV	XR0		R7	POINT TOP PROCESS_DESCRIPTOR;
00001444	1	0219	UNS		6		RE-ESTABLISH REGS;
00001445	1	021A	SUS			R7	
00001446	1	021B	LDS			R4	SWITCH TO REQUESTED PAGE;
00001447	1	021C					END PRINT-OUT;
00001448	1	021C	SOB		UT1580	R0	
00001449	1	021D	SOB		UT1580	R3	END FOR_UNTIL;
00001450	1	021E	LDS			R7	RETURN OWN PAGE;
00001451	1	021F	JMP		UT5010		RETURN MAIN LINE;
00001452	1	0220					
00001453	1	0220	;/FF				

```

00001454 1 0220 ;
00001455 1 0220 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001456 1 0220 ;PATCH
00001457 1 0220 ;-----
00001458 1 0220 ; PA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A @<PATCH_DATA>A
00001459 1 0220 ; PA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A+<SIZE> <PATTERN>
00001460 1 0220 ; PA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>A*<REWRITES> <PATTERN>
00001461 1 0220 ;
00001462 1 0220 ;USING: UT16SV
00001463 1 0220 ;CALLS: GETADDRESS, GETHEXA
00001464 1 0220 ;RETURN: UT5020
00001465 1 0220 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001466 1 0220 ;
00001467 1 0220 UT1600: ;PATCH:
00001468 1 0220 JMPI S6 GETADDRESS ; GET START_ADDRESS, AREA_SIZE;
00001469 1 0221 IER R0 <:!:> ; IF REQUEST FOR REWRITES
00001470 1 0223 JMP UT1670 ; THEN SERVE ACCORDINGLY;
00001471 1 0224 IER R3 1 ; IF MULTIPATCH
00001472 1 0225 JMP UT1660 ; THEN
00001473 1 0226 JMPI S6 GETHEXA ; GET HEXA (PATCH_PATTERN);
00001474 1 0227 UT1620: ; FOR I:=1 UNTIL COUNT
00001475 1 0227 SVS R7 ; SAVE OWN PSW;
00001476 1 0228 LDS R4 ; SWITCH TO RELEVANT PAGE;
00001477 1 0229 IER R1 TRUE ; IF VALID PATCH DATA
00001478 1 022A MOV R2 0. X5 ; PATCH;
00001479 1 022B LDS R7 ; RE-ESTABLISH OWN PSW;
00001480 1 022C ADDC 1 R5 ; ADJUST POINTER;
00001481 1 022D SOB R3 UT1620 ; END MULTI-PATCH LOOP;
00001482 1 022E JMP UT1690 ; ELSE
00001483 1 022F UT1640: ; REPEAT
00001484 1 022F JMPI S6 GETHEXA ; GET NEXT PATCH VALUE
00001485 1 0230 SVS R7 ; SAVE OWN PSW;
00001486 1 0231 LDS R4 ; SWITCH TO RELEVANT PAGE;
00001487 1 0232 IER R1 TRUE ; IF PATCH DATA
00001488 1 0233 MOV R2 0. X5 ; REPLACE OLD VALUE;
00001489 1 0234 LDS R7 ; RE-ESTABLISH OWN PSW;
00001490 1 0235 ADDC 1 R5 ; ADJUST POINTER;
00001491 1 0236 UT1660: INE R0 CR ;
00001492 1 0237 JMP UT1640 ; UNTIL LAST_CHAR .EQ. CR/NL;
00001493 1 0238 JMP UT1690 ; END SERVING NORMAL PATCH;
00001494 1 0239
00001495 1 0239 UT1670: ;REWRITE:
00001496 1 0239 JMPI S6 GETHEXA ; GET AREA_SIZE;
00001497 1 023A MOVCL 0 R3 ; INITIALIZE MOST SIGNIFICANT;
00001498 1 023B IEOP R0 <:!:> ; IF REQUEST FOR LEAST SIGN
00001499 1 023D MOV & R2 R3 ; THEN SET UP
00001500 1 023E JMPI & S6 GETHEXA ; AND GET LEAST SIGNIFICANT;
00001501 1 023F MOV R2 UT16SV ; SAVE LEAST SIGNIFICANT PART;
00001502 1 0240 JMPI S6 GETHEXA ; GET WRITE_PATTERN;
00001503 1 0241 MOV UT16SV R0 ; RE-ESTABLISH LEAST SIGNIFICANT;
00001504 1 0242 ADDCL 1 R3 ;
00001505 1 0243 SVS R7 ; SAVE OWN STATUS;

```

00001501	1	023F	MOV		R2	UT1680	;	SAVE LEAST SIGNIFICANT PART;	;
00001502	1	0240	JMPI		S6	GETHEXA	;	GET WRITE PATTERN;	;
00001503	1	0241	MOV	UT1680		R0	;	RE-ESTABLISH LEAST SIGNIFICANT;	;
00001504	1	0242	ADDC	1		R3	;		;
00001505	1	0243	SVS			R7	;	SAVE OWN STATUS;	;
00001506	1	0244	LDS		R4		;	SWITCH TO REQUESTED PAGE;	;
00001507	1	0245	UT1680:				;	FOR I:=1 UNTIL AREA_SIZE	;
00001508	1	0245	MOV		R2	X5	;	WRITE PATTERN TO MEMORY;	;
00001509	1	0246	SOB		R0	UT1680	;		;
00001510	1	0247	SOB		R3	UT1680	;	END;	;
00001511	1	0248	LDS		R7		;	RE-ESTABLISH OWN PSW;	;
00001512	1	0249	UT1690:	JMP		UT5020	;	RETURN UTILITIES_MAINLINE;	;
00001513	1	024A	JFF						

```

00001514 1 024A ; FR/810915 ;
00001515 1 024A ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001516 1 024A ;SEARCH ;
00001517 1 024A ;----- ;
00001518 1 024A ; SA<PAGE>A <BASE_ADDRESS>A.<OFF-SET>AA+<SIZE>A <PATTERN>A/<MASK>A ;
00001519 1 024A ; ;
00001520 1 024A ;USING: UT17SV, PROCESS_DESCRIPTORA0:CONTEXT_SIZEA ;
00001521 1 024A ;CALLS: GETADDRESS, GETHEXA, OUTHEXA, OUTNL, OUTSTRING ;
00001522 1 024A ;RETURN: UT5020 ;
00001523 1 024A ; ;
00001524 1 024A ;NOTE: NEGATED SEARCH IF "/" BUT NO MASK IS SPECIFIED ;
00001525 1 024A ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001526 1 024A ; ;
00001527 1 024A UT1700: ;
00001528 1 024A JMPI S6 GETADDRESS ; GET START_ADDRESS, AREA_SIZE ;
00001529 1 024B JMPI S6 GETHEXA ; GET SEARCH PATTERN ;
00001530 1 024C MOV R2 UT17SV ; SAVE DURING GETTING MASK ;
00001531 1 024D MOVC #FFFF R2 ; SET UP DEFAULT: ALL BITS ;
00001532 1 024E IER R0 <:/> ; IF REQUEST FOR MASK ;
00001533 1 0250 JMPI S6 GETHEXA ; THEN GET IT ;
00001534 1 0251 MOV UT17SV R0 ; RE-ESTABLISH LOOK_FOR_PATTERN ;
00001535 1 0252 SVS R7 ; SAVE OWN PSW (FOR PAGE RETURN) ;
00001536 1 0253 LDS R4 ; SWITCH TO SEARCH PAGE ;
00001537 1 0254 IER R2 0 ; IF REQUEST FOR NEGATED SEARCH ;
00001538 1 0255 JMP UT1750 ; THEN SERVE AS SUCH ;
00001539 1 0256 ; ;
00001540 1 0256 UT1720: ; REPEAT ;
00001541 1 0256 MOV X5 R6 ; GET MEMORY WORD ;
00001542 1 0257 AND R2 R6 ; ISOLATE RELEVANT PORTION ;
00001543 1 0258 SEQ R0 R6 ; IF MATCH ;
00001544 1 0259 JMP UT1740 ; THEN ;
00001545 1 025A LDS R7 ; RE-ESTABLISH OWN PSW ;
00001546 1 025B SVP ; SAVE REGS DURING PRINT-OUT ;
00001547 1 025C JMPI S4 OUTSTRING, <:MATCH AT :>, EOT ;
00001548 1 0263 MOV XBASE R2 ; ;
00001549 1 0264 ADD R5 R2 ; COMPUTE PATTERN_ADDRESS ;
00001550 1 0265 JMPI S6 OUTHEXA ; ;
00001551 1 0266 JMPI S6 OUTNL ; ;
00001552 1 0267 MOVC XRO R7 ; POINT TOP PROC_DESCRIPTOR ;
00001553 1 0268 UNS 6 ; RE-ESTABLISH REGS ;
00001554 1 0269 SVS R7 ; SAVE OWN PSW ;
00001555 1 026A LDS R4 ; SWITCH TO SEARCH PAGE ;
00001556 1 026B UT1740: ; END SERVING MATCH ;
00001557 1 026B ADDC 1 R5 ; ADJUST POINTER ;
00001558 1 026C SOB R3 UT1720 ; UNTIL END OF AREA ;
00001559 1 026D JMP UT1799 ; END SEARCH FOR MATCHES ;
00001560 1 026E ; ;
00001561 1 026E ; ;
00001562 1 026E ;NEGATED_SEARCH: ;
00001563 1 026E UT1750: ; REPEAT ;
00001564 1 026E SNE X5 R0 ; IF MISMATCH ;
00001565 1 026E JMP UT1760 ; THEN ;

```

Address	Hex	Instruction	Register	Comment
00001561	1 026E			NEGATED SEARCH;
00001562	1 026E			REPEAT
00001563	1 026E	UT1759:		IF MISMATCH
00001564	1 026E	SNE	X5	THEN
00001565	1 026F	JMP	UT1760	RETURN OWN PAGE;
00001566	1 0270	LDS	R7	SAVE REGS DURING PRINT-OUT;
00001567	1 0271	SVP		<:MISMATCH AT :>, EOT
00001568	1 0272	JMPI	S4	OUTSTRING,
00001569	1 027A	MOV	XBASE	R2
00001570	1 027B	ADD	R5	R2
00001571	1 027C	JMPI	S6	OUTHEXA
00001572	1 027D	JMPI	S6	OUTNL
00001573	1 027E	MOVC	XR0	R7
00001574	1 027F	UNS	6	
00001575	1 0280	SVS		R7
00001576	1 0281	LDS	R4	
00001577	1 0282	UT1760:		
00001578	1 0282	ADDC	1	R5
00001579	1 0283	SOB	R3	UT1750
00001580	1 0284	UT1799:LDS	R7	
00001581	1 0285	JMP	UT5020	
00001582	1 0286			
00001583	1 0286			
00001584	1 0286	;		FF


```

00001585 1 0286 ; FR/800303 ;
00001586 1 0286 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001587 1 0286 ;EXECUTE ;
00001588 1 0286 ;----- ;
00001589 1 0286 ; X <PROCESS_BASE_ADDRESS> ;
00001590 1 0286 ; ;
00001591 1 0286 ;CALLS: GETADDRESS ;
00001592 1 0286 ;RETURN: NO RETURN (ACTIVATES AS OTHER PROCESS) ;
00001593 1 0286 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001594 1 0286 ; ;
00001595 1 0286 UT1800: ;
00001596 1 0286 JMPI S6 GETADDRESS ; GET PAGE/BASE OF PROCESS ;
00001597 1 0287 MOD ADD XBASE R5 ; COMPUTE BASE_ADDRESS ;
00001598 1 0287 IF AMOS THEN ;
00001599 1 0289 LDN R5 ; ACTIVATE PROCESS ;
00001600 1 0289 ELSE ;
00001601 1 0289 SLN R5 ; ACTIVATE PROCESS ;
00001602 1 028A FI ;
00001603 1 028A JFF ;

```

```

FR/800303 ;
00001604 1 028A ;
00001605 1 028A ;
00001606 1 028A ;COPY
00001607 1 028A ;-----
00001608 1 028A ; CR(PAGE)A <SOURCE>A+<SIZE>A <OBJECT>A/<PAGE>A
00001609 1 028A ;
00001610 1 028A ; <SOURCE>/<OBJECT>:= <BASE_ADDRESS>A.<OFFSET>A
00001611 1 028A ;
00001612 1 028A ;CALLS: GETADDRESS, GETHEXA, INCHAR, COPY(SB1900)
00001613 1 028A ;RETURN: UT5010
00001614 1 028A ;
00001615 1 028A ;
00001616 1 028A UT1900: ;COPY_MEMORY
00001617 1 028A JMPI S6 GETADDRESS ; GET PAGE/SOURCE_ADDRESS/COUNT
00001618 1 028B JMPI S6 GETHEXA ; GET OBJECT_ADDRESS
00001619 1 028C SUB XBAGE R2 ; ADJUST FOR OWN BASE
00001620 1 028E IER R0 <:/:/> ; IF PAGE TO BE SPECIFIED
00001621 1 0290 JMPI S6 INCHAR ; GET SPECIFIED PAGE
00001622 1 0291 ADDC -<:0:> R0 ;
00001623 1 0292 JMP S6 SB1900 ; COPY SOURCE TO OBJECT_AREA
00001624 1 0293 JMP UT5010 ; RETURN UTILITIES_MAINLINE
00001625 1 0294 ;
00001626 1 0294 ;
00001627 1 0294 ;
00001628 1 0294 ; PROCEDURE COPY 1:(REF(SOURCE), REF(OBJECT), PAGE);
00001629 1 0294 ; R0 OBJECT_MEMORY_SECTION DESTR
00001630 1 0294 ; R1 - DESTR
00001631 1 0294 ; R2 REF(START_OBJECT_AREA) DESTR
00001632 1 0294 ; R3 WORDS TO COPY DESTR
00001633 1 0294 ; R4 PSW FOR SOURCE_AREA KEPT
00001634 1 0294 ; R5 REF(START_SOURCE_AREA) DESTR
00001635 1 0294 ; R6 LINK KEPT
00001636 1 0294 ; R7 - DESTR
00001637 1 0294 ;
00001638 1 0294 ;READS: SB16PAGE#0,1,2,3A
00001639 1 0294 ;RETURN: LINK
00001640 1 0294 ;
00001641 1 0294 ;
00001642 1 0294 SB1900: ; COPY
00001643 1 0294 SHS R0 4 ; IF VALID SPECIFIED PAGE
00001644 1 0295 MOD R0 ; USE IT
00001645 1 0296 MOV SB16PAGE R0 ; SET-UP RELEVANT PSW_PATTERN
00001646 1 0297 SVS R7 ; SAVE OWN PSW
00001647 1 0298 UT1950: ; REPEAT
00001648 1 0298 LDS R4 ; SWITCH TO SOURCE_SECTION
00001649 1 0299 MOV 0. X5 R1 ; GET WORD
00001650 1 029A LDS R0 ; SWITCH TO ALTERNATIVE_PAGE
00001651 1 029B MOV R1 X2 ;
00001652 1 029C INCD R5 R2 ; ADJUST ADDRESSES
00001653 1 029D STD R3 UT1950 ; UNTIL_SOURCE_AREA_COPYEO
00001654 1 029E LDS R7 ; RE-ESTABLISH "OWN" MEM_SECTION

```

00001657	1	029D	SDR	R3	UNTIL SOURCE AREA DOPKE01	;
00001654	1	029E	LDS	R7	RE-ESTABLISH "OWN" MEML SECTION	;
00001655	1	029F	JMP	X6	RETURN CALLER	;
00001658	1	02A0	;			

```

FR/800303 ;
00001657 1 02A0 ;
00001658 1 02A0 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001659 1 02A0 ;IO-UTILITIES
00001660 1 02A0 ;-----
00001661 1 02A0 ; I ?S,R?  @((MODLPATTERN))A  <DEV_ADDRESS>#<REPEATS>A
00001662 1 02A0 ; I ?C,H?  @((MODLPATTERN))A  <DEV_ADDRESS>#<REPEATS>A  <OUTPUT_WORD>;
      1 02A0 ; I ?K,V?  @((MODLPATTERN))A  <DEV_ADDRESS>#<REPEATS>A  <OUTPUT_WORD>;
00001663 1 02A0 ;
00001664 1 02A0 ;CALLS:  BREAKCHECK, GETADDRESS, GETHEXA, INCHAR, OUTHEXA, OUTNL
00001665 1 02A0 ;      OUTSTRING
00001666 1 02A0 ;USING:  UT20SV#0:2A
00001667 1 02A0 ;RETURN: UT5010
00001668 1 02A0 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001669 1 02A0 ;
00001670 1 02A0 UT2000: ;IO:
00001671 1 02A0      MOV  0      R3 ;
00001672 1 02A1      MOV      R3  UT20SV+1 ;  DEFAULT NO MODIFICATION;
00001673 1 02A2      MOV      R3  UT20SV+2 ;  RESET LOOP_COUNT (LONG_INTEGER);
00001674 1 02A3      JMPI      S6  OUTSTRING,  <:0-:), EOT
00001675 1 02A6      JMPI      S6  INCHAR      ;  GET_IO_TYPE;
00001676 1 02A7      ADDC  -<:0:)      R0 ;
00001677 1 02A8      MOV      R0  UT20SV ;
00001678 1 02A9 ;
00001679 1 02A9      JMPI      S6  GETADDRESS ;
00001680 1 02AA      SEQ      R0  <:):) ;  IF MODIFICATION ACTIVE
00001681 1 02AC      JMP      UT2010 ;  THEN
00001682 1 02AD      ADD      XBASE      R5 ;  RE-ESTABLISH ENTERED DATA;
00001683 1 02AF      MOV      R5  UT20SV+1 ;
00001684 1 02B0      JMPI      S6  GETADDRESS ;  GET DEVICE_ADDRESS, REPEATS;
00001685 1 02B1 UT2010: ;  END GETTING MODIFICATION;
00001686 1 02B1      ADD      XBASE      R5 ;  BEWARE WAS BASE_REL;
00001687 1 02B3      SEQ      R0  <:~:~) ;  IF LONG_INTEGER
00001688 1 02B5      JMP      UT2020 ;  THEN
00001689 1 02B6      JMPI      S6  GETHEXA ;
00001690 1 02B7      IEQ      R2  0 ;  IF LEAST SIGNIFICANT PART 0
00001691 1 02B8      ADDC  -1      R3 ;  THEN ADJUST MOST SIGN;
00001692 1 02B9      MOV      R3  UT20SV+2 ;  PREPARE MOST SIGNIFICANT PART;
00001693 1 02BA      MOV      R2  R3 ;  SET UP LEAST SIGNIFICANT PART;
00001694 1 02BB UT2020: ;  END GETTING LONG INTEGER;
00001695 1 02BB      MOV      UT20SV      R4 ;
00001696 1 02BC ;
00001697 1 02BC      IEQ  R4  <:S:)-<:0:), JMP  UT2050
00001698 1 02BE      IEQ  R4  <:R:)-<:0:), JMP  UT2060
00001699 1 02C0      JMPI      S6  GETHEXA ;  GET ADDITIONAL PARAMETER;
00001700 1 02C1      IEQ  R4  <:C:)-<:0:), JMP  UT2070
00001701 1 02C3      IEQ  R4  <:K:)-<:0:), JMP  UT2075
00001702 1 02C5      IEQ  R4  <:W:)-<:0:), JMP  UT2080
00001703 1 02C7      IEQ  R4  <:V:)-<:0:), JMP  UT2085
00001704 1 02C9      JMPI      ERROR ;
00001705 1 02CA ;
00001706 1 02CA UT2050: ;SID:

```

```

00001705 1 02CA 00001706 1 02CA UT2050: ;SID:
00001707 1 02CA JMP S6 UT2090 ; S
00001708 1 02CB MOD R7 ;
00001709 1 02CC SID R2 R5 ;
00001710 1 02CD JMP UT2140 ;
00001711 1 02CE UT2060: ;RIO:
00001712 1 02CE JMP S6 UT2090 ; R
00001713 1 02CF MOD R7 ;
00001714 1 02D0 RIO R2 R5 ;
00001715 1 02D1 JMP UT2140 ;
00001716 1 02D2 UT2070: ;CIO:
00001717 1 02D2 JMP S6 UT2090 ; C
00001718 1 02D3 MOD R7 ;
00001719 1 02D4 CIO R2 R5 ;
00001720 1 02D5 JMP UT2180 ;
00001721 1 02D6 UT2075: ;CIO: SINGLE_STEP
00001722 1 02D6 JMP S6 UT2090 ; K
00001723 1 02D7 MOD R7 ;
00001724 1 02D8 CIO R2 R5 ;
00001725 1 02D9 JMP UT2180 ;
00001726 1 02DA UT2080: ;WIO:
00001727 1 02DA JMP S6 UT2090 ; H
00001728 1 02DB MOD R7 ;
00001729 1 02DC WIO R2 R5 ;
00001730 1 02DD JMP UT2180 ;
00001731 1 02DE UT2085: ;WIO: SINGLE_STEP
00001732 1 02DE JMP S6 UT2090 ; V
00001733 1 02DF MOD R7 ;
00001734 1 02E0 WIO R2 R5 ;
00001735 1 02E1 JMP UT2180 ;
00001736 1 02E2 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001737 1 02E2 UT2090: ;COMMON_INITIALIZATION:
00001738 1 02E2 MOVC 0 R4 ;
00001739 1 02E3 MOV UT20SV+1 R7 ;
00001740 1 02E4 SNE R7 0 ; IF NO MODIFICATION
00001741 1 02E5 ADDC 1 R6 ; THEN ADJUST POINTER;
00001742 1 02E6 MOV R6 UT20SV ;
00001743 1 02E7 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001744 1 02E7 UT2100: ; REPEAT
00001745 1 02E7 MOV UT20SV+1 R7 ; GET MODIFICATION REGISTER;
00001746 1 02E8 JMPI UT20SV ; EXECUTE RELEVANT IO_INSTR;
00001747 1 02E9 ;
00001748 1 02E9 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001749 1 02E9 UT2140: ;WRITE_IO_RESULTS:
00001750 1 02E9 MOV PRINTOUT R7 ;
00001751 1 02EA SED R7 TRUE ; IF PRINT_OUT
00001752 1 02EB JMP UT2180 ; THEN
00001753 1 02EC JMPI S6 OUTHEXA ; DUMP STATUS
00001754 1 02ED ADDC 1 R4 ; MAINTAIN DATA COUNT
00001755 1 02EE MOV R4 R0 ;
00001756 1 02EF XTR R0 3 ; ISOLATE BUT LOWER 3;
00001757 1 02F0 ISZ R0 0 ; IF (COUNT MOD 3) EQ 0
00001758 1 02F1 JMPI S6 OUTNL ; THEN TIME FOR NEW LINE;

```

00001754	1	02ED	ADDC	1		R4		R4	/	MAINTAIN DATA COUNT	/
00001755	1	02EE	MOV			R0	3		/	ISOLATE BUT LOWER 3;	/
00001756	1	02EF	XTR			R0	0		/	IF (COUNT MOD 8) EQ 0	/
00001757	1	02F0	IEQ						/	THEN TIME FOR NEW LINE;	/
00001758	1	02F1	JMPI			S6	OUTNL		/		/
00001759	1	02F2	JMP				UT2180		/		/
00001760	1	02F3	UT2160:						/	GET_NEXT_PARAMETER;	/
00001761	1	02F3	JMPI			S6	GETHEXA		/		/
00001762	1	02F4	UT2180:						/		/
00001763	1	02F4	SOB			R3	UT2100		/	PERFORM LEAST_SIGN_LOOP_CONTROL;	/
00001764	1	02F5	JMPI			S6	BREKCHECK		/	BREAK_CHECK	/
00001765	1	02F6	JMP				UT2190		/	R:(BREAK OCCURED);	/
00001766	1	02F7	MOV	UT20SV+2		R6			/	GET MOST SIGNIFICANT PART	/
00001767	1	02F8	ADDC	-1		R6			/	MAINTAIN COUNTER;	/
00001768	1	02F9	INEF			R6	FFFF		/		/
00001769	1	02FA	MOV			R6	UT20SV+2		/		/
00001770	1	02FB	JMP				UT2100		/	UNTIL LONG_INTER ZEROED; (-1);	/
00001771	1	02FC	UT2190:						/	END_OF_IO_UTILITY;	/
00001772	1	02FC	JMP				UT5010		/	RETURN UTILITIES_MAIN_LINE;	/
00001773	1	02FD	JFF						/		/

```

00001774 1 02FD ; FR/800303 ;
00001775 1 02FD ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001776 1 02FD ;LOOP UTILITY ;
00001777 1 02FD ;----- ;
00001778 1 02FD ; *?F,<CR>? <CR> - BREAKABLE ;
00001779 1 02FD ; F - NON-BREAKABLE ;
00001780 1 02FD ; ;
00001781 1 02FD ; THIS UTILITY ALLOWS FOR A CONTINUOUS LOOP ON ANY INSTRUCTION AS ;
00001782 1 02FD ; SPECIFIED BY THE OPERATOR. ;
00001783 1 02FD ; IT IS THE RESPONSIBILITY OF THE OPERATOR TO PROPERLY SET UP THE ;
00001784 1 02FD ; REGISTERS BY PATCHING HIS PROPER INITIAL VALUES INTO THE BASE ;
00001785 1 02FD ; REGISTERS OF THE AMU PROCESS: REGS 0-4, 6, 7. ;
00001786 1 02FD ; REG 5 MUST CONTAIN THE INSTRUCTION WHICH THE OPERATOR WANTS TO ;
00001787 1 02FD ; LOOP UPON. ;
00001788 1 02FD ; THE LOOP MAY BE CANCELLED BY BREAK. ;
00001789 1 02FD ; ;
00001790 1 02FD ; NOTE NONE OF THE REGISTERS ARE RE-INITIALIZED IN FAST MODE, ;
00001791 1 02FD ; NEITHER CAN THE OPERATION BE BREAK'D. ;
00001792 1 02FD ; ;
00001793 1 02FD ;CALLS: INCHAR, BREAKCHECK(BREAKABLE LOOP ONLY) ;
00001794 1 02FD ;READS: XRO-XR7 OF BASE PROCESS DESCRIPTOR ;
00001795 1 02FD ;RETURN: UTILITIES ON "BREAK" IF ALLOWED ;
00001796 1 02FD ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00001797 1 02FD UT2300: ;INSTRUCTION_LOOP: ;
00001798 1 02FD JMPI S6 INCHAR ; GET TYPE OF LOOP ;
00001799 1 02FE SNE R0 <:F: > ; IF RE-INITIALIZING LOOP THEN ;
00001800 1 0300 JMP UT2350 ; BEGIN ;
00001801 1 0301 UT2320: ; REPEAT ;
00001802 1 0301 MOV XRO R0 ; INITIALIZE REGS; ;
00001803 1 0302 MOV XR1 R1 ; ;
00001804 1 0303 MOV XR2 R2 ; ;
00001805 1 0304 MOV XR3 R3 ; ;
00001806 1 0305 MOV XR4 R4 ; ;
00001807 1 0306 MOV XR6 R6 ; ;
00001808 1 0307 MOV XR7 R7 ; ;
00001809 1 0308 MOVC XR5 R5 ; POINT INSTR TO BE TESTED; ;
00001810 1 0309 XCU X5 ; EXEC SELECTED INSTRUCTION; ;
00001811 1 030A JMPI S6 BREAKCHECK ; UNTIL BREAK ;
00001812 1 030B JMPI & UTILITIES ;*** R:(BREAK OCCURED); ;
00001813 1 030C JMP UT2320 ; END; ;
00001814 1 030D UT2350: ; ELSE ;
00001815 1 030D MOV XRO R0 ; INITIALIZE REGS ;
00001816 1 030E MOV XR1 R1 ; ;
00001817 1 030F MOV XR2 R2 ; ;
00001818 1 0310 MOV XR3 R3 ; ;
00001819 1 0311 MOV XR4 R4 ; ;
00001820 1 0312 MOV XR6 R6 ; ;
00001821 1 0313 MOV XR7 R7 ; ;
00001822 1 0314 UT2360: ; REPEAT ;
00001823 1 0314 MOVC XR5 R5 ; ;
00001824 1 0315 XCU X5 ; EXEC SELECTED INSTRUCTION; ;
00001825 1 0316 JMP UT2360 ; ENDLESSLY (UNTIL MASTER CL); ;

```

00001821	1	0313	MOV	XR7	R7	;	
00001822	1	0314	UT2360:			;	REPEAT
00001823	1	0314	MOV	XR5	R5	;	
00001824	1	0315	XCU		X5	;	EXEC SELECTED INSTRUCTION;
00001825	1	0316	JMP			;	ENDLESSLY (UNTIL MASTER CL);
00001826	1	0317	JFF			;	


```

00001827 1 0317 ; FR/800303 ;
00001828 1 0317 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001829 1 0317 ;WAIT FOR INTERRUPT ;
00001830 1 0317 ;----- ;
00001831 1 0317 ; W <DEVICE_ADDRESS>A ;
00001832 1 0317 ; ;
00001833 1 0317 ;USING: CPUIVT#0:3A, CPUIVR, CPUIVB#0:63A, CPUIVB+64#0:CONTEXT_SIZEAA ;
00001834 1 0317 ;CALLS: GETHEXA ;
00001835 1 0317 ;RETURN: UT5020 ;
00001836 1 0317 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001837 1 0317 ; ;
00001838 1 0317 UT2500: ;WAIT_INTERRUPT: ;
00001839 1 0317 JMPI S6 GETHEXA ; GET_HEXA_DEVICE); ;
00001840 1 031B XTR R2 6 ; ISOLATE_DEVICE_NO; ;
00001841 1 0319 MOV XBASE R5 ; "OWN" PROCESS BASE; ;
00001842 1 031A SVP UT2520 ; ;
00001843 1 031B JMP UT2540 ; CONTINU_PROCESSING; ;
00001844 1 031C ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001845 1 031C ; WAIT DUMMY INTERRUPT ; ;
00001846 1 031C ; EXECUTED BY DUMMY PROCESS ; ;
00001847 1 031C ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001848 1 031C NOLIST IF SCMODE THEN LIST ; ;
00001849 1 031C ;WAIT_DUMMY_INTERRUPTS: ; ;
00001850 1 031C UT2520: ; REPEAT ; ;
00001851 1 031C LDM 2#110 ; OPEN FOR I/O INTERRUPTS; ;
00001852 1 031B MOV SCMDR R7 ; GET_DEVICE_ADDRESS; ;
00001853 1 031E MODC SCMDSTATUS ; ;
00001854 1 031F SIO R0 R7 ; SENSE(STATUS, DEVICE); ;
00001855 1 0320 SBZ R0 SCMDIRY ; CHECK IF BREAK OCCURED ;
00001856 1 0321 IBZ R0 SCMDFRAME ; SINCE LAST CHECK ;
00001857 1 0322 JMP UT2520 ; UNTIL BREAK OCCURED; ;
00001858 1 0323 LDM 2#111 ; CLOSE I/O WINDOW; ;
00001859 1 0324 JMPI S4 OUTSTRING, <:MISSING_INTERRUPT<13>:), EOT ;
00001860 1 032F LDM R5 ; ACTIVATE_MASTER_PROCESS; ;
00001861 1 0330 NOLIST ELSE LIST ; ;
00001873 1 0330 NOLIST FI LIST ; ;
00001874 1 0330 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001875 1 0330 ; INITIALIZE INTERRUPT TABLES ; ;
00001876 1 0330 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00001877 1 0330 UT2540: ;CONTINUATION: ;
00001878 1 0330 IEQP R1 TRUE ; IF DATA ENTERED THEN ;
00001879 1 0331 IF AMOS THEN ; ;
00001880 1 0331 MODC & CPUIVB+64 ; ;
00001881 1 0331 ELSE ; ;
00001882 1 0331 MODC & CPUDBASE ; ;
00001883 1 0332 FI ; ;
00001884 1 0332 MODN & XBASE ; COMPENSATE FOR DEFAULT ALL; ;
00001885 1 0333 MOD XBASE ; ;
00001886 1 0334 MOVC 0 R0 ; SET UP PROCESS EXPECTING INT'PT; ;
00001887 1 0335 NEG R5 ; POINT TOP OF MEMORY SECTION 0; ;
00001888 1 0336 MOVC CPUIVB R4 ; POINT INTERRUPT_VECTOR_TABLE; ;
00001889 1 0338 ADD R5 R4 ; ;

```

```

00001885 1 0333 MOD XBASE
00001886 1 0334 MOVC 0 R0 ; SET UP PROCESS EXPECTING INT/PT;
00001887 1 0335 NES R5 ; POINT TOP OF MEMORY SECTION 0;
00001888 1 0336 MOVC CPUIVB R4 ; POINT INTERRUPT VECTOR TABLE;
00001889 1 0338 ADD R5 R4 ;
00001890 1 0339 MOVC 64 R7 ; FOR I:=1 UNTIL MAX_DEVICES;
00001891 1 033A UT2550: ; DO
00001892 1 033A MOV R0 0. X4 ; INSERT BASE IN INTERRUPT TAB;
00001893 1 0338 ADDC 1 R4 ; ADJUST POINTER;
00001894 1 033C SOB R7 UT2550 ; UNTIL TABLE PREPARED;
00001895 1 033D ;
00001896 1 033D IF AMOS THEN ;
00001897 1 033D ADDC CPUDLEVEL- % ;
00001898 1 033D (CPUIVB+64) R4 ;
00001899 1 033E FI ;
00001900 1 033E MOVC CPUIVB R3 ; PREPARE ALL PRIORITY LEVEL2
00001901 1 0340 MOV R3 CPUIVT+0.X5; WITH
00001902 1 0341 MOV R3 CPUIVT+1.X5; ADDRESS OF SAME
00001903 1 0342 MOV R3 CPUIVT+2.X5; INTERRUPT
00001904 1 0343 MOV R3 CPUIVT+3.X5; VECTOR TABLE;
00001905 1 0344 IF AMOS THEN ;
00001906 1 0344 ADDC 64 R3 ; BASE DUMMY PROCESS;
00001907 1 0344 ELSE ;
00001908 1 0344 ADDC CPUDBASE-CPUIVB R3 ;
00001909 1 0345 FI ;
00001910 1 0345 MOV R3 CPUIVR.X5 ; PREPARE SYSTEM BASE WITH "OWN";
00001911 1 0346 ;
00001912 1 0346 IF AMOS THEN ;
00001913 1 0346 MOVC XR0 R7 ; POINT TOP OWN PROCESS;
00001914 1 0346 ELSE ;
00001915 1 0346 MOVC XLEVEL R7 ;
00001916 1 0347 FI ;
00001917 1 0347 MOVM X7 X4 ; CREATE
00001918 1 0348 MOVM X7 X4 ; DUMMY PROCESS;
00001919 1 0349 IF AMOS THEN ;
00001920 1 0349 ADDC -2*16 R4 ; DUMMY PROCESS;
00001921 1 0349 ELSE ;
00001922 1 0349 ADDC -2*16-XLEVEL R4 ; DUMMY PROCESS;
00001923 1 034A FI ;
00001924 1 034A MOV R3 XBASE. X4 ; PREPARE BASE
00001925 1 034B MOV R3 XMOD. X4 ; AND MOD REGISTER;
00001926 1 034C ;
00001927 1 034C UT2560: ;
00001928 1 034C ; REARM INTERRUPT TABLE;
00001929 1 034C MODN XBASE ;
00001930 1 034D MOVC CPUIVB R5 ; POINT START INTERRUPT TABLE;
00001931 1 034F MOV XBASE R1 ;
00001932 1 0350 MOV R1 0. X5 ; PREPARE FOR TIMER;
00001933 1 0351 MOD OCADR ;
00001934 1 0352 MOV R1 0. X5 ; OPERATOR'S CONSOLE
00001935 1 0353 MOD R2 ; SELECTED DEVICE;
00001936 1 0354 MOV R1 0. X5 ;
00001937 1 0355 ;
00001938 1 0355 SUB UT2580 ;
00001939 1 0356 CON R3 ; ACTIVATE DUMMY INTERRUPT PROC;

```

```

00001938 1 0355 SUP UT2580 ;
00001939 1 0356 LDN R3 ; ACTIVATE DUMMY INTERRUPT_PROC;
00001940 1 0357 ;
00001941 1 0357 ; INTERRUPT OCCURED ;
00001942 1 0357 ;
00001943 1 0357 UT2580: ;WAKE_UP:
00001944 1 0357 NOLIST IF SCHMODE THEN LIST ;
00001945 1 0357 JMPI S6 BREAKCHECK ; PERFORM BREAK-CHECK;
00001946 1 0358 JMP UT5020 ;*** RETURN(BREAK ACTIVATED);
00001947 1 0359 NOLIST ELSE LIST ;
00001954 1 0359 NOLIST FI LIST ;
00001955 1 0359 ;
00001956 1 0359 MOV XDEVICE R2 ;
00001957 1 035A IEQ R2 $FFFF ; IF CPU_INTERRUPT OR LDP
00001958 1 035B JMPI ERROR ; THEN ERROR
00001959 1 035C ; ELSE
00001960 1 035C MOV PRINTOUT R0 ;
00001961 1 035D SNE R0 TRUE ; IF PRINT-OUT
00001962 1 035E IEQ R2 0 ; .AND.NOT. TIMER_INTERRUPT
00001963 1 035F JMP UT2590 ; THEN
00001964 1 0360 MOV OCADR R7 ;
00001965 1 0361 SRL R2 2 ; ISOLATE_DEVICE_NO;
00001966 1 0362 SNE R2 R7 ; IF .NOT. OC_INTERRUPT
00001967 1 0363 JMP UT2590 ; THEN
00001968 1 0364 JMPI S4 OUTSTRING, <:IO INTERRUPT, DEV=:>, EOT ;
00001969 1 036F JMPI S6 OUTHEXA ;
00001970 1 0370 JMPI S4 OUTSTRING, <:, PRIO=:>, EOT ;
00001971 1 0376 MOV XDEVICE R0 ;
00001972 1 0377 XTR R0 2 ; ISOLATE_PRIORITY;
00001973 1 0378 ADDC <:0:> R0 ; CONVERT TO ASCII
00001974 1 0379 JMPI S6 OUTCHAR ;
00001975 1 037A JMPI S6 OUTNL ;
00001976 1 037B UT2590: ; END PRINTING;
00001977 1 037B JMP UT2560 ; UNTIL BREAK OCCURED;
00001978 1 037C JFF ;

```

```

00001979 1 037C ;
00001980 1 037C ;
00001981 1 037C ;SWITCH_CPU
00001982 1 037C ;-----
00001983 1 037C ; 0(CPU_NUMBER)
00001984 1 037C ;
00001985 1 037C ;CALLS: INCHAR, SEND_CPU_MESSAGE(SB2800)
00001986 1 037C ;RETURN: UT5010
00001987 1 037C ;
00001988 1 037C ;
00001989 1 037C UT2800: ;SWITCH_CPU:
00001990 1 037C JMPI S6 INCHAR ; GET NEW MASTER CPU_NO;
00001991 1 037D ADDC -(0:0) R0 ;
00001992 1 037E IHS R0 B ; IF INVALID
00001993 1 037F JMPI ERROR ; THEN FLAG B0;
00001994 1 0380 JMP S6 SB2800 ; SEND_CPU_MESSAGE;
00001995 1 0381 JMP UT5010 ; RETURN UTILITIES_MAIN_LINE;
00001996 1 0382 ;
00001997 1 0382 ;
00001998 1 0382 ;
00001999 1 0382 ;
00002000 1 0382 ; PROCEDURE SEND_CPU_MESSAGE I:(CPU_NO);
00002001 1 0382 ; THE PROCEDURE WILL PREPARE THE CPU_MESSAGE_AREA SUCH THAT THE
00002002 1 0382 ; SPECIFIED CPU WILL CONTINUE WHILE CURRENT ONE GOES ASLEEP.
00002003 1 0382 ; THE SPECIFIED CPU IS AWAKENED BY A CPU_INTERRUPT.
00002004 1 0382 ; N O T E NO CHECK IS PERFORMED TO VERIFY THE EXISTANCE OF THE OTHER.
00002005 1 0382 ;
00002006 1 0382 ; R0 CPU TO BE ACTIVATED DESTROYED
00002007 1 0382 ; R2,3,7 - DESTROYED
00002008 1 0382 ; R4 - POINTER TO TOP OF PAGE 0
00002009 1 0382 ; R6 LINK KEPT
00002010 1 0382 ;
00002011 1 0382 ;USING: CPUCPU#0:7A, CPUDUMMY#0:CONTEXT_SIZE#A
00002012 1 0382 ;RETURN: CALLER
00002013 1 0382 ;
00002014 1 0382 ;
00002015 1 0382 SB2800: ;SEND_CPU_MESSAGE:
00002016 1 0382 MOVD XBASE
00002017 1 0383 MOVC #0000 R4 ; POINT TOP OF MEMORY "OWN" PAGE;
00002018 1 0384 MOVC CPUDUMMY R7 ;
00002019 1 0385 MOVC CPUNSG R2 ; OTHER'S CPU'S MESSAGE;
00002020 1 0386 MOD R2 ;
00002021 1 0387 MOV R7 0. X4 ; INSERT DUMMY_BASE IN MESSAGE;
00002022 1 0388 SETS R2 BIT15 ; FORCE MESSAGE_RESERVED;
00002023 1 0389 MOVC 8 R3 ; FOR U:=1 UNTIL 8 CPU'S
00002024 1 038A SB2810: ; DO
00002025 1 038A MOV R2 CPUCPU.X4 ; INSERT POINTER;
00002026 1 038B ADDC 1 R4 ; ADJUST ADDRESS;
00002027 1 038C SDB R3 SB2810 ; UNTIL CPU_MESSAGE_AREA DONE;
00002028 1 038D ADDC -8 R4 ; RE-ESTABLISH POINTER
00002029 1 038E ;

```

00002027	1	038D	ADDC	-8	R4	;	RE-ESTABLISH POINTER	;
00002029	1	038E				;		;
00002030	1	038E	MOV	XBASE	R2	;	USE "OWN" BASE REGISTER	;
00002031	1	038F	ADDC	XBASE	R2	;	AS POINTER;	;
00002032	1	0390	MOD		R0	;		;
00002033	1	0391	MOV		R2	;	CPUCPU.X4	;
00002034	1	0392				;		;
00002035	1	0392	SVP		SB2820	;	PREPARE OTHER CPU'S PROCESS;	;
00002036	1	0393	JMP		SB2825	;		;
00002037	1	0394	;;;;;;;;;;;;;			;		;
00002038	1	0394	;	OTHER CPU'S FAITH		;		;
00002039	1	0394	;;;;;;;;;;;;;			;		;
00002040	1	0394	SB2820:			;	OTHER'S FAITH:	;
00002041	1	0394	LDM	2#101		;		;
00002042	1	0395	JMP		LOC	;	WAIT FOR CPU INTERRUPT;	;
00002043	1	0396	;;;;;;;;;;;;;			;		;
00002044	1	0396	;	CREATE LOOSER'S SLEEPING PROCESS		;		;
00002045	1	0396	;;;;;;;;;;;;;			;		;
00002046	1	0396	SB2825:			;	MASTER_PROCESS_CONTINUATION:	;
00002047	1	0396	ADD		R7	;	POINT DUMMY CPU_PROCESS;	;
00002048	1	0397	IF AMOS THEN			;		;
00002049	1	0397	MOVC	XRO	R3	;		;
00002050	1	0397	ELSE			;		;
00002051	1	0397	MOVC	XLEVEL	R3	;		;
00002052	1	0398	ADDC	CPUDUMMYLEVEL -	X	;		;
00002053	1	0398		CPUDUMMY	R4	;		;
00002054	1	0399	FI			;		;
00002055	1	0399	MOV	X3	X4	;	CREATE DUMMY	;
00002056	1	039A	MOV	X3	X4	;	CPU_INTERRUPT SERVICE PROCES;	;
00002057	1	039B	IF AMOS THEN			;		;
00002058	1	039B	ADDC	-32	R4	;		;
00002059	1	039B	ELSE			;		;
00002060	1	039B	ADDC	-32-XLEVEL	R4	;		;
00002061	1	039C	FI			;		;
00002062	1	039C	MOV	R7 XBASE.	X4	;	PREPARE BASE	;
00002063	1	039D	MOV	R7 XMOD.	X4	;	AND MOD REGISTERS;	;
00002064	1	039E	SUB	R7	R4	;		;
00002065	1	039F				;		;
00002066	1	039F	SVS		R2	;		;
00002067	1	03A0	LDM	2#101		;		;
00002068	1	03A1	CPU			;	ACTIVATE ALL CPU'S	;
00002069	1	03A2	MOV	MSEC	R7	;		;
00002070	1	03A3	SUB	R7 LOC		;	WAIT 1 MSEC;	;
00002071	1	03A4	LDS	R2		;	RE-ESTABLISH OLD PSW;	;
00002072	1	03A5	JMP		X6	;	RETURN CALLER;	;
00002073	1	03A6	IFF			;		;

Add F800 to Address

```

                                FR/800303;
00002074 1 03A6 ;
00002075 1 03A6 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002076 1 03A6 ;UNIT MAPPER
00002077 1 03A6 ;-----
00002078 1 03A6 ; U
00002079 1 03A6 ;
00002080 1 03A6 ;CALLS: SEND_CPU_MESSAGE(UT2800), OUTCHAR, OUTSTRING
00002081 1 03A6 ;RETURN: UT5010
00002082 1 03A6 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002083 1 03A6 ;
00002084 1 03A6 UT2900: ;MAP_CPU'S:
00002085 1 03A6 JMPI S4 OUTSTRING, <(<13>CPU: :), EOT
00002086 1 03AB SVS R5 ;
00002087 1 03AC MOV R5 R0 ;
00002088 1 03AD SWP R0 ;
00002089 1 03AE XTR R0 3 ; ISOLATE "OWN" CPU_NO)
00002090 1 03AF JMP S6 SB2800 ; SEND_CPU_MESSAGE("OWN"...CPU);
00002091 1 03B0 ;
00002092 1 03B0 MOVC 8 R3 ; FOR I:=1 UNTIL 8 CPU'S
00002093 1 03B1 UT2930: DO
00002094 1 03B1 MOVC <:1:> R0 ; CPU EXIST BY DEFAULT;
00002095 1 03B2 SOZ CPUCPU.X4 ; IF NO_MESSAGE ACCEPTED
00002096 1 03B3 MOVC <:1:> R0 ; THEN FLAG SO;
00002097 1 03B4 JMPI S6 OUTCHAR ;
00002098 1 03B5 ADDC 1 R4 ;
00002099 1 03B6 SOB R3 UT2930 ; END MAPPING CPU'S;
00002100 1 03B7 ;
00002101 1 03B7 JMP S4 UT2950 ;
00002102 1 03B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002103 1 03B8 ; SERVE MISSING MODULE/TIME_OUT ;
00002104 1 03B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002105 1 03B8 ;TIME_OUT:
00002106 1 03B8 MOV XCAUSE R6 ;
00002107 1 03B9 IEQ R6 2 ; IF PARITY_ERROR
00002108 1 03BA JMP UT2940 ; THEN OK;
00002109 1 03B9 INE R6 3 ; IF TIME_OUT
00002110 1 03BC JMP LOCALINT ;*** THEN IT DESERVES SPECIAL CARE;
00002111 1 03BD MOVC <:1:> R0 ; FLAG MISSING 4K_RAM/IO_MODULE;
00002112 1 03BE UT2940: ;PRINT_RESULT:
00002113 1 03BE LDS R5 ; RE-ESTAB "OWN" PSW, LOWER BIT11;
00002114 1 03BF IF XAMDS THEN
00002115 1 03BF MOVC 0 R6 ;
00002116 1 03C0 LDL R6 ; RESET LEVEL
00002117 1 03C1 FI
00002118 1 03C1 MOV R3 R6 ;
00002119 1 03C2 XTR R6 4 ; IF (COUNT MOD 16) .EQ. 0
00002120 1 03C3 IEQ R6 7 ; IF (COUNT MOD 16) .EQ. 7
00002121 1 03C4 ADDC <:1:><8 R0 ; THEN INSERT EXTRA SPACE;
00002122 1 03C6 IEQ R6 15 ; IF (COUNT MOD 16) .EQ. 15
00002123 1 03C7 ADDC <:1/><8 R0 ; THEN ADJUST;
00002124 1 03C8 JMPI S6 OUTCHAR ; PRINT CURRENT CHAR;

```

```

00002123 1 03D7 ADDC <1/2>XB R0 ; THEN ADJUST;
00002124 1 03C9 JMFI S4 OUTCHAR ; PRINT CURRENT CHAR;
00002125 1 03CA SRL R0 8 ; POSITION EVENTUAL SECOND CHAR;
00002126 1 03CB INE R0 0 ; IF ANY 2ND CHAR
00002127 1 03CC JMFI S4 OUTCHAR ; THEN PRINT AS WELL;
00002128 1 03CD ADDC 1 R3 ; MAINTAIN CURRENT MODULE_NO;
00002129 1 03CE SHS R3 64 ; IF STILL MODULES LEFT
00002130 1 03D0 JMP 1. X4 ; THEN RETURN SERVE
00002131 1 03D1 JMP 0. X4 ; ELSE DONE WITH THIS SECTION;
00002132 1 03D2 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002133 1 03D2 ; MAP RAM'S ;
00002134 1 03D2 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002135 1 03D2 UT2950: ;MAP_RAM^:
00002136 1 03D2 MOV R4 XLOACT ; SET-UP OWN LOCAL ACTION;
00002137 1 03D3 JMFI S4 OUTSTRING, <13>RAM: ;>, EDT
00002138 1 03D8 MOVC 0 R3 ; FOR I:=1 UNTIL 4*16 4K_RAM'S
00002139 1 03D9 JMP S4 UT2960 ; DO
00002140 1 03DA JMP UT2970 ; **EXIT WHEN RAM'S TESTED;
00002141 1 03DB UT2960: MOVC <1:1> R0 ; SET-UP UNIT_EXISTS;
00002142 1 03DC MOV R3 R6 ;
00002143 1 03DD SRL R6 4 ; POSITION MEMORY_SECTION;
00002144 1 03DE MOD R6 ;
00002145 1 03DF MOV SB16PAGE R6 ; SET-UP PAGE;
00002146 1 03E0 MOV R3 R7 ;
00002147 1 03E1 SLL R7 12 ; COMPUTE CORRECT ADDRESS;
00002148 1 03E2 SUB XBASE R7 ;
00002149 1 03E4 LDS R6 ; SWITCH TO RELEVANT PAGE;
00002150 1 03E5 MOV 0. X7 R6 ; TEST RELEVANT 4K_RAM;
00002151 1 03E6 JMP UT2940 ; UNTIL ALL MODULES SERVED;
00002152 1 03E7 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002153 1 03E7 ; MAP IO_MODULES ;
00002154 1 03E7 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002155 1 03E7 UT2970: ;MAP IO_MODULES:
00002156 1 03E7 JMFI S4 OUTSTRING, <13>IO: ;>, EDT
00002157 1 03EC MOVC 0 R3 ; FOR I:=0 UNTIL 63 IO_MODULES
00002158 1 03ED JMP S4 UT2980 ; DO
00002159 1 03EE JMP UT2990 ; **EXIT WHEN ALL IO_MODULES DONE;
00002160 1 03EF UT2980: MOVC <1:1> R0 ; DEFAULT EXISTING MODULE;
00002161 1 03F0 SIO R6 R3 ; SENSE(STATUS, DEVICE);
00002162 1 03F1 JMP UT2940 ; UNTIL ALL MODULES SERVED;
00002163 1 03F2 ;
00002164 1 03F2 UT2990: ;
00002165 1 03F2 JMP UT5010 ; RETURN UTILITIES_MAIN_LINE;
00002166 1 03F3 ;FF

```

```

00002167 1 03F3 ;
00002168 1 03F3 ;
00002169 1 03F3 ;OPTIONS
00002170 1 03F3 ;-----
00002171 1 03F3 ; 0
00002172 1 03F3 ;
00002173 1 03F3 ;CALLS: GETHEXA, OUTSTRING
00002174 1 03F3 ;USING: PRINTOUT, PGMLOOP+0:1A
00002175 1 03F3 ;RETURN: UT5020
00002176 1 03F3 ;
00002177 1 03F3 OPT00: ;OPTIONS:
00002178 1 03F3 JMPI S4 OUTSTRING, <(<13>PRINT-OUT ?<62>:)>, EOT
00002179 1 03FC JMPI S6 INCHAR ; (Y)
00002180 1 03FD ADDC TRUE-YES R0 ;
00002181 1 03FE MOV R0 PRINTOUT ;
00002182 1 03FF ;
00002183 1 03FF MOVC 0 R0 ;
00002184 1 0400 MOV R0 PGMNEXT ; RESET TEST_PGM_CONTROL;
00002185 1 0401 ;
00002186 1 0401 JMPI S4 OUTSTRING, <(<13>NO OF TESTS <62>:)>, EOT
00002187 1 040A JMPI S6 GETHEXA ; GET FIRST PART OF COUNT
00002188 1 040B MOV R2 R3 ;
00002189 1 040C MOVC 0 R2 ; PRESET MOST SIGNIFICANT
00002190 1 040D INEP R0 CR ; IF LEAST SIGNIFICANT TOO
00002191 1 040E JMPI & S6 GETHEXA ;
00002192 1 040F XCH & R3 R2 ; POSITION;
00002193 1 0410 ADDC -1 R3 ; ADJUST LEAST SIGNIFICANT PART;
00002194 1 0411 IEQ R3 -1 ; IF LOWER PART OVERFLOW
00002195 1 0412 ADDC -1 R2 ; THEN ADJUST MOST SIGNIFICANT;
00002196 1 0413 MOV R2 PGMLOOP+1 ; SAVE MOST SIGNIFICANT PART
00002197 1 0414 MOV R3 PGMLOOP ; LEAST SIGNIFICANT AS WELL;
00002198 1 0415 JMP UT5020 ; RETURN UTILITIES MAIN LINE;
00002199 1 0416 ;FF

```



```

00002200 1 0416 ; B O O T   U T I L I T I E S
00002201 1 0416 ;
00002202 1 0416 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002203 1 0416 ;BOOT UTILITIES
00002204 1 0416 ;-----
00002205 1 0416 ; B ?F,X? A#<DEVICE_ADDRESS>.A<DRIVE_NO>
00002206 1 0416 ;
00002207 1 0416 ;CALLS:  UT3050, FDBLOAD, INCHAR, OUTNL,OUTSTRING
00002208 1 0416 ;RETURN: NO RETURN IF SUCCESFULL DISKETTE LOAD (ACTIVATES PROCESS)
00002209 1 0416 ;          UT5010 ON UNSUCCESFUL DISKETTE LOAD
00002210 1 0416 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002211 1 0416 UT3000:                                ;BOOT_LOAD:
00002212 1 0416      MOV      FDDEFAULT      R0      ; SET UP DEFAULT FLOPPY DISK
00002213 1 0417      MOV      R0      FDADR      ;      DEVICE CONTROLLER ADDRESS;
00002214 1 0418      JMPI      S4      OUTSTRING,    <:OOT :>, EOT
00002215 1 041C      JMPI      S6      INCHAR      ;      GET TRANSFER TYPE
00002216 1 041D      MOV      R0      R2      ;
00002217 1 041E      JMPI      S4      OUTSTRING,    <:D: :>, EOT
00002218 1 0422      JMPI      S6      INCHAR      ;      GET FD DRIVE_NO;
00002219 1 0423      IER      R0      <:#:>      ;      IF REQUEST FOR ALTERNATIVE
00002220 1 0425      JMP      S6      UT3050      ;      DEVICE CONTROLLER;
00002221 1 0426      JMPI      S6      FDBLOAD      ;      LOAD FILE FROM FD
00002222 1 0427      JMP &          UT5010      ;*** FAILED: RETURN MAIN LINE;
00002223 1 0428      ;          R:(FILE OK);
00002224 1 0428      JMPI      S6      OUTNL      ;
00002225 1 0429 IF XAMOS THEN
00002226 1 0429      LDN          R3      ;      ACTIVATE BOOT FILE;
00002227 1 042A ELSE
00002228 1 042A      SLN          R3      ;      ACTIVATE BOOT FILE;
00002229 1 042A FI
00002230 1 042A
00002231 1 042A
00002232 1 042A
00002233 1 042A ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002234 1 042A ; PROCEDURE GET_ALTERNATIVE_DEVICE_ADDRESS;
00002235 1 042A ;
00002236 1 042A ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002237 1 042A UT3050:
00002238 1 042A      MOV      R6      UT30SV      ;
00002239 1 042B      MOV      R2      UT30SV+1      ;      SAVE REG CONTENT;
00002240 1 042C      JMPI      S6      GETHEXA      ;      GET ALTERNATIVE DEVICE ADDRESS;
00002241 1 042D      MOV      R2      FDADR      ;
00002242 1 042E      SEQ      R0      <:.:>      ;      IF REQUEST FOR DEFAULT
00002243 1 0430      MOVC      <:0:>          R0      ;      SET UP DRIVE 0 AS DEFAULT
00002244 1 0431      IER      R0      <:.:>      ;      ELSE
00002245 1 0433      JMPI      S6      INCHAR      ;      GET ALTERNATIVE ADDRESS;
00002246 1 0434      MOV      UT30SV+1      R2      ;      RE-ESTABLISH REG 2 CONTENT;
00002247 1 0435      JMPI      UT30SV      ;      RETURN CALLER;
00002248 1 0436
00002249 1 0436 ;FF

```

FR/810915 ;

FR/810915 ;

00002247 1 0435
00002248 1 0436
00002249 1 0436 JFF

JMPI

UT3050

RETURN CALLER

FR/810915 ;

```
00002250 1 0436 ;  
00002251 1 0436 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
00002252 1 0436 ;LOAD UTILITIES  
00002253 1 0436 ;-----  
00002254 1 0436 ; L ?F,X?  ##<DEVICE_ADDRESS>.<DRIVE_NUMBER>  
00002255 1 0436 ;  
00002256 1 0436 ;CALLS:  UT3050, FDBLOAD, INCHAR, OUTSTRING  
00002257 1 0436 ;RETURN:  UT5010  
00002258 1 0436 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  
00002259 1 0436 UT3100: ;LOAD_FILE:  
00002260 1 0436      MOV      FDDEFAULT      R0 ; SET UP DEFAULT FLOPPY DISK  
00002261 1 0437      MOV      R0      FADDR ;      DEVICE CONTROLLER ADDRESS;  
00002262 1 0438      JMPI     S4      OUTSTRING, <:OAB :>, EOT  
00002263 1 043C      JMPI     S6      INCHAR ;      GET DEVICE IDENTIFICATION  
00002264 1 043D      MOV      R0      R2 ;  
00002265 1 043E      JMPI     S4      OUTSTRING, <:D: :>, EOT  
00002266 1 0442      JMPI     S6      INCHAR ;      GET FD DRIVE_NO;  
00002267 1 0443      IEQ      R0      <:F: > ;      IF REQUEST FOR ALTERNATIVE CTL R;  
00002268 1 0445      JMP      S6      UT3050 ;      THEN GET DEVICE SET UP;  
00002269 1 0446      JMPI     S6      FDBLOAD ;      LOAD_FILE FROM FD;  
00002270 1 0447      JMP      &      UT5010 ;*** FAILED: RETURN_MAINLINE;  
00002271 1 0448      JMP      UT5010 ;      RETURN UTILITIES_MAINLINE;  
00002272 1 0449 ;JFF
```

```

00002273 1 0449 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002274 1 0449 UT5000: ;UTILITIES: ;
00002275 1 0449      MOV      R6  UT50SV      ; SAVE RETURN ADDRESS;
00002276 1 044A UT5010: ;
00002277 1 044A      JMPI     S6  OUTNL      ;
00002278 1 044B UT5020: ;
00002279 1 044B      MOV     LOCALACT      R0 ;
00002280 1 044C      MOV     R0  XLOACT      ; RE-ESTABLISH TO COMMON LOCAL
00002281 1 044D ;
00002282 1 044D      MOV     <:62>:      R0 ;
00002283 1 044E      JMPI     S6  OUTCHAR    ; WRITE '>'
00002284 1 044F      JMPI     S6  INCHAR     ;
00002285 1 0450      ADDC    -<:0>:      R0 ;
00002286 1 0451      JMP      S6  UT5050      ; ESTABLISH LINK
00002287 1 0452      JMP      UT5010      ;
00002288 1 0453 UT5050: ;
00002289 1 0453 ; CASE CHAR OF
00002290 1 0453      IEQ     R0  <:B>:-<:0>:, JMPI  BOOTLOAD
00002291 1 0455      IEQ     R0  <:C>:-<:0>:, JMPI  COPYMEMORY
00002292 1 0457      IEQ     R0  <:D>:-<:0>:, JMPI  DUMP
00002293 1 0459      IEQ     R0  <:E>:-<:0>:, JMPI  EXECUTE
00002294 1 045B      IEQ     R0  <:I>:-<:0>:, JMPI  IO
00002295 1 045D      IEQ     R0  <:L>:-<:0>:, JMPI  LOADFILE
00002296 1 045F      IEQ     R0  <:M>:-<:0>:, JMPI  MEMCHECK
00002297 1 0461      IEQ     R0  <:O>:-<:0>:, JMPI  OPTIONS
00002298 1 0463      IEQ     R0  <:P>:-<:0>:, JMPI  PATCH
00002299 1 0465      IEQ     R0  <:Q>:-<:0>:, JMPI  SWITCHCPU
00002300 1 0467      IEQ     R0  <:R>:-<:0>:, JMPI  RETURNCALLER
00002301 1 0469      IEQ     R0  <:S>:-<:0>:, JMPI  SEARCH
00002302 1 046B      IEQ     R0  <:T>:-<:0>:, JMPI  TESTCPU
00002303 1 046D      IEQ     R0  <:U>:-<:0>:, JMPI  UNITS
00002304 1 046F      IEQ     R0  <:W>:-<:0>:, JMPI  WAITINTERRUPT
00002305 1 0471      IEQ     R0  <:U>:-<:0>:, JMPI  UNITS
00002306 1 0473      IEQ     R0  <:X>:-<:0>:, JMPI  EXECUTE
00002307 1 0475      IEQ     R0  <:Z>:-<:0>:, JMPI  SETPARITY
00002308 1 0477      IEQ     R0  <:K>:-<:0>:, JMPI  UT50SV
00002309 1 047A      IEQ     R0  <:W>:-<:0>:, JMPI  INSTRUCTIONLOOP
00002310 1 047D ;
00002311 1 047D ;
00002312 1 047D UT5090: ;ERROR:
00002313 1 047D      MOV     <:7>:      R0 ;
00002314 1 047E      JMPI     S6  OUTCHAR    ;
00002315 1 047F      JMP      UT5010      ;
00002316 1 0480 ;FF

```

```

00002317 1 0480 NOLIST IF FDSUPPORT THEN LIST
00002318 1 0480 ;
00002319 1 0480 ;
00002320 1 0480 ; PROJECT: PDS
00002321 1 0480 ;
00002322 1 0480 ; MODULE NAME: AMU FDL0AD
00002323 1 0480 ; MODULE ID NMB: PDS/1000
00002324 1 0480 ; MODULE VERSION: 0400
00002325 1 0480 ; MODULE TYPE: SYSTEM
00002326 1 0480 ; MODULE FILES: AMUFDLOAD.S
00002327 1 0480 ;
00002328 1 0480 ;
00002329 1 0480 ;
00002330 1 0480 ; PROCEDURE LOAD_BOOT_FILE I:(MODE_OF_OPERATION)
00002331 1 0480 ; R(LINK: FAILED, LINK+1 SUCCES);
00002332 1 0480 ; LOAD OK
00002333 1 0480 ; R0 DRIVE_NUMBER DESTROYED
00002334 1 0480 ; R2 TYPE_OF_XFER (-/'X') PSW CORRESPONDING TO BOOT_FILE
00002335 1 0480 ; DEVICE STATUS IF FAILED LOAD;
00002336 1 0480 ; R3 - BASE OF LOADED FILE
00002337 1 0480 ; R6 LINK DESTROYED
00002338 1 0480 ;
00002339 1 0480 ; FDREAD= 2#00010<11
00002340 1 0480 ; FDRESTORE= 2#01000<11
00002341 1 0480 ; FDSEEK= 2#00100<11
00002342 1 0480 ; FDSELECT= 2#00000<11
00002343 1 0480 ; FDWRITE= 2#00001<11
00002344 1 0480 ;
00002345 1 0480 ;
00002346 1 0480 ; FD1000: ;LOAD_BOOT_FILE:
00002347 1 0480 ; MOV R6 FD10SV ;
00002348 1 0481 ; XTR R0 2 ; ISOLATE DRIVE NUMBER (0..3);
00002349 1 0482 ; MOV R0 FD10DRV ;
00002350 1 0483 ;
00002351 1 0483 ; ADDC -(0:0) R2 ; NORMALIZE;
00002352 1 0484 ; MOV R2 FDMODE ; SAVE MODE_OF_OPERATION (FD/DMA);
00002353 1 0485 ; MOV DMAADR R7 ; GET DMA DEVICE_ADR;
00002354 1 0486 ; IEGP R2 (X:0)-(0:0) ; IF INDIRECT TRANSFER
00002355 1 0487 ; MODC & DMRESET ; THEN
00002356 1 0488 ; WIO & R7 R7 ; MASTER_CLEAR DMA_DEVICE;
00002357 1 0489 ;
00002358 1 0489 ; JMPT S6 SETPARITY ;
00002359 1 048A ;
00002360 1 048A ; MOVF FDZERO R5 ; POINT TOP OF AREA TO BE INIT'D;
00002361 1 048B ; MOVF 0 R7 ;
00002362 1 048C ; MOVF FDEND-FDZERO R3 ; FOR I:=1 UNTIL ENDFD_AREA
00002363 1 048D ; FD1010: ; DO
00002364 1 048D ; MOV R7 0. X5 ; CLEAR OFF AREA;
00002365 1 048E ; ADDC 1 R5 ;
00002366 1 048F ; SOB R3 FD1010 ;
00002367 1 0490 ; MOVF FDBENTRY R5 ; POINT BOOT_ENTRY DESCRIPTOR;

```

		SOB	R3	FD1010		
00002364	1	049F				
00002367	1	0490	MOV	FDBENTRY	R5	POINT BOOT_ENTRY DESCRIPTOR;
00002368	1	0491	INC	FDAREASIZE.X5		AREASIZE:= 1 SECTOR
00002369	1	0492	INC	FDSECTORS.X5		SECTOR_PER_BLOCK:= 1
00002370	1	0493	INC	FDBLOCKS.X5		BLOCKS_PER_AREA:= 1
00002371	1	0494	INC	FDAREA.X5		1ST_AREA ADDRESS:= 0(8+1
00002372	1	0495	MOD	XBASE		
00002373	1	0496	MOV	FDDIRBUF	R4	POINT FD_DIRECTORY BUFFER;
00002374	1	0497	MOV	R4	FDCLOAD	
00002375	1	0498				
00002376	1	0498				NOTE R3:=0 (BLOCK 0)
00002377	1	0498	MOV	FDSEEK	R2	SET-UP TRACK 0 OF LOAD DRIVE;
00002378	1	049A	JMPI	S6	FDCONTROL	SEEK_TRACK OF LOAD DRIVE;
00002379	1	049B	JMPI	S6	FDREADBLOCK;	
00002380	1	049C	JMPI	S6	FDTRANSFER	TRANSFER LAST SECTOR (ONLY ONE);
00002381	1	049D	JMPI	S6	FDCOPY	
00002382	1	049E			FDENTRYLENGTH	
00002383	1	049F	;;;;;;;;;;			
00002384	1	049F	;	LOAD	BOOT_FILE	
00002385	1	049F	;;;;;;;;;;			
00002386	1	049F	MOV	0	R6	
00002387	1	04A0	MOV	R6	FD20CTL	RISE 1ST TIME FLAG;
00002388	1	04A1	MOV	FDDIRBUF	R5	POINT DIRECTORY_ENTRY;
00002389	1	04A2	MOVL	FDLOAD.X5	R01	GET LOAD_ADDRESS, PAGE
00002390	1	04A3	MOVL	R01	FDCLOAD.X6	INITIALIZE
00002391	1	04A4	MOV	FDENTRYLENGTH-1	R3	FOR I:=1 UNTIL ENTRY_LENGTH
00002392	1	04A5	FD1020:			DO
00002393	1	04A5	ADD	0.	X5	COMPUTE DIR_ENTRY CHECKSUM;
00002394	1	04A6	ADDC	1	R5	
00002395	1	04A7	SOB	R3	FD1020	END SUMMING UP;
00002396	1	04A8	INE	R6	0.	IF DISCREPANCY
00002397	1	04A9	JMP		FDENTRYERROR	
00002398	1	04AA				
00002399	1	04AA	MOV	FDDIRBUF	R5	POINT BOOT DIRECTORY BUFFER;
00002400	1	04AB	FD1040:			REPEAT
00002401	1	04AB	MOV	R3	FDBLOCKNO	SAVE BLOCKNO
00002402	1	04AC	JMPI	S6	FDREADBLOCK;	GET NEXT BLOCK TO MEM_BUFFER;
00002403	1	04AD	MOV	FDBLOCKNO	R3	
00002404	1	04AE	INEP	R3	FDHIBLK.X5	TEST IF LAST AREA (BLOCK)
00002405	1	04B0	ADDC	1	R3	
00002406	1	04B1	JMP		FD1040	UNTIL ALL BLOCKS DONE;
00002407	1	04B2				
00002408	1	04B2	JMPI	S6	FDTRANSFER	LAST_TRANSFER;
00002409	1	04B3	JMPI	S6	FDCOPY	REMEMBER COPYING LAST SECTOR;
00002410	1	04B4			64	
00002411	1	04B5	;;;;;;;;;;			
00002412	1	04B5	;	FILE	LOADED, VALIDATE CHECKSUM	
00002413	1	04B5	;;;;;;;;;;			
00002414	1	04B5	MOV	XBASE	R1	
00002415	1	04B6	SYS		R4	"OWN" PSW;
00002416	1	04B7	MOV	FDLOAD.X5	R6	GET FILE LOAD ADDRESS (START)
00002417	1	04B8	MOV	FDPAGE.X5	R7	GET CORRESPONDING PAGE;
00002418	1	04B9	IDR	R4	R7	COMPUTE RELEVANT PSW;
00002419	1	04BA	MOV	FDCOUNT.X5	R3	FOR I:=1 UNTIL FILE_LENGTH
00002420	1	04BB	FEQ	R3	0	

```

00002415 1 04B6 SVS ; GET FILE LOAD ADDRESS (START) ;
00002416 1 04B7 MOV FDLOAD.X5 R3 ; GET CORRESPONDING PAGE ;
00002417 1 04B8 MOV FDPAGE.X5 R7 ; COMPUTE RELEVANT PSW ;
00002418 1 04B9 IOR R4 R3 ; FOR I:=1 UNTIL FILE.LENGTH ;
00002419 1 04BA MOV FDCOUNT.X5 R3 ;
00002420 1 04BB IEQ R3 0 ;
00002421 1 04BC JMP FD1060 ;
00002422 1 04BD MOVE 0 R0 ;
00002423 1 04BE LDS R7 ;
00002424 1 04BF FD1050: DO ;
00002425 1 04BF MODN R1 ;
00002426 1 04C0 ADD 0. X6 R0 ; COMPUTE CHECKSUM ;
00002427 1 04C1 ADDC 1 R6 ;
00002428 1 04C2 IERF R6 0 ; IF PAGE BOUNDARY ;
00002429 1 04C3 ADDC & 1<2 R7 ; THEN ADJUST (NEGLECT CARRY) ;
00002430 1 04C4 LDS & R7 ; TO NEXT MEMORY SECTION ;
00002431 1 04C5 SOB R3 FD1050 ; END COMPUTING CHECK-SUM ;
00002432 1 04C6 ;
00002433 1 04C6 LDS R4 ; RETURN "OWN" MEMORY SECTION ;
00002434 1 04C7 INE R0 FDCHECKSUM.X5 ; BEWARE CHECKSUM_ERROR ;
00002435 1 04C9 JMP FDCHECKSUMERROR ;
00002436 1 04CA ;
00002437 1 04CA ; FILE CHECKED: OK - GET BASE, PAGE ;
00002438 1 04CA ;
00002439 1 04CA FD1060: ;
00002440 1 04CA JMPI S4 OUTSTRING, <:, START=:>, EOT ;
00002441 1 04D0 MOV FDLOAD.X5 R2 ;
00002442 1 04D1 JMPI S6 OUTHEXA ;
00002443 1 04D2 JMPI S4 OUTSTRING, <:, BASE=:>, EOT ;
00002444 1 04D3 MOV FDBASE.X5 R2 ;
00002445 1 04D9 JMPI S6 OUTHEXA ;
00002446 1 04DA MOVL FDPAGE.X5 R23 ; GET PAGE, BASE ;
00002447 1 04DB SVS R7 ;
00002448 1 04DC IOR R7 R2 ; COMPUTE RELEVANT PSW ;
00002449 1 04DD MOV FD10SV R6 ; GET LINK ;
00002450 1 04DE JMP 1. X6 ; SUCCESSFULL RETURN ;
00002451 1 04DF ;
00002452 1 04DF FDENTRYERROR: ;
00002453 1 04DF JMPI S4 OUTSTRING, <:, ENTRY=:>, EOT ;
00002454 1 04E5 JMP FD1080 ;
00002455 1 04E6 FDCHECKSUMERROR: ;
00002456 1 04E6 JMPI S4 OUTSTRING, <:, CHECKSUM=:>, EOT ;
00002457 1 04ED FD1080: ;
00002458 1 04ED JMPI S4 OUTSTRING, <: ERROR=:>, EOT ;
00002459 1 04F2 JMP FD1099 ;
00002460 1 04F3 ;
00002461 1 04F3 ;
00002462 1 04F3 ; DEVICE ERROR ;
00002463 1 04F3 ; R2 DEVICE_STATUS ;
00002464 1 04F3 ;
00002465 1 04F3 DMAERROR: ; DMA_ERROR: ;
00002466 1 04F3 JMPI S4 OUTSTRING, <:, DMA=:>, EOT ;
00002467 1 04F8 JMP FD1090 ;
00002468 1 04F9 FDDISKERROR: ;
00002469 1 04F9 JMPI S4 OUTSTRING, <:, DISK=:>, EOT ;

```

```

00002467 1 04F8 JMPI
00002468 1 04F9 JMPI S4 OUTSTRING, <:, DISK:>, EDT ;
00002469 1 04FE FD1090: ;ERROR_RETURN ;
00002471 1 04FE JMPI S4 OUTSTRING, <: ERROR=:>, EDT ;
00002472 1 0504 JMPI S6 OUTHEXA ; ;
00002473 1 0505 FD1099: JMPI FD10SV ; RETURN CALLER (FAILED LOAD); ;
00002474 1 0506 JFF

```

```

00002475 1 0506 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002476 1 0507 ; PROCEDURE FD_READ_BLOCK ; (SEE(ATA ENTRY)) ;

```

```

00002475 1 0506 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002476 1 0506 ; PROCEDURE FD_READ_BLOCK I:(REF(DIR_ENTRY))
00002477 1 0506 ;
00002478 1 0506 ; THIS PROCEDURE SUPPORTS THE CR8062 FD CONTROLLER. THE CONTROLLER
00002479 1 0506 ; WORKS WITH A 64 WORD BUFFER.
00002480 1 0506 ; THE PROCEDURE COPIES ALL BUT THE LAST SECTOR OF A BLOCK TO THE
00002481 1 0506 ; DEDICATED OUTPUT AREA VIA ITS MEMORY_BUFFER.
00002482 1 0506 ;
00002483 1 0506 ; HIERARCHY:
00002484 1 0506 ; FILE/AREA:1:10A/BLOCK:1:-A/SECTOR:1:64A/WORD:1:16A/BIT
00002485 1 0506 ; THE RELATION BETWEEN PHYSICAL AND LOGICAL SECTORS ARE:
00002486 1 0506 ; TRACK 0: 2, 4, ..., 26, 1, 3, ..., 25
00002487 1 0506 ; TRACK 1: 14, 16, ..., 12, 13, 15, ..., 11
00002488 1 0506 ; TRACK 2: 26, 1, 3, 26, 1, 3, 12
00002489 1 0506 ; TRACK 3: 11, 13, 15, ...
00002490 1 0506 ;
00002491 1 0506 ; ... UNTIL TRACK 76
00002492 1 0506 ;
00002493 1 0506 ; R3 BLOCK_NO DESTROYED
00002494 1 0506 ; R5 REF(DIR_ENTRY) KEPT
00002495 1 0506 ; R6 LINK DESTROYED
00002496 1 0506 ; R0-2,4,7 - DESTROYED
00002497 1 0506 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002498 1 0506 ;
00002499 1 0506 ; FD2000: ; FD_READ_BLOCK:
00002500 1 0506 ; MOV R6 FD20SV ;
00002501 1 0507 ; MOV R5 R6 ;
00002502 1 0508 ; MOV FDBLOCKS.X5 R1 ; GET_BLOCKS_PER_AREA;
00002503 1 0509 ; IEQ R1 0 ;
00002504 1 050A ; JMP S6 FENTRYERROR;
00002505 1 050B ;
00002506 1 050B ; MOVC FDAREAS R2 ;
00002507 1 050C ; FD2010: ; COMPUTE
00002508 1 050C ; ILT R3 R1 ; OFF_SET:=
00002509 1 050D ; JMP FD2020 ;
00002510 1 050E ; SUB R1 R3 ; BLOCK_NO MOD BLOCKS_PER_AREA;
00002511 1 050F ; ADDC 1 R6 ; ADJUST AREA_REFERENCE;
00002512 1 0510 ; SOB R2 FD2010 ;
00002513 1 0511 ; JMP S6 FENTRYERROR; TOO HIGH BLOCK_NO: ENTRY)ERR;
00002514 1 0512 ; FD2020:
00002515 1 0512 ; MOV FDSECTORS.X5 R1 ; GET_SECTORS_PER_BLOCK;
00002516 1 0513 ; MOV FDAREA.X6 R0 ; GET AREA'S START LOG_SECTOR_NO;
00002517 1 0514 ; IEQ R0 -1 ; IF NOT USED
00002518 1 0515 ; JMP S6 FENTRYERROR; THEN ENTRY_ERROR;
00002519 1 0516 ; JOZ R3 FD2060 ; IF .NOT. 1ST BLOCK IN AREA
00002520 1 0517 ; MOV R0 R2 ;
00002521 1 0518 ; XTR R2 8 ; ISOLATE LOGICAL_SECTOR_NO;
00002522 1 0519 ; SWP R0 ; POSITION TRACK_NO;
00002523 1 051A ; FD2030: ; COMPUTE
00002524 1 051A ; ADD R1 R2 ; 1ST_SECT+BLOCK*SECT'S_PER_BLK;
00002525 1 051B ; SOB R3 FD2030 ;

```



```

00002524 1 051A ADD R1 1ST_SECT=BLOCK*SECT'S_PER_BLK;
00002525 1 051B SOB R3 FD2030 ;
00002526 1 051C FD2040: ; COMPUTE ;
00002527 1 051C ILT R2 27 ;
00002528 1 051E JMP FD2030 ;
00002529 1 051F ADDC 1 R0 ; ADJUST_TRACK_NUMBER;
00002530 1 0520 ADDC -26 R2 ; ADJUST_SECTOR_NO;
00002531 1 0521 JMP FD2040 ; END_ADJUSTING_TRACK/SECTOR;
00002532 1 0522 FD2050: ;
00002533 1 0522 SLL R0 8 ; POSITION_TRACK, ZERO_SECTOR;
00002534 1 0523 IOR R2 R0 ; INSERT_LOGICAL_SECTOR_NO;
00002535 1 0524 FD2060: ; FD_SEEK_TRACK:
00002536 1 0524 ; R0 - TRACK, /LOGICAL_SECTOR_NO;
00002537 1 0524 ; R1 - SECTORS PER BLOCK;
00002538 1 0524 ; REPEAT;
00002539 1 0524 MOV R0 R2 ; TRACK_OFFSET:=
00002540 1 0525 SRL R2 7 ; (TRACK_NO*2
00002541 1 0526 MOV R2 R3 ;
00002542 1 0527 ADD R2 R3 ;
00002543 1 0528 ADD R2 R3 ; *3)
00002544 1 0529 FD2070: IGEP R3 26 ;
00002545 1 052B ADDC -26 R3 ; MOD 26;
00002546 1 052C JMP FD2070 ;
00002547 1 052D ;
00002548 1 052D SRL R2 1 ; ISOLATE_TRACK_NO;
00002549 1 052E MOV FDTRACK R6 ;
00002550 1 052F IEQ R2 R6 ; IF_NEW_TRACK .NOT. CURRENT
00002551 1 0530 JMP FD2080 ; THEN
00002552 1 0531 MOV R2 FDTRACK ; ESTABLISH_AS_CURRENT_TRACK;
00002553 1 0532 JMPI S6 FDWAITCMDCOMPLETE; COMPLETE_PREVIOUS_COMMAND;
00002554 1 0533 MOV FDTRACK R2 ;
00002555 1 0534 ADDC FDSEEK R2 ; ESTABLISH_SEEK;
00002556 1 0536 JMPI S6 FDCONTROL ;
00002557 1 0537 ; END_ESTABLISHING_NEW_TRACK;
00002558 1 0537 FD2080: ; REPEAT
00002559 1 0537 JMPI S6 FDTRANSFER ; GET_DATA_INTO_MEMORY_BUFFER;
00002560 1 0538 MOV R0 R2 ;
00002561 1 0539 XTR R2 8 ; ISOLATE_LOGICAL_SECTOR_NO;
00002562 1 053A ADD R3 R2 ; ADJUST_FOR_TRACK_OFFSET;
00002563 1 053B ; PHYSICAL_SECTOR_NO:=
00002564 1 053B ADD R2 R2 ; ((2*(TRACK_OFFSET-LOG_SECT))
00002565 1 053C IGE R2 53 ;
00002566 1 053E ADDC -52 R2 ; MOD 53;
00002567 1 053F IGE R2 27 ;
00002568 1 0541 ADDC -27 R2 ; MOD 27;
00002569 1 0542 ADDC FDREAD R2 ; SET-UP_PROPER_READ_COMMAND;
00002570 1 0544 JMPI S6 FDCONTROL ; CONTROL(READ, SECTOR);
00002571 1 0545 MOV FD20CTL R4 ; GET_1ST_TIME_FLAG;
00002572 1 0546 INEP R4 0 ; IF_COPY_DATA_FROM_LAST_READ;
00002573 1 0547 JMPI S6 FDCOPY ; THEN_DO_SO;
00002574 1 0548 ; 64 ;
00002574 1 0549 ***** A ***** NODE-WRN 301 ;
00002575 1 0549 MOV R6 FD20CTL ;
00002576 1 054A ; NEXT_SEGMENT: ;
00002577 1 054A ADDC -1 R1 ;

```

TRACK SECTOR R0
 10040 121A 0

```

00002573 1 0547 JMP 1 0547 64 ;
00002574 1 0548 ;
00002575 1 0549 MOV R6 FD20CTL ;
00002576 1 054A ; NEXT_SEGMENT: ;
00002577 1 054B ADDC -1 R1 ; ;
00002578 1 054C SGE R1 1 ; * PERFORM OUTER LOOP CONTROL ;
00002579 1 054D JMP1 FD20SV ; * IF DONE RETURN CALLER ;
00002580 1 054E ; ;
00002581 1 054F ADDC 256 -26 R0 ; ADJUST TRACK/SECTOR_NUM ;
00002582 1 0550 IBNP R0 BIT7 ; CHECK IF NEXT TRACK ;
00002583 1 0551 ADDC -256 +27 R0 ; ;
00002584 1 0552 JMP FD2080 ; UNTIL SEC'S THIS TRACK DONE ;
00002585 1 0553 ADDC 1 R0 ; NOTE: START SECTOR 1, NOT 0 ;
00002586 1 0554 JMP FD2060 ; UNTIL AREA DONE (ALL BLOCKS) ;
00002587 1 0555 ;
00002588 1 0556 ;FF

```

OFFSET: TRACK * 6 - N * 26 , N volges sector

at ($0 \leq \text{OFFSET} \leq 26$)

Sector #: $((\text{OFFSET} + \text{Sector}) - x * 52) - y * 27$

$x = \begin{cases} 0 \\ 1 \end{cases}$, $y = \begin{cases} 0 \\ 1 \end{cases}$, og volges sector at

($1 \leq \text{Sector} \leq 26$)

```

00002589 1 0553 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002590 1 0553 ; PROCEDURE FD_TRANSFER I:(REF(BUFFER));
00002591 1 0553 ; WAIT FOR I/F TO GO READY;
00002592 1 0553 ; TRANSFER DATA FROM I/F BUFFER TO MEMORY;
00002593 1 0553 ; R2,4,7 - DESTROYED
00002594 1 0553 ; R6 LINK KEPT
00002595 1 0553 ; USING: XR0(X4), XR2(R2)
00002596 1 0553 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002597 1 0553 ;
00002598 1 0553 ;FDLWAIT_COMMAND_COMPLETE:
00002599 1 0553 FDLTRANSFER:
00002600 1 0553 MOV R6 FD303V ;
00002601 1 0554 MOVC 60 R4 ;
00002602 1 0555 SOB R4 LOC ; WAIT 60-120 USEC;
00002603 1 0556 ;
00002604 1 0556 MOV FDMODE R7 ; GET MODE_OF_OPERATION
00002605 1 0557 IEQ R7 <:X:>-<:D:>; IF DIRECT TRANSFER
00002606 1 0558 JMP FD3050 ; THEN
00002607 1 0559 MOV FDADR R7 ; GET FD_DEVICE_ADDRESS;
00002608 1 055A FD3010: ; REPEAT
00002609 1 055A MODC 1<15
00002610 1 055B SIO R2 R7 ; SENSE(ADDITIONAL_STATUS,DEV);
00002611 1 055C IBN R2 BIT6 ; TEST DEVICE STILL BUSY;
00002612 1 055D JMP FD3010 ; UNTIL FORMATTER READY;
00002613 1 055E ;
00002614 1 055E MOVC FDMEMBUF R2 ; POINT DEDICATED MEMORY BUFFER;
00002615 1 0560 MOVC 64 R4 ; FOR I:=1 UNTIL BLOCK_SIZE
00002616 1 0561 FD3020: ; DO
00002617 1 0561 RIO X2 R7 ; READ(BUFFER_WORD, DEVICE);
00002618 1 0562 ADCC 1 R2 ;
00002619 1 0563 SOB R4 FD3020 ; END_FOR_UNTIL;
00002620 1 0564 ;
00002621 1 0564 SIO R2 R7 ; SENSE(STATUS, DEVICE);
00002622 1 0565 JMP FD3060 ; ELSE
00002623 1 0566 FD3050: ; REPEAT
00002624 1 0566 JMP S6 DMA ; DMA: SENSE REMOTE FD
00002625 1 0567 2,48000+FD,DNSMODE,XR0,DMEMMODE,DMINPUT ;
00002626 1 056D IBN X4 BIT6 ; TEST REMOTE FORMATTER BUSY;
00002627 1 056E JMP FD3050 ; UNTIL FORMATTER READY
00002628 1 056F JMP S6 DMA ; DMA: READ 64 WORDS FROM REM.FD;
00002629 1 0570 128,FD,DMRMODE,FDMEMBUF,DMEMMODE,DMINPUT ;
00002630 1 0576 JMP S6 DMA ; DMA: SENSE REMOTE FD
00002631 1 0577 2,FD,DNSMODE,XR2,DMEMMODE,DMINPUT ;
00002632 1 057D MOV XR2 R2 ; GET PRESENT STATUS;
00002633 1 057E FD3060: ; END TRANSFERING SECTOR;
00002634 1 057E IBN R2 BIT0 ; RESET
00002635 1 057F ADCC #FFFF R2 ; BIT0 ONLY;
00002636 1 0580 IEQ R2 0 ; IF TRANSFER COMPLETE
00002637 1 0581 JMPI FD303V ; RETURN CALLER AS DONE;
00002638 1 0582 ; ELSE
00002639 1 0582 IBZ R2 BIT7 ; IF NOT. DISK_CHANGE.OR.OVERRUN
00002640 1 0583 JMP FDDISKERROR; THEN
00002641 1 0584 MOVC FPRESTORE R2 ; SET-UP FD RESTORE;

```

00002635	1	0580	ILG	R2	FD303V	;	RETURN CALLER AS DONE;	;
00002637	1	0581	JMPI			;	ELSE	;
00002638	1	0582	IBZ	R2	BIT7	;	IF %NOT. DISK_CHANGE.OR.OVERRUN	;
00002639	1	0582	JMP		FDDISKERROR;	;	THEN	;
00002640	1	0583	MOVC	FDRESTORE	R2	;	SET-UP FDLRESTORE:	;
00002641	1	0584	JMPI	S6	FDCONTROL	;	STEP DRIVE OUT 79 TIMES TO	;
00002642	1	0585				;	USE TRACK_00 FOR RESYNC;	;
00002643	1	0587	MOV	R6	FDTRACK	;	FORCE NEW SEEK;	;
00002644	1	0587	MOVC	FDRECOUNT	R6	;		;
00002645	1	0588	INC		0.	X6	;	MAINTAIN RESTORE_COUNT;
00002646	1	0589	IBZ	X6	BIT4	;	CONTINUATION ALLOWED	;
00002647	1	058A	JMP		FD2060	;	AS LONG AS BELOW 16;	;
00002648	1	058B	JMP			;		;
00002649	1	058C	MOVC	%FFFF	R2	;	SET-UP ERROR FLAG	;
00002650	1	058D	JMP		FDDISKERROR;	;	END DISKERROR HANDLING;	;
00002651	1	058E	JFF					

```

00002652 1 058E ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002653 1 058E ; PROCEDURE FD_MEM_COPY ;
00002654 1 058E ; R6 LINK KEPT ;
00002655 1 058E ; R7 - DESTROYED ;
00002656 1 058E ; ;
00002657 1 058E ; USING: PROCESS_DESCRIPTOR ;
00002658 1 058E ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;
00002659 1 058E ; ;
00002660 1 058E FD3100: ;FD_COPY: ;
00002661 1 058E SVP ; ;
00002662 1 058F SVS R2 ; GET OWN PSW ;
00002663 1 0590 ;NOTE HAS TO BE MEM_SECTION 0 ;
00002664 1 0590 MOV XBASE R4 ; AND BASE ;
00002665 1 0591 MOV FDMEMBUF R5 ; POINT FD MEM_BUFFER ;
00002666 1 0593 MOV FDCPAGE R3 ; GET CURRENT OUTPUT PAGE ;
00002667 1 0594 IOR R2 R3 ; ESTABLISH NEW PSW THAT SECTION ;
00002668 1 0595 MOV FDCLOAD R7 ; POINT OUTPUT_AREA ;
00002669 1 0596 MVP 0. X6 R1 ; FOR I:=1 UNTIL BUFFER_SIZE ;
00002670 1 0597 FD3120: ; DO ;
00002671 1 0597 MOV 0. X5 R0 ; GET NEXT WORD ;
00002672 1 0598 LDS R3 ; SWITCH TO RELEVANT SECTION ;
00002673 1 0599 MODN R4 ;
00002674 1 059A MOV R0 0. X7 ; INSERT WORD IN OUTPUT_AREA ;
00002675 1 059B LDS R2 ; RETURN OWN PAGE ;
00002676 1 059C INCD R5 R7 ; ADJUST POINTERS ;
00002677 1 059D IER R7 0 ; IF PAGE BOUNDARY ;
00002678 1 059E ADDC 1<2 R3 ; THEN SWITCH PAGE ;
00002679 1 059F ; (NEGLECTING CARRY BIT) ;
00002680 1 059F SOB R1 FD3120 ; UNTIL WHOLE BUFFER COPIED ;
00002681 1 05A0 MOV R7 FDCLOAD ;
00002682 1 05A1 MOV R3 FDCPAGE ;
00002683 1 05A2 MOV XRO R7 ; RE-ESTABLISH ;
00002684 1 05A3 UNS 7 ; ALL REGISTERS ;
00002685 1 05A4 JMP 1. X6 ; RETURN CALLER ;
00002686 1 05A5 ;FF ;

```

```

00002687 1 05A5 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002688 1 05A5 ; PROCEDURE FD_CONTROL ; I:(COMMAND) ;

```

```

00002687 1 05A5 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002688 1 05A5 ; PROCEDURE FD_CONTROL I:(COMMAND);
00002689 1 05A5 ; R2 COMMAND<11+TRACK/SECTOR DESTROYED
00002690 1 05A5 ; R6 LINK KEPT
00002691 1 05A5 ; R7 - DESTROYED
00002692 1 05A5 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002693 1 05A5 FD3500: ;FD_CONTROL:
00002694 1 05A5 MOV FD10DRV R7 ; GET DRIVE NUMBER;
00002695 1 05A6 SWP R7 ; POSITION IN LEFTMOST BYTE;
00002696 1 05A7 ADD R7 R2 ; INSERT DRIVE_NO IN COMMAND;
00002697 1 05A8 MOV FDMODE R7 ;
00002698 1 05A9 SNE R7 <:X:~<:0:~> IF DIRECT FD_LOAD
00002699 1 05AA JMP FD3580 ; THEN
00002700 1 05AB MOV FDADR R7 ; GET DEVICE ADDRESS;
00002701 1 05AC CIO R2 R7 ; CONTROL(DATA, DEVICE);
00002702 1 05AD JMP X6 ; RETURN CALLER;
00002703 1 05AE FD3580: ; ELSE
00002704 1 05AE MOV R6 FD30SV ;
00002705 1 05AF MOV R2 XR2 ;
00002706 1 05B0 JMP S6 DMA ; DMA: CONTROL REMOTE DEVICE
00002707 1 05B1 2,FD,DMCMODE,XR2,DMMEMMODE,DMOUTPUT ;
00002708 1 05B7 JMPI FD30SV ; RETURN CALLER;
00002709 1 05B8 ; END CONTROLLING FD;
00002710 1 05BB ;FF

```

```

00002711 1 05B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002712 1 05B8 ; PROCEDURE DO_DMA (CNT, REM_LADR, REM_PAGE, LOC_LADR, LOC_PAGE, REQUEST);
00002713 1 05B8 ; R2 - DESTROYED
00002714 1 05B8 ; RL LINK DESTROYED
00002715 1 05B8 ; R7 - DESTROYED
00002716 1 05B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002717 1 05B8 ; DMA IF CHARACTERISTICS
00002718 1 05B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002719 1 05B8 ;
00002720 1 05B8 ;DMA_STATUS:
00002721 1 05B8 DMRQSNT= BIT3 ; REQUEST SENT
00002722 1 05B8 DMNTEOB= BIT5 ; NOT_LEOB
00002723 1 05B8 DMERRM1= 1<BIT6 OR %> STOPPED
00002724 1 05B8 1<BIT7 OR %> REMOTE TIME-OUT
00002725 1 05B8 1<BIT8 OR %> LOCAL TIME-OUT
00002726 1 05B8 1<BIT9 ; PARITY ERROR
00002727 1 05B8 ;DMA_COM'DS:
00002728 1 05B8 DMLDBC= 0<8 ; LOAD BYTE COUNT
00002729 1 05B8 DMLDAD= 1<8 ; LOAD ADDRESS
00002730 1 05B8 DMLDPG= 2<8 ; LOAD PAGE
00002731 1 05B8 DMSNRQ= 3<8 ; SEND REQUEST
00002732 1 05B8 DMSTART= 5<8 ; START DMA
00002733 1 05B8 DMSTOP= 6<8 ; STOP DMA
00002734 1 05B8 DMRESET= 7<8 ; RESET DMA IF
00002735 1 05B8 DMRNBC= 8<8 ; LOAD REMOTE BYTE COUNT
00002736 1 05B8 DMRHAD= 9<8 ; LOAD REMOTE ADDRESS
00002737 1 05B8 DMRMPG= 10<8 ; LOAD REMOTE PAGE
00002738 1 05B8 ;
00002739 1 05B8 DMINPUT= 2#01 ; INPUT REQUEST, TYPE 0
00002740 1 05B8 DMOUTPUT= 2#00 ; OUTPUT REQUEST, TYPE 0
00002741 1 05B8 DMMEMMODE= #302 ; MEMORY MODE: LOWER BYTE, PAGE 0
00002742 1 05B8 DMSMODE= #2 ; SENSE IO MODE
00002743 1 05B8 DMCMODE= #2 ; CONTROL IO MODE
00002744 1 05B8 DMRMODE= #6 ; READ IO MODE
00002745 1 05B8 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002746 1 05B8 DMA: ;DMA:
00002747 1 05B8 DMA00: ; REPEAT
00002748 1 05B8 MOV DMAADR R7 ; GET DEVICE ADDRESS;
00002749 1 05B9 MOVC DMOUTPUT R2 ; SET-UP CONTROL
00002750 1 05BA MOVC DMSNRQ
00002751 1 05BB WIO R2 R7 ; OUTPUT(CMD, CTL, DEVICE);
00002752 1 05BC MOVC 0 R2 ; WAIT
00002753 1 05BD MOVC 0 R2 ; 2 USEC;
00002754 1 05BE RIO R2 R7 ; INPUT(STATUS, DEVICE);
00002755 1 05BF IBZ R2 DMRQSNT ; UNTIL REQUEST
00002756 1 05C0 JMP DMA00 ; ACCEPTED;
00002757 1 05C1 MVP 0. X6 R2 ; GET COUNT_FIELD;
00002758 1 05C2 MOVC DMRNBC
00002759 1 05C3 WIO R2 R7 ; OUTPUT(CMD, COUNT, DEVICE);
00002760 1 05C4 MVP 1. X6 R2 ; GET REMOTE_LADR;
00002761 1 05C5 MOVC DMRHAD
00002762 1 05C6 WIO R2 R7 ; OUTPUT(CMD, REM_LADR, DEVICE);
00002763 1 05C7 MVP 2. X6 R2 ; GET REMOTE_PAGE (MODE);

```

00002758	1	05C2	WIO		R2	R7	;	OUTPUT(CMD, COUNT, DEVICE);	;
00002759	1	05C3	MVP	1.	X6	R2	;	GET REMOTE_ADDR;	;
00002760	1	05C4	MODC						
00002761	1	05C5	WIO		R2	R7	;	OUTPUT(CMD, REM_ADDR, DEVICE);	;
00002762	1	05C6	MVP	2.	X6	R2	;	GET REMOTE_PAGE (MODE);	;
00002763	1	05C7	MODC						
00002764	1	05C8	WIO		R2	R7	;	OUTPUT(CMD, REM_PAGE, DEVICE);	;
00002765	1	05C9	MVP	5.	X6	R2	;	GET REQUEST_TYPE;	;
00002766	1	05CA	MODC						
00002767	1	05CB	WIO		R2	R7	;	OUTPUT(CMD, SEND_REQUEST, DEVICE);	;
00002768	1	05CC	MVP	0.	X6	R2	;	GET COUNT	;
00002769	1	05CD	MODC						
00002770	1	05CE	WIO		R2	R7	;	OUTPUT(CMD% COUNT% DEVICE);	;
00002771	1	05CF	MVP	3.	X6	R2	;	GET POINTER TO OUTPUT_AREA;	;
00002772	1	05D0	ADD	XBASE		R2	;		;
00002773	1	05D1	MODC						
00002774	1	05D3	WIO		R2	R7	;	OUTPUT(CMD, LOAD_ADDR, DEVICE);	;
00002775	1	05D4	MVP	4.	X6	R2	;	GET LOAD_PAGE;	;
00002776	1	05D5	MODC						
00002777	1	05D6	WIO		R2	R7	;	OUTPUT(CMD, LOAD_PAGE, DEVICE);	;
00002778	1	05D7	MODC						
00002779	1	05D8	WIO		R2	R7	;	OUTPUT(CMD, -, DEVICE);	;
00002780	1	05D9					;	REPEAT	;
00002781	1	05DA DMA10:	RIO		R2	R7	;	INPUT(STATUS, DEVICE);	;
00002782	1	05DB	MOV	DMERRM1		R7	;	SET-UP ERROR MASK;	;
00002783	1	05DB	AND		R2	R7	;	IF RESULT.>0 THEN	;
00002784	1	05DE	INE		R7		;	IF RESULT<>0 THEN	;
00002785	1	05DF	JMP				;	DMA_ERROR;	;
00002786	1	05E0	MOV	DMAADR		R7	;		;
00002787	1	05E1	IBN		R2		;	UNTIL STATUS.NOT_EOB	;
00002788	1	05E2	JMP				;	GOES LOW (EOB);	;
00002789	1	05E3	MODC						
00002790	1	05E4	WIO		R2	R7	;	OUTPUT(CMD, -, DEVICE);	;
00002791	1	05E5	JMP			X6	;	RETURN CALLER;	;
00002792	1	05E6 NOLIST	FI LIST						
00002793	1	05E6 JFF							


```

00002794 1 05E6 ; INITIALIZE CPU TEST PROGRAM ;
00002795 1 05E6 ; ; ;
00002796 1 05E6 ; ; ;
00002797 1 05E6 ; ; ;
00002798 1 05E6 TEC00: ; TEST_CPU: ;
00002799 1 05E6 MOV C B5555 R5 ; ;
00002800 1 05E7 MOVL 0. X5 R23 ; ;
00002801 1 05E8 MOV P5555 R0 ; ;
00002802 1 05E9 MOV PAAAA R1 ; ;
00002803 1 05EA INE R0 R2 ; IF B5555_FIELD NOT CORRECT
00002804 1 05EB JMP TEC10 ; .OR.
00002805 1 05EC SNE R1 R3 ; BAAAA_FIELD NOT CORRECT
00002806 1 05ED JMP TEC90 ; THEN
00002807 1 05EE TEC10: ; ;
00002808 1 05EE ; ;
00002809 1 05EE JMPI S6 GETADDRESS ; GET POSITION OF (PROMMED)
00002810 1 05EF ; ; CPU_TEST SEQUENCES;
00002811 1 05EF MOV C #1000 R3 ; SET-UP COUNT (4K PROM);
00002812 1 05F1 MOV C 0 R0 ; OBJECT_SECTION 0;
00002813 1 05F2 MOV TCBASE R2 ; GET "BASE" CPU_TEST PROGRAMS;
00002814 1 05F3 MOV XBASE R7 ; ;
00002815 1 05F4 SUB R7 R2 ; MAKE REL "OWN" BASE;
00002816 1 05F5 JMPI S6 COPY ; COPY CPU_TEST PROGRAMS
00002817 1 05F6 ; (58700) ; TO MEMORY #1000 UPWARDS;
00002818 1 05F6 MOV XBASE R3 ; ;
00002819 1 05F7 MOV TCBASE R1 ; GET START OF CPU_TEST_PROGRAMS;
00002820 1 05F8 SUB R3 R1 ; SET-UP CPU-TEST-"OWN"=BASE;
00002821 1 05F9 MOV TCBASE R2 ; ;
00002822 1 05FA MOV XPROD R7 ; ;
00002823 1 05FB SUB R7 R2 ; SET-UP CPU_TEST-"OWN"_PROD;
00002824 1 05FC MOV R1 R7 ; ;
00002825 1 05FD MOV 0. X7 R0 ; GET FIRST STATEMENT;
00002826 1 05FE INE R0 #0158 ; IF NOT 'JMP LOC'
00002827 1 0600 JMPI ERROR ; THEN ERRORNEOUS MODULE;
00002828 1 0601 ; ; ;
00002829 1 0601 ; RAM RESIDENT VERSION ACCEPTED ; ;
00002830 1 0601 ; ; ;
00002831 1 0601 JMP S4 LOC+2 ; BY-PASS INSTRUCTION;
00002832 1 0602 JMPI S4 OUTSTRING ; ;
00002833 1 0603 MVL 0. X4 R0 ; GET INSTRUCTION
00002834 1 0604 MOV R0 0. X7 ; INSERTED AS FIRST OF AREA;
00002835 1 0605 MOV R2 R6 ; ;
00002836 1 0606 JMP S5 LOC+1 ; ESTABLISH LINK BACK;
00002837 1 0607 JMP X6 ; ACTIVATE RAM_RESIDING PART;
00002838 1 0608 ; ; ;
00002839 1 0608 ; RETURN_FROM_CPU_TEST; ;
00002840 1 0608 ; R0,4,6,7 DESTROYED ;
00002841 1 0608 ; R4 REF CPU_TEST PROGRAM_TABLE ;
00002842 1 0608 ; ; ;
00002843 1 0608 ; SET-UP CPU_TEST CONTROL ; ;
00002844 1 0608 ; ; ;
00002845 1 0608 ADD R1 R4 ; ADJUST REF(CPU_TEST_PROGS);
00002846 1 0609 MOV R1 R7 ; ;

```

```

00002841 1 0600 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002842 1 0608 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002843 1 0609 ; SET-UP CPU TEST CONTROL ;
00002844 1 0608 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00002845 1 0608 ADD R1 R4 ; ADJUST REF(CPU_TEST_PGMS);
00002846 1 0609 MOV R1 R7 ;
00002847 1 060A MOVC TEP00L R6 ;
00002848 1 060B TEC20: ; REPEAT ;
00002849 1 060B MOVL 0. X4 R01 ;
00002850 1 060C IEQ R0 #BE ; IF TEST UNDEFINED (TRAP) ;
00002851 1 060E SUB R0 R0 ; THEN FORCE DUMMY TEST ;
00002852 1 060F IEQ R0 0 ; IF DUMMY TEST ;
00002853 1 0610 JMP TEC30 ; THEN ;
00002854 1 0611 INE R0 #8000 ; IF NOT END_OF_TEST_FILE ;
00002855 1 0613 ADD R2 R0 ; ADJUST TO REL "ACTUAL" PROG ;
00002856 1 0614 TEC30: ; END ADJUSTING PROG_REL ;
00002857 1 0614 ADD R7 R1 ; ADJUST TO REL "ACTUAL" BASE ;
00002858 1 0615 MOVL R01 0. X6 ; MAKE ;
00002859 1 0616 ADDC 2 R4 ; REFERENCES ;
00002860 1 0617 ADDC 2 R6 ; RELATIVE ;
00002861 1 0618 IBZ R0 BIT15 ;
00002862 1 0619 JMP TEC20 ; UNTIL CPU TEST AREA INITIALIZED ;
00002863 1 061A ;
00002864 1 061A ;
00002865 1 061A MOVE 0 R7 ;
00002866 1 061B MOV R7 PGMLOOP ; INITIALIZE COUNTERS ;
00002867 1 061C MOV R7 PGMLOOP+1 ;
00002868 1 061D MOV R7 PGMNEXT ; AND CURRENT TEST PGM POINTER ;
00002869 1 061E TEC90: JMPI S6 OUTNL ;
00002870 1 061F ;CONTINUE WITH CPU TEST MAIN LINE ;
00002871 1 061F ;FF ;

```

```

00002872 1 061F ; CPU TEST PROGRAM - MAIN LINE
00002873 1 061F ;
00002874 1 061F USE BASE ;
00002875 0 0048 ; CPU_TEST_CONTROL_BLOCK LAY-OUT
00002876 0 0048 ; -----
00002877 0 0000 CTCBID: LOC= LOC+1 ; IDENTIFICATION OF TEST
00002878 0 0001 CTCBCNT:LOC= LOC+1 ; NUMBER OF DATA IN TEST RUN
00002879 0 0002 CTCBRST:LOC= LOC+1 ; RESET VALUE (IF 0 THEN FIRST AREA COPIED)
00002880 0 0003 CTCBDATA:LOC=LOC+1 ; DATA AREA:
00002881 0 0004 ; 1ST AREA (CNT) WORDS, EXPECTED DATA
00002882 0 0004 ; 2ND AREA, INPUT DATA AREA
00002883 0 0004 ;
00002884 0 0004 USE PROG ; FOR I:=1 UNTIL TEST_LOOP_COUNT
00002885 1 061F ; DO
00002886 1 061F ; REPEAT UNTIL TABLE SCANNED
00002887 1 061F MAINLOOP:
00002888 1 061F MOV PGMEVALUATE R0 ; GET CHECK/PRINT-OUT FLAG
00002889 1 0620 MOV PGMNEXT R6 ; GET REF(CUR'T TEST_PGM_NDX)
00002890 1 0621 MOV TCPOOL+TCPPROG.X6 R4 ; GET REF(CUR'T TEST_PGM_ADR)
00002891 1 0622 SNE R0 TRUE ; IF EVALUATION
00002892 1 0623 IEQ R4 0 ; .AND. VALID TEST_PROGRAM
00002893 1 0624 JMP MAISO ; THEN
00002894 1 0625 ; -----
00002895 1 0625 ; EVALUATE TEST RESULTS ;
00002896 1 0625 ; -----
00002897 1 0625 MOV TCPOOL+TCBDATA.X6 R4 ; GET REF(ASSOCIATED DATA_AREA)
00002898 1 0626 MOVC CTCBDATA R5 ; OFF-SET TO EXPECTED DATA;
00002899 1 0627 ADD R4 R5 ; REF (EXPECTED_DATA_AREA);
00002900 1 0628 MOV REFWORK R7 ; POINT WORK AREA;
00002901 1 0629 MOV CTCBCNT.X4 R3 ; FOR I:=1 UNTIL NO_WORDS
00002902 1 062A MAI10: ; DO
00002903 1 062A IEQ X5 X7 ; IF DISCREPANCY
00002904 1 062B JMP MAI20 ; THEN
00002905 1 062C MOV R7 MAISV ;
00002906 1 062D MOV R4 MAISV+1 ;
00002907 1 062E JMPI S4 OUTSTRING, <:TEST :>, EOT
00002908 1 0633 MOV MAISV+1 R4 ;
00002909 1 0634 MOV CTCBID.X4 R2 ; GET TEST IDENTIFICATION
00002910 1 0635 JMPI S6 OUTHEXA ;
00002911 1 0636 JMPI S4 OUTSTRING, <: WORD :>, EOT
00002912 1 063B MOV MAISV+1 R4 ;
00002913 1 063C MOV CTCBCNT.X4 R2 ; COMPUTE WORD
00002914 1 063D SUB R3 R2 ; NUMBER;
00002915 1 063E JMPI S6 OUTHEXA ;
00002916 1 063F JMPI S4 OUTSTRING, <: TO BE :>, EOT
00002917 1 0645 MOV X5 R2 ; GET ACTUAL RESULT
00002918 1 0646 JMPI S6 OUTHEXA ;
00002919 1 0647 JMPI S4 OUTSTRING, <:- WAS :>, EOT
00002920 1 064C MOV MAISV R7 ;
00002921 1 064D MOV X7 R2 ; GET EXPECTED
00002922 1 064E JMPI S6 OUTHEXA ; GET EXPECTED
00002923 1 064F JMPI S6 OUTNL ; WRITE NEWLINE;
00002924 1 0650 MOV MAISV+1 R4 ;

```

```

00002919 1 0640 MOV MAISV R7
00002920 1 0640 MOV X7 R2
00002921 1 0640 JMPI S6 OUTHEXA
00002922 1 0640 JMPI S6 OUTNL
00002923 1 0640 MOV MAISV+1 R4
00002924 1 0650 MOV MAISV R7
00002925 1 0651
00002926 1 0652 MAI20: ADDC 1 R5, ADDC 1 R7
00002927 1 0652 SOB R3 MAI10
00002928 1 0654
00002929 1 0655 ;-----
00002930 1 0655 ; CHECK IF BREAK OCCURED
00002931 1 0655 ;-----
00002932 1 0655 MAI50: JMPI S6 BREAKCHECK ; BREAK_CHECK
00002933 1 0655 JMPI & S6 UTILITIES ;*** R:(BREAK OCCURED);
00002934 1 0656
00002935 1 0657 ;-----
00002936 1 0657 ; SEARCH NEXT TEST SEQUENCE
00002937 1 0657 ;-----
00002938 1 0657 MOV PGMNEXT R6
00002939 1 0658 MAI55: ADDC 2 R6
00002940 1 0658 MOV TCPOOL+TCPPROG.X6 R7
00002941 1 0659 IEQ R7 0
00002942 1 065A JMP MAI55
00002943 1 065B MOV R6 PGMNEXT
00002944 1 065C SNE R7 #8000
00002945 1 065D JMP MAI80
00002946 1 065F MOV TCPOOL+TCpdata.X6 R4
00002947 1 0660 MOVL CTCBCNT.X4 R01
00002948 1 0661 ADD R0 R4
00002949 1 0662 MOV R4 R5
00002950 1 0663 MOV REFWORK R6
00002951 1 0664 IEQ R1 0
00002952 1 0665 JMP MAI65
00002953 1 0666
00002954 1 0667 MAI60: MOV R1 X6
00002955 1 0667 ADDC 1 R6
00002956 1 0668 SOB R0 MAI60
00002957 1 0669 JMP MAI70
00002958 1 066A
00002959 1 066B MAI65: MOV CTCBdata.X4 R2
00002960 1 066B MOV R2 X6
00002961 1 066C ADDC 1 R4, ADDC 1 R6
00002962 1 066D SOB R0 MAI65
00002963 1 066F
00002964 1 0670 MAI70: ADDC CTCBdata R5
00002965 1 0670 MOVL 0. X5 R01
00002966 1 0671 MOVL 2. X5 R23
00002967 1 0672 MOV 4. X5 R4
00002968 1 0673 MOV REFWORK R6
00002969 1 0674 JMP X7
00002970 1 0675
00002971 1 0676 ;-----
00002972 1 0676 ; EXECUTE TEST SEQUENCE
00002973 1 0676 ;-----

```

```

GET EXPECTED
GET EXPECTED
WRITE(NEWLINE)

RE-ESTABLISH POINTER;
END DISCREPANCY;

```

```

END FOR UNTIL;

```

```

BREAK_CHECK
R:(BREAK OCCURED);

```

```

GET REF(TEST_PGM TABLE INDEX);
REPEAT

```

```

POINT NEXT TEST_RECORD;
GET REF(TEST_PROGRAM);
IF VALID TEST
THEN
SAVE REFERENCE;
IF STILL TEST PROGRAMS LEFT
THEN

```

```

GET REF(ASSOCIATED DATA);
GET NO_WORDS, RESET VALUE;
ADJUST FOR EXPECTED_AREA;

```

```

POINT WORK AREA;
IF WHOLE AREA EQUAL 'R1'
THEN
REPEAT

```

```

INITIALIZE AREA;
ADJUST POINTER;
UNTIL AREA COVERED;
ELSE
REPEAT

```

```

GET NEXT WORD
COPIED;
ADJUST POINTERS;
END COPYING AREA;

```

```

ADJUST POINTER TO INPUT;

```

```

POINT WORK AREA;
EXECUTE TEST PROGRAM;

```

```

00002972 1 0676 ; EXECUTE TEST SEQUENCE ;
00002973 1 0676 ; ; TEST_PROGRAM_ENTRY: ;
00002974 1 0676 ; R0-4:FIRST 5 WORK_WORDS ;
00002975 1 0676 ; R5 - REF(WORK_AREA) ;
00002976 1 0676 ; R6 - REF(OUTPUT_AREA) ;
00002977 1 0676 ; ;
00002978 1 0676 ; ;
00002979 1 0676 ; ;
00002980 1 0676 ; MAINTAIN LONG_INTEGER ;
00002981 1 0676 ; ;
00002982 1 0676 MAI80: ; END TABLE_SCAN;
00002983 1 0676 MOVCL 0 R6 ; RE-ADJUST TO TOP
00002984 1 0677 MOV R6 PGMNEXT ; PROGRAM_INDEX;
00002985 1 0678 MOVCL PGMLOOP R6 ; REF(TEST_LOOP_COUNT);
00002986 1 0679 MOVL 0. X6 R01 ; GET NO_TEST LEFT (LONG INT);
00002987 1 067A ADDC -1 R0 ; ADJUST LEAST SIGNIFICANT;
00002988 1 067B IEQ R0 -1 ; IF LEAST_SIGN'T BELOW
00002989 1 067C ADDC -1 R1 ; THEN ADJUST MOST_SIGNIFI'T;
00002990 1 067D SNEP R0 -1 ;
00002991 1 067E IEQ R1 -1 ; CPU_LOOP_TEST
00002992 1 067F JMP TESTEND ;
00002993 1 0680 ;
00002994 1 0680 MOVL R01 0. X6 ; SAVE;
00002995 1 0681 JMPL MAINLINE ; END CPU_TEST_LOOP;
00002996 1 0682 ;
00002997 1 0682 ;
00002998 1 0682 ;
00002999 1 0682 ;
00003000 1 0682 ; END PROGRAM ;
00003001 1 0682 ; ;
00003002 1 0682 TESTEND: ;
00003003 1 0682 MOVCL (:*:) R0 ;
00003004 1 0683 MOVCL 44 R3 ; FOR I:=1 UNTIL 44
00003005 1 0684 MAI99: ; DO
00003006 1 0684 JMPL S6 OUTCHAR ; WRITE_CHAR;
00003007 1 0685 SOB R3 MAI99 ; END FOR_UNTIL;
00003008 1 0686 ;
00003009 1 0686 JMP UT5010 ; RETURN UTILITIES_MAIN_LINE;
00003010 1 0687 ;
00003011 1 0687 ;-----;
00003012 1 0687 ;FF

```

```

00003013 1 0687 ;
00003014 1 0687 ;
00003015 1 0687 ; AUTO BOOT LOADERS
00003016 1 0687 ;
00003017 1 0687 ;
00003018 1 0687 ; FUNCTION:
00003019 1 0687 ;     BOOT FROM DRIVE 0
00003020 1 0687 ;     BOOT FROM DRIVE 1
00003021 1 0687 ;     UTILITIES
00003022 1 0687 ;
00003023 1 0687 ; AUTODISKETTE:
00003024 1 0687 ;
00003025 1 0687 ;     JMP S6 AUTO90 ; SET-UP DEFAULT FD-DRIVE ADDRESS ;
00003026 1 0688 ;     JMPI & ERROR ; (BAD: CALL UTILITIES, OK: ;
00003027 1 0689 ;     MOV C (<:F:>) R2 ; SET UP DIRECT DISKETTE BOOT ;
00003028 1 068A ;     MOV C (<:0:>) R0 ; DRIVE 0 1ST ATTEMPT ;
00003029 1 068B ;     JMPI S6 FDBLOAD ; LOAD BOOT MODULE ;
00003030 1 068C ;     JMP & AUTO10 ; (BAD: TRY DRIVE 1, ;
00003031 1 068D ;     JMPI S6 OUTNL ; ;
00003032 1 068E IF AMOS THEN ;
00003033 1 068E ;     LDN R3 ; OK: ACTIVATE PROCESS ;
00003034 1 068E ELSE ;
00003035 1 068E ;     SLN R3 ; OK: ACTIVATE PROCESS ;
00003036 1 068F FI ;
00003037 1 068F ;
00003038 1 068F AUTO10: ;
00003039 1 068F ;     MOV C (<:F:>) R2 ;
00003040 1 0690 ;     MOV C (<:1:>) R0 ; DRIVE 1 2ND ATTEMPT ;
00003041 1 0691 ;     JMPI S6 FDBLOAD ; LOAD BOOT MODULE ;
00003042 1 0692 ;     JMPI & ERROR ; (BAD: CALL UTILITIES, ;
00003043 1 0693 ;     JMPI S6 OUTNL ; ;
00003044 1 0694 IF AMOS THEN ;
00003045 1 0694 ;     LDN R3 ; OK: ACTIVATE PROCESS ;
00003046 1 0694 ELSE ;
00003047 1 0694 ;     SLN R3 ; OK: ACTIVATE PROCESS ;
00003048 1 0695 FI ;
00003049 1 0695 ;
00003050 1 0695 AUTODMA: ;
00003051 1 0695 ;
00003052 1 0695 ;     JMP S6 AUTO90 ; SETUP DEFAULT FD-DRIVE ADDRESS ;
00003053 1 0696 ;     JMPI & ERROR ; (BAD: CALL UTILITIES, OK: ;
00003054 1 0697 ;     MOV C (<:X:>) R2 ; SETUP INDIRECT BOOT VIA DMA ;
00003055 1 0698 ;     MOV C (<:1:>) R1 ; DRIVE 0 1ST ATTEMPT ;
00003056 1 0699 ;     JMPI S6 FDBLOAD ; LOAD BOOT MODULE ;
00003057 1 069A ;     JMP & AUTO20 ; (BAD: TRY WITH DRIVE 1, ;
00003058 1 069B ;     JMPI S6 OUTNL ; ;
00003059 1 069C IF AMOS THEN ;
00003060 1 069C ;     LDN R3 ; OK: ACTIVATE PROCESS ;
00003061 1 069C ELSE ;
00003062 1 069C ;     SLN R3 ; OK: ACTIVATE PROCESS ;
00003063 1 069D FI ;

```

```

00003062 1 069D SLN R3 ; OK: ACTIVATE PROCESS);
00003063 1 069D FI ;
00003064 1 069D AUTO20: ;
00003065 1 069D MOVC (:X:) R2 ;
00003066 1 069E MOVC (:0:) R0 ; DRIVE 0 2ND ATTEMPT;
00003067 1 069F JMPI S6 FDBLOAD ; LOAD_BOOT_MODULE
00003068 1 06A0 JMPI & ERROR ; (BAD: CALL UTILITIES,
00003069 1 06A1 JMPI S6 OUTNL ;
00003070 1 06A2 IF AMOS THEN ;
00003071 1 06A2 LDN R3 ; OK: ACTIVATE PROCESS);
00003072 1 06A2 ELSE ;
00003073 1 06A2 SLN R3 ; OK: ACTIVATE PROCESS);
00003074 1 06A3 FI ;
00003075 1 06A3 JFF ;

```

```

00003076 1 06A3 ;
00003077 1 06A3 FUNCTION:

```

```

00003076 1 06A3 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00003077 1 06A3 ;FUNCTION:
00003078 1 06A3 ;      BOOT FROM DISKETTE (IF POSSIBLE)
00003079 1 06A3 ;      COPY OTHER MEMORY SECTION AS SPECIFIED "AMUPROCESS+1"
00003080 1 06A3 ;      AND EXECUTE (CALL COPYLANDEXECUTE).
00003081 1 06A3 ;
00003082 1 06A3 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00003083 1 06A3 ;
00003084 1 06A3 AUTOEXTERNAL:
00003085 1 06A3 ;-----
00003086 1 06A3
00003087 1 06A3      JMP      S6 AUTO90      ; SETUP DEFAULT FD-DRIVE ADDRESS;
00003088 1 06A4      JMPI &      ERROR      ; (BAD: CALL UTILITIES, OK:
00003089 1 06A5      MOVC <:F:>      R2      ; SETUP DIRECT BOOT LOAD)
00003090 1 06A6      MOVC <:0:>      R0      ; DRIVE 0 1ST CANDIDATE;
00003091 1 06A7      JMPI      S6 FDBLOAD      ; LOAD_BOOT_MODULE
00003092 1 06A8      JMP &      AUTO50      ; (BAD: TRY DRIVE 1,
00003093 1 06A9      JMPI      S6 OUTNL      ;
00003094 1 06AA IF AMOS THEN
00003095 1 06AA      LDN      R3      ; OK: ACTIVATE PROCESS);
00003096 1 06AA ELSE
00003097 1 06AA      SLN      R3      ; OK: ACTIVATE PROCESS);
00003098 1 06AB FI
00003099 1 06AB AUTO50:
00003100 1 06AB      MOVC <:F:>      R2      ;
00003101 1 06AC      MOVC <:1:>      R0      ; DRIVE 1 2ND ATTEMPT;
00003102 1 06AD      JMPI      S6 FDBLOAD      ; LOAD_BOOT_MODULE
00003103 1 06AE      JMP &      AUTO60      ; (BAD: TRY EXTERNAL PROCESS,
00003104 1 06AF      JMPI      S6 OUTNL      ;
00003105 1 06B0 IF AMOS THEN
00003106 1 06B0      LDN      R3      ; OK: ACTIVATE PROCESS);
00003107 1 06B0 ELSE
00003108 1 06B0      SLN      R3      ; OK: ACTIVATE PROCESS);
00003109 1 06B1 FI
00003110 1 06B1 AUTO60:
00003111 1 06B1      MOV      PROGSTART+8      R4      ;
00003112 1 06B2      XTR      R4 4      ; ISOLATE MEMORY SECTION;
00003113 1 06B3      MOD      R4      ;
00003114 1 06B4      MOV      SB16PAGE      R4      ; GET ASSOCIATED PSW;
00003115 1 06B5      MOV      PROGSTART+8      R5      ;
00003116 1 06B6      SRL      R5 4      ; ISOLATE
00003117 1 06B7      SLL      R5 4      ; PROGRAM_START_ADDRESS;
00003118 1 06B8      SUB      XBASE      R5      ; MAKE RELATIVE OWN BASE;
00003119 1 06BA      SVS      R7      ; GET OWN PSW;
00003120 1 06BB      LDS      R4      ; SWITCH TO EXTERNAL PROGRAM;
00003121 1 06BC      MOV      PROGSTART.X5      R0      ; GET FIRST EXTERNAL PROGRAM-WORD;
00003122 1 06BD      SEQF      R0 #1A4      ; IF NOT ?MODE -1?-INSTRUCTION
00003123 1 06BF      LDS &      R7      ; THEN
00003124 1 06C0      JMPI &      ERROR      ; NOT VALID EXTERNAL PROGRAM;
00003125 1 06C1      MOV      AMUPROCESS+1.X5      R3      ; GET SIZE OF EXTERNAL MODULE;
00003126 1 06C2      MOV      R5      R6      ;

```


00003125	1	06C1	MOV	AMUPROCESS+1,X5	R3	;	GET SIZE OF EXTERNAL MODULE	;
00003126	1	06C2	MOV	R5	R6	;		;
00003127	1	06C3	ADD	AMUPROCESS,X5	R6	;	POINT BUILT-IN BASE	;
00003128	1	06C4	MOV	XBASE,X6	R1	;	TO GET DEFAULT BASE	;
00003129	1	06C5	LDS	R7		;	RETURN OWN PAGE	;
00003130	1	06C6	MOV	XPROG	R2	;	GET AMU PROG	;
)	1	06C7	SUB	R3	R2	;	COMPUTE NEWLPROG (JUST BELOW AMU)	;
00003131	1	06C8	JMP	UT1410		;	COPY_AND_EXECUTE	;
00003132	1	06CA						;
00003133	1	06CA	JFF					;

00003134 1 06CA ;;
00003135 1 06CA ;PROCEDURE SETUP_DEFAULT_FBI_DRIVE_ADDRESS;

```

00003134 1 06CA ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00003135 1 06CA ;PROCEDURE SETUP_DEFAULT_FD_DRIVE_ADDRESS;
00003136 1 06CA ;
00003137 1 06CA ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00003138 1 06CA ;
00003139 1 06CA AUTO90: ;SETUP_DEFAULT_FD_DRIVE_ADDRESS;
00003140 1 06CA JMP S4 AUTO92 ;
00003141 1 06CB AUTO95 ;
00003142 1 06CC AUTO92: MOV 0. X4 R0 ; GET NEW LOCAL ACTION LOCATION;
00003143 1 06CD MOV XLOCACT R4 ; SAVE PRESENT LOCACT DURING SIO;
00003144 1 06CE MOV R0 XLOCACT ; SETUP NEW LOCAL INTERRUPT;
00003145 1 06CF MOV 0 R0 ; PRESET
00003146 1 06D0 MOV R0 XCAUSE ; INTERRUPT CAUSE FIELD;
00003147 1 06D1 MOV FDDEFAULT R0 ; SETUP DEFAULT FD-DRIVE
00003148 1 06D2 MOV R0 FDADR ; DEVICE CONTROLLER ADDRESS;
00003149 1 06D3 SID R0 ; SENSE IF CONTROLLER EXIST;
00003150 1 06D4 AUTO95: ;
00003151 1 06D4 MOV R4 XLOCACT ; REESTABLISH OLD LOCAL ACTION;
00003152 1 06D5 MOV XCAUSE R0 ; GET CAUSE;
00003153 1 06D6 INE R0 0 ; IF NON-EXISTING
00003154 1 06D7 JMP 0. X6 ; THEN BAD RETURN
00003155 1 06D8 JMP 1. X6 ; ELSE OK_RETURN;
00003156 1 06D9 JFF

```

```

FR/811020 ;
00003157 1 06D9 ;
00003158 1 06D9 ; M A S T E R   C L E A R   E N T R Y   P O I N T ;
00003159 1 06D9 ; ENTRY POINT UPON MASTER CLEAR ;
00003160 1 06D9 ;
00003161 1 06D9 ; R0   *** AMU ENTRY-POINT - FOR USAGE BY OTHER PROGRAMS *** ;
00003162 1 06D9 ; R1   BASE DUMMY PROCESS FOR SAVE_PROCESS OF LOOSERS'S PROCESS ;
00003163 1 06D9 ;      AS RESULT OF CPU-INTERRUPT FROM WINNING CPU. ;
00003164 1 06D9 ; R2   MSEC TO SLAPSE UNTIL WINNER CLAIMS MEMORY ;
00003165 1 06D9 ; R3   AMU BASE ADDRESS ;
00003166 1 06D9 ; R4   REFERENCE FROM LOCATION "AMUCONTINUATION" WHICH DETERMINES ;
00003167 1 06D9 ;      WHICH OF THE BUILT-IN START-UP SEQUENCES TO BE ACTIVATED ;
00003168 1 06D9 ;      WHEN MASTERCLEAR INITIALIZATION HAS COMPLETED. ;
00003169 1 06D9 ; R5   PROGRAM START ADDRESS ;
00003170 1 06D9 ; R6   BASE ADDRESS (XR0) ;
00003171 1 06D9 ; R7   HALF_SIZE PROM-AREA TO BE COPIED ;
00003172 1 06D9 ;
00003173 1 06D9 ;
00003174 1 06D9 MCE05: ; ENTRY_POINT UPON MASTER CLEAR ;
00003175 1 06D9      MOV      AMUPROCESS      R0 ; GET OFFSET TO BUILT-IN PROCESS ;
00003176 1 06DA      ADD      R0              R5 ; COMPUTE DESCRIPTOR ADDRESS ;
00003177 1 06DB      SUB      R3              R5 ; MAKE REL AMU BASE ;
00003178 1 06DC ;
00003179 1 06DC ;      FOR I:=1 UNTIL PROM_AREA_SIZE ;
00003180 1 06DC MCE05: ; DO ;
00003181 1 06DC      MOVL    0.          X5      R01 ; COPY ;
00003182 1 06DD      MOVL    R01 0.      X6      ; PROM_AREA ;
00003183 1 06DE      ADDC    2              R5 ;
00003184 1 06DF      ADDC    2              R6 ;
00003185 1 06E0      SOB      R7          MCE05 ; END FOR_UNTIL ;
00003186 1 06E1      MOVL    XR0.      X7      R01 ; RE-ESTABLISH RED 0,1 CONTENT ;
00003187 1 06E2 ;
00003188 1 06E2 ; ENSURE AGAINST OVERLAP MASTERCLEAR ;
00003189 1 06E2 ; BY WAITING LONGER THAN SLOWEST ;
00003190 1 06E2 ; CPU START EXECUTING AMU. ;
00003191 1 06E2      MOV      R2              R0 ; REPEAT ;
00003192 1 06E3 MCE10:      MOV      MSEC      R6 ;
00003193 1 06E4      SOB      R6          LOC      ; WAIT 1 MSEC ;
00003194 1 06E5      SOB      R2          MCE10 ; WAIT SPECIFIED MSEC ;
00003195 1 06E6 ;
00003196 1 06E6      MOVC    SEMAPHORE      R6 ; POINT CPU_SEMAPHORE ;
00003197 1 06E7      RESS      X6          BIT0 ; RESERVE MEMORY ;
00003198 1 06E8      JMP      MCE15 ; (LOST: WAIT FOR CPU_INTERRUPT, ;
00003199 1 06E9 ;      WON: CONTINUE) ;
00003200 1 06E9      JMP      MCE30 ;
00003201 1 06EA ;
00003202 1 06EA ; WAIT FOR WINNER'S CPU INTERRUPT ;
00003203 1 06EA ;
00003204 1 06EA ; LOOSER'S ;
00003205 1 06EA MCE15: ;
00003206 1 06EA      MOV      R1          XBASE ; SET-UP FOR SLEEPING PROCESS ;
00003207 1 06EB      MOV      R1          XMOD ;
00003208 1 06EC      JMP      S6          MCE20 ; ESTABLISH POINTER ;
00003209 1 06EB      JMP      MCE25 ;
00003210 1 06EB MCE20:      MOV      0.      X4      R0 ; GET PROGRAM_POINTER ;

```

```

00003205 1 06EA MCE15: MOV R1 XBASE ; SET-UP FOR SLEEPING PROCESS
00003206 1 06EB MOV R1 XMOD ;
00003207 1 06EB JMP S6 MCE20 ; ESTABLISH POINTER;
00003208 1 06EC MCE25 ;
00003209 1 06EB MVF 0. X6 R0 ; SET PROGRAM POINTER;
00003210 1 06EE MCE20: ADD XPROG R0 ;
00003211 1 06EF MOV R0 XPRPC ; MODIFY PRPC;
00003212 1 06F1 ;
00003213 1 06F2 IF AMOS THEN LDN R3 ; AWAKEN SLEEPING PROCESS;
00003214 1 06F2 ;
00003215 1 06F2 ELSE SLN R3 ; AWAKEN SLEEPING PROCESS;
00003216 1 06F2 ;
00003217 1 06F3 FI ;
00003218 1 06F3 ;
00003219 1 06F3 MCE25: LDN 2#101 ; ENABLE CPU INTERRUPTS ONLY;
00003220 1 06F4 JMP LOC ; WAIT FOR INTERRUPT;
00003221 1 06F5 ;
00003222 1 06F5 ; INITIALIZE WINNING PROCESS ;
00003223 1 06F5 ;
00003224 1 06F5 MCE30: ; INITIALIZE;
00003225 1 06F5 ADD R0 R0 ; REPEAT
00003226 1 06F6 MCE35: MOV MSEC R6 ;
00003227 1 06F7 SOB R6 LOC ;
00003228 1 06F8 SOB R0 MCE35 ; UNTIL 2*SPECIFIED MSEC'S;
00003229 1 06F9 ;
00003230 1 06F9 MOV R3 XBASE ; RE-ESTABLISH BASE
00003231 1 06FA MOV R3 XMOD ; AND MODIFY REGISTER.
00003232 1 06FB ;
00003233 1 06FB NOLIST IF SCNMODE THEN LIST ;
00003234 1 06FB MCE40: ; RETRY_AUTO_BOOT:
00003235 1 06FB MOV SCNADR R7 ; GET OC'S DEVICE ADDRESS;
00003236 1 06FC MODC SCNSTATUS ;
00003237 1 06FD SIO R0 R7 ; SENSE(STATUS, DEVICE);
00003238 1 06FE MOV SCNASYNC 2# R1 ;
00003239 1 06FF MODC SCMMR1 ;
00003240 1 0700 CIO R1 R7 ; LOAD MODE REGISTER 1;
00003241 1 0701 IF AMOS THEN ;
00003242 1 0701 MOV - R0 R2 ;
00003243 1 0701 SRL R2 BIT11 ; POSITION FRONT-PANEL SWITCH
00003244 1 0701 IBN R2 BIT15-BIT11 ; IF AUTOMATIC MODE
00003245 1 0701 MOV 2#0101 R2 ;... THEN FORCE 300 BAUD;
00003246 1 0701 ELSE ;
00003247 1 0701 MOV 2#0111 R2 ; FORCE 1200 BAUD
00003248 1 0702 FI ;
00003249 1 0702 XTR R2 4 ; ISOLATE SWITCH SETTING (BAUD);
00003250 1 0703 ADDC SCMBAUD R2 ; SET-UP PROPER MODE REG 2;
00003251 1 0704 MODC SCMMR2 ;
00003252 1 0705 CIO R2 R7 ; LOAD MODE REGISTER 2;
00003253 1 0706 MOV SCMRESET R1 ;
00003254 1 0707 MODC SCMCMD ; ENABLE
00003255 1 0708 CIO R1 R7 ; SCM FOR INPUT/OUTPUT;
00003256 1 0709 ;
00003257 1 0709 SBZ R0 BIT15 ; IF MANUAL MODE
00003258 1 070A JMP MCE70 ; THEN

```

Her genereret et local int cause?
når level løses, andre gange local int cause 4
årsag: Level og Bound kopieres ikke ud
i RAM under initialisering.

00003292 1 0729

```

00003291 1 0729 ; A U T O M A T I C   M O D E
00003292 1 0729 ;
00003293 1 0729 ;   A U T O   M O D E
00003294 1 0729 ; R0   S C M   S T A T U S   W O R D
00003295 1 0729 ;   B I T 1 2 , 1 1   D R I V E   I D E N T I F I C A T I O N
00003296 1 0729 ;   B I T 1 3   T Y P E   O F   D E V I C E :   0 = F D ,   1 = C D C - D I S K
00003297 1 0729 ;   B I T 1 4   B O O T   I N D I R E C T   V I A   D M A - C H A N N E L
00003298 1 0729 ;   B I T 1 5   A U T O   M O D E   F L A G   ( H I G H )
00003299 1 0729 ;
00003300 1 0729 ;
00003301 1 0729 MCE70: ; A U T O _ M O D E :
00003302 1 0729   S E Z   R0   B I T 1 3   ;   I F   A U T O   L O A D   F R O M   F L O P P Y
00003303 1 072A   J M P   M C E 8 0   ;   T H E N
00003304 1 072B   M O V   R0   R2   ;   F O R C E   D I R E C T   L O A D   A S   D E F A U L T
00003305 1 072C   I B N   R2   B I T 1 4   ;   I F   R E Q U E S T   F O R   V I A   D M A
00003306 1 072D   M O V C   < : X : >   R2   ;   T H E N   P R E P A R E   I N D I R E C T   L O A D
00003307 1 072E   S R L   R0   B I T 1 1   ;   P O S I T I O N   D R I V E   N U M B E R
00003308 1 072F   J M P I   S 6   F D B L O A D   ;   L O A D _ F I L E   F R O M   F D
00003309 1 0730   J M P &   M C E 4 0   ;   F A I L E D : ( R E T R Y )
00003310 1 0731 I F A M O S   T H E N
00003311 1 0731   L D N   R3   ; ***   A C T I V A T E   L O A D E D   P R O C E S S
00003312 1 0731 E L S E
00003313 1 0731   S L N   R3   ; ***   A C T I V A T E   L O A D E D   P R O C E S S
00003314 1 0732 F I
00003315 1 0732 M C E 8 0 :   ;   E L S E
00003316 1 0732   T R P   0   ;   I N V A L I D   S E T T I N G
00003317 1 0733
00003318 1 0733 N O L I S T   F I   L I S T
00003319 1 0733
00003320 1 0733
00003321 1 0733 ; //////////////////////////////////////// ;   E N D   B A S I C _ I N I T I A L I Z A T I O N
00003322 1 0733 M C E 9 0 : ; R E T U R N _ C A L L E R :
00003323 1 0733   M O V   X O L D P R C   R7   ;   G E T   B A S E   C A L L I N G   P R O C E S S
00003324 1 0734 I F A M O S   T H E N
00003325 1 0734   L D N   R7   ;   R E - A C T I V A T E   C A L L I N G   P R O C E S S
00003326 1 0734 E L S E
00003327 1 0734   S L N   R7   ;   R E - A C T I V A T E   C A L L I N G   P R O C E S S
00003328 1 0733 F I
00003329 1 0733 E N D O F P R G M :
00003330 1 0733 ; F F

```

```

00003331 1 0735 PROGRAMSTART:= CPUROM+PROGSTART
00003332 1 0735 PROGRAMBO:= PROGRAMSTART+MCENTRYPOINT
00003333 1 0735 ABSMCE50:= PROGRAMSTART+MCE50
00003334 1 0735
00003335 1 0735
00003336 1 0735 IF LOC LT #0FE0
00003337 1 0735 THEN #FFFF, REPEAT (#FE0-LOC) AN #7FF
00003338 1 07E0 ELSE MESSAGE LOC FI
00003339 1 07E0
00003340 1 07E0 RELPROMBASE:
00003341 1 07E0 USE BASE
00003342 0 0004 LOC= ENDOFDATA
00003343 0 0019 IF LOC LT #1E
00003344 0 0019 #FFFF
00003345 0 001A IF LOC LT #1E
00003346 0 001A REPEAT (#1E-LOC)
00003347 0 001E FI
00003348 0 001E FI
00003349 0 001E PROGRAMSTART+EMERGENCY ; EMERGENCY ACTION SERVICE PROGRAM;
00003350 0 001F #0000 ; PROM CHECK-SUM
00003351 0 0020 BEGIN
00003352 0 0020 EXTERNAL EOT
00003353 0 0020 EXTERNAL BIT0, BIT1, BIT2, BIT3, BIT4, BIT5, BIT6, BIT7,
00003354 0 0020 EXTERNAL BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15
00003355 0 0020 USE BASE
00003356 0 0020 EXTERNAL CPUMON
00003357 0 0020 EXTERNAL XBASE, XPROG
00003358 0 0020 EXTERNAL TCBASE, TCREFWORK
00003359 0 0020 EXTERNAL B5555, BAAAA, CP06, CP20INS, CPWORK
00003360 0 0020 EXTERNAL CP46P, CP46
00003361 0 0020 USE PROG
00003362 1 07E0 EXTERNAL MAINLINE, P5555, PAAAA, ANUPROCESS
00003363 1 07E0 NAMES

```

```

; 00003363 1 07E0 LEVEL 2 NAMES:

```

```

BIT2= #0002
BIT3= #0003
BIT4= #0004
BIT5= #0005
BIT6= #0006
BIT7= #0007
BIT8= #0008
BIT9= #0009
BIT10= #000A
BIT11= #000B
BIT12= #000C
BIT13= #000D
BIT14= #000E
BIT15= #000F
EOT= #00FF
BIT0= #0000
BIT1= #0001

```

```

USE BASE

```

```

PAAAA:= #0020

```

BIT14= #0001
BIT15= #000F
EOT= #00FF
BIT0= #0000
BIT1= #0001

USE BASE

BAAAA:= #0028
CP2015:= #0029
XBASE:= #0008
B5555:= #0027
CP46P:= #002D
CPUMON:= #0040
CP06:= #0029
CPWORK:= #0029
CP46:= #002E
TCBASE:= #0018
XPROG:= #000A
TCREFK:= #0024

USE PROG

PAAAA:= #0002
MAINLE:= #0003
P5555:= #0001
AMUPRS:= #0004

USE #1

00003364 1 07E0 END
00003365 0 0020 JFF

00003366 0 0020 END

Z0FB00 T0000

P

F800L 01A4 5555 AAAA 061F 07E0 0800 0000 0000
F808L 0003 0000 0000 0000 0000 0000 0000 0000
F810L 0727 0687 0695 06B1 06A3 0000 0000 0000
F818L 0001 0002 0007 E000 E004 E008 E00C 02D7
F820L 00D8 0294 010D 00E4 00A4 00B0 0100 00AF
F828L 00CF 0123 0449 0416 028A 01E7 01C0 02A0
F830L 02FD 0436 0155 03F3 0220 037C 0733 024A
F838L 05E6 03A6 0317 0286 047D 0480 0506 0553
F840L 05BE 05A5 20BE 022F C027 1068 012A 114F
F848L F2EF 0D4F 7B11 02D8 A07C 2FD8 0D6F 7FA3
F850L E7BC 2BFD 4C0D 434F 4C41 4920 544E 5245
F858L 5552 5450 202C 4143 5355 3D45 00FF 26FF
F860L 2AFF 2BFD 4C0D 434F 4C41 5220 5445 5255
F868L 204E 4441 5244 5345 2053 00FF 106A 26FF
F870L 2BFD 202C 4E45 4554 2052 464F 5346 5445
F878L 0020 00FF 23FF 1068 208A 084C 0230 0B70
F880L 4C3C 40BE 0B14 222C 0B68 E056 004F E7BC
F888L 0A1E 068B 02AE 4EE9 4EE9 1E4E 9348 8A30
F890L 8BB0 8B1F 17BD 2BFD 450D 454D 4752 4E45
F898L 5943 4120 5443 4F49 204E 5942 0020 00FF
F8A0L 0E6A 26FF 2AFF 0258 189F 2748 30E5 8F96
F8A8L 10E5 0F96 01F0 0858 0F97 07EC 01D8 0D48
F8B0L 189F 10E5 7F96 75B2 71F2 13D8 70F0 0858
F8B8L 189F 8F97 0FBB 78EC FD7E 80BF 1976 03A8
F8C0L 115F 0AAB 135F 145F 155F 165F 175F 0DC8
F8C8L 19E7 189F 3748 30E5 8F96 0F97 7BDB 0020
F8D0L FFE4 807C 01BF 25FF 60BC 25FF 01CC 0958
F8D8L 189F 10E5 7F96 75F2 71F0 81BF 189F 3748
F8E0L 30E5 8F96 0F97 80BF 1A76 0049 004A 24FF
F8E8L 7FE4 807E 03D8 2A48 25FF 0958 0EBD 3AAE
F8F0L E746 0A0B 03CE EA42 07CE 10E4 E044 04D8
F8F8L A4A0 628A 0129 1558 8D3E 117E 1858 1AE7
F900L 1B76 0449 A4A2 2B3B 04EC 8A40 07C8 30C8
F908L 25FF 08C1 2048 25FF 1BE7 1C76 24FF 30A8
F910L 8446 70B0 1B9C 23FF 2DBB 0B14 00CD 2EE4
F918L 807E 03D8 23FF 258A 0658 014A 2BE4 807C
F920L 23FF 2B3B 1CE7 1D76 1E76 0CDD E7BC 0049
F928L 81BD 1169 927E 03D8 1E71 E6BC 0FD8 937C
F930L 10D8 F058 0F74 044A 72BD 1A9E 804B D7BC
F938L 3BBB D9A0 0256 004C E6BC 5BFB 8DBB 01CD
F940L 04C4 0AC3 E7BC 10C2 4248 0F70 1E68 827E
F948L 1DE7 2BFD 202C 454D 204D 4150 4952 5954
F950L 4520 5252 524F 00FF 1DE7 0198 2070 0298
F958L 2170 22FF 8D7C 05D8 23FF 2072 8D7E 23FF
F960L 2172 02DF 013A 017C 8020 0F70 8120 0A94
F968L 00C8 0B14 00C8 2270 3EBB 2068 2169 226A
F970L 07BC E4BC 52F7 093B 4062 40D0 0256 0048
F978L 01C0 4066 40B2 06D8 80F0 4062 40D0 0256
F980L 004B 01C0 4060 40B2 E7BC 206A 216B 02B8
F988L 13F8 31D8 E7BC 004A 82BD 156A 217E 2BD8
F990L 5555 5450 202C 4143 5355 3D45 00FF 26FF

F968L 0008 0814 0008 2270 3EBB 2068 2169 226A
F970L 07BC 143C 53F7 0908 4081 4090 025A 0048
F978L 01C0 4060 40D2 08D8 80F0 4062 40D0 025A
F980L 0048 01C0 4060 40D2 E7BC 206A 216B 02BB
F988L 13F8 31D8 E7BC 004A 82BD 156A 217E 2BD8
F990L 20BE 114A 2170 17D8 A23F A37F 5559 0128
F998L 04EE 827E 09D8 28FD 4150 4952 5954 4520
F9A0L 5232 524F 00FF 0DD8 28FD 4951 454D 4F2D
F9A8L 5455 00FF 06D8 28FD 494D 4D53 5441 4843
F9B0L 00FF 28FD 4120 2054 00FF 086A 528A 26FF
F9B8L 27FF 004F A9BE 02CD 01AE E07E 52C6 8BDA
F9C0L 22FF 0869 F056 004A 8D7C 07D8 23FF 2072
F9C8L F056 004A 8D7E 23FF 2069 2EBB 0814 00CA
F9D0L 20BE 0048 D7BC E4BC 451B E7BC 21FF 004F
F9D8L A9BE 2DBB 4435 4381 4586 4881 4981 4A60
F9E0L 01BB 618A 68B0 4A80 0894 00CD 15B0 22FF
F9E8L 2AE4 807C 1CD8 538A 8D7C 04D8 23FF 2BBB
F9F0L 0814 01CB D424 D4A0 538B 5B8B 03EC 8B3C
F9F8L 2048 25FF 84F0 27FF 086A 528A 26FF D7BC
FA00L E4BC 5AFB E7BC 26FF 01CD 11C3 18D8 23FF
FA08L 004B 5FE4 803C 2BBB 23FF 28BB 01CB 1569
FA10L D7BC E4BC 5AFB 117E 07D8 E7BC 20BE 26FF
FA18L 004F A9BE D7BC E4BC 0BC0 0CC3 E7BC 2ADA
FA20L 22FF 2AE4 807C 15D8 B17C 10D8 23FF D7BC
FA28L E4BC 117C ADBB E7BC 01CD 07C3 1AD8 23FF
FA30L D7BC E4BC 117C ADBB E7BC 01CD 8D7E 095B
FA38L 10D8 23FF 004B 5FE4 803C 2BBB 23FF 2072
FA40L 23FF 2068 01CB D7BC E4BC ADBB 02C0 03C3
FA48L E7BC 01DA 22FF 23FF 2072 012A 2FE4 807C
FA50L 23FF 2068 D7BC E4BC A07C 18D8 5EFB 268C
FA58L 06FA 11D8 E7BC 20BE 28FD 414D 4354 2048
FA60L 5441 0020 00FF 086A 528A 26FF 27FF 004F
FA68L A9BE D7BC E4BC 01CD 17C3 16D8 D0F8 12D8
FA70L E7BC 20BE 28FD 494D 4D53 5441 4843 4120
FA78L 2054 00FF 086A 528A 26FF 27FF 004F A9BE
FA80L D7BC E4BC 01CD 16C3 E7BC C5D9 22FF 0894
FA88L 00CD 45BD 22FF 23FF 0814 00CA 2FE4 807C
FA90L 24FF 30AB 01DF B6D9 B446 70BD 1B98 D7BC
FA98L E4BC 59FB E0BC 9ABB 528E 06C3 E7BC 80BF
FAA0L 004B 2173 2273 28FD 2D4F 00FF 24FF 4FAB
FAABL 2070 22FF 29E4 807E 04D8 0894 00CD 2175
FAB0L 22FF 0894 00CD 5FE4 807E 05D8 23FF A07C
FAB8L 01AB 2273 2BBB 206C C47C 0CD8 C37C 0ED8
FAC0L 23FF 4C7C 0FD8 447C 11D8 C87C 13D8 C77C
FAC8L 15D8 3CFC 17DF 77BD 2D96 1BD8 13DF 77BD
FAD0L 2D97 17D8 0FDF 77BD AD96 1ED8 0BDF 77BD
FAD8L AD96 19D8 07DF 77BD AD97 16D8 03DF 77BD
FAE0L AD97 11D8 004C 216F F07C 01CE 2076 216F
FAE8L 20E7 156F 717E 08D8 26FF 01CC 48BB 03EC
FAF0L 807C 27FF 01D8 23FF 0EC3 20FF 05D8 226E
FAF8L 01AE 613E 2276 155B 4DD9 24FF 46E4 807C
FB00L 0CDB 0068 0169 026A 036B 046C 066E 076F
FB08L 054D 85BC 20FF 2AEC 0C58 0068 0169 026A
FB10L 036B 046C 066E 076F 054D 85BC 035B 23FF

FB08L 054D 055C 20FF 2AFC 0C58 0068 0169 026A
FB10L 036B 046C 066E 076F 054D 85BC 0358 23FF
FB18L 26EC 086D 21BE 14D8 D6BE 189F 10E5 0F96
FB20L 01FC 05F0 075B D73E 2BFD 494D 5353 4E49
FB28L 2047 4E49 4554 5252 5D55 0D54 00FF 45BC
FB30L 113C 1DE5 0814 0894 0048 F5BC 0156 B04C
FB38L 548A 404F 8CBB 01CC 03C7 0ECC 0156 B04B
FB40L BDBB 4183 4283 4383 50CB 4483 022F 7CE9
FB48L 7CE9 1EAC 0883 0983 0814 0156 804D 0869
FB50L 9DBB 1895 9DBB 72BD 9DBB 213E 438C 20FF
FB58L F2D8 126A 217C 3CFC 1568 017C A07C 1BD8
FB60L 189F A224 27F8 17D8 28FD 4F49 4920 544E
FB68L 5245 5552 5450 202C 4544 3D56 00FF 26FF
FB70L 28FD 202C 5250 4F49 003D 00FF 1268 02EC
FB78L 30CB 25FF 27FF 3058 24FF 30A8 8844 3CFC
FB80L 01DF C8DB 0814 004C 204F 074A 72BD FCBB
FB88L 2FA3 084B 0882 01CC 03C3 08AC 086A 08CA
FB90L 70BD 0882 21BE 02DB D5BE 015B 748A 022B
FB98L 02AC 3CE9 3CE9 1EAC 0887 0987 748B D2BC
FBA0L D5BE 6FBE 1F9F 01C7 E2BC 80BF 28FD 430D
FBA8L 5550 203A 00FF D5BC 58BB 60BC 03EC 2E5F
FBB0L 084B 314B 0B3A 2E48 25FF 01CC 06C3 1ADD
FBB8L 116E E27C 03D8 E37E 7B5B 2E48 E5BC 004E
FBC0L 86BD 3E8B 64EC E73C 2056 00C8 EF3C 2F56
FBC8L 00C8 25FF 8824 807E 25FF 01CB 40E4 B046
FBD0L 01BF 00BF 0F74 28FD 520D 4D41 203A 00FF
FBD8L 004B 01DD 0CD8 3148 3EBB E424 76BD 1B9E
FBE0L 3FBB FCA0 0814 00CF E6BC 7EFB 295B 28FD
FBE8L 490D 3A4F 2020 00FF 004B 01DD 03D8 3148
FBF0L 6B96 345B 57DB 28FD 500D 4952 544E 4F2D
FBF8L 5455 2020 3E3F 00FF 24FF 5AAB 1570 0048
FC00L 2F70 28FD 4E0D 204F 464F 5420 5345 5354
FC08L 3E20 00FF 23FF 2BBB 004A 8D3E 23FF 32EE
FC10L 01AB 317C 01AA 1772 1673 35D8 199B 2370
FC18L 28FD 4F4F 2054 00FF 24FF 0ABB 28FD 3A44
FC20L 0020 00FF 24FF 23E4 807C 04DF 3DFF 22DB
FC28L 27FF 43BC 2076 2172 23FF 2372 2EE4 807E
FC30L 3048 2EE4 807C 24FF 216A 20E7 199B 2370
FC38L 28FD 414F 2044 00FF 24FF 0ABB 28FD 3A44
FC40L 0020 00FF 24FF 23E4 807C 1C5F 3DFF 02DB
FC48L 01DB 1F76 27FF 4248 0F70 3E48 25FF 24FF
FC50L 4FAB 01DF 095B 0D7C 2BFC 0C7C 2CFC 0B7C
FC58L 2DFC 0A7C 2EFC 067C 2FFC 037C 31FC 027C
FC60L 32FC 807C 33FC 817C 34FC 827C 35FC 837C
FC68L 36FC 847C 37FC 857C 38FC 867C 39FC 887C
FC70L 3AFC 867C 39FC 897C 3BFC 8B7C 29FC 13A4
FC78L 807C 1FE7 25A4 807C 30FC 3F48 25FF 3658
FC80L 2576 02EC 2670 4FAA 2472 1A9F A93C 70E5
FC88L FF97 29FF 274D 004F 214B FDBB 01CD 03C3
FC90L 2F4D 4016 4116 4316 4416 0894 344C 2C74
FC98L 2056 004A 41FF 3EFF 3FFF 40FF 0914 004E
FCA0L 2976 344D 4E60 AC80 1748 A036 01C3 03C3
FCA8L 6BFA 38DB 344D 2E73 3EFF 2E6D 02E4 5DBA
FCB0L 01CD 075B 3FFF 40FF 0040 0889 D43C 4E1E
FCB8L 4F1F 478D 511B B07C 0DD8 0048 E7BC 31BD
FCC0L 8030 01C5 E03C 04C5 E7BC 07C3 EABC 12E4

FC9SL 2056 0044 41FF 3EFF 401F 0014 004E
FCAOL 2976 3449 4E69 AC89 1348 4036 010D 03C3
FCABL 6DFA 35D8 344B 2E73 3EFF 2E6B 02E4 3DBA
FCBOL 01CB 073B 3EFF 40FF 0040 0B67 D43C 4E1E
FCBBL 4F1F 478D 511B B07C 0DD8 0048 E7BC 31BD
FCCOL 8030 01CE E03C 04CF E7BC 07C3 E4BC 12E4
FCCBL 0DFA 1CD8 28FD 202C 5453 5241 3D54 00FF
FCDOL 4E1A 26FF 28FD 202C 4142 4553 203D 00FF
FCDBL 501A 26FF 4F62 D7BC 72BD 256E 81BF 28FD
FCEOL 202C 4E45 5254 0057 00FF 07DB 28FD 202C
FCEBL 4843 4345 534B 4D55 00FF 28FD 4520 5252
FDFOL 524F 00FF 12DB 28FD 202C 4D44 0041 00FF
FCFBL 05DB 28FD 202C 4944 4B53 00FF 28FD 4520
FD0OL 5252 524F 003D 00FF 26FF 25E7 2676 5E38
FD0BL 4319 907C 2C5F 0A4A 31F6 04DB 138B 01CE
FD1OL 05C2 335F 4119 841B 017C 375F BDF2 0ABB
FD1BL 28EC 60BC 128A 02C3 1BE4 A042 03DB 01C8
FD2OL 1AAA 065B 8BA0 20BD 0ABB A724 2BBB 23BA
FD2BL 23BA 1AE4 B000 1AAB 045B A124 276E 26FB
FD3OL 0608 2772 3FFF 276A 2056 00CA 41FF 3FFF
FD3BL 0ABB 28EC 32BA 22BA 35E4 A040 34AA 1BE4
FD4OL A040 1BAA 1056 00CA 41FF 296C C03E 40FF
FD4BL 0040 2976 01A9 9142 28E7 E6C8 07B2 E5A8
FD5OL 1A5B 01C8 2F5B 2A76 3C4C 01C4 246F F97C
FD5BL 0DD8 236F 8056 2F96 26F2 045B FE56 F04A
FD6OL 404C 2FD7 01CA 03C4 2F96 18DB 51DF 0002
FD6BL 8002 0002 0000 0302 0001 46F3 0958 48DF
FD7OL 0080 0002 0006 FEF0 0302 0001 41DF 0002
FD7BL 0002 0002 0002 0302 0001 026A 20F2 01AA
FD8OL A07C 2AE7 27F0 8B5B 4056 004A 41FF 2776
FD8BL 2B4E 8016 64F1 685B 012A 955B 20BE D2BC
FD9OL 0B6C FE56 F04D 2D6B 23BD 2C6F 8021 58FB
FD9BL E3BC 34BD 8FBB E2BC 578E F07C 04CB 09C1
FDAOL 2C77 2D73 004F AB8E 81BF 266F 67BC 72BA
FDBOL 246F F97C 03DB 236F AF96 80BF 2A76 0272
FDBBL 07DF 0002 0002 0002 0002 0302 0000 2AE7
FDBOL 1A9F 004A 30E5 AF97 004A 004A 2F97 23F0
FDCOL 095B 8022 80E5 AF97 8122 90E5 AF97 8222
FDCBL A0E5 AF97 8522 30E5 AF97 8022 00E4 AF97
FDDOL 8322 0B94 00CA 10E5 AF97 8422 20E5 AF97
FDDBL 50E5 AF97 2F97 0356 C04F 278C F07E ED58
FDEOL 1A9F 25F2 095B 60E5 AF97 86BF 274D 4062
FDEBL 0198 0299 02FA 02D8 13F8 30D8 22FF 1056
FDFOL 004B 004B 186A 0B6F 72BB 21FF 0B6B 1869
FDFBL 318B 186A 0A6F 72BB 1FBB 78FB 15E5 887E
FE0OL 3CFC 01DD 28FD 0020 8FBB 2EBB 00DE 80BF
FE0BL 148A 1FBB 304E 0060 BEE4 807C 008B 807C
FE1OL 03DB 8056 807E 208A 71BA 80D0 02CC 02CE
FE1BL 0FF0 0F5B 004F 1677 1777 2F77 27FF 1568
FE2OL 2F6E 801C 017C C07C 30DB B11C 034D 45BA
FE2BL 246F 011B DFFB 24DB 2577 2674 28FD 4554
FE3OL 5453 0020 00FF 266C 4AFB 26FF 28FD 5720
FE3BL 524F 2044 00FF 266C 011A 328B 26FF 28FD
FE4OL 5420 204F 4542 0020 00FF 5AFB 26FF 28FD
FE4BL 0020 0020 0020 0020 0020 0020 0020 0020

[illegible]