

B1

() M
A ?

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

IDENTIFICATION

PRODUCT CODE: AC-U115A-MC
PRODUCT NAME: CZKXAAO KXJ11 CA DIAGNOSTIC
PRODUCT DATE: 20-MAR-1986
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: HENRY ENMAN

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1986 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

D1

KXJ11-CA FUNCTIONAL TEST

MACRO Y05.02 Thursday 03-Apr-86 14:11 Page 2

40
41
42
43
44
45
46

HISTORY
-- --

.REM &

MAR-86 REV. A FIRST RELEASE

&

E1

48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

TABLE OF CONTENTS

.REM E

1.0	GENERAL INFORMATION
1.1	PROGRAM ABSTRACT
1.2	SYSTEM REQUIREMENTS
1.3	RELATED DOCUMENTS AND STANDARDS
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES
1.5	ASSUMPTIONS
2.0	OPERATING INSTRUCTIONS
3.0	ERROR INFORMATION
4.0	PROGRESS REPORT

E

67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

This program is a diagnostic program that is designed to test the functionality of the KXJ11-CA. It can run as a standalone diagnostic or under the XXDP monitor. The purpose of the diagnostic is to provide a thorough functional test of the KXJ11-CA module, and to report any defects found to the operator. It is to be used in a manufacturing or engineering environment. It can be used with the APT system.

The diagnostic consists of two major sections: the KXJ11-CA test code, and the support routines which reside in the arbiter's memory. The support routines include routines to: size arbiter memory, determine what type of processor the arbiter cpu is (KDJ11, KDF11 etc.), establish operating environment (number of KXJ11-CAs to be tested etc.), determine if an IOP set to ID# 15 is resident and is to be used as part of the test system, transfer information to/from the arbiter and KXJ11-CA being tested, and handle interrupts generated by the KXJ11-CA under test.

When the diagnostic is started it will first try to determine which of three modes it is supposed to be testing in. The three modes are SBC mode, Single IOP mode, and Multiple IOP mode.

SBC mode tests the KXJ11-CA as a single board computer. To enter this mode, bit 11 in the software switch register (address 176) must be set = 1, and the IOP ID switch on the KXJ11-CA to be tested must be set to 0 or 1. The module can reside in the backplane with the arbiter or any other backplane or even on a bench with power applied to it. When the KXJ11-CA is configured as a SBC its Q Bus interface is disabled, therefore any tests which require the Q Bus are not performed.

Single IOP mode tests the KXJ11-CA as the only I/O processor in an arbiter system. The KXJ11-CA under test must reside on the arbiter's Q Bus. The ID switch on the KXJ11-CA can be set to any value except 0 or 1.

Multiple IOP mode can test up to 14 KXJ11-CA modules on the arbiter's Q Bus though this is not possible in practice due to Q bus loading limitations. The KXJ11-CAs are tested sequentially - not concurrently. The advantage of Multiple IOP Mode is that it reduces the amount of equipment needed to test a given number of KXJ11-CA modules. The disadvantage is that the modules are not executing test code as they spend time waiting for other KXJ11-CAs being tested to complete their turn. Each KXJ11-CA to be tested requires the arbiter to have a dedicated serial port for program transfer and status reporting. Each KXJ11-CA in an arbiter system must have its IOP ID switch set to a different value. In other words no two KXJ11-CAs in the same arbiter system can have their ID switches set to the same value.

If the diagnostic determines that the SBC test mode is invoked it will jump to the remote loader. The remote loader routine is executed by the

124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180

arbiter. It establishes contact with the KXJ11-CA to be tested via an arbiter serial line port connected to the console serial line port of the KXJ11-CA to be tested. When contact has been successfully made the arbiter will down line load the test program from its memory to the memory of the KXJ11-CA to be tested via the serial line ports.

Once the test program is resident in the KXJ11-CA under test the arbiter is no longer needed, although it will be monitoring its dedicated serial line waiting for characters from the KXJ11 CA console. If the arbiter is needed for other purposes, the connector to the console of the KXJ11-CA can be removed and a terminal, set to the correct baud rate, can be connected in its place to monitor program progress.

If the arbiter determines that SBC mode is not invoked it will execute routines to: determine what type of processor the arbiter cpu is, size arbiter memory, determine if 22 bit addresses are supported: determine if more than one KXJ11 CA is resident in the system, and report the observed configuration. If only one KXJ11-CA is found Single IOP mode is entered. If an IOP with an ID# of 15 is detected the program will question the operator as to whether that IOP is a known good module to be used as part of the test bed. An IOP that is to be used as a known good test module may be either a KXT11-CA or a KXJ11-CA.

The arbiter will establish communication with the KXJ11-CA to be tested, as in SBC mode, and a small program will be down line loaded into the memory of the KXJ11-CA to be tested. This program will attempt to use the DMA functionality of the KXJ11-CA to be tested to execute a DMA transfer of the test code from arbiter memory into the memory of the KXJ11-CA to be tested. Failing this it will load the test code from the arbiter's memory via the arbiter's dedicated serial line port and the console of the KXJ11-CA being tested. The test program starts by itself once it has been transferred to the target KXJ11-CA.

If more than one IOP is detected in the system; Multiple IOP mode is entered. Some hardware requirements must be observed for correct operation of Multiple IOP mode. The arbiter must have a serial line dedicated to each of the KXJ11-CA modules to be tested. The dedicated serial line with the lowest Q bus address (176500) must be connected to the KXJ11-CA with the lowest ID switch setting, and the next lowest address serial line (176510) connected to the KXJ11-CA with the next lowest ID switch setting and so on. The position of the KXJ11-CAs in the arbiter backplane with respect to their ID switch setting does not matter.

The arbiter establishes communication with the KXJ11-CA to be tested and uses the methods described previously to transfer the program to the target KXJ11-CA. When the KXJ11-CA has successfully loaded the program it will start executing the test code. When the KXJ11-CA has completed a pass it notifies the arbiter by causing an interrupt to the arbiter. The KXJ11-CA then waits for the arbiter to signal it to continue testing when its turn comes again. The arbiter does this by writing to TPR12 of the KXJ11-CA which is to resume testing.

The arbiter's interrupt routine will then load the test code into the next KXJ11-CA to be tested and so on. Once the arbiter has initially transferred the test code into each KXJ11-CA to be tested there is no

181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227

need to reload the test code on subsequent turns.

There are two possible diagnostic operating environments: Operation under the XXDP monitor and operation under APT. When operating in the APT environment certain tests are not performed. Namely the console port data test. If this test were run when in the APT environment, it would could cause characters to be sent to the APT monitor when they were not expected and give a false failure indication to APT.

1.2 SYSTEM REQUIREMENTS

- * KDF11 OR KDJ11 ARBITER PROCESSOR MODULE
- * AT LEAST 56KB OF Q-BUS RANDOM ACCESS MEMORY
- * 22 BIT ADDRESS Q BUS BACKPLANE (18 BIT QBUS MAY BE USED WITH REDUCED TEST COVERAGE)
- * DL COMPATIBLE SERIAL LINE PORT FOR EACH KXJ11-CA TO BE TESTED PLUS ONE FOR THE ARBITER'S CONSOLE PORT.
- * CONSOLE TERMINAL, OR CONNECTION TO APT SERIAL LINE.
- * ONE TO FOURTEEN KXJ11-CA MODULES TO BE TESTED.
- * OPTIONALLY, A Q BUS EXERCISER MAY BE INSTALLED OR A KXT11-CA WITH AN I/O PAGE ADDRESS OF 175740 (ID# 15) MAY BE INSTALLED FOR FURTHER Q BUS INTERFACE TEST COVERAGE

1.3 RELATED DOCUMENTS AND STANDARDS

KXJ11-CA MODULE SPECIFICATION
PDP11 MAINDEC SYSMAC PACKAGE

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

Diagnostic programs for the arbiter system components should be executed successfully prior to invoking the KXJ11-CA diagnostic

1.5 ASSUMPTIONS

It is assumed that the diagnostic operator is familiar with the XXDP+ operating system and the J11 micro-odt.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEEDURE

The diagnostic for the KXJ11-CA is loaded into the arbiter's memory from either a mass storage device connected to the arbiter system using the XXDP operating system commands, or from APT. The diagnostic is then automatically down line loaded into the KXJ11-CA under test via the KXJ11-CA's console serial line port.

229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261

2.2 PROGRAM OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE SUPPORTED:

SWR	OCTAL	FUNCTION
15	100000	HALT ON ERROR
14	040000	INHIBIT ERROR SUMMARY (AT EOP)
13	020000	INHIBIT ERROR REPORTS
12	010000	IOP ID# 15 IS KNOWN GOOD FOR TESTING
11	004000	TEST STAND ALONE IOP
10	002000	EXECUTE EXTENDED MEMORY TESTS
09	001000	LOOP ON ERROR
08	000400	LOOP ON TEST IN SWR<6:0>
07	000200	INHIBIT TEST NUMBER/TITLE

3.0 ERROR INFORMATION

If bit 15 in the software switch register is set then all errors will cause the KXJ11-CA to halt after reporting the error to the arbiter via the \$ERROR routine. All test errors will type an error message, a unique error number, and the PC at the time of the error.

4.0 PROGRESS REPORT

At the end of each pass the id number of the KXJ11-CA under test, the diagnostic name and pass count are printed.

263
264
265
266 000000
279
280

```
.NLIST MC,MD,CND,BEX,TOC ;DON'T LIST MACRO CALLS, MACRO DEFINITIONS,
;CONDITIONAL ASSEMBLERS, BINARY EXTENTIONS,
;OR TABLE OF CONTENTS
```

```
.ENABL ABS,AMA,LC,CRF
.LIST ME
.TITLE KXJ11-CA FUNCTIONAL TEST
;*COPYRIGHT (C) MCMLXXXVI
;*DIGITAL EQUIPMENT CORP.
;MAYNARD, MASS. 01754
```

```
;*
;*PROGRAM BY H. ENMAN
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC 11-DZQAC-C5), JAN, 1981.
```

000001
160000

```
;$TN=1
;$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
```

281
282 101400
283 000200
284 000000
285 000123
286

```
;$SWR= 101400 ; REDEFINE ACTIVE SWITCHES...
;$SWRMK= 000200 ;...AND ADD SWR<7>.
;$EN= 0
;PRGSIZ= <LASTAD/BIT8> ; PROGRAM SIZE (IN 1/8 K UNITS)...
;...EXCLUDING BUFFERS AND REMOTE STUFF.
```

287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312

```
*****
*****
; PROGRAM CONTROL PARAMETERS ARE LOCATED IN THE $ETABLE AREA STARTING
; AT LOC 1216, AND ARE CONFIGURED IAW PROCESSOR TYPE AT ASSEMBLY TIME.
;
; THE PROGRAM WILL BYPASS ANY TEST IN WHICH THE DEVICE/FUNCTION REQUIRED
; APPEARS TO BE NON-EXISTANT (WHETHER INTENTIONAL OR NOT).
; YOU MAY FORCE AN ENTRY TO ANY/ALL TEST(S) VIA SWR<08> OPTION.
```

```
; THE FOLLOWING SWITCH OPTIONS ARE SUPPORTED:
```

SWR	OCTAL	FUNCTION
15	100000	HALT ON ERROR
14	040000	INHIBIT ERROR SUMMARY (AT EOP)
13	020000	INHIBIT ERROR REPORTS
12	010000	IOP ID# 15 IS KNOWN GOOD FOR TESTING
11	004000	EXECUTE EXTENDED MEMORY TESTS
10	002000	UNUSED
09	001000	LOOP ON ERROR
08	000400	LOOP ON TEST IN SWR<6:0>
07	000200	INHIBIT TEST NUMBER/TITLE

```
*****
*****
.SBTTL BASIC DEFINITIONS
```

001100
104000
000004

```
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
ERROR= EMT
SCOPE= IOT
```

000011

```
;*MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB
```

000012	LF=	12	::CODE FOR LINE FEED
000015	CR=	15	::CODE FOR CARRIAGE RETURN
000200	CRLF=	200	::CODE FOR CARRIAGE RETURN-LINE FEED
177776	PS=	177776	::PROCESSOR STATUS WORD
177776	PSW=	PS	
177774	STKLMT=	177774	::STACK LIMIT REGISTER
177772	PIRQ=	177772	::PROGRAM INTERRUPT REQUEST REGISTER
177570	DSWR=	177570	::HARDWARE SWITCH REGISTER
177570	DDISP=	177570	::HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

000000	R0=	%0	::GENERAL REGISTER
000001	R1=	%1	::GENERAL REGISTER
000002	R2=	%2	::GENERAL REGISTER
000003	R3=	%3	::GENERAL REGISTER
000004	R4=	%4	::GENERAL REGISTER
000005	R5=	%5	::GENERAL REGISTER
000006	R6=	%6	::GENERAL REGISTER
000007	R7=	%7	::GENERAL REGISTER
000006	SP=	%6	::STACK POINTER
000007	PC=	%7	::PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS

000000	PR0=	0	::PRIORITY LEVEL 0
000040	PR1=	40	::PRIORITY LEVEL 1
000100	PR2=	100	::PRIORITY LEVEL 2
000140	PR3=	140	::PRIORITY LEVEL 3
000200	PR4=	200	::PRIORITY LEVEL 4
000240	PR5=	240	::PRIORITY LEVEL 5
000300	PR6=	300	::PRIORITY LEVEL 6
000340	PR7=	340	::PRIORITY LEVEL 7

;*“SWITCH REGISTER” SWITCH DEFINITIONS

100000	SW15=	100000
040000	SW14=	40000
020000	SW13=	20000
010000	SW12=	10000
004000	SW11=	4000
002000	SW10=	2000
001000	SW09=	1000
000400	SW08=	400
000200	SW07=	200
000100	SW06=	100
000040	SW05=	40
000020	SW04=	20
000010	SW03=	10
000004	SW02=	4
000002	SW01=	2
000001	SW00=	1
001000	SW9=	SW09
000400	SW8=	SW08
000200	SW7=	SW07
000100	SW6=	SW06
000040	SW5=	SW05
000020	SW4=	SW04
000010	SW3=	SW03
000004	SW2=	SW02

000002
000001SW1= SW01
SW0= SW00100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
BIT9= BIT09
BIT8= BIT08
BIT7= BIT07
BIT6= BIT06
BIT5= BIT05
BIT4= BIT04
BIT3= BIT03
BIT2= BIT02
BIT1= BIT01
BIT0= BIT00000004
000010
000014
000014
000014
000020
000024
000030
000034
000060
000064
000240

;*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14 ;; "T" BIT
TRTVEC= 14 ;;TRACE TRAP
BPTVEC= 14 ;;BREAKPOINT TRAP (RPT)
IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 ;;POWER FAIL
EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34 ;; "TRAP" TRAP
TKVEC= 60 ;;TTY KEYBOARD VECTOR
TPVEC= 64 ;;TTY PRINTER VECTOR
PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR

313

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;*KT11 VECTOR ADDRESS

000250

MMVEC= 250

;*KT11 STATUS REGISTER ADDRESSES

177572
177574
177576
172516SR0= 177572
SR1= 177574
SR2= 177576
SR3= 172516

;*USER "I" PAGE DESCRIPTOR REGISTERS

177600	UIPDR0= 177600
177602	UIPDR1= 177602
177604	UIPDR2= 177604
177606	UIPDR3= 177606
177610	UIPDR4= 177610
177612	UIPDR5= 177612
177614	UIPDR6= 177614
177616	UIPDR7= 177616

;*USER "D" PAGE DESCRIPTOR REGISTORS

177620	UDPDR0= 177620
177622	UDPDR1= 177622
177624	UDPDR2= 177624
177626	UDPDR3= 177626
177630	UDPDR4= 177630
177632	UDPDR5= 177632
177634	UDPDR6= 177634
177636	UDPDR7= 177636

;*USER "I" PAGE ADDRESS REGISTERS

177640	UIPAR0= 177640
177642	UIPAR1= 177642
177644	UIPAR2= 177644
177646	UIPAR3= 177646
177650	UIPAR4= 177650
177652	UIPAR5= 177652
177654	UIPAR6= 177654
177656	UIPAR7= 177656

;*USER "D" PAGE ADDRESS REGISTERS

177660	UDPAR0= 177660
177662	UDPAR1= 177662
177664	UDPAR2= 177664
177666	UDPAR3= 177666
177670	UDPAR4= 177670
177672	UDPAR5= 177672
177674	UDPAR6= 177674
177676	UDPAR7= 177676

;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS

172200	SIPDR0= 172200
172202	SIPDR1= 172202
172204	SIPDR2= 172204
172206	SIPDR3= 172206
172210	SIPDR4= 172210
172212	SIPDR5= 172212
172214	SIPDR6= 172214
172216	SIPDR7= 172216

;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS

N1

172220	SDPDR0= 172220
172222	SDPDR1= 172222
172224	SDPDR2= 172224
172226	SDPDR3= 172226
172230	SDPDR4= 172230
172232	SDPDR5= 172232
172234	SDPDR6= 172234
172236	SDPDR7= 172236

;*SUPERVISOR "I" PAGE ADDRESS REGISTERS

172240	SIPAR0= 172240
172242	SIPAR1= 172242
172244	SIPAR2= 172244
172246	SIPAR3= 172246
172250	SIPAR4= 172250
172252	SIPAR5= 172252
172254	SIPAR6= 172254
172256	SIPAR7= 172256

;*SUPERVISOR "D" PAGE ADDRESS REGISTERS

172260	SDPAR0= 172260
172262	SDPAR1= 172262
172264	SDPAR2= 172264
172266	SDPAR3= 172266
172270	SDPAR4= 172270
172272	SDPAR5= 172272
172274	SDPAR6= 172274
172276	SDPAR7= 172276

;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

172300	KIPDR0= 172300
172302	KIPDR1= 172302
172304	KIPDR2= 172304
172306	KIPDR3= 172306
172310	KIPDR4= 172310
172312	KIPDR5= 172312
172314	KIPDR6= 172314
172316	KIPDR7= 172316

;*KERNEL "D" PAGE DESCRIPTOR REGISTERS

172320	KDPDR0= 172320
172322	KDPDR1= 172322
172324	KDPDR2= 172324
172326	KDPDR3= 172326
172330	KDPDR4= 172330
172332	KDPDR5= 172332
172334	KDPDR6= 172334
172336	KDPDR7= 172336

;*KERNEL "I" PAGE ADDRESS REGISTERS

172340	KIPAR0= 172340
172342	KIPAR1= 172342

```

172344      KIPAR2= 172344
172346      KIPAR3= 172346
172350      KIPAR4= 172350
172352      KIPAR5= 172352
172354      KIPAR6= 172354
172356      KIPAR7= 172356

; *KERNEL "D" PAGE ADDRESS REGISTERS

172360      KDPAR0= 172360
172362      KDPAR1= 172362
172364      KDPAR2= 172364
172366      KDPAR3= 172366
172370      KDPAR4= 172370
172372      KDPAR5= 172372
172374      KDPAR6= 172374
172376      KDPAR7= 172376

314         177572      MMR0= 177572
315         172516      MMR3= 172516
316         177746      CCR= 177746      ;KDJ11A CACHE CONTROL REGISTER
317         ;
318         ; ADD A FEW MORE.
319         ;
320         174470      MMR= 174470      ; MASTER MODE REGISTER.
321         174462      COP1= 174462      ; CHANNEL 1 CURRENT OP COUNT
322         174460      COP2= 174460      ; CHANNEL 2 CURRENT OP COUNT
323         174456      STAT1= 174456     ; CHANNEL 1 STATUS.
324         174454      STAT2= 174454     ; CHANNEL 2 STATUS...
325         174454      CMDR= 174454     ; ...AND COMMAND REGISTER (FOR BOTH).
326         174446      CHA1H= 174446     ; CHAIN ADDRESS REGISTERS HIGH (SEG/TAG)...
327         174444      CHA2H= 174444
328         174442      CHA1L= 174442     ; ...AND THE LOW (OFFSET) HALFS.
329         174440      CHA2L= 174440
330         174432      CARAH1= 174432    ; CURRENT ADDRESS REGISTER A HI CHANNEL 1
331         174412      CARAL1= 174412
332         174422      CARBH1= 174422    ; CURRENT ADDRESS REGISTER B HI CHANNEL 1
333         174402      CARBL1= 174402    ; CURRENT ADDRESS REGISTER B LO CHANNEL 1
334
335         000135      MMRDEF= 135      ; MMR default setting <block lower level hardware
336         000102      SSRCH1= 102      ; Set Software Request CH 1
337         000103      SSRCH2= 103      ; Set Software Request CH 2
338         000240      SCCCH1= 240      ; Start Chain Command CH 1
339         000241      SCCCH2= 241      ; Start Chain Command CH 2
340         000060      CLIE1= 060      ; Clear channel 1 interrupt enable
341         000061      CLIE2= 061      ; Clear channel 2 interrupt enable
342         000062      SETIE1= 062      ; Set channel 1 interrupt enable
343         000063      SETIE2= 063      ; Set channel 2 interrupt enable
344         000054      CLIP1= 054      ; Clear channel 1 current and pending interrupt
345         000055      CLIP2= 055      ; Clear channel 2 current and pending interrupt
346
347         ;
348         ; Chain Reload Word Bit Definitions
349         ;
350         000001      CHAD= 1      ; Chain address
351         000002      MODE= 2      ; Channel Mode Register
352         000004      IV= 4      ; Interrupt Vector
353         000010      PATMSK= 10     ; Pattern and Mask

```

```

353      000020      BOP= 20      ;Base Operation count
354      000040      BARB= 40      ;Base Address Reg B
355      000100      BARA= 100     ;Base Address Reg A
356      000200      COP= 200     ;Current Operation count
357      000400      CARB= 400     ;Current Address Register B
358      001000      CARA= 1000    ;Current Address Register A
359      ;
360      ;DTC Current and Base Address Segment/tag bit definitions
361      0C0000      UP= 0        ;INCREMENT ADDRESS
362      000010      DOWN= 10      ;DECREMENT ADDRESS
363      000020      HOLD= 20      ;HOLD ADDRESS
364      040000      K2IO= 40000   ;ADDRESS IS IN IOP I/O SPACE
365      000000      K2MEM= 0       ;ADDRESS IS IN IOP MEMORY
366      140000      QBUSIO= 140000 ;ADDRESS IS IN Q BUS I/O SPACE
367      100000      QBMEM= 100000 ;ADDRESS IS IN Q BUS MEMORY
368      ;
369      ;DTC Channel mode register bit definitions
370      ;Operation type field
371      000001      TRBB= 1        ;BYTE-BYTE TRANSFER
372      000010      TRFUNL= 10     ;FUNNEL TRANSFER BYTE - WORD
373      000000      TRWM= 0        ;WORD WORD TRANSFER
374      000005      STBB= 5        ;BYTE-BYTE SEARCH TRANSFER
375      000014      STFUNL= 14     ;FUNNEL SEARCH TRANSFER BYTE - WORD
376      000004      STWM= 4        ;WORD-WORD SEARCH TRANSFER
377      000017      SRCHBB= 17     ;BYTE-BYTE SEARCH
378      000012      SRCHWM= 12     ;WORD-WORD SERRCH
379      ;
380      000020      FLIP= 20       ;FLIP BIT
381      ;Transfer type field
382      000000      SNGL= 0        ;SINGLE TRANSFER-HARDWARE INITIATED
383      000040      BUSHOG= 40     ;DEMAND DEDICATED/BUS HOLD
384      000100      BUSREL= 100    ;DEMAND DEDICATED/BUS RELEASE
385      000140      INTLV= 140     ;DEMAND INTERLEAVE
386      ;Interrupt Enable Field
387      000200      IEOP= 200     ;INTERRUPT ON EOP
388      000400      IMC= 400      ;INTERRUPT ON MATCH CONDITION
389      001000      ITC= 1000     ;INTERRUPT ON TRANSFER COMPLETE
390      ;Base to Current reload enable field
391      002000      REOP= 2000    ;RELOAD ON EOP
392      004000      RMC= 4000     ;RELOAD ON MATCH CONDITION
393      010000      RTC= 10000    ;RELOAD ON TRANSFER COMPLETE
394      ;Chain enable field
395      020000      CEOP= 20000   ;CHAIN ON EOP
396      040000      CMC= 40000    ;CHAIN ON MATCH CONDITION
397      100000      CTC= 100000   ;CHAIN ON TRANSFER COMPLETE
398      ;Match control field
399      000000      SNM= 0        ;STOP IF NO MATCH
400      000002      SUM= 2       ;STOP ON WORD MATCH
401      000003      SBM= 3       ;STOP ON BYTE MATCH
402      ;
403      000010      HM= 10        ;HARDWARE INTERRUPT MASK
404      000020      SWRQ= 20     ;SOFTWARE REQUEST
405      ;
406      ;DTC Status Register bit definitions
407      ;
408      000001      TC= 1         ;TRANSFER COMPLETE
409      000002      EOP= 2       ;END OF PROCESS (NXM)

```

D2

KXJ11-CA FUNCTIONAL TEST
MEMORY MANAGEMENT DEFINITIONS

```
410      004000      CA=      4000      ;CHAIN ABORT
411      010000      NAC=     10000     ;NO AUTO-LOAD OR C.HAIN
412      020000      IP=      20000     ;INTERRUPT PENDING
413      100000      CIE=     100000    ;CHANNEL INTERRUPT ENABLE
414
427      ;
428      177520      K2CSRA= 177520    ;THESE ARE EQUATES FOR THE KXJ11-CA CONTROL STATUS REGISTERS
429      177522      K2CSR8= 177522
430      177524      K2CSRC= 177524
431      177526      K2CSRE= 177526
432      177530      K2CSRD= 177530
433      177532      K2QIR=  177532
434      177534      K2CSRf= 177534
435      177536      K2CSRH= 177536
436      177540      K2CSRJ= 177540
437      ;
438      000001      APTENV= 1
439      177766      CPEREG= 177766    ;CPU ERROR REGISTER
440      000400      WLZ=      400      ; CALL TYPOS WITH-LEAD-ZEROS...
441      000000      NLZ=      0       ;...OR NULL-LEAD-ZEROS.
442      000401      SKP1=    BR+1     ; A COUPLE OF HANDY SKIPS.
443      000402      SKP2=    BR+2
444      000403      SKP3=    BR+3
445      000003      F11=      3
446      000004      T11=      4       ; \
447      000005      J11=      5       ; > PROCESSOR TYPE (MFPT) CODES.
448      000006      ACK=      6       ; /
449      000025      NACK=     25
450      001000      STBOT=    1000
```


E2

KXJ11-CA FUNCTIONAL TEST
TRAP CATCHER

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 16

548

.SBTTL TRAP CATCHER

000000

```

=0
;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
;*   PC=YYYYYY UNEXPECTED TRAP TO XXX
;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
;*WHERE XXX=LOCATION OF ILLEGAL TRAP
;*   YYYYYY=PC AT TIME OF TRAP
;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
;*   CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.

```

000000 000000
000002 000737

000004 051510
000006 000340
000174 000174

000174 000000
000176 000000

000200 000137 051510

```

$40CAT: HALT           ;;HALT
        BR            .-100      ;;BRANCH TO 177700 & TIME OUT (NOT ON
                                ;;11/05)
        .WORD        $$ARB       ;;VECTOR TO STARTING ADDRESS
        .WORD        340        ;;WITH PRIORITY LEVEL 7
        .=174
DISPREG: .WORD        0          ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD        0          ;;SOFTWARE SWITCH REGISTER
.SBTTL  STARTING ADDRES(ES)
        JMP          @$$ARB ;;GO TO START OF PROGRAM

```

```

550                                     .SBTTL REDEFINE THE VECTOR SPACE.
551                                     ;*****
552                                     .=0                                     ; BACK UP...
553 000000 000002 000000                .WORD .+2, 0                               ;...AND FIX ZERO...
554 000004 000006 000000                .WORD .+2, 0                               ;...AND FIX FOUR
555                                     .=24
556 000024 002542 000340                PWRV: .WORD DISMISS, PR7                 ; SET PWR (BINIT) VECTOR.
557                                     .=34
558 000034 043002 000340                .WORD $TRAP, PR7                          ; SET TRAP VECTOR TO POINT TO TRAP HANDLER
559                                     .=70
560 000070 002542 000200                NECV: .WORD DISMISS, PR4                 ; SET NEC7201 (SYNC/ASYN) VECTOR (T).
561                                     .=100
562 000100 002542 000300                BEVNT: .WORD DISMISS, PR6                 ; SET BEVNT (LTC) AND...
563 000104 002542 000300                PEVNT: .WORD DISMISS, PR6                 ;...PEVNT (Z8036 OR I8254 PIT).
564 000110                                     .BLKW 2                                     ; OPEN
565 000114 002542 000340                PRTYV: .WORD DISMISS, PR7                 ; SET PARITY ERROR...
566 000120 002542 000240                DPRV4: .WORD DISMISS, PR5                 ;...DUAL-PORT-RAM WORD 4 (T)...
567 000124 002542 000240                DPRV8: .WORD DISMISS, PR5                 ;...DUAL-PORT-RAM WORD 8 (T)...
568 000130 003042 000240                TBIACK: .WORD BITRAP, PR5                 ;...BIACKI (T)... ;FX8JC
569 000134 002542 000240                DPRV12: .WORD DISMISS, PR5                ;...AND DUAL-PORT-RAM WORD 12 (T).
570 000140                FALCON: .BLKW 2                       ; *** RESERVED FOR "FALCON-BREAK" ***
572                                     .=144
573 000144 056642 000200                QIRV: .WORD QREQ, FR4                     ; Q-REQUEST, ARBITER EXECUTES "RTI".
574 000150 056630 000200                QIRV1: .WORD QREQ1, PR4                   ; Q-REQUEST, ARBITER EXECUTES "RESET".
575 000154 056534 000200                QIRV2: .WORD QREQ2, PR4                   ; Q-REQUEST, ARBITER WRITES DPR
577                                     .=160
578 000160 002542 000200                PIOAV: .WORD DISMISS, PR4                 ; PIO PORT A...
579 000164 002542 000200                PIOBV: .WORD DISMISS, PR4                 ;...AND PORT B VECTORS.
580                                     .=174
581 000174 000000                                     .WORD 0
582 000176 140000                                     .WORD 140000                               ; OPEN (DISPREG/SWREG).
583 000200 000137 051510                RSTART: JMP $ARB                          ; DEFAULT START; DETERMINES OPERATING
584                                     ; ENVIRONMENT AND INVOKES REMOTE LOADER.
585 000204 000137 001622                DSTART: JMP START                          ; DIAGNOSTIC START/RESTART.
586                                     .=214
587 000214 002542 000200                DMAV:DMAV1: .WORD DISMISS, PR4             ; DTC channel on k2
588 000220 002542 000200                DMAV2: .WORD DISMISS, PR4                 ;DTC channel 2 on k2 ;k2hhe
589                                     .=240
590 000240                                     .PIRVEC
591 000240 003056 000340                PIRQV: .WORD UNEXPI, PR7                 ;Vector for unexpected PIRQ interrupts
592                                     .=250
593 000250 002542 000200                .WORD DISMISS, PR4                         ;VECTOR FOR UNEXPECTED MMU ERRORS
594 000254 052032 000200                NXTIOP: .WORD NEXT1, PR4                  ;VECTOR THE ARBITER TO LOOK FOR THE NEXT IOP
595                                     .=410
596                                     .LIST ME
597 000410                IOPVEX:
600 000410 055656 000340                .WORD IOPV02,PR7
        000414 055662 000340                .WORD IOPV03,PR7
        000420 055666 000340                .WORD IOPV04,PR7
        000424 055672 000340                .WORD IOPV05,PR7
        000430 055676 000340                .WORD IOPV06,PR7
        000434 055702 000340                .WORD IOPV07,PR7
        000440 055706 000340                .WORD IOPV08,PR7
        000444 055712 000340                .WORD IOPV09,PR7
        000450 055716 000340                .WORD IOPV10,PR7
        000454 055722 000340                .WORD IOPV11,PR7
        000460 055726 000340                .WORD IOPV12,PR7

```

G2

KXJ11-CA FUNCTIONAL TEST
REDEFINE THE VECTOR SPACE.

MACRO Y05.02 Thursday 03 Apr-86 14:11 Page 18

000464	055732	000340	.WORD	IOPV13,PR7	
000470	055736	000340	.WORD	IOPV14,PR7	
000474	055742	000340	.WORD	IOPV15,PR7	
601			.NLIST ME		
602	001000		. =1000		; RESET PC.

606

```

.EVEN
001166 000000 $MAIL: .WORD AMSTY :: APT MAILBOX
001170 000000 $FATAL: .WORD AFATAL :: MESSAGE TYPE CODE
001172 000000 $TESTN: .WORD ATESTN :: FATAL ERROR NUMBER
001174 000000 $PASS: .WORD APASS :: TEST NUMBER
001176 000000 $DEVCT: .WORD ADEVCT :: PASS COUNT
001200 000000 $UNIT: .WORD AUNIT :: DEVICE COUNT
001202 000000 $MSGAD: .WORD AMSGAD :: I/O UNIT NUMBER
001204 000000 $MSGLG: .WORD AMSGLG :: MESSAGE ADDRESS
001206 000 $ETABLE: .WORD AENV :: MESSAGE LENGTH
001206 000 $ENV: .BYTE AENVM :: APT ENVIRONMENT TABLE
001207 000 $ENVM: .BYTE AENVM :: ENVIRONMENT BYTE
001210 000000 $SWREG: .WORD ASWREG :: ENVIRONMENT MODE BITS
001212 000000 $USWR: .WORD AUSWR :: APT SWITCH REGISTER
001214 000000 $CPUOP: .WORD ACPUOP :: USER SWITCHES
; CPU TYPE, OPTIONS
; BITS 15-11=CPU TYPE
; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
; 11/70=06,PDQ=07,Q=10
; BIT 10=REAL TIME CLOCK
; BIT 9=FLOATING POINT PROCESSOR
; BIT 8=MEMORY MANAGEMENT
001216 000 $MAMS1: .BYTE AMAMS1 :: HIGH ADDRESS, M.S. BYTE
001217 000 $MTYP1: .BYTE AMTYP1 :: MEM. TYPE, BLK#1
; MEM. TYPE BYTE -- (HIGH BYTE)
; 900 NSEC CORE=001
; 300 NSEC BIPOLAR=002
; 500 NSEC MOS=003
001220 000000 $MADR1: .WORD AMADR1 :: HIGH ADDRESS, BLK#1
; MEM. LAST ADDR. = 3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
001222 000 $MAMS2: .BYTE AMAMS2 :: HIGH ADDRESS, M.S. BYTE
001223 000 $MTYP2: .BYTE AMTYP2 :: MEM. TYPE, BLK#2
001224 000000 $MADR2: .WORD AMADR2 :: MEM. LAST ADDRESS, BLK#2
001226 000 $MAMS3: .BYTE AMAMS3 :: HIGH ADDRESS, M.S. BYTE
001227 000 $MTYP3: .BYTE AMTYP3 :: MEM. TYPE, BLK#3
001230 000000 $MADR3: .WORD AMADR3 :: MEM. LAST ADDRESS, BLK#3
001232 000 $MAMS4: .BYTE AMAMS4 :: HIGH ADDRESS, M.S. BYTE
001233 000 $MTYP4: .BYTE AMTYP4 :: MEM. TYPE, BLK#4
001234 000000 $MADR4: .WORD AMADR4 :: MEM. LAST ADDRESS, BLK#4
001236 000000 $VECT1: .WORD AVECT1 :: INTERRUPT VECTOR#1, BUS PRIORITY#1
001240 000000 $VECT2: .WORD AVECT2 :: INTERRUPT VECTOR#2, BUS PRIORITY#2
001242 000000 $BASE: .WORD ABASE :: BASE ADDRESS OF EQUIPMENT UNDER TEST
001244 000000 $DEVN: .WORD ADEVN :: DEVICE MAP
001246 000000 $CDW1: .WORD ACDW1 :: CONTROLLER DESCRIPTION WORD#1
001250 000000 $CDW2: .WORD ACDW2 :: CONTROLLER DESCRIPTION WORD#2
001252 000000 $DDW0: .WORD ADDW0 :: DEVICE DESCRIPTOR WORD#0
001254 000000 $DDW1: .WORD ADDW1 :: DEVICE DESCRIPTOR WORD#1
001256 000000 $DDW2: .WORD ADDW2 :: DEVICE DESCRIPTOR WORD#2
001260 000000 $DDW3: .WORD ADDW3 :: DEVICE DESCRIPTOR WORD#3
001262 000000 $DDW4: .WORD ADDW4 :: DEVICE DESCRIPTOR WORD#4
001264 000000 $DDW5: .WORD ADDW5 :: DEVICE DESCRIPTOR WORD#5
001266 000000 $DDW6: .WORD ADDW6 :: DEVICE DESCRIPTOR WORD#6
001270 000000 $DDW7: .WORD ADDW7 :: DEVICE DESCRIPTOR WORD#7
001272 000000 $DDW8: .WORD ADDW8 :: DEVICE DESCRIPTOR WORD#8
001274 000000 $DDW9: .WORD ADDW9 :: DEVICE DESCRIPTOR WORD#9
001276 000000 $DDW10: .WORD ADDW10 :: DEVICE DESCRIPTOR WORD#10
001300 000000 $DDW11: .WORD ADDW11 :: DEVICE DESCRIPTOR WORD#11

```

I2

KXJ11-CA FUNCTIONAL TEST
APT MAILBOX-ETABLE

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 20

001302 000000
001304 000000
001306 000000
001310 000000
001312

\$DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
\$DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
\$DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
\$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
\$ETEND:
.MEXIT

```

610 .SBTTL PROGRAM CONTROL PARAMETERS
611 ;*****
612 ; THE FOLLOWING PARAMETERS GOVERN THE BEHAVIOR OF THE PROGRAM AND
613 ; ARE CONDITIONALLY CONFIGURED IAW PROCESSOR TYPE AT ASSEMBLY TIME.
614 ;
615 ;
616 ;
617 ;
618 001312 000005 $CPTYP: J11
619 ;
620 001314 177530 $CSR: 177530 ;T ; PRIMARY CONTROL/STATUS REGISTER.
621 001316 177532 $QIR: 177532 ;T ; Q-BUS INT (VECTOR) REGISTER.
622 001320 175000 $DPR: 175000 ;T ; DUAL-PORT-RAM LOCAL ADDRESS.
623 001322 160000 $DPRQ: 160000 ;T ; DUAL-PORT-RAM Q-BUS BASE ADDRESS.
624 001324 174400 $DMA: 174400 ;T ; AMZ8016 (OR QMIC) DMA CHIP BASE.
625 001326 175700 $SL2: 175700 ;T ; SECOND SERIAL LINE ADDRESS.
626 001330 177000 $PIO: 177000 ;T ; Z8036 PIO/PIT BASE ADDRESS.
627 001332 177520 $CLK: 177520 ;T ; LINE CLOCK (BEVNT) CSR.
628 ;
629 000010 $PNUM= <.-$CSR>/2 ;TFJ; NUMBER OF VARIABLE ITEMS.
630 ;
631 001334 177520 $CSRA: 177520 ;T ; SECONDARY CONTROL/STATUS REGISTERS...
632 001336 177522 $CSRB:$CID: 177522 ;TFJ; ...INCLUDES CID ON F/J...
633 001340 177524 $CSRC:$LEDS:$TCID: 177524 ;TFJ; ...INCLUDES LEDES (AND CID ON T).
634 ;
635 001342 000001 $QDLY: 1 ;TFJ; Q-IACK WAIT LOOP CONTROL...
636 ; ...STAND-ALONE (1) == 163. MSEC...
637 ; ...APT-AUTO (62.) == 10.1 SEC
638 001344 000001 RFLAG: 1 ; FLAG FOR TRAP THROUGH 24. 0=EXPECTED ;FX8JC
639 ; 1=UNEXPECTED. ;FX8JC
640 001346 000001 TPROTR: 1 ; FLAG FOR TRAP CAUSED BY WRITE TO TPROO
641 001350 000001 HLTFLG: 1 ; SAY HALT TRAP UNEXPECTED
642 001352 000001 BFLAG: 1 ; FLAG FOR TRAP THROUGH 130. 0=EXPECTED ;FX8JC
643 ; 1=UNEXPECTED. ;FX8JC
644 001354 000000 SLOC00: .WORD 0
645 001356 000000 SLOC01: .WORD 0
651 ;
652 ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
653 001360 000000 EXPDAT: .WORD 0 ;STORES EXPECTED (GOOD) DATA FOR COMPARISONS
654 001362 000000 RECDAT: .WORD 0 ;STORES RECIEVED DATA TO BE VERIFIED
655 001364 000000 COUNT: .WORD 0 ;ERROR INDICATOR FOR FLOATING POINT TESTS
656 001366 000000 FLAG: .WORD 0 ;USED TO STORE "FLAG" CONDITIONS
657 001370 000000 ERRCNT: .WORD 0 ;STORAGE FOR ERROR COUNT
658 ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
659 001372 000000 DCOUNT: .WORD 0
660 001374 000000 ALLCTR: .WORD 0
661 001376 000000 LOOPIN: .WORD 0
662 001400 000000 SAVSP1: .WORD 0 ;STORAGE FOR UNEXPECTED TRAP DATA
663 001402 000000 SAVSP2: .WORD 0 ; " " " "
673 001404 000000 C1FLAG: .WORD 0 ;DTC channel 1 event indicator ;k2hhe
674 001406 000000 C2FLAG: .WORD 0 ;DTC channel 2 event indicator ;k2hhe
675 001410 000000 SOURCE: .WORD 0 ;Scratch location ;k2hhe
676 001412 .BLKW 20. ;Scratch locations ;k2hhe
677 ;

```

K2

```

679          .SBTTL GLOBAL SUBROUTINES SECTION
680          ;**
681          ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
682          ; THAT ARE USED IN MORE THAN ONE TEST.
683          ;--
684          ;
685          ;MMU GLOBAL SUBROUTINES
686          ;
687          ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
688          ;
689          MMU:  MOV     R0,-(SP)          ;SAVE CONTENTS OF REGISTERS
690              MOV     R1,-(SP)          ;
691              MOV     R2,-(SP)          ;
692              MOV     #177600,R0        ;
693              JSR     PC,PDR            ;INIT I AND D USER PDR'S
694              JSR     PC,PAR            ;INIT I USER PAR'S
695              JSR     PC,PAR            ;INIT D USER PAR'S
696              MOV     #172200,R0        ;
697              JSR     PC,PDR            ;INIT I AND D SUP PDR'S
698              JSR     PC,PAR            ;INIT I SUP PAR'S
699              JSR     PC,PAR            ;INIT D SUP PAR'S
700              JSR     PC,PDR            ;INIT I AND D KER PDR'S
701              JSR     PC,PAR            ;INIT I KER PAR'S
702              JSR     PC,PAR            ;INIT D KER PAR'S
703          001544 012737 000027 172516  MOV     #27,@#172516  ;INIT MMR3: ENABLE 22 BIT MAPPING
704              ; AND KERNEL, USER AND SUPR DATA SPACE
705          001552 012602              MOV     (SP)+,R2  ;RESTORE REGISTERS
706          001554 012601              MOV     (SP)+,R1  ;
707          001556 012600              MOV     (SP)+,R0  ;
708          001560 000207              RTS      PC          ;RETURN
709          ;
710          ;ROUTINE TO INITIALIZE PDR'S
711          ;
712          PDR:  MOV     #16.,R2          ;INIT CNTR
713          1$:  MOV     #77406,(R0)+      ;INIT PDR
714              SOB     R2,1$            ;
715              RTS      PC              ;RETURN
716          ;
717          ;ROUTINE TO INITIALIZE PAR'S
718          ;
719          PAR:  CLR     R1                ;SETUP TO INIT PAR
720          1$:  MOV     R1,(R0)+          ;INIT PAR
721              ADD     #200,R1           ;GET READY FOR NEXT PAR
722              CMP     #1600,R1         ;REACHED A PAR?
723              BNE     1$               ;BRANCH IF NOT
724              MOV     #177600,(R0)+    ;INIT PAR?
725              RTS      PC              ;RETURN
726          ;
727          ;*****
728          ;*****
729          ;*****
730          ;*****
731          ;*****
732          ;*****
733          ;*****
734          ;*****
735          ;*****
736          ;*****
737          ;*****
738          ;*****
739          ;*****
740          ;*****
741          ;*****
742          ;*****
743          ;*****
744          ;*****
745          ;*****
746          ;*****
747          ;*****
748          ;*****
749          ;*****
750          ;*****
751          ;*****
752          ;*****
753          ;*****
754          ;*****

```

L2

```

756 .SBTTL PROGRAM START/RESTART
757 ;
758 ; PROGRAM START/RESTART HERE.
759 ;
760 001622 000005 START:RESTART:
761 001622 000005 RESET
763 .SBTTL INITIALIZE THE COMMON TAGS
; CLEAR THE COMMON TAGS ($CMTAG) AREA
001624 012706 001100 MOV $CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
001630 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
001632 022706 001140 CMP #SWR,R6 ;:DONE?
001636 001374 BNE -6 ;:LOOP BACK IF NO
001640 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
;:INITIALIZE A FEW VECTORS
001644 012737 040206 000020 MOV #SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
001652 012737 000340 000022 MOV #340,@#IOTVEC+2 ;:LEVEL 7
001660 012737 044314 000030 MOV #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
001666 012737 000340 000032 MOV #340,@#EMTVEC+2 ;:LEVEL 7
001674 012737 043002 000034 MOV #TRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
001702 012737 000340 000036 MOV #340,@#TRAPVEC+2 ;:LEVEL 7
001710 013737 037662 037654 MOV $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
001716 005037 001160 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
001722 112737 000001 001. 5 MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
001730 012737 001730 001.1) MOV #,$LPERR ;:SETUP THE ERROR LOOP ADDRESS
;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001736 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
001742 012737 001776 000004 MOV #30000$,@#ERRVEC ;:SET UP ERROR VECTOR
001750 012737 177570 001140 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
001756 012737 177570 001142 MOV #DISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
001764 022777 177777 177146 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
001772 001012 BNE 30002$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
;:AND THE HARDWARE SWR IS NOT = -1
001774 000403 BR 30001$ ;:BRANCH IF NO TIMEOUT
001776 012716 002004 30000$: MOV #30001$, (SP) ;:SET UP FOR TRAP RETURN
002002 000002 RTI
002004 012737 000176 001140 30001$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
002012 012737 000174 001142 MOV #DISPREG,DISPLAY
002020 012637 000004 30002$: MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
002024 005037 001174 CLR #PASS ;:CLEAR PASS COUNT
002030 132737 000200 001207 BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
002036 001403 BEQ 30003$ ;:YES,USE NON-APT SWITCH
002040 012737 001210 001140 MOV #SWREG,SWR ;:NO,USE APT SWITCH REGISTER
002046 30003$:
764 .SBTTL CONFIGURE P-TABLES IAW CPU TYPE
765 ;
766 ; INIT A FEW MORE THINGS.
767 ;
768 002046 INIT:
769 002046 106427 000340 MTPS #PR7 ; RAISE CPU.
770 002052 012737 000004 000002 MOV #IOT,@#2 ;:LOAD VECTOR AREA
771 002060 013737 000002 000006 MOV @#2,@#6 ;
772 002066 005037 001102 CLR #TSTNM ;:RESET THE TEST NUMBER
773 ;LETS CALCULATE THE VECTOR THAT THIS PARTICULAR IOP IS GOING TO
774 ;INTERRUPT THROUGH.
775 002072 013700 177524 MOV @#K2CSRC,R0 ;GET ID SWITCH SETTING

```


M2

KXJ11-CA FUNCTIONAL TEST
CONFIGURE P-TABLES IAW CPU TYPE

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 24

```

776 002076 010002          MOV      R0,R2          ;
777 002100 032700 000340  BIT      #340,R0       ;IF WE'RE NOT IN STAND ALONE CONFIG.
778 002104 001007          BNE     3$             ;...SKIP NEXT FEW INSTRUCTIONS
779 002106 042702 177417  BIC     #+C3E0,R2     ;MASK OFF ID SWITCH FIELD
780 002112 072227 177774  ASH     #-4,R2        ;SHIFT IT OVER
781 002116 010237 050742  MOV     R2,@#IOP.ID   ;LOAD IOP.ID
782 002122 000410          BR      2$             ;
783
784 002124 042700 177417  3$:    BIC     #+C360,R0   ;MASK OFF THE ID SWITCH FIELD
785 002130 072027 177776  ASH     #-2,R0        ;SHIFT IT RIGHT 2 POSITIONS
786 002134 062700 000400  ADD     #400,R0       ;ADD IN THE BASE INTERRUPT VECTOR
787 002140 010037 036434  MOV     R0,INTVEC     ;.. AND THERE WE HAVE IT.
788
789 002144 013737 050742 001200 2$:    MOV     @#IOP.ID,$UNIT ; UPDATE MAIL BOX
790 002152 012737 000256 000254  MOV     #256,@#254    ;FIX VECTOR IN SBC
791 002160 012737 000004 000256  MOV     #IOT,@#256   ;...WITH STANDARD STUFF
792 002166 012702 000016  MOV     #14.,R2      ;NUMBER OF VECTORS TO FIX
793 002172 012700 000410  MOV     #410,R0      ;STARTING AT ADDRESS 410
794 002176 010001          1$:    MOV     R0,R1         ;
795 002200 062701 000002  ADD     #2,R1        ;
796 002204 010120          MOV     R1,(R0)+     ;PATCH VECTOR AREA
797 002206 012720 000004  MOV     #IOT,(R0)+  ;
798 002212 077207          SOB     R2,1$        ;LOOP CONTROL
799 002214 004737 002250  CALL    KXINIT       ; XCT BOARD-UNIQUE INITIALIZATION.
800
801 002220 012737 001622 000202 4$:    MOV     #START,@#202 ; ENABLE DIAG RESTART ON "200G".
802 002226 012700 051034  MOV     #ELOG,R0     ;
803 002232 012701 000225  MOV     #ELOGN,R1    ;
804 002236 005020          5$:    CLR     (R0)+        ; CLEAR THE ERROR LOG (KILLS REMOTE).
805 002240 077102          SOB     R1,5$        ;
806 002242 104414          I$SWR              ; CHECK MODE (APT VS STANDALONE)...
807                                     ;...AND GET INITIAL $SWR AND $QDLY.
808 002244 000137 003064  7$:    JMP     AGAIN       ; SKIP OVER THE INITIALIZERS.
809                                     ;
810                                     ; BOARD-UNIQUE INITIALIZATION.
811                                     ;
812                                     ; SOME OF THE CHIPS ON THESE BOARDS MAY DO FUNNY THINGS
813                                     ; (OR PREVENT US FROM DOING THE RIGHT THINGS) ON POWER-UP.
814                                     ; THESE ROUTINES WILL INIT, RESET, DISABLE ... OR WHATEVER.
815                                     ; BUS-ERRORS ARE IGNORED (TRAP4X = -1).
816
817 002250 012737 177777 002426 KXINIT: MOV     #-1,TRAP4X ; DISMISS BUS ERRORS.
818 002256 004737 001462          JSR     PC,MMU       ;INIT K2 MMU REGISTERS
819 002262 013700 001324          MOV     #DMA,R0     ;
820 002266 005060 000054          CLR     54(R0)      ; INIT (RESET) AMZ8016 DMA CHIP.
821 002272 013700 001326          MOV     #SL2,R0     ;
822 002276 112760 000030 000004  MOVB   #<3*BIT3>,4(R0) ; INIT (RESET) NEC7201 SYNC/ASYNC CHIP.
823 002304 112760 000030 000014  MOVB   #<3*BIT3>,14(R0); BOTH CHANNELS.
824 002312 012737 002614 000024  MOV     #RSTRAP,PWRV ; SET UP RESET TRAP VECTOR. ;FX8JC
825 002320 012737 000340 000026  MOV     #340,PWRV+2 ; PRIORITY 7. ;FX8JC
826 002326 005037 177766          CLR     @#177766   ;Clear the CPU error register
828 002332 012737 002406 000240  MOV     #PIRTRP,@#PIRVEC ;ACTION ON UNEXPECTED PIRQ ERRORS
829 002340 112777 000001 176762  MOVB   #1,@#PIO     ; INIT (RESET) Z8036 PIO CHIP.
830 002346 012737 000001 001352  MOV     #1,@#BFLAG  ; SAY TRAP THRU 130 NOT EXPECTED. ;FX8JC
831 002354 012737 000001 001344  MOV     #1,@#RFLAG  ; SAY TRAP THRU 24 NOT EXPECTED. ;FX8JC
832 002362 012777 002000 176724  MOV     #2000,@#CSR ; ENABLE TRAP ON Q-BUS RESET. ;FX8JC
833 002370 005037 177520          CLR     @#K2CSRA   ; \

```

N2

KXJ11-CA FUNCTIONAL TEST
CONFIGURE P-TABLES IAW CPU TYPE

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 25

834 0023.4 005037 177524
835 002400 005037 177540
836 002404 000207

CLR @#K2CSRC
CLR @#K2CSRJ
RETURN

: > EVERYTHING OFF !!
: /

```

838 .SBTTL BUS ERROR AND ILLEGAL TRAP HANDLERS.
839 ;
840 ; COME HERE FROM $SCOPE ON ANY UNDEFINED TRAP (IOT)
841 ; WITH VECTOR ADDRESS ON TOP OF THE STACK.
842 ; BUS-ERROR (VECTOR 4) IS A SPECIAL CASE WHERE "TRAP4X"
843 ; DEFINES THE ACTION TAKEN AS FOLLOWS:
844 ; 0 = ERROR, REPORT AND ABORT TEST.
845 ; -1 = DISMISS AND RETURN "V" = 1.
846 ; ADR - DISMISS AND RETURN TO USER AT ADDRESS SPECIFIED.
847 ;
848
849 002406 005037 177772 PIRTRP: CLR @#177772 ;CLEAR THE PIRQ REG
850 002412 012746 000240 MOV #240,-(SP) ;
851 002416 022627 000004 UNXTRP: CMP (SP)+,#4 ; WAS IT A BUS-ERROR ??
852 002422 001015 BNE TRAPE ; BR IF NOT.
853 002424 005727 TST (PC)+ ; WAS IT ANTICIPATED ??
854 002426 000000 TRAP4X: .WORD 0
855 002430 001412 BEQ TRAPE ; ERROR AND ABORT IF NOT.
856 002432 023727 002426 177777 CMP TRAP4X,#-1 ; ALTERNATE RETURN PC ??
857 002440 001402 BEQ 1$ ; SKIP IF NOT...
858 002442 013716 002426 MOV TRAP4X,(SP) ;...ELSE, TAKE NEW PC.
859 002446 052766 000002 1$: BIS #2,2(SP) ; SET "V" IN CALLERS PSW...
860 002454 000432 BR DISMISS ;...AND RETURN.
861
862 002456 016637 177776 001122 TRAPE: MOV -2(SP),#BDADR ; GET BAD TRAP VECTOR...
863 002464 011637 001120 MOV (SP),#GDADR ; ...AND PC AT TIME OF HAPPENING.
864 002470 012737 046217 044746 MOV #UNXT,EMADR ;:\
002476 012737 045442 044752 MOV #EF7,ESADR ;: >> ERROR 1, UNEXPECTED TRAP THRU VECTOR NNN...
002504 104001 ERROR+1 ;://
865 ;...AND FALL THRU
866 002506 113700 001102 ABORT: MOVB $TSTNM,RO ; GET CURRENT TEST NUMBER.
867 002512 001564 BEQ AGAIN ; BR IF NONE.
868 002514 006300 ASL RO ; SHIFT UP TO A WORD OFFSET.
869 002516 016000 040446 MOV $SW08TBL(RO),RO ; GET NEXT TEST DISPATCH ADDRESS...
870 002522 162700 000002 SUB #2,RO ; ...AND BACK IT UP 1 WORD.
871 002526 020027 037632 CMP RO,#$EOP ; VALID ADDRESS ??
872 002532 103402 BLO 1$ ; USE IT IF SO.
873 002534 012700 037632 MOV #EOP,RO ; OTHERWISE, TAKE END-PASS.
874 002540 010016 1$: MOV RO,(SP)
875 002542 000002 DISMISS: RTI ; NEXT TEST OR END-PASS.
876
877 ;+ ;FX8JC
878 ; ALWAYS COME HERE ON TRAP THROUGH 24. ;FX8JC
879 ;- ;FX8JC
880 002544 NMITRP:
881 002544 032737 000200 177530 BIT #BIT07,@#K2CSR ;DID WE TRAP DUE TO WRITE TO TPRO?
882 002552 001420 BEQ RSTRAP ;NO
883 002554 005737 001346 TST TPROTR ;WAS IT EXPECTED (TPROTR = ZERO)
884 002560 001407 BEQ 1$
885 002562 012737 002762 044746 MOV #UNXTPR,EMADR ;:\
002570 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 2, UNEXPECTED TRAP CAUSED BY WRITE TO TPRO
002576 104002 ERROR+2 ;://
886 002600 005237 001346 1$: INC TPROTR ;SAY TRAP OCCURED
887 002604 042737 000200 177530 BIC #BIT07,@#K2CSR ;CLEAR FLAG
888 002612 000447 BR NMIEND
889 002614 RSTRAP:
890 002614 032737 004000 177530 BIT #BIT11,@#K2CSR ;WAS TRAP CAUSED BY BINIT?
    
```

C3

```

891 002622 001420
892 002624 005737 001344
893 002630 001407
894 002632 012737 002734 044746
      002640 012737 045206 044752
      002646 104003
895 002650 005237 001344 1$:
896 002654 042737 004000 177530
897 002662 000423
898 002664
899 002664 032737 001000 177530
900 002672 001417
901 002674 005737 001350
902 002700 001407
903 002702 012737 003015 044746
      002710 012737 045206 044752
      002716 104004
904 002720 005237 001350 1$:
905 002724 042737 001000 177530
906 002732 000002
907 002734 125 116 105
908 002762 125 116 105
909 003015 125 116 105
910
911
912
913
914 003042 005737 001352
915 003046 001002
916 003050 005237 001352
917 003054 000002
918
919
920
921 003056 005037 177772
922 003062 000002
923
924
925
926 003064 000005
927 003066 013746 050742
928 003072 104401 003102
929 003076 104405
930 003100 000425
931 003102 200 103 132 1$:
932
933 003154
934 003154 000230
935 003156 000400
936
937
938
003160 000004
003162 012706 001100
003166 012737 177777 002426

```

```

      BEQ      HALTRP
      TST     @#RFLAG          ; WAS TRAP EXPECTED?
      BEQ     1$              ; NO, CALL ERROR
      MOV     @UNXRES,EMADR    ;:\
      MOV     @NOSIG,ESADR    ;>> ERROR 3, UNEXPECTED RESET TRAP
      ERROR+3                  ;://
      INC     @#RFLAG          ; SAY TRAP OCCURRED.
      BIC     @#4000,@#177530 ; ... AND CLEAR FLAG.
      BR     NMIEND           ; AND EXIT.
HALTRP:
      BIT     @BIT09,@#K2CSR0 ; WAS TRAP CAUSED BY BHALT?
      BEQ     NMIEND           ; NO
      TST     @#HLTFLG        ; WAS TRAP EXPECTED?
      BEQ     1$
      MOV     @UNXHLT,EMADR    ;:\
      MOV     @NOSIG,ESADR    ;>> ERROR 4, UNEXPECTED BUS HALT
      ERROR+4                  ;://
      INC     @#HLTFLG        ; SAY HALT TRAP OCCURRED
      BIC     @BIT09,@#K2CSR0 ; CLEAR HALT FLAG IN CSR
      NMIEND: RTI              ; RETURN.
UNXRES: .ASCIZ /UNEXPECTED RESET TRAP/
UNXTPR: .ASCIZ /UNEXPECTED TPRO WRITE TRAP/
UNXHLT: .ASCIZ /UNEXPECTED HALT TRAP/
      .EVEN
      ;*
      ; ALWAYS COME HERE ON TRAP THROUGH 130.
      ;-
BITRAP: TST     @#BFLAG          ; WAS TRAP EXPECTED?
      BNE     1$              ; NO, JUST DISMISS.
      INC     @#BFLAG          ; SAY TRAP OCCURRED.
      RTI     1$              ; AND RETURN.
      ;
      ; COME HERE ON UNEXPECTED PIRQ INTERRUPT IN ARBITER
UNEXPI: CLR     @#177772        ;Clear the PIRQ
      RTI
      ;
      ;COME HERE ON END OF PASS
      ;
AGAIN:  RESET    ; FRESH START
      MOV     @#IOP.ID, -(SP) ; PUSH IOP ID NUMBER ONTO STACK
      TYPE    .1$
      TYPOS   ; TYPE ID NUMBER
      BR     2$
      .ASCIZ  <CRLF>/CZKXA - FUNCTIONAL TEST OF KXJ11-CA ID# /
      .EVEN
      2$:
      SPL     0                ; SET CPU LEVEL 0...
      BR     TST1              ; ...AND FALL THRU.
      ;
      ;J11K2.MAC
      ;*****
      ;*TEST 1 DCJ11 CPU
      ;*****
TST1:  SCOPE
      MOV     @STACK,SP        ; SET A CLEAN STACK.
      MOV     #-1,TRAP4X      ; DISMISS BUS-ERRORS.

```

D3

KXJ11 CA FUNCTIONAL TEST
T1 DCJ11 CPU

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 28

```

003174 005037 044742          CLR      EMPRE          ;; CLEAR ERROR PREFIX.
003200 000402                BR       30005$         ;; SKIP NEXT.
003202 000137 003404      30004$: JMP      TST2          ;; BYPASS.
003206 104415 003214      30005$: T$NAME, .+4     ;; PRINT TEST NUMBER AND NAME...
003212 000406                BR       30006$         ;; ...AND SKIP OVER THE ASCII.
003214      040      040      104      .ASCIZ  \ DCJ11 CPU\
                                .EVEN

003230
939 003230      30006$:
940                J11REG:
941                ;This test does a rudimentary test of the DCJ11 chip. Let's face it
942                ;folks; if we got through the power up stuff then the ;DCJ11 is probably
943                ;functional.
944                ;The test accesses the on chip registers. Many of the registers will have
945                ;been accessed during the initialization micro-routine, and the power
946                ;up routine.
947 003230 012737 003556 044746      MOV      #CPUER,EMADR    ;; SET ERROR MESSAGE...
003236 012737 045206 044752      MOV      #NOSIG,ESADR   ;...AND SIGNATURE POINTERS...
003244 000240                NOP                      ;...ERROR WILL BE CALLED LATER.
948 003246 000007      MFPT                      ;What type of processor is this?
949 003250 022700 000005      CMP      #5,R0          ;5= DCJ11 micro-processor
950 003254 001401      BEQ     100$           ;Branch if it's a DCJ11
951
                                ;;\
                                ;; >> ERROR 5,
                                ;;//

003256 104005                ERROR+5
952
953                ; wrong environment or we have a
954                ; major problem.
955 003260 012737 177777 002426 100$: MOV      #-1,TRAP4X    ;Anticipate NXM
956 003266 005037 177766      CLR      @#177766      ;Write CPU error register with zeros
957 003272 102403      BVS     101$           ;NXM sets the V bit
958 003274 005737 177766      TST     @#177766      ;Access CPU error register. (internal
959                ; register access BS<1:0> = 1,1)
960 003300 102001      BVC     1$             ;
961                ;
962                ;;\
963                ;; >> ERROR 6, TIME OUT ON ERROR REGISTER ACCESS
964                ;;//

003302 104006                ERROR+6
965                ;Access the major internal registers of the DCJ11
966 003304 012737 003350 002426 1$: MOV      #20$,TRAP4X    ;Action on I/O timeout
967 003312 005737 177776      TST     @#177776      ;Access PSW
968 003316 005737 177772      TST     @#177772      ;Access PIRQ
969 003322 005737 177752      TST     @#177752      ;Access Cache Hit/Miss register
970 003326 005737 177572      TST     @#177572      ;Access Memory Mngmnt Reg 0
971 003332 005737 177574      TST     @#177574      ;Access Memory Mngmnt Reg 1
972 003336 005737 177576      TST     @#177576      ;Access Memory Mngmnt Reg 2
973 003342 005737 172516      TST     @#172516      ;Access Memory Mngmnt Reg 3
974 003346 000403      BR       2$             ;
975                ;
976                ;;\
977                ;; >> ERROR 7,
978                ;;//

003350 104007                ERROR+7
979 003352 005037 177766      CLR     @#177766
980 003356 005037 002426      CLR     TRAP4X
981 003362 005737 177772      TST     @#177772
982                ;Access PIRQ. It should have been
983                ; cleared by the initialization microroutines.
984 003366 001401      BEQ     3$

```

E3

```

978                                     ;;\
                                     ;; >> ERROR 10,
                                     ;;//
003370 104010
979 003372 005737 177752          3$:  TST  ERROR+10 @#177752
980 003376 001401                BEQ  4$
981
982                                     ;;\
                                     ;; >> ERROR 11,
                                     ;;//
003400 104011
983 003402
003402 000400          4$:  BR  TST2
984                                     ;;
;*****
;*TEST 2 LOCAL K2 REGISTERS
;*****
TST2:  SCOPE
003404 000004
003406 012706 001100          MOV  #STACK,SP          ;; SET A CLEAN STACK.
003412 012737 177777 002426    MOV  #-1,TRAP4X        ;; DISMISS BUS ERRORS.
003420 005037 044742          CLR  EMPRF           ;; CLEAR ERROR PREFIX.
003424 000402          BR  30$          ;; SKIP NEXT.
003426 000137 005652          30007$: JMP  TST          ;; BYPASS.
003432 104415 003440          30008$: T$NAME, .+4      ;; PRINT TEST NUMBER AND NAME...
003436 000413          BR  30009$      ;; ...AND SKIP OVER THE ASCII.
003440 040 040 114          .ASCIZ \ LOCAL K2 REGISTERS\
                                     .EVEN
985 003466          30009$:
;*****
;*TEST 2.1 SYS REGISTERS
;*****
T2.1::
;
;This test will access the system registers. This causes a DCJ11 bus cycle
;with signals BS<1:0> = 0,1
;
990 003466 012737 003571 044746    MOV  #SYSER,EMADR    ;; SET ERROR MESSAGE...
003474 012737 045206 044752    MOV  #NOSIG,ESADR   ;; ...AND SIGNATURE POINTERS...
003502 000240          NOP          ;; ...ERROR WILL BE CALLED LATER.
991 003504 012737 003524 002426    MOV  #5$,TRAP4X     ;Anticipate NXM
992 003512 005737 177746          TST  @#177746        ;Access cache control register
993 003516 005737 177750          TST  @#177750        ;Access Maintenance Register
994 003522 000401
995 003524          5$:
                                     ;;\
                                     ;; >> ERROR 12, SYSTEM REGISTER ERROR
                                     ;;//
003524 104012
996 003526 005037 002426          7$:  CLR  TRAP4X
997 003532 005737 177746          TST  @#177746
998
999
1000 003536 001401          BEQ  10$
1001
                                     ;;\
                                     ;; >> ERROR 13, DCJ11 CACHE CNTRL REG NOT CLEARED
                                     ;;//
003540 104013
1002
1003 003542 132737 000060 177750 10$: BITB #1800110000,@#177750 ;Cache control register isn't clear.
1004
1005 003550 001001          BNE  11$          ;Is this a K2 module? (system reg
; access BS<1:0> = 0,1)
1006                                     ; Check ID code in maint. register
                                     ;;\

```

F3

```

                                ;; >> ERROR 14, MAINT REG ID NOT CORRECT
                                ;;//
                                ERROR+14
1007 003552 104014                11$: BR      K2CSRS
1008 003554 000421                CPUER: .ASCIZ <CRLF>/CPU ERROR/
1009 003556 200      103      120 SYSER: .ASCIZ <CRLF>/SYSTEM REGISTER ERROR/
1010 003571 200      123      131 .EVEN
1011                                ;*****
                                ;*TEST 2.2 CONTROL/STATUS REGISTERS
                                ;*****
                                T2.2::
1012 003620                                ;
1013                                ;Test the K2 control and status registers
1014                                ;
1015 177532                                QIR=177532
1016                                ;
1017 003620 012737 004715 044742 K2CSRS: MOV    #CSRADR,EMPRE ;ERROR MESSAGE PREFIX
1018 003626 012737 004662 044746   MOV    #CSPER,EMADR  ;; SET ERROR MESSAGE...
                                003634 012737 045206 044752   MOV    #NOSIG,ESADR  ;...AND SIGNATURE POINTERS...
                                003642 000240                NOP    ;...ERROR WILL BE CALLED LATER.
1019 003644 012737 177777 002426   MOV    #-1,TRAP4X   ;ANTICIPATE NXM
1020 003652 112737 000101 004676   MOV    #'A,@#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1021 003660 012704 177520                MOV    #K2CSRA,R4   ;START OF K2CSRS
1022 003664 011403                MOV    (R4),R3      ;READ K2CSRA
1023 003666 102414                BVS    5$           ;NXM IF V BIT SET
1024 003670 052703 177400                BIS    #177400,R3   ;TRY TO WRITE UPPER BYTE
1025                                ; SHOULD'N'T BE ABLE TO.
1026 003674 010314                MOV    R3,(R4)      ;WRITE K2CSRA
1027 003676 102410                BVS    5$           ;
1028 003700 042703 177400                BIC    #177400,R3   ;CLEAR UPPER BYTE
1029 003704 020314                CMP    R3,(R4)      ;UPPER BYTE IN R4 SHOULD BE ZERO
1030 003706 001004                BNE    5$           ;
1031 003710 110324                MOV    R3,(R4)+     ;TEST LOW BYTE OF K2CSRA
1032 003712 102402                BVS    5$           ;
1033 003714 105014                CLRB  (R4)          ;WRITE UPPER BYTE
1034 003716 102001                BVC    1$           ;V BIT INDICATES THAT TIMEOUT OCCURRED
1035 003720                                5$:
                                ;;\
                                ;; >> ERROR 15, ERROR IN K2CSRA
                                ;;//
                                ERROR+15
1036 003720 104015                1$:
1037 003722 112737 000102 004676   MOV    #'B,@#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1038 003730 012704 177522                MOV    #K2CSRB,R4   ;POINT TO K2CSRB
1039 003734 011403                MOV    (R4),R3      ;READ K2CSRB
1040 003736 102406                BVS    6$           ;NXM IF V BIT IS SET
1041 003740 010314                MOV    R3,(R4)      ;WRITE K2CSRB
1042 003742 102404                BVS    6$           ;
1043 003744 105724                TSTB  (R4)+         ;TRY BYTE ACCESS TO LOW BYTE
1044 003746 102402                BVS    6$           ;NXM IF V BIT SET
1045 003750 105724                TSTB  (R4)+         ;TRY BYTE ACCESS TO HIGH BYTE
1046 003752 102001                BVC    2$           ;SHOULD NOT TIME OUT
1047 003754                                6$:
                                ;;\
                                ;; >> ERROR 16, ERROR IN K2CSR8
                                ;;//
                                ERROR+16
1048 003756 104016                2$:
1049 003764 112737 000103 004676   MOV    #'C,@#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1050 003770 012704 177524                MOV    #K2CSRC,R4   ;POINT TO K2CSRC
                                011403                MOV    (R4),R3      ;TEST K2CSRC

```

```

1051 003772 102410      BVS      7$           :NXM IF V BIT SET
1052 003774 010314      MOV      R3,(R4)     :WRITE K2CSRC
1053 003776 102406      BVS      7$           :
1054 004000 105714      TSTB    (R4)         :READ LOW BYTE
1055 004002 102404      BVS      7$           :
1056 004004 110324      MOVB    R3,(R4)+    :WRITE LOW BYTE
1057 004006 102402      BVS      7$           :
1058 004010 105714      TSTB    (R4)         :READ HIGH BYTE
1059 004012 102001      BVC      3$           :SHOULD NOT TIME OUT
1060 004014              7$:
                                ;;\
                                ;; >> ERROR 17, ERROR IN K2CSRC
                                ;;//

004014 104017      ERROR+17

1061
1062 004016 112737 000104 004676 3$:  MOVB    #'D,@#CSRER+<12.> :ASCII FOR ERROR MESSAGE
1063 004024 012704 177530      MOV     #K2CSRJ,R4      :POINT TO K2CSRJ
1064 004030 011403      MOV     (R4),R3         :READ K2CSRJ
1065 004032 102406      BVS    10$             :NXM IF V BIT IS SET
1066 004034 010314      MOV     R3,(R4)        :WRITE K2CSRJ
1067 004036 102404      BVS    10$             :
1068 004040 105724      TSTB   (R4)+          :LOW BYTE READ
1069 004042 102402      BVS    10$             :
1070 004044 105724      TSTB   (R4)+          :HIGH BYTE READ
1071 004046 102001      BVC    4$             :NXM IF V BIT IS SET
1072 004050              10$:
                                ;;\
                                ;; >> ERROR 20, ERROR IN K2CSRJ
                                ;;//

004050 104020      ERROR+20

1073
1074 004052 112737 000112 004676 4$:  MOVB    #'J,@#CSRER+<12.> :ASCII FOR ERROR MESSAGE
1075 004060 012704 177540      MOV     #K2CSRJ,R4      :POINT TO NEXT CSR
1076 004064 011403      MOV     (R4),R3         :READ K2CSRJ
1077 004066 102406      BVS    11$             :V BIT SET BY NXM
1078 004070 010314      MOV     R3,(R4)        :WRITE K2CSRJ
1079 004072 102404      BVS    11$             :V BIT SET BY NXM
1080 004074 110324      MOVB   R3,(R4)+      :WRITE LOW BYTE
1081 004076 102402      BVS    11$             :
1082 004100 105014      CLRB   (R4)           :WRITE HIGH BYTE
1083 004102 102001      BVC    8$             :V BIT SET BY NXM
1084 004104              11$:
                                ;;\
                                ;; >> ERROR 21, ERROR IN K2CSRJ
                                ;;//

004104 104021      ERROR+21

1085
1086 004106 112737 000106 004676 8$:  MOVB    #'F,@#CSRER+<12.> :ASCII FOR ERROR MESSAGE
1087 004114 012704 177534      MOV     #K2CSRJ,R4      :POINT TO NEXT CSR
1088 004120 011403      MOV     (R4),R3         :READ K2CSRJ
1089 004122 102402      BVS    12$             :V BIT SET BY NXM
1090 004124 010314      MOV     R3,(R4)        :WRITE K2CSRJ
1091 004126 102001      BVC    9$             :V BIT SET BY NXM
1092 004130              12$:
                                ;;\
                                ;; >> ERROR 22, ERROR IN K2CSRJ
                                ;;//

004130 104022      ERROR+22

1093
1094 004132 112737 000110 004676 9$:  MOVB    #'H,@#CSRER+<12.> :ASCII FOR ERROR MESSAGE
1095 004140 012704 177536      MOV     #K2CSRH,R4      :POINT TO K2CSRH

```


H3

```

1096 004144 011403      MOV      (R4),R3      ;READ K2CSRH
1097 004146 102402      BVS      13$          ;NXM SETS THE V BIT
1098 004150 010314      MOV      R3,(R4)     ;WRITE K2CSRH
1099 004152 102001      BVC      15$          ;NXM SETS THE V BIT
1100 004154                13$:
                                ;;\
                                ;; >> ERROR 23, ERROR IN K2CSRH
                                ;;//

1101 004154 104023      ERROR+23
1101 004156 012704 177532 15$:  MOV      #K2QIR,R4    ;POINT TO QBUS INTERRUPT REGISTER
1102 004162 005714      TST      (R4)        ;READ K2CSRH; SHOULD NOT TIME OUT
1103 004164 102402      BVS      14$          ;
1104 004166 005014      CLR      (R4)        ;WRITE K2QIR
1105 004170 102007      BVC      16$          ;NXM SETS THE V BIT
1106 004172                14$:
                                MOV      #QIRER,EMADR  ;;\
                                MOV      #NOSIG,ESADR  ;; >> ERROR 24, ERROR IN K2QIR
                                ERROR+24            ;;//
1107 004210 005037 002426 16$:  CLR      TRAP4X
1108
1109
                                ;*****
                                ;*TEST 2.3   CSR DATA TEST
                                ;*****
1110 004214                2.3.:
1110 004214 012737 004662 044746  MOV      #CSRER,EMADR  ;; SET ERROR MESSAGE...
1110 004222 012737 045210 044752  MOV      #EF2,ESADR    ;...AND SIGNATURE POINTERS...
1110 004230 000240      NOP                    ;...ERROR WILL BE CALLED LATER.
1111 004232 012737 004731 044742  MOV      #CSRDAT,EMPRE ;ERROR MESSAGE PREFIX
1112 004240 112737 000101 004676  MOVB     #'A,#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1113 004246 012704 177520      MOV      #K2CSRA,R4
1114 004252 011403      MOV      (R4),R3
1115 004254 106427 000340      MTPS     #PR7          ;NO INTERRUPTS
1116 004260 012702 000252      MOV      #252,R2       ;SET AND CLEAR ALTERNATE BITS
1117 004264 010214      MOV      R2,(R4)
1118 004266 020214      CMP      R2,(R4)
1119 004270 001005      BNE      5$
1120 004272 000241      CLC
1121 004274 006002      ROR      R2            ;R2 =125
1122 004276 010214      MOV      R2,(R4)       ;SET AND CLEAR ALTERNATE BITS
1123 004300 020214      CMP      R2,(R4)
1124 004302 001407      BEQ      1$
1125 004304 010437 001120 5$:  MOV      R4,$GDADR
1126 004310 011437 001126      MOV      (R4),$BDDAT
1127 004314 010237 001124      MOV      R2,$GDDAT
1128
                                ;;\
                                ;; >> ERROR 25, DATA ERROR IN K2CSRA
                                ;;//

1129 004320 104025      ERROR+25
1129 004322 010314      1$:  MOV      R3,(R4)      ;RESTORE CSR
1130
1131 004324 112737 000103 004676  MOVB     #'C,#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1132 004332 012704 177524      MOV      #K2CSRC,R4
1133 004336 011403      MOV      (R4),R3
1134 004340 010302      MOV      R3,R2
1135 004342 042702 000017      BIC      #17,R2        ;GET R3 CONTENTS
1136 004346 012714 000372      MOV      #372,(R4)     ;TAKE OUT LED BITS
1137 004352 052702 000012      BIS      #12,R2        ;TEST BITS
1138 004356 020214      BIS      #12,R2
1139 004360 001010      CMP      R2,(R4)
                                ;IS R4 OK?
                                ;

```

```

1140 004362 042702 000017      BIC      #17,R2      ;TAKE OUT LED BITS
1141 004366 012714 000365      MOV      #365,(R4)  ;
1142 004372 052702 000005      BIS      #5,R2      ;
1143 004376 020214      CMP      R2,(R4)    ;IS R4 OK?
1144 004400 001407      BEQ      3$
1145 004402 010437 001120      7$:     MOV      R4,$GDADR
1146 004406 011437 001126      MOV      (R4),$BDDAT
1147 004412 010237 001124      MOV      R2,$GDDAT
1148
                                ;;\
                                ;; >> ERROR 26, K2CSRC DATA ERROR
                                ;;//
                                ERROR+26
1149 004416 104026      MOV      R3,(R4)
1150 004420 010314      MOV      #'J,@#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1151 004422 112737 000112 004676 3$:     MOV      #K2CSRJ,R4
1152 004424 011403      MOV      (R4),R3
1153 004436 012702 000252      MOV      #252,R2
1154 004442 010214      MOV      R2,(R4)
1155 004444 012701 000212      MOV      #212,R1
1156 004450 020114      CMP      R1,(R4)
1157 004452 001007      BNE      11$
1158 004454 000241      CLC
1159 004456 006002      ROR      R2          ;R2 =125
1160 004460 010214      MOV      R2,(R4)    ;SET AND CLEAR ALTERNATE BITS
1161 004462 012701 000105      MOV      #105,R1
1162 004466 020114      CMP      R1,(R4)
1163 004470 001407      BEQ      8$
1164 004472 010437 001120      11$:    MOV      R4,$GDADR
1165 004476 011437 001126      MOV      (R4),$BDDAT
1166 004502 010137 001124      MOV      R1,$GDDAT
1167
                                ;;\
                                ;; >> ERROR 27, K2CSRJ DATA ERROR
                                ;;//
                                ERROR+27
1168 004506 104027      MOV      R3,(R4)
1169 004510 010314      MOV      #'F,@#CSRER+<12.> ;ASCII FOR ERROR MESSAGE
1170 004512 112737 000106 004676 8$:     MOV      #K2CSRJ,R4 ;POINT TO NEXT CSR
1171 004520 012704 177534      MOV      (R4),R3
1172 004524 011403      MOV      #125200,(R4)
1173 004526 012714 125200      MOV      #125200,R1 ;ALT PATTERN
1174 004532 012701 125200      MOV      #125200,R1
1175 004536 020114      CMP      R1,(R4)
1176 004540 001006      BNE      12$
1177 004542 012714 052400      MOV      #52400,(R4) ;NEXT ALT PATTERN
1178 004546 012701 052400      MOV      #52400,R1
1179 004552 020114      CMP      R1,(R4)
1180 004554 001407      BEQ      9$
1181 004556 010437 001120      12$:    MOV      R4,$GDADR
1182 004562 011437 001126      MOV      (R4),$BDDAT
1183 004566 010137 001124      MOV      R1,$GDDAT
1184
                                ;;\
                                ;; >> ERROR 30, K2CSRF DATA ERROR
                                ;;//
                                ERROR+30
1185 004572 104030      MOV      R3,(R4)
1186 004574 010314      MOV      #'H,@#CSRER+<12.> ;RESTORE CSR
1187 004576 112737 000110 004676 9$:     MOV      #K2CSRH,R4 ;ASCII FOR ERROR MESSAGE
1188 004604 012704 177536      MOV      (R4),R3
1189 004612 012702 052525      MOV      #52525,R2
1190 004616 010214      MOV      R2,(R4)    ;WRITE ALT PATTERN

```

```
1191 004620 021402      CMP      (R4),R2      ;
1192 004622 001005      BNE      13$
1193 004624 000241      CLC
1194 004626 006102      ROL      R2           ;SHIFT PATTERN
1195 004630 010214      MOV      R2,(R4)     ;WRITE PATTERN
1196 004632 021402      CMP      (R4),R2      ;
1197 004634 001407      BEQ      10$
1198 004636 010437 001120 13$:      MOV      R4,$GDADR
1199 004642 011437 001126      MOV      (R4),$BDDAT
1200 004646 010237 001124      MOV      R2,$GDDAT
1201
                                ;;\
                                ;; >> ERROR 31, K2CSRH DATA ERROR
                                ;;//

                                ERROR+31
1202 004652 104031      10$:      MOV      R3,(R4)      ;RESTORE CSR
1203 004654 010314      JMP      LOCTPR
1204 004656 000137 004740 122 CSRER: .ASCIZ /ERROR IN CSRX/
1205 004700 105 122 122 QIRER: .ASCIZ /ERROR IN QIR/
1206 004715 101 104 104 CSRADR: .ASCIZ /ADDRESSING /
1207 004731 104 101 124 CSRDAT: .ASCIZ /DATA /
1208 .EVEN
1209 ;Now test the TPRs from the K2 side
1210 ;
1211 175000 TPR00=175000
1212 175002 TPR01=175002
1213 175004 TPR02=175004
1214 175006 TPR03=175006
1215 175010 TPR04=175010
1216 175012 TPR05=175012
1217 175014 TPR06=175014
1218 175016 TPR07=175016
1219 175020 TPR08=175020
1220 175022 TPR09=175022
1221 175024 TPR10=175024
1222 175026 TPR11=175026
1223 175030 TPR12=175030
1224 175032 TPR13=175032
1225 175034 TPR14=175034
1226 175036 TPR15=175036
1227 ;
1228 ;*****
; *TEST 2.4 LOCAL SIDE TPR ACCESS TEST
;*****
T2.4::
1229 004740 005037 044742 177530 LOCTPR: CLR      EMPRE      ;CLEAR ERROR MESSAGE PREFIX
1230 004744 042737 000100      BIC      @BIT06,@K2CSR0 ;DISABLE QBUS TPR ACCESS
1231 004752 013746 175000      MOV      @TPR00,-(SP)   ;SAVE CONTENTS OF TPR00..
1232 004756 013746 175002      MOV      @TPR01,-(SP)   ;... AND TPR01
1233 004762 012704 175000      MOV      @TPR00,R4      ;POINT TO TPROO
1234 004766 012737 177777 002426      MOV      #-1,TRAP4X     ;ANTICIPATE NXM
1235 004774 011403 1$:      MOV      (R4),R3        ;READ THE TPRS
1236 004776 102410      BVS      2$             ;V BIT BY NXM
1237 005000 010314      MOV      R3,(R4)        ;WRITE THE TPRS
1238 005002 102406      BVS      2$             ;V BIT SET BY NXM
1239 005004 062704 000002      ADD      #2,R4          ;POINT R4 TO NEXT TPR
1240 005010 022704 175040      CMP      @TPR15+2,R4    ;DID WE READ/WRITE ALL TPRS?
1241 005014 001367      BNE      1$
1242 005016 000411      BR       3$
```

```

1243 005020 010437 001120 2$: MOV R4,@#GDADR ;
1244 005024 012737 005532 044746 MOV #TPRER,EMADR ;;\
      005032 012737 045230 044752 MOV #LA16,ESADR ;; >> ERROR 32, TPR TIMED OUT ON LOCAL ACCESS
      005040 104032 ERROR+32 ;://

```

```

1245
1246 005042 005037 002426 3$: CLR TRAP4X ;LOOK FOR UNEXPECTED NXMS
1247 ;*****
; *TEST 2.5 LOCAL SIDE TPR DATA TEST
;*****
T2.5:

```

```

005046
1248 005046 012703 125252 DATPR: MOV #125252,R3 ;INIT R3 WITH DATA
1249 005052 012702 052525 MOV #52525,R2 ;INIT R2 WITH DATA
1250 005056 012704 175000 MOV #TPR00,R4 ;POINT TO FIRST TPR
1251 005062 010324 1$: MOV R3,(R4)+ ;WRITE CHECKER BOARD TO TPR ARRAY
1252 005064 010224 MOV R2,(R4)+
1253 005066 022704 175040 CMP #175040,R4 ;DID WE RUN OUT OF TPRS YET?
1254 005072 001373 BNE 1$
1255 005074 012704 175000 MOV #TPR00,R4 ;POINT TO TPRO0
1256 005100 020324 2$: CMP R3,(R4)+ ;CHECK DATA
1257 005102 001022 BNE 5$ ;BRANCH IF BAD
1258 005104 020224 CMP R2,(R4)+ ;CHECK DATA
1259 005106 001015 BNE 3$ ;BRANCH IF BAD
1260 005110 022704 175040 CMP #TPR15+2,R4 ;DID WE CHECK ALL THE TPRS?
1261 005114 001371 BNE 2$
1262 005116 022702 125252 CMP #125252,R2 ;DID WE TRY BOTH PATTERNS?
1263 005122 001427 BEQ 4$
1264 005124 012704 175000 MOV #TPR00,R4 ;RESET POINTER
1265 005130 012702 125252 MOV #125252,R2 ;CHANGE PATTERN
1266 005134 012703 052525 MOV #52525,R3 ;
1267 005140 000750 BR 1$
1268 005142 010237 001124 3$: MOV R2,@#GDDAT ;ERROR CALL DATA
1269 005146 000402 BR 6$
1270 005150 010337 001124 5$: MOV R3,@#GDDAT
1271 005154 014437 001126 6$: MOV -(R4),@#BDDAT ;GET BAD DATA
1272 005160 010437 001122 MOV R4,@#BDADR ;... AND FAILING TPR ADDRESS
1273 005164 012737 005554 044746 MOV #TPRDER,EMADR ;;\
      005172 012737 045210 044752 MOV #EF2,ESADR ;; >> ERROR 33, BAD DATA IN TPRS
      005200 104033 ERROR+33 ;://

```

```

1274
1275 005202 012637 175002 4$: MOV (SP)+,@#TPR01 ;RESTORE TPRS
1276 005206 012637 175000 MOV (SP)+,@#TPR00 ;
1277 005212 052737 000100 177530 BIS #BIT06,@#K2CSR0 ;RE-ENABLE TPR ACCESS FROM QBUS
1278
1279

```

```

;*****
; *TEST 2.6 SLU ADDRESS ACCESS TEST
;*****
T2.6:

```

```

005220
1280 005220 012702 177560 MOV #177560,R2 ;FIRST SLU ADDRESS
1281 005224 012701 000004 MOV #4,R1 ;
1282 005230 012737 177777 002426 MOV #-1,@#TRAP4X ;ANTICIPATE NXM
1283 005236 005722 1$: TST (R2)+ ;
1284 005240 102007 BVC 2$
1285 005242 012737 005573 044746 MOV #SLU1TO,EMADR ;;\
      005250 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 34, SLU1 REGISTER TIMED OUT ON READ
      005256 104034 ERROR+34 ;://
1286 005260 077112 2$: SOB R1,1$
1287 005262 012702 177560 MOV #177560,R2

```

L3

```

1288 005266 012701 000010      MOV    #10,R1
1289 005272 105722      3$:   TSTB  (R2)+
1290 005274 102007      BVC   4$
1291 005276 012737 005573 044746      MOV    #SLU1TO,EMADR  ;;\
      005304 012737 045206 044752      MOV    #NOSIG,ESADR  ;; >> ERROR 35, SLU1 REGISTER TIMED OUT ON BYTE READ
      005312 104035      ERROR+35  ;;//
1292 005314 077112      4$:   SOB   R1,3$
1293
1294

```

```

;*****
;*TEST 2.7      DTC ADDRESS ACCESS TEST
;*****
T2.7::

```

```

      005316
1295 005316 012702 174400      MOV    #174400,R2      ;FIRST DMA ADDRESS
1296 005322 012737 177777 002426      MOV    #-1,@#TRAP4X   ;ANTICIPATE NXM
1297 005330 005722      1$:   TST  (R2)+
1298 005332 102011      BVC   2$
1299 005334 010237 001120      MOV    R2,$GDADR
1300 005340 012737 005631 044746      MOV    #DTC0,EMADR  ;;\
      005346 012737 045222 044752      MOV    #LA16M2,ESADR ;; >> ERROR 36, DTC REGISTER TIMED OUT ON READ
      005354 104036      ERROR+36  ;;//
1301 005356 022702 174540      2$:   CMP  #174540,R2
1302 005362 001362      BNE   1$
1303
1304

```

```

;*****
;*TEST 2.10    SLU2 ADDRESS ACCESS TEST
;*****
T2.10::

```

```

      005364
1305 005364 012702 175704      MOV    #175704,R2
1306 005370 005022      CLR   (R2)+           ;WRITE ONLY 175704
1307 005372 102445      BVS  10$
1308 005374 005022      CLR   (R2)+           ;      "      175706
1309 005376 102443      BVS  10$
1310 005400 062702 000004      ADD   #4,R2           ;BUMP UP TO 175714
1311 005404 005022      CLR   (R2)+           ;WRITE ONLY 175714
1312 005406 102437      BVS  10$
1313 005410 005012      CLR   (R2)           ;      "      175716
1314 005412 102435      BVS  10$
1315 005414 012702 175700      MOV    #175700,R2
1316 005420 012703 175704      MOV    #175704,R3
1317 005424 005013      5$:   CLR   (R3)           ;Register Pointer = 0
1318 005426 012701 000001      MOV    #1,R1
1319 005432 005712      1$:   TST  (R2)           ;Read the status registers
1320 005434 102424      BVS  10$
1321 005436 010113      MOV    R1,(R3)       ;Next status register
1322 005440 005201      INC   R1
1323 005442 022701 000003      CMP   #3,R1
1324 005446 001371      BNE   1$
1325 005450 005722      TST  (R2)+           ;Increment R2 by 2
1326 005452 005722      TST  (R2)+           ;Read the Receiver Data Buffer
1327 005454 102414      BVS  10$
1328 005456 005722      TST  (R2)+           ;Try to read control registers; should time-out
1329 005460 102412      BVS  10$
1330 005462 005722      TST  (R2)+           ;Try to read xmit buffer; should time-out
1331 005464 102410      BVS  10$
1332 005466 022702 175720      CMP   #175720,R2
1333 005472 001416      BEQ  15$
1334 005474 012703 175714      MOV    #175714,R3

```

M3

1335	005500	012702	175710					MOV	#175710,R2		
1336	005504	000747						BR	5\$		
1337	005506	010237	001120			10\$:		MOV	R2,\$GDADR		
1338	005512	012737	005612	044746				MOV	#SLU2TO,EMADR	::\	
	005520	012737	045222	044752				MOV	#LA16M2,ESADR	:: >>	ERROR 37, SLU2 REGISTER TIMED OUT
	005526	104037							ERROR+37	:://	
1339	005530					15\$:					
	005530	000450						BR	TST3	::	
1340	005532	124	120	122	TPRER:	.ASCIZ	/TPR ADDRESS ERROR/			::	
1341	005554	124	120	122	TPRDER:	.ASCIZ	/TPR DATA ERROR/			::	
1342	005573	123	114	125	SLU1TO:	.ASCIZ	/SLU1 TIMED-OUT/			::	
1343	005612	123	114	125	SLU2TO:	.ASCIZ	/SLU2 TIMED-OUT/			::	
1344	005631	104	124	103	DTCTO:	.ASCIZ	/DTC TIMED-OUT/			::	
1345						.EVEN					

N3

KXJ11-CA FUNCTIONAL TEST
T2.10

SLU2 ADDRESS ACCESS TEST

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 38

1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371

```
;  
;MODULE NAME:  RAMBO - K2 512KB RAM TEST  
;  
;AUTHOR:      HENRY ENMAN      CREATION DATE 28 JUNE-85  
;  
;FUNCTIONAL DESCRIPTION:  
;  
;      THIS TEST PROVIDES A FUNCTIONAL TEST OF 512K BYTES OF DYNAMIC  
;      RAM ON THE KXJ11-CA MODULE. ANY ERROR IS CONSIDERED FATAL AND  
;      WILL CAUSE AN EXIT FROM THE TEST. THE PROGRAM EXECUTES IN  
;      KERNEL MODE WITH MEMORY MANAGEMENT, AND 22 BIT ADDRESSING  
;      ENABLED. PRIOR TO TESTING THE RAM THE CONTENTS OF THE STACK  
;      ARE SAVED IN THE USER D SPACE PARS. A 16 WORD DEEP  
;      STACK IS SET UP USING THE SUPERVISOR I AND D SPACE PARS. THIS  
;      PREVENTS CORRUPTING THE RAM BEING TESTED OR WIPING OUT THE STACK  
;      CONTENTS.  
;  
;      ERRORS: ANY ERROR IS CONSIDERED FATAL AND WILL CAUSE THE TEST  
;      TO BE EXITED AND AN ERROR TO BE POSTED.  
;  
;INPUTS:  
;  
;REGISTERS CHANGED:  
;      R0 - R5  
;
```

B4

1373 005650 017600
1374 005652

TOPMEM: .WORD 17600

K2RAM:

::**::*****

::*TEST 3 RAM TEST

::**::*****

TST3: SCOPE

MOV #STACK,SP ;: SET A CLEAN STACK.
MOV #-1,TRAP4X ;: DISMISS BUS-ERRORS.
CLR EMPRE ;: CLEAR ERROR PREFIX.
BR 30011\$;: SKIP NEXT.
30010\$: JMP TST4 ;: BYPASS.
30011\$: T\$NAME, .+4 ;: PRINT TEST NUMBER AND NAME...
BR 30012\$;: ...AND SKIP OVER THE ASCII.
.ASCIZ \ RAM TEST\
.EVEN

005652 000004
005654 012706 001100
005660 012737 177777 002426
005666 005037 044742
005672 000402
005674 000137 011312
005700 104415 005706
005704 000406
005706 040 040 122

005722

30012\$:

1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394

;This is a test of the on board RAM on the K2 module. There are 256k words of RAM on the board. It consists of eighteen 256 k x 1 dynamic RAMs. Sixteen RAMs are for data storage and two are for storage of a parity bit for each byte. The data RAMs are in leadless chip carriers which are mounted on ceramic substrates which form a Single In line Package (SIP). There are four data RAMs per SIP, and four SIPs per module. The parity RAMs are in standard DIP packages.

;Access to the RAM is via a dynamic RAM control circuit which resides in the Data Path and LSI Bus Interface gate array. The dynamic RAM controller circuitry generates the signals needed to interface to the RAM. Refresh for the RAMs is derived from the 14 mhz. clock which also drives the DCJ11 micro-processor. The clock frequency is divided down to provide a signal which generates a refresh request every XXX micro-seconds. The refresh request originates in GA_1 and is sent to the DMA State Machine - a 82S167 PAL, which produces a refresh DMA request, from here on called REFDMR. This signal requests a DMA cycle from the DCJ11. When the DMA cycle is granted the memory refresh is executed.

1395 005722 005037 177572
1396 005726 012737 010540 000004
1397 005734 005037 002426
1398 005740 012737 000340 177776
1399 005746 012737 011174 000114
1400 005754 004737 001462
1401 005760 012737 177777 177672
1402
1403 005766 010637 177676
1404 005772 012706 172300
1405
1406
1407 005776 012704 172352
1408 006002 005037 177540
1409 006006 012737 000020 172516
1410 006014 052737 000001 177572

CLR @MMIO ;Turn off MMU
MOV #TRP.4,@4 ;Bus timeout error routine
CLR @TRAP4X ;Not expecting any traps via 4
MOV #340,@#177776 ;Clear the PSW, set kernel mode.
MOV #PARERR,@#114 ;Set up parity error handler
JSR PC,MMU ;Go initialize the MMU registers
MOV #-1,@#UDPAR5 ; Use UDPAR5 as indicator as to whether
; parity error is anticipated
MOV SP,@#UDPAR7 ;Save the value of sp in UDPAR7
MOV #172300,SP ;Set up a stack using the Supervisor
; I space and D space PARs so that the
; RAM under test will not be corrupted
; R4 points to KIPAR5
CLR @#K2CSRJ ;Disable parity error generation
MOV #20,@#172516 ;Enable 22 bit addresses, disable D space
BIS #1,@#MMIO ;TURN ON MMU

1411
1412
1413
1414
1415

;Unexpected parity errors, non-existent memory traps will vector thru the Kernel I space PAR to the firmware ROM.

;The parity error handler in ROM will set the V bit in the PSW

C4

KXJ11-CA FUNCTIONAL TEST
T3 RAM TEST

MACRO Y05.02 Thursday 03-Apr-86 14:11 Page 40

1416
1417
1418
1419
1420

```
;on each occurrence of a parity error and then do an RTI.  
;  
;The non-existent memory error handler will cause the K2 to post  
;an error indication, exit the test and return to the firmware monitor.  
;
```

D4

KXJ11 CA FUNCTIONAL TEST
T3 RAM TEST

MACRO Y05.02 Thursday 03 Apr-86 14:11 Page 41

```
1422 ;First we'll establish a checksum for the diagnostic stuff resident in
1423 ;the RAM
1424 ;
1425
1426 006022 005002 CLR R2
1427 006024 012703 051506 MOV #LASTAD,R3
1428 006030 005001 CLR R1
1429 006032 062102 1$: ADD (R1)+,R2 ;R2 WILL HOLD SUM IGNORING CARRIES
1430 006034 020301 CMP R3,R1
1431 006036 002375 BGE 1$
1432 006040 010237 177666 MOV R2,@#177666 ;SAVE CHECK SUM IN TPR15
1433
```

E4

1435
1436
1437
1438
1439
1440
1441
1442
1443
1444

;Let's check that the RAM is working in a rudimentary fashion. Meaning that
;the address decoders are working, the rams hold data and that refresh cycles
;are occurring.
;
;In the first 32 kwords of memory each address will be written with its own
;address. The next 32 kword bank will be written with virtual address 0 = 177776
;thru to virtual address 177776 = 0. The third 32 kword bank of memory will
;repeat the pattern in the first bank and so on...

;*TEST 3.1 RAM ADDRESSING TEST

T3.1::

006044
1445 006044 004737 010450
1446
1447 006050 072327 000006
1448 006054 010200
1449 006056 162700 120000
1450 006062 060300
1451 006064 000404
1452 006066 012701 010000
1453 006072 012702 120000
1454 006076 010022
1455 006100 062700 000002
1456 006104 077104
1457 006106 023714 005650
1458 006112 001424
1459 006114 062714 000200
1460 006120 005700
1461 006122 001361
1462
1463 006124 012701 010000
1464 006130 012702 120000
1465 006134 162700 000002
1466 006140 010022
1467
1468 006142 077104
1469 006144 023714 005650
1470 006150 001405
1471 006152 062714 000200
1472 006156 005700
1473 006160 001361
1474 006162 000741
1475
1476
1477 006164 004737 010514
1478 006170 001052
1479 006172 004737 010450
1480
1481 006176 072327 000006
1482 006202 010200
1483 006204 162700 120000
1484 006210 060300
1485 006212 000404
1486 006214 012701 010000
1487 006220 012702 120000
1488 006224 020022

JSR PC,PROMLO ;GO SEE IF PROM IS MAPPED LOW
; AND INIT KIPARS ACCORDINGLY
;
ASH #6,R3
MOV R2,R0
SUB #120000,R0
ADD R3,R0 ;STARTING DATA FOR R0
BR 1\$
15\$: MOV #4096.,R1 ;INIT COUNTER
MOV #120000,R2
1\$: MOV R0,(R2)+ ;WRITE 32 KWORDS OF MEMORY WITH ADDRESS
ADD #2,R0
SOB R1,1\$
CMP TOPMEM,(R4) ;DID WE HIT THE TOP?
BEQ 10\$;BRANCH OUT OF SETUP IF YES
ADD #200,(R4) ;GO TO NEXT WINDOW
TST R0 ;DID R0 WRAP AROUND TO ZERO YET?
BNE 15\$
;WRITE NEXT 32 KWORDS
2\$: MOV #4096.,R1
MOV #120000,R2
3\$: SUB #2,R0
MOV R0,(R2)+ ;WRITE ADDRESS WITH 2S COMPLIMENT
;...OF ADDRESS
SOB R1,3\$
CMP TOPMEM,(R4) ;DID WE HIT TOP
BEQ 10\$
ADD #200,(R4) ;GO TO NEXT WINDOW
TST R0
BNE 2\$;DID IT HIT ZERO YET?
BR 15\$
;Now check memory for the right stuff.
10\$: JSR PC,CHEKSUM
BNE 21\$
JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
;INIT KIPARS AND R1 ACCORDINGLY
;
ASH #6,R3
MOV R2,R0
SUB #120000,R0
ADD R3,R0 ;STARTING DATA FOR R0
BR 11\$
16\$: MOV #4096.,R1
MOV #120000,R2
11\$: CMP R0,(R2)+ ;CHECK THE DATA

F4

KXJ11-CA FUNCTIONAL TEST
T3.1 RAM ADDRESSING TEST

1489	006226	001044				BNE	20\$		
1490	006230	062700	000002			ADD	#2,R0		
1491	006234	077105				SOB	R1,11\$		
1492	006236	023714	005650			CHP	TOPMEM,(R4)		;DID WE HIT THE TOP YET?
1493	006242	001447				BEQ	25\$;IF YES THEN TEST IS DONE
1494	006244	062714	000200			ADD	#200,(R4)		;GO TO NEXT WINDOW
1495	006250	005700				TST	R0		
1496	006252	001360				BNE	16\$		
1497	006254	012701	010000	17\$:		MOV	#4096.,R1		
1498	006260	012702	120000			MOV	#120000,R2		
1499	006264	162700	000002	12\$:		SUB	#2,R0		
1500	006270	020022				CHP	R0,(R2)+		;IS DATA STILL OK?
1501	006272	001022				BNE	20\$		
1502	006274	077105				SOB	R1,12\$		
1503	006276	023714	005650			CHP	TOPMEM,(R4)		;DID WE HIT THE TOP?
1504	006302	001427				BEQ	25\$		
1505	006304	062714	000200			ADD	#200,(R4)		;GO TO NEXT WINDOW
1506	006310	005700				TST	R0		;IS R0=0
1507	006312	001360				BNE	17\$		
1508	006314	000737				BR	16\$;DO THE NEXT 32 KWORDS
1509	006316	013706	177676	21\$:		MOV	@#UDPAR7,SP		
1510	006322	012737	010712	044746		MOV	#CKSUM,EMADR	::\	
	006330	012737	010704	044752		MOV	#FATALR,ESADR	:: >>	ERROR 40, CHECKSUM ERROR
	006336	104040					ERROR+40	:://	
1511	006340	013706	177676	20\$:		MOV	@#UDPAR7,SP	::\	
1512	006344	012737	011006	044746		MOV	#FRAME,EMADR	::\	
	006352	012737	010634	044752		MOV	#FATALW,ESADR	:: >>	ERROR 41,
	006360	104041					ERROR+41	:://	
1513									;POSSIBLE CAUSE OF ERROR IS A
1514									;BAD RAM CELL, ADDRESS DECODER
1515									;OR REFRESH NOT OCCURRING
1516	006362	032777	002000	172550	25\$:	BIT	#BIT10,@SWR		;DO EXTENDED RAM TESTS?
1517	006370	001002				BNE	26\$;YES
1518	006372	000137	006752			JMP	PARCHK		;NO, BUT DO CHECK THE PARITY STUFF.
1519	006376				26\$:				

```

1521 ;Now we'll check the address decoders, and look for interaction between
1522 ;memory cells in the same RAM.
1523 ;
1524 ;*****
;*TEST 3.2 RAM ADDRESS DECODERS TEST
;*****
T3.2::
;Initialize memory with all zeros. This uses a DATO and DATI bus cycle to RAM.
;
; JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
; ;INIT KIPARS AND R1 ACCORDINGLY
1525 006376
1526
1527 006376 004737 010450
1528
1529 006402 000404
1530 006404 012701 010000 10$: MOV #4096.,R1 ;Use R1 as a counter
1531 006410 012702 120000 MOV #120000,R2 ;Set R2 to 120000
1532 006414 005012 15$: CLR (R2) ;Write zeros at all addresses (DATO)
1533 006416 005722 TST (R2)+ ;Ensure that it is zero (DATI)
1534 006420 001023 BNE 20$ ; Branch to error if not zero
1535 006422 007104 SOB R1,15$ ;Loop control
1536 006424 000000 005650 CMP TOPMEM,(R4) ;Have we hit top yet?
1537 006430 000000 BEQ 25$ ; Branch if yes
1538 006432 002714 000200 ADD #200,(R4) ;Move window up 4 kwords
1539 006436 000762 BR 10$ ;Do it again
1540 006440 004737 010514 25$: JSR PC,CHEKSUM
1541 006444 001422 BEQ 26$
1542 006446 013706 177676 MOV @#UDPAR7,SP
1543 006452 012737 010712 044746 MOV #CKSUM,EMADR ;;\
006460 012737 010704 044752 MOV #FATALR,ESADR ;; >> ERROR 42, CHECKSUM ERROR
006466 104042 ERROR+42 ;://
1544 006470 013706 177676 20$: MOV @#UDPAR7,SP
1545 006474 012737 011006 044746 MOV #RAME,EMADR ;;\
006502 012737 010640 044752 MOV #FATAL1,ESADR ;; >> ERROR 43,
006510 104043 ERROR+43 ;://
;Possible cause of this error
;is a bad RAM cell that can't be cleared,
;or bad MDAL. Failing location is contents
;of R2 minus two.
1546
1547
1548
1549
1550 006512 26$:
1607 ;
1608 ;Check that writing ones at a low address does not affect a
1609 ;memory cell at a higher address
1610 ;
1611 ;Read zeros/ compliment/ write ones at 256 k addresses starting at
1612 ;low RAM working up to address 1777776
1613 ;
1614 006512 012700 177777 MOV #-1,R0 ;LOAD R0 WITH EXPECTED DATA
1615 006516 004737 010176 JSR PC,$UPWRD
1616 ;Memory should be all ones, KIPARS should = 17600, R2 should = 140000
1617 ;
1618 ;Check that writing zeros at a high address does not affect a
1619 ;memory cell at a lower address
1620 ;
1621 ;Read ones/ compliment/ write zeros at 256k addresses starting at
1622 ;1777776 working down to low RAM.
1623 ;
1624 006522 005000 CLR R0 ;LOAD R0 WITH EXPECTED DATA
1625 006524 004737 010314 JSR PC,$DNWRD
1626 ;

```

```
1627                    ;Check that writing ones at a high address does not affect a
1628                    ;memory cell at a lower address
1629                    ;
1630                    ;Read zeros/ compliment/ write ones starting at address 1777776
1631                    ;working down to low RAM
1632                    ;
1633                    ;            MOV        #-1,R0                    ;LOAD R0 WITH EXPECTED DATA
1634                    ;            JSR        PC,$DNWRD
1635                    ;
1636                    ;Memory should now be all ones. KIPAR5 should = 0 or lowest RAM.
1637                    ;R2 should = 120000
1638                    ;
1639                    ;Check that writing zeros at a low address does not affect a
1640                    ;memory cell at a higher address
1641                    ;
1642                    ;Read ones/ compliment/ write zeros at 256k addresses
1643                    ;starting at address 0 working up to address 1777776
1644                    ;
1645                    ;            CLR        R0                    ;LOAD R0 WITH EXPECTED DATA
1646                    ;            JSR        PC,$UPWRD
1647                    ;
1648                    ;Memory should be all zeros, KIPAR5 should = 17600, R2 should = 140000
```

1650
1651
1652
1653
1703

;Now check out the DATIO(B) functions of the module.
;Read byte zeros/ compliment/ write byte ones at 512kbyte addresses
;starting at address 1777777 working down to lowest RAM address.

;;*****
;*TEST 3.3 RAM CELL INTERACTION TEST
;;*****

1704 006546 004737 010450
006552 013714 005650
006556 012701 020000
006562 012702 140000
006566 105142
006570 122712 177777
006574 001020
006576 077105
006600 162714 000200
006604 020314
006606 001363
006610 004737 010450
006614 006301
006616 012702 140000
006622 105142
006624 122712 177777
006630 001002
006632 077105
006634 000411
006636 013706 177676
006642 012737 011006 044746
006650 012737 010640 044752
006656 104044
006660

T3.3:: JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
;INIT KIPAR5 AND R1 ACCORDINGLY
MOV TOPMEM,(R4) ;Start at top of memory
30013\$: MOV #8192.,R1 ;Use R1 as 30013\$ counter
MOV #140000,R2 ;Set R2 =140000
30014\$: COMB -(R2) ;Read byte/ modify / write byte
CMPB #-1,(R2) ;Should be ones
BNE 30015\$; Branch if not
SOB R1,30014\$;Loop control
SUB #200,(R4) ;Move window down 4 kwords
CMP R3,(R4) ;Have we hit low ram?
BNE 30013\$
JSR PC,PROMLO
ASL R1 ;MAKE IT BYTES
MOV #140000,R2
15\$: COMB -(R2)
CMPB #-1,(R2)
BNE 30015\$
SOB R1,15\$
BR 30016\$
30015\$: MOV @#UDPAR7,SP
MOV #FRAME,EMADR ;;\\
MOV #FATAL1,ESADR ;; >> ERROR 44,
ERROR+44 ;://

1705 006660 004737 010450
006664 006301
006666 000404
006670 012701 020000
006674 012702 120000
006700 105012
006702 105712
006704 001010
006706 105222
006710 077105
006712 023714 005650
006716 001414
006720 062714 000200
006724 000761
006726 013706 177676
006732 012737 011006 044746
006740 012737 010640 044752
006746 104045

JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
;INIT KIPAR5 AND R1 ACCORDINGLY
;MAKE IT BYTES
ASL R1
BR 30018\$
30017\$: MOV #8192.,R1 ;Use R1 as 30017\$ counter
MOV #120000,R2 ;Set R2 = 120000
30018\$: CLRB (R2) ;Write byte (DATOB)
TSTB (R2) ;Read. (DATI)
BNE 30019\$; Branch to error if not equal
INCB (R2)+
SOB R1,30018\$;Loop control
CMP TOPMEM,(R4) ;Have we hit top yet?
BEQ 30020\$
ADD #200,(R4) ;Move window up 4 kwords
BR 30017\$
30019\$: MOV @#UDPAR7,SP
MOV #FRAME,EMADR ;;\\
MOV #FATAL1,ESADR ;; >> ERROR 45,
ERROR+45 ;://

1706

30020\$: ;Possible cause of error is
;DATOB access of RAM is faulty.
;Memory should be all zeros. R2 = 140000, KIPAR5 = 17600

1708
1709
1710
1711 006750 000004
1712
1713
1714
1715
1716
1717
1718 006752

;The parity checker/generators maintain odd parity on each byte of
;local RAM. If write wrong parity is enabled then even parity is written
;
PARDEL: .WORD 4 ;See following comment to explain this.
;
;Parity error notification does not take place immediately upon detection of
;bad memory parity. When expecting a parity error a delay should be inserted
;between the occurrence of the parity error and prior to checking to see that
;it has occurred.

PARCHK:
;*****
;*TEST 3.4 PARITY CHECKER/GENERATOR TEST
;*****
T3.4.:

1719 006752 004737 010450
1720 006752 004737 010450
1721 006756 013700 006750
1722 006762 062714 000200
1723 006766 012737 007032 177672
1724 006774 012737 000001 177540
1725 007002 012702 120000
1726 007006 005012
1727 007010 005712
1728 007012 077001
1729 007014 013700 006750
1730 007020 012712 000401
1731 007024 005712
1732 007026 077001
1733 007030 000411
1734 007032 013706 177676
1735 007036 012737 010733 044746
007044 012737 010704 044752
007052 104046

JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
;INIT KIPAR5 AND R1 ACCORDINGLY
MOV PARDEL,R0 ;Load R0 with delay value
ADD #200,(R4) ;Make sure address used is not in program
MOV #1,@#UDPAR5 ;Anticipate parity errors
MOV #1,@#K2CSRJ ;Enable parity error detection
MOV #120000,R2 ;Init R2
CLR (R2) ;Even low RAM location/ parity RAMs = 1
TST (R2) ;Read low RAM location
SOB R0, ;Wait here for interrupt
MOV PARDEL,R0 ;Load R0 with delay value
MOV #401,(R2) ;Make low RAM bytes odd/Parity RAM now = 0
TST (R2) ;Read low RAM
SOB R0, ;Wait here for interrupt
BR 2\$
1\$: MOV @#UDPAR7,SP
MOV #PARE,EMADR ;:\
MOV #FATALR,ESADR ;>> ERROR 46.
ERROR+46 ;://

1736
1737
1738
1739 007054 013700 006750
1740 007060 005037 177672
1741 007064 052737 000002 177540
1742 007072 011212
1743
1744
1745 007074 042737 000002 177540
1746
1747 007102 005712
1748
1749 007104 077001
1750
1751 007106 102014
1752 007110 013700 006750
1753 007114 052737 000002 177540
1754 007122 005012
1755
1756 007124 042737 000002 177540
1757 007132 005712
1758 007134 077001

2\$: MOV PARDEL,R0 ;Load R0 with delay value
CLR @#UDPAR5 ;
BIS #2,@#K2CSRJ ;Enable write wrong parity
MOV (R2),(R2) ;Write it again with write wrong parity
; enabled. Odd low RAM location/ parity
; RAMs = 1
BIC #2,@#K2CSRJ ;Disable write wrong parity so that
; stack doesn't get wrong parity.
TST (R2) ;Read low RAM location expecting parity
; error
SOB R0, ;Filler to make sure we get a demand
; read from J11.
BVC 3\$;Error if no parity error indication.
MOV PARDEL,R0 ;Load R0 with delay value
BIS #2,@#K2CSRJ ;Enable write wrong parity
CLR (R2) ;Make low RAM bytes even/Parity RAM
; now = 0
BIC #2,@#K2CSRJ ;Disable writing wrong parity
TST (R2) ;Read low RAM
SOB R0, ;Filler make sure we get a demand


```

1759                                     ; read from J11.
1760
1761 007136 102412                       BVS    4$
1762 007140 005012                       3$:   CLR    (R2)                       ;Did a parity error occur?
                                           ;try restoring correct parity
                                           ;...prior to calling error
1763
1764 007142 013706 177676                 MOV    @#UDPAR7,SP
1765 007146 012737 010733 044746         MOV    #PARE,EMADR                       ;;\
                                           MOV    #FATALR,ESADR                      ;; >> ERROR 47.
                                           ERROR+47                                  ;://
1766                                     ;Possible cause of error is bad parity
1767                                     ; RAM or parity generator/checker
1768 007164 005012                       4$:   CLR    (R2)                       ;Restore correct parity.
                                           ;Test parity generator/checker with all patterns
1769
1770                                     ;
1771                                     ;Check low byte parity generator/checker
1772                                     ;
1773 007166 012737 007204 177672         MOV    #10$,@#UDPAR5                     ;
1774 007174 105012                       CLR    (R2)                             ;Zero low byte
1775 007176 105212                       5$:   INCB   (R2)                       ;Do all patterns
1776 007200 001412                       BEQ    6$
1777 007202 000775                       BR     5$
                                           ;Loop control
                                           ;Zero if no parity errors
1778 007204 013706 177676                 10$:  MOV    @#UDPAR7,SP
1779 007210 012737 010733 044746         MOV    #PARE,EMADR                       ;;\
                                           MOV    #FATALR,ESADR                      ;; >> ERROR 50.
                                           ERROR+50                                  ;://
1780                                     ;Possible cause of error is
1781                                     ; bad parity checker/generator
1782                                     ;Check high byte parity checker/generator
1783                                     ;
1784 007226 012737 007246 177672         6$:   MOV    #11$,@#UDPAR5                 ;
1785 007234 005202                       INC    R2
                                           ;R2 = 120001
1786 007236 105012                       CLR    (R2)                             ;Zero high byte
1787 007240 105212                       7$:   INCB   (R2)                       ;Do all patterns
1788 007242 001412                       BEQ    8$
1789 007244 000775                       BR     7$
                                           ;Loop control
                                           ;Zero if no parity errors
1790 007246 013706 177676                 11$:  MOV    @#UDPAR7,SP
1791 007252 012737 010733 044746         MOV    #PARE,EMADR                       ;;\
                                           MOV    #FATALR,ESADR                      ;; >> ERROR 51.
                                           ERROR+51                                  ;://
1792                                     ;Possible cause of error is
1793                                     ; bad parity checker/generator
1794 007270 012737 177777 177672         8$:   MOV    #-1,@#UDPAR5

```

1796
1797
1798
1799
1800
1801 007276 032777 002000 171634
1802 007304 001004
1803 007306 013706 177676
1804 007312 000137 011232
1805
1806 007316

;The parity RAM consists of two 256k x 1 RAM chips. One for the low
;bytes and one for the high bytes of 256k words of data RAM.

;Test the low byte parity RAM if extended memory tests are wanted.

```

;
; BIT      #BIT10,@SWR      ;DO EXTENDED RAM TESTS?
; BNE     LOWPAR           ;YES
; MOV     @#UDPAR7,SP      ;NO, RESTORE STACK
; JMP     ENDRAM          ;AND EXIT TEST

```

LOWPAR:

```

;*****
; *TEST 3.5  LOW BYTE PARITY RAM TEST
;*****
T3.5::

```

1807 007316 004737 010450

007322 000404
007324 012701 010000
007330 012702 120000
007334 112712 000001
007340 005722

007342 077104
007344 023714 005650
007350 001403
007352 062714 000200
007356 000762
007360 004737 010514
007364 001422
007366 013706 177676
007372 012737 010712 044746
007400 012737 010704 044752
007406 104052
007410 013706 177676
007414 012737 010762 044746
007422 012737 010634 044752
007430 104053

```

;JSR     PC,PROMLO      ;FIND LAST DIAGNOSTIC ADDRESS AND
;                               ;INIT KIPAR5 AND R1 ACCORDINGLY
;BR      30022$
;MOV     #4096.,R1      ;Use R1 as counter
;MOV     #120000,R2     ;Reset R2 to 120000
;MOV     #1,(R2)       ;Read / modify / write
;TST     (R2)+
; Branch to error
;SOB     R1,30022$      ;Loop control
;CMP     TOPMEM,(R4)    ;Have we hit top yet?
;BEQ     30024$
;ADD     #200,(R4)      ;Move window up 4 kwords
;BR      30021$
;30024$: JSR     PC,CHEKSUM
;BEQ     30025$
;MOV     @#UDPAR7,SP
;MOV     #CKSUM,EMADR   ;;\
;MOV     #FATALR,ESADR  ;; >> ERROR 52, CHECKSUM ERROR
;                               ;://
;30023$: MOV     @#UDPAR7,SP
;MOV     #PRAME,EMADR   ;;\
;MOV     #FATALW,ESADR  ;; >> ERROR 53,
;                               ;://
;                               ;possible causes of this error
;                               ;are bad address decoder, bad
;                               ;MDAL, interaction between RAM cells.
;30025$:
;Low byte data RAM should = 00000001, low byte parity RAM should = 0
;Check that writing ones at a low parity RAM address does not affect a
;memory cell at a higher address
;Read zero/ negate/ write one at 256 k low byte parity RAM locations
;starting at address 0 working up to address 177776
;
;MOV     #120000,R0
;JSR     PC,$UPPAR
;Low byte parity RAMs should be all ones, low byte data RAMs should = -1
;KIPAR5 should = 17600, ;R2 should = 140000
;Check that writing zero at a high parity RAM address does not affect a

```

007432
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817 007432 012700 120000
1818 007436 004737 010046
1819
1820
1821
1822

```
1823                                   ;memory cell at a lower address
1824                                   ;
1825                                   ;Read one/ negate/ write zero at 256k low byte parity RAM locations
1826                                   ;starting at 1777776 working down to lowest RAM address.
1827                                   ;
1828 007442 012700 140000            MOV     #140000,R0
1829 007446 004737 007674            JSR     PC,$DNPAR
1830                                   ;
1831                                   ;Low byte parity RAMs should = 0, low byte data RAM should = 1
1832                                   ;KIPAR5 should = 0, R2 should = 120000
1833                                   ;
1834                                   ;Check that writing one at a high address in the low byte parity RAM
1835                                   ;does not affect a memory cell at a lower address
1836                                   ;
1837                                   ;Read zero/ compliment/ write one starting at address 1777776
1838                                   ;working down to address 0.
1839                                   ;
1840 007452 012700 140000            MOV     #140000,R0
1841 007456 004737 007674            JSR     PC,$DNPAR
1842                                   ;
1843                                   ;Low byte parity RAM should = all ones, Low byte data RAM should be all ones.
1844                                   ;KIPAR5 should = 0, R2 should = 120000
1845                                   ;
1846                                   ;Check that writing zero at a low address in the low byte parity RAM
1847                                   ;does not affect a memory cell at a higher address
1848                                   ;
1849                                   ;Read ones/ negate/ write zeros at 256k addresses
1850                                   ;starting at low RAM working up to address 1777776
1851                                   ;
1852 007462 012700 120000            MOV     #120000,R0
1853 007466 004737 010046            JSR     PC,$UPPAR
```

```

1855                                     ;
1856                                     ;Test high byte parity RAM
1857                                     ;
1858
1859 007472                               HIPAR:
                                       ;*****
                                       ;*TEST 3.6      HIGH BYTE PARITY RAM TEST
                                       ;*****
1860 007472 012737 007576 177672         T3.6::
1861 007500 004737 010450                MOV    #25$,@#UDPAR5
                                       JSR    PC,PROMLO                ;FIND LAST DIAGNOSTIC ADDRESS AND
                                       ;INIT KIPAR5 AND R1 ACCORDINGLY
1862
1863 007504 005202                        INC    R2
1864 007506 000404                        BR     15$
1865 007510 012701 010000                10$:  MOV    #4096.,R1                ;Use R1 as a counter
1866 007514 012702 120001                MOV    #120001,R2                ;Point to high byte
1867 007520 112712 000001                15$:  MOVB  #1,(R2)                ;Write zeros at all high byte
                                       ;           parity RAM locations (DAT0)
1868                                       ;           and increment R2 by one
1869 007524 105722                        TSTB  (R2)+                ;Ensure that it is zero (DATI)
1870                                       ;           and increment R2 by one
1871 007526 005202                        INC    R2                ;Make R2 point to next high byte
1872 007530 077105                        SOB    R1,15$                ;Loop control
1873 007532 023714 005650                CMP    TOPMEM,(R4)                ;Is KIPAR5 = Hi RAM
1874 007536 001403                        BEQ    26$
1875 007540 062714 000200                ADD    #200,(R4)                ;Move window up 4 kwords
1876 007544 000761                        BR     10$
1877 007546 004737 010514                26$:  JSR    PC,CHEKSUM
1878 007552 001422                        BEQ    27$
1879 007554 013706 177676                MOV    @#UDPAR7,SP
1880 007560 012737 010712 044746         MOV    #CKSUM,EMADR            ;;\
007566 012737 010704 044752         MOV    #FATALR,ESADR           ;; >> ERROR 54, CHECKSUM ERROR
007574 104054                        ERROR+54                       ;;//
1881 007576 013706 177676                25$:  MOV    @#UDPAR7,SP
1882 007602 012737 010762 044746         MOV    #PRAME,EMADR            ;;\
007610 012737 010626 044752         MOV    #FATALB,ESADR           ;; >> ERROR 55,
007616 104055                        ERROR+55                       ;;//
1883 007620 005037 177672                27$:  CLR    @#UDPAR5
1884
1885                                     ;
1886                                     ;High byte data RAM should = 00000001, high byte parity RAM should = 0
1887                                     ;
1888                                     ;Check that writing ones at a low address in the high byte parity RAM
1889                                     ;does not affect a memory cell at a higher address
1890                                     ;
1891                                     ;Read zero/ negate/write one at 256 k high byte parity RAM locations
1892                                     ;starting at lowest physical RAM address +1 working up to address 1777777
1893                                     ;
1894 007624 012700 120001                MOV    #120001,R0
1895 007630 004737 010046                JSR    PC,$UPPAR
1896
1897                                     ;
1898                                     ;High byte parity RAMs should be all ones, high byte data RAMs should = -1
1899                                     ;KIPAR5 should = 17600, ;R2 should = 140001
1900                                     ;
1901                                     ;Check that writing zero at a high address in the high byte parity RAM
1902                                     ;does not affect a memory cell at a lower address
1903                                     ;
1903                                     ;Read one/ negate/ write zero at 256k high byte parity RAM locations
    
```

B5

```

1904 ;starting at physical address 1777777 working down to low RAM-1.
1905 ;
1906 007634 012700 140001      MOV    #140001,R0
1907 007640 004737 007674      JSR    PC,$DNPARG
1908 ;
1909 ;High byte parity RAMs should = 0, high byte data RAM should = 1
1910 ;KIPARS should = 0, R2 should = 120001
1911 ;
1912 ;Check that writing one at a high address in the high byte parity RAM
1913 ;does not affect a memory cell at a lower address
1914 ;
1915 ;Read zero/ negate/write one starting at physical address 1777777
1916 ;working down to address 1.
1917 ;
1918 007644 012700 140001      MOV    #140001,R0
1919 007650 004737 007674      JSR    PC,$DNPARG
1920 ;
1921 ;High byte parity RAM should = all ones, high byte data RAM should be all ones.
1922 ;KIPARS should = 0, R2 should = 120001
1923 ;
1924 ;Check that writing zeros at a low address in the high byte parity RAM
1925 ;does not affect a memory cell at a higher address
1926 ;
1927 ;Read ones/ negate/ write zeros at 256k addresses
1928 ;starting at physical address 1 working up to address 1777777.
1929 ;
1930 007654 012700 120001      MOV    #120001,R0      ;STARTING BYTE
1931 007660 004737 010046      JSR    PC,$UPPARG
1932 ;
1933 ;At this point the functionality of 256k words of on board RAM
1934 ;has been demonstrated. Applications can now be executed from RAM,
1935 ;and trap and interrupt vectors can be initialized.
1936 ;
1937 007664 013706 177676      MOV    @#UDPAR7,SP    ;GET OLD STACK CONTENTS
1938 007670 000137 011232      JMP    ENDRAM         ;BAIL OUT

```

```

1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950 007674 012737 010024 177672 $DNP: MOV #3$,@#UDPAR5 ;Anticipate parity error
1951 007702 004737 010450 JSR PC,PROMLO ;FIND LAST DIAGNOSTIC ADDRESS AND
1952 ;INIT KIPARS AND R1 ACCORDINGLY
1953 007706 013714 005650 MOV TOPMEM,(R4) ;Init KIPARS to top of memory
1954 007712 012701 010000 1$: MOV #4096.,R1 ;Initialize R1
1955 007716 010002 MOV R0,R2 ;Set R2 = starting address
1956 007720 162702 000002 2$: SUB #2,R2 ;Get low byte
1957 007724 105412 NEGB (R2) ;Read /Mod /Write
1958 007726 077104 SOB R1,2$ ;Loop control
1959 007730 162714 000200 SUB #200,(R4) ;Move window down 4 kwords
1960 007734 020314 CMP R3,(R4) ;Is KIPARS = low RAM
1961 007736 001365 BNE 1$ ;
1962 007740 004737 010450 JSR PC,PROMLO ;
1963 007744 012702 140000 MOV #140000,R2 ;Set R2= start
1964 007750 162702 000002 4$: SUB #2,R2 ;
1965 007754 105412 NEGB (R2) ;
1966 007756 077104 SOB R1,4$ ;
1967 007760 012737 010024 177672 MOV #3$,@#UDPAR5 ;Return to 3$ on parity error
1968 007766 004737 010514 JSR PC,CHEKSUM
1969 007772 001003 BNE 5$
1970 007774 005037 177672 CLR @#UDPAR5 ;
1971 010000 000207 RTS PC ;
1972 010002 013706 177676 5$: MOV @#UDPAR7,SP ;
1973 010006 012737 010712 044746 MOV #CKSUM,EMADR ;;\
010014 012737 010704 044752 MOV #FATALR,ESADR ;; >> ERROR 56. CHECKSUM ERROR
010022 104056 ERROR+56 ;://
1974 010024 013706 177676 3$: MOV @#UDPAR7,SP ;
1975 010030 012737 010762 044746 MOV #PRAME,EMADR ;;\
010036 012737 010640 044752 MOV #FATAL1,ESADR ;; >> ERROR 57.
010044 104057 ERROR+57 ;://
1976
    
```

D5

```

1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995 010046 012737 010154 177672 $UPPAR: MOV #20$,@#UDPAR5 ;Anticipate parity error
1996 010054 004737 010450 JSR PC,PROML0 ;FIND LAST DIAGNOSTIC ADDRESS AND
;INIT KIPAR5 AND R1 ACCORDINGLY
1997
1998 010060 150002 BISB R0,R2
1999 010062 000403 BR 15$
2000 010064 012701 010000 10$: MOV #4096.,R1 ;Use R1 as a counter
2001 010070 010002 MOV R0,R2 ;Reset R2 to starting byte
2002 010072 105412 15$: NEGB (R2) ;Read/modify/write ones to parity RAM
; write all ones to low byte data RAM
2003
2004 010074 005202 INC R2 ;Increment R2 by two to next high or low
2005 010076 005202 INC R2 ;..byte depending on what we're testing.
2006 010100 077104 SOB R1,15$ ;Loop control
2007 010102 023714 005650 CMP TOPMEM,(R4) ;Have we hit top yet?
2008 010106 001403 BEQ 25$
2009 010110 062714 000200 ADD #200,(R4) ;Move window up 4 kwords
2010 010114 000763 BR 10$
2011 010116 004737 010514 25$: JSR PC,CHEKSUM
2012 010122 001003 BNE 22$
2013 010124 005037 177672 CLR @#UDPAR5
2014 010130 000207 RTS PC
2015 010132 013706 177676 22$: MOV @#UDPAR7,SP
2016 010136 012737 010712 044746 MOV #CKSUM,EMADR ;;\
010144 012737 010704 044752 MOV #FATALR,ESADR ;; >> ERROR 60, CHECKSUM ERROR
010152 104060 ERROR+60 ;://
2017 010154 013706 177676 20$: MOV @#UDPAR7,SP ;Comes here on a parity error
2018 010160 012737 010762 044746 MOV #PRAME,EMADR ;;\
010166 012737 010640 044752 MOV #FATAL1,ESADR ;; >> ERROR 61.
010174 104061 ERROR+61 ;://
2019
2020 ;possible causes of this error
2021 ;are bad address decoder, bad
;MDAL, interaction between RAM cells.

```

E5

```

2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
010176 010176 004737 010450
010202 000404
010204 012701 010000
010210 012702 120000
010214 005112
010216 020022
010220 001023
010222 077104
010224 023714 005650
010230 001403
010232 062714 000200
010236 000762
010240 004737 010514
010244 001422
010246 013706 177676
010252 012737 010712 044746
010260 012737 010704 044752
010266 104062
010270 013706 177676
010274 012737 011006 044746
010302 012737 010634 044752
010310 104063

2034 010312 000207

;
;-----
;SUBROUTINE $UPWRD
;-----
;SUBROUTINE TO READ / MODIFY / WRITE ALL MEMORY LOCATIONS
;STARTING AT LOW RAM.
;ENTRY PARAMETERS:      R0 = EXPECTED DATA AFTER READ/ MOD/ WRITE
;                        R4 = KIPARS
;
$UPWRD:
      JSR      PC,PROMLO           ;FIND LAST DIAGNOSTIC ADDRESS AND
;                               ;INIT KIPARS AND R1 ACCORDINGLY
      BR      30027$
      MOV     #4096.,R1           ;Use R1 as counter
      MOV     #120000,R2         ;Reset R2 to 120000
30026$: COM      (R2)             ;Read / modify / write
      CMP     R0,(R2)+           ;
      BNE     30028$             ; Branch to error
      SOB     R1,30027$         ;Loop control
      CMP     TOPMEM,(R4)       ;Have we hit top yet?
      BEQ     30029$
      ADD     #200,(R4)         ;Move window up 4 kwords
      BR      30026$
30029$: JSR     PC,CHEKSUM
      BEQ     30030$
      MOV     @#UDPAR7,SP
      MOV     #CKSUM,EMADR      ;;\
      MOV     #FATALR,ESADR     ;; >> ERROR 62, CHECKSUM ERROR
      MOV     ERROR+62          ;;//
30028$: MOV     @#UDPAR7,SP
      MOV     #FRAME,EMADR      ;;\
      MOV     #FATALW,ESADR     ;; >> ERROR 63.
      MOV     ERROR+63          ;;//
;possible causes of this error
;one bad address decoder, bad
;MUL, interaction between RAM cells.
30030$: RTS      PC

```


2036
 2037
 2038
 2039
 2040
 2041
 2042
 2043
 2044
 2045
 2046

010314
 010314 004737 010450

 010320 013714 005650
 010324 012701 010000
 010330 012702 140000
 010334 005142
 010336 020012
 010340 001031
 010342 077104
 010344 162714 000200
 010350 020314
 010352 001364
 010354 004737 010450
 010360 012702 140000
 010364 005142
 010366 020012
 010370 001015
 010372 077104
 010374 004737 010514
 010400 001422
 010402 013706 177676
 010406 012737 010712 044746
 010414 012737 010704 044752
 010422 104064
 010424 013706 177676
 010430 012737 011006 044746
 010436 012737 010634 044752
 010444 104065
 010446
 2047 010446 000207

```

;-----
;SUBROUTINE $DNWRD
;-----
;SUBROUTINE TO READ / MODIFY / WRITE ALL MEMORY LOCATIONS
;STARTING AT TOP OF MEMORY
;ENTRY PARAMETERS:      R0 = EXPECTED DATA AFTER READ/ MOD/ WRITE
;                        R4 = KIPARS
;-----
$DNWRD:
    JSR    PC,PROMLO           ;FIND LAST DIAGNOSTIC ADDRESS AND
                                ;INIT KIPARS AND R1 ACCORDINGLY
                                ;Init KIPARS to top of memory
                                ;Initialize R1
                                ;Set R2 = 140000
                                ;Read / modify / write (DATIO)
                                ;Find (DATI)
                                ;Should be zeros
                                ;Loop control
    MOV    TOPMEM,(R4)        ;Move window down 4 kwords
    30031$: MOV    #4096.,R1    ;Is KIPARS = low RAM
    MOV    #140000,R2
    30032$: COM    -(R2)
    CMP    R0,(R2)
    BNE    30033$
    SUB    #200,(R4)
    15$:  COM    -(R2)
    CMP    R0,(R2)
    BNE    30033$
    SOB    R1,15$
    JSR    PC,CHEKSUM
    BEQ    30034$
    MOV    @#UDPAR7,SP
    MOV    #CKSUM,EMADR      ;;\
    MOV    #FATALR,ESADR    ;; >> ERROR 64, CHECKSUM ERROR
    MOV    ERROR+64        ;;//
    30033$: MOV    @#UDPAR7,SP
    MOV    #RAME,EMADR      ;;\
    MOV    #FATALW,ESADR    ;; >> ERROR 65,
    MOV    ERROR+65        ;;//
    30034$: RTS    PC
    
```

```

2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060 010450 012702 051510
2061 010454 010203
2062 010456 042703 017777
2063 010462 042702 160000
2064 010466 072327 177772
2065 010472 010337 172352
2066 010476 012701 020000
2067 010502 160201
2068 010504 006201
2069 010506 062702 120000
2070 010512 000207

```

```

:-----:
:SUBROUTINE PROMLO
:-----:
:SUBROUTINE TO DETERMINE WHERE LAST DIAGNOSTIC ADDRESS IS SO THAT
:DIAGNOSTIC WILL NOT GET BLOWN AWAY BY RAM TEST
:
:ENTRY PARAMETERS: R4 = KIPARS
:
:ON EXIT: KIPARS AND R3 WILL CONTAIN ADDRESS OF LOW RAM
:
PROMLO: MOV    #LASTAD ,R2
:
:
:MOV    R2,R2
:
:GET PAGE BITS
:BIC    #17777,R2
:GET OFFSET
:BIC    #+C17777,R2
:SHIFT IT FOR PAR ALIGNMENT
:ASH    #-6,R3
:PAGE WHERE TOP OF PROGRAM IS
:MOV    R3,@KIPARS
:NOW FIGURE HOW MANY BYTES WE ARE
:SUB    R2,R1
:FROM THE TOP OF THE PAGE
:ASR    R1
:NUMBER OF WORDS REMAINING IN 8KB BLK
:ADD    #120000,R2
:
:RTS    PC
:Return.

```

H5

```

2072
2073
2074
2075
2076
2077 010514
2078 010514 005002
2079 010516 012703 051506
2080 010522 005001
2081 010524 062102
2082 010526 020301
2083 010530 002375
2084 010532 020237 177666
2085 010536 000207

```

```

-----
;SUBROUTINE TO OBTAIN THE CHECK SUM OF THE DIAGNOSTIC
;RESIDENT IN RAM TO DETERMINE IF ANY LOCATIONS HAVE BEEN
;CORRUPTED BY ACTIVITY IN OTHER AREAS OF RAM.
-----

```

```

CHECKSUM:
          CLR      R2
          MOV      #LASTAD,R3
          CLR      R1
1$:      ADD      (R1)+,R2          .2 WILL HOLD SUM IGNORING CARRIES
          CMP      R3,R1
          BGE      1$
          CMP      R2,@#177666
          RTS      PC

```

```

2087
2088
2099
2090 010540 042737 000002 177540 TRP.4: BIC #2,@#K2CSRJ ;DISABLE WRITING WRONG PARITY JUST IN CASE
2091 010546 011616 MOV (SP),(SP) ;AND ENSURE THAT STACK PARITY IS OK TOO
2092 010550 016666 000002 000002 MOV 2(SP),2(SP) ;WE DON'T WANT POSSIBLE BAD PARITY
2093 010556 005037 177572 CLR @#MMRO ;TURN OFF MMU
2094 010562 010605 MOV SP,R5 ;SAVE CONTENTS OF THE STACK
2095 010564 013706 MOV @#UDPAR7,SP ;SET NEW STACK
2096 010570 011537 001122 MOV (R5),@#BDADR ;SAVE PC AT TIME OF TRAP TO 4
2097 010574 162737 000002 001122 SUB #2,@#BDADR ;CALCULATE ADDRESS AT WHICH TIME OUT OCCURRED.
2098
2099 010602 012737 011030 044746 MOV #BUSTO,EMADR ;:\
010610 012737 011110 044752 MOV #TOEH,ESADR ;: >> ERROR 66, RAM TIMED OUT
010616 104066 ERROR+66 ;://
2100 010620 012716 011232 MOV #ENDRAM,(SP) ;ALTER RETURN ADDRESS
2101 010624 000002 RTI
2102
2103 010626 162702 000001 FATALB: SUB #1,R2 ;GET ADDRESS OF FAILING BYTE
2104 010632 000402 BR FATAL1
2105
2106 010634 162702 000002 FATALW: SUB #2,R2 ;GET ADDRESS OF FAILING WORD
2107 010640 005037 177572 FATAL1: CLR @#MMRO ;MMU OFF
2108 010644 010237 001122 MOV R2,@#BDADR ;MOVE FAILING VIRTUAL LOCATION TO $BDADR
2109 010650 011437 001120 MOV (R4),@#GDADR ;MOVE CONTENTS OF KIPARS TO $GDADR
2110 010654 004737 011120 JSR PC,PACONV
2111 010660 104401 011063 TYPE ,RAMTXT
2112 010664 013746 001120 MOV @#GDADR,-(SP)
2113 010670 104403 TYPOS
2114 010672 002 .BYTE 2 ;TYPE 2 DIGITS
2115 010673 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
2116 010674 013746 001122 MOV @#BDADR,-(SP)
2117 010700 104403 TYPOS
2118 010702 005 .BYTE 5 ;TYPE 5 DIGITS
2119 010703 001 .BYTE 1 ;PRINT LEADING ZEROS
2120 010704 012716 011232 FATALR: MOV #ENDRAM,(SP) ;ALTER RETURN ADDRESS
2121 010710 000207 RTS PC
2122 010712 015 012 103 CKSUM: .ASCIZ <CR><LF>/CHECKSUM ERROR/
2123 010733 015 012 120 PARE: .ASCIZ <CR><LF>/PARITY CIRCUIT ERROR/
2124 010762 015 012 120 PRAME: .ASCIZ <CR><LF>/PARITY RAM ERROR /
2125 011006 015 012 122 RAME: .ASCIZ <CR><LF>/RAM TEST ERROR /
2126 011030 015 012 102 BUSTO: .ASCIZ <CR><LF>/BUS TIME OUT AT ADDRESS /
2127 011063 101 124 040 RAMTXT: .ASCIZ /AT PHYSICAL ADDRESS /
2128 .EVEN
2129
2130 ;BUS TIME-OUT ERROR HANDLER
2131
2132 011110 013746 001122 TOEH: MOV @#BDADR,-(SP) ;LOAD FAILING VIRTUAL ADDRESS
2133 011114 104402 TYPOC ;TYPE IT
2134 011116 000207 RTS PC
2135
2136 ;RECONSTRUCT THE FAILING PHYSICAL ADDRESS
2137
2138 ;ON ENTRY: $BDADR CONTAINS THE ADDRESS OF THE FAILING VIRTUAL ADDRESS.
2139 ; $GDADR CONTAINS THE VALUE IN KIPARS
2140
2141 ;ON EXIT: $BDADR CONTAINS THE LOW FIVE DIGITS OF THE FAILING PHYSICAL ADR
    
```

J5

```

2142                                     ;           $GDADR CONTAINS THE HIGH FIVE DIGITS OF THE FAILING PHYSICAL ADR
2143                                     ;
2144 011120 005000                       PACONV: CLR      RO
2145 011122 012701 000006                 MOV      #6,R1
2146 011126 006337 001120                 1$:     ASL      @#$GDADR           ;ROTATE PAGE NUMBER
2147 011132 006100                       ROL      RO                       ;... INTO RO
2148 011134 077104                       SOB      R1,1$
2149 011136 042737 160000 001122         BIC      #17777,@#$BDADR           ;
2150 011144 063737 001120 001122         ADD      @#$GDADR,@#$BDADR
2151 011152 042737 100000 001122         BIC      #BIT15,@#$BDADR          ;MAKE BIT 15 ZERO
2152 011160 006337 001120                 ASL      @#$GDADR                 ;\
2153 011164 006100                       ROL      RO                       ; >FOR OCTAL PRINT OUT FORMAT
2154 011166 010037 001120                 MOV      RO,@#$GDADR              ;/
2155 011172 000207                       RTS      PC
2156                                     ;-----

```

```

2158
2159
2160
2161
2162 011174 005737 177672
2163
2164
2165
2166
2167 011200 001410
2168 011202 022737 177777 177672
2169 011210 001002
2170 011212 000137 002416
2171 011216 013716 177672
2172 011222 052766 000002 000002
2173
2174 011230 000002
2175
2176
2177 011232 005037 177572
2178 011236 005037 002426
2179 011242 012737 000006 000004
2180 011250 012737 000116 000114
2181 011256 012737 000004 000116
2182 011264 032737 000001 001174
2183 011272 001004
2184 011274 052737 000001 177540
2185 011302 000402
2186 011304 005037 177540
2187 011310
      011310 000400
;-----
;Parity error handler
;PARERR: TST      @#UDPAR5
;Using UDPAR5 as indicator of expected
;...or unexpected parity error.
;0= expected
;1= unexpected
;other non-zero = address to return to.
      BEQ      1$
2168      CMP      #-1,@#UDPAR5
;Is UDPAR5 = -1?
      BNE      2$
2170      JMP      @#UNXTRP
;Go to unexpected trap routine
2171      MOV      @#UDPAR5,(SP)
;New return address
2172      BIS      #2,2(SP)
;Set the V bit in new psw to indicate
; a parity error
      RTI
;Before going to the next test restore the stack.
;ENDRAM: CLR      @#MMRO
;OFF THE MMU
      CLR      TRAP4X
;
      MOV      #6,@#4
;Restore trap vector
      MOV      #116,@#114
;Further parity errors are not expected.
      MOV      #IOT,@#116
;
      BIT      #BIT00,$PASS
;IS PASS COUNTER EVEN
      BNE      1$
2184      BIS      #BIT00,@#K2CSRJ
;ENABLE PARITY ERROR DETECTION ON EVEN
      BR       2$
; PASSES
2186      CLR      @#K2CSRJ
;DISABLE PARITY ERROR DETECTION
2187      BR       TST4      ;;

```

L5

2189
2190

```
;SLUIK2.MAC      23-OCT 85
;*****
;*TEST 4        CONSOLE SERIAL LINE DC319
;*****
```

011312 000064
011314 012706 001100
011320 012737 177777 002426
011326 005037 044742
011332 000402
011334 000137 012506
011340 104415 011346
011344 000416
011346 040 040 103

```
TST4: SCOPE
MOV #STACK,SP ;: SET A CLEAN STACK.
MOV #-1,TRAP4X ;: DISMISS BUS ERRORS.
CLR EMPRE ;: CLEAR ERROR PREFIX.
BR 30036$ ;: SKIP NEXT.
30035$: JMP TST5 ;: BYPASS.
30036$: T$NAME, .+4 ;: PRINT TEST NUMBER AND NAME...
BR 30037$ ;: ...AND SKIP OVER THE ASCII.
.ASCIZ \ CONSOLE SERIAL LINE DC319\
.EVEN
```

011402

```
30037$:
;
; TEST THAT THE CONSOLE PORT INTERRUPTS AS ADVERTISED.
```

2191
2192
2193
2194
2195
2196

```
; *** NOTE: LOOP OPTIONS (SWR<9> AND SWR<8>) ARE DISALLOWED,  
; *** LEST WE LOSE CONTACT WITH THE CONSOLE TTY:.
```

2197 011402 032777 000400 167530
2198 011410 001410
2199 011412 032777 000177 167520
2200 011420 001404
2201 011422 104401 012436
2202 011426 104407
2205 011430 000730
2206 011432
2207 011440 012737 011516 001160
2208 011444 013705 001144
2209 011444 012704 000060
2209 011450 004737 011526
2210 011454 000401
2211

```
;
BIT #BIT8,@SWR ; SWR<8> SET ??
BEQ 1$ ; BR IF NOT.
BIT #177,@SWR ; SWR<6:0> CLEAR ??
BEQ 1$ ; BR IF SO.
TYPE ,CANT ; LOOP-ON-TEST NOT PERMITTED.
GTSWR ; GET ANOTHER SWITCH.
BR TST4
```

2212 011456 104067
2213 011460 132737 000001 001206
2214 011470 005737 001174
2215 011474 001010
2216 011476 004737 011750
2217 011502 000401
2218

```
1$:
MOV #3$, $ESCAPE ;: ESCAPE TO 3$ ON ERROR
MOV $TKS,R5 ;: RCSR POINTER => R5.
MOV #TKVEC,R4 ;: VECTOR POINTER => R4.
CALL DC.XMT ;: EXECUTE XMTR INTERRUPTS...
BR 2$ ;: ...AND RETURN HERE IF OK
```

2219 011504 104070
2220 011506 004737 012222
2221 011512 000401

```
ERROR+67
;://
;: >> ERROR 67, TRANSMIT INTERRUPT ERROR.
;://
;: IN APT MODE?
;: DO TEST IF NOT
;: IS THIS THE FIRST PASS
;: IF NOT THEN DON'T DO THE NEXT TESTS.
;: EXECUTE RCVR INTERRUPTS...
;: ...AND RETURN HERE IF OK.
;://
;: >> ERROR 70, RCVR INTERRUPT ERROR.
```

2222 011514 104071
2223 011516 105037 001103
2224 011522 000137 012504

```
4$:
CALL DC.DAT ;: EXECUTE DATA TESTS
BR 3$ ;: ...AND RETURN HERE IF OK
;://
;: >> ERROR 71, CONSOLE DATA TEST ERROR
;://
;: DISALLOW SWR<9> IN NEXT .SCOPE.
;< AND THATS ALL>
```

```

2226 ;
2227 ; ON ENTRY, (R5) = RCSR AND (R4) = VECTOR.
2228 ; RETURN TO CALL+2 IF OK, CALL+4 IF ERROR.
2229 ;
2230 ; *** TRANSMITTER ***
2231 ; CHECK THAT THE XMTR INTERRUPTS THRU VECTOR (R4) AT LEVEL 4
2232 ; AND THAT THE INTERRUPT ACKNOWLEDGE DROPS THE REQUEST.
2233 ;
2234 000004 TCS= 4
2235 000006 TBUF= 6
2236 000004 TVEC= TCS
2237 000006 TPRI= TBUF
2238 ;
2239 011526 012737 047211 044742 DC.XMT: MOV #SLIX,EMPRE ; SET ERROR PREFIX.
2240 011534 016446 000004 MOV TVEC(R4),-(SP) ; SAVE XMTR VECTOR...
2241 011540 016446 000006 MOV TPRI(R4),-(SP) ; ...AND LEVEL.
2242 011544 106427 000200 MTPS #PR4 ; RAISE CPU TO XMTR LEVEL.
2243 011550 012764 011604 000004 MOV #2$,TVEC(R4) ; SET XMTR VECTOR...
2244 011556 012764 000200 000006 MOV #PR4,TPRI(R4) ; ...AND LEVEL.
2245 011564 105765 000004 1$: TSTB TCS(R5) ; WAIT FOR XMTRDY.
2246 011570 100375 BPL 1$
2247 011572 052765 000100 000004 BIS #100,TCS(R5) ; SET INTERRUPT ENABLE.
2248 011600 000240 240 ; INTERRUPT SHOULD BE HELD OFF...
2249 011602 000411 BR 3$ ; ...BR IF SO.
2250 011604 022626 2$: CMP (SP)+,(SP)+ ; IT WASN' T...
2251 011606 012737 047225 044746 MOV #SLI2,EMADR ; SET ERROR MESSAGE...
011614 012737 045206 044752 MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
011622 000240 NOP ; ...ERROR WILL BE CALLED LATER.
2252 011624 000436 BR 6$ ; GET OUT "INT LEVEL INCORRECT".
2253 ;
2254 011626 012764 011662 000004 3$: MOV #4$,TVEC(R4) ; CHANGE THE VECTOR.
2255 011634 106427 000000 MTPS #PRO ; LOWER CPU PRIORITY.
2256 011640 000240 240 ; INTERRUPT SHOULD HAVE COME IN.
2257 011642 012737 047265 044746 MOV #SLI3,EMADR ; SET ERROR MESSAGE...
011650 012737 045206 044752 MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
011656 000240 NOP ; ...ERROR WILL BE CALLED LATER.
2258 011660 000420 BR 6$ ; GET OUT "INT NOT RECEIVED".
2259 ;
2260 011662 012764 011702 000004 4$: MOV #5$,TVEC(R4) ; INT REC'D, CHANGE THE VECTOR.
2261 011670 012716 011676 MOV #.+6,(SP)
2262 011674 000002 RTI ; RESTORE CPU PRI 0...
2263 011676 000240 240 ; ...SHOULD NOT GET ANOTHER.
2264 011700 000413 BR 7$ ; ...OK, EXIT.
2265 011702 022626 5$: CMP (SP)+,(SP)+ ; 2ND INTERRUPT -- TROUBLE...
2266 011704 012737 047314 044746 MOV #SLI4,EMADR ; SET ERROR MESSAGE...
011712 012737 045206 044752 MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
011720 000240 NOP ; ...ERROR WILL BE CALLED LATER.
2267 ;
2268 011722 062766 000002 000004 6$: ADD #2,4(SP) ; ERROR -- TAKE SKIP RETURN.
2269 011730 042765 000100 000004 7$: BIC #100,TCS(R5) ; CLEAR THE INT ENABLE.
2270 011736 012664 000006 MOV (SP)+,TPRI(R4) ; RESET VECTOR.
2271 011742 012664 000004 MOV (SP)+,TVEC(R4)
2272 011746 000207 RETURN

```



```

2274
2275
2276
2277
2278
2279      000000      RCS= 0
2280      000002      RBUF= 2
2281      000000      RVEC= RCS
2282      000002      RPRI= RBUF
2283
2284 011750 012737 047217 044742 DC.RCV: MOV #SLIR,EMPRE ; SET ERROR PREFIX.
2285 011756 016446 000000      MOV RVEC(R4),-(SP) ; SAVE RCVR VECTOR...
2286 011762 016446 000002      MOV RPRI(R4),-(SP) ; ...AND LEVEL.
2287 011766 106427 000200      MTPS #PR4 ; RAISE CPU LEVEL.
2288 011772 012764 012052 000000      MOV #2$,RVEC(R4) ; SET RCVR VECTOR.
2289 012000 012764 000200 000002      MOV #PR4,RPRI(R4) ; ...AND LEVEL.
2290 012006 052765 000004 000004      BIS #4,TCS(R5) ; CLOSE "MAINT" SWITCH...
2291 012014 005000      CLR R0
2292 012016 110065 000006      MOV#B R0,TBUF(R5) ; ...TRANSMIT A NULL CHARACTER...
2293 012022 105765 000000      1$: TSTB RCS(R5) ; ...WAIT 'TIL RCVR GETS IT...
2294 012026 100401      BMI .+4
2295 012030 077004      SOB R0,1$ ; ... (BUT DON'T WAIT FOREVER)...
2296 012032 042765 000004 000004      BIC #4,TCS(R5) ; ...AND OPEN "MAINT" SWITCH.
2297 012040 052765 000100 000000      BIS #100,RCS(R5) ; NOW, SET RCVR INTERRUPT ENABLE.
2298 012046 000240      240 ; INTERRUPT SHOULD BE HELD OFF...
2299 012050 000411      BR 3$ ; ...BR IF SO.
2300 012052 022626      2$: CMP (SP)+,(SP)+ ; IT WASN'T...
2301 012054 012737 047225 044746      MOV #SLI2,EMADR ; SET ERROR MESSAGE...
      012062 012737 045206 044752      MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
      012070 000240      NOP ; ...ERROR WILL BE CALLED LATER.
2302 012072 000436      BR 6$ ; GET OUT "INT LEVEL INCORRECT".
2303
2304 012074 012764 012130 000000      3$: MOV #4$,RVEC(R4) ; CHANGE VECTOR...
2305 012102 106427 000000      MTPS #PRO ; ...AND LOWER CPU PRIORITY.
2306 012106 000240      240 ; INTERRUPT SHOULD COME IN.
2307 012110 012737 047265 044746      MOV #SLI3,EMADR ; SET ERROR MESSAGE...
      012116 012737 045206 044752      MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
      012124 000240      NOP ; ...ERROR WILL BE CALLED LATER.
2308 012126 000420      BR 6$ ; GET OUT "INT NOT RECEIVED".
2309
2310 012130 012764 012150 000000      4$: MOV #5$,RVEC(R4) ; INT RECEIVED, CHANGE THE VECTOR.
2311 012136 012716 012144      MOV #.+6,(SP)
2312 012142 000002      RTI ; RESTORE CPU PRI 0...
2313 012144 000240      240 ; SHOULD NOT GET ANOTHER...
2314 012146 000413      BR 7$ ; ...OK, EXIT.
2315 012150 022626      5$: CMP (SP)+,(SP)+ ; DOUBLE RCVR INTERRUPT.
2316 012152 012737 047314 044746      MOV #SLI4,EMADR ; SET ERROR MESSAGE...
      012160 012737 045206 044752      MOV #NOSIG,ESADR ; ...AND SIGNATURE POINTERS...
      012166 000240      NOP ; ...ERROR WILL BE CALLED LATER.
2317
2318 012170 062766 000002 000004      6$: ADD #2,4(SP) ; ERROR -- TAKE SKIP RETURN.
2319 012176 042765 000100 000000      7$: BIC #100,RCS(R5) ; CLEAR THE INT ENABLE.
2320 012204 116500 000002      MOV#B RBUF(R5),R0 ; LOWER RCVR DONE FLAG.
2321 012210 012664 000002      MOV (SP)+,RPRI(R4) ; RESET VECTOR...
2322 012214 012664 000000      MOV (SP)+,RVEC(R4)
2323 012220 000207      RETURN ; RETURN FROM RECEIVER INTERRUPT TESTS
2324

```

```

2325
2326 ; SLIDE A BIT THRU THE INTERNAL (MAINT) LOOP.
2327 ;
2328 ; NOTE: IF THERE'S A TERMINAL ATTACHED, YOU'LL SEE GARBAGE
2329 ; CHARACTERS ON THE SCREEN - DON'T WORRY ABOUT IT !!
2330 ;
2331 012222 005037 044742 DC.DAT: CLR EMPRE ; NO PREFIX.
2332 012226 105237 037625 INCB VARC ; BUMP THE VARIABLE CHARACTER...
2333 012232 123727 037625 000177 CMPB VARC,#177 ;...TIL IT REACHES <DEL>...
2334 012240 103403 BLO 1$ ;...THEN...
2335 012242 112737 000040 037625 MOVB #40,VARC ;...RESET IT TO <SPACE>.
2336 012250 052765 000004 000004 1$: BIS #4,TCS(R5) ; SET XMIT MAINT BIT.
2337 012256 012700 037600 MOV #FLT10,R0 ; DATA TABLE POINTER => RO...
2338 012262 112001 MOVB (R0)+,R1 ;...1ST CHAR => R1...
2339 012264 105737 177564 16$: TSTB @#177564 ;
2340 012270 100375 BPL 16$ ;
2341 012272 105037 177566 CLRB @#177566 ;FLUSH OUT ANY ERRORS
2342 012276 105737 177560 15$: TSTB @#177560 ;
2343 012302 100375 BPL 15$ ;
2344 012304 105737 177562 17$: TSTB @#177562 ;CLEAR OUT RECEIVER BUFFER
2345 012310 001372 BNE 15$ ;
2346 012312 005737 177562 TST @#177562 ;
2347 012316 100762 BMI 16$ ;ANY ERRORS IN RX BUFFER?
2348 012320 105737 177560 TSTB @#177560 ;
2349 012324 100767 BMI 17$ ;IS BUFFER EMPTY NOW?
2350 012326 005002 CLR R2 ;...AND KEEP-ALIVE TIMER => R2.
2351 012330 110165 000006 10$: MOVB R1,TBUF(R5) ; XMIT 1ST CHAR <CR>...
2352 012334 105765 000000 2$: TSTB RCS(R5) ;...AND WAIT FOR RCVR DONE.
2353 012340 100406 BMI 6$ ; OK, CONTINUE AT 6$.
2354 012342 077204 SOB R2,2$ ; DON'T WANT TO HANG !!!
2355 012344 000413 BR 7$ ; LOOP'S OPEN, QUIT (WHILE YOU'RE AHEAD).
2356
2357 012346 112001 4$: MOVB (R0)+,R1 ; NOW FLOAT DATA THRU THE LOOP...
2358 012350 001426 BEQ 8$ ;...UNTIL DONE.
2359 012352 010165 000006 5$: MOV R1,TBUF(R5) ; XMIT A BYTE...
2360 012356 105765 000000 6$: TSTB RCS(R5) ;
2361 012362 100375 BPL 6$ ;
2362 012364 016502 000002 MOV RBUF(R5),R2 ;...AND READ IT BACK.
2363 012370 120102 CMPB R1,R2 ;
2364 012372 001765 BEQ 4$ ; LOOP IF OK.
2365
2366 012374 010137 001124 7$: MOV R1,#GDOAT ; DATA ERROR, SAVE EXP'D...
2367 012400 010237 001126 MOV R2,#BDDAT ;...REC'D...
2368 012404 012737 047356 044746 MOV #SLI5,EMADR ;: SET ERROR MESSAGE...
012412 012737 045326 044752 MOV #EF4,ESADR ;...AND SIGNATURE POINTERS...
012420 000240 NOP ;...ERROR WILL BE CALLED LATER.
2369 012422 062716 000002 ADD #2,(SP) ;...AND TAKE SKIP RETURN.
2370
2371 012426 042765 000004 000004 8$: BIC #4,TCS(R5) ; DONE, CLEAR MAINT SWITCH...
2372 012434 000207 RETURN ;...AND RETURN.
2373 012436 200 114 117 CANT: .ASCIZ <CRLF>\LOOP ON CONSOLE TEST NOT PERMITTED\CRLF>
2374 .EVEN
2375 012504 ENDSL: BR TST5 ;:
012504 000400

```

C6

2377

```
*****
*TEST 5 LINE CLOCK (BEVNT) INTERRUPT
*****
```

012506 000004
012510 012706 001100
012514 012737 177777 002426
012522 005037 044742
012526 000402
012530 000137 013026
012534 104415 012542
012540 000420
012542 040 040 114

012602

2378
2379
2380
2381
2382
2383 012602 013705 001332
2384 012606 012737 012616 000100
2385 012614 000402
2386
2387 012616 005201
2388 012620 000002
2389
2390 012622
012622 012737 012630 001110
2391 012630 106427 000300
2392 012634 005000
2393 012636 005001
2394 012640 052715 000100
2395 012644 000240
2396 012646 102450
2397 012650 077001
2398 012652 005701
2399 012654 001407
2400 012656 012737 046337 044746
012664 012737 045206 044752
012672 104072
2401 012674 106427 000240
2402 012700 077001
2403 012702 005301
2404 012704 003010
2405 012706 012737 046405 044746
012714 012737 045206 044752
012722 104073
2406 012724 000404
2407
2408 012726 012701 177777
2409 012732 005701
2410 012734 100776
2411 012736 042715 000100
2412 012742 077001
2413 012744 005701
2414 012746 001421
2415 012750 012737 046442 044746

```
TST5: SCOPE
MOV #STACK,SP ; SET A CLEAN STACK.
MOV #-1,TRAP4X ; DISMISS BUS ERRORS.
CLR EMPRE ; CLEAR ERROR PREFIX.
BR 30039$ ; SKIP NEXT.
30038$: JMP TST6 ; BYPASS.
30039$: T$NAME, .+4 ; PRINT TEST NUMBER AND NAME...
BR 30040$ ; ...AND SKIP OVER THE ASCII.
.ASCIZ \ LINE CLOCK (BEVNT) INTERRUPT\
.EVEN
30040$:
;
; LINE CLOCK IS JUMPERED FROM CONSOLE DLART OR Q-BUS BEVNT.
; VERIFY THAT WE CAN ENABLE/DISABLE THE CLOCK ($CLK<6>),
; AND THAT INTERRUPTS OCCUR AT BR LEVEL 6, THRU VECTOR 100.
;
MOV $CLK,R5
MOV #1$,BEVNT ; SET VECTOR.
BR 10$
;
1$: INC R1 ; ON BEVNT, TICK R1 AND RETURN.
RTI
;
10$: MOV #.+6,$LPERR ; LOOP HERE ON ERROR.
MTPS #PR6 ; SET CPU PRI 06.
CLR R0 ; CLEAR LOOP CONTROL...
CLR R1 ; ...AND TICKER.
BIS #BIT6,(R5) ; TURN ON CLOCK INTERRUPT.
NOP
BVS 5$ ; ERROR IF NXM.
SOB R0,, ; DELAY ABOUT 250 MSEC, INTERRUPT...
TST R1 ; ... SHOULD BE MASKED AT THIS LEVEL.
BEQ 2$ ; BR IF SO.
MOV #CLK2,EMADR ; \
MOV #NOSIG,ESADR ; >> ERROR 72, CLOCK INTERRUPT LEVEL INCORRECT
; \
; \
2$: MTPS #PR5 ; NOW, LOWER CPU TO PRI 5.
SOB R0,, ; DELAY AGAIN, CLOCK SHOULD COME IN...
DEC R1 ; ... AND GIVE US A FEW TICKS (MORE THAN 1).
BGT 3$ ; BR IF SO.
MOV #CLK3,EMADR ; \
MOV #NOSIG,ESADR ; >> ERROR 73, CLOCK DIDN'T INTERRUPT
; \
; \
3$: BR 4$ ; SKIP OVER THE SYNC CODE.
;
4$: MOV #-1,R1 ; NOW SYNC-UP ON THE NEXT TICK...
TST R1
BMI .-2
BIC #BIT6,(R5) ; ... AND TURN IT OFF.
SOB R0,, ; DELAY ONCE MORE...
TST R1 ; ... CLOCK SHOULD BE SHUT OFF.
BEQ 6$ ; BR IF SO.
MOV #CLK4,EMADR ; \
```

D6

```

012756 012737 045206 044752      MOV    #NOSIG,ESADR    ;; >> ERROR 74, CAN'T SHUT IT OFF
012764 104074      ERROR-74              ;;//
2416 012766 000411      BR     6$
2417
2418 012770 010537 001120      5$:   MOV    R5,$GDADR      ; NXM
2419 012774 012737 046303 044746      MOV    #CLK1,EMADR    ;;\
013002 012737 045230 044752      MOV    #LA16,ESADR    ;; >> ERROR 75, BUS ERROR ON CLOCK ADDRESS
013010 104075      ERROR-75              ;;//
2420 013012 012737 002542 000100      6$:   MOV    #DISMISS,BEVNT ; RESET CLOCK VECTOR.
2421 013020 106427 000000      MTPS   #PRO           ; LOWER CPU TO 0...
2422 013024 000400      BR     TST6           ; ...AND FALL THRU
2424
;*****
;*TEST 6 DMA FACILITY AMZ8016
;*****
013026 000004      TST6: SCOPE
013030 032737 000340 177524      BIT    #340,@#K2CSRC  ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
013036 001417      BEQ    30041$         ;; SKIP THIS TEST IF IT IS
013040 012706 001100      MOV    #STACK,SP     ;; SET A CLEAN STACK.
013044 012737 177777 002426      MOV    #-1,TRAP4X    ;; DISMISS BUS-ERRORS.
013052 005037 044742      CLR    EMPRE         ;; CLEAR ERROR PREFIX.
013056 032777 000400 166054      BIT    #BIT8,@SWR    ;; FORCED ENTRY SELECTED ??
013064 001006      BNE    30042$         ;; BR IF SO.
013066 005777 166232      TST    @DMA          ;; CHECK FOR VALID $DMA.
013072 000240      NOP
013074 102002      BVC    30042$         ;; SKIP NEXT IF OK.
013076 000137 015756 30041$: JMP    TST7           ;; BYPASS.
013102 104415 013110 30042$: T$NAME, .+4         ;; PRINT TEST NUMBER AND NAME...
013106 000414      BR     30043$         ; ...AND SKIP OVER THE ASCII.
013110 040 040 104      .ASCIZ \ DMA FACILITY AMZ8016\
.EVEN
013140      30043$:
2425 ;
2426 ; AMZ8016 DMA CONTROLLER HAS 46. REGISTERS.
2427 ; RESET THE CHIP AND VERIFY THAT ALL REGISTERS RESPOND.
2428 ;
2429 MMRDEF= 135 ; MMR default setting <block lower level hardware
2430 SSRCH1= 102 ;Set Software Request CH 1
2431 SSRCH2= 103 ;Set Software Request CH 2
2432 SCCCH1= 240 ;Start Chain Command CH 1
2433 SCCCH2= 241 ;Start Chain Command CH 2
2434 ; ...interrupts, not hog mode>
2435 TC= BIT0 ; TERMINAL COUNT STATUS BIT.
2436 EOP= BIT1 ; END-OF-PROCESS (NXM) STATUS BIT.
2437 IP= BIT13 ; INTERRUPT POSTED (DONE) STATUS BIT.
2438
2439 013140 TDMA1:
013140 012737 013146 001110      MOV    #.+6,$LPERR   ;; LOOP HERE ON ERROR.
2440 013146 106427 000200      MTPS   #PR4          ; RAISE TO CHIP LEVEL.
2441 013152 012704 000056      MOV    #46.,R4       ; Init R4 with number of registers
2442 013156 013705 001324      MOV    $DMA,R5       ; BASE ADDRESS.
2443 013162 005037 174454      CLR    @#CMDR        ; Issue "RESET" command...
2444 013166 000240      NOP
2445 013170 005025      1$:   CLR    (R5)+         ; ...AND CLEAR ALL REGISTERS.
2446 013172 000240      NOP
2447 013174 102404      BVS    2$            ; V bit set by trap thru 4 routine
2448 013176 077404      SOB    R4,1$         ; Loop control
2449 013200 005037 002426      CLR    TRAP4X        ; ...AND TRAPPER...

```

E6

KXJ11 CA FUNCTIONAL TEST
T6 DMA FACILITY AMZ8016

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 68

```

2450 013204 000413
2451 013206 010537 001120
2452 013212 012737 020042 044746
      013220 012737 045222 044752
      013226 104076
2453 013230 000137 015742
2454
2455
2456
2457
2458
2459 013234
      013234 012737 013242 001110
2460 013242 012700 004000
2461 013246 012701 061022
2462 013252 010021
2463 013254 077002
2464 013256 012737 013434 000214
2465
2466 013264 012737 000115 174470
2467 013272 005037 174446
2468 013276 012737 014042 174442
2469 013304 005037 174444
2470 013310 012737 014042 174440
2471 013316 012737 013746 001412
2472 013324 112700 000061
2473 013330
2475 013330 110037 020107
      013334 110037 020140
      013340 110037 020175
      013344 110037 020226
      013350 110037 020257
2476 013354 012704 013772
2477 013360 013703 001412
2478 013364 012437 044742
2479 013370 012700 177777
2480 013374 000230
2481 013376 012337 174454
2482 013402 012337 174454
2483 013406 077001
2484
2485 013410 012737 020413 044746
      013416 012737 045206 044752
      013424 104077
2486 013426 024646
2487 013430 000137 000214
2488
2489 013434 013700 174456
2490 013440 000402
2491 013442 013700 174454
2492 013446 022626
2493 013450 032700 000001
2494 013454 001022
2495 013456 032700 000002
2496 013462 001010
2497 013464 012737 020451 044746
      013472 012737 045206 044752

      BR      TDMA2          ;...AND CONTINUE.
      MOV     R5,$GDADR      ; BAD ADDRESS (+2).
      MOV     #DMA1,EMADR    ;:\
      MOV     #LA16M2,ESADR ;: >> ERROR 76, BUS ERROR AT DMA CHIP ADDRESS.
      ERROR+76 ;://
      JMP     DMAEND        ; GET OUT.

; EXECUTE 5 PASS DMA DATA TEST AS FOLLOWS:
; 1. LOCAL => LOCAL+2K, 2KW.
; 2. LOCAL I/O PAGE => LOCAL, 1KW.

TDMA2:
      MOV     #.+6,$LPERR    ;: LOOP HERE ON ERROR.
      MOV     #2048.,R0      ;: INIT MEMORY WITH KNOWN DATA
      MOV     #BUFR1,R1
      MOV     R0,(R1)+
      SOB     R0,100$
      MOV     #21$,@#DMAV    ; SERVICE CHAN 1 FIRST
; SET COMMANDS FOR CHAN 1.
      MOV     #115,@#M1R    ; SET MASTER MODE <VI!WAIT!CPINTLV!ENAB>.
      CLR     @#CHA1H        ; LOCAL ACCESS
      MOV     @#CHAIN1,@#CHA1L ; SET CHAIN ADDRESS(ES)...
      CLR     @#CHA2H        ; LOCAL ACCESS
      MOV     @#CHAIN1,@#CHA2L ; ... = 1ST LINK.
      MOV     #8$,@#TSTLOC   ; POINTER TO CHANNEL 1 COMMANDS
      MOVB    #'1,R0        ; STUFF ASCII 1 INTO MESSAGES

1$:
      MOVB    R0,DMAC1+5
      MOVB    R0,DMAC2+5
      MOVB    R0,DMAC3+5
      MOVB    R0,DMAC4+5
      MOVB    R0,DMAC5+5
      MOV     #CKTBL,R4     ; SET CHECK TABLE POINTER IN R4.
2$:
      MOV     @#TSTLOC,R3
      MOV     (R4)+,EMPRE
      MOV     #-1,R0
      SPL     0
      MOV     (R3)+,@#CMDR   ; ISSUE SET INTERRUPT ENABLE COMMAND CHAN1(2)...
      MOV     (R3)+,@#CMDR   ; ...THEN ISSUE START CHAIN CMND CHAN 1(2).
      SOB     R0,,          ; TC (OR EOP) INTERRUPT SHOULD...
      ;...HAPPEN BEFORE THIS TIMES OUT.
      MOV     #DMA2,EMADR    ;:\
      MOV     #NOSIG,ESADR   ;: >> ERROR 77, CHAN INTERRUPT NOT RECEIVED
      ERROR+77 ;://
      CMP     -(SP),-(SP)    ; FAKE THE INTERRUPT...
      JMP     @#DMAV        ; ...AND PROCEED.

21$:
      MOV     @#STAT1,R0     ; GET CHAN 1 STATUS.
      BR     30$            ; OR
22$:
      MOV     @#STAT2,R0     ; GET CHAN 2 STATUS.
30$:
      CMP     (SP)+,(SP)+    ; ADJUST STACK.
      BIT     #TC,R0        ; DID DMA TRANSFER COMPLETE?
      BNE    24$            ; BR IF TC IS SET.
      BIT     #EOP,R0       ; OR DID WE HAVE AN NXM?
      BNE    23$            ; BR IF EOP (NXM) IS SET.
      MOV     #DMA3,EMADR    ;:\
      MOV     #NOSIG,ESADR   ;: >> ERROR 100, DMA INCOMPLETE, TC!EOP BOTH CLEAR.

```

F6

KXJ11 CA FUNCTIONAL TEST
T6 DMA FACILITY AMZ8016

MACRO Y05.02 Thursday 03-Apr-86 14:11 Page 69

```

013500 104100
2498 013502 000407
2499 013504
013504 012737 020513 044746
013512 012737 045206 044752
013520 104101
2500 013522 012337 174454
2501
2502
2503
2504 013526 012400
2505 013530 001466
2506 013532 012401
2507 013534 012402
2508 013536 020427 014032
2509 013542 003434
2510
2511
2512
2513 013544 012737 021600 172352
2514 013552 052737 000020 172516
2515 013560 052737 000001 177572
2516 013566 012737 000020 172516
2517 013574 010037 001120
2518 013600 112037 001124
2519 013604 010137 001122
2520 013610 112137 001126
2521 013614 123737 001124 001126
2522 013622 001020
2523 013624 077215
2524 013626 005037 177572
2525 013632 000425
2526 013634
2527 013634 010037 001120
2528 013640 012037 001124
2529 013644 010137 001122
2530 013650 012137 001126
2531 013654 023737 001124 001126
2532 013662 001410
2533 013664
013664 012737 020562 044746
013672 012737 045360 044752
013700 104102
2534 013702 000401
2535 013704 077225
2536
2537 013706 020427 014042
2538 013712 103622
2539 013714 022703 013762
2540 013720 001420
2541 013722 012737 013442 000214
2542 013730 012737 013754 001412
2543 013736 112700 000062
2544 013742 000137 013330
2545 013746
2546 013746 000062
2547 013750 000240

ERROR-100
BR 24$
23$: MOV #DMA4,EMADR
MOV #NOSIG,ESADR
ERROR-101
24$: MOV (R3)+,a@CHDR
; Now check and see that the transfer occurred correctly
3$: MOV (R4)+,R0
BEQ 6$
MOV (R4)+,R1
MOV (R4)+,R2
CMP R4,#CKBYTE
BLE 4$
; check results of byte transfer from I/O page
MOV #21600,a@KIPARS
BIS #20,a@172516
BIS #1,a@MMRO
MOV #20,a@172516
35$: MOV RO,$GDADR
MOVB (RO)+,$GDDAT
MOV R1,$BDADR
MOVB (R1)+,$BDDAT
CMPB $GDDAT,$BDDAT
BNE 33$
34$: SOB R2,35$
CLR a@MMRO
BR 6$
4$: ; Check results of word transfers
MOV RO,$GDADR
MOV (RO)+,$GDDAT
MOV R1,$BDADR
MOV (R1)+,$BDDAT
CMPB $GDDAT,$BDDAT
BEQ 5$
33$: MOV #DMA5,EMADR
MOV #EF6,ESADR
ERROR-102
BR 6$
5$: SOB R2,4$
6$: CMP R4,#CHAIN1
LO 2$
CMP #7$,R3
BEQ 7$
MOV #22$,DMAV
MOV #9$,a@TSTLOC
MOVB #2,R0
JMP 1$
8$: .WORD 62
.WORD 240

```

G6

KXJ11 CA FUNCTIONAL TEST
T6 DMA FACILITY AMZ8016

MACRO Y05.02 Thursday 03-Apr 86 14:11 Page 70

```
2548 013752 000054          .WORD 54          ; CLEAR IUS AND IP COMMAND.
2549 013754          9$: .WORD 63          ; CHANNEL _NTERRUPT ENABLE
2550 013754 000063          .WORD 241         ; CHANNEL 2 START CHAIN COMMAND
2551 013756 000241          .WORD 55          ; CLEAR INTERRUPT UNDER SERVICE & ...
2552 013760 000055          .WORD 55          ; ...INTERRUPT PENDING CHANNEL 2
2553
2554
2555 013762 005037 174454    7$: CLR @#CMDR     ; ALL DONE RESET CHIP...
2556 013766 000137 014204    JMP DTCEXP        ;
```

H6

```

2558
2559
2560
2561 013772 020102 061022 071022 CKTBL: DMAC1, BUFR1, BUFR2, 2048. ; TEXT, SRC, DST, WC.
2562 014002 020133 177750 071022 DMAC2, 177750, BUFR2, 1
2563 014012 020170 000000 DMAC3, 0 ; CHAINS 3, 4, 5...
2564 014016 020221 000000 DMAC4, 0 ;... INVOLVE Q-BUS SPACE...
2565 014022 020252 061022 071022 DMAC5, BUFR1, BUFR2, 2048. ;... POSTPONE DATA CHECK UNTIL...
2566 ;... CHAIN 5 IS DONE.
2567 014032 020102 120000 071022 CKBYTE: DMAC1, 120000, BUFR2, 4096. ; CHECK BYTE TRANSFER
2568
2569 ; AND THIS IS THE 5 LINK DMA CHAIN.
2570
2571 014042 001606 CHAIN1: 1606 ; LOAD ARA, ARB, OPK, VECT, AND CHAN MODE.
2572 014044 000000 061022 0, BUFR1 ; FROM LOCAL BUFR1...
2573 014050 000000 071022 0, BUFR2 ;... TO LOCAL BUFR2
2574 014054 004000 2048. ;... 2K WORDS.
2575 014056 000214 DMAV ; VECTOR.
2576 014060 000030 001340 30, 1340 ; SFTREQ, HDMSK, INT TC!EOP, INTLV, WORD-WORD..
2577 ; ...FLOWTHROUGH
2578
2579 014064 001602 CHAIN2: 1602 ; RE-LOAD ARA, ARB, OPK, AND CHAN MODE ONLY.
2580 014066 040000 177750 40000, 177750 ; FROM LOCAL I/O (K2 MAINTENANCE REGISTER)
2581 014072 000000 071022 0, BUFR2 ;... TO BUFR2...
2582 014076 000001 1 ;... 1 WORD...
2583 014100 000030 001340 30, 1340 ;... SAME MODE AND TERMINATION.
2584
2585 014104 001602 CHAIN3: 1602 ; DITTO.
2586 014106 000000 061022 0, BUFR1 ; FROM LOCAL BUFR1...
2587 014112 100000 071022 100000, BUFR2 ;... TO Q-BUS BUFR2
2588 014116 004000 2048. ;... 2K WORDS.
2589 014120 000030 001340 30, 1340 ; DITTO.
2590
2591 014124 001602 CHAIN4: 1602 ; DITTO.
2592 014126 100000 071022 100000, BUFR2 ; FROM Q-BUS BUFR2
2593 014132 100000 061022 100000, BUFR1 ;... TO Q-BUS BUFR1
2594 014136 004000 2048. ;... 2K WORDS.
2595 014140 000030 001340 30, 1340 ; DITTO.
2596
2597 014144 001602 CHAIN5: 1602 ; DITTO.
2598 014146 100000 061022 100000, BUFR1 ; FROM Q-BUS BUFR1
2599 014152 000000 071022 0, BUFR2 ;... TO LOCAL BUFR2
2600 014156 004000 2048. ;... 2K WORDS.
2601 014160 000030 001340 30, 1340 ; DITTO.
2602
2603 014164 001602 CHAIN6: <CARA!CARB!COP!MODE>
2604 014166 004000 160000 <K2MEM!4000!UP>, 160000 ; FROM K2 ROM
2605 014172 000000 071022 <K2MEM!UP>, BUFR2 ; TO BUFFER 2
2606 014176 010000 4096. ; 4 KB
2607 014200 000030 001301 <SWRQ!HM>, <ITC!IEOP!BUSREL!TRBB> ; INTERRUPT ON TC OR EOP,..
2608 ; BUS RELEASE, BYTE-BYTE TRANSFER

```



```

2610 014204          ;DTCXP: ;NOW TEST THAT THE DTC EXCEPTIONS WORK CORRECTLY
2611 014204 012737 014400 000214  MOV    #21$,@#DMAV      ; SERVICE CHAN 1 FIRST
2612                ; SET COMMANDS FOR CHAN 1.
2613 014212 012737 000115 174470  MOV    #115,@#MMR      ; SET MASTER MODE <VI!WAIT!CPINTLV!ENAB>.
2614 014220 005037 174446          CLR    @#CHA1H          ;
2615 014224 012737 014754 174442  MOV    #DTCX01,@#CHA1L ; SET CHAIN ADDRESS(ES)...
2616 014232 005037 174444          CLR    @#CHA2H          ; LOCAL ACCESS
2617 014236 012737 014754 174440  MOV    #DTCX01,@#CHA2L ;... = 1ST LINK.
2618 014244 012737 014624 001412  MOV    #8$,@#TSTLOC    ; POINTER TO CHANNEL 1 COMMANDS
2619 014252 012737 014640 001414  MOV    #10$,@#TSTLOC+2 ; POINTER TO CHANNEL 1 REGISTERS
2620 014260 012700 000061          MOV    #'1,R0          ;
2621 014264                1$:
2623 014264 110037 015221          MOVB   R0,DTCXM1+5
      014270 110037 015271          MOVB   R0,DTCXM2+5
      014274 110037 015341          MOVB   R0,DTCXM3+5
      014300 110037 015410          MOVB   R0,DTCXM4+5
      014304 110037 015457          MOVB   R0,DTCXM8+5
      014310 110037 015537          MOVB   R0,DTCXM9+5
2624 014314 012704 014654          MOV    #EOP,CL,R4      ; SET CHECK TABLE POINTER IN R4.
2625 014320 013703 001412          2$: MOV    @#TSTLOC,R3      ; LOAD COMMAND POINTER
2626 014324 013702 001414          MOV    @#TSTLOC+2,R2  ; LOAD REGISTER POINTER
2627 014330 012437 044742          MOV    (R4)+,EMPRE    ; SET ERROR PREFIX.
2628 014334 012700 177777          MOV    #-1,R0         ; KEEP-ALIVE TIMER.
2629 014340 000230          SPL    0              ; LOWER CPU.
2630 014342 012337 174454          MOV    (R3)+,@#CMDR   ; ISSUE SET INTERRUPT ENABLE COMMAND CHAN1(2)...
2631 014346 012337 174454          MOV    (R3)+,@#CMDR   ; ...THEN ISSUE START CHAIN CMND CHAN 1(2).
2632 014352 077001          SOB    R0              ; EOP INTERRUPT SHOULD...
2633                ; ...HAPPEN BEFORE THIS TIMES OUT.
2634 014354 012737 020413 044746  MOV    #DMA2,EMADR    ;:\
      014362 012737 045206 044752  MOV    #NOSIG,ESADR   ;: >> ERROR 103, CHAN INTERRUPT NOT RECEIVED
      014370 104103          ERROR+103           ;://
2635 014372 024646          CMP    -(SP),-(SP)    ; FAKE THE INTERRUPT...
2636 014374 000137 000214          JMP    @#DMAV          ; ...AND PROCEED.
2637 014400 013700 174456          21$: MOV    @#STAT1,R0     ; GET CHAN 1 STATUS.
2638 014404 000402          BR    30$             ; OR
2639 014406 013700 174454          22$: MOV    @#STAT2,R0     ; GET CHAN 2 STATUS.
2640 014412 022626          30$: CMP    (SP)+,(SP)+  ; ADJUST STACK.
2641 014414 032700 000002          BIT    #EOP,R0        ; DID WE HAVE AN NXM?
2642 014420 001010          BNE   23$             ; BR IF EOP (NXM) IS SET.
2643 014422 012737 015612 044746  MOV    #DTCXM5,EMADR  ;:\
      014430 012737 045206 044752  MOV    #NOSIG,ESADR   ;: >> ERROR 104, EXPECTED NXM, EOP CLEAR.
      014436 104104          ERROR+104           ;://
2644 014440 000412          BR    24$             ;
2645 014442 032700 000001          23$: BIT    5,R0         ; DID DMA TRANSFER COMPLETE?
2646 014446 001407          BEQ   24$             ; BR IF TC IS NOT SET.
2647 014450 012737 015612 044746  MOV    #DTCXM5,EMADR  ;:\
      014456 012737 045206 044752  MOV    #NOSIG,ESADR   ;: >> ERROR 105, TC SHOULD NOT BE SET
      014464 104105          ERROR+105           ;://
2648 014466 012337 174454          24$: MOV    (R3)+,@#CMDR   ; INT SERVICE COMPLETE, CLEAR "IUS!IP".
2649 014472 012437 001120          MOV    (R4)+,$GDADR   ; IS THERE DATA TO CHECK?
2650 014476 001424          BEQ   25$             ;
2651 014500 012437 001122          MOV    (R4)+,$BDADR   ;
2652 014504 012437 001124          MOV    (R4)+,$GDDAT   ;
2653 014510 023237 001120          CMP    @#(R2)+,$GDADR ; IS OFFSET REGISTER AS EXPECTED?
2654 014514 001006          BNE   26$             ;
2655 014516 023237 001122          CMP    @#(R2)+,$BDADR ; IS B ADDRESS OFFSET REGISTER AS EXPECTED?
2656 014522 001003          BNE   26$             ;

```

```

2657 014524 023237 001124      CMP      @R2+,$GDDAT ;IS OP COUNT AS EXPECTED?
2658 014530 001407      BEQ      25$
2659 014532      26$:      MOV      #DTCXM7,EMADR ;;\
        014532 012737 015700 044746      MOV      #NOSIG,ESADR ;; >> ERROR 106, CONTENTS OF DTC NOT CORRECT AFTER EOP
        014540 012737 045206 044752      ERROR-106 ;;//
        014546 104106
2660 014550 020427 014754      25$:      CMP      R4,#DTCX01 ;
2661 014554 022703 014640      BLT      24$
2662 014556 001004      CMP      #10$,R3 ;HAVE BOTH CHANNELS BEEN TESTED?
2663 014562 005037 174454      BNE      27$
2664 014564 000137 015742      CLR      @#CMDR ;RESET CHIP
2665 014570 012737 014632 001412      JMP      DMAEND
2666 014574 012737 014646 001414      27$:      MOV      #9$,@#TSTLOC ;LOAD CHANNEL 2 COMMAND POINTER
2667 014602 012737 014406 000214      MOV      #11$,@#TSTLOC+2 ;LOAD CHANNEL 2 REGISTER POINTER
2668 014610 012700 000620      MOV      #22$,@#DMAV ;LOAD CHANNEL 2 INTERRUPT HANDLER
2669 014616 000620      MOV      #'2,R0 ;
2670 014622 000620      BR       14$
2671
2672 014624      8$:      ;CHANNEL 1 COMMANDS
2673 014624 000062      .WORD   62 ; SET CIE COMMAND.
2674 014626 000240      .WORD   240 ; START CHAIN COMMAND.
2675 014630 000054      .WORD   54 ; CLEAR IUS AND IP COMMAND.
2676 014632      9$:      ;CHANNEL 2 COMMANDS
2677 014632 000063      .WORD   63 ; CHANNEL 2 INTERRUPT ENABLE
2678 014634 000241      .WORD   241 ; CHANNEL 2 START CHAIN COMMAND
2679 014636 000055      .WORD   55 ; CLEAR INTERRUPT UNDER SERVICE & ...
2680 ; ...INTERRUPT PENDING CHANNEL 2
2681 014640      10$:     ;CHANNEL 1 DTC REGISTERS
2682 014640 174412      .WORD   174412 ;DTC CHANNEL 1 CURRENT ADDRESS REG A
2683 014642 174402      .WORD   174402 ;DTC CHANNEL 1 CURRENT ADDRESS REG B
2684 014644 174462      .WORD   174462 ;DTC CHANNEL 1 CURRENT OP COUNT
2685 014646      11$:     ;CHANNEL 2 DTC REGISTERS
2686 014646 174410      .WORD   174410 ;DTC CHANNEL 2 CURRENT ADDRESS REG A
2687 014650 174400      .WORD   174400 ;DTC CHANNEL 2 CURRENT ADDRESS REG B
2688 014652 174460      .WORD   174460 ;DTC CHANNEL 2 CURRENT OP COUNT
2689
2690 014654 015214 000000      EOPTBL: DTCXM1, 0
2691 014660 015264 000000      DTCXM2, 0
2692 014664 015334 000000      DTCXM3, 0
2693 014670 015403 000000      DTCXM4, 0
2694 014674 015214 061024 170002      DTCXM1, BUFR1+2, 170002, 3
2695 014704 015264 061024 170002      DTCXM2, BUFR1+2, 170002, 3
2696 014714 015334 061024 177572      DTCXM3, BUFR1+2, 177572, 3
2697 014724 015403 061024 177572      DTCXM4, BUFR1+2, 177572, 3
2698 014734 015452 061024 177572      DTCXM8, BUFR1+2, 177572, 3
2699 014744 015532 061024 177572      DTCXM9, BUFR1+2, 177572, 3
2700
2701 014754      DTCX01: ; This transaction will try to transfer to a non-existent local
2702 ; memory location. Expecting an EOP. Check that the Chain Abort and
2703 ; the No Auto-Reload or Chain bits are set.
2704 014754 001762      <CARA!CARB!BARA!BARB!COP!BOP!MODE> ;Chain reload word
2705 014756 000000 061022      0,BUFR1 ;Current Address Register A (dma source)
2706 014762 037420 170000      37420,170000 ;Current Address Register B (dma destination-- hold)
2707 014766 000004      4 ;Current op-count
2708 014770 000000 061022      0,BUFR1 ;Base Address Register A
2709 014774 037420 170000      37420,170000 ;Base Address Register B
2710 015000 000004      4 ;Base op-count

```

K6

KXJ11-CA FUNCTIONAL TEST
T6 DMA FACILITY AMZ8016

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 74

```

2711 015002 000030 003340      30,3340      ;Channel mode register
2712
2713      ; This transaction will try to transfer from a non-existent local
2714      ; memory location.
2715 015006 000002      2      ;Chain reload word
2716 015010 000030 001360      30,1360      ;<SWRQ!HM>,<RELEOP!ITC!IEOP!FLIP>;Channel mode register
2717      ;
2718      ;Try some transfers to the Qbus
2719      ;
2720 015014      CHNQ10: ; This transaction will try to transfer from a non-existent Q bus
2721      ; I/O location.
2722 015014 001762      <CARA!CARB!BARA!BARB!COP!BOP!MODE>      ;Chain reload word
2723 015016 177420 177570      177420,177570 ;Current Address Register A (dma source-- hold)
2724 015022 000000 061022      0,BUFR1      ;Current Address Register B (dma destination)
2725 015026 000004      4      ;Current op-count
2726 015030 177420 177570      177420,177570 ;Base Address Register A (dma source-- hold)
2727 015034 000000 061022      0,BUFR1      ;Base Address Register B
2728 015040 000004      4      ;Base op-count
2729 015042 000030 003340      30,3340      ;Channel mode register
2730
2731      ; This transaction will try to transfer to a non-existent Q bus
2732      ; I/O location.
2733 015046 000002      2      ;Chain reload word
2734 015050 000030 001360      30,1360      ;Channel mode register
2735
2736      ;CHECK THE CONTENTS OF THE CURRENT ADDRESS REGISTERS AND OP COUNT AFTER
2737      ;A TRANSFER THAT RESULTS IN A TIME-OUT
2738
2739 015054 001602      <CARA!CARB!COP!MODE>      ;Chain reload word
2740 015056 000000 061022      0,BUFR1      ;Current Address Register A (dma source)
2741 015062 037400 170000      37400,170000 ;Current Address Register B (dma destination-- hold)
2742 015066 000004      4      ;Current op-count
2743 015070 000030 001340      <SWRQ!HM>,<ITC!IEOP!TRWW!INTLV>      ;channel mode register
2744
2745 015074 001602      <CARA!CARB!COP!MODE>      ;Chain reload word
2746 015076 000000 061022      0,BUFR1      ;Current Address Register A (dma source)
2747 015102 037400 170000      37400,170000 ;Current Address Register B (dma destination-- hold)
2748 015106 000004      4      ;Current op-count
2749 015110 000030 001360      <SWRQ!HM>,<ITC!IEOP!TRWW!INTLV!FLIP> ;channel mode register
2750
2751 015114 001602      <CARA!CARB!COP!MODE>      ;Chain reload word
2752 015116 000000 061022      0,BUFR1      ;Current address register a
2753 015122 177400 177570      <QBUSIO!37400>,177570 ;Current address register b
2754 015126 000004      4      ;Current operation count
2755 015130 000030 001340      <SWRQ!HM>,<ITC!IEOP!TRWW!INTLV>      ;channel mode register
2756
2757 015134 001602      <CARA!CARB!COP!MODE>      ;Chain reload word
2758 015136 000000 061022      0,BUFR1      ;Current address register a
2759 015142 177400 177570      <QBUSIO!37400>,177570 ;Current address register b
2760 015146 000004      4      ;Current operation count
2761 015150 000030 001360      <SWRQ!HM>,<ITC!IEOP!TRWW!INTLV!FLIP> ;channel mode register
2762
2763 015154 001602      <CARA!CARB!COP!MODE>      ;Chain reload word
2764 015156 000000 061022      0,BUFR1      ;
2765 015162 177400 177570      <QBUSIO!37400>,177570 ;
2766 015166 000004      4      ;
2767 015170 000030 001350      <SWRQ!HM>,<ITC!IEOP!TRFUNL!INTLV>

```

L6

```

2768
2769 015174 001602          <CARA!CARB!COP!MODE> ;Chain reload word
2770 015176 000000 061022 0,BUFR1 ;
2771 015202 177400 177570 <QBUSIO!37400>,177570 ;
2772 015206 000004          4
2773 015210 000030 001370 <SWRQ!HM>,<ITC!IEOP!TRFUNL!INTLV!F.L1 >
2774
2775 015214 103 110 101 DTCXM1: .ASCIZ /CHAN X DMA XFER LOCAL MEM => LOCAL NXM/<CRLF>
2776 015264 103 110 101 DTCXM2: .ASCIZ /CHAN X DMA XFER LOCAL NXM => LOCAL MEM/<CRLF>
2777 015334 103 110 101 DTCXM3: .ASCIZ /CHAN X DMA XFER LOCAL MEM => QBUS NXM/<CRLF>
2778 015403 103 110 101 DTCXM4: .ASCIZ /CHAN X DMA XFER QBUS NXM => LOCAL MEM/<CRLF>
2779 015452 103 110 101 DTCXM8: .ASCIZ /CHAN X DMA FUNNEL XFER - LOCAL MEM => QBUS NXM/<CRLF>
2780 015532 103 110 101 DTCXM9: .ASCIZ /CHAN X DMA FUNNEL XFER - QBUS NXM => LOCAL MEM/<CRLF>
2781 015612 040 105 130 DTCXM5: .ASCIZ / EXPECTED EOP ON DMA TRANSFER/<CRLF>
2782 015651 040 105 130 DTCXM6: .ASCIZ / EXPECTED CHAIN ABORT/<CRLF>
2783 015700 103 117 - 116 DTCXM7: .ASCIZ /CONTENTS OF DTC NOT AS EXPECTED/<CRLF>
2784 .EVEN
2785 015742 012737 002542 000214 DMAEND: MOV #DISMISS,DMAV ; RESTORE VECTOR.
2786 015750 005037 174454 CLR @#CMDR ;RESET DTC CHIP
2787 015754 000400 BR TST7 ;;NEXT TEST.

```

M6

2789

015756	000004		
015760	032737	000340	177524
015766	001410		
015770	012706	001100	
015774	012737	177777	002426
016002	005037	044742	
016006	000402		
016010	000137	022134	
016014	104415	016022	
016020	000411		
016022	040	040	121

016044

2790
2791
2792
2793
2794
2795
2796
2797
2798
2799 016044 013701 177522
2800 016050 032701 000004
2801 016054 001415
2802
2803 016056 032701 000002
2804 016062 001405
2805
2806 016064
2807
2808 016064 012704 000026
2809 016070 012703 000006
2810 016074 000416
2811 016076
2812
2813 016076 012704 000022
2814 016102 012703 000002
2815 016106 000411
2816 016110
2817 016110 032701 000002
2818 016114 001763
2819
2820
2821 016116 012704 000020
2822 016122 020437 050736
2823 016126 001413
2824 016130 000403
2825 016132 020437 050736
2826 016136 001411
2827 016140
016140 012737 016604 044746
016146 012737 016536 044752
016154 104107

```

;*****
;*TEST 7      Q BUS INTERFACE
;*****
TST7:  SCOPE
      BIT      #340,#K2CSRC      ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
      BEQ      30044$           ;; SKIP THIS TEST IF IT IS
      MOV      #STACK,SP        ;; SET A CLEAN STACK.
      MOV      #1,TRAP4X        ;; DISMISS BUS-ERRORS.
      CLR      EMPRE            ;; CLEAR ERROR PREFIX.
      BR       30045$           ;; SKIP NEXT.
30044$: JMP      TST10          ;; BYPASS.
30045$: T$NAME, .+4            ;; PRINT TEST NUMBER AND NAME...
      BR       30046$           ;;...AND SKIP OVER THE ASCII.
      .ASCIZ   \ Q-BUS INTERFACE\
      .EVEN
30046$:
;
;TEST THE UPPER ADDRESS BITS 16 THRU 21 ON THE QBUS INTERFACE.
;FIRST WRITE TSTLOC IN QBUS MEMORY WITH ALL ONES. THEN ATTEMPT TO
;WRITE ZEROS TO TSTLOC IN QBUS MEMORY, BUT THIS TIME WITH ONE OF
;THE ADDRESS BITS 16 THRU 21 SET. IF TSTLOC IS CLEAR AS A RESULT
;OF THIS WRITE THEN THE ADDRESS BIT DID NOT GET WRITTEN ONTO THE BUS.
;
;FIRST LET'S MAKE SURE THAT WE HAVE MORE THAN 16 QBUS ADDRESS BITS.
;
      MOV      #K2CSRB,R1        ;GET BUS SIZE BITS IN K2CSRB
      BIT      #BIT02,R1        ; IS BIT 02 = 1
      BEQ      104$             ; NO, BRANCH
;COMES HERE IF BIT 02 IS =1
      BIT      #BIT01,R1        ; IS BIT 01 = 1
      BEQ      102$             ; NO, BRANCH
105$:  ;COMES HERE IF BIT 02 =1 AND BIT 01 =1, OR IF BIT 01 =0 AND BIT 02 = 0
;INDICATES A 22 BIT JUMPER CONFIGURATION.
      MOV      #22,,R4          ;THEN IT'S A 22 BITTER
      MOV      #6,R3            ;INIT LOOP COUNTER
      BR       106$
102$:  ;COMES HERE IF BIT 01 = 0 AND BIT 02 = 1.
;INDICATES AN 18 BIT JUMPER CONFIGURATION.
      MOV      #18,,R4
      MOV      #2,R3            ;INIT LOOP COUNTER
      BR       106$
104$:  ;COMES HERE IF BIT 02 = 0
      BIT      #BIT01,R1        ; IS BIT 01 = 1
      BEQ      105$             ; NO, BRANCH
;COMES HERE IF BIT 01 = 1, BIT 02 = 0.
;INDICATES A 16 BIT JUMPER CONFIGURATION
      MOV      #16,,R4
      CMP      R4,,ABUSW        ;DOES IT MATCH WHAT ARBITER SAYS?
      BEQ      100$             ; 16 BIT BACKPLANE.
      BR       103$             ;NO, FLAG ERROR
106$:  CMP      R4,,ABUSW        ;DOES THIS MATCH WHAT ARBITER SAYS?
      BEQ      101$
103$:  MOV      #CNFGER,EMADR    ;;\
      MOV      #CNFG01,ESADR    ;; >> ERROR 107, NUMBER OF ADDRESS BITS CONFIGURED WRONG
      ERROR+107                ;;//

```

```

2828 016156 000137 017106          100$: JMP      FIN22
2829 016162 012704 000020          101$: MOV      #16,R4          ;INIT R4 WITH FIRST BIT TO BE TESTED
2830 016166 012737 177777 001412  MOV      #-1,@#TSTLOC    ;MAKE TSTLOC = -1
2831 016174 005037 001414          CLR      @#TSTLOC+2      ;MAKE TSTLOC+2 = 0
2832 016200 005037 001416          CLR      @#TSTLOC+4      ;MAKE TSTLOC+4 = 0
2833 016204 012702 000400          MOV      #400,R2        ;INIT R2 WITH ADDRESS BIT 16 =1
2834 016210 050237 017044          BIS      R2,AD1621+6     ;LOAD SEGMENT FIELD OF CHAIN
2835 016214 005037 174454          CLR      @#CMDR         ;RESET DTC CHIP
2836 016220 012737 000111 174470  MOV      #111,@#MMR      ;ENABLE DTC. NO CPU INTERLEAVE
2837 016226 005037 174444          CLR      @#CHA2H
2838 016232 012737 017016 174440  MOV      #INIT0,@#CHA2L   ;LOAD CHAIN ADDRESS
2839 016240 012737 000061 174454  MOV      #CLIE2,@#CMDR    ;DON'T ALLOW INTERRUPT FROM DTC
2840 016246 012737 000241 174454  MOV      #SCCCH2,@#CMDR   ;ISSUE START CHAIN COMMAND CH2
2841 016254 013746 000220          MOV      @#DMAV2,-(SP)    ;SAVE VECTORS
2842 016260 013746 000222          MOV      @#DMAV2+2,-(SP) ;
2843 016264 013746 000214          MOV      @#DMAV1,(SP)
2844 016270 013746 000216          MOV      @#DMAV1+2,(SP) ;
2845 016274 012737 016420 000220  MOV      #10,@#DMAV2      ;SETUP INTERRUPT HANDLER FOR DMA
2846 016302 012737 000340 000222  MOV      #PR7,@#DMAV2+2   ;
2847 016310 012737 016370 000214  MOV      #21,@#DMAV1      ;
2848 016316 012737 000340 000216  MOV      #PR7,@#DMAV1+2   ;
2849 016324 012737 017036 174440  MOV      #AD1621,@#CHA2L  ;LOAD CHAIN ADDRESS
2850 016332 012737 000063 174454  MOV      #SETIE2,@#CMDR   ;ENABLE CH2 INTERRUPT
2851 016340 042737 000040 177540  5$: BIC      #BIT05,@#K2CSRJ ;CLEAR SACK TIME-OUT BIT
2852 016346 106427 000000          MTPS     #0              ;SET PRIORITY LEVEL TO ZERO
2853 016352 012737 000241 174454  MOV      #SCCCH2,@#CMDR   ;ISSUE START CHAIN COMMAND CH2
2854 016360 012737 000103 174454  MOV      #SSRCH2,@#CMDR   ;ISSUE SET SOFTWARE REQUEST COMMAND
2855 016366 000001          WAIT                    ;WAIT FOR INTERRUPT
2856
2857          ;
2858          ;COMES HERE AFTER TRANSFER FROM QBUS MEMORY TO TSTLOC+4
2859 016370 022737 177777 001414  21$: CMP      #-1,@#TSTLOC+2 ;IS K2 TSTLOC+2 = -1? IT SHOULD BE.
2860          ; IF NOT THEN EITHER THE ADDRESS
2861          ; BIT IS BAD OR THE BACKPLANE DOES
2862          ; NOT SUPPORT 22 BIT ADDRESSES.
2863 016376 001423          BEQ      25$             ;BRANCH IF OK
2864 016400 012737 016654 044746  MOV      #BADBIT,EMADR    ;;\
016406 012737 016572 044752  MOV      #BADBAL,ESADR    ;; >> ERROR 110, ONE OF THE UPPER ADDRESS BITS 16-21 IS BAD
016414 104110          ERROR+110              ;://
2865 016416 000413          BR       25$             ;BAIL OUT
2866
2867          ;
2868          ;COMES HERE IF EOP (NXM) OCCURS ON TRANSFER TO Q BUS MEMORY
2869 016420 032737 000002 174454  10$: BIT      #BIT01,@#STAT2 ;DID EOP OCCUR?
2870 016426 001007          BNE      25$             ;ERROR IF NO EOP
2871 016430 012737 017004 044746  MOV      #OOPS,EMADR      ;;\
016436 012737 045206 044752  MOV      #NOSIG,ESADR     ;; >> ERROR 111, SHOULD ONLY COME HERE ON EOP
016444 104111          ERROR+111              ;://
2872
2873          ;
2874          ;SET UP FOR NEXT ADDRESS BIT IF ANY
2875 016446 022626          25$: CMP      (SP)+,(SP)+   ;FAKE AN RTI AND FIX THE STACK
2876 016450 005037 001414          CLR      @#TSTLOC+2      ;CLEAR TSTLOC+2
2877 016454 040237 017044          BIC      R2,AD1621+6     ;CLEAN UP
2878 016460 005204          INC      R4              ;ADD 1 TO R4
2879 016462 006302          ASL      R2              ;
2880 016464 050237 017044          BIS      R2,AD1621+6     ;LOAD SEGMENT FIELD

```

B7

```

2881 016470 012737 000055 174454      MOV    @CLIP2,@#CHDR      ;ISSUE CLEAR INTERRUPT COMMAND
2882
2883 016476 077360              SOB    R3,5$             ;DO IT SIX TIMES FOR 22 BITS, TWICE FOR
2884                                ;..EIGHTEEN.
2885 016500 012637 000216      26$:  MOV    (SP)+,DMAV1+2 ;RESTORE THE VECTORS
2886 016504 012637 000214      MOV    (SP)+,DMAV1
2887 016510 012637 000222      MOV    (SP)+,DMAV2+2
2888 016514 012637 000220      MOV    (SP)+,DMAV2
2889 016520 042737 077757 017044      BIC    @+C@QBEM!HOLD>,AD1621+6 ;CLEAN UP
2890 016526 005037 174454      CLR    @#CHDR            ;RESET DTC CHIP
2891 016532 000137 017106      JMP    FIN22             ;SKIP OVER CHAIN TABLES
2892
2893                                ;ERROR ROUTINES
2894
2895 016536 010446              ;CNFG01: MOV    R4,-(SP)    ;PUSH NUMBER OF ADDRESS BITS KXJ IS JUMPERED FOR
2896 016540 104401 016726      TYPE    ,JMP$AY          ;TYPE MESSAGE
2897 016544 104405              TYPDS   ;TYPE DECIMAL NUMBER
2898 016546 104401 016760      TYPE    ,ADRBIT
2899 016552 013746 050736      MOV    ,ABUSW,-(SP)     ;PUSH NUMBER OF ADDRESS BITS ARBITER SUPPORTS
2900 016556 104401 016703      TYPE    ,ARBSUP         ;TYPE MESSAGE
2901 016562 104405              TYPDS   ;TYPE DECIMAL NUMBER
2902 016564 104401 016760      TYPE    ,ADRBIT
2903 016570 000207              RTS     PC
2904
2905 016572 010446              ;BADBAL: MOV    R4,-(SP)  ;PUSH NUMBER OF BAD ADDRESS BIT
2906 016574 104401 016776      TYPE    ,BIT            ;TYPE MESSAGE
2907 016600 104405              TYPDS   ;TYPE DECIMAL NUMBER
2908 016602 000207              RTS     PC
2909
2910                                ;ERROR MESSAGES
2911 016604      116      125      115  CNFGR: .ASCIZ /NUMBER OF ADDRESS BITS CONFIGURED WRONG/
2912 016654      102      101      104  BADBIT: .ASCIZ /BAD Q-BUS ADDRESS LINE/
2913 016703      200      101      122  ARBSUP: .ASCIZ <CRLF>/ARBITER SUPPORTS /
2914 016726      200      113      130  JMP$AY: .ASCIZ <CRLF>/KXJ11-CA CONFIGURED FOR /
2915 016760      040      101      104  ADRBIT: .ASCIZ / ADDRESS BITS/
2916 016776      040      102      111  BIT: .ASCIZ / BIT /
2917 017004      104      124      103  OOPS: .ASCIZ /DTC ERROR/
2918
2919                                .EVEN
2920                                ;DMA CHAIN TABLES FOR TESTING ADDRESS BITS 16-21
2921
2922 017016              ;INITO: ;WRITE Q BUS MEMORY LOCATION WITH -1
2923 017016 001602              <CARA!CARB!COP!MODE> ;CHAIN RELOAD WORD
2924 017020 000020 001412      <K2MEM!HOLD>,TSTLOC ;CARA (SOURCE)
2925 017024 100020 001412      <QBEM!HOLD>,TSTLOC ;CARB (DEST)
2926 017030 000001              1 ;COP
2927 017032 000030 000040      <SWRQ!HM>,<TRWM!BUSHOG> ;MODE
2928
2929 017036              ;AD1621: ;CHAIN TO START TEST OF QBUS BITS 16 THRU 21
2930 017036 001606              <CARA!CARB!COP!IV!MODE> ;CHAIN RELOAD WORD
2931 017040 000020 001416      <K2MEM!HOLD>,TSTLOC+4 ;CARA (SOURCE) WRITE 0
2932 017044 100020              <QBEM!HOLD>
2933 017046 001412              .WORD TSTLOC ;CARB (DEST)
2934 017050 000002              2 ;COP
2935 017052 000220              DMAV2 ;IV
2936 017054 000010 100240      <HM>,<IEOP!CTC!TRWM!BUSHOG> ;MODE-- NEEDS SOFTWARE REQ TO GO
2937

```

C7

KXJ11-CA FUNCTIONAL TEST
T7 Q-BUS INTERFACE

MACRO Y05.02 Thursday 03 Apr-86 14:11 Page 79

2938 017060
2939 017060 001607
2940 017062 100020 001412
2941 017066 000020 001414
2942 017072 000001
2943 017074 000214
2944 017076 000030 001040
2945 017102 000000 017036
2946
2947 017106

READ0: ;READ ADDRESS TSTLOC IN QBUS MEMORY
<CARA!CARB!COP!IV!MODE!CHAD> ;CHAIN RELOAD WORD
<QBEM!HOLD>,TSTLOC ;CARA (SOURCE)
<K2MEM!HOLD>,TSTLOC+2 ;CARB (DEST)
1 ;COP
DMAV1 ;IV
<SWRQ!HM>,<ITC!TRWM!BUSHOG> ;MODE
0,AD1621 ;CHAIN ADDRESS
;FIN22:


```

2949          170000          QBE1=170000
2950          170020          QBE2=170020
2951          ;QBUS EXERCISER TESTS FOR THE K2
2952          ;LOOK FOR 18 AND/OR 22 BIT Q BUS EXERCISER. THE 18 BIT QBE IS EXPECTED TO
2953          ;BE CONFIGURED FOR AN ADDRESS OF 770000. THE 22 BIT QBE IS EXPECTED TO BE
2954          ;BE CONFIGURED FOR AN ADDRESS OF 17770020. BOTH QBES SHOULD BE AT A HIGHER
2955          ;INTERRUPT PRIORITY THAN THE K2 UNDER TEST. IN OTHER WORDS THEY SHOULD BE
2956          ;INSTALLED IN THE BACKPLANE SO THAT THEY ARE BETWEEN THE ARBITER CPU AND THE
2957          ;K2 UNDER TEST.
2958
2959          ;Definitions for QBE CSR1 bits
2960          WRD= 1          ;WORD TRANSFER
2961          BYT= 0          ;BYTE TRANSFER
2962          HD= 0          ;HOLD DATA
2963          RR= 2          ;ROTATE DATA RIGHT
2964          RL= 4          ;ROTATE DATA LEFT
2965          LN= 6          ;LOAD NEW DATA
2966          DR= 0          ;QBE DATA REGISTER IS DATA SOURCE
2967          AD= 20         ;QBE ADDRESS REGISTER IS DATA SOURCE
2968          HOG= 0         ;HOG MODE
2969          B1= 40         ;ONE TRANSFER PER DMG
2970          B2= 100        ;TWO TRANSFERS PER DMG
2971          B4= 140        ;FOUR TRANSFERS PER DMG
2972          DATIO= 200     ;DATIO TRANSFER
2973          DIN= 400       ;DATA IN TO QBE TRANSFER
2974          DOUT= 600      ;DATA OUT FROM QBE TRANSFER
2975
2976          ;
2977          ;QBES CSR2 bit definitions
2978          ;
2979          GO= 1           ;SET GO BIT
2980          IRQ4= 2        ;ENABLE INTERRUPT PRI 4
2981          IRQ5= 4        ;ENABLE INTERRUPT PRI 5
2982          IRQ6= 10       ;ENABLE INTERRUPT PRI 6
2983          IRQ7= 20       ;ENABLE INTERRUPT PRI 7
2984          A16= 40000     ;ASSERT ADDRESS BIT 16
2985          A17= 100000    ;ASSERT ADDRESS BIT 17
2986
2987          ;
2988          ;
2989          ;
2990          ;
2991          ;
2992          ;
2993          ;
2994          ;
2995          ;
2996          ;
2997          ;
2998          ;
2999          ;
3000          ;
3001          ;
3002          ;
3003          ;
3004          ;
3005          ;
3006          ;
3007          ;
3008          ;
3009          ;
3010          ;
3011          ;
3012          ;
3013          ;
3014          ;
3015          ;
3016          ;
3017          ;

```

```

;Command chain to check for QBE at location 170000

```

E7

KXJ11-CA FUNCTIONAL TEST
T7 Q-BUS INTERFACE

MACRO Y05.02 Thursday 03-Apr-86 14:11 Page 81

```

3018 017200 001606          24$: 1606          ;Chain load word
3019 017202 000020 001410    20,SOURCE      ;hold source
3020 017206 177400 170000    177400,QBE1    ;destination is the Q bus exerciser on the LSI bus
3021 017212 000001          1              ;operation count
3022 017214 000214          DMAV           ;interrupt vector
3023 017216 000030 000140    30,140        ;channel mode register
3024
3025 017222 005037 050760    1$: CLR        @#QBE18          ;NO 18 BIT QBE PRESENT
3026
3027
3028 017226 005737 050756    2$: TST        QBE22          ;IS THERE A Q22 BUS EXERCISER PRESENT?
3029 017232 100401          BMI           51$
3030 017234 000436          BR            4$
3031 017236 005000          51$: CLR        R0              ;Init R0
3032 017240 005037 174446    CLR        @#CHA1H          ;
3033 017244 012737 017312 174442  MOV        #25$,@#CHA1L     ;Load chain address register
3034 017252 012737 000240 174454  MOV        #SCCCH1,@#CMDR   ;Issue start chain command
3035 017260 032737 000001 174456  20$: BIT        #BIT00,@#STAT1 ;Was TC received?
3036 017266 001001          BNE          21$
3037 017270 077005          SOB          R0,20$        ;Loop control
3038
3039 017272 032737 000002 174456  21$: BIT        #BIT01,@#STAT1 ;has EOP been received?
3040 017300 001012          BNE          3$
3041 017302 012737 000001 050756  MOV        #1,@#QBE22      ;SET FLAG INDICATING Q22 PRESENT
3042 017310 000410          BR            4$
3043
3044
3045 017312 000602          ;Command chain to check for QBE at location 170020
3046 017314 177400 170020    25$: 602          ;Chain load word
3047 017320 000001          177400,QBE2    ;Destination is the Q bus exerciser on the LSI bus.
3048 017322 000030 000140    1              ;Op count
3049
3050 017326 005037 050756    30,140        ;channel mode register
3051
3052 017332 033737 050760 050756  3$: CLR        @#QBE22      ;NO Q22 BUS EXERCISER PRESENT
3053 017340 001404          4$: BIT        @#QBE18,@#QBE22 ;ARE BOTH QBES PRESENT?
3054 017342 004737 017412    BEQ        5$              ;BRANCH IF NOT
3055 017346 000137 021000    JSR        PC,Q1822        ;GO TEST ALL QBE FUNCTIONS
3056 017352 032737 000001 050760  5$: JMP        QBEFIN        ;GO TO END OF PASS AFTER QBE TESTS
3057 017360 001404          BIT        #BIT00,@#QBE18  ;IS THERE AN 18 BIT QBE PRESENT?
3058 017362 004737 017442    BEQ        6$              ;BRANCH IF NOT
3059 017366 000137 021000    JSR        PC,Q18BEX       ;GO DO 18 BIT QBE TESTS ONLY
3060 017372 032737 000001 050756  6$: JMP        QBEFIN        ;GO TO END OF PASS AFTER 18 BIT QBE TESTS
3061 017400 001402          BIT        #BIT00,@#QBE22  ;IS THERE A 22 BIT QBE PRESENT?
3062 017402 004737 017424    BEQ        7$              ;BRANCH IF NOT
3063 017406 000137 021000    JSR        PC,Q22BEX       ;GO DO 22 BIT QBE TESTS ONLY
3064
3065 017412 004737 017442    7$: JMP        QBEFIN        ;GO TO END OF PASS ROUTINE
3066 017416 004737 017424    Q1822: JSR        PC,Q18BEX
3067 017422 000207          JSR        PC,Q22BEX
3068
3069
3070          ;SUBROUTINE TO INITIATE DMA TRANSFERS TO/FROM THE QBUS EXERCISERS
3071          ;
3072
3073 017424 012737 170022 017656  Q22BEX: MOV        #QBE2+2,QBEG0+10 ;QBE 22 STARTING ADDRESS
3074 017432 012737 017670 174442  MOV        #QBCHN2,@#CHA1L ;LOAD CHAIN ADDRESS

```

```

3075 017440 000406
3076 017442 012737 170002 017656 Q188EX: BR QBEXER
3077 017450 012737 017716 174442 MOV #QBE1+2,QBEGO+10 ;SET UP FOR QBE 18 TESTS
;FIRST LOAD THE QBUS EXERCISER WITH DESIRED PARAMETERS, AND THEN
;TELL IT TO GO EXECUTE THE OPERATION
3078
3079 QBEXER: CLR @#CHA1H ;CLEAR CHAIN ADDRESS SEGMENT/TAG
3080 017456 005037 174446 MOV #1,@#SOURCE
3081 017462 012737 000001 001410 MOV #BDCXFR,EMADR ;; SET ERROR MESSAGE...
3082 017470 012737 020601 044746 MOV #NOSIG,ESADR ;...AND SIGNATURE POINTERS...
017476 012737 045206 044752 NOP ;...ERROR WILL BE CALLED LATER.
017504 000240
3083 017506 042737 000040 177540 BIC #BIT05,@#K2CSRJ ;CLEAR THE SACK TIME-OUT BIT
3084 017514 012737 000240 174454 MOV #SCCCH1,@#CMDR ;ISSUE START CHAIN COMMAND
3085 017522 004737 017572 JSR PC,1$ ;CHECK THAT TRANSFER WAS SUCCESSFUL
3086 017526 005000 CLR RO
3087 017530 077001 SOB RO, ;GIVE IT TIME
;NOW CHECK THE RESULTS OF THE TRANSFER
3088
3089 017532 012737 017744 174442 MOV #QBCHN4,@#CHA1L
3090 017540 012737 020513 044746 MOV #DMA4,EMADR ;; SET ERROR MESSAGE...
017546 012737 045206 044752 MOV #NOSIG,ESADR ;...AND SIGNATURE POINTERS...
017554 000240 NOP ;...ERROR WILL BE CALLED LATER.
3091 017556 012737 000240 174454 MOV #SCCCH1,@#CMDR
3092 017564 004737 017572 JSR PC,1$
3093 017570 000207 RTS PC
3094
3095 017572 1$: ;SUBROUTINE TO CHECK STATUS OF DMA TRANSFER
3096
3097 017572 032737 014000 174456 BIT #14000,@#STAT1 ;DID TRANSFER FINISH
3098 017600 001774 BEQ 1$
3099 017602 032737 000002 174456 BIT #EOP,@#STAT1 ;WAS THERE AN NXM?
3100 017610 001402 BEQ 2$
3101
;;\
;; >> ERROR 112, BAD TRANSFER
;;//
3102 017612 104112 ERROR+112
3103 017614 000413 BR 3$
3104 017616 032737 000001 174456 2$: BIT #TC,@#STAT1 ;DID TRANSFER COMPLETE SUCCESSFULLY?
3105 017624 001007 BNE 3$
017626 012737 020451 044746 MOV #DMA3,EMADR ;;\
017634 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 113, INCOMPLETE TRANSFER, EOP!TC NOT SET
017642 104113 ERROR+113 ;;//
3106 017644 000207 3$: RTS PC
3107
;Command chain to tell the qbe to start operation
3108 QBEGO: <CARA!CARB!COP!IV!MODE> ;Chain load word
3109 017646 001606 20,SOURCE ;hold source
3110 017650 000020 001410 177420
3111 017654 177420 .WORD 0 ;REPLACE ZERO WITH ADDRESS OF QBE CSR 2
3112 017656 000000 1 ;Operation count
3113 017660 000001 DMAV
3114 017662 000214 <SWRQ!HM>,<TRW!INTLV>
3115 017664 000030 000140
3116
;QBCHN2: 1607 ;Chain load word
3117 017670 000007 0,QBETO1 ;Source is
3118 017672 000000 017770 177400,QBE2 ;Destination is the q22 bus exerciser
3119 017676 177400 170020 7 ;Load Seven words
3120 017702 000007 DMAV ;DMA vector
3121 017704 000214 <SWRQ!HM>,<CTC!TRW!INTLV>
3122 017706 000030 100140 0,QBEGO ;Chain address
3123 017712 000000 017646

```

```

3124
3125
3126 017716 001607
3127 017720 000000 017770
3128 017724 177400 170000
3129 017730 000007
3130 017732 000214
3131 017734 000030 100140
3132 017740 000000 017646
3133
3134
3135
3136 017744 001612
3137 017746 100000 061022
3138 017752 000000 061022
3139 017756 001000
3140 017760 125252 000000
3141 017764 000031 000344
3142
3143 017770
017770 000641
017772 000000
017774 061022
017776 177000
020000 125252
020002 000000
020004 000000
3144 020006
020006 000621
020010 000400
020012 001412
020014 100000
020016 000000
020020 000000
020022 000000
3145 020024
020024 000621
020026 000400
020030 001414
020032 100000
020034 000000
020036 000000
020040 000000
3146
3147
3148
3149 020042 102 125 123
3150 020102 103 110 101
3151 020133 103 110 101
3152 020170 103 110 101
3153 020221 103 110 101
3154 020252 103 110 101
3155 020303 103 110 101
3156 020336 103 110 101
3157 020413 124 103 041
3158 020451 104 115 101
3159 020513 104 115 101

;Command chain to load QBE with data.
QBCHN3: 1607 ;Chain load word
0,QBET01 ;Source is
177400,QBE1 ;Destination is the q18 bus exerciser
7 ;Load Seven words
DMAV ;DMA vector
<SWRQ!HM>,<CTC!TRWW!INTLV>
0,QBEGO ;Chain address

;Command chain to transfer and search from Q bus memory to IOP memory.
;Will search for end of data pattern 125252.
QBCHN4: <CARA!CARB!COP!PATMSK!MODE> ;CHAIN LOAD WORD
<QBEMEM!UP>,BUFR1 ;LSI MEMORY
0,BUFR1 ;IOP MEMORY IS DESTINATION
1000 ;OP COUNT
125252,0 ;PATTERN /MASK
31,344 ;<SWRQ!HM!SNM>,<INTLV,STWW>

QBET01:
WRD!HD!DR!B1!DOUT ;DATA FOR FIRST CSR
0 ;DATA FOR SECOND CSR
BUFR1 ;DMA TRANSFER ADDRESS
-1000 ;WORD COUNT
125252 ;DATA FOR QBE DATA REG
0 ;LATENCY COUNTER/ READ ONLY
0 ;MAX VALUE LATENCY COUNTER

QBET02:
WRD!AD!HOG!DOUT ;DATA FOR FIRST CSR
400 ;DATA FOR SECOND CSR
TSTLOC ;DMA TRANSFER ADDRESS
-100000 ;WORD COUNT
0 ;DATA FOR QBE DATA REG
0 ;LATENCY COUNTER/ READ ONLY
0 ;MAX VALUE LATENCY COUNTER

QBET03:
WRD!AD!HOG!DOUT ;DATA FOR FIRST CSR
400 ;DATA FOR SECOND CSR
TSTLOC+2 ;DMA TRANSFER ADDRESS
-100000 ;WORD COUNT
0 ;DATA FOR QBE DATA REG
0 ;LATENCY COUNTER/ READ ONLY
0 ;MAX VALUE LATENCY COUNTER

;ASCII ERROR MESSAGES
;
DMA1: .ASCIZ /BUS TIME-OUT AT DMA CONTROLLER /
DMAC1: .ASCIZ /CHAN X (LOCAL => LOCAL)/<CRLF>
DMAC2: .ASCIZ \CHAN X (LOCAL(I/O)=> LOCAL)\<CRLF>
DMAC3: .ASCIZ /CHAN X (LOCAL => Q-BUS)/<CRLF>
DMAC4: .ASCIZ /CHAN X (Q-BUS => Q-BUS)/<CRLF>
DMAC5: .ASCIZ /CHAN X (Q-BUS => LOCAL)/<CRLF>
DMAC6: .ASCIZ \CHAN X (Q-BUS I/O =>LOCAL)\<CRLF>
DMAC7: .ASCIZ /CHAN X (Q-BUS SHARED MEM =>LOCAL SHARED MEM)/<CRLF>
DMA2: .ASCIZ /TC!EOP INTERRUPT NOT RECEIVED/
DMA3: .ASCIZ /DMA INCOMPLETE, TC!EOP BOTH CLEAR/
DMA4: .ASCIZ /DMA INCOMPLETE, EOP SET (BUS TIME-OUT)/

```

H7

KXJ11-CA FUNCTIONAL TEST
T7 Q-BUS INTERFACE

MACRO Y05.02 Thursday 03-Apr-86 14:11 Page 84

3160	020562	104	115	101	DMA5:	.ASCIZ	/DMA DATA ERROR/
3161	020601	200	102	101	BDQXFR:	.ASCIZ	<CRLF>/BAD TRANSFER TO QBUS/
3162	020627	123	101	103	SACKTO:	.ASCIZ	/SACK TIME-OUT BIT NOT SET/
3163	020661	105	117	120	EOPER:	.ASCIZ	/EOP NOT SET BY SACK TIME-OUT/
3164	020716	104	124	103	SACKER:	.ASCIZ	/DTC CONTENTS NOT AS EXPECTED AFTER SACK TIME-OUT/
3165						.EVEN	
3166	021000				QBFIN:		

```

3168 ;NOW SEE IF THERE IS A KNOWN GOOD IOP IN THE SYSTEM
3169 021000 032777 010000 160132 BIT #BIT12,@SWR ;IS THERE A KNOWN GOOD IOP
3170 021006 001002 BNE GD$IOP
3171 021010 000137 021754 JMP IOPEND
3172 ;KNOWN GOOD IOP MUST BE SET TO ID# 15 AND RESIDE AT ADDRESS 1775740
3173 ;BOOT/SELF TEST SWITCH SHOULD BE SET TO 5 OR 6.
3174 021014 GD$IOP: ;INIT SOME STUFF IN LOCAL MEMORY.
3175 021014 012701 061022 MOV #BUFRI,R1 ;
3176 021020 012702 010000 MOV #4096.,R2 ;
3177 021024 005003 CLR R3 ;
3178 021026 010321 199$: MOV R3,(R1)+ ;
3179 021030 062703 000002 ADD #2,R3 ;
3180 021034 077204 SOB R2,199$ ;
3181 021036 012702 001412 MOV #TSTLOC,R2 ;INIT MORE MEMORY LOCATIONS
3182 021042 012722 177777 MOV #-1,(R2)+ ;
3183 021046 005022 CLR (R2)+ ;
3184 021050 012722 125252 MOV #125252,(R2)+ ;
3185 021054 012722 052525 MOV #52525,(R2)+ ;
3186 021060 012722 063636 MOV #63636,(R2)+ ;
3187 021064 012712 036363 MOV #36363,(R2) ;
3188 021070 005037 174454 CLR @#CMDR ;RESET THE DTC
3189 021074 012737 000111 174470 MOV #111,@#MMR ;LOAD DTC MASTER MODE REGISTER
3190 021102 005037 174444 CLR @#CHA2H ;CLEAR CHAIN ADDRESS SEGMENT/TAG
3191 ;INIT QBUS MEMORY WITH KNOWN DATA
3192 021106 012737 021356 174440 1$: MOV #230,@#CHA2L ;LOAD CHAIN ADDRESS
3193 021114 012737 000241 174454 MOV #SCCCH2,@#CMDR ;ISSUE START CHAIN COMMAND
3194 021122 004737 021512 JSR PC,CH2STA ;
3195 021126 102767 BVS 1$ ;
3196 021130 012702 000020 MOV #20,R2 ;GIVE THE KNOWN GOOD IOP 16 CHANCES
3197 021134 012737 021416 174440 200$: MOV #250,@#CHA2L ;LOAD CHAIN ADDRESS
3198 021142 012737 000241 174454 MOV #SCCCH2,@#CMDR ;ISSUE START CHAIN COMMAND
3199 021150 004737 021512 JSR PC,CH2STA ;
3200 021154 102767 BVS 200$ ;TRY AGAIN IF V RTT IS SET
3201 021156 005737 001412 210$: TST TSTLOC ;WAS TPRO OF THE KNOWN GOOD IOP = ZERO?
3202 021162 001007 BNE 212$ ;
3203 021164 042737 177770 001414 BIC #C7,TSTLOC+2 ;STRIP OFF NON-STATUS BITS
3204 021172 022737 000004 001414 CMP #4,TSTLOC+2 ; IS KNOWN GOOD IOP WAITING FOR A Q-BUS COMMAND?
3205 021200 001412 BEQ 214$ ;
3206 021202 077224 SOB R2,200$ ;NO, TRY AGAIN UP TO 16 TIMES
3207 021204 012737 021756 044746 MOV #KGNR,EMADR ;:\
021212 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 114, NO RESPONSE FROM KNOWN GOOD IOP
021220 104114 ERROR+114 ;://
3208 021222 000137 021754 JMP IOPEND
3209 021226 012737 021476 021442 214$: MOV #255$,251$+4 ;
3210 021234 012737 021502 021462 MOV #255$+4,251$+24 ;
3211 021242 012737 021436 174440 MOV #251$,@#CHA2L ;LOAD CHAIN ADDRESS
3212 021250 012737 000241 174454 MOV #SCCCH2,@#CMDR ;ISSUE START CHAIN COMMAND THAT WILL
3213 ;LOAD KNOWN GOOD IOP TPRS 2 & 3 WITH LOAD
3214 ;PARAMETERS AND THEN COMMAND IT TO DO A
3215 ;DMA LOAD
3216 021256 004737 021512 JSR PC,CH2STA ;
3217 021262 102761 BVS 214$ ;TRY AGAIN IF SACK TIME OUT OCCURRED
3218 021264 004737 021622 JSR PC,TPRCHK ;
3219 ;KNOWN GOOD IOP HAS NOW BEEN INITIALIZED
3220
3221 021270 012737 021504 021442 211$: MOV #256$,251$+4 ;
3222 021276 012737 021510 021462 MOV #256$+4,251$+24 ;

```

```

3223 021304 012737 021436 174440      MOV    #251$,@#CHA2L    ;LOAD CHAIN ADDRESS
3224 021312 012737 000241 174454      MOV    #SCCCH2,@#CMDR  ;ISSUE START CHAIN COMMAND
3225 021320 004737 021512                JSR    PC,CH2STA
3226 021324 102761                BVS    211$
3227 021326 004737 021622                JSR    PC,TPRCHK
3228 021332 012737 017744 174440      MOV    #QBCHN4,@#CHA2L ;CHECK THAT TRANSFER COMPLETED
3229 021340 012737 000241 174454      MOV    #SCCCH2,@#CMDR  ;ISSUE START CHAIN COMMAND
3230 021346 004737 021512                JSR    PC,CH2STA
3231
3232 021352 000137 021754      220$:  JMP    IOPEND
3233
3234 021356                230$:  ;TRANSFER DATA IN K2 LOCATIONS TO Q BUS MEMORY
3235 021356 001602                <CARA!CARB!COP!MODE>
3236 021360 000000 061022                <K2MEM!UP>,BUFR1
3237 021364 100000 061022                <QBMEM!UP>,BUFR1
3238 021370 010000                4096.
3239 021372 000030 100140                <SWRQ!HM>,<CTC!TRWW!INTLV>
3240
3241                ;TRANSFER DATA IN K2 LOCATIONS TO QBUS MEM
3242 021376 001602                <CARA!CARB!COP!MODE>
3243 021400 000000 001416                <K2MEM!UP>,TSTLOC+4
3244 021404 100000 001416                <QBMEM!UP>,TSTLOC+4
3245 021410 000004                4
3246 021412 000030 000140                <SWRQ!HM>,<TRWW!INTLV>
3247
3248 021416                250$:  ;CHAIN FILE TO READ TPRO AND TPR1 OF KNOWN GOOD IOP
3249 021416 001602                <CARA!CARB!COP!MODE>
3250 021420 177400 175740                <QBUSIO!37400!UP>, 175740                ;ADDRESS OF TPRO IN KGIOP
3251 021424 000000 001412                <K2MEM!UP>, TSTLOC
3252 021430 000002                2
3253 021432 000030 000140                <SWRQ!HM>,<TRWW!INTLV>
3254
3255 021436                251$:  ;CHAIN FILE TO LOAD PARAMETERS INTO TPRS 2 & 3
3256 021436 001602                <CARA!CARB!COP!MODE>
3257 021440 000000                <K2MEM!UP>
3258 021442 021476                .WORD  255$
3259 021444 177400 175744                <QBUSIO!37400!UP>, 175744
3260 021450 000002                2
3261 021452 000030 100140                <SWRQ!HM>,<CTC!TRWW!INTLV>
3262
3263                ;ISSUE A DMA LOAD COMMAND
3264 021456 001602                <CARA!CARB!COP!MODE>
3265 021460 000000                <K2MEM!UP>
3266 021462 021502                .WORD  255$+4
3267 021464 177400 175740                <QBUSIO!37400!UP>, 175740
3268 021470 000001                1
3269 021472 000030 000140                <SWRQ!HM>,<TRWW!INTLV>
3270
3271 021476 022054                255$:  .WORD  KG#T01                ;CHAIN TO INIT KNOWN GOOD IOP MEMORY
3272 021500 100000                .WORD  <QBMEM!UP>                ;...RESIDES IN Q BUS MEMORY
3273 021502 000002                .WORD  2                ;DMA LOAD COMMAND FOR KNOWN GOOD IOP TPRO
3274 021504 022114                256$:  .WORD  KG#T03
3275 021506 100000                .WORD  <QBMEM!UP>
3276 021510 000002                .WORD  2                ;DMA LOAD COMMAND FOR KNOWN GOOD IOP
3277
3278 021512                CH2STA: ;SUBROUTINE TO POLL THE CHANNEL 2 STATUS REGISTER TO
3279                ;INDICATE WHEN A TRANSFER HAS FINISHED AND IF IT WAS SUCCESSFUL.

```

K7

```

3280 021512 032737 014000 174454 BIT #14000,@#STAT2 ;LOOK FOR THE CA:NAC BIT TO SET
3281 021520 001774 BEQ CH2STA ;
3282 021522 032737 000002 174454 BIT #EOP,@#STAT2 ;WAS THE TRANSFER SUCCESSFUL?
3283 021530 001420 BEQ 1$ ;
3284 ;TRANSFER WAS NOT SUCCESSFUL. WAS IT DUE TO SACK TIME-OUT?
3285 021532 032737 000040 177540 BIT #BIT05,@#K2CSRJ ;
3286 021540 001007 BNE 2$ ;
3287 021542 012737 020513 044746 MOV #DMA4,EMADR ;;\
021550 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 115, TRANSFER INCOMPLETE, EOP SET
021556 104115 ERROR+115 ;://
3288 021560 042737 000040 177540 2$: BIC #BIT05,@#K2CSRJ ;CLEAR THE SACK TIME-OUT BIT
3289 021566 000262 SEV ;SET THE V BIT IN PSW
3290 021570 000413 BR 3$ ;
3291 021572 032737 000001 174454 1$: BIT #TC,@#STAT2 ;WAS TC SET?
3292 021600 001007 BNE 3$ ;
3293 021602 012737 020451 044746 MOV #DMA3,EMADR ;;\
021610 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 116, TRANSFER INCOMPLETE, EOP:TC NOT SET
021616 104116 ERROR+116 ;://
3294 021620 000207 3$: RTS PC
3295
3296 021622 TPRCHK: ;SUBROUTINE TO CHECK TPRO & 1 OF THE KNOWN GOOD IOP AFTER
3297 ;ISSUING A QBUS COMMAND TO IT
3298
3299 021622 012737 177777 001412 1$: MOV #-1,TSTLOC ;INIT TEST LOCATIONS
3300 021630 005037 001414 CLR TSTLOC+2 ;
3301 021634 2$: ;NOW READ TPR 0 & 1 OF THE KNOWN GOOD IOP TO MAKE SURE COMMAND IS DONE.
3302 021634 012737 021734 174440 MOV #5$,@#CHA2L ;
3303 021642 012737 000241 174454 MOV #SCCCH2,@#CMDR ;ISSUE START CHAIN COMMAND
3304 021650 004737 021512 JSR PC,CH2STA ;
3305 021654 102767 BVS 2$ ;TRY AGAIN IF SACK TIME-OUT IS SET
3306 021656 005737 001412 TST TSTLOC ;IS TPRO OF KNOWN GOOD IOP = ZERO
3307 021662 001357 BNE 1$ ;
3308 021664 032737 154000 001414 BIT #154000,TSTLOC+2 ;ANY ERRORS?
3309 021672 001410 BEQ 3$ ;
3310 021674 012737 022016 044746 MOV #KG.ER,EMADR ;;\
021702 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 117, KNOWN GOOD IOP ERROR
021710 104117 ERROR+117 ;://
3311 021712 000407 BR 4$ ;
3312 021714 042737 177770 001414 3$: BIC #1C7,TSTLOC+2 ;STRIP OFF NON-STATUS BITS
3313 021722 022737 000004 001414 CMP #4,TSTLOC+2 ;
3314 021730 001341 BNE 2$ ;
3315 021732 000207 4$: RTS PC
3316
3317 021734 5$: ;CHECK TPR 0 & 1 OF KNOWN GOOD IOP TO SEE THAT COMMAND WAS SUCCESSFUL
3318 021734 001602 <CARA!CARB!COP!MODE>
3319 021736 140000 175740 <QBUSIO!UP>,175740 ;TRANSFER FROM TPRO & 1 OF KNOWN GOOD IOP
3320 021742 000000 001412 <K2MEM!UP>,TSTLOC ;TO K2 MEMORY
3321 021746 000002 2
3322 021750 000030 000140 <SWRQ!HM>,<TRMM!INTLV>
3323
3324 021754 IOPEND:
021754 000467 BR TST10 ;;
3325 021756 116 117 040 KGNR: .ASCIZ /NO RESPONSE FROM KNOWN GOOD IOP/
3326 022016 113 116 117 KG.ER: .ASCIZ /KNOWN GOOD IOP COMMAND ERROR/
3327 .EVEN
3328 ;
3329 ;THE FOLLOWING CHAIN FILES WILL BE LOADED BY THE KNOWN GOOD IOP.

```


3330
3331 022054
3332 022054 001602
3333 022056 100000 061022
3334 022062 000000 061022
3335 022066 010000
3336 022070 000030 100040
3337
3338 022074
3339 022074 001602
3340 022076 100000 001416
3341 022102 000000 001416
3342 022106 000004
3343 022110 000030 000040
3344
3345 022114
3346 022114 001602
3347 022116 000020 001416
3348 022122 100000 061022
3349 022126 001000
3350 022130 000030 000040
3351

KG#T01: ;INIT KNOWN GOOD IOP'S MEMORY WITH KNOWN DATA
<CARA!CARB!COP!MODE>
<QBEM!UP>, BUFRI
<K2MEM!UP>, BUFRI
4096
<SWRQ!HM>, <CTC!TRWW!BUSHOG>

KG#T02: ;INIT KNOWN GOOD IOP'S TSTLOCS WITH KNOWN DATA
<CARA!CARB!COP!MODE>
QBEM!UP, TSTLOC+4
K2MEM!UP, TSTLOC+4
4
<SWRQ!HM>, <TRWW!BUSHOG>

KG#T03: ;CAUSE THE KNOWN GOOD IOP TO TRANSFER TO QBUS
<CARA!CARB!COP!MODE>
<K2MEM!HOLD>, TSTLOC+4 ;TRANSFER FROM KNOWN GOOD IOP MEMORY..
<QBEM!UP>, BUFRI ;...TO Q BUS MEMORY
1000
<SWRQ!HM>, <TRWW!BUSHOG>

3355

```

022134 000004
022136 012706 001100
022142 012757 177777 002426
022150 005037 044742
022154 032777 000400 156756
022162 001006
022164 005777 157136
022170 000240
022172 102002
022174 000137 024602
022200 104415 022206
022204 000417
022206 040 040 123
    
```

```

*****
; *TEST 10 SECOND SERIAL LINE NEC7201
*****
TST10: SCOPE
MOV #STACK,SP ; SET A CLEAN STACK.
MOV #1,TRAP4Y ; DISMISS BUS-ERRORS.
CLR EMPRE ; CLEAR ERROR PREFIX.
BIT #BIT8,@SWR ; FORCED ENTRY SELECTED ??
BNE 30048$ ; BR IF SO.
TST @SL2 ; CHECK FOR VALID SL2.
NOP
BVC 30048$ ; SKIP NEXT IF OK.
30047$: JMP TST11 ; BYPASS.
30048$: T$NAME, ; PRINT TEST NUMBER AND NAME...
+.4 ; ...AND SKIP OVER THE ASCII.
BR 30049$
.ASCIIZ \ SECOND SERIAL LINE NEC7201\
.EVEN
    
```

3356 022244

022244

3357
3358
3359
3360
3361
3362
3363
3364

```

022244 012737 022252 001110
022252 012705 175736
022256 012737 022344 002426
022264 000236
022266 112715 0000b6
022272 112715 000166
022276 112715 000264
022302 013745 022474
022306 113715 022475
022312 013745 022476
022316 113715 022477
022322 013745 022476
022326 113715 022477
022332 005745
022334 020527 175720
022340 101374
022342 000413
022344 010537 001120
022350 012737 047406 044746
022356 012737 045230 044752
022364 104120
022366 000137 024536
022372 005037 002426
022376 012737 022470 000104
022404 052777 000200 156722
022412 005001
    
```

```

30049$:
*****
; *TEST 10.1 CLOCK GENERATOR I8254
*****
T10.1.:
; NEC7201 SYNC/ASYNC (MULTI-PROTOCOL) SERIAL INTERFACE.
; INCLUDES AN INTEL 8254 TRIPLE PIT FOR BAUD RATE GENERATION.
; FIRST, SET UP THE BAUD GENERATOR, AND VERIFY THAT
; GENERAL PURPOSE TIMER 2 (800HZ) WORKS AS ADVERTISED.
;
; BAUDSET:
MOV #+.6,$LPERR ; LOOP HERE ON ERROR.
MOV #175736,R5 ; I8254 CONTROL REG.
MOV #2$,TRAP4X ; SET A TRAP CATCHER.
SPL 6 ; RAISE CPU.
MOV #066,(R5) ; CHAN A <TIMER-0!2-BYTES!MODE-3!BIN>.
MOV #166,(R5) ; CHAN B <TIMER-1! SAME >.
MOV #264,(R5) ; 800HZ <TIMER-2! SAME EXCEPT MODE-2>.
MOV HZ800,-(R5) ; SET 800HZ INTERVAL == 50HZ
MOV A4800,-(R5) ; SET CHAN B...
MOV A4800+1,(R5)
MOV A4800,-(R5) ; ...AND CHAN A FOR ASYNC 4800 (SYNC 76.8K).
MOV A4800+1,(R5)
1$: TST -(R5) ; VERIFY THE REMAINING REGISTER ADDRESSES.
CMP R5,#175720
BHI 1$
BR 3$ ; SO FAR SO GOOD, SKIP OVER THE TRAPPER.
2$: MOV R5,$GDADR
MOV #NEC1,EMADR ; \
MOV #LA16,ESADR ; >> ERROR 120, BUS ERROR AT 7201 OR 8254 ADDRESS
; //
JMP NEC.END
3$: CLR TRAP4X ; RESET BUS-ERROR...
MOV #5$,PEVNT ; ...SET PEVNT VECTOR.
BIS #BIT7,@CSRA ; TURN IT ON.
CLR R1
    
```

N7

KXJ11-CA FUNCTIONAL TEST
T10.1 CLOCK GENERATOR I8254

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 90

```

3389 022414 005000          CLR      R0
3390 022416 106427 000000    MTPS    #PRO      ; LOWER CPU.
3391 022422 077001          SOB      R0,..    ; DELAY, PEVNT SHOULD INTERRUPT AND...
3392 022424 005301          DEC      R1      ; ...GIVE A FEW TICKS (MORE THAN 1).
3393 022426 003007          BGT     4$
3394 022430 012737 047455 044746    MOV     #NEC2,EMADR ;:\
      022436 012737 045206 044752    MOV     #OSIG,ESADR ;: >> ERROR 121, 8254 TIMER 2 (800HZ) DOESN'T WORK
      022444 104121          ERROR+121 ;://
3395 022446 005077 156662 4$:    CLR     @#CSRA    ; TURN OFF PEVNT...
3396 022452 012737 000260 175736    MOV     #260,@#175736 ;...AND TIMER 2 (MODE-0).
3397 022460 012737 002542 000104    MOV     #DISMISS,PEVNT ;...RESET THE VECTOR...
3398 022466 000434          BR      NEC.ASYNC ;...AND PROCEED.
3399
3400 022470 005201 000002 5$:    .WORD  INC+1, RTI ; ON INTERRUPT, TICK AND RETURN.
3401
3402 022474 000020          HZ800: 16.      ; DIVIDER YIELDS == 50HZ
3403 022476 000200          A4800: 128.    ; DIVIDER YIELDS ASYNC 4800 AND/OR SYNC 76.8K.

```

3405

022500

3406

3407

3408

3409

3410

3411

3412

3413

3414

3415

3416

3417

3418

3419

3420

3421

3422

3423

3424

3425

3426

3427

3428

3429

3430

3431

3432

3433

3434

3435

3436

3437

3438

3439

3440

3441

3442

3443

3444

3445

3446

3447

3448

3449

3450

3451

3452

3453

3454

3455

3456

3457

3458

000004
000005
177774
177776
000000
000002

177777

177777

177777

156524

```
*****
*TEST 10.2 SYNC/ASYNC DATA
*****
T10.2::
```

```
; FIVE PASS SYNC/ASYNC DATA TEST (REQUIRES EXTERNAL LOOP-BACKS).
```

- ; 1. CHAN B, ASYNC (4800 BAUD).
- ; 2. CHAN A, ASYNC (4800 BAUD).
- ; 3. CHAN B, SYNC (76.8K BAUD).
- ; 4. CHAN A, SYNC (76.8K BAUD).
- ; 5. CHAN A, SYNC, DMA DRIVEN.

```
CHA= #4 ; DEDICATE R4 AS "CHAN-A" AND...
CHX= #5 ; ...R5 AS "CHAN-IN-TEST" PORT POINTERS.
STAT= -4 ; INDICES TO CHAN A/B STATUS...
RDB= -2 ; ...RCVR DATA...
CTRL= 0 ; ...CONTROL...
XDB= 2 ; ...AND XMTR DATA PORTS.
```

```
; ASYNC SET-UP R CODE FUNCTION
```

A.SET:	.BYTE	R	CODE	FUNCTION
	0	030	030	; COMMAND <CHANNEL RESET>.
	2	024	024	; BUS INTERFACE <NON-VECTORED!PRI1!NO-DMA>.
	4	104	104	; PROTOCOL <X16!ASYNC-1-STOP!NO-PARITY>.
	3	301	301	; RCVR CONTROL <8-BPC!RX-ENAB>.
	5	152	152	; XMTR CONTROL <8-BPC!TX-ENAB!PRTYX-B>.
	0	020	020	; COMMAND <RESET-EXT-INTS>.
	1	036	036	; INT CONTROL <RXI-ON-ALL!CAV!TX-IE!NO-EXT-IE>.
			-1	

```
; SYNC SET-UP R CODE FUNCTION
```

S.SET:	.BYTE	R	CODE	FUNCTION
	0	030	030	; COMAND <CHANNEL RESET>.
	2	024	024	; BUS INTERFACE <NON-VECTORED!PRI1!NO-DMA>.
	4	040	040	; PROTOCOL <X1!SDLC!SYNC!NO-PARITY>.
	6	000	000	; SYNC 1 <SEARCH ADDRESS -- UNUSED>.
	7	176	176	<FLAG CODE>.
	3	311	311	; RCVR CONTROL <8-BPC!CRC-ENAB!RX-ENAB>.
	5	153	153	; XMTR CONTROL <8-BPC!TX-ENAB!CCITT!PRTYX-B!CRC-ENAB>.
	0	200	200	; COMMAND <RESET TX-CRC-GEN>.
	0	100	100	; COMMAND <RESET RX-CRC-GEN>.
	0	020	020	; COMMAND <RESET EXT-INT>.
	1	036	036	; INT CONTROL <RXI-ON-ALL!CAV!TX-IE!NO-EXT-IE>.
			-1	
C.CLR:	.BYTE	1	004	; INT CONTROL <KEEP-CAV!DISABLE-ALL-ELSE>
	.BYTE	3	300	; RCVR <RX-DISABLE>.
	.BYTE	5	142	; XMTR <TX-DISABLE>.
			-1	

```
; NEC.ASYNC: MOV #A,RO ; ASYNC MODE FIRST...
```

```
MOV #A,R1
BR NEC01
```

```
; ...THEN, SYNC (SDLC) MODE...
```

```
NEC.SYNC: MOV #S,RO
MOV #177,R1
```

```
BIS #4,@CSRA ; ...REQUIRES <SYNCHA! SYNCMB> IN CSRA.
NEC01: MOV RO,8 ; SAVE POINTER AS SYNC/ASYNC FLAG.
```

```

3459 022614 012737 047531 044742 MOV #NECCM,EMPRE ; SET ERROR PREFIX...
3460 022622 112737 000102 047536 MOV #B,NECC ;...CHAN B...
3461 022630 110137 047541 MOV R1,NECC ;...ASYN/SYN.
3462 022634 012737 022642 001110 MOV #.6,$LPERR ;; LOOP HERE ON ERROR.
3463 022642 012706 001100 MOV #STACK,SP
3464 022646 013704 001326 MOV $SL2,CHA ; GET NEC7201 BASE...
3465 022652 062704 000004 ADD #4,CHA ;...AND RAISE TO CHAN-A CONTROL PORT.
3466 022656 010405 MOV CHA,CHX ; COPY...
3467 022660 062705 000010 ADD #10,CHX ;...AND RAISE TO CHAN-B CONTROL PORT.
3468 022664 111014 MOV (R0),(CHA)
3469 022666 116014 000001 MOV 1(R0),(CHA) ; COMMAND <CHAN-A-RESET (INCLUDES INT LOGIC)>.
3470 022672 000240 000240 240,240 ; WAIT A MOMENT.
3471 022676 112015 MOV (R0)+,(CHX)
3472 022700 112015 MOV (R0)+,(CHX) ; COMMAND <CHAN-B-RESET>.
3473 022702 000240 000240 240,240 ; WAIT AGAIN.
3474 022706 106427 000200 1$: MTPS #PR4 ; RAISE CPU.
3475 022712 112014 MOV (R0)+,(CHA) ; POINT...
3476 022714 112014 MOV (R0)+,(CHA) ;...AND SET BUS-INTERFACE (CR2A ALWAYS).
3477 022716 112015 2$: MOV (R0)+,(CHX) ; POINT...
3478 022720 112015 MOV (R0)+,(CHX) ;...AND SET UP THE REST.
3479 022722 105710 TST (R0)
3480 022724 100374 BPL 2$
3481 022726 012703 061022 MOV #BUFR1,R3
3482 022732 012700 000020 MOV #16.,R0
3483 022736 005023 3$: CLR (R3)+ ; CLEAR SOME FREE SPACE.
3484 022740 077002 SOB R0,3$
3485 022742 012703 061022 MOV #BUFR1,R3 ; RCV BUFFER POINTER => R3.
3486 022746 012702 037600 MOV #FLT10,R2 ; XMT BUFFER POINTER => R2.
3487 022752 005037 001124 CLR $GDDAT ; CLEAR FOR EXP'D...
3488 022756 005037 001126 CLR $BDDAT ;...AND REC'D DATA...
3489 022762 005037 001130 CLR $BDDAT+2 ;...AND FINAL STATUS.
3490 022766 005037 001132 CLR $BDDAT+4
3491 022772 012737 023674 000070 MOV #NEC.I,NECV ; SET VECTOR...
3492 023000 106427 000000 MTPS #PRO ;...LOWER CPU.
3493 023004 005000 CLR R0
3494 023006 012701 000001 MOV #1,R1 ; INIT BYTE COUNTS (RCV=0, XMT=1)...
3495 023012 112265 000002 MOV (R2)+,XDB(CHX) ;...XMT 1ST BYTE...
3496 023016 112715 000300 MOV #<3*BIT6>,(CHX) ;...AND COMMAND <RESET-IDLE/CRC-LATCH>.
3497 023022 023727 023254 022520 CMP #S.SET ; SYNC MODE ??
3498 023030 001405 BEQ 41$ ; BR IF SO.
3499
3500 023032 020127 011022 4$: CMP R1,#<18.*BIT8>!18. ; ASYN -- 18 BYTES TRANSFERRED ??
3501 023036 001435 BEQ 44$ ; BR IF SO.
3502 023040 077004 SOB R0,4$ ; ELSE, HANG-AROUND.
3503 023042 000416 BR 43$ ; ASYN WAIT LOOP TIME-OUT.
3504
3505 023044 105737 001132 41$: TST $BDDAT+4 ; SYNC -- <EOF> RECEIVED ??
3506 023050 100410 BMI 42$ ; BR IF SO.
3507 023052 077004 SOB R0,41$ ; ELSE, SAME-OLE-CRAP.
3508 023054 012737 047625 044746 MOV #NEC3A,EMADR ;\
023062 012737 045610 044752 MOV #EF11,ESADR ;>> ERROR 122, <EOF-SDLC> NOT RECEIVED
023070 104122 ERROR-122 ;//
3509 023072 020127 012022 42$: CMP R1,#<20.*BIT8>!18.> ; BYTE COUNTS RIGHT ??
3510 023076 001415 BEQ 44$ ; PROCEED IF SO.
3511
3512 023100 110137 001124 43$: MOV R1,$GDDAT ; SENT THIS MANY...
3513 023104 000301 SWAB R1

```

D8

KXJ11-CA FUNCTIONAL TEST
T10.2 SYNC/ASYNC DATA

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 93

```

3514 023106 110137 001126          MOV      R1,$BDDAT          ;...AND RECEIVED THIS MANY.
3515 023112 012737 047574 044746  MOV      #NEC3,EMADR        ;:\\
      023120 012737 045556 044752  MOV      #EF10,ESADR        ;: >> ERROR 123, LOOP TIME-OUT, DATA XFER INCOMPLETE
      023126 104123          ERROR+123                    ;://
3516          ;: BR      5$
3517 023130 000240          NOP
3518
3519 023132 012701 000022          44$: MOV      #18.,R1
3520 023136 012702 037600          MOV      #FLT10,R2          ; POINT TO XMT'D...
3521 023142 012703 061022          MOV      #BUFR1,R3          ;...AND REC'D DATA BUFFERS.
3522 023146 121312          45$: CMPB     (R3),(R2)        ; RECV'D SAME AS THAT XMIT'D ??
3523 023150 001415          BEQ      46$                ; BR IF SO.
3524 023152 010337 001120          MOV      R3,$GDADR
3525 023156 111337 001126          MOVB     (R3),$BDDAT
3526 023162 111237 001124          MOVB     (R2),$GDDAT
3527 023166 012737 047356 044746  MOV      #NEC4,EMADR        ;:\\
      023174 012737 045210 044752  MOV      #EF2,ESADR        ;: >> ERROR 124, RECEIVED DATA INCORRECT
      023202 104124          ERROR+124                    ;://
3528 023204 122322          46$: CMPB     (R3)+,(R2)+    ; BUMP...
3529 023206 077121          SOB      R1,45$            ;...AND LOOP
3530
3531 023210 012700 022550          5$: MOV      #C.CLR,R0        ; DONE, CLEAN-UP TABLE POINTER => R0.
3532 023214 112015          6$: MOVB     (R0)+,(CHX)      ; POINT...
3533 023216 112015          MOVB     (R0)+,(CHX)        ;...AND CLEAR (RESET) AS NECESSARY.
3534 023220 105710          TSTB    (R0)
3535 023222 100?          BPL      6$
3536 023224 020E          CMP      CHX,CHA           ; BOTH CHANNELS DONE ??
3537 023226 001410          BEQ      7$                ; BR IF SO.
3538 023230 010405          MOV      CHA,CHX           ; ELSE, POINT TO CHAN A...
3539 023232 112737 000101 047536  MOVB     #'A,NECC           ;...ADJUST ERROR PREFIX...
3540 023240 013700 023254          MOV      8$,R0             ;...GET SET UP POINTER => R0...
3541 023244 005720          TST      (R0)+            ;...BUMP PAST THE CHAN-RESET...
3542 023246 000617          BR       1$                ;...AND GO 'ROUND ONCE.
3543
3544 023250 022727 022520          7$: CMP      #S.SET,(PC)+    ; SYNC MODE DONE ??
3545 023254 022500          8$: A.SET
3546 023256 001402          BEQ      NEC.DMA           ; SKIP IF SO...
3547 023260 000137 022572          JMP      NEC.SYNC          ;...ELSE, DO IT NOW.
3548
3549          ; NOW ONCE MORE USING DMA (ASSUMING THAT 8016 IS ALIVE AND WELL).
3550          ; THE 7201 CHIP IS STILL SET-UP FOR "CHAN A SYNC" MODE.
3551
3552 023264          ; NEC.DMA:
      023264 012737 023272 001110  MOV      #.+6,$LPERR        ;; LOOP HERE ON ERROR.
3553 023272 106427 000200          MTPS     #PR4
3554 023276 012701 023316          MOV      #2$,R1           ; ADJUST CHAN A SET-UP.
3555 023302 012702 000004          MOV      #4,R2
3556 023306 112114          1$: MOVB     (R1)+,(CHA)      ; POINT TO...
3557 023310 112114          MOVB     (R1)+,(CHA)      ;...AND CHANGE...
3558 023312 077203          SOB      R2,1$            ;...THE FOLLOWING 4 REGISTERS.
3559 023314 000404          BR       3$
3560 023316          002          025          2$: .BYTE    2, 025          ; CHANGE BUS-INTERFACE TO <CHAN-A-DMA>
3561 023320          003          311          .BYTE    3, 311          ; ENABLE RECVR...
3562 023322          005          151          .BYTE    5, 151          ;...AND XMITTR.
3563 023324          001          016          .BYTE    1, 016          ; CHANGE INT CONTROL TO <RXI-ON-1ST-CHAR>
3564
3565 023326 012700 061022          3$: MOV      #BUFR1,R0

```

E8

KXJ11-CA FUNCTIONAL TEST
T10.2 SYNC/ASYNC DATA

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 94

```

3566 023332 012701 000020      MOV     #16.,R1
3567 023336 005020      CLR     (R0)+      ; CLEAR RECEIVING BUFFER.
3568 023340 077102      SOB     R1,-2
3569 023342 012737 024026 000070      MOV     #IXIT+2,NECV ; CHANGE THE VECTOR (INTS DISMISSED).
3570 023350 012737 047551 044742      MOV     #NECDM,EMPRE ; CHANGE THE ERROR PREFIX.
3571 023356 005005      CLR     R5         ; SET-UP THE DMA ENGINE.
3572 023360 005065 174454      CLR     CMDR(R5)   ; CHIP RESET.
3573 023364 012765 000115 174470      MOV     #115,MHR(R5) ; SET MASTER MODE <VI!WAIT!CPINTLV!ENAB>.
3574 023372 012765 000074 174454      MOV     #40!34,CMDR(R5) ; CLEAR IE'S...
3575 023400 012765 000075 174454      MOV     #41!34,CMDR(R5) ; ... BOTH CHANNELS.
3576 023406 005065 174446      CLR     CHA1H(R5)  ; CLEAR HI CHAIN ADDRESS...
3577 023412 005065 174444      CLR     CHA2H(R5)  ; ... BOTH CHANNELS.
3578 023416 012765 023450 174440      MOV     #4$,CHA2L(R5) ; SET CHAN 2 (XMT)...
3579 023424 012765 023470 174442      MOV     #5$,CHA1L(R5) ; ... AND CHAN 1 (RCV) CHAIN ADDRESSES.
3580 023432 012765 000240 174454      MOV     #SCCCH1,CMDR(R5) ; ISSUE START CHAIN COMMAND CHAN 1 (RCVR).
3581 023440 012765 000241 174454      MOV     #SCCCH2,CMDR(R5) ; ISSUE START CHAIN COMMAND CHAN 2 (XMTR).
3582 023446 000420      BR      6$
3583 023450 001602      4$:    1602      ; CHAIN-LOAD ARA, ARB, OPK, AND CH MODE.
3584 023452 000000 037600      0, FLT10      ; FROM -- DATA TABLE.
3585 023456 040020 175707      <K2IO!HOLD>, 175706+1 ; TO ---- 7201 XDB(LB), NO AUTO-INCR.
3586 023462 000022      18.         ; OPK --- 18 BYTES.
3587 023464 000020 001201      SWRQ, 1201    ; MODE -- SFT-REQ,IP-ON-TC!EOP,SNGL,BYTE BYTE.
3588 023470 001602      5$:    1602      ; FROM -- 7201 RDB(LB), NO AUTO INCR.
3589 023472 040020 175703      <K2IO!HOLD>, 175702+1 ; TO ---- BUFFER 1.
3590 023476 000000 061022      0, BUFR1     ; OPK --- 20 BYTES (18 DATA + 2 CRC).
3591 023502 000024      20.         ; MODE -- SAME AS ABOVE (EXCEPT NO SFT-REQ).
3592 023504 000000 001201      0, 1201
3593
3594 023510 112714 000300      6$:    MOVB    #<3*BIT6>,(CHA) ; COMMAND <RESET-IDLE/CRC-LATCH>...
3595 023514 005000      CLR     RO
3596 023516 106400      MTPS   RO      ; ...LOWER CPU.
3597 023520 032765 020000 174456 7$:    BIT     #IP,STAT1(R5) ; ...AND WAIT FOR CHAN1 (RCVR) DONE.
3598 023526 001001      BNE    8$
3599 023530 077005      SOB    RO,7$
3600
3601 023532 016500 174462      8$:    MOV     COP1(R5),RO ; GET RCVR (CHAN1) BYTES REMAINING.
3602 023536 001422      BEQ    11$      ; BR IF NONE (ALL RECEIVED).
3603 023540 012737 000024 001126      MOV     #20.,$BDDAT
3604 023546 160037 001126      SUB    RO,$BDDAT ; ELSE, CALCULATE BYTES RCVD (CHAN1 OP COUNT)...
3605 023552 012737 000022 001124      MOV     #18.,$GDDAT
3606 023560 166537 174460 001124      SUB    COP2(R5),$GDDAT ; ...VERSUS BYTES XMTD (CHAN2 OP COUNT).
3607 023566 012737 047574 044746      MOV     #NEC3,EMADR ; ;\
      023574 012737 045556 044752      MOV     #EF10,ESADR ; ; >> ERROR 125, DMA DATA XFER INCOMPLETE
      023602 104125      ERROR+125 ; ;//
3608
3609 023604 012700 037600      11$:   MOV     #FLT10,RO
3610 023610 012701 061022      MOV     #BUFR1,R1
3611 023614 012702 000022      MOV     #18.,R2
3612 023620 121011      12$:   CMPB   (R0),(R1) ; CHECK DATA.
3613 023622 001415      BEQ    13$      ; BR IF OK.
3614 023624 010137 001120      MOV     R1,$GDADR
3615 023630 111037 001124      MOVB   (R0),$GDDAT
3616 023634 111137 001126      MOVB   (R1),$BDDAT
3617 023640 012737 047356 044746      MOV     #NEC4,EMADR ; ;\
      023646 012737 045210 044752      MOV     #EF2,ESADR ; ; >> ERROR 126, RECEIVED DATA (DMA) INCORRECT
      023654 104126      ERROR+126 ; ;//
3618 023656 122021      13$:   CMPB   (R0)+,(R1)+ ; BUMP...

```

```

3619 023660 077221          SOB      R2,12$          ;...AND LOOP.
3620
3621 023662 005077 155446    CLR      @#CSRA          ; ALL DONE  FINALLY, CLEAR CSRA...
3622 023666 005065 174454    CLR      CMDR(R5)        ;...RESET DMA CHIP...
3623 023672 000460          BR       NEC.MDM         ;...AND PROCEED TO MODEM CONTROL TEST.
3624
3625          ; SYNC/ASYNC INTERRUPT HANDLER (DATA TESTS ONLY).
3626          ; BYTE COUNTS MAINTAINED IN R1 (LO=XMT, HI=RCV).
3627
3628 023674 112764 000002 000010  NEC.I:  MOVB    #2,10(CHA)      ; INTERRUPT -- POINT TO SR2B...
3629 023702 116446 000004          MOVB    STAT+10(CHA),-(SP) ;...AND PUSH VECT/COND.
3630 023706 006016          ROR     (SP)              ; BIT 0 => "C"...
3631 023710 103421          BCS    ISEXT             ;...AND BR IF SPEC/EXT (1, 3, 5, OR 7).
3632 023712 006016          ROR     (SP)              ; BIT 1 => "C"...
3633 023714 103412          BCS    IRX               ;...AND BR IF RCVR (2 OR 6)...
3634
3635          ;...ELSE FALL THRU (0 OR 4).
3636 023716 120127 000022          ITX:   CMPB   R1,#18.      ; XMIT -- ANYTHING LEFT ??
3637 023722 001404          BEQ    1$               ; BR IF NOT.
3638 023724 112265 000002          MOVB   (R2)+,XDB(CHX)    ; SEND NEXT BYTE...
3639 023730 105201          INCB   R1                ;...AND COUNT IT.
3640 023732 000434          BR     IXIT
3641 023734 112715 000050  1$:   MOVB   #<5*BIT3>,(CHX) ; XMIT DONE, COMMAND <RESET-TXIP>...
3642 023740 000431          BR     IXIT              ;...AND RETURN
3643
3644 023742 116523 177776          IRX:   MOVB   RDB(CHX),(R3)+ ; RCVR -- STUFF THE CHAR...
3645 023746 062701 000400          ADD    #<1*BIT8>,R1      ;...AND COUNT IT.
3646 023752 000424          BR     IXIT
3647
3648 023754 116537 177774 001130  ISEXT:  MOVB   STAT(CHX),#BDDAT+2 ; SPECIAL/EXTERNAL, GET SRO...
3649 023762 112715 000001          MOVB   #1,(CHX)
3650 023766 116537 177774 001132          MOVB   STAT(CHX),#BDDAT+4 ;...AND SR1.
3651 023774 100407          BMI    1$               ; JUST RETURN IF <SDLC-EOF>.
3652 023776 012737 047655 044746          MOV    #NEC5,EMADR       ;: \
024004 012737 045610 044752          MOV    #EF11,ESADR      ;: >> ERROR 127, UNEXPECTED EXT OR ERROR INTERRUPT
024012 104127          ERROR+127              ;: //
3653 024014 112715 000060  1$:   MOVB   #<6*BIT3>,(CHX) ; COMMAND <RESET-ERROR>...
3654 024020 112715 000020          MOVB   #<2*BIT3>,(CHX) ;...AND <RESET-EXTERNAL>.
3655
3656 024024 005726          IXIT:  TST    (SP)+       ; FIX THE STACK.
3657 024026 112714 000070          MOVB   #<7*BIT3>,(CHA)  ; SIGNAL <END-OF-INTERRUPT>.
3658 024032 000002          RTI

```


3660

```

;*****
;*TEST 10.3  MODEM CONTROLS
;*****
T10.3::
CHB=  #5          ; REDEFINE R5 AS CHAN B POINTER.
;
; NEC.MDM:
3661 024034 000005
3662 024034
3663 024034 012737 024042 001110
3664 024042 106427 000200
3665 024046 010405
3666 024050 062705 000010
3667 024054 112714 000030
3668 024060 112715 000030
3669 024064 004737 037564
3670 024070 112714 000021
3671 024074 112715 000021
3672 024100 112714 000005
3673 024104 112715 000005
3674 024110 012737 047721 044742
3675 024116 012737 024546 000070
3676 024124 005003
3677 024126 005000
3678 024130 106400
3679
3680
3681
3682 024132 005037 001126
3683 024136 005037 001130
3684 024142 112714 000005
3685 024146 112714 000002
3686 024152 105737 001126
3687 024156 001010
3688 024160 077004
3689 024162 012737 047746 044746
    024170 012737 045206 044752
    024176 104130
3690 024200 032737 000040 001126
3691 024206 001404
3692 024210 032737 000010 001126
3693 024216 001007
3694 024220
    024220 012737 050004 044746
    024226 012737 045646 044752
    024234 104130
3695
3696
3697
3698 024236 005037 001130
3699 024242 112777 000010 155064
3700 024250 105737 001130
3701 024254 001010
3702 024256 077004
3703 024260 012737 047746 044746
    024266 012737 045206 044752
    024274 104132
3704 024276 032737 000010 001130
3705 024304 001404

;*****
;*TEST 10.3  MODEM CONTROLS
;*****
T10.3::
CHB=  #5          ; REDEFINE R5 AS CHAN B POINTER.
;
; NEC.MDM:
MOV     #. +6, $LPCON    ;; LOOP HERE ON ERROR.
MTPS   #PR4             ; RAISE CPU.
MOV     CHA, CHB
ADD     #10, CHB        ; SET CHAN B POINTER.
1$:    MOVB  #<3*BIT3>, (CHA) ; <CHAN RESET>.
        MOVB  #<3*BIT3>, (CHB)
        CALL  D100        ; DELAY 100US (MORE OR LESS).
        MOVB  #<2*BIT3>!1, (CHA) ; <RESET-EXT>, POINT TO CR1...
        MOVB  #<2*BIT3>!1, (CHB)
        MOVB  #5, (CHA)    ; ...AND SET <CAV!INT-ON-EXT>.
        MOVB  #5, (CHB)
2$:    MOV     #NECM1, EMPRE ; SET ERROR PREFIX.
        MOV     #NEC.EX, NECV ; SET THE VECTOR.
        CLR     R3          ; *** DEBUG, COUNT INTERRUPTS IN R3.
        CLR     R0          ; 250 MSEC KEEP-ALIVE TIMER.
        MTPS   R0         ; LOWER CPU.
;
; FIRST, TEST <REQ-TO-SEND>, <CLR-TO-SEND> AND <RCVR READY>.
;
3$:    CLR     $BDDAT      ; CLEAR A SPACE FOR SRO(A)...
        CLR     $BDDAT+2  ; ...AND SRO(B).
        MOVB   #5, (CHA)  ; POINT TO CR5(A)...
        MOVB   #2, (CHA)  ; ...AND SET <REQ-TO-SEND>.
4$:    TSTB   $BDDAT      ; SHOULD GET SRO(A) STATUS INTERRUPT.
        BNE   5$         ; BR IF SO.
        SOB   R0, 4$
        MOV   #NEC6, EMADR ;; \\
        MOV   #NOSIG, ESADR ;; >> ERROR 130, STATUS INTERRUPT NOT RECEIVED
        ERROR+130        ;; //
5$:    BIT    #BITS, $BDDAT ; SHOULD HAVE <CLR-TO-SEND>...
        BEQ   6$
        BIT    #BIT3, $BDDAT ; ...AND <RCVR-READY>.
        BNE   7$
6$:    MOV     #NEC7, EMADR ;; \\
        MOV     #EF12, ESADR ;; >> ERROR 131, STATUS WRONG WITH REQ-TO-SEND SET
        ERROR+131        ;; //
;
; NEXT, TEST CSRA<TT108/2> AND <DATA-MODE> BITS.
;
7$:    CLR     $BDDAT+2
        MOVB   #BIT3, @CSRA ; SET <TT108/2>.
8$:    TSTB   $BDDAT+2    ; SHOULD GET SRO(B) STATUS INTERRUPT.
        BNE   9$         ; BR IF SO.
        SOB   R0, 8$
        MOV   #NEC6, EMADR ;; \\
        MOV   #NOSIG, ESADR ;; >> ERROR 132, STATUS INT NOT RECEIVED
        ERROR+132        ;; //
9$:    BIT    #BIT3, $BDDAT+2 ; SHOULD HAVE <DATA-MODE>...
        BEQ   10$

```

```

3706 024306 032737 000040 001130      BIT    #BIT5,#BDDAT+2 ;...AND NOT <INCOMING CALL>.
3707 024314 001407                      BEQ    11$
3708 024316                                10$:
    024316 012737 050040 044746      MOV    #NEC8,EMADR    ;;\
    024324 012737 045646 044752      MOV    #EF12,ESADR   ;; >> ERROR 133, STATUS WRONG WITH TT108/2 SET
    024332 104133                      ERROR+133            ;;//

3709                                ;
3710                                ; NEXT, TEST CSRA<TERM-IN-SERV>, CSRB<TT142> AND <INCOMING-CALL>.
3711                                ;
3712 024334 005037 001130      11$: CLR    $BDDAT+2
3713 024340 052777 000020 154766      BIS    #BIT4,@$CSRA ; SET <TERM-IN-SERV>.
3714 024346 105737 001130      12$: TSTB  $BDDAT+2     ; SHOULD GET ANOTHER SRO(B) CHANGE.
3715 024350 001010      BNE    13$
3716 024354 077004      SOB    R0,12$
3717 024356 012737 047746 044746      MOV    #NEC6,EMADR    ;;\
    024364 012737 045206 044752      MOV    #NOSIG,ESADR  ;; >> ERROR 134, STATUS INT NOT RECEIVED
    024372 104134      ERROR+134            ;;//

3718 024374 032777 000001 154734 13$: BIT    #BIT0,@$CSRB ; SHOULD HAVE <TT142> IN CSRB...
3719 024402 001404      BEQ    14$
3720 024404 032737 000040 001130      BIT    #BIT5,#BDDAT+2 ;...AND <INCOMING-CALL>.
3721 024412 001007      BNE    15$
3722 024414                                14$:
    024414 012737 050110 4746      MOV    #NEC9,EMADR    ;;\
    024422 012737 045646 044752      MOV    #EF12,ESADR  ;; >> ERROR 135, STATUS WRONG WITH TERM-IN-SERV SET
    024430 104135      ERROR+135            ;;//

3723                                ;
3724                                ; FINALLY, TURN ALL THAT STUFF OFF !!!!!
3725                                ;
3726 024432 042777 000010 154674 15$: BIC    #BIT3,@$CSRA ; CLEAR <TT108/2>
3727 024440 004737 037564      CALL   D100          ; DELAY.
3728 024444 005077 154664      CLR    @$CSRA       ; CLEAR <TERM-IN-SERV>.
3729 024450 004737 037564      CALL   D100          ; DELAY.
3730 024454 112714 000005      MOVB   #5,(CHA)     ; POINT TO CR5(A)...
3731 024460 112714 000000      MOVB   #0,(CHA)    ; ...AND CLEAR <REQ-TO SEND>.
3732 024464 004737 037564      CALL   D100          ; DELAY.
3733 024470 032737 000050 001126      BIT    #BITS:BIT3,$BDDAT ; <CS:RR> IN SRO(A) SHOULD BE CLEAR ..
3734 024476 001004      BNE    16$
3735 024500 032737 000050 001130      BIT    #BITS:BIT3,$BDDAT+2 ;...AND <IC!DM> IN SRO(B) AS WELL.
3736 024506 001407      BEQ    17$
3737 024510                                16$:
    024510 012737 050165 044746      MOV    #NEC10,EMADR  ;;\
    024516 012737 045646 044752      MOV    #EF12,ESADR  ;; >> ERROR 136, STATUS WRONG WITH EVERYTHING OFF
    024524 104136      ERROR+136            ;;//

3738 024526 112714 000030      17$: MOVB   #<3*BIT3>,(CHA) ; ALL DONE, RESET THE CHIP...
3739 024532 112715 000030      MOVB   #<3*BIT3>,(CHB)
3740 024536 012737 002542 000070 NEC.END: MOV  #DISMISS,NECV ;...AND THE VECTOR.
3741 024544 000416      BR     TST11        ;NEXT TEST.

3742                                ;
3743                                ; EXTERNAL (MODEM CONTROLS) INTERRUPT HANDLER.
3744                                ;
3745 024546 005203      NEC.EX: INC    R3          ; *** DEBUG, COUNT THE INTERRUPTS.
3746 024550 112714 000020      MOVB   #<2*BIT3>,(CHA) ; COMMAND <RESET-EXTERNAL>.
3747 024554 112715 000020      MOVB   #<2*BIT3>,(CHB) ; DITTO
3748 024560 116437 177774 001126      MOVB   STAT(CHB),$BDDAT ; GET SRO(A)...
3749 024566 116537 177774 001130      MOVB   STAT(CHB),$BDDAT+2 ;...AND SRO(B).
3750 024574 112714 000070      MOVB   #<7*BIT3>,(CHA) ; SIGNAL <END-OF-INTERRUPT>.
3751 024600 000002      RTI

```

3754

```

*****
;*TEST 11 PARALLEL I/O PORT Z8036
*****
TST11: SCOPE
024602 000004
024604 012706 001100
024610 012737 177777 002426
024616 005037 044742
024622 032777 000400 154310
024630 001006
024632 005777 154472
024636 000240
024640 102002
024642 000137 026220 30050$: JMP TST11 ; SKIP NEXT IF OK.
024646 104415 024654 30051$: T$NAME, .+4 ; BYPASS.
024652 000415 BR 30052$ ; PRINT TEST NUMBER AND NAME...
024654 040 040 120 .ASCIZ \ PARALLEL I/O PORT Z8036\ ; ...AND SKIP OVER THE ASCII.
.EVEN

30052$:
;
; ZILOG 8036 CHIP HAS 48 REGISTERS.
; RESET THE CHIP AND VERIFY THAT ALL REGISTERS RETURN ZERO.
;
3755
3756
3757
3758
3759 024706 012737 024714 001110 MOV #.+6,$LPERR ; LOOP HERE ON ERROR.
3760 024714 012737 025044 002426 MOV #4$,TRAP4X ; SET TRAP RETURN.
3761 024722 012704 000060 MOV #48,$R4 ; CHECK 48 REGISTERS...
3762 024726 013705 001330 MOV $PIO,R5 ; ...STARTING HERE.
3763 024732 005037 001126 CLR $BDDAT
3764 024736 012737 000001 001124 MOV #1,$GDDAT
3765 024744 112715 000001 MOVB #1,(R5) ; SET "CHIP-RESET" IN MICR.
3766 024750 000240 240 ; IF IT TRAPS, WE'RE DEAD.
3767 024752 111537 001126 MOVB (R5),$BDDAT ; READ IT BACK...
3768 024756 023737 001124 001126 CMP $GDDAT,$BDDAT ; ...SHOULD SEE THE 1 BIT.
3769 024764 000410 BR 2$ ; CHECK IT AT 2$.
3770
3771 024766 005037 001124 1$: CLR $GDDAT ; EXPECT 0 FROM ALL THE REST.
3772 024772 112715 177777 MOVB #-1,(R5) ; WRITES SHOULD BE IGNORED...
3773 024776 000240 240
3774 025000 111537 001126 MOVB (R5),$BDDAT ; ...AND READS RETURN ZERO.
3775 025004 000240 240
3776 025006 001411 2$: BEQ 3$ ; BR IF OK.
3777 025010 010537 001120 MOV R5,$GDADR
3778 025014 012737 050257 044746 MOV #PI02,EMADR ;\
025022 012737 045210 044752 MOV #EF2,ESADR ; >> ERROR 137, RESET STATE INCORRECT
025030 104137 ERROR+137 ;//
3779 025032 005725 3$: TST (R5)+ ; NEXT ADDRESS...
3780 025034 077424 SOB R4,1$ ; ...AND LOOP 'TIL DONE.
3781 025036 005037 002426 CLR TRAP4X ; THEN, RESET TRAP CATCHER...
3782 025042 000413 BR PIOT ; ...AND GO ON.
3783
3784 025044 010537 001120 4$: MOV R5,$GDADR
3785 025050 012737 050232 044746 MOV #PI01,EMADR ;\
025056 012737 045230 044752 MOV #LA16,ESADR ; >> ERROR 140, BUS ERROR ON PIO ADDRESS
025064 104140 ERROR+140 ;//
3786 025066 000137 026046 JMP ENDPPIO ; NEXT TEST.

```

3788

025072

```

3789
3790 - 000000
3791 000002
3792 000010
3793 000070
3794 000024
3795 000054
3796 000040
3797
3798
3799
3800
3801 025072 013705 001330
3802 025076 142715 000001
3803 025102 00504
3804 025104 012703 000060
3805 025110 005024
3806 025112 077302
3807 025114 112715 000200
3808 025120 112765 000104 000010
3809 025126 112765 000160 000002
3810
3811
3812 025134 062705 000024
3813 025140 012737 050311 044742
3814 025146 112737 000061 050323
3815 025154 012737 025162 001110
3816 025162 012706 001100
3817 025166 010537 001120
3818 025172 106427 000200
3819 025176 012737 025306 000104
3820 025204 005000
3821 025206 012701 000004
3822 025212 112715 000040
3823 025216 112715 000306
3824 025222 000240
3825 025224 132715 000001
3826 025230 001011
3827 025232 077004
3828 025234 012737 050326 044746
    025242 012737 045206 044752
    025250 104141
3829 025252 000445
3830 025254 132715 000001
3831 025260 001423
3832 025262 077004
3833 025264 077105
3834 025266 012737 050342 044746
    025274 012737 045206 044752
    025302 104142
3835 025304 000430
3836 025306 000240
3837 025310 012737 050356 044746

```

```

;*****
;*TEST 11.1 COUNTER/TIMERS
;*****
T11.1::
;
MIC= 000 ; MASTER INTERRUPT CONTROL REG.
MCC= 002 ; MASTER CONFIGURATION CONTROL REG.
CTV= 010 ; TIMER(S) INTERRUPT VECTOR REG.
CTMS= 070 ; 1ST (OF 3) TIMER MODE SPEC REG.
CTCS= 024 ; 1ST (OF 3) TIMER CMD/STATUS REG.
CTTC= 054 ; 1ST (OF 6) TIME CONSTANT REG (3 PAIR).
CTCC= 040 ; 1 (OF 6) CURRENT COUNT REG (3 PAIR).
;
; VERIFY THAT EACH TIMER FUNCTIONS AS ADVERTISED.
; CHIP IS "RESET" ON ENTRY AND EXIT.
;
PIOT: MOV $PIO,R5 ; GET BASE ADDRESS.
      BICB #1,(R5) ; CLEAR "CHIP-RESET"...
      MOV R5,R4
      MOV #48.,R3
1$: CLR (R4)+ ; ...AND CLEAR ALL REGISTERS.
   SOB R3,1$
      MOVB #200,(R5) ; SET "MASTER INT ENAB".
      MOVB #PEVNT,CTV(R5) ; SET TO VECTOR THRU "PEVNT".
      MOVB #160,MCC(R5) ; ENABLE ALL 3 TIMERS...
; ...CTMS = 0 = PULSE-OUT, ONCE-ONLY...
; ...CTTC - 0 = TIME K 65536.
; NOW RAISE R5 TO 1ST TIMER CSR.
; SET ERROR PREFIX...
; ...AND INIT ASCII TIMER NUMBER.
; LOOP HERE ON ERROR.
PIOT1: MOV R5,$GDADR ; SAVE CURRENT CSR ADDRESS.
      MTPS #PR4 ; RAISE CPU TO PIO LEVEL.
      MOV #3$,PEVNT ; SET VECTOR.
      CLR R0
      MOV #4,R1 ; SET A 1 SECOND KEEP-ALIVE TIMER.
      MOVB #40,(R5) ; CLEAR "IUS!IP"...
      MOVB #306,(R5) ; ...AND SET "IE!GCB!TCB"...
      240 ; ...TIMER SHOULD TAKE-OFF !!
1$: BITB #1,(R5) ; TIMER RUNNING (CIP=1) ??
   BNE 2$ ; BR IF SO.
   SOB R0,1$
      MOV #PIO3,EMADR ;:\
      MOV #NOSIG,ESADR ; >> ERROR 141, TIMER DIDN'T START
      ERROR+141 ;://
      BR 7$ ; ABORT.
2$: BITB #1,(R5) ; TIMER DONE (CIP=0) ??
   BEQ 4$ ; BR IF SO.
   SOB R0,2$
   SOB R1,2$
      MOV #PIO4,EMADR ;:\
      MOV #NOSIG,ESADR ; >> ERROR 142, TIMER NEVER STOPS.
      ERROR+142 ;://
      BR 7$ ; ABORT.
3$: NOP ; INTERRUPT -- SHOULD HAVE BEE'V MASKED.
   MOV #PIO5,EMADR ;:\

```

```

025316 012737 045206 044752      MOV  #NOSIG,ESADR  ;; >> ERROR 143, PIO INTERRUPT LEVEL INCORRECT.
025324 104143      ERROR+143          ;://
3838 025326 000416      BR    6$
3839
3840 025330 012737 025364 000104 4$:5$: MOV  #6$,PEVNT      ; NOW, CHANGE VECTOR...
3841 025336 106427 000000      MTPS #PRO          ; ...LOWER THE CPU...
3842 025342 000240      240                ; ...AND LET INTERRUPT COME IN.
3843 025344 012737 050416 044746      MOV  #PIO6,EMADR   ;:\\
025352 012737 045206 044752      MOV  #NOSIG,ESADR ;: >> ERROR 144, TIMER INTERRUPT NOT RECEIVED.
025360 104144      ERROR+144          ;://
3844 025362 000401      SKP1
3845
3846 025364 022626      6$: CMP  (SP)+,(SP)+  ; INTERRUPT -- FIX THE STACK.
3847 025366 012715 000040      7$: MOV  #40,(R5)    ; CLEAR "IUS!IP!GCB"...
3848 025372 012715 000160      MOV  #160,(R5)     ; ...AND "IE" (TIMER OFF).
3849 025376 005725      TST  (R5)+         ; BUMP TO NEXT TIMER CMD/STAT...
3850 025400 105237 050323      INCB PITX          ; ...BUMP ASCII TOO...
3851 025404 123727 050323 0.0063      CMPB PITX,#'3
3852 025412 101665      BLOS PIOT1        ; ...AND LOOP 'TIL ALL 3 DONE.
3853
3854 025414 112777 000001 153706      MOVB #1,@$PIO      ; THEN, RESET THE CHIP...
3855 025422 012737 002542 000104      MOV  #DISMISS,PEVNT ; ...PEVNT VECTOR...
3856      ; ...AND FALL THRU TO DATA TEST.

```

3858

025430

3859

3860

3861

3862

3863

3864

3865

3866

3867

3868

3869

3870

3871

3872

3873

3874

3875

3876

3877

3878

3879

3880

3881

3882

3883

3884

3885

3886

3887

3888

3889

3890

3891

3892

3893

3894

3895

3896

3897

3898

3899

3900

3901

3902

3903

3904

3905

3906

3907

3908

3909

3910

000004

000020

000032

000100

000102

000006

000022

000034

000120

000122

000014

000036

000140

164377

165400

013705

142715

012765

012765

012737

012737

012737

112765

112765

112765

112765

112765

112765

112765

112765

012765

012737

112765

112765

112765

112765

012765

012765

025616

012737

012706

106427

112765

012704

012700

005024

077002

012704

012703

005037

0000001

000160

000006

000160

000164

000100

000102

000020

000120

000122

000022

000022

000140

000417

050464

000100

000020

000020

000120

000140

001110

001100

000200

000377

061022

000020

061022

037600

037576

```
*****
;TEST 11.2 DATA PORTS
*****
T11.2::
```

```
PAV= 004 ; PORT A VECTOR...
PACS= 020 ; ...CMD/STATUS...
PADB= 032 ; ...DATA BUFFER...
PAMS= 100 ; ...MODE...
PAHS= 102 ; ...AND HANDSHAKE SPEC.
PBV= 006 ; PORT B VECTOR...
PBCS= 022 ; ...CMD/STATUS...
PBD8= 034 ; ...DATA BUFFER...
PBMS= 120 ; ...MODE...
PBHS= 122 ; ...AND HANDSHAKE SPEC.
PCDD= 014 ; PORT C DATA DIRECTION...
PCDB= 036 ; ...AND BUFFER (HANDSHAKE BITS).
XCVR= 140 ; 1ST REGISTER ABOVE THE CHIP HOLDS...
AI.B0= <350*BIT8>!377 ; ...XCVR CONTROL BITS TTL/A(IN)!B(OUT).
AO.BI= <353*BIT8>!0 ; DITTO TTL/A(OUT)!B(IN).
```

```
;
; *** DATA PORT(S) TEST REQUIRES EXTERNAL LOOP-BACK CONNECTOR ***
; CHIP IS "RESET" ON ENTRY AND EXIT.
```

```
PIOD: MOV $PIO,R5 ; GET B' SE ADDRESS.
BICB #1,(R5) ; CLEAR "CHIP-RESET".
MOV #PIOAV,PAV(R5) ; SET VECTORS.
MOV #PIOBV,PBV(R5)
MOV #AI,PIOAV
MOV #BI,PIOBV
PIOD1: MOV #PORTBA,EMPRE ; SET ERROR PREFIX (B=>A).
MOVB #100,PAMS(R5) ; SET PORT A INPUT...
MOVB #000,PAHS(R5) ; ...INTERLOCKED HS...
MOVB #300,PACS(R5) ; ...INT ENABLED.
MOVB #200,PBMS(R5) ; SET PORT B OUTPUT...
MOVB #000,PBHS(R5) ; ...INTERLOCKED HS...
MOVB #300,PBCS(R5) ; ...INT ENABLED.
MOVB #200,PBCS(R5) ; SET OUTPUT "IP" TO PRIME THE LOOP.
MOV #AI.B0,XCVR(R5) ; SET XCVR CONTROL BITS.
BR PIOD3
PIOD2: MOV #PORTAB,EMPRE ; CHANGE ERROR PREFIX (A=>B).
MOVB #200,PAMS(R5) ; CHANGE PORT A TO OUTPUT...
MOVB #200,PACS(R5) ; ...AND SET "IP".
MOVB #100,PBMS(R5) ; CHANGE PORT B TO INPUT.
MOV #AO.BI,XCVR(R5) ; CHANGE XCVR CONTROL BITS.
PIOD3: MOV #.+6,$LPERR ; ; LOOP HERE ON ERROR.
MOV #STACK,SP
MTPS #PR4 ; RAISE CPU.
MOVB #377,PCDD(R5) ; PORT C DIRECTION ALWAYS "IN".
MOV #BUFR1,R4
MOV #16.,R0
1$: CLR (R4)+ ; CLEAR SOME FREE SPACE.
SOB R0,1$
MOV #BUFR1,R4 ; RCV BUFFER POINTER => R4.
MOV #FLT10,R3 ; XMT BUFFER POINTER => R3.
CLR BYTES ; CLEAR BYTE COUNTERS.
```

```

3911 025672 005037 001124      CLR      $GDDAT
3912 025676 005037 001126      CLR      $BDDAT
3913 025702 112715 000200      MOVB     #200,(R5)          ; SET "MIE"...
3914 025706 112765 000224 000002      MOVB     #224,MCC(R5)      ; ...AND ENABLE ALL THREE PORTS.
3915 025714 116500 000032      MOVB     PADB(R5),R0       ; ??? DUMMY READ (FLUSH)...
3916 025720 116500 000034      MOVB     PBDB(R5),R0      ; ...BOTH DB'S ???
3917 025724 106427 000000      MTPS     #PRO             ; LOWER CPU...
3918 025730 005002                CLR      R2                ; ...DATA SHOULD START TO FLOW.
3919 025732 023727 037576 011022 2$:      CMP      BYTES,#<18.*BIT8>!18. ; 18 BYTES TRANSFERRED ??
3920 025740 001417                BEQ      3$                ; BR IF SO.
3921 025742 077205                SOB      R2,2$            ; HANG-AROUND...
3922 025744 000240                240                          ; ...FOR A WHILE.
3923 025746 113737 037576 001124      MOVB     BYTES,$GDDAT      ; SENT...
3924 025754 113737 037577 001126      MOVB     BYTES+1,$BDDAT    ; ...VS RECEIVED.
3925 025762 012737 050503 044746      MOV      #PI07,EMADR       ; \
025770 012737 045556 044752      MOV      #EF10,ESADR      ; >> ERROR 145. LOOP TIME-OUT. DATA XFER INCOMPLETE
025776 104145                ERROR+145                  ; //
3926 026000 000240                3$:      NOP
3927 026002 000240                NOP
3928 026004 105765 000100      TSTB     PAMS(R5)          ; PORT A SENDING ??
3929 026010 100404                BMI      4$                ; ALL DONE IF SO.
3930 026012 105015                CLRB     (R5)              ; ELSE, CLEAR MASTER...
3931 026014 105065 000002      CLRB     MCC(R5)          ; ...CONTROL REGISTERS...
3932 026020 000657                BR       PI0D2             ; ...AND GO 'ROUND ONCE MORE.
3933
3934 026022 112715 000001      4$:      MOVB     #1,(R5)          ; DONE, RESET THE CHIP...
3935 026026 005065 000140      CLR      XCVR(R5)         ; ...XCVR CONTROLS...
3936 026032 012737 002542 000160      MOV      #DISMISS,PIOAV    ; ...AND RESTORE VECTORS.
3937 026040 013737 000160 000164      MOV      TOAV,PIOBV
3938 026046 000464                ENDPPIO:  BR      TST12      ; ...AND TAKE NEXT TEST.
026046 000464
3939
3940      ; PORT INTERRUPT HANDLER(S).
3941
3942      ;
3943 026050 012700 000032      AI:      .ENABL  LSB
3944 026054 105765 000100      MOV      #PADB,R0          ; PORT A -- GET DATA BUFFER INDEX.
3945 026060 100407                TSTB     PAMS(R5)          ; CHECK DIRECTION.
3946 026062 000423                BMI      1$                ; OUTPUT
3947                                BR       3$                ; INPUT
3948 026064 012700 000034      BI:      MOV      #PBDB,R0          ; PORT B -- DITTO.
3949 026070 105765 000120      TSTB     PBMS(R5)
3950 026074 100401                BMI      1$                ; OUTPUT
3951 026076 000415                BR       3$                ; INPUT
3952
3953 026100 060500                1$:      ADD      R5,R0             ; OUTPUT -- SET DB POINTER.
3954 026102 123727 037576 000022      CMPB     BYTES,#18.        ; ANY BYTES LEFT ??
3955 026110 001404                BEQ      2$                ; BR IF NOT.
3956 026112 112310                MOVB     (R3)+,(R0)        ; ELSE, SEND NEXT BYTE...
3957 026114 105237 037576      INCB     BYTES             ; ...BUMP THE COUNT...
3958 026120 000433                BR       5$                ; ...AND RETURN.
3959 026122 112760 000240 177766 2$:      MOVB     #240,-12(R0)     ; WHEN XMIT DONE, CLEAR "IP"...
3960 026130 000427                BR       5$                ; ...AND RETURN.
3961
3962 026132 060500                3$:      ADD      R5,R0             ; INPUT -- SET DB POINTER.
3963 026134 111014                MOVB     (R0),(R4)         ; STORE THE BYTE...
3964 026136 105237 037577      INCB     B'IES+1          ; ...AND BUMP THE COUNT.

```

N8

KXJ11 CA FUNCTIONAL TEST
T11.2 DATA PORTS

MACRO Y05.02 Thursday 03 Apr-86 14:11 Page 103

396	026142	121464	156556			CMPB	(R4),FLT10 BUFR1(R4) ; COMPARE DATA XMTD/RCVD.
396	026146	001416				BEQ	4\$; BR IF IT'S RIGHT.
3967	026150	010437	001120			MOV	R4,\$GDADR
3968	026154	111437	001126			MOVB	(R4),\$BDDAT
3969	026160	116437	156556	001124		MOVB	FLT10-BUFR1(R4),\$GDDAT
3970	026166	012737	047356	044746		MOV	#PI08,EMADR ;:\
	026174	012737	045210	044752		MOV	#EF2,ESADR ;: >> ERROR 146, DATA INCORRECT
	026202	104146					ERROR+146 ;://
3971	026204	000240			4\$:	NOP	
3972	026206	005204				INC	R4 ; BUMP RCVR POINTER.
3973	026210	112760	000140	177766	5\$:	MOVB	#140, 12(R0) ; COMMON EXIT, CLEAR "IUS"...
3974	026216	000002				RTI	...AND RETURN.
3975						.DSABL	LSB
3976		000000				PWRFAIL	= 0

3978
3979

026220 000004
026222 032737 000340 177524
026230 001410
026232 012706 001100
026236 012737 177777 002#26
026244 005037 044742
026250 000402
026252 000137 027412
026256 104415 026264
026262 000411
026264 040 040 121

026306
3980
3981
3982
3983
3984 026306 012737 026314 001110
3985
3986
3987
3988 026314 106427 000240
3989 026320 005037 002426
3990 026324 013705 001314
3991 026330 010537 001120
3992 026334 011537 001124
3993 026340 005000
3994 026342 077001
3995 026344 012777 000144 152744
3996 026352 011537 001126
3997
3998
3999
4000
4001
4002
4003 000130
4004 040000
4005 020000
4006 010000
4007 002000
4008
4009 026356 052737 040000 001124
4010 026364 032737 040000 001126
4011 026372 001007
4012 026374 012737 027157 044746
026402 012737 045210 044752
026410 104147
4013 026412 042737 040000 001124
4014 026420 052737 030000 001124
4015 026426 005037 001352
4016 026432 052715 030000
4017 026436 106427 000000

```

;QIRAK2.MAC 19-NOV 85
;*****
;*TEST 12 Q-BUS INTERRUPT
;*****
TST12: SCOPE
        BIT    #340,#K2CSRC    ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
        BEQ    30053$          ;; SKIP THIS TEST IF IT IS
        MOV    #STACK,SP      ;; SET A CLEAN STACK.
        MOV    #1,TRAP4X      ;; DISMISS BUS-ERRORS.
        CLR    EMPRE          ;; CLEAR ERROR PREFIX.
        BR     30054$          ;; SKIP NEXT.
30053$: JMP    TST13           ;; BYPASS.
30054$: T$NAME, .+4           ;; PRINT TEST NUMBER AND NAME...
        BR     30055$          ;;...AND SKIP OVER THE ASCII.
        .ASCIZ \ Q BUS INTERRUPT\
        .EVEN
30055$:
;
; VERIFY THAT THE QIR CAN INTERRUPT THE ARBITER, AND THAT
; THE POST-INTERRUPT PROCESS WORKS AS ADVERTISED.
;
        MOV    #.+6,$LPERR    ;; LOOP HERE ON ERROR.
;
; FIRST, WRITE THE QIR AND SEE WHAT HAPPENS.
;
2$:     MTPS   #PR5            ; RAISE CPU.
        CLR    TRAP4X
        MOV    $CSR,R5        ; CSR POINTER (TO SPEED THINGS UP).
        MOV    R5,$GDADR      ; CSR ADDRESS...
        MOV    (R5),$GDDAT    ;...AND EXP'D STATUS.
        CLR    R0             ; TIMER, 200MS (MORE OR LESS).
        SOB   R0              ; DELAY -- ENSURE TTY OUTPUT IS DONE.
        MOV   #QIRV,#QIR     ; WRITE A VECTOR IN QIR<9:2>.
        MOV   (R5),$BDDAT    ; GET CSR, QIREQ SHOULD BE SET.
;
; NOW, BRQ4 SHOULDN'T GET POSTED UNTIL/UNLESS THE ENABLE IS SET.
; ON BIACK, REQUEST GETS CLEARED AND IF ENABLED POSTS A
;
; LOCAL INTERRUPT THRU VECTOR 130 (AT PRI 4).
;
BIACKV= TBIACK
BRQ=    BIT14
ACKEN=  BIT13
BRQEN=  BIT12
BIEN=   BIT10
;
QIT1:   BIS    #BRQ,$GDDAT    ; SET EXPECTED STATUS.
        BIT    #BRQ,$BDDAT
        BNE   1$             ;...AND BR IF REQUEST IS SET.
        MOV   #QI2,EMADR     ;;\
        MOV   #EF2,ESADR     ;; >> ERROR 147, BRQ DIDN'T SET
        ERROR+147           ;;//
1$:     BIC    #BRQ,$GDDAT
        BIS    #BRQEN!ACKEN,$GDDAT ; SET EXPECTED STATUS.
        CLR   #BFLAG        ; SAY "EXPECTING TRAP THROUGH 130." ;FXBJC
        BIS   #BRQEN!ACKEN,(R5) ; SET QIR AND QIACK ENABLES (POST BRQ4)...
        MTPS  #PRO          ;...LOWER THE CPU...
    
```

```

4018 ; LOOK FOR ARBITER'S 'Y SIGNIFYING IT HAS RESPONDED. ;FX8JC
4019 026442 105777 152476 2$: TSTB @TKS ; WAIT FOR CHAR FROM ARBITER. ;FX8JC
4020 026446 100375 BPL 2$ ; KEEP LOOKING. ;FX8JC
4021 026450 017704 152472 MOV @TKB,R4 ; GET THE CHARACTER. ;FX9JC
4022 026454 120427 000131 CMPB R4,#'Y ; DID ARBITER SAY YES? ;FX8JC
4023 026460 001370 BNE 2$ ; IF NO, KEEP LOOKING. ;FX8JC
4024 026462 005000 CLR R0 ; ARBITER SAYS IT RESPONDED. ;FX8JC
4025 026464 005737 001352 3$: TST @BFLAG ; DID TRAP OCCUR YET? ;FX8JC
4026 026470 001023 BNE 4$ ; YES, CONTINUE. ;FX8JC
4027 026472 077004 SOB R0,3$ ; KEEP LOOKING FOR A WHILE. ;FX8JC
4028 026474 042715 030000 BIC #BRQEN!ACKEN,(R5); DISABLE INTERRUPTS WHILE REPORTING. ;FX8JC
4029 026500 012737 000001 001352 MOV #1,@BFLAG ; AND SAY THEY'RE NOT EXPECTED EITHER. ;FX8JC
4030 026506 012737 027257 044746 MOV #QI4,EMADR ;:\\
026514 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 150, INT-ON-BIACK NOT RECEIVED
026522 104150 ERROR+150 ;://
4031 026524 077001 000005 077001 .WORD SOB+1, RESET, SOB+1 ;: LOCAL-RESET.
4032 026532 052777 002000 152554 BIS #2000,@CSR ; BUT ENABLE TRAP ON Q-BUS RESET. ;FX8JC
4033 026540 011537 001126 4$: MOV (R5),#BDDAT ; GET FINAL STATUS...
4034 026544 032737 040000 001126 BIT #BRQ,#BDDAT ;...REQUEST SHOULD BE GONE.
4035 026552 001407 BEQ 5$ ; BR IF SO. ;FX8JC
4036 026554 012737 027210 044746 MOV #QI3,EMADR ;:\\
026562 012737 045210 044752 MOV #EF2,ESADR ;: >> ERROR 151, BREQ STILL SET, BIACK NOT RECEIVED
026570 104151 ERROR+151 ;://

4037 ;
4038 ; DO IT ONCE MORE USING THE ALTERNATE VECTOR.
4039 ; ARBITER WILL RESPOND WITH A "RESET", WHICH SHOULD
4040 ; CAUSE A "BUS-INIT" AND LOCAL TRAP THRU VECTOR 24.
4041 ;
4042 026572 5$: CMPB #APTENV,$ENV ; RUNNING IN APT MODE? ;FX8JC
4043 026572 122737 000001 001206 BNE 55$ ; NO, CONTINUE ;FX9JC
4044 026600 001007 TST $PASS ; IF APT AND NOT FIRST PASS ... ;FX9JC
4045 026602 005737 001174 BNE QIEND ; ... SKIP THIS PART OF THE TEST. ;FX9JC
4046 026606 001142 BIT #<16*BIT4>,@TCID ; IF ID 0 OR 1...
4047 026610 032777 000340 152522 BEQ QIEND ;...JUST GET OUT.
4048 026616 001536 55$: BIS #BIT06,K2CSRJ ; ARM B-INIT TRAP.
4049 026620 052737 000100 177540 BIC #<BRQEN!ACKEN>,(R5) ;...DISABLE IACK...
4050 026626 042715 030000
4051
4052 026632 106427 000340 MTPS #PR7 ; TRAP 24 SHOULD BE NON-MASKABLE.
4053 026636 004737 001462 JSR PC,MMU ;INIT MMU REGISTERS
4054 026642 052737 140000 177776 BIS #140000,@PSW ;SET USER MODE
4055 026650 012706 140000 MOV #140000,R6 ;SET UP USER STACK
4056 026654 013746 000404 MOV @404,-(SP)
4057 026660 013746 000406 MOV @406,-(SP)
4058 026664 012737 000137 000404 MOV #137,@404 ;
4059 026672 012737 002614 000406 MOV #RSTRAP,@406 ;
4060 026700 012737 177654 172356 MOV #177654,@KIPAR7;
4061 026706 052737 000001 177572 BIS #1,@MMRO ;TURN ON MMU
4062 026714 005037 001344 CLR @RFLAG ; SAY "EXPECTING A TRAP THROUGH 24." ;FX8JC
4063
4064 026720 012777 000150 152370 MOV #QIRV1,@QIR ; POST Q-BUS REQUEST.
4065 026726 000240 NOP
4066 026730 032737 040000 177530 BIT #BIT14,@K2CSRD ; IS INTERRUPT POSTED?
4067 026736 001007 BNE 15$
4068 026740 012737 027371 044746 MOV #DPR11,EMADR ;:\\
026746 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 152, QIR NOT WRITTEN
026754 104152 ERROR+152 ;://

```

D9

KXJ11-CA FUNCTIONAL TEST
T12 Q-BUS INTERRUPT

MACRO Y05.02 Thursday 03-Apr 86 14:11 Page 106

```

4069 026756 052715 012000      15$: BIS      #<BIT12!BIEN>,(R5)      ;ALLOW QBUS INTERRUPT
4070                               ; WAIT FOR ARBITER TO SEND A 'Y SIGNIFYING IT HAS RESPONDED.      ;FX8JC
4071 026762 105777 152156      6$: TSTB     @!TKS      ; WAIT FOR CHAR FROM ARBITER.      ;FX8JC
4072 026766 100375              BPL      6$           ; KEEP LOOKING.      ;FX8JC
4073 026770 017705 152152      MOV      @!TKB,R5     ; GET THE CHARACTER.      ;FX9JC
4074 026774 120527 000131      CMPB     R5,#'Y      ; IS IT A YES?      ;FX8JC
4075 027000 001370              BNE      6$           ; IF NOT, KEEP LOOKING.      ;FX8JC
4076 027002 012737 177600 172356 MOV      #177600,@#KIPAR7;
4077 027010 042737 000001 177572 BIC      #1,@#VRO     ;
4078 027016 012637 000406      MOV      (SP)+,@#406  ;RESTORE VECTORS
4079 027022 012637 000404      MOV      (SP)+,@#404  ;
4080 027026 042737 140000 177776 BIC      #140000,PSW  ;SET KERNAL MODE
4081
4082 027034 005000              CLR      RO           ; ARBITER HAS EXECUTED A BUS-RESET      ;FX8JC
4083 027036 005737 001344      7$: TST      @#RFLAG   ; DID TRAP OCCUR YET?      ;FX8JC
4084 027042 001024              BNE      QIEND        ; YES, CONTINUE.      ;FX8JC
4085 027044 077004              SOB     RO,7$        ; KEEP LOOKING FOR A WHILE.      ;FX8JC
4086 027046 042737 002000 177530 BIC      #2000,@#177530 ; DISABLE TRAPS WHILE REPORTING ERROR. ;FX8JC
4087 027054 012737 000001 001344 MOV      #1,@#RFLAG   ; AND SAY THEY'RE NOT EXPECTED EITHER. ;FX8JC
4088 027062 012737 027323 044746 MOV      #QI5,EMADR   ;;\
      MOV      #NOSIG,ESADR ;; >> ERROR 153, BRESET TRAP THRU 24 DIDN'T HAPPEN
      ERROR+153 ;;//
4089 027100 077001 000005 077001 .WORD   SOB+1, RESET, SOB+1 ;; LOCAL-RESET.
4090 027106 052777 002000 152200 BIS      #2000,@#CSR  ; BUT ENABLE TRAP ON Q-BUS RESET.      ;FX8JC
4091 027114 042737 000100 177540 QIEND: BIC      #BIT06,@#K2CSRJ ;
4092 027122 012777 002000 152164 MOV      #2000,@#CSR  ;...AND CSR'S.      ;FX8JC
4093 027130 000530              BR      TST13        ;;AND THAT'S ALL THERE IS TO IT.
4094 027132      102      125      123 QI1: .ASCIZ  /BUS TIME-OUT AT QIR /
4095 027157      121      055      102 QI2: .ASCIZ  /Q-BUS REQUEST DIDN'T SET/
4096 027210      042      102      111 QI3: .ASCIZ  /"BIACK" NOT RECEIVED (Q-REQ STILL SET)/
4097 027257      111      116      124 QI4: .ASCIZ  /INTERRUPT (ON "BIACK") NOT RECEIVED/
4098 027323      124      122      101 QI5: .ASCIZ  /TRAP THRU 24 NOT RECEIVED ON "BRESET"/
4099 027371      121      111      122 DPR11: .ASCIZ  /QIR NOT WRITTEN/
4100                               .EVEN

```

E9

4102

```

027412 000004
027414 032737 000340 177524
027422 001417
027424 012706 001100
027430 012737 177777 002426
027437 005037 044742
027442 032777 000400 151470
027450 001006
027452 005777 151642
027456 000240
027460 102002
027462 000137 032124
027466 104415 027474
027472 000413
027474 040 040 124

```

```

*****
;*TEST 13 TWO PORT REGISTERS
*****
TST13: SCOPE
BIT #340,@#K2CSRC ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
BEQ 30056$ ;; SKIP THIS TEST IF IT IS
MOV #STACK,SP ;; SET A CLEAN STACK.
MOV #-1,TRAP4X ;; DISMISS BUS-ERRORS.
CLR EMPRE ;; CLEAR ERROR PREFIX.
BIT #BIT8,@SWR ;; FORCED ENTRY SELECTED ??
BNE 30057$ ;; BR IF SO.
TST @DPR ;; CHECK FOR VALID $DPR.
NOP
BVC 30057$ ;; SKIP NEXT IF OK.
30056$: JMP TST14 ;; BYPASS.
30057$: T$NAME, +4 ;; PRINT TEST NUMBER AND NAME...
BR 30058$ ;; ...AND SKIP OVER THE ASCII.
.ASCIZ \ TWO PORT REGISTERS\
.EVEN
30058$:

```

4103

027522

```

*****
;*TEST 13.1 READ/WRITE ACCESS
*****
T13.1::

```

4104

4105

4106

4107

4108

027522

```

027522 012737 027530 001110
027530 012737 177777 001124
027536 012777 002000 151550
027544 013700 001320
027550 012701 000017
027554 120027 000002
027560 001001
027562 005720
027564 010037 001120
027570 013710 001124
027574 102421
027576 012037 001126
027602 023737 001124 001126
027610 001407
027612 012737 046533 044746
027620 012737 045210 044752
027626 104154
027630 077127
027632 005037 002426
027636 000411
027640
027640 012737 046500 044746
027646 012737 045230 044752
027654 104155
4128 027656 000137 032122

```

```

; FIRST, DISABLE THE DPR AND VERIFY WRITE/READ
; FROM THE LOCAL (II) SIDE.
TDP1:
MOV #.+6,$LPERR ;; LOOP HERE ON ERROR.
MOV #-1,$GDDAT ;; SET EXPECTED DATA.
MOV #2000,@$CSR ;; EVERYTHING OFF BUT TRAP ON Q-BUS ;FX8JC
;; RESET. ;FX8JC
MOV $DPR,R0 ;; DPR BASE ADDRESS (LOCAL).
MOV #15.,R1
1$: CMPB R0,#2 ;; QDT'S STATUS REG ??
BNE 2$ ;; SKIP IF NOT.
TST (R0)+ ;; YES, DON'T WRITE IT.
2$: MOV R0,$GDADR ;; SET ADDRESS.
MOV $GDDAT,(R0) ;; WRITE ALL BITS...
BVS 4$ ;; ...AND READ IT BACK.
MOV (R0)+,$BDDAT
CMP $GDDAT,$BDDAT
BEQ 3$
MOV #DPR2,EMADR ;;\
MOV #EF2,ESADR ;; >> ERROR 154, LOCAL DPR WRITE-READ ERROR
ERROR+154 ;;//
3$: SOB R1,1$
CLR TRAP4X ;; RESET TRAPPER...
BR TDP2 ;; ...AND PROCEED.
4$: MOV #DPR1,EMADR ;;\
MOV #LA16,ESADR ;; >> ERROR 155, BUS-ERROR ON DPR ADDRESS
ERROR+155 ;;//
JMP DP2END ;; ABORT TEST.

```

F9

```

4130
4131 ; NOW THE DPR IS FULL OF 1'S (EXCEPT FOR WORD 1).
4132 ; READ ALL WORDS FROM THE Q BUS SIDE USING THE DMA ENGINE.
4133 ; EXPECT ALL ADDRESSES TO RESPOND (NO EOP), BUT DATA = 0'S.
4134
4135 LOQB= 160000 ; 162000 ; LOW RANGE Q-BUS BASE ADDRESS (ID 0-7).
4136 HIQB= 175400 ; 177400 ; HIGH DITTO (ID 8 F).
4137 LO2HI= HIQB-LOQB ; OFFSET TO CHANGE FROM LO TO HI RANGE.
4138
4139 027662 017700 151452 ; DPR2: MOV @%TCID,R0 ; GET CPU ID NUM (CSRC<7:4>).
4140
4141 ;TEST THAT THE IOP ID DETERMINED BY THE ARBITER MATCHES THE ACTUAL
4142 ;SETTING OF THE ID SWITCH REPRESENTED IN K2CSRC <7-4>
4143 027666 010001 MOV R0,R1 ;
4144 027670 072127 177774 ASH #-4,R1 ;
4145 027674 020137 050742 CMP R1,IOP.ID ; IOP ID NUMBER SHOULD = SWITCH SETTING
4146 027700 001407 BEQ 10$ ;
4147 027702 012737 031156 044746 MOV #IDERR,EMADR ;\
TI0N 027710 012737 045206 044752 MOV #NOSIG,ESADR ; >> ERROR 156, IOP ID NUMBER DOES NOT MATCH ARBITER CALCULA
027716 104156 ERROR+156 ;\
4148 027720 042700 177617 10$: BIC #%C160,R0 ; STRIPPED TO <6:4> = NUM*16...
4149 027724 006300 ASL R0 ; ...TIMES 2 = NUM*32...
4150 027726 063700 001322 ADD $DPRQ,R0 ; ...PLUS BASE = Q-BUS DPR ADDRESS.
4151 027732 032777 000200 151400 BIT #BIT7,@%TCID ; ID >= 8. ??
4152 027740 001402 BEQ 1$ ; NO.
4153 027742 062700 015400 ADD #L02HI,R0 ; YES, ADJUST TO HIGH RANGE.
4154 027746 010027 1$: MOV RO,(PC)+ ; SAVE QDPR ADDRESS...
4155 027750 000000 QDPR: 0 ; ...HERE.
4156 027752 032737 000010 177522 BIT #BIT03,@%K2CSRB ; IS THE HI RANGE JUMPER INSTALLED?
4157 027760 001404 BEQ 5$
4158 027762 052737 002000 027750 BIS #2000,QDPR ; HI RANGE JUMPER INSTALLED
4159 027770 000403 BR 4$
4160 027772 042737 002000 027750 5$: BIC #2000,QDPR ; HI RANGE JUMPER NOT INSTALLED
4161
4162 030000 012701 061022 4$: MOV #BUFR1,R1 ; SET LOCAL ADDRESS...
4163 030004 012702 000020 MOV #16.,R2 ; ...AND WORD COUNT.
4164 030010 004737 031722 CALL RQIO ; READ FROM Q-BUS SIDE VIA DMA.
4165 030014 102012 BVC 1$ ; BR IF NO TIME-OUT "EOP".
4166 030016 010037 001120 MOV RO,$GDADR ; BAD Q ADDRESS (+2).
4167 030022 012737 046566 044746 MOV #DPR3,EMADR ;\
030030 012737 045222 044752 MOV #LA16M2,ESADR ; >> ERROR 157, BUS ERROR ON Q-BUS DPR READ
030036 104157 ERROR+157 ;\
4168 030040 000431 BR TIPV1
4169 030042 012700 061022 1$: MOV #BUFR1,R0
4170 030046 012701 000020 MOV #16.,R1
4171 030052 013737 027750 001120 MOV QDPR,$GDADR ; SRC ADDRESS(ES)...
4172 030060 005037 001124 CLR $GDDAT ; ...SHOULD HAVE RETURNED ZERO.
4173 030064 010037 001122 2$: MOV RO,$BDADR ; DST ADDRESS(ES)...
4174 030070 012037 001126 MOV (R0)+,$BDDAT ; ...AND WHAT WE REALLY GOT.
4175 030074 001407 BEQ 3$ ; BR IF ZERO.
4176 030076 012737 046633 044746 MOV #DPR4,EMADR ;\
030104 012737 045360 044752 MOV #EF6,ESADR ; >> ERROR 160, NON-ZERO DATA ON Q-BUS READ
030112 104160 ERROR+160 ;\
4177 030114 062737 000002 001120 3$: ADD #2,$GDADR
4178 030122 077120 SOB R1,2$
4179
;*****
;*TEST 13.2 Q-BUS WRITE

```

```

030124
4180
4181
4182
4183
4184
4185 030124
030124 012737 030132 001110
4186 030132 012737 031126 000120
4187 030140 012737 031134 000124
4188 030146 012737 031142 000134
4189 030154 106427 000340
4190 030160 004737 001462
4191 030164 052737 140000 177776
4192 030172 012706 140000
4193 030176 013746 000404
4194 030202 013746 000406
4195 030206 012737 000137 000404
4196 030214 012737 002544 000406
4197 030222 012737 177654 172356
4198
4199
4200
4201 030230 052737 000001 177572
4202 030236 013700 027750
4203 030242 012701 061022
4204 030246 012702 000001
4205 030252 012711 000040
4206 030256 010037 001120
4207 030262 052737 000100 177540
4208 030270 005037 001346
4209 030274 004737 031732
4210 030300 102007
4211 030302 012737 046774 044746
030310 012737 045230 044752
030316 104161
4212 030320 005737 001346
4213 030324 001007
4214 030326
030326 012737 046700 044746
030334 012737 045206 044752
030342 104162
4215 030344 012737 177600 172356
4216 030352 042737 000001 177572
4217 030360 012726 000406
4218 030364 012726 000404
4219 030370 012737 000340 177776
4220 030376 106427 000000
4221 030402 005004
4222 030404 012703 000017
4223 030410 012702 000001
4224 030414 010037 001120
4225 030420 004737 031732
4226 030424 102407
4227 030426 012737 046771 044746
030434 012737 045230 044752

```

```

*****
T13.2::
;
; SO FAR, SO GOOD -- NOW WRITE ALL FROM THE Q-BUS SIDE.
; 1ST WRITE SHOULD FORCE A NON MASKABLE RESTART TRAP (COMMAND INT).
; ALL OTHERS SHOULD APPEAR NON-EXISTANT (EOP).
;
TIPV1:
MOV    #.+6,$LPERR    ;; LOOP HERE ON ERROR.
MOV    #IR4,DPRV4    ; LEVEL 5 INTS SHOULD...
MOV    #IR8,DPRV8    ; ...NOT HAPPEN...
MOV    #IR12,DPRV12  ; .....(YET)...
MTPS   #PR7          ; ... "COMMAND" INTERRUPT SHOULD.
JSR    PC,MMU        ; INIT MMU REGISTERS
BIS    #140000,@#PSW ; SET USER MODE
MOV    #140000,R6    ; SET UP USER STACK
MOV    @#404,-(SP)
MOV    @#406,-(SP)
MOV    #137,@#404
MOV    #NMITRP,@#406
MOV    #177654,@#KIPAR7 ; WILL CAUSE NON MASKABLE INTERRUPT TO POINT
; ..TO PHYSICAL ADDRESS 404 RATHER THAN 1773004
; ...WHICH IS THE FIRMWARE RESTART ADDRESS. GETS
; ... AROUND THE NEED TO HAVE FIRMWARE TO DO TEST
BIS    #1,@#MMRO    ; TURN ON MMU
MOV    QDPR,R0      ; SET GLOBAL DPR ADDRESS...
MOV    #BUF1,R1     ; ...LOCAL ADDRESS...
MOV    #1,R2        ; ...AND WORD COUNT.
MOV    #40,(R1)     ; SET 1ST WORD = "NOP" COMMAND.
MOV    R0,$GDADR
BIS    #BIT06,@#K2CSRJ ; ARM INTERRUPTS
CLR    @#TPROTR    ; SAY EXPECTING RESET TRAP
CALL   WQIO        ; WRITE DPR WORD 0...
BVC    1$          ; ...SHOULD NOT HAVE ABORTED.
MOV    #DPR7,EMADR ;
MOV    #LA16,ESADR ;
; //
; >> ERROR 161, TIME-OUT ON WRITE TO DPRO
; //
1$:    TST    @#TPROTR ; DID TRAP OCCUR?
BNE    2$
11$:
MOV    #DPR5,EMADR ;
MOV    #NOSIG,ESADR ;
; //
; >> ERROR 162, COMMAND INTERRUPT DOESN'T WORK
; //
2$:    MOV    #177600,@#KIPAR7 ;
BIC    #1,@#MMRO    ;
MOV    #406,(SP)+
MOV    #404,(SP)+
MOV    #340,@#PSW   ; SET KERNEL MODE
MTPS   #PRO        ; LOWER CPU.
CLR    R4          ; CLEAR INT RECEIVED FLAG.
MOV    #15.,R3     ; SET LOOP FOR 15 MORE WRITES.
4$:    MOV    #1,R2   ; ONE AT A TIME.
MOV    R0,$GDADR
CALL   WQIO        ; WRITE NEXT FROM Q-BUS SIDE...
BVS    5$          ; ... SHOULD RETURN "EOP".
MOV    #DPR6,EMADR ;
MOV    #LA16,ESADR ;
; >> ERROR 163, Q-BUS DPR WRITE DIDN'T TIME-OUT

```

```

030442 104163          ERROR+163          ://
4228 030444 077317      5$: SOB      R3,4$          ://
4229 030446 005704      TST      R4          ; NONE SHOULD HAVE INTERRUPTED.
4230 030450 001407      BEQ          TIPV2
4231 030452 012737 047042 044746      MOV      #DPR8,EMADR      ://
030460 012737 045206 044752      MOV      #NOSIG,ESADR    ; >> ERROR 164, INT HAPPENED WITH DPR DISABLED
030466 104164          ERROR+164          ://

;
; NOW, ENABLE THE DPR, AND WRITE WORDS 1-15 AGAIN.
; EXPECT EOP AT WORDS 1, 5, AND 9,
; AND INTERRUPTS AT WORDS 4, 8, AND 12.
;
4232
4233
4234
4235
4236
4237 030470 012777 002170 150616 TIPV2: MOV      #2170,@$CSR      ; SET ALL ENABLES INC. Q BUS RESET TRAP.;FX8JC
4238 030476 013700 027750      MOV      QDPR,R0        ; SET Q ADDRESS...
4239 030502 062700 000002      ADD      #2,R0          ;...FOR DPR1...
4240 030506 012701 061024      MOV      #BUF1+2,R1     ;...AND LOCAL ADDRESS.
4241 030512 012703 000002      MOV      #BIT1,R3       ; BIT = CURRENT WORD UNDER TEST.
4242 030516 010037 001120      1$: MOV      R0,$GDADR    ; SET CURRENT DPR ADDR FOR ERROR.
4243 030522 005004      CLR      R4             ; CLEAR "INT RECVD" FLAG.
4244 030524 012702 000001      MOV      #1,R2          ; SET WC...
4245 030530 004737 031732      CALL     WQIO           ;...AND WRITE A WORD.
4246 030534 102413      BVS     3$             ; BR IF WRITE TIMED OUT (EOP).
4247
4248 030536 030327 001042      2$: BIT      R3,#<BIT1!BIT5!BIT9> ; TIME-OUT EXPECTED ??
4249 030542 001422      BEQ     4$             ; NO, PROCEED.
4250 030544 012737 046771 044746      MOV      #DPR6,EMADR    ://
030552 012737 045230 044752      MOV      #LA16,ESADR    ; >> ERROR 165, WRITE TO DPR DIDN'T TIME-OUT (NO EOP)
030560 104165          ERROR+165          ://
4251 030562 000412      BR      4$
4252 030564 030327 001042      3$: BIT      R3,#<BIT1!BIT5!BIT9> ; TIMED-OUT EXPECTED ??
4253 030570 001007      BNE     4$             ; YES, SO FAR SO GOOD.
4254 030572 012737 046774 044746      MOV      #DPR7,EMADR    ://
030600 012737 045230 044752      MOV      #LA16,ESADR    ; >> ERROR 166, WRITE TO DPR TIMED-OUT (EOP)
030606 104166          ERROR+166          ://
4255
4256 030610 030327 000020      4$: BIT      R3,#BIT4      ; INTERRUPT 4 EXPECTED ??
4257 030614 001406      BEQ     5$             ; NO.
4258 030616 012737 032060 047112      MOV      #04,DPR9+3     ; YES...
4259 030624 020437 047112      CMP      R4,DPR9+3      ;...RECEIVED ??
4260 030630 001022      BNE     66$           ; NOPE.
4261 030632 030327 000400      5$: BIT      R3,#BIT8      ; INTERRUPT 8 EXPECTED ??
4262 030636 001406      BEQ     6$             ; NO.
4263 030640 012737 034060 047112      MOV      #08,DPR9+3     ; YES...
4264 030646 020437 047112      CMP      R4,DPR9+3      ;...RECEIVED ??
4265 030652 001011      BNE     66$           ; NOPE.
4266 030654 030327 010000      6$: BIT      R3,#BIT12     ; INTERRUPT 12 EXPECTED ??
4267 030660 001415      BEQ     7$             ; NO.
4268 030662 012737 031061 047112      MOV      #12,DPR9+3     ; YES...
4269 030670 020437 047112      CMP      R4,DPR9+3      ;...RECEIVED ??
4270 030674 001407      BEQ     7$             ; YUP.
4271 030676
030676 012737 047107 044746      66$: MOV      #DPR9,EMADR    ://
030704 012737 045206 044752      MOV      #NOSIG,ESADR   ; >> ERROR 167, DPRXX INTERRUPT NOT RECEIVED
030712 104167          ERROR+167          ://
4272
4273 030714 006303      7$: ASL      R3          ; SHIFT UP TO NEXT WORD (BIT)...
4274 030716 001277      BNE     1$            ;...AND LOOP 'TIL ALL WORDS DONE.

```

```

4275
4276
4277
4278
4279 030720 013700 001320
4280 030724 016001 000036
4281 030730 042777 010000 150356
4282 030736 012777 000154 150352
4283 030744 032777 040000 150342
4284 030752 000240
4285 030754 110
4286 030756 110 027371 044746
      030764 012737 045206 044752
      030772 104170
4287 030774 000443
4288 030776 052777 010000 150310 11$:
4289
4290 031004 105777 150134 1$:
4291 031010 100375
4292 031012 017703 150130
4293 031016 120327 000131
4294 031022 001370
4295
4296 031024 026001 000036
4297 031030 001012
4298 031032 042777 010000 150254
4299 031040 012737 047144 044746
      031046 012737 045206 044752
      031054 104171
4300
4301 031056 012777 002000 150230 2$:
4302
4303 031064 013700 001320
4304 031070 012701 000016
4305 031074 005020
4306 031076 005720
4307 031100 005020 3$:
4308 031102 077102
4309
4310 031104 012700 002542 10$:
4311 031110 010037 000120
4312 031114 010037 000124
4313 031120 010037 000134
4314 031124 000423
4315
4316 031126 012704 032060 IR4:
4317 031132 000402
4318 031134 012704 034060 IR8:
4319 031140 000402
4320 031142 012704 031061 IR12:
4321 031146 042777 000007 150140
4322 031154 000002
4323 031156 111 117 120 IDERR:
4324 .EVEN

```

```

; FINALLY, USE THE QIR FACILITY AND VERIFY THAT THE
; ARBITER PROCESSOR CAN INDEED WRITE INTO THE DPR.
;
;IPV3: MOV    $DPR,R0
      MOV    36(R0),R1      ; SAVE DPR<15>.
      BIC    #BIT12,@$CSR   ; DISABLE Q BUS INTERRUPT
      MOV    #QIRV2,@$QIR   ; SET Q-REQUEST...
      BIT    #BIT14,@$CSR   ; WAS QIR WRITTEN?
      NOP                    ; DELAY A SKOSH
      BNE    11$
      MOV    #DPR11,EMADR   ;
      MOV    #NOSIG,ESADR   ; \
      ERROR+170             ; >> ERROR 170, QIR NOT WRITTEN
      BR     10$           ; //
      BIS    #BIT12,@$CSR   ; ENABLE Q BUS INTERRUPT
; WAIT FOR ARBITER TO SEND A 'Y' SIGNIFYING IT HAS RESPONDED.      ;FX8JC
      TSTB   @$TKS         ; WAIT FOR CHAR FROM ARBITER.      ;FX8JC
      BPL    1$           ; KEEP LOOKING.      ;FX8JC
      MOV    @$TKB,R3      ; GET THE CHARACTER.      ;FX9JC
      CMPB   R3,#'Y       ; IS IT A YES?      ;FX8JC
      BNE    1$           ; IF NOT, KEEP LOOKING.      ;FX8JC
      CMP    36(R0),R1     ; ...ARBITER SHOULD COMPLIMENT DPR<15>. ;FX8JC
      BNE    2$           ; BR IF IT DID.
      BIC    #RQEN,@$CSR   ; DISABLE REQUEST.      ;FX8JC
      MOV    #DPR10,EMADR ;
      MOV    #NOSIG,ESADR ;
      ERROR+171           ; >> ERROR 171, ARBITER WRITE TO DPR<15> FAILS
      BR     10$           ; //
      MOV    #2000,@$CSR   ; ALL DONE, CLEAR STATUS BUT ALLOW      ;FX8JC
; TRAP ON Q BUS RESET.      ;FX8JC
      MOV    $DPR,R0
      MOV    #14.,R1
      CLR    (R0)+         ; CLEAR DPR0...
      TST    (R0)+         ; ...SKIP DPR1...
      CLR    (R0)+         ; ...AND CLEAR THE REST.
      SOB   R1,3$
      MOV    #DISMISS,R0
      MOV    R0,DPRV4      ; ...RESTORE VECTORS...
      MOV    R0,DPRV8
      MOV    R0,DPRV12
      BR     ENDIRQ
      IR4:  MOV    #"04,R4  ; DPRV4 INTERRUPT RECEIVED.
      IR8:  MOV    #"08,R4  ; DITTO DPRV8.
      IR12: MOV    #"12,R4  ; DITTO DPRV12.
      BIC    #7,@$CSR      ; *** TEMP, ENSURE REQ'S ARE CLEAR.
      RTI
      IDERR: .ASCIIZ /IOP ID ERROR/
      .EVEN

```


4326 031174
 4327

031174

4328
 4329
 4330
 4331

4332 031174 012737 177777 175000
 4333 031202 013746 000004
 4334 031206 013746 000006
 4335 031212 012737 031450 000004
 4336 031220 012737 000340 000006
 4337 031226 012737 000006 175012
 4338 031234 004737 036330
 4339
 4340 031240 012701 000400
 4341 031244 077101
 4342 031246 005037 177766
 4343
 4344

4345

4346 031252 012703 125252
 4347 031256 012702 052525
 4348 031262 012704 175000
 4349 031266 010324
 4350 031270 010224
 4351 031272 022704 175040
 4352 031276 001373
 4353 031300 012704 175000
 4354 031304 020324
 4355 031306 001042
 4356 031310 020224
 4357 031312 001035
 4358 031314 022704 175040
 4359 031320 001371
 4360 031322 022702 125252
 4361 031326 001407
 4362 031330 012704 175000
 4363 031334 012702 125252
 4364 031340 012703 052525
 4365 031344 000750
 4366 031346 012704 175040
 4367 031352 012703 000020
 4368 031356 005044
 4369 031360 000240
 4370 031362 000240
 4371 031364 005714
 4372 031366 001002
 4373 031370 077306
 4374 031372 000474
 4375 031374 005037 001124
 4376 031400 011437 001126
 4377 031404 000407
 4378 031406 010237 001124
 4379 031412 000402

ENDIRQ:

 ;*TEST 13.3 ACCESS TPRS FROM QBUS AND LOCAL SIDES SIMULTANEOUSLY

 T13.3.:

;
 ;NOW TEST THAT TPRS CAN BE ACCESSED FROM THE LOCAL SIDE WHEN THERE
 ;IS HEAVY Q-BUS ACTIVITY TO THEM.
 ;

```

;      MOV      # 1,@#TPR00      ;INIT TPRO WITH -1
;      MOV      @#4,-(SP)        ;SAVE TRAP VECTOR
;      MOV      @#6,-(SP)        ;
;      MOV      @#8,@#4          ;LOAD NEW VECTOR
;      MOV      @#340,@#6        ;
;      MOV      @#6,@#TPR05      ;POINT TO ARBITER ROUTINE IN TPRS
;      JSR      PC,ARBDIS        ;GO TO ARBITER DISPATCH ROUTINE
;                                  ;... TPRS WILL BE ENABLED ON RETURN
;      MOV      @#400,R1         ;
;      SOB      R1,              ;GIVE THE ARBITER TIME TO SET UP
;      CLR      @#177766         ;CLEAR THE CPU ERROR REGISTER

```

;
 ;TEST THE TPRS WHILE THE ARBITER IS READING TPRO IN A TIGHT LOOP
 ;

```

;      MOV      #125252,R3       ;INIT R3 W. H DATA
;      MOV      #52525,R2       ;INIT R2 WITH DATA
;      MOV      #TPR00,R4       ;POINT TO FIRST TPR
1$:    MOV      R3,(R4)+         ;WRITE CHECKER BOARD TO TPR ARRAY
;      MOV      R2,(R4)+
;      CMP      #TPR15+2,R4     ;DID WE RUN OUT OF TPRS YET?
;      BNE     1$
;      MOV      #TPR00,R4       ;POINT TO TPRO0
2$:    CMP      R3,(R4)+         ;CHECK DATA
;      BNE     5$               ;BRANCH IF BAD
;      CMP      R2,(R4)+         ;CHECK DATA
;      BNE     3$               ;BRANCH IF BAD
;      CMP      #TPR15+2,R4     ;DID WE CHECK ALL THE TPRS?
;      BNE     2$
;      CMP      #125252,R2     ;DID WE TRY BOTH PATTERNS?
;      BEQ     4$
;      MOV      #TPR00,R4       ;RESET POINTER
;      MOV      #125252,R2     ;CHANGE PATTERN
;      MOV      #52525,R3
;      BR     1$
4$:    MOV      #TPR15+2,R4
;      MOV      #16,R3          ;SET UP LOOP
11$:   CLR      -(R4)           ;CLEAR TPRO0 TO SIGNAL ARBITER THAT
;      NOP
;      NOP
;      ST      (R4)            ;IS IT CLEAR
;      BNE     12$
;      SOB    R3,11$
;      BR     10$
12$:   CLR      @#GDDAT         ;ERROR CALL DATA
;      MOV     (R4),@#BDDAT     ;ACTUAL DATA
;      BR     13$
3$:    MOV      R2,@#GDDAT     ;ERROR CALL DATA
;      BR     6$

```

```

4380 031414 010337 001124      5$:  MOV    R3,@#$GDDAT
4381 031420 014437 001126      6$:  MOV    -(R4),@#$BDDAT      ;GET BAD DATA
4382 031424 010437 001122      13$: MOV    R4,@#$BDADR      ;... AND FAILING TPR ADDRESS
4383 031430 012737 005554 044746      MOV    #TPRDER,EMADR      ;;\
      031436 012737 045210 044752      MOV    #EF2,ESADR      ;; >> ERROR 172, BAD DATA IN TPRS
      031444 104172      ERROR+172      ;://
4384 031446 000446      BR     10$      ;OK!
4385
4386      ;COMES HERE IF WE SHOULD GET A TIME-OUT WHILE TRYING TO ACCESS TPRS
4387
4388 031450 022626      8$:  CMP    (SP)+,(SP)+      ;FIX STACK
4389 031452 013746 000130      MOV    @#130,-(SP)      ;SAVE VECTOR
4390 031456 012737 031524 000130      MOV    #7$,@#130      ;
4391 031464 012702 177530      MOV    #K2CSR0,R2      ;
4392 031470 052712 000100      BIS    #BIT06,(R2)      ;ENABLE TPRS
4393 031474 042712 010000      BIC    #BIT12,(R2)      ;DON'T ALLOW BUS INTERRUPTS
4394 031500 012737 000110 177532      MOV    #110,@#K2QIR      ;TELL ARBITER TO COOL IT.
4395 031506 032712 040000      14$: BIT    #BIT14,(R2)      ;DID THE QIR GET WRITTEN?
4396 031512 001775      BEQ    14$
4397 031514 052712 030000      BIS    #<BIT13!BIT12>,(R2) ;ENABLE Q-BUS INTERRUPT & IACK INTERRUPT
4398 031520 000234      SPL    4      ;LOWER PRIORITY LEVEL
4399 031522 000001      WAIT
4400 031524 022626      7$:  CMP    (SP)+,(SP)+      ;FIX STACK
4401 031526 042712 030070      BIC    #30070,(R2)      ;TURN OFF MASKABLE INTERRUPTS
4402 031532 012637 000130      MOV    (SP)+,@#130      ;RESTORE VECTOR
4403 031536 010437 001122      MOV    R4,$BDADR      ;SAVE ADDRESS OF TPR CAUSING TIME-OUT
4404 031542 012737 031710 044746      MOV    #TPRTO,EMADR      ;;\
      031550 012737 031600 044752      MOV    #TPRNXM,ESADR      ;; >> ERROR 173, TIME-OUT ON SIMULTANEOUS ACCESS TO TPRS
      031556 104173      ERROR+173      ;://
4405 031560 005037 177766      CLR    @#177766      ;CLEAR CPU ERROR REGISTER
4406
4407 031564 012637 000006      10$: MOV    (SP)+,@#6      ;RESTORE STACK
4408 031570 012637 000004      MOV    (SP)+,@#4      ;
4409 031574 000137 032122      JMP    DPREND
4410
4411 031600      TPRNXM: ;ERROR ROUTINE FOR TPR NXM
4412 031600 042737 175000 001122      BIC    #175000,$BDADR      ;STRIP OFF UN-NEEDED BITS
4413 031606 006237 001122      ASR    $BDADR      ;SHIFT IT OVER ONE
4414 031612 013746 001122      MOV    $BDADR,-(SP)      ;
4415 031616 104405      TYPDS
4416 031620 104401 031626      TYPE  ,1$
4417 031624 000207      RTS    PC
4418 031626 040 124 111 1$: .ASCIZ /TIMED-OUT WHILE ARBITER WAS ACCESSING FROM Q-BUS/
4419 031710 114 117 103 TPRTO: .ASCIZ /LOCAL TPR/
4420      .EVEN

```

```

4422 ; SUBROUTINE TO ACCESS THE Q-BUS I/O PAGE USING THE DMA ENGINE.
4423 ; ON CALL, R0 = QIO ADDRESS, R1 = LOCAL ADDRESS, R2 = WORD COUNT.
4424 ; ON RETURN, THEY CONTAIN UPDATED (TERMINAL) VALUES.
4425 ; IF EOP TERMINATION, "V" BIT IS SET.
4426 ;
4427 ;
4428 031722 042737 000020 032116 RQIO: BIC #BIT4,CM+2 ; READ -- CLEAR "FLIP" BIT.
4429 031730 000403 SKP3
4430 031732 052737 000020 032116 WQIO: BIS #BIT4,CM+2 ; WRITE - SET "FLIP" BIT.
4431 031740 010037 032104 MOV R0,QA+2 ; SET Q-BUS I/O ADDRESS.
4432 031744 010137 032110 MOV R1,LA+2 ; SET LOCAL ADDRESS.
4433 031750 010237 032112 MOV R2,WC ; SET WORD COUNT.
4434 ; MOV $DMA,R5
4435 031754 005005 CLR R5
4436 031756 012765 000115 174470 MOV #115,MMR(R5) ; MODE - VI!WAIT!CPINTLV!ENABLE.
4437 031764 012765 000074 174454 MOV #40!34,CMR(R5) ; CLEAR INTERRUPT ENABLES.
4438 031772 005065 174446 CLR CHA1H(R5)
4439 031776 012765 032100 174442 MOV #QIOC,CHAIL(R5) ; SET START CHAIN ADDRESS.
4440 032004 000403 SKP3
4441 032006 012765 032120 174442 CNTNU: MOV #QIOCC,CHAIL(R5) ; SET CONTINUE CHAIN (DUMMY).
4442 032014 012765 000240 174454 MOV #240,CMR(R5) ; CHAIN-LOAD (OR RELOAD 0) REGISTERS.
4443 032022 012765 000102 174454 MOV #102,CMR(R5) ; SET SOFT-REQ (START/CONTINUE DMA)...
4444 032030 000240 240
4445 032032 032765 020000 174456 1$: BIT #IP,STAT1(R5) ;...AND WAIT FOR DONE/ABORT (IP=1).
4446 032040 001774 1$ BEQ 1$
4447 032042 012765 000054 174454 MOV #40!14,CMR(R5) ; OK, CLEAR IUS AND IP.
4448 032050 016500 174412 MOV 174412(R5),R0 ; GET FINAL Q ADDR (ARAL)...
4449 032054 016501 174402 MOV 174402(R5),R1 ;...LOCAL ADDR (ARBL)...
4450 032060 016502 174462 MOV 174462(R5),R2 ;...AND WORD COUNT (OPK).
4451 032064 032765 000002 174456 BIT #EOP,STAT1(R5) ; "EOP" TERMINATION ??
4452 032072 001401 BEQ 2$ ; SKIP IF NOT...
4453 032074 000262 SEV ;...OTHERWISE, SET "V" IN PSW.
4454 032076 000207 2$: RETURN
4455 ;
4456 032100 001602 QIOC: 1602 ; LOAD CARA, CARB, COPK, AND CH MODE.
4457 032102 177400 000000 QA: 177400, 0 ; Q-BUS I/O ADDRESS (CARA)
4458 032106 000000 000000 LA: 0, 0 ; LOCAL ADDRESS (CARB)
4459 032112 000000 WC: 0 ; WORD COUNT (COPK)
4460 032114 000010 00140 CM: 10, 1340 ; CHAN MODE (1360 IF "FLIP" = 1).
4461 032120 000000 QIOCC: 0 ; DUMMY RELOAD TO CLEAR "NAC" AND CONTINUE.
4462 ;
4463 032122 DPREND:
032122 000400 BR TST14 ;:NEXT TEST

```

M9

4465

; *TEST 14 SHARED MEMORY MAPPING

```

TST14: SCOPE
032124 000004
032126 032737 000340 177524 BIT #340,@#K2CSRC ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
^32134 001413 BEQ 30059$ ;; SKIP THIS TEST IF IT IS
032136 005737 050754 TST MAXBLK ;; IS THERE ANY ROOM FOR SHARED MEMORY
032142 001410 BEQ 30059$ ;; SKIP THIS TEST IF NOT
032144 012706 001100 MOV #STACK,SP ;; SET A CLEAN STACK.
032150 012737 177777 002426 MOV #1,TRAP4X ;; DISMISS BUS-ERRORS.
032156 005037 044742 CLR EMPRE ;; CLEAR ERROR PREFIX.
032162 000402 BR 30060$ ;; SKIP NEXT.
032164 000137 033174 30059$: JMP TST15 ;; BYPASS.
032170 104415 032176 30060$: T#NAME, ;PRINT TEST NUMBER AND NAME...
032174 000414 BR 30061$ ;...AND SKIP OVER THE ASCII.
032176 040 040 123 .ASCIZ \ SHARED MEMORY MAPPING\
.EVEN

30061$:
4466 032226 000237 SPL 7 ;NO INTERRUPTS
4467 032230 012737 032574 000134 MOV #INT134,@#134 ;INTERRUPT ROUTINE ADDRESS
4468 032236 012737 000340 000136 MOV #PR7,@#136 ;PRI 7
4473 032244 012737 032772 000120 MOV #INT120,@#120
4474 032252 012737 000340 000122 MOV #PR7,@#122
4475 032260 004737 010450 JSR PC,PROML0 ;FIGURE OUT THE MAXIMUM AMOUNT
4476 032264 013770 172352 MOV @#KIPAR5,R0 ; OF ALLOCATED SHARED MEMORY THAT
4477 032270 062700 000200 ADD #200,R0 ; CAN BE WRITTEN WITHOUT CORRUPTING
4478 032274 072027 177771 ASH #-7,R0 ; THE CODE RUNNING IN THE K2
4479 032300 012701 000100 MOV #64.,R1
4480 032304 160001 SUB R0,R1
4481 032306 010137 032572 MOV R1,MAXWRT ;THIS IS THE MAX
4482 032312 004737 001462 JSR PC,MMU ;GO INITIALIZE THE PDRS AND PARS
4483 032316 012737 000020 172516 MOV #20,@#172516 ;DISABLE D SPACE
4484 032324 010637 177676 MOV SP,@#UDPAR7 ;SAVE THE STACK POINTER
4485 032330 012706 172300 MOV #172300,SP ;SET UP A STACK IN SUPR D PARS
4486 032334 052737 000001 177572 BIS #1,@#PR0 ;TURN ON MMU
4488
4489 ;First we'll establish a checksum for the diagnostic stuff resident in
4490 ;the RAM
4491 ;
4492 032342 005002 CLR R2
4493 032344 012703 051506 MOV #LASTAD,R3
4494 032350 005001 CLR R1
4495 032352 062102 1$: ADD (R1)+,R2 ;R2 WILL HOLD SUM IGNORING CARRIES
4496 032354 020301 CMP R3,R1
4497 032356 002375 BGE 1$
4498 032360 010237 177666 MOV R2,@#177666 ;SAVE CHECK SUM
4499
4500 032364 005037 175010 CLR @#175010 ;CLEAR TPR4; IT WILL INDICATE WHICH
4501 ;ARBITER SIDE ERROR OCCURRED
4502 032370 005037 177766 CLR @#177766 ;CLEAR CPU ERROR REGISTER
4503 032374 052737 000100 177530 BIS #BIT06,@#K2CSR0 ;ENABLE TPRS AND INTERRUPTS FROM 134
4504 032402 013746 000130 MOV @#130,-(SP) ;SAVE VECTOR AT 130
4505 032406 013746 000132 MOV @#132,-(SP)
4506 032412 012737 032502 000130 MOV #3,@#130
4507 032420 012737 000200 000132 MOV #PR4,@#132
4508 032426 000234 SPL 4 ;LOWER PRIORITY LEVEL
4509 032430 013737 032572 175012 MOV MAXWRT,@#175012 ;PUT MAX WRITE VALUE IN TPRS

```

N9

KXJ11 CA FUNCTIONAL TEST
T14 SHARED MEMORY MAPPING

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 116

```

4510 032436 012737 000404 177532      MOV    #404,@#K2QIR      ;LET ARBITER KNOW WERE READY
4511 032444 032737 040000 177530      BIT    #BIT14,@#K2CSR0  ;WAS VECTOR WRITTEN?
4512 032452 001007                BNE                    2$
4513 032454 012737 004700 044746      MOV    #QIRER,EMADR     ;;\
      032462 012737 045206 044752      MOV    #NOSIG,ESADR     ;; >> ERROR 174, QIR ERROR
      032470 104174                ERROR-174              ;;//
4514 032472 052737 030000 177530 2$:    BIS    #<BIT13!BIT12>,@#K2CSR0 ;ENABLE QIR TO INTERRUPT ARBITER
4515                                ;..AND NOTIFY VIA 130
4516 032500 000001                WAIT
4517
4518 032502 022626                3$:    CMP    (SP)+,(SP)+      ;FAKE THE RTI
4519 032504 012637 000132      MOV    (SP)+,@#132      ;
4520 032510 012637 000130      MOV    (SP)+,@#130      ;RESTORE VECTOR
4521 032514 032737 040000 177530      BIT    #BIT14,@#K2CSR0  ;WAS BIT 14 CLEARED BY IAK?
4522 032522 001407                BEQ    4$
4523 032524 012737 004700 044746      MOV    #QIRER,EMADR     ;;\
      032532 012737 045206 044752      MOV    #NOSIG,ESADR     ;; >> ERROR 175, QIR REQUEST WAS NOT CLEARED BY IAK
      032540 104175                ERROR-175              ;;//
4524 032542 042737 020000 177530 4$:    BIC    #BIT13,@#K2CSR0  ;DISABLE IAK INTERRUPTS
4525 032550 052737 000040 177530      BIS    #BIT05,@#K2CSR0  ;ENABLE INTERRUPTS VIA 134
4526 032556 000001                WAIT    ;AND WAIT FOR IT TO INTERRUPT
4527                                ; WITH START UP INFORMATION
4528 032560 052737 000030 177530      BIS    #<BIT04!BIT03>,@#K2CSR0 ;ENABLE INTERRUPT FROM 124 & 120
4529 032566 000001                6$:    WAIT    ;WAIT FOR ARBITER TO SIGNAL IT IS DONE
4530 032570 000776                BR     6$
4531 032572 000000                MAXWRT: .WORD 0

```

B10

```

4533
4534
4535
4536
4537
4538 032574 010046
4539 032576 010146
4540 032600 010246
4541 032602 010346
4542 032604 013700 175026
4543 032610 001434
4544 032612 013701 175030
4545 032616 010137 177534
4546 032622 010103
4547 032624 020027 000100
4548 032630 101024
4549 032632 010002
4550 032634 062701 000200
4551 032640 077003
4552 032642 162701 000200
4553 032646 010137 177536
4554
4555
4556
4557 032652 072327 177771
4558 032656 060203
4559 032660 005403
4560 032662 042703 177700
4561 032666 050337 177536
4562
4563 032672 052737 000004 177540
4564 032700 000407
4565 032702
      032702 012737 032740 044746
      032710 012737 045206 044752
      032716 104176
4566 032720 012603
4567 032722 012602
4568 032724 012601
4569 032726 012600
4570 032730 012737 000374 177532
4571
4572 032736 000002
4573 032740 200 116 125 BLKER:
4574 .EVEN
; INTERRUPT ROUTINE WILL ACCEPT VALUES PASSED IN TPRS 11&12 .THE NUMBER
; OF BLOCKS TO BE ALLOCATED TO SHARED MEMORY AND THE STARTING Q BUS ADDRESS
; OF SHARED MEMORY RESPECTIVELY
;
INT134: MOV R0,-(SP)
        MOV R1,-(SP)
        MOV R2,(SP)
        MOV R3,-(SP)
        ;SAVE REGS
        MOV @#175026,R0
        ;SAVE NUMBER OF BLOCKS TO SHARE
        BEQ 5$
        ;ERROR IF ZERO
        MOV @#175030,R1
        ;AND STARTING Q BUS ADDRESS
        MOV R1,@#K2CSRF
        ;LOAD K2CSRF WITHSTARTING QBUS ADDR
        MOV R1,R3
        ; AND R3 FOR LATER USE
        CMP R0,#64.
        BHI 5$
        MOV R0,R2
        ;SAVE NUMBER OF BLOCKS IN R2
1$: ADD #200,R1
        ;CALCULATE ENDING ADDRESS
        SOB R0,1$
        ;
        SUB #200,R1
        ;BUMP IT DOWN BY ONE BLOCK
        MOV R1,@#K2CSRH
        ;LOAD ENDING Q BUS ADDRESS
;
;CALCULATE NUMBER OF BLOCKS FIELD FOR K2CSRH
;
        ASH #-7,R3
        ;SHIFT STARTING ADDRESS DOWN
        ADD R2,R3
        ;ADD NUMBER OF BLOCKS
        NEG R3
        ;PUT NUMBER OF BLOCKS IN CSRH FORMAT
        BIC #1C77,R3
        ;STRIP OFF UNNEEDED BITS
        BIS R3,@#K2CSRH
        ;LOAD SHARED BLOCKS FIELD
;
        BIS #BIT02,@#K2CSRJ
        ;ENABLE SHARED MEMORY ONTO QBUS
        BR 6$
5$: MOV #BLKER,EMADR ;;\
        MOV #NOSIG,ESADR ;; >> ERROR 176, NUMBER OF BLOCKS FIELD IS INCORRECT
        ERROR+176 ;://
6$: MOV (SP)+,R3
        MOV (SP)+,R2
        MOV (SP)+,R1
        MOV (SP)+,R0
        MOV #374,@#K2QIR
        ;TELL ARBITER THAT SHARED MEMORY IS
        ; ENABLED; GO WRITE IT.
        RTI
        .ASCIZ <CRLF>/NUMBER OF BLKS NOT VALID/
        .EVEN

```

C10

KXJ11-CA FUNCTIONAL TEST
T14 SHARED MEMORY MAPPING

MACRO Y05.02 Thursday 03 Apr-86 14:11 Page 118

```

4656
4657 032772
4658 032772 013706 177676
4659 032776 013701 175010
4660 033002 005701
4661 033004 001424
4662 033006 005301
4663 033010 072127 000004
4664 033014 000161 033020
4665 033020
      033020 012737 033076 044746
      033026 012737 045206 044752
      033034 104177
4666 033036 000407
4667 033040 012737 033144 044746
      033046 012737 045206 044752
      033054 104200
4668 033056 005037 177530
4669 033062 042737 000004 177540
4670 033070 005037 177572
4671 033074
      033074 000437
4672 033076      123      110      101 SHM1:
4673 033144      123      110      101 SHM2:
4674                                     .EVEN

INT120:
MOV @#UDPAR7,SP ;RESTORE OLD STACK
MOV @#175010,R1
TST R1 ;IS ARBITER REPORTING AN ERROR?
BEQ ENDSHM ;NO, MUST BE DONE TLSTING
DEC R1
ASH #4,R1
JMP 1$(R1)

1$:
MOV #SHM1,EMADR ;;\
MOV #NOSIG,ESADR ;; >> ERROR 177, SHARED MEMORY OUT OF ASSIGNED BOUNDARY
ERROR+177 ;;//
BR ENDSHM
MOV #SHM2,EMADR ;;\
MOV #NOSIG,ESADR ;; >> ERROR 200, SHARED MEMORY TIMED OUT
ERROR+200 ;;//
ENDSHM: CLR @#K2CSR0 ;ALL INTERRUPTS OFF
BIC #BIT02, @#K2CSRJ ;TURN OFF SHARED MEMORY
CLR @#MMR0 ;MMU OFF

5$:
BR TST15 ;;
.ASCIZ /SHARED MEMORY OUT OF ASSIGNED BOUNDARY/
.ASCIZ /SHARED MEMORY TIMED-OUT/

```

D10

KXJ11-CA FUNCTIONAL TEST
T15 SHARED MEMORY LMGH

4676

```

033174 000004
033176 032737 000340 177524
033204 001413
033206 005737 050754
033212 001410
033214 012706 001100
033220 012737 177777 002426
033226 005037 044742
033232 000402
033234 000137 037176
033240 104415 033246
033244 000413
033246 040 040 123

```

```

;*****
;*TEST 15 SHARED MEMORY LMGH
;*****
TST15: SCOPE
      BIT #340,@#K2CSR0      ;; IS THE ID SWITCH IN POSITIONS 0 OR 1?
      BEQ 30062$             ;; SKIP THIS TEST IF IT IS
      TST MAXBLK             ;; IS THERE ANY ROOM FOR SHARED MEMORY
      BEQ 30062$             ;; SKIP THIS TEST IF NOT
      MOV #STACK,SP          ;; SET A CLEAN STACK.
      MOV #-1,TRAP4X         ;; DISMISS BUS-ERRORS.
      CLR EMPRE              ;; CLEAR ERROR PREFIX.
      BR 30063$              ;; SKIP NEXT.
30062$: JMP TST16            ;; BYPASS.
30063$: T$NAME, .+4          ;; PRINT TEST NUMBER AND NAME...
      BR 30064$              ;; ...AND SKIP OVER THE ASCII.
      .ASCIZ \ SHARED MEMORY LMGH\
      .EVEN

```

033274

4677

```

033274
4678
4679
4680 033274 042737 000004 177540
4681
4682
4683 033302 013737 050752 177534
4684
4685 033310 013700 050752
4686 033314 013702 050752
4687 033320 062700 000200
4688 033324 013737 050752 177536
4689 033332 163700 050752
4690 033336 072027 177771
4691 033342 072227 177771
4692 033346 060200
4693 033350 005400
4694 033352 042700 177700
4695 033356 050037 177536
4696 033362 052737 000004 177540
4697
4698
4699
4700
4701 033370 004737 001462
4702 033374 012737 000020 172516
4703 033402 012737 017600 172352
4704
4705 033410 052737 000001 177572
4706 033416 005000
4707 033420 012702 120000
4708 033424 012701 010000
4709 033430 010022
4710 033432 062700 000002
4711 033436 077104
4712

```

```

30064$:
;*****
;*TEST 15.1 DTC DMA ACCESSES TO SHARED MEMORY
;*****
T15.1:-
;THE SHARED MEMORY MAPPING TEST MUST PASS PRIOR TO RUNNING THIS TEST
      BIC #BIT02,@#K2CSRJ      ;DISABLE SHARED MEMORY
;ENABLE ONE BLOCK OF SHARED MEMORY
;LOAD LOWEST Q BUS ADDRESS OF SHARED MEMORY
      MOV LOWLIM,@#177534      ;SET LOW LIMIT OF SHARED MEMORY
;NOW FIGURE SHARED MEMORY'S HIGHEST QBUS ADDRESS LIMIT
      MOV LOWLIM,R0
      MOV LOWLIM,R2
      ADD #200,R0              ;MAKE SHARED MEMORY ONE BLOCK LONG
      MOV LOWLIM,@#K2CSRH     ;LOAD HI LIMIT OF SHARED MEM IN CSRH
      SUB LOWLIM,R0           ;GET NUMBER OF BLOCKS
      ASH #-7,R0
      ASH #-7,R2
      ADD R2,R0
      NEG R0
      BIC #1C77,R0           ;CLEAR AWAY UN-NEEDED BITS
      BIS R0,@#K2CSRH        ;LOAD NUMBER OF BLOCKS FIELD
      BIS #BIT02,@#K2CSRJ     ;ENABLE SHARED MEMORY
;INIT MEMORY TO BE SHARED FROM THE K2 (LOCAL) SIDE WITH A PATTERN THAT
;INCREMENTS BY 2 STARTING WITH ZERO AT THE LOWEST SHARED MEMORY LOCATION
      JSR PC,MMU              ;INIT MMU PARS AND PDRS
      MOV #20,@#172516        ;22 BIT ADDRESSING; NO D SPACE
      MOV #17600,@#KIPARS     ;KIPARS WILL POINT TO LAST 8KB BLOCK
      ; OF LOCAL MEMORY
      ;ENABLE MMU
      BIS #1,@#MMR0
      CLR R0
      MOV #120000,R2
      MOV #4096.,R1           ;INIT 4KW OF MEMORY
1$: MOV R0,(R2)+              ;WRITE MEMORY WITH CONTENTS OF R0
      ADD #2,R0               ;INCREMENT R0 BY 2
      SOB R1,1$              ;LOOP CONTROL
;DO A DTC CONTROLLED TRANSFER OF THE DATA IN SHARED MEMORY TO LOCAL MEMORY

```


E10

```

4713 033440 012705 034546      MOV      #100$,R5      ;POINT TO TABLE THAT HOLDS DTC XFER
4714                          ;PARAMETERS
4715 033444 004737 034762      JSR      PC,SHMXFR    ;GO TO SUBROUTINE THAT LOADS PARAMETERS
4716                          ;INTO DTC
4717                          ;RETURN HERE AFTER SUCCESSFUL DTC XFER.
4718 033450 004737 034510      JSR      PC,32$      ;CHECK RESULTS OF TRANSFER FROM SHARED
4719                          ;MEMORY TO LOCAL MEMORY
4720                          ;DO A DTC CONTROLLED TRANSFER OF THE DATA IN SHARED MEMORY TO LOCAL MEMORY
4721                          ;SO THAT THE PHYSICAL ADDRESSES ARE THE SAME.
4722 033454 012705 034564      MOV      #101$,R5      ;POINT TO TABLE THAT HOLDS DTC XFER
4723                          ;PARAMETERS
4724 033460 004737 034762      JSR      PC,SHMXFR    ;GO TO SUBROUTINE THAT LOADS PARAMETERS
4725                          ;INTO DTC
4726 033464 004737 034510      JSR      PC,32$      ;CHECK RESULTS OF TRANSFER FROM SHARED
4727                          ;MEMORY TO LOCAL SHARED MEMORY.
4728                          ;DO A DMA TRANSFER FROM QBUS SHARED MEMORY TO QBUS SHARED MEMORY
4729 033470 012705 034602      MOV      #102$,R5      ;POINT TO TABLE THAT HOLDS DTC XFER
4730                          ;PARAMETERS
4731 033474 004737 034762      JSR      PC,SHMXFR    ;GO TO SUBROUTINE THAT LOADS PARAMETERS
4732                          ;INTO DTC
4733                          ;
4734                          ;CHECK RESULTS OF TRANSFER FROM SHARED MEMORY TO SHARED MEMORY;
4735                          ;CHECK 2048 WORDS OF DATA TRANSFERED.
4736                          ;
4737 033500 012702 120000      50$:    MOV      #120000,R2      ;POINT TO LOWER HALF OF SHARED MEM
4738 033504 012703 130000      MOV      #130000,R3      ;POINT TO UPPER HALF OF SHARED MEM
4739 033510 022223 51$:    CMP      (R2)+,(R3)+      ;DATA SHOULD BE THE SAME IN EACH HALF
4740 033512 001407      BEQ      52$            ;
4741 033514 012737 036544 044746      MOV      #LMGH3,EMADR    ;;\
033522 012737 045206 044752      MOV      #NOSIG,ESADR   ;; >> ERROR 201, DATA NOT TRANSFERED CORRECTLY
033530 104201      ERROR+201              ;;//
4742 033532 077112      52$:    SOB      R1,51$      ;LOOP CONTROL
4743                          ;
4744                          ;NOW DO A TRANSFER FROM SHARED MEMORY TO SHARED MEMORY HAVING THE DTC
4745                          ;TRANSFER DATA FROM THE LAST LOCATION IN MEMORY TO THE FIRST LOCATION ETC..
4746 033534 012705 034620      MOV      #103$,R5      ;POINT TO TABLE THAT HOLDS DTC XFER
4747                          ;PARAMETERS
4748 033540 004737 034762      JSR      PC,SHMXFR    ;
4749                          ;
4750                          ;RETURN HERE TO CHECK TRANSFER FROM SHARED MEM TO SHARED MEM
4751                          ;
4752 033544 012702 120000      55$:    MOV      #120000,R2      ;POINT TO BOTTOM OF SHARED MEMORY
4753 033550 012703 140000      MOV      #140000,R3      ;POINT TO TOP OF SHARED MEMORY
4754 033554 022243      56$:    CMP      (R2)+,-(R3)      ;
4755 033556 001407      BEQ      57$            ;
4756 033560 012737 036544 044746      MOV      #LMGH3,EMADR    ;;\
033566 012737 045206 044752      MOV      #NOSIG,ESADR   ;; >> ERROR 202, DATA NOT TRANSFERED CORRECTLY
033574 104202      ERROR+202              ;;//
4757 033576 077112      57$:    SOB      R1,56$

```

```

4759 033600
4760
4761
4762
4763
4764
4765 033600 032777 010000 145332
4766 033606 001455
4767 033610 013700 050752
4768 033614 004737 036764
4769
4770 033620 052737 100000 037042
4771 033626 005037 174444
4772 033632 012737 033746 174440
4773 033640 012737 000061 174454
4774 033646 012737 000241 174454
4775 033654 032737 014000 174454
4776 033662 001774
4777 033664 032737 000002 174454
4778 033672 001410
4779 033674 012737 020451 044746
      033702 012737 045206 044752
      033710 104203
4780 033712 000413
4781 033714 032737 000001 174454
4782 033722 001031
4783 033724 012737 020513 044746
      033732 012737 045206 044752
      033740 104204
4784 033742 000137 034342
4785
4786 033746
4787
4788 033746 001602
4789 033750 000000 037042
4790 033754 100000 056476
4791 033760 000002
4792 033762 000030 100140
4793
4794 033766 001602
4795 033770 000000 037042
4796 033774 100000 056516
4797 034000 000002
4798 034002 000030 000140
4799
4800 034006 012737 010000 001372
4801 034014 012705 034636
4802
4803 034020 004737 034762
4804
4805 034024 012705 034654
4806 034030 004737 034762
4807
4808
4809 034034
4810 034034 012737 034322 174440
4811 034042 012737 000241 174454

60$: ;ROUTINE TO SET UP FOR TESTS USING THE KNOWN GOOD IOP. FIRST IT MUST BE ENSURED
;THAT THE KNOWN GOOD IOP RESPONDS TO COMMANDS. ONCE THAT HAS BEEN SHOWN IT WILL
;BE COMMANDED TO EXECUTE A DMA TRANSFER OF 4 K WORDS FROM QBUS SHARED MEMORY TO
;THE KNOWN GOOD IOP THEN IT WILL TRANSFER 2K WORDS FROM THE KNOWN GOOD IOP BACK
;TO QBUS SHARED MEMORY.
      BIT    #BIT12,@SWR          ;IS THERE A KNOWN GOOD IOP?
      BEQ    65$
      MOV    LOWLIM,R0
      JSR    PC,PDPDTC          ;PUT STARTING ADDRESS OF SHARED MEM
; INTO DTC FORMAT
;
      BIS    #<QBMEM!UP> ,CARAHI
      CLR    @#CHA2H
      MOV    #66$,@#CHA2L
      MOV    #CLIE2,@#CMDR      ;DISABLE INTERRUPTS
      MOV    #SCCCH2,@#CMDR    ;ISSUE START CHAIN COMMAND
63$:  BIT    #14000,@#STAT2      ;WAIT FOR IT TO BE DONE
      BEQ    63$
      BIT    #EOP,@#STAT2      ;NXM?
      BEQ    62$
      MOV    #DMA3,EMADR        ;;\
      MOV    #NOSIG,ESADR      ;; >> ERROR 203.
      ERROR+203                ;://
      BR     65$
62$:  BIT    #TC,@#STAT2        ;WAS TRANSFER SUCCESSFUL?
      BNE    64$
      MOV    #DMA4,EMADR        ;;\
      MOV    #NOSIG,ESADR      ;; >> ERROR 204.
      ERROR+204                ;://
      JMP    35$
65$:
66$:  ;CHAINS TO LOAD ADDRESSES IN QBUS MEMORY WITH STARTING ADDRESS
;OF SHARED MEMORY
<CARA!CARB!COP!MODE>
<K2MEM!UP> ,CARAHI
<QBMEM!UP> ,KG#T06+2
2
<SWRQ!HM> ,<CTC!TRWW!INTLV>
<CARA!CARB!COP!MODE>
<K2MEM!UP> ,CARAHI
<QBMEM!UP> ,KG#T07+2
2
<SWRQ!HM> ,<TRWW!INTLV>
64$:  MOV    #10000,DCOUNT      ;SET UP A COUNT LOCATION TO BE DECREMENTED
      MOV    #104$,R5         ;POINT TO START OF TABLE WITH DTC
;PARAMETERS
      JSR    PC,SHMXFR        ;GO TO SUBROUTINE THAT LOADS PARAMETERS
;INTO DTC
      MOV    #105$,R5
      JSR    PC,SHMXFR        ;GO TO SUBROUTINE THAT LOADS PARAMETERS
;INTO DTC
70$:  ;COMES HERE AFTER TRANSFERS TO FROM KG IOP
      MOV    #77$,@#CHA2L      ;
      MOV    #SCCCH2,@#CMDR    ;ISSUE START CHAIN COMMAND

```

```

4812 034050 032737 014000 174454 78$: BIT #14000,@#STAT2 ;SPIN HERE
4813 034056 001774 BEQ 78$ ;
4814 034060 005000 CLR R0 ;
4815 034062 077001 SOB R0, ;WAIT FOR LATE EOP
4816 034064 032737 000002 174454 BIT #EOP,@#STAT2 ;EOP SET?
4817 034072 001414 BEQ 79$ ;
4818 034074 032737 000040 177540 BIT #BIT05,@#K2CSRJ ;WAS EOP CAUSED BY SACK TIMING OUT?
4819 034102 001071 BNE 71$ ;
4820 034104 012737 020513 044746 MOV #DMA4,EMADR ;;\
034112 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 205,
034120 104205 ERROR+205 ;;//
4821 034122 000507 BR 80$ ;RETURN TO NEXT SERIES OF TESTS
4822 034124 032737 000001 174454 79$: BIT #TC,@#STAT2 ;WAS TRANSFER COMPLETE?
4823 034132 001007 BNE 73$ ;
4824 034134 012737 020451 044746 MOV #DMA3,EMADR ;;\
034142 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 206,
034150 104206 ERROR+206 ;;//
4825 034152 005737 001412 73$: TST TSTLOC ;IS TPRO OF KG IOP ZERO?
4826 034156 001043 BNE 71$ ;
4827 034160 032737 140000 001414 BIT #140000,TSTLOC+2 ;ANY ERRORS IN KGIOP?
4828 034166 001407 BEQ 72$ ;
4829 034170 012737 022016 044746 MOV #KG.ER,EMADR ;;\
034176 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 207,
034204 104207 ERROR+207 ;;//
4830 034206 042737 177770 001414 72$: BIC #+C7,TSTLOC+2 ;CLEAR OFF UN-NEEDED BITS
4831 034214 022737 000004 001414 CMP #4,TSTLOC+2 ;IN WAITING FOR COMMAND STATE?
4832 034222 001021 BNE 71$ ;
4833 ;TRANSFER WAS SUCCESSFUL. GO CHECK RESULTS
4834 034224 012702 120000 MOV #120000,R2 ;
4835 034230 012703 130000 MOV #130000,R3 ;
4836 034234 012701 004000 MOV #2048.,R1 ;INIT LOOP COUNTER
4837 034240 022223 75$: CMP (R2)+,(R3)+ ;TOP 2 KWORDS OF SHARED MEM SHOULD =
4838 ; LOWER 2 KWORDS OF SHARED MEM.
4839 034242 001407 BEQ 76$ ;
4840 034244 012737 036544 044746 MOV #LMGH3,EMADR ;;\
034252 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 210, DATA NOT TRANSFERED CORRECTLY TO/FROM KG IOP
034260 104210 ERROR+210 ;;//
4841 034262 077112 76$: SOB R1,75$ ;LOOP CONTROL
4842 034264 000426 BR 80$ ;
4843 034266 042737 000040 177540 71$: BIC #BIT05,@#K2CSRJ ;CLEAR OUT SACK TIME OUT BIT
4844 034274 005337 001372 DEC DCOUNT ;GIVE IT A FEW CHANCES
4845 034300 001255 BNE 70$ ;WHEN DCOUNT GETS TO ZERO ITS HAD
4846 ;ENOUGH TIME. FLAG AN ERROR
4847 034302 74$: MOV #KGNR,EMADR ;;\
034302 012737 021756 044746 MOV #NOSIG,ESADR ;; >> ERROR 211, KNOWN GOOD IOP WENT TO LUNCH
034310 012737 045206 044752 ERROR+211 ;;//
034316 104211 BR 80$ ;
4848 034320 000410 77$: <CARA!CARB!COP!MODE>
4849 034322 001602 <QBUSIO!UP>,175740
4850 034324 140000 175740 <K2MEM!UP>,TSTLOC
4851 034330 000000 001412 2
4852 034334 000002 <SWRQ!HM>,<TRWM!INTLV>
4853 034336 000030 000140 80$:
4854 034342
4855
4856 ;CHECK TO SEE IF A Q-BUS EXERCISER IS PRESENT. CHECK FOR AN 18 BIT QBE AT
4857 ;ADDRESS 770000 FIRST. IF NONE IS FOUND LOOK FOR A 22 BIT QBE AT ADDRESS

```

H10

```

4858 ;17770020. IF BOTH ARE RESIDENT ONLY THE 18 BIT QBE WILL BE USED.
4859 ;
4860 034342 022737 000001 050760 35$: CMP #1,@QBE18 ;IS THERE AN 18 BIT Q-BUS EXERCISER
4861 034350 001011 BNE 36$
4862 034352 023727 050752 007600 CMP LOWLIM,#7600 ;IF LOWLIM IS OUT OF RANGE OF 18 BITS
4863 034360 103005 BHIS 36$ ;THEN BAIL OUT
4864 ;GET ADDRESS FORMAT OF LOWLIM FOR QBE18
4865 034362 004737 037052 JSR PC,Q18ADR
4866 034366 012705 034672 MOV #106$,R5
4867 034372 000421 BR 37$
4868 034374 022737 000001 050756 36$: CMP #1,@QBE22 ;IS THERE A 22 BIT QBE?
4869 034402 001007 BNE 39$ ;NO QBES PRESENT
4870 034404 022737 007600 050752 CMP #7600,LOWLIM ;DO WE NEED 22 BITS TO ACCESS SHARED
4871 034412 101005 BHI 38$ ;MEMORY AND IF SO DOES THE SYSTEM
4872 034414 023727 050736 000026 CMP .ABUSW,#22. ; SUPPORT IT?
4873 034422 000137 035226 39$: JMP SHMXIT
4874 ;GET ADDRESS FORMAT OF LOWLIM FOR QBE22
4875 034426 004737 037076 38$: JSR PC,Q22ADR ;
4876 034432 012705 034726 MOV #107$,R5 ;LOAD POINTER WITH TABLE DATA
4877 034436 004737 034762 37$: JSR PC,SHMXFR ;
4878 034442 004737 034762 JSR PC,SHMXFR ;
4879 ;
4880 ;CHECK TRANSFER FROM QBE TO SHARED MEMORY
4881 ;
4882 034446 005002 400$: CLR R2
4883 034450 077201 SOB R2, ;WAIT A BIT
4884 034452 012702 120000 MOV #120000,R2
4885 034456 022227 063636 402$: CMP (R2)+,#63636
4886 034462 001407 BEQ 401$
4887 034464 012737 036544 044746 MOV #LMGH3,EMADR ;;\
034472 012737 045206 044752 MOV #NOSIG,ESADR ;; >> ERROR 212, BAD DATA TRANSFER FROM QBE TO SHARED MEM
034500 104212 ERROR+212 ;://
4888 034502 077113 401$: SOB R1,402$
4889 034504 000137 035226 JMP SHMXIT ;BAIL OUT AFTER ERROR
  
```

I10

```
4891                                   ;SUBROUTINE TO CHECK TRANSFERS FROM SHARED MEM TO LOCAL
4892                                   ;COMPARE THE DATA TO ENSURE THAT THINGS WENT OK
4893                                   ;
4894 034510 012702 120000           32$:   MOV     #120000,R2
4895 034514 012703 061022                MOV     #BUF1,R3
4896 034520 022223           33$:   CMP     (R2),R3
4897 034522 001407                BEQ     34$
4898 034524 012737 036544 044746          MOV     #LMGH3,EMADR     ;;\
     034532 012737 045206 044752          MOV     #NOSIG,ESADR    ;; >> ERROR 213, DATA NOT TRANSFERED CORRECTLY
     034540 104213                ERROR+213     ;;//
4899 034542 077112           34$:   SOB     R1,33$     ;R1 WAS INITIALLY LOADED WITH OP COUNT
4900 034544 000207                RTS     PC
```

```

4902 034546          100$:
4903                ; FIRST TRANSFER
4904                ; --DATA FOR TRANSFER FROM QBUS SHARED MEMORY TO LOCAL MEMORY---
4905 034546 050752 000000 100000          LOWLIM, 0,    <QBMEM!UP>,    LCLMEM, BUFR1, <K2MEM!UP>,    4096.
4906 034564          101$:
4907                ; --SECOND TRANSFER
4908                ; ---DATA FOR TRANSFER FROM LOCAL SHARED MEMORY TO QBUS SHARED MEMORY---
4909                ; -----
4910 034564 172352 000000 000000          <KIPAR5, 0,    <K2MEM!UP>,    LOWLIM, 0,    <QBMEM!UP>,    4096.
4911 J34602
4912                ; ---THIRD TRANSFER
4913                ; -- DATA FOR TRANSFER FROM QBUS SHARED MEM TO QBUS SHARED MEM-
4914 034602 050752 000000 100000          LOWLIM, 0,    <QBMEM!UP>,    LOWLIM, 10000, <QBMEM!UP>,    2048.
4915 034620
4916                ; ---FOURTH TRANSFER
4917                ; ---NOW FOR SOMETHING COMPLETELY DIFFERENT; SOURCE ADDRESS INCREMENTS
4918                ; ---WHILE DESTINATION DECREMENTS
4919 034620 050752 000000 100000          LOWLIM, 0,    <QBMEM!UP>,    KIPAR5, 17776, <K2MEM!DOWN>,    2048.
4920 034636
4921                ; -- TRANSFER PARAMETERS TO TPR2 AND 3 OF GOOD IOP
4922 034636 172340 056466 100000          KIPAR0, KG$T04, <QBMEM!UP>,    KIPAR0, 175744, <QBUSIO!UP>,    2
4923 034654
4924                ; ---TRANSFER COMMAND TO DMA LOAD CHAIN INTO KNOWN GOOD IOP
4925 034654 172340 056472 100000          KIPAR0, KG$T05, <QBMEM!UP>,    KIPAR0, 175740, <QBUSIO!HOLD>,    1
4926 034672
4927                ; ---TRANSFER TO INIT THE 18 BIT QBUS EXERCISER
4928 034672 172340 037142 000000          KIPAR0, QBLOAD, <K2MEM!UP>,    KIPAR7, 10000, <QBUSIO!UP>,    7
4929
4930                ; ---TRANSFER TO SET THE GO BIT IN CSR2 OF THE 18 BIT QBE
4931 034710 172340 050760 000020          KIPAR0, QBE18, <K2MEM!HOLD>,    KIPAR7, 10002, <QBUSIO!UP>,    1
4932 034726
4933                ; ---TRANSFER TO INIT 22 BIT QBUS EXERCISER
4934 034726 172340 037142 000000          KIPAR0, QBLOAD, <K2MEM!UP>,    KIPAR7, 10020, <QBUSIO!UP>,    7
4935
4936                ; ---TRANSFER TO SET THE GO BIT IN CSR2 OF THE 22 BIT QBE
4937 034744 172340 050756 000020          KIPAR0, QBE22, <K2MEM!HOLD>,    KIPAR7, 10022, <QBUSIO!UP>,    1
4938
4939                ;
4940 034762          ; SHM$FR:
4941                ; SUBROUTINE TO INITIATE AN DTC DMA TRANSFER TO/FROM SHARED MEMORY.
4942                ;
4943                ; ON ENTRY:      R5= START OF TABLE TO LOAD XFER PARAMETERS INTO A
4944                ;                          CHAIN FILE THAT WILL BE LOADED INTO DTC CHIP.
4945                ;                          TOP 8KB OF LOCAL MEMORY IS ENABLED AS SHARED MEMORY
4946                ;                          TO QBUS.
4947                ;
4948 034762 005037 174454          CLR      @#CHDR          ;RESET THE DTC
4949 034766 012737 000115 174470          MOV      @115,@#MFR    ;SET UP THE DTC
4950 034774 005037 174446          CLR      @#CHAIN
4951 035000 012737 035202 174442          MOV      @25,@#CHAIN  ;LOAD CHAIN ADDRESS
4952 035006 013500          15$:  MOV      @ (R5)+,R0
4953 035010 004737 036764          JSR      PC,PDPDTC    ;PUT ADDRESS IN DTC FORMAT
4954 035014 013737 037042 035204          MOV      CARAHI,26$
4955 035022 013737 037044 035206          MOV      CARALO,27$
4956 035030 062537 035206          ADD     (R5)+,27$
4957 035034 052537 035204          BIS     (R5)+,26$
4958 035040 013500          MOV     @ (R5)+,R0

```

```

4959 035042 004737 036764 JSR PC,PDPOTC ;PUT ADDRESS IN DTC FORMAT
4960 035046 013737 037042 035210 MOV CARAHI,28$
4961 035054 013737 037044 035212 MOV CARALO,29$
4962 035062 062537 035212 ADD (R5)+,29$
4963 035066 052537 035210 BIS (R5)+,28$
4964 035072 012501 MOV (R5)+,R1 ;LOAD R1 FOR CHECK ROUTINES
4965 035074 010137 035214 MOV R1,24$ ;LOAD OP COUNT
4966 035100 012737 000240 174454 MOV #SCCCH1,@#CMDR ;ISSUE START CHAIN COMMAND CH1
4967 035106 012737 000102 174454 MOV #SSRCH1,@#CMDR ;ISSUE SET SOFTWARE REQUEST COMMAND
4968 035114 032737 014000 174456 14$: BIT #14000,@#STAT1 ;LOOK FOR CHAIN ABORT OR NAC
4969 035122 001774 BEQ 14$ ;WAIT FOR NAC TO SET
4970 035124 032737 000002 174456 BIT #EOP,@#STAT1 ;DID WE GET AN EOP?
4971 035132 001407 BEQ 12$ ;
4972 035134 012737 020513 044746 MOV #DMA4,EMADR ;:;\
035142 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 214,
035150 104214 ERROR+214 ;:;/
4973 035152 032737 000001 174456 12$: BIT #TC,@#STAT1 ;WAS TRANSFER SUCCESSFUL?
4974 035160 001007 BNE 13$
4975 035162 012737 020451 044746 MOV #DMA3,EMADR ;:;\
035170 012737 045206 044752 MOV #NOSIG,ESADR ;: >> ERROR 215,
035176 104215 ERROR+215 ;:;/
4976 035200 000207 13$: RTS PC
4977
4978 ;
4979 ;CHAIN FILE FOR SHARED MEMORY TRANSFERS
4980 ;
4981 035202 001603 25$: .WORD <CARA!CARB!COP!MODE!CHAD>
4982 035204 000000 26$: .WORD 0 ;SOURCE SEGMENT
4983 035206 000000 27$: .WORD 0 ;SOURCE OFFSET
4984 035210 000000 28$: .WORD 0 ;DESTINATION SEGMENT
4985 035212 000000 29$: .WORD 0 ;DESTINATION OFFSET
4986 035214 000000 24$: .WORD 0 ;OP COUNT
4987 035216 000010 000240 .WORD <HM>,<TRWW!BUSHOG!IEOP>
4988 035222 000000 035202 .WORD <K2MEM>,<25$
4989
4990 035226 005037 177572 SHMXIT: CLR @#MMRO ;OFF MMU
4991 035232 042737 000004 177540 BIC #BIT02,@#K2CSRJ ;DISABLE SHARED MEMORY
4992 035240 042737 176000 037142 BIC #176000,QBLOAD ;CLEAR EXTENDED ADDRESS BITS
4993 035246 005037 037146 CLR QBLOAD+4
4994
4995 ;*****
;*TEST 15.2 TEST QHITS
;*****
T15.2::
QHIT: ;TEST THAT DMA STATE MACHINE TERM THAT HANDLES QHITS IS OK
4996 035252 052737 000004 177540 BIS #BIT02,@#K2CSRJ ;ENABLE SHARED MEMORY
4997 035252 012737 076543 001412 MOV #76543,TSTLOC ;INIT TSTLOC WITH KNOWN DATA
4998 035260 052737 174454 CLR @#CMDR ;CLEAR DTC
4999 035266 005037 174454 CLR @#MMR ;LOAD DTC MASTER MODE REG
5000 035272 012737 000135 174470 MOV #135,@#MMR ;LOAD DTC MASTER MODE REG
5001 035300 005037 174444 CLR @#CHA2H ;CHAIN SEG/TAG FIELD
5002 035304 012737 000061 174454 MOV #CLIE2,@#CMDR ;DISABLE INTERRUPTS FROM DTC
5003 035312 012737 035544 174440 MOV #10$,@#CHA2L ;CHAIN OFFSET
5004 035320 012737 000010 175012 MOV #10,@#TPROS ;TELL ARBITER TO BANG ON SHARED MEMORY
5005 035326 004737 036330 JSR PC,ARBDIS
5006
5007 035332 012737 000241 174454 MOV #SCCCH2,@#CMDR ;ISSUE START CHAIN COMMAND
5008 035340 032737 014000 174454 1$: BIT #14000,@#STAT2 ;

```

```

5009 035346 001774      BEQ      1$
5010 035350 013746 000120    MOV      @#120,-(SP)      ;SAVE VECTOR
5011 035354 013746 000130    MOV      @#130,(SP)      ;SAVE VECTOR
5012 035360 013746 000132    MOV      @#132,-(SP)
5013 035364 012737 035450 000120    MOV      #6$,@#120      ;LOAD TRAP HANDLER
5014 035372 012737 035442 000130    MOV      #7$,@#130      ;
5015 035400 012737 000340 000132    MOV      #PR7,@#132      ;
5016 035406 012702 177530    MOV      #K2CSR0,R2      ;
5017 035412 042712 010000    BIC      #BIT12,(R2)      ;DON'T ALLOW BUS INTERRUPTS
5018 035416 012737 000110 177532    MOV      #110,@#K2QIR    ;TELL ARBITER TO STOP ACCESSING SHARED MEMORY
5019 035424 032712 040000 14$:    BIT      #BIT14,(R2)      ;DID THE QIR GET WRITTEN?
5020 035430 001775      BEQ      14$
5021 035432 052712 030110    BIS      #30110,(R2)      ;ENABLE Q BUS INTERRUPT, IACK INTERRUPT, TPR
5022                                ;..ACCESS FROM Q BUS AND TPRO4 INTERRUPT
5023 035436 000234      SPL      4                ;LOWER PRIORITY LEVEL
5024 035440 000001      WAIT
5025 035442 022626 7$:    CMP      (SP)+,(SP)+      ;FIX STACK INTERRUPT VIA 130
5026 035444 000234      SPL      4                ;WAIT FOR INTERRUPT CAUSED BY ARBITER WRITING
5027 035446 000001      WAIT                      ;TPRO4 TO TELL K2 THAT IT'S STOPPED WRITING MEM.
5028 035450 022626 6$:    CMP      (SP)+,(SP)+
5029 035452 012637 000132    MOV      (SP)+,@#132      ;RESTORE THE VECTORS
5030 035456 012637 000130    MOV      (SP)+,@#130      ;
5031 035462 012637 000120    MOV      (SP)+,@#120      ;
5032 035466 032737 000002 174454    BIT      #EOP,@#STAT2    ;DID EOP OCCUR?
5033 035474 001407      BEQ      2$
5034 035476 012737 020513 044746    MOV      #DMA4,EMADR      ;;\
035504 012737 045206 044752    MOV      #NOSIG,ESADR     ;; >> ERROR 216,
035512 104216      ERROR+216                ;;//
5035 035514 032737 000001 174454 2$:    BIT      #TC,@#STAT2      ;DID WE GET TC?
5036 035522 001007      BNE      3$
5037 035524 012737 020451 044746    MOV      #DMA3,EMADR      ;;\
035532 012737 045206 044752    MOV      #NOSIG,ESADR     ;; >> ERROR 217, NO TC!EOP
035540 104217      ERROR+217                ;;//
5038 035542 000410 3$:    BR      15$
5039
5040 035544 001602 10$:    <CARA!CARB!COP!MODE>
5041 035546 000020 001412    <K2MEM!HOLD>, TSTLOC
5042 035552 000400 000000    <K2MEM!400!UP>, 0
5043 035556 001000      1000
5044 035560 000030 000140    <SWRQ!HM>,<TRWV!INTLV>
5045 035564
5046                                15$:
;*****
;*TEST 15.3 ACCESS SHARED MEMORY WITH LOCKED INSTRUCTIONS
;*****
T15.3.:
SHLOCK: ;TEST THE SHARED MEMORY USING THE LOCK INSTRUCTIONS
5047 035564                                BIS      #BIT02,@#K2CSRJ    ;ENABLE SHARED MEMORY
5048 035564 052737 000004 177540    MOV      #17600,@#KIPAR5  ;POINT TO SHARED MEMORY
5049 035572 012737 017600 172352    BIS      #1,@#MMIO        ;TURN ON MMU
5050 035600 052737 000001 177572    MOV      #120000,R0
5051 035606 012700 120000      MOV      #120000,R2
5052 035612 012702 120000      MOV      #120000,R2
5053                                ;INIT 4KW BLOCK OF SHARED MEMORY WITH PATTERN 120000 TO 137776
5054 035616 012701 010000      MOV      #4096.,R1
5055 035622 007322 1$:    .WORD   7322              ;***TEST WRTLCK INSTRUCTION
5056 035624 062700 000002      ADD      #2,R0
5057 035630 077104      SOB     R1,1$
5058 035632 012702 120000      MOV      #120000,R2

```



```

5059 035636 012700 120000      MOV      #120000,R0      ;
5060 035642 012701 010000      MOV      #4096.,R1      ;
5061 035646 020220      2$:    CMP      R2,(R0)+      ;IS PATTERN THERE?
5062 035650 001410      BEQ      3$              ;
5063 035652 012737 036571 044746  MOV      #LMGH4,EMADR   ;;\
      035660 012737 045206 044752  MOV      #NSIG,ESADR   ;; >> ERROR 220, PATTERN NOT CORRECT
      035666 104220      ERROR+220           ;;//
5064 035670 000442      BR      7$              ;
5065 035672 062702 000002      3$:    ADD      #2,R2      ;
5066 035676 077115      SOB      R1,2$          ;
5067                                ;NOW TEST THE TSTSET LOCK INSTRUCTION
5068 035700 012701 010000      MOV      #4096.,R1      ;
5069 035704 012702 120000      MOV      #120000,R2     ;
5070
5071                                ;WRITE AN INCREMENTING PATTERN TO SHARED MEMORY STARTING
5072                                ;AT VIRTUAL LOACTION 120000 WHICH IS INITIALIZED WITH THE
5073                                ;VALUE 120000. THE TSTSET INSTRUCTION WILL READ THE VALUE
5074                                ;120000 FROM VA 120000. IT WILL THEN OR THE LSB =1 AND
5075                                ;WRITE IT BACK OUT. THE RESULT WILL BE THAT THE DATA IN
5076                                ;EACH VIRTUAL ADDRESS = ITSELF +1
5077
5078 035710 007222      4$:    .WORD   7222      ;****TEST TSTSET INSTRUCTION
5079 035712 077102      SOB      R1,4$          ;LOOP CONTROL
5080 035714 012702 010000      MOV      #4096.,R2      ;
5081 035720 012701 120000      MOV      #120000,R1     ;
5082 035724 012703 120001      MOV      #120001,R3     ;
5083 035730 020321      5$:    CMP      R3,(R1)+      ;CHECK RESULTS
5084 035732 001416      BEQ      6$              ;
5085 035734 010337 001124      MOV      R3,$GDAT      ;GET EXPECTED DATA
5086 035740 014137 001126      MOV      -(R1),$BDDAT   ;...AND ACTUAL DATA
5087 035744 010137 001120      MOV      R1,$GDADR     ;...AND FAILING ADDRESS
5088 035750 012737 036631 044746  MOV      #LMGH5,EMADR   ;;\
      035756 012737 045244 044752  MOV      #EF3,ESADR   ;; >> ERROR 221, TSTSET INSTRUCTION FAILED
      035764 104221      ERROR+221           ;;//
5089 035766 000403      BR      7$              ;BAIL OUT ON ERROR
5090 035770 062703 000002      6$:    ADD      #2,R3      ;UPDATE DATA
5091 035774 077223      SOB      R2,5$          ;
5092 035776
5093 035776
5094
5095 ARBLMGH:
5096 ;CONDUCT SOME TESTS OF LMGH FROM THE ARBITER SIDE.
5097 ;THE TESTS ARE INITIATED BY CAUSING AN INTERRUPT TO THE ARBITER CPU
5098 ;VIA THE QIR. TPRO5 OF THE INTERRUPTING IOP WILL CONTAIN A VALUE THAT
5099 ;WILL BE USED BY THE INTERRUPT ROUTINE EXECUTED BY THE ARBITER TO INDEX
5100 ;TO A TEST SUB-ROUTINE. TEST COMPLETION OR ERRORS ARE REPORTED VIA TPRO4
5101 ;OF THE IOP UNDER TEST.
5102 ;
5103 ;NOW LET'S GET DOWN TO THE BUSINESS OF TESTING
5104 ;
5105 ;CAUSE ARBITER TO WRITE THE SHARED MEMORY USING THE BUS LOCK INSTRUCTIONS
5106 ;
5105 035776 005737 050762      TST      KDJ$          ;IS THE ARBITER A KDJ11
5106 036002 001002      BNE      1$            ;SKIP THIS TEST IF IT IS NOT.
5107 036004 000137 037160      JMP      ENDLMG        ;
5108 036010 013746 000120      1$:    MOV      @#120,-(SP) ;SAVE VECTOR
5109 036014 012737 017600 172352  MOV      #17600,KIPAR5 ;MAP KIPAR5 TO TOP OF PHYSICAL MEMORY
5110 036022 012701 010000      MOV      #4096.,R1     ;SET UP FOR LOOP
5111 036026 012700 120000      MOV      #120000,R0    ;INIT R0 AS A POINTER

```

```

5112 036032 052737 000001 177572    BIS    #1,@MMRO      ;TURN ON MMU
5113 036040 005020          10$:    CLR    (R0)+        ;CLEAR OUT TOP OF MEMORY
5114 036042 077102          SOB    R1,10$      ;LOOP CONTROL
5115 036044 012737 036074 000120    MOV    #15$,@#120  ;SET UP INTERRUPT HANDLER
5116 036052 005037 175010          CLR    @#TPR04     ;CLEAR TPR4
5117 036056 012737 000002 175012    MOV    #2,@#TPR05  ;INDEX VALUE THAT DISPATCHES ARBITER TO TEST..
5118          JSR    PL ARBDIS ;... TPR READ ACCESSES FROM ARBITER SIDE.
5119 036064 004737 036330          ; ROUTINE TO GENERATE AN INTERRUPT TO THE
5120          ;... ARBITER.
5121 036070 000234          SPL    4           ;LOWER PRIORITY LEVEL
5122 036072 000001          WAIT
5123          ;COMES HERE ON INTERRUPT FROM ARBITER WRITING TPR04 AFTER DOING WRTLCK TEST
5124 036074 042737 000010 177530 15$:    BIC    #BIT03,@#K2CSR ;STOP INTERRUPTS FROM WRITES TO TPR04
5125 036102 012701 010000          MOV    #4096,R1   ;SET UP FOR LOOP
5126 036106 012700 120000          MOV    #120000,R0 ;INIT RO AS A POINTER
5127 036112 005002          CLR    R2         ;
5128 036114 020210          20$:    CMP    R2,(R0)    ;IS DATA CORRECT
5129 036116 001415          BEQ    25$       ;
5130 036120 010237 001124          MOV    R2,$GDDAT  ;GET EXPECTED DATA
5131 036124 011037 001126          MOV    (R0),$BDDAT ;GET ACTUAL DATA
5132 036130 010037 001120          MOV    R0,$GDADR  ;GET FAILING ADDRESS
5133 036134 012737 036667 044746    MOV    #WTLK,EMADR ;:\
      036142 012737 045210 044752    MOV    #EF2,ESADR ;: >> ERROR 222, ERROR WITH WRITE LOCK FROM ARB SIDE
      036150 104222          ;://
5134 036152 062702 000002          25$:    ADD    #2,R2      ;ADD TWO TO DATA
5135 036156 062700 000002          ADD    #2,R0      ;... AND TO ADDRESS
5136 036162 077124          SOB    R1,20$    ;LOOP CONTROL
5137 036164 042737 000001 177572    BIC    #1,@MMRO   ;
5138 036172 022626          CMP    (SP)+,(SP)+ ;FAKE THE RTI
5139          ;NOW SET UP TO CHECK TSTSET INSTRUCTION FROM ARBITER SIDE
5140 036174 012737 036216 000120    MOV    #30$,@#120 ;NEW INTERRUPT HANDLER
5141 036202 012737 000004 175012    MOV    #4,@#TPR05 ;POINTER TO TSTSET TEST ON ARBITER SIDE
5142 036210 004737 036330          JSR    PC,ARBDIS  ;WRITE VECTOR TO ARBITER
5143 036214 000001          WAIT
5144          ;COMES HERE ON INTERRUPT FROM ARBITER WRITING TPR04 AFTER DOING TSTSET TEST
5145 036216 042737 000010 177530 30$:    BIC    #BIT03,@#K2CSR ;STOP INTERRUPTS FROM WRITES TO TPR04
5146 036224 012701 010000          MOV    #4096,R1   ;SET UP FOR LOOP
5147 036230 012700 120000          MOV    #120000,R0 ;INIT RO AS A POINTER
5148 036234 012702 000001          MOV    #1,R2     ;
5149 036240 052737 000001 177572    BIS    #1,@MMRO   ;TURN ON MMU
5150 036246 020210          35$:    CMP    R2,(R0)    ;IS DATA CORRECT
5151 036250 001415          BEQ    40$       ;
5152 036252 010237 001124          MOV    R2,$GDDAT  ;GET EXPECTED DATA
5153 036256 011037 001126          MOV    (R0),$BDDAT ;GET ACTUAL DATA
5154 036262 010037 001120          MOV    R0,$GDADR  ;GET FAILING ADDRESS
5155 036266 012737 036726 044746    MOV    #TSET,EMADR ;:\
      036274 012737 045210 044752    MOV    #EF2,ESA7R ;: >> ERROR 223, ERROR WITH TEST SET FROM ARB SIDE
      036302 104223          ;://
5156 036304 062702 000002          40$:    ADD    #2,R2      ;ADD TWO TO DATA
5157 036310 062700 000002          ADD    #2,R0      ;... AND TO ADDRESS
5158 036314 077124          SOB    R1,35$    ;LOOP CONTROL
5159 036316 022626          CMP    (SP)+,(SP)+ ;FAKE THE RTI
5160 036320 012637 000120          MOV    (SP)+,@#120 ;RESTORE VECTOR
5161 036324 000137 037160          JMP    ENDLMG
5162
5163 036330          ARBDIS:
5164          ;SUB-ROUTINE TO CAUSE AN INTERRUPT TO THE ARBITER BY WRITING THE QIR OF

```

```

5165 ;IOP WITH THE APPROPRIATE VECTOR
5166 ;
5167 ;ON ENTRY: TPRO5 CONTAINS A VALUE THAT THE ARBITER WILL USE TO
5168 ; INDEX TO THE DESIRED TEST.
5169
5170 036330 052737 000100 177530 BIS #BIT06,@#K2CSR0 ;ENABLE TPRS
5171 036336 042737 010000 177530 BIC #BIT12,@#K2CSR0 ;DON'T ALLOW QBUS INTERRUPT
5172 036344 013737 036434 177532 MOV INTVEC,@#K2QIR ;WRITE THE QIR
5173 036352 032737 040000 177530 1$: BIT #BIT14,@#K2CSR0 ;DID IT GET WRITTEN?
5174 036360 001774 BEQ 1$ ;BRANCH 'TIL IT DOES
5175 036362 013746 000130 MOV @#130,-(SP) ;SAVE VECTOR
5176 036366 012737 036406 000130 MOV #2$,@#130 ;TRAP HANDLER
5177 036374 052737 030000 177530 BIS #<BIT12!BIT13>,@#K2CSR0 ;THEN ENABLE Q BUS INTERRUPT AND
5178 ; IACK INTERRUPTS
5179 036402 000234 SPL 4
5180 036404 000001 WAIT
5181 036406 022626 2$: CMP (SP)+,(SP)+ ;FIX STACK
5182 036410 012637 000130 MOV (SP)+,@#130 ;RESTORE VECTOR
5183 036414 042737 030000 177530 BIC #<BIT13!BIT12>,@#K2CSR0 ;DISALLOW QBUS INTERRUPTS AND
5184 ; IACK INTERRUPTS
5185 036422 052737 000010 177530 BIS #BIT03,@#K2CSR0 ;ENABLE INTERRUPTS FROM TPRO4
5186 036430 000234 SPL 4 ;ALLOW INTERRUPT
5187 036432 000207 RTS PC
5188
5189 036434 000000 INTVEC: .WORD 0 ;HOLDS VECTOR TO BE WRITTEN TO QIR
5190 ;
5191 036436 200 124 122 LMGH1: .ASCIZ <CRLF>/TRANSFER TO SHARED MEMORY TIMED-OUT/
5192 036503 200 124 122 LMGH2: .ASCIZ <CRLF>\TRANSFER TO Q-BUS I/O TIMED-OUT\
5193 036544 200 102 101 LMGH3: .ASCIZ <CRLF>/BAD DATA TRANSFERED/
5194 036571 200 127 122 LMGH4: .ASCIZ <CRLF>/WRITE LOCK TO SHARED MEM ERROR/
5195 036631 200 124 105 LMGH5: .ASCIZ <CRLF>/TEST SET TO SHARED MEM ERROR/
5196 036667 200 101 122 WTLK: .ASCIZ <CRLF>/ARBITER SIDE WRITE LOCK ERROR/
5197 036726 200 101 122 TSET: .ASCIZ <CRLF>/ARBITER SIDE TEST SET ERROR/
5198 .EVEN
  
```

C11

5200 036764

5201
5202
5203
5204
5205
5206
5207
5208
5209
5210

PDPDTC:

:
;ROUTINE TO FORMAT A PDP/LSI11 PHYSICAL ADDRESS FOR DTC USE.
;TAKE THE STARTING ADDRESS (GLOBAL) OF THE FIRST 8KB BLOCK OF SHARED MEMORY.
;SHIFT IT TWICE TO THE RIGHT TO LINE IT UP WITH THE DTC FORMAT AND MASK OFF
;THE BITS WHICH REPRESENT ADDRESS BITS <15:13>. THE REMAINING BITS WILL BE
;SET IN THE UPPER CURRENT ADDRESS REGISTER. THEN SHIFT R0, WITH ALL UPPER
;ADDRESS BITS STILL SET, UNTIL THE SIGNIFICANT ADDRESS BITS HAVE BEEN SHIFTED
;INTO R3. THEN R3 IS THE OFFSET ADDRESS OF THE Q BUS (GLOBAL) SHARED MEMORY.
:
:

5211 036764 010146
5212 036766 010246
5213 036770 010346
5214 036772 072027 177776
5215
5216 036776 010001
5217 037000 042701 140377
5218 037004 005003
5219 037006 012702 000010
5220 037012 000241
5221 037014 006200
5222 037016 006003
5223 037020 077203
5224 037022 010137 037042
5225 037026 010337 037044
5226 037032 012603
5227 037034 012602
5228 037036 012601
5229 037040 000207
5230
5231 037042 000000
5232 037044 000000
5233 037046 177600
5234 037050 000000

MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
ASH #2,R0 ; BUMP IT RIGHT A COUPLE FOR DTC
; ALIGNMENT
MOV R0,R1 ;SAVE IT IN R1
BIC #1C37400,R1 ;MASK OUT THE NON SEGMENT BITS
CLR R3 ;INIT R3
MOV #10,R2
CLC
40\$: ASR R0 ;CLEAR THE C BIT
ROR R3 ;SHIFT 8 TIMES
SOB R2,40\$
MOV R1,CARAH1 ;LOAD CURRENT ADDR REG A HIGH WORD
MOV R3,CARALO ;LOAD CURRENT ADDR REG A LOW WORD
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
RTS PC

CARAH1: .WORD 0
CARALO: .WORD 0
STQBIO: .WORD 177600 ;START OF QBUS IO
LCLMEM: .WORD 0 ;LOCAL MEMORY

D11

```

5236 037052          Q18ADR: ;SUBROUTINE TO CONVERT AN ADDRESS TO A FORMAT USABLE BY THE QBUS
5237                ;EXERCISER
5238
5239 037052 010246      MOV     R2,-(SP)      ;SAVE THE REGISTERS THAT ARE...
5240 037054 010346      MOV     R3,(SP)      ; GOING TO BE USED BY THIS ROUTINE.
5241 037056 013702 050752 MOV     LOWLIM,R2    ;PUT THE FIRST NON-EXISTANT MEMORY ADDRESS IN R2
5242                ;.. WHICH BY THE WAY, IS ALSO THE FIRST SHARED
5243                ;... MEMORY ADDRESS.
5244 037062 010203      MOV     R2,R3        ; R2 AS WELL
5245 037064 042702 171777 BIC     #†C6000,R2   ;EXTRACT WHAT WILL BE ADDRESS BITS 17,16
5246 037070 072227 000004 ASH     #4,R2        ;ALIGN THEM
5247 037074 000407      BR      QADR
5248
5249 037076          Q22ADR: ;SUBROUTINE TO CONVERT THE FIRST SHARED MEMORY ADDRESS TO A FORMAT
5250                ;USABLE BY THE 22 BIT QBUS EXERCISER
5251
5252 037076 010246      MOV     R2,-(SP)      ;SAVE THE REGISTERS THAT ARE...
5253 037100 010346      MOV     R3,-(SP)      ; GOING TO BE USED BY THIS ROUTINE.
5254 037102 013702 050752 MOV     LOWLIM,R2    ;PUT THE FIRST NON-EXISTANT MEMORY ADDRESS IN R2
5255                ;.. WHICH, BY THE WAY, IS ALSO THE FIRST SHARED
5256                ;... MEMORY ADDRESS.
5257 037106 010203      MOV     R2,R3        ;.....AND R3
5258 037110 042702 001777 BIC     #†C176000,R2 ;EXTRACT BITS 21-16 OF ADDRESS
5259 037114 042703 176177 QADR:  BIC     #†C1600,R3 ;EXTRACT BITS 15-13 OF ADDRESS
5260 037120 072327 000006 ASH     #6,R3        ;ALLIGN BITS
5261                ;WHEN THE ABOVE HAS COMPLETED R3 WILL CONTAIN
5262                ;.ADDRESS BITS 15-13 OF THE QBE22 DESTINATION
5263                ;..ADDRESS. R2 WILL CONTAIN ADDRESS BITS 21-16.
5264 037124 050337 037146 BIS     R3,QBLOAD+4 ;STUFF BITS 15-12 INTO QBE 22 WORD
5265 037130 050237 037142 BIS     R2,QBLOAD   ;STUFF BITS 21-16 INTO QBE 22 WORD
5266                ; OR BITS 17 16 INTO QBE 18 WORD
5267 037134 012603      MOV     (SP)+,R3    ;RESTORE REGISTERS
5268 037136 012602      MOV     (SP)+,R2
5269 037140 000207      RTS     PC
5270
5271 037142 000641      QBLOAD: .WORD 641      ;DATA OUT/ ONE XFER PER DMG/ WORD
5272 037144 000000      .WORD 0          ;CSR2
5273 037146 000000      .WORD 0          ;ADDRESS BITS 15-0
5274 037150 170000      .WORD -4096     ;WORD COUNT
5275 037152 063636      .WORD 63636     ;DMA DATA
5276 037154 000000      .WORD 0
5277 037156 000000      .WORD 0
5278
5279 037160 042737 010010 177530 ; ENDLMG: BIC #<BIT12!BIT03>, @#K2CSR0 ;DISALLOW INTERRUPTS TO QBUS AND
5280                ;... FROM TPRO4
5281 037166 042737 000004 177540 BIC     #BIT02,@#K2CSRJ ;TURN OFF SHARED MEMORY
5282 037174 000400      BR      TST16     ;;

```

E11

5284

037176	000004		
037200	012706	001100	
037204	012737	177777	002426
037212	005037	044742	
037216	000402		
037220	000137	037632	
037224	104415	037232	
037230	000412		
037232	040	040	106

```

*****
; *TEST 16 FIRMWARE CHECKSUM
*****
TST16: SCOPE
        MOV #STACK,SP ; SET A CLEAN STACK.
        MOV #-1,TRAP4X ; DISMISS BUS-ERRORS.
        CLR EMPRE ; CLEAR ERROR PREFIX.
        BR 30066$ ; SKIP NEXT.
30065$: JMP $EOP ; BYPASS.
30066$: T$NAME, .+4 ; PRINT TEST NUMBER AND NAME...
        BR 30067$ ; ...AND SKIP OVER THE ASCII.
        .ASCIZ \ FIRMWARE CHECKSUM\
        .EVEN

```

037256

5285			
5286			
5287			
5288			
5289			
5290			
5291			
5292			
5293			
5294			
5295			
5296			
5297			
5298			
5299			
5300			
5301	037256	013746	172354
5302	037262	012737	021400 172354
5303			
5304	037270	012702	017777
5305			
5306			
5307			
5308			
5309			
5310			
5311			
5312			
5313			
5314			
5315			
5316	037274	005004	
5317	037276	032702	170000
5318	037302	001002	
5319			
5320	037304	010201	
5321			
5322	037306	000402	
5323	037310	012701	010000
5324	037314	160102	
5325			
5326	037316	012703	140000
5327	037322	062304	

```

30067$:
;CHECK OUT THE NATIVE FIRMWARE
;Module ROMTST
;
;Description: This module calculates the checksum of the native firmware
;              in PROM. It then compares the calculated checksum with the
;              checksum blasted into the last location in the native
;              firmware PROM section. If they are equal the PROM is good.
;
;              The following algorithm is used to calculate the checksum:
;
;              CHECKSUM = 0
;              FOR I = number of PROM addresses to be checksummed DO
;                  CHECKSUM = CHECKSUM + contents of address
;                  CHECKSUM = ROTATE_LEFT_ONE_BIT
;              NEXT I
;
NATIVE: MOV @#KIPAR6,-(SP) ;Save contents of KIPAR6
        MOV #21400,@#KIPAR6 ;Point KIPAR6 to physical addresses starting
;              .. at 2140000.
        MOV #17777,R2 ;Check 8K words of PROM - 1
;
;Routine to calculate native firmware checksum and compare it with the
;checksum value located in the last native firmware PROM word location.
;
;Inputs: R2 = Number of words of PROM to be checksummed minus 1 for the
;          word which contains the PROM checksum value.
;          To checksum 16kB of native firmware PROM, R2 = 17777
;
;          KIPAR6 = 21400. Points to begining of PROM address space.
;
SUMCHK: CLR R4 ;Init checksum location.
10$: BIT #170000,R2 ;Are any of the bits 15 thru 12 = 1?
      BNE 1$ ;Branch if any are set
;          ; Otherwise
;          ; Load loop control with remaining value in R2
;          ; which will be less than 10000
;
      MOV R2,R1
;
2$: BR 2$ ;
1$: MOV #10000,R1 ;Check 4K words of PROM
2$: SUB R1,R2 ;Subtract the number words of PROM being tested
;          ;... from R2
      MOV #140000,R3 ;Point to KIPAR6
3$: ADD (R3)+,R4 ;Add contents of address to checksum

```

```

5328 037324 000241          CLC          ;Clear the carry bit in psw
5329 037326 100001          BPL          11$      ;Branch if plus
5330 037330 000261          SEC          ;..else set carry bit
5331 037332 006104          11$: ROL      R4      ;..end around carry
5332 037334 077106          SOB      R1,3$      ;Loop control
5333 037336 005702          TST      R2        ;If R2 = zero all locations have been checked.
5334 037340 001404          BEQ          ;Go see if checksum is good.
5335 037342 062737 000200 172354 ADD      #200,@#KIPAR6 ;Otherwise prepare to check the next 4k word
5336 037350 000752          BR       10$      ;... section of PROM.
5337 037352 021304          4$: CMP      (R3),R4 ;R3 now points to last user PROM address.
5338                                ;Are checksums equal?
5339 037354 001412          BEQ      5$
5340 037356 012637 172354          MOV      (SP)+,@#KIPAR6 ;Restore KIPAR2
5341 037362 012737 037412 044746          MOV      #PROMCK,EMADR ;:\
      037370 012737 045206 044752          MOV      #NOSIG,ESADR ;: >> ERROR 224, PROM CHECKSUM ERROR
      037376 104224          ERROR+224 ;://
5342 037400 000402          BR       6$
5343 037402 012637 172354          5$: MOV      (SP)+,@#KIPAR6 ;Restore KIPAR6
5344 037406 000137 037632          6$: JMP      $EOP
5345 037412 120 122 117 PROMCK: .ASCIZ /PROM CHECKSUM ERROR/
5346                                .EVEN

```

G11

KXJ11-CA FUNCTIONAL TEST
T16 FIRMWARE CHECKSUM

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 135

```

5348
5349
5350
5351
5352
5353
5354
5355
5356
5357 037436 010046
5358 037440 010146
5359 037442 010100
5360 037444 042700 017777
5361 037450 000241
5362 037452 000005
5364 037452 006100
      037454 006100
      037456 006100
      037460 006100
      037462 006100
5365 037464 016037 172340 037540
5366 037472 005037 037536
5367 037476 012700 000006
5368 037502 006337 037540
5369 037504 006137 037536
5370 037512 077005
5371 037514 042701 160000
5372 037520 060137 037540
5373 037524 005537 037536
5374 037530 012601
5375 037532 012600
5376 037534 000207
5377
5378 037536 000000
5379 037540 000000
5380
5381
5382
5383 037542 113700 001102
5384 037546 042700 177760
5385 037552 110077 141562
5386 037556 000240
5387 037560 000207
5388
5389
5390
5391 037562 004717
5392 037564 004717
5393 037566 012700 000024
5394 037572 077001
5395 037574 000207
;KXJBUT.MAC      16-JAN-86
;
; SUBROUTINE TO CONVERT 16 BIT (PAR FORMAT) VIRTUAL
; ADDRESS TO A 22 BIT GLOBAL (PHYSICAL) ADDRESS.
;
; ON ENTRY, R1 = PAR FORMAT VIRTUAL ADDRESS.
; ON EXIT, GAHI<5:0> AND GALO<15:0> = GLOBAL ADDRESS<21:0>.
;
VA2GA:  MOV     RO,-(SP)
        MOV     R1,-(SP)
        MOV     R1,RO          ; COPY VA.
        BIC     #17777,RO     ; STRIP PAR NUMBER...
        CLC                    ; ...ROTATE IT AROUND TO AN INDEX.
        .REPT   5
        ROL     RO
        ROL     RO
        ROL     RO
        ROL     RO
        ROL     RO
        MOV     KIPARO(RO),GALO ; GET PAR VALUE => GALO.
        CLR     GAHI          ; ZERO => GAHI.
        MOV     #6,RO
1$:     ASL     GALO
        ROL     GAHI          ; ROTATE PAGE TO BIT<21>...
        SOB    RO,1$
        BIC     #17777,R1     ; ...STRIP VA<12:00>...
        ADD     R1,GALO       ; ...AND COMBINE THE PIECES.
        ADC     GAHI
        MOV     (SP)+,R1
        MOV     (SP)+,RO
        RETURN
GAHI:   0                      ; GA<21:16>
GALO:   0                      ; GA<15:00>
;
; SUBROUTINE TO PUT TEST NUMBER IN THE LEDS.
;
TNLEDS: MOVB   #TSTNM,RO     ; TEST NUMBER.
        BIC   #17,RO         ; STRIP THE NUMBER...
        MOVB  RO,@#LEDS     ; ...AND LITE THE LITES.
        NOP
        RETURN
;
; SUBROUTINE(S) TO DELAY IN SOUS INCREMENTS.
;
D200:  CALL   @PC            ; 200 USEC.
D100:  CALL   @PC            ; 100 USEC.
D50:   MOV    #20.,RO       ; 50 USEC.
        SOB  RO,.           ; == 2.6US/COUNT.
        RETURN

```


H11

KXJ11-CA FUNCTIONAL TEST
T16 FIRMWARE CHECKSUM

```

5398
5399
5400
5401
5402 037576 000000
5403 037600 015 012
5404 037602 001 002
5405 037612 376 375
5406 037622 040 055
5407 037625
5408
5450

```

```

; FLOATING 1 AND 0 DATA TABLE.
; USED IN SERIAL LINE (DC319 AND NEC7201) AND PIO TESTS.
;
BYTES: 0 ; BYTE COUNTS -- LOB XMTR, HIB RCVR.
FLT10: .ASCII <CR><LF> ; SYNC PAIR.
FLT1: .BYTE 001, 002, 004, 010, 020, 040, 100, 200 ; FLOATING 1.
FLT0: .BYTE 376, 375, 373, 367, 357, 337, 277, 177 ; FLOATING 0.
; AND END WITH "--X--<DING>".
VARC= .-5 ; POINTS TO THE VARIABLE (0).
.EVEN
.SBTTL END OF PASS ROUTINE

```

```

;*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO AGAIN

```

```

037632
037632 000004
037634 005037 001102
037640 005237 001174
037644 042737 100000 001174
037652 005327
037654 000001
037656 003123
037660 012737
037662 000001
037664 037654
037666 005037 177524
037672 005237 001176
037676 013746 001112
037702 013746 001174
037706 013746 050742
037712 104401 040136 104405
037720 104401 040155 104405
037726 104401 040165 104405
037734 104401 001163
037740 004737 045016
037744 104416
037746 023727 050746 000001
037754 003454
037756 032737 000340 177522
037764 001450
037766 042737 010000 177530
037774 012737 000254 177532
040002 000240
040004 032737 040000 177530 1$:
040012 001774
040014 042737 002000 177530
040022 052737 010000 177530
040030 032737 040000 177530 5$:
040036 001374
040040 042737 010000 177530
040046 013746 000134
040052 012737 040074 000134
040060 052737 000140 177530

```

```

$EOP:
SCOPE
CLR $TSTNM ;:ZERO THE TEST NUMBER
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;:YES
MOV (PC)+,$(PC)+ ;:RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT
CLR @#K2CSRC ;TURN OFF THE LEDS
INC $DEVCT
MOV $ERTTL,-(SP) ;: PUSH ERROR...
MOV $PASS,-(SP) ;:...AND PASS COUNTS.
MOV IOP.ID,-(SP) ;: .... AND IOP ID NUMBER
TYPE ;$EOPID,TYPDS
TYPE ;$EOPP,TYPDS
TYPE ;$EOPE,TYPDS
TYPE ;$CRLF
CALL ESUMRY ;: PRINT ERROR SUMMARY IF REQ'D.
A$MAIL ;: UPDATE ARBITERS APT-MAIL-BOX.
CMP @#IOPCNT,@1 ;: IS THERE MORE THAN 1 IOP BEING TESTED?
BLE 4$ ;: IF NOT JUST KEEP TESTING THIS ONE
BIT #340,@#K2CSR0 ;: IF IOP=0 OR 1 FORGET IT
BEQ 4$
BIC @BIT12,@#K2CSR0 ;: TURN OFF QIR INTERRUPT
MOV @NXTIOP,@#K2QIR
NOP
BIT @BIT14,@#K2CSR0 ;: WAS VECTOR WRITTEN?
BEQ 1$
BIC @BIT10,@#K2CSR0 ;: DON'T ALLOW QBUS RESET INTERRUPT
BIS @BIT12,@#K2CSR0 ;: ALLOW Q BUS INTERRUPT
BIT @BIT14,@#K2CSR0 ;: BIAK RECEIVED??
BNE 5$
BIC @BIT12,@#K2CSR0 ;: NO INTERRUPTS TO Q BUS
MOV @#134,-(SP) ;: SAVE VECTOR
MOV @2$,@#134 ;: LOAD VECTOR TO LET US RESTART
BIS @140,@#K2CSR0 ;: ALLOW INTERRUPT WHEN TPR12 IS WRITTEN

```

```

040066 000234          SPL      4          ;; ...BY ARBITER
040070 000001          WAIT          ;; LOWER PRIORITY LEVEL
040072 000000          HALT          ;; WAIT HERE FOR INTERRUPT

;
; COMES HERE WHEN ARBITER WRITES TPR12
040074 012716 040102 2$: MOV      #3$(SP)      ;; SET UP STACK FOR FAKE RETURN
040100 000002          RTI
040102 012637 000134 3$: MOV      (SP)+, @#134    ;; RSTORE VECTOR
040106          4$:
040106 013700 000042 $GET42: MOV      @#42,R0      ;; GET MONITOR ADDRESS
040112 001405          BEQ      $DOAGN      ;; BRANCH IF NO MONITOR
040114 000005          RESET          ;; CLEAR THE WORLD
040116 004710          $ENDAD: JSR     PC,(R0)    ;; GO TO MONITOR
040120 000240          NOP
040122 000240          NOP          ;; SAVE ROOM
040124 000240          NOP          ;; FOR
040126          $DOAGN:          ;; ACT11
040126 000137          JMP      @PC)+      ;; RETURN
040130 003064          $RTNAD: .WORD  AGAIN
040132          377      000 $ENULL: .BYTE  -1,-1,0    ;; NULL CHARACTER STRING
          .EVEN

5451          ;
5452          ; END-PASS TEXT STRINGS.
5453          ;
5454 040136      200      113      130 $EOPID: .ASCIZ  <CRLF>/KXJ11-CA ID# /
5455 040155      054      040      120 $EOPP: .ASCIZ  /, PASS /
5456 040165      054      040      124 $EOPE: .ASCIZ  /, TOTAL ERRORS /
5457          .EVEN

```

5467

.SBTTL SCOPE HANDLER ROUTINE

```

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW09=1      LOOP ON ERROR
;SW08=1      LOOP ON TEST IN SWR<6:0>
;CALL
;*          SCOPE          ;;SCOPE-IOT
  
```

```

040206          *SCOPE:
040206 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT SWR
040210 021627 001000          CMP      (SP),#1000          ;; IOT CALL FROM TRAP CATCHER ??
040214 101005          BHI      1$          ;; SKIP IF NOT.
040216 162716 000004          SUB      #4,(SP)          ;; ADJUST VECTOR...
040222 012616          MOV      (SP)+,(SP)          ;;...POP IT ONCE...
040224 000137 002416          JMP      UNXTRP          ;;...AND DO SOMETHING !!
040230          1$:
;*****START OF CODE FOR THE XOR TESTER*****
;XTSTR: BR      6$          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
040232 013746 000004          MOV      @#ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
040236 012737 040256 000004          MOV      #5,@#ERRVEC          ;;SET FOR TIMEOUT
040244 005737 177060          TST      @#177060          ;;TIME OUT ON XOR?
040250 012637 000004          MOV      (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
040254 000450          BR      $SVLAD          ;;GO TO THE NEXT TEST
040256 022626          5$:      CMP      (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
040260 012637 000004          MOV      (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
040264 000436          BR      7$          ;;LOOP ON THE PRESENT TEST
040266          6$:;*****END OF CODE FOR THE XOR TESTER*****
040266 032777 000400 140644          BIT      #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
040274 001423          BEQ      2$          ;;BR IF NO
040276 005046          CLR      -(SP)          ;;CLEAR A TEMP. LOCATION
040300 117716 140634          MOVB    @SWR,(SP)          ;;PICKUP THE DESIRED TEST NUMBER
040304 042716 000200          BIC      #$SWRMK,(SP)          ;;MASK OUT UNDESIRED BITS
040310 001414          BEQ      8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
040312 022716 000016          CMP      #16,(SP)          ;;CHECK THE NUMBER IN THE SWR
040316 002411          BLT      8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
040320 011637 001102          MOV      (SP),$TSTNM          ;;UPDATE THE TEST NUMBER
040324 005316          DEC      (SP)          ;;BACKUP BY ONE
040326 006316          ASL      (SP)          ;;SCALE THE TEST NUMBER AS AN INDEX
040330 062716 040446          ADD      #$SW08TBL,(SP)          ;;FORM THE ADDRESS OF TEST POINTER
040334 013637 001106          MOV      @((SP)+,$LPADR)          ;;SET LOOP ADDRESS TO DESIRED TEST
040340 000434          BR      $OVER          ;;GO LOOP ON THE TEST
040342 005726          8$:      TST      (SP)+          ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
040344 105737 001103          2$:      TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
040350 001412          BEQ      $SVLAD          ;;BR IF NO
040352 032777 001000 140560          BIT      #BIT09,@SWR          ;;LOOP ON ERROR?
040360 001404          BEQ      4$          ;;BR IF NO
040362 013737 001110 001106          7$:      MOV      $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
040370 000420          BR      $OVER
040372 105037 001103          4$:      CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
040376 105237 001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
040402 113737 001102 001172          MOVB    $TSTNM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX
040410 011637 001106          MOV      (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
  
```

```

040414 011637 001110      MOV      (SP), $LPERR      ;;SAVE ERROR LOOP ADDRESS
040420 005037 001160      CLR      $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
040424 112737 000001 001115      MOVB     #1, $ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
040432 013777 001102 140502 $OVER:   MOV      $TSTNM, @DISPLAY  ;;DISPLAY TEST NUMBER
040440 013716 001106      MOV      $LPADR, (SP)    ;;FUDGE RETURN ADDRESS
040444 000002      RTI                      ;;FIXES PS
040446      $SW08TBL:
      .REPT  $TN 1
040446 003162      .WORD   TST1+2          ;;STARTING ADDRESS OF TEST 1
040450 003406      .WORD   TST2+2          ;;STARTING ADDRESS OF TEST 2
040452 005654      .WORD   TST3+2          ;;STARTING ADDRESS OF TEST 3
040454 011314      .WORD   TST4+2          ;;STARTING ADDRESS OF TEST 4
040456 012510      .WORD   TST5+2          ;;STARTING ADDRESS OF TEST 5
040460 013030      .WORD   TST6+2          ;;STARTING ADDRESS OF TEST 6
040462 015760      .WORD   TST7+2          ;;STARTING ADDRESS OF TEST 7
040464 022136      .WORD   TST10+2         ;;STARTING ADDRESS OF TEST 10
040466 024604      .WORD   TST11+2        ;;STARTING ADDRESS OF TEST 11
040470 026222      .WORD   TST12+2        ;;STARTING ADDRESS OF TEST 12
040472 027414      .WORD   TST13+2        ;;STARTING ADDRESS OF TEST 13
040474 032126      .WORD   TST14+2        ;;STARTING ADDRESS OF TEST 14
040476 033176      .WORD   TST15+2        ;;STARTING ADDRESS OF TEST 15
040500 037200      .WORD   TST16+2        ;;STARTING ADDRESS OF TEST 16

```

5469

.SBTTL TYPE ROUTINE

```

;*****
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;* TYPE
;* MESADR
;*
040502 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
040506 100002 BPL 1$ ;;BR IF YES
040510 000000 HALT ;;HALT HERE IF NO TERMINAL
040512 000430 BR 3$ ;;LEAVE
040514 010046 1$: MOV RO,-(SP) ;;SAVE RO
040516 017600 000002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
040522 122737 000001 001206 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
040530 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
040532 132737 000100 001207 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
040540 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
040542 010037 040552 MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
040546 004737 042542 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
040552 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
040554 132737 000040 001207 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
040562 001003 BNE 60$ ;;YES,SKIP TYPE OUT
040564 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
040566 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
040570 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
040572 012600 60$: MOV (SP)+,RO ;;RESTORE RO
040574 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
040600 000002 RTI ;;RETURN
040602 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
040606 001430 BEQ 8$
040610 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
040614 001006 BNE 5$
040616 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
040620 104401 TYPE ;;TYPE A CR AND LF
040622 001163 $CRLF
040624 105037 041032 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
040630 000755 BR 2$ ;;GET NEXT CHARACTER
040632 004737 040714 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
040636 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
040642 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
040644 013746 001154 MOV $NULL,(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
040650 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
040654 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
040656 004737 040714 JSR PC,$TYPEC ;;GO TYPE A NULL
040662 105337 041032 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
040666 000770 BR 7$ ;;LOOP

```

```

;HORIZONTAL TAB PROCESSOR

040670 112716 000040      8$:   MOVB   #' ,(SP)      ;;REPLACE TAB WITH SPACE
040674 004737 040714      9$:   JSR    PC,$TYPEC     ;;TYPE A SPACE
040700 132737 000007 041032  BITB   #7,$CHARCNT     ;;BRANCH IF NOT AT
040706 001372              BNE    9$              ;;TAB STOP
040710 005726              TST    (SP)+           ;;POP SPACE OFF STACK
040712 000724              BR     2$              ;;GET NEXT CHARACTER
040714                                $TYPEC:
040714 105777 140224      TSTB   @TKS            ;;CHAR IN KYBD BUFFER?      ;MJD001
040720 100022              BPL    10$            ;;BR IF NOT                ;MJD001
040722 017746 140220      MOV     @TKB,-(SP)     ;;GET CHAR                  ;MJD001
040726 042716 177600      BIC    #177600,(SP)   ;;STRIP EXTRANEIOUS BITS   ;MJD001
040732 122716 000023      CMPB   #XOFF,(SP)    ;;WAS CHAR XOFF            ;MJD001
040736 001012              BNE    102$          ;;BR IF NOT                ;MJD001
040740                                101$:
040740 105777 140200      TSTB   @TKS            ;;WAIT FOR CHAR            ;MJD001
040744 100375              BPL    101$          ;;BR IF NOT                ;MJD001
040746 117716 140174      MOVB   @TKB,(SP)     ;;GET CHAR                  ;MJD001
040752 042716 177600      BIC    #177600,(SP)   ;;STRIP IT                  ;MJD001
040756 122716 000021      CMPB   #XON,(SP)    ;;WAS IT XON?             ;MJD001
040762 001366              BNE    101$          ;;BR IF NOT                ;MJD001
040764                                102$:
040764 005726              TST    (SP)+         ;;FIX STACK                ;MJD001
040766                                10$:
040766 105777 140156      TSTB   @TPS            ;;WAIT UNTIL PRINTER IS READY ;MJD001
040772 100375              BPL    10$            ;;BR IF NOT                ;MJD001
040774 116677 000002 140150  MOVB   2(SP),@TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
041002 122766 000015 000002  CMPB   #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
041010 001003              BNE    1$            ;;BRANCH IF NO
041012 105037 041032      CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
041016 000406              BR     $TYPEX        ;;EXIT
041020 122766 000012 000002  1$:   CMPB   #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
041026 001402              BEQ    $TYPEX        ;;BRANCH IF YES
041030 105227              INCB   (PC)+         ;;COUNT THE CHARACTER
041032 000000      $CHARCNT: .WORD    0 ;;CHARACTER COUNT STORAGE
041034 000207      $TYPEX: RTS        PC

```

5471

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

;*****
;THIS ROUTINE IS USED TO CHANGE A 16 BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:

```

```

;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS                    ;;GO TO THE ROUTINE

```

```

041036          $TYPDS:
041036 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
041040 010146      MOV      R1,(SP)      ;;PUSH R1 ON STACK
041042 010246      MOV      R2,(SP)      ;;PUSH R2 ON STACK
041044 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
041046 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
041050 012746 020200  MOV      #20200,(SP)  ;;SET BLANK SWITCH AND SIGN
041054 016605 000020  MOV      20(SP),R5  ;;GET THE INPUT NUMBER
041060 100004      BPL      1$      ;;BR IF INPUT IS POS.
041062 005405      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
041064 112766 000055 000001  MOVB    #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
041072 005000      CLR      R0      ;;ZERO THE CONSTANTS INDEX
041074 012703 041252      MOV      #DBLK,R3  ;;SETUP THE OUTPUT POINTER
041100 112723 000040      MOVB    #' ,(R3)+  ;;SET THE FIRST CHARACTER TO A BLANK
041104 005002      CLR      R2      ;;CLEAR THE BCD NUMBER
041106 016001 041242      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
041112 160105      SUB      R1,R5  ;;FORM THIS BCD DIGIT
041114 002402      BLT      4$      ;;BR IF DONE
041116 005202      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
041120 000774      BR      3$
041122 060105      ADD      R1,R5  ;;ADD BACK THE CONSTANT
041124 005702      TST      R2      ;;CHECK IF BCD DIGIT=0
041126 001002      BNE      5$      ;;FALL THROUGH IF 0
041130 105716      TSTB   (SP)      ;;STILL DOING LEADING 0'S?
041132 100407      BMI      7$      ;;BR IF YES
041134 106316      ASLB   (SP)      ;;MSD?
041136 103003      BCC      6$      ;;BR IF NO
041140 116663 000001 177777  MOVB    1(SP),-1(R3)  ;;YES--SET THE SIGN
041146 052702 000060      BIS      #'0,R2  ;;MAKE THE BCD DIGIT ASCII
041152 052702 000040      BIS      #' ,R2  ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
041156 110223      MOVB    R2,(R3)+  ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
041160 005720      TST      (R0)+  ;;JUST INCREMENTING
041162 020027 000010      CMP      R0,#10  ;;CHECK THE TABLE INDEX
041166 002746      BLT      2$      ;;GO DO THE NEXT DIGIT
041170 003002      BGT      8$      ;;GO TO EXIT
041172 010502      MOV      R5,R2  ;;GET THE LSD
041174 000764      BR      6$      ;;GO CHANGE TO ASCII
041176 105726      TSTB   (SP)+  ;;WAS THE LSD THE FIRST NON-ZERO?
041200 100003      BPL      9$      ;;BR IF NO
041202 116663 177777 177776  MOVB    -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
041210 105013      CLRB   (R3)      ;;SET THE TERMINATOR
041212 012605      MOV      (SP)+,R5  ;;POP STACK INTO R5
041214 012603      MOV      (SP)+,R3  ;;POP STACK INTO R3
041216 012602      MOV      (SP)+,R2  ;;POP STACK INTO R2
041220 012601      MOV      (SP)+,R1  ;;POP STACK INTO R1

```

B12

```

041222 012600          MOV    (SP)+,R0      ;;POP STACK INTO R0
041224 104401 041252  TYPE    ,#DBLK      ;;NOW TYPE THE NUMBER
041230 016666 000002 000004  MOV    2(SP),4(SP)   ;;ADJUST THE STACK
041236 012616          MOV    (SP)+,(SP)
041240 000002          RTI                    ;;RETURN TO USER
041242 023420          $DTBL: 10000.
041244 001750          1000.
041246 000144          100.
041250 000012          10.
041252          $DBLK: .BLKW 4
;//////////////////// OVERLAY PATCH //////////////////////////////////////
;
; BACK UP AND PATCH $TYPDS TO REALLY NULL LEAD ZEROS
; INSTEAD OF REPLACING THEM WITH SPACES.
; NOTE: 100000(8) GETS PRINTED AS -0 (IT'S REALLY -32768.).
;
5472          SVPC=          ; SAVE PC...
5473          .=$DTBL-32     ; ...AND POINT TO 9$.
5474          9$: BR      $$TDX ; BR TO PATCH.
5475          .=$VPC
5476          9$: BR      $$TDX
5477          .=$VPC
5478 041262          $$TDX: CLR    (R3)      ; SET STRING TERMINATOR.
5479 041210          1$: CMP    -(R3),#40
5480 000424          BNE    2$          ; SKIP IF NOT <SP>.
5481 041262          MOV    #177,(R3)     ; ELSE, CHANGE TO A <NULL>.
5482 105013          2$: CMP    R3,#$DBLK
5483 041264 124327 000040  BHI    1$          ; LOOP 'TIL DONE.
5484 041270 001002          BR      $DTBL-30 ; RETURN TO 9$+2.
5485 041272 112713 000177
5486 041276 020327 041252
5487 041302 101370
5488 041304 000742

```


C12

5490

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;*      MOV      NUM, (SP)      ;;NUMBER TO BE TYPED
;*      TYPOS    ;;CALL FOR TYPEOUT
;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;*      .BYTE   M              ;;M=1 OR 0
;*                               ;;1=TYPE LEADING ZEROS
;*                               ;;0=SUPPRESS LEADING ZEROS
;
;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;*$TYPOS OR $TYPOC
;CALL:
;*      MOV      NUM, -(SP)     ;;NUMBER TO BE TYPED
;*      TYPON    ;;CALL FOR TYPEOUT
;
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;CALL:
;*      MOV      NUM, -(SP)     ;;NUMBER TO BE TYPED
;*      TYPOC    ;;CALL FOR TYPEOUT

```

```

041306 017646 000000          $TYPOS: MOV      @ (SP), -(SP)      ;;PICKUP THE MODE
041312 116637 000001 041531  MOVVB   1 (SP), $OFILL    ;;LOAD ZERO FILL SWITCH
041320 112637 041533          MOVVB   (SP)+, $OMODE+1  ;;NUMBER OF DIGITS TO TYPE
041324 062716 000002          ADD      @2, (SP)      ;;ADJUST RETURN ADDRESS
041330 000406                    BR      $TYPON
041332 112737 000001 041531  $TYPOC: MOVVB   #1, $OFILL    ;;SET THE ZERO FILL SWITCH
041340 112737 000006 041533  MOVVB   #6, $OMODE+1  ;;SET FOR SIX(6) DIGITS
041346 112737 000005 041530  $TYPON: MOVVB   #5, $OCNT    ;;SET THE ITERATION COUNT
041354 010346                    MOV      R3, -(SP)      ;;SAVE R3
041356 010446                    MOV      R4, -(SP)      ;;SAVE R4
041360 010546                    MOV      R5, -(SP)      ;;SAVE R5
041362 113704 041533          MOVVB   $OMODE+1, R4  ;;GET THE NUMBER OF DIGITS TO TYPE
041366 005404                    NEG      R4
041370 062704 000006          ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
041374 110437 041532          MOVVB   R4, $OMODE    ;;SAVE IT FOR USE
041400 113704 041531          MOVVB   $OFILL, R4    ;;GET THE ZERO FILL SWITCH
041404 016605 000012          MOV      12 (SP), R5  ;;PICKUP THE INPUT NUMBER
041410 005003                    CLR      R3          ;;CLEAR THE OUTPUT WORD
041412 006105                    1$:  ROL      R5          ;;ROTATE MSB INTO "C"
041414 000404                    BR      3$          ;;GO DO MSB
041416 006105                    2$:  ROL      R5          ;;FORM THIS DIGIT
041420 006105                    ROL      R5
041422 006105                    ROL      R5
041424 010503                    MOV      R5, R3
041426 006103                    3$:  ROL      R3          ;;GET LSB OF THIS DIGIT
041430 105337 041532          DECB   $OMODE        ;;TYPE THIS DIGIT?
041434 100016                    BPL     7$          ;;BR IF NO
041436 042703 177770          BIC     #177770, R3  ;;GET RID OF JUNK
041442 001002                    BNE     4$          ;;TEST FOR 0
041444 005704                    TST     R4          ;;SUPPRESS THIS 0?
041446 001403                    BEQ     5$          ;;BR IF YES
041450 005204                    4$:  INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S

```

D12

KXJ11-CA FUNCTIONAL TEST
BINARY TO OCTAL (ASCII) AND TYPE

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 145

041452	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
041456	052703	000040	5\$:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
041462	110337	041526		MOVB	R3,B\$::SAVE FOR TYPING
041466	104401	041526		TYPE	.8\$::GO TYPE THIS DIGIT
041472	105337	041530	7\$:	DECB	\$OCNT	::COUNT BY 1
041476	003347			BGT	2\$::BR IF MORE TO DO
041500	002402			BLT	6\$::BR IF DONE
041502	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
041504	000744			BR	2\$::GO DO THE LAST DIGIT
041506	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
041510	012604			MOV	(SP)+,R4	::RESTORE R4
041512	012603			MOV	(SP)+,R3	::RESTORE R3
041514	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
041522	012616			MOV	(SP)+,(SP)	
041524	000002			RTI		::RETURN
041526	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
041527	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
041530	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
041531	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
041532	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

3492

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16 BIT
 ;*BINARY-ASCII NUMBER AND TYPE IT.

```

;*C' ':
;*      MOV      NUMBER, (SP)      ;;NUMBER TO BE TYPED
;*      TYPBN
;
041534 010146          $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
041536 016601 000006  MOV      6(SP),R1      ;;GET THE INPUT NUMBER
041542 000261          SEC                          ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
041544 112737 000060 041606 1$:  MOVB     #'0,$BIN      ;;SET CHARACTER TO AN ASCII "0".
041552 006101          ROL      R1                          ;;GET THIS BIT
041554 001406          BEQ      2$                          ;;DONE?
041556 105537 041606  ADCB     $BIN      ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
041562 104401 041606  TYPE     , $BIN      ;;GO TYPE THIS BIT
041566 000241          CLC                          ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
041570 000765          BR       1$                          ;;GO DO THE NEXT BIT
041572 012601          MOV      (SP)+,R1      ;;POP THE STACK INTO R1
041574 016666 000002 000004 2$:  MOV      2(SP),4(SP)      ;;ADJUST THE STACK
041602 012616          MOV      (SP)+,(SP)
041604 000002          RTI                          ;;RETURN TO USER
041606      000      000  $BIN:  .BYTE  0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
  
```

5494

.SBTTL TTY INPUT ROUTINE

```

;*****
.ENABL LSB

;*****
;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
;*WHEN OPERATING IN TTY FLAG MODE.
041610 022737 000176 001140 $CKSWR: CMP      $SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
041616 001074                BNE      15$          ;; BRANCH IF NO
041620 105777 137320        TSTB     @$TKS          ;; CHAR THERE?
041624 100071                BPL      15$          ;; IF NO, DON'T WAIT AROUND
041626 117746 137314        MOVB     @$TKB,-(SP)    ;; SAVE THE CHAR
041632 042716 177600        BIC      #+C177,(SP)  ;; STRIP-OFF THE ASCII
041636 022726 000007        CMP      #7,(SP)+     ;; IS IT A CONTROL G?
041642 001062                BNE      15$          ;; NO, RETURN TO USER
041644 123727 001134 000001  CMPB     $AUTOB,#1    ;; ARE WE RUNNING IN AUTO-MODE?
041652 001456                BEQ      15$          ;; BRANCH IF YES

041654 104401 042345                TYPE     , $CNTLG     ;; ECHO THE CONTROL-G (+G)
041660 104401 042352        $GTSWR: TYPE     , $MSWR     ;; TYPE CURRENT CONTENTS
041664 013746 000176        MOV      SWREG,-(SP)  ;; SAVE SWREG FOR TYPEOUT
041670 104402                TYPOC    ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
041672 104401 042363        TYPE     , $MNEW     ;; PROMPT FOR NEW SWR
041676 005046                19$:    CLR      -(SP)    ;; CLEAR COUNTER
041700 005046                CLR      -(SP)    ;; THE NEW SWR
041702 105777 137236        7$:    TSTB     @$TKS          ;; CHAR THERE?
041706 100375                BPL      7$          ;; IF NOT TRY AGAIN

041710 117746 137232        MOVB     @$TKB,-(SP)  ;; PICK UP CHAR
041714 042716 177600        BIC      #+C177,(SP) ;; MAKE IT 7-BIT ASCII

041720 021627 000025        9$:    CMP      (SP),#25  ;; IS IT A CONTROL-U?
041724 001005                BNE      10$         ;; BRANCH IF NOT
041726 104401 042340        TYPE     , $CNTLU   ;; YES, ECHO CONTROL-U (+U)
041732 062706 000006        20$:   ADD      #6,SP    ;; IGNORE PREVIOUS INPUT
041736 000757                BR       19$         ;; LET'S TRY IT AGAIN

041740 021627 000015        10$:   CMP      (SP),#15   ;; IS IT A <CR>?
041744 001022                BNE      16$         ;; BRANCH IF NO
041746 005766 000004        TST      4(SP)       ;; YES, IS IT THE FIRST CHAR?
041752 001403                BEQ      11$         ;; BRANCH IF YES
041754 016677 000002 137156  MOV      2(SP),@SWR   ;; SAVE NEW SWR
041762 062706 000006        11$:   ADD      #6,SP    ;; CLEAR UP STACK
041766 104401 001163        14$:   TYPE     , $CRLF  ;; ECHO <CR> AND <LF>
041772 123727 001135 000001  CMPB     $INTAG,#1   ;; RE-ENABLE TTY KBD INTERRUPTS?
042000 001003                BNE      15$         ;; BRANCH IF NOT
042002 012777 000100 137134  MOV      #100,@$TKS  ;; RE-ENABLE TTY KBD INTERRUPTS
042010 000002                15$:   RTI              ;; RETURN
042012 004737 040714        16$:   JSR      PC,$TYPEC ;; ECHO CHAR
042016 021627 000060        CMP      (SP),#60   ;; CHAR < 0?
042022 002420                BLT      18$         ;; BRANCH IF YES
    
```

```

042024 021627 000067      CMP      (SP),#67      ;;CHAR > ??
042030 003015             BGT      18$           ;;BRANCH IF YES
042032 042726 000060      BIC      #60,(SP)+     ;;STRIP-OFF ASCII
042036 005766 000002      TST      2(SP)        ;;IS THIS THE FIRST CHAR
042042 001403             BEQ      17$           ;;BRANCH IF YES
042044 006316             ASL      (SP)         ;;NO, SHIFT PRESENT
042046 006316             ASL      (SP)         ;; CHAR OVER TO MAKE
042050 006316             ASL      (SP)         ;; ROOM FOR NEW ONE.
042052 005266 000002      17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
042056 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
042062 000707             BR       7$           ;;GET THE NEXT ONE
042064 104401 001162      18$: TYPE   ,#QUES     ;;TYPE ?<CR><LF>
042070 000720             BR       20$        ;;SIMULATE CONTROL-U
.DSABL  LSB

```

*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

*CALL:

```

*      RDCHR           ;;INPUT A SINGLE CHARACTER FROM THE TTY
*      RETURN HERE     ;;CHARACTER IS ON THE STACK
*                               ;;WITH PARITY BIT STRIPPED OFF
;

```

```

042072 011646             $RDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC
042074 016666 000004 000002  MOV      4(SP),2(SP)  ;;SAVE THE PS
042102 105777 137036      1$: TSTB   @TKS        ;;WAIT FOR
042106 100375             BPL      1$           ;;A CHARACTER
042110 117766 137032 000004  MOVB   @TKB,4(SP)     ;;READ THE TTY
042116 042766 177600 000004  BIC    #C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
042124 026627 000004 000023  CMP    4(SP),#23     ;;IS IT A CONTROL-S?
042132 001013             BNE     3$           ;;BRANCH IF NO
042134 105777 137004      2$: TSTB   @TKS        ;;WAIT FOR A CHARACTER
042140 100375             BPL     2$           ;;LOOP UNTIL ITS THERE
042142 117746 137000      MOVB   @TKB,-(SP)    ;;GET CHARACTER
042146 042716 177600      BIC    #C177,(SP)    ;;MAKE IT 7-BIT ASCII
042152 022627 000021      CMP    (SP)+,#21     ;;IS IT A CONTROL-Q?
042156 001366             BNE     2$           ;;IF NOT DISCARD IT
042160 000750             BR      1$           ;;YES, RESUME
042162 026627 000004 000021  3$: CMP    4(SP),#XON  ;;IS IT A RANDOM XON?           ;RAN001
042170 001744             BEQ     1$           ;;BRANCH IF YES                 ;RAN001
042172 026627 000004 000140  CMP    4(SP),#140    ;;IS IT UPPER CASE?
042200 002407             BLT     4$           ;;BRANCH IF YES
042202 026627 000004 000175  CMP    4(SP),#175    ;;IS IT A SPECIAL CHAR?
042210 003003             BGT     4$           ;;BRANCH IF YES
042212 042766 000040 000004  BIC    #40,4(SP)    ;;MAKE IT UPPER CASE
042220 000002             RTI     4$:      ;;GO BACK TO USER

```

*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

*CALL:

```

*      RDLIN          ;;INPUT A STRING FROM THE TTY
*      RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                               ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
;

```

```

042222 010346             $RDLIN: MOV      R3,-(SP)  ;;SAVE R3
042224 012703 042330      1$: MOV    #TTYIN,R3  ;;GET ADDRESS
042230 022703 042340      2$: CMP    #TTYIN+8.,R3 ;;BUFFER FULL?

```

H12

KXJ11-CA FUNCTIONAL TEST
TTY INPUT ROUTINE

042234	101405			BLOS	4\$::BR IF YES
042236	104411			RDCHR		::GO READ ONE CHARACTER FROM THE TTY
042240	112613			MOVB	(SP)+,(R3)	::GET CHARACTER
042242	122713	000177		CMPB	#177 (R3)	::IS IT A RUBOUT
042246	001003		10\$:	BNE	3\$::SKIP IF NOT
042250	104401	001162		TYPE	,\$QUES	::TYPE A '?'
042254	000763			BR	1\$::CLEAR THE BUFFER AND LOOP
042256	111337	042326		MOVB	(R3),9\$::ECHO THE CHARACTER
042262	104401	042326	3\$:	TYPE	,9\$	
042266	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
042272	001356			BNE	2\$::LOOP IF NOT RETURN
042274	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
042300	104401	001164		TYPE	,\$LF	::TYPE A LINE FEED
042304	012603			MOV	(SP)+,R3	::RESTORE R3
042306	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
042310	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
042316	012766	042330	000004	MOV	#\$TTYIN,4(SP)	
042324	000002			RTI		::RETURN
042326	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
042327	000			.BYTE	0	::TERMINATOR
042330				\$TTYIN:	.BLKB	8.
042340	136	125	015	\$CNTLU:	.ASCIZ	/+U/<15><12>
042345	136	107	015	\$CNTLG:	.ASCIZ	/+G/<15><12>
042352	015	012	123	\$MSWR:	.ASCIZ	<15><12>/SWR = /
042363	040	040	116	\$MNEW:	.ASCIZ	/ NEW = /

5496

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

```

;*****
;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;CHANGE IT TO BINARY.
;THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
;OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
;FOLLOWED BY A CARRIAGE RETURN LINE FEED. THE COMPLETE NUMBER MUST
;THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
;CALL:
;*      RDOCT          ;;READ AN OCTAL NUMBER
;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
;*                  ;;HIGH ORDER BITS ARE IN $HIOCT

042374 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
042376 016666          MOV      4(SP),2(SP)      ;;INPUT NUMBER
042404 010046          MOV      RO,-(SP)        ;;PUSH RO ON STACK
042406 010146          MOV      R1,-(SP)        ;;PUSH R1 ON STACK
042410 010246          MOV      R2,-(SP)        ;;PUSH R2 ON STACK
042412 104412          1$:  RDLIN          ;;READ AN ASCII LINE
042414 012600          MOV      (SP)+,RO        ;;GET ADDRESS OF 1ST CHARACTER
042416 010037          MOV      RO,5$          ;;AND SAVE IT
042422 005001          CLR      R1          ;;CLEAR DATA WORD
042424 005002          CLR      R2
042426 112046          2$:  MOVB      (RO)+,-(SP)      ;;PICKUP THIS CHARACTER
042430 001420          BEQ      3$          ;;IF ZERO GET OUT
042432 122716          CMPB     #'0,(SP)        ;;MAKE SURE THIS CHARACTER
042436 003026          BGT      4$          ;;IS AN OCTAL DIGIT
042440 122716          CMPB     #'7,(SP)
042444 002423          BLT      4$
042446 006301          ASL      R1          ;;*2
042450 006102          ROL      R2
042452 006301          ASL      R1          ;;*4
042454 006102          ROL      R2
042456 006301          ASL      R1          ;;*8
042460 006102          ROL      R2
042462 042716          BIC      #'C7,(SP)      ;;STRIP THE ASCII JUNK
042466 062601          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
042470 000756          BR       2$          ;;LOOP
042472 005726          3$:  TST      (SP)+          ;;CLEAN TERMINATOR FROM STACK
042474 010166          MOV      R1,12(SP)      ;;SAVE THE RESULT
042500 010237          MOV      R2,$HIOCT
042504 012602          MOV      (SP)+,R2      ;;POP STACK INTO R2
042506 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
042510 012600          MOV      (SP)+,RO      ;;POP STACK INTO RO
042512 000002          RTI          ;;RETURN
042514 005726          4$:  TST      (SP)+          ;;CLEAN PARTIAL FROM STACK
042516 105010          CLRB     (RO)          ;;SET A TERMINATOR
042520 104401          TYPE          ;;TYPE UP THRU THE BAD CHAR.
042522 000000          5$:  .WORD     0
042524 104401          TYPE     , $QUES      ;; "?" "CR" & "LF"
042530 000730          BR       1$          ;;TRY AGAIN
042532 000000          $HIOCT: .WORD     0      ;;HIGH ORDER BITS GO HERE
  
```

5498

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
042534 112737 000001 043000 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
042542 112737 000001 042776 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
042550 000403 BR $ATYC
042552 112737 000001 043000 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
042560 $ATYC:
042560 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
042562 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
042564 105737 042776 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
042570 001450 BEQ 5$ ;;IF NOT: BR
042572 122737 000001 001206 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
042600 001031 BNE 3$ ;;IF NOT: BR
042602 132737 000100 001207 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
042610 001425 BEQ 3$ ;;IF NOT: BR
042612 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
042616 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
042624 005737 001166 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
042630 001375 BNE 1$ ;;IF NOT: WAIT
042632 010037 001202 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
042636 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
042640 001376 BNE 2$
042642 163700 001202 SUB $MSGAD,RO ;;SUB START OF MESSAGE
042646 006200 ASR RO ;;GET MESSAGE LNTH IN WORDS
042650 010037 001204 MOV RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
042654 012737 000004 001166 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
042662 000413 BR 5$
042664 017637 000004 042710 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
042672 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
042700 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
042704 004737 040502 JSR PC,$TYPE ;;CALL TYPE MACRO
042710 000000 4$: .WORD 0
042712 5$:
042712 105737 043000 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
042716 001416 BEQ 12$ ;;IF NOT: BR
042720 005737 001206 TST $ENV ;;RUNNING UNDER APT?
042724 001413 BEQ 12$ ;;IF NOT: BR
042726 005737 001166 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
042732 001375 BNE 11$ ;;IF NOT: WAIT
042734 017637 000004 001170 MOV @4(SP),$FATAL ;;GET ERROR #
042742 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
042750 005237 001166 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
042754 105037 043000 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
042760 105037 042777 CLRB $LFLG ;;CLEAR LOG FLAG
042764 105037 042776 CLRB $MFLG ;;CLEAR MESSAGE FLAG
042770 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
042772 012600 MOV (SP)+,RO ;;POP STACK INTO RO
042774 000207 RTS PC ;;RETURN
042776 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
042777 000 $LFLG: .BYTE 0 ;;LOG FLAG
043000 000 $FFLG: .BYTE 0 ;;FATAL FLAG
.EVEN
000200 APTSIZE=200
000001 APTENV=001
000100 APTSPOOL=100
000040 APTCSUP=040

```


5500

.SBTTL TRAP DECODER

```

;*****
;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;GO TO THAT ROUTINE.

```

```

043002 010046          $TRAP:  MOV    RO,-(SP)      ;;SAVE R0
043004 016600 000002  MOV    2(SP),RO      ;;GET TRAP ADDRESS
043010 005740          TST    -(RO)         ;;BACKUP BY 2
043012 111000          MOV    (RO),RO        ;;GET RIGHT BYTE OF TRAP
043014 006300          ASL    RO             ;;POSITION FOR INDEXING
043016 016000 043036  MOV    $TRPAD(RO),RO  ;;INDEX TO TABLE
043022 000200          RTS     RO           ;;GO TO ROUTINE

```

```

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

```

043024 011646          $TRAP2: MOV   (SP),-(SP)  ;;MOVE THE PC DOWN
043026 016666 000004 000002 MOV   4(SP),2(SP)     ;;MOVE THE PSW DOWN
043034 000002          RTI                    ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;BY THE "TRAP" INSTRUCTION.

```

```

;          ROUTINE
;          -----
043036 043024          $TRPAD: .WORD  $TRAP2
043040 040502          $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEDOUT ROUTINE
043042 041332          $TYPOC  ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
043044 041306          $TYPOS  ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
043046 041346          $TYPON  ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
043050 041036          $TYPDS  ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
043052 041534          $TYPBN  ;;CALL=TYPBN     TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

043054 041660          $GTSWR  ;;CALL=GTSWR     TRAP+7(104407) GET SOFT-SWR SETTING

043056 041610          $CKSWR  ;;CALL=CKSWR     TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
043060 042072          $RDCHR  ;;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
043062 042222          $RDLIN  ;;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
043064 042374          $RDOCT  ;;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY

5501 043066 043076          I$SWR  ;;CALL=I$SWR     TRAP+14(104414) GET INITIAL SWITCH OPTIONS
5502 043070 043740          T$NAME  ;;CALL=T$NAME     TRAP+15(104415) PRINT TEST NUMBER AND TITLE
5503 043072 044046          A$MAIL  ;;CALL=A$MAIL     TRAP+16(104416) UPDATE ARBITERS APT MAIL-BOX
5504 043074 055746          IOP$TR  ;;CALL=IOP$TR    TRAP+17(104417) IOP INTERRUPT HANDLER

```

```

5507 ;*****
5508 ; ROUTINE TO CHECK OPERATING MODE AND ADJUST SET-UP ACCORDINGLY.
5509 ; IF STAND-ALONE, DISPLAY SWITCH OPTIONS AND GET INITIAL SWR.
5510 ;
5511 i$SWR:
5512 043076 005737 000042 TST @#42
5513 043102 001017 BNE 7$ ; BR IF CHAINED UNDER XXDP/ACT.
5514 043104 123727 001206 000001 CMPB $ENV,#APTENV
5515 043112 001410 BEQ 6$ ; BR IF RUNNING IN APT MODE.
5516 043114 023727 050746 000001 CMP @#IOPCNT,#1 ; ARE THERE MULTIPLE UNITS?
5517 043122 003012 BGT 8$ ; BR IF MORE THAN ONE UNIT
5518 043124 104401 043152 TYPE ,SWROPT ; DISPLAY SUPPORTED SWITCHES. ;FXC
5519 043130 104407 1$: GTSWR ; GET INITIAL SWITCH SETTING
5520 043132 000406 BR 8$
5521 ; *****
5522 ; $QDLY FOR APT MODE DOUBLED FROM 62. TO 248. TO DECREASE CHANCE OF TIMEOUT
5523 ; ERROR WHILE APT IS MONITORING. JEANELL CUNNINGHAM, 30 MAY, 1984.
5524 ; *****
5525 043134 012737 000370 001342 6$: MOV @248.,$QDLY ; APT MODE -- SET Q DELAY FOR 40 SECONDS.
5526 043142 112737 000001 001134 7$: MOVB #1,$AUTOB ; APT OR CHAINED - SET AUTO-MODE FLAG.
5527 043150 000002 8$: RTI ; RETURN.
5528
5529 043152 200 123 127 SWROPT: .ASCII <CRLF>' SWR OCTAL FUNCTION'
5530 043202 200 055 055 .ASCII <CRLF>' --- --- -'
5531 043232 200 040 061 .ASCII <CRLF>' 15 100000 HALT ON ERROR'
5532 043265 200 040 061 .ASCII <CRLF>' 14 040000 INHIBIT ERROR SUMMARY'
5533 043330 200 040 061 .ASCII <CRLF>' 13 020000 INHIBIT ERROR REPORTS'
5534 043373 200 040 061 .ASCII <CRLF>' 12 010000 IOP ID# 15 KNOWN GOOD FOR TESTING'
5535 043452 200 040 061 .ASCII <CRLF>' 11 004000 TEST STAND ALONE IOP'
5536 043514 200 040 061 .ASCII <CRLF>' 10 002000 ENABLE EXTENDED MEMORY TESTS'
5537 043566 200 040 060 .ASCII <CRLF>' 09 001000 LOOP ON ERROR'
5538 043621 200 040 060 .ASCII <CRLF>' 08 000400 LOOP ON TEST IN SWR<6:0>'
5539 043667 200 040 060 .ASCII <CRLF>' 07 000200 INHIBIT TEST NUMBER/TITLE'
5540 043736 200 000 .ASCIZ <CRLF>
5541 .EVEN
  
```

```

5543 ;*****
5544 ; ROUTINE TO PRINT TEST NUMBER AND NAME.
5545 ; ALSO, SETS TEST NUMBER IN CPU LEDS.
5546 ; IF SWR<7> = 1, DON'T PRINT ANYTHING.
5547 ;
5548 ; CALL: T$NAME, TXT ADDRESS
5549 ;
5550 043740 004737 037542 T$NAME: CALL TNLEDS ; PUT CURRENT TEST NUMBER IN LEDS.
5551 043744 104416 A$MAIL ; UPDATE ARBITERS COPY OF $MAIL.
5552 043746 042777 004000 135340 BIC #BIT11,@$CSR ; CLEAR RESET FLAG. ;FX8JC
5553 043754 052777 002000 135332 BIS #BIT10,@$CSR ; SET TRAP ON Q-BUS RESET ENABLE. ;FX8JC
5554 043762 032777 000200 135150 BIT #BIT7,@SWR
5555 043770 001023 BNE 4$ ; BR IF TITLES SUPPRESSED...
5556 043772 113727 001102 MOVB $TSTNM,(PC)+ ; OTHERWISE, SAVE THE NUMBER...
5557 043776 000000 1$: .WORD 0 ;...HERE.
5558 044000 104401 044006 TYPE ,30069$ ;:TYPE ASCIZ STRING
044004 000404 BR 30068$ ;:GET OVER THE ASCIZ
;:30069$: .ASCIZ <CRLF>/TEST /
30068$:
5559 044016 113746 001102 MOVB $TSTNM,-(SP)
5560 044022 104403 000402 TYPOS ,2!WLZ ;...AND NUMBER.
5561
5562 044026 017667 000000 000002 3$: MOV @ (SP),2(PC) ; SET TITLE ADDRESS...
5563 044034 104401 000000 TYPE ,0 ;...AND TYPE IT TOO.
5564 044040 062716 000002 4$: ADD #2,(SP) ; ADJUST RETURN PC...
5565 044044 000002 RTI ;...AND RETURN
5566
5567 ;*****
5568 ; ROUTINE TO COPY SLAVE MAILBOX UP-LINE TO THE ARBITER.
5569 ; BYPASS IF SBC IS NOT A SLAVE (ID 1 OR 0) OR NOT APT MODE.
5570 ;
5571 044046 A$MAIL:
5572 044046 032777 000340 135264 BIT #<16*BIT4>,@$TCID ; IF ID 0 OR 1...
5573 044054 001514 BEQ 8$ ;...DON'T BOTHER. ;FX9JC
5574 044056 123727 001206 000001 CMPB $ENV,@APTENV ; IF NOT RUNNING UNDER APT... ;FX9JC
5575 044064 001110 BNE 8$ ;...DITTO. ;FX9JC
5576 ; *****
5577 ; NOW COMPUTE A CHECKSUM ON THE MAILBOX. ;FX9JC
5578 ;
5579 044066 013705 177524 MOV @#K2CSRC,R5 ;GET THE IOP ID NUMBER
5580 044072 042705 177417 BIC #tC360,R5 ; MASK OFF OTHER BITS
5581 044076 000305 SWAB R5 ;SWAP BYTES
5582 044100 050537 001172 BIS R5,$TESTN ;PUT IOP ID CODE IN MAIL BOX SO THAT WE KNOW
5583 ; WHICH IOP IS REPORTING THE ERROR
5584 044104 012705 000020 MOV #16.,R5 ; BYTE COUNT IN R5. ;FX9JC
5585 044110 012704 001166 MOV #$MAIL,R4 ; STARTING ADDRESS IN R4. ;FX9JC
5586 044114 005037 044310 CLR CHKSUM ; CLEAR ACCUMULATOR FOR CHECKSUM. ;FX9JC
5587 044120 112403 1$: MOVB (R4)+,R3 ; GET A BYTE IN R3. ;FX9JC
5588 044122 042703 177400 BIC #tC377,R3 ; STRIP OFF HIGH BYTE IN R3. ;FX9JC
5589 044126 060337 044310 ADD R3,CHKSUM ; OPERATE ON THE BYTE. ;FX9JC
5590 044132 106337 044310 ASLB CHKSUM ; ;FX9JC
5591 044136 005537 044310 ADC CHKSUM ; ;FX9JC
5592 044142 077512 SOB R5,1$ ; LOOP UNTIL DONE. ;FX9JC
5593
5594 ; Routine has been modified to send 15 "tZ"'s if wait constant
5595 ; expired and "tZ" was not echoed from arbiter or if "NACK" has
5596 ; been received. After the third not echoed "tZ" the program still

```

```

5597          ; hangs in a wait loop.          KXTFXD 17-Sep 84
5598
5599 044144 012737 000017 044312      MOV    #17,CNTZ      ; COUNTER FOR 15 "+Z"'S      :FXD
5600 044152 012746 000032      MOV    #32,-(SP)    ; SLAVE N APT MODE -- PUSH... :FX9JC
5601 044156 004737 040714      2$:   CALL $TYPEC    ; ... AND SEND <tZ>.        :FX9JC
5602 044162 012705 177777      200$: MOV    #177777,R5 ; WAIT CONSTANT FOR "+Z"    :FXD
5603 044166 105777 134752      101$: TSTB   @TKS      ; WAIT FOR ARB TO ECHO <tZ> :FXAJC
5604 044172 100405                22$:   BMI    102$      ;                               :FXAJC
5605 044174 077504                SOB    R5,22$      ; WAIT UNTIL DELAY EXPIRED  :FXD
5606 044176 005337 044312      DEC    CNTZ        ; DECREMENT "+Z" COUNTER    :FXD
5607 044202 001365                BNE    200$        ; IF NOT 15 TIMES, SEND ONE MORE :FXD
5608 044204 000777                BR     .           ; OTHERWISE STAY IN WAIT FOREVER :FXD
5609 044206 117703 134734      102$: MOVB   @TKB,R3  ; GET BYTE IN R3.          :FXAJC
5610 044212 120327 000032      CMPB  R3,#32      ; WAS IT THE <tZ>?        :FXAJC
5611 044216 001363                BNE    22$        ; NO, WAIT FOR IT.        :FXAJC
5612 044220 012704 001166      3$:   MOV    #MAIL,R4 ;                               :FX9JC
5613 044224 012705 000020      MOV    #16.,R5    ;                               :FX9JC
5614 044230 112416                4$:   MOVB   (R4)+,(SP) ; GET A BYTE...          :FX9JC
5615 044232 004737 040714      CALL  $TYPEC      ; ... SEND IT ...        :FX9JC
5616 044236 077504                SOB    R5,4$      ; ... ETC.                :FX9JC
5617 044240 113716 044310      MOVB  CHKSUM,(SP) ; THEN PUSH CHECKSUM ...  :FX9JC
5618 044244 004737 040714      CALL  $TYPEC      ; ... AND SEND IT.        :FX9JC
5619 044250 005726                TST   (SP)+       ; FIX STACK...           :FX9JC
5620 044252 105777 134666      5$:   TSTB   @TKS      ; WAIT FOR CHARACTER TO COME BACK. :FX9JC
5621 044256 100375                BPL    5$         ;                               :FX9JC
5622 044260 017705 134662      6$:   MOV    @TKB,R5  ; GET THE CHARACTER IN R5.  :FX9JC
5623 044264 120527 000006      CMPB  R5,#ACK     ; WAS TRANSFER CORRECT?    :FX9JC
5624 044270 001404                BEQ    7$         ; YES, EXIT.              :FX9JC
5625 044272 120527 000025      CMPB  R5,#NACK    ; DID ARBITER SEND NACK?  :FX9JC
5626 044276 001725                BEQ    2$         ; YES, TRY AGAIN.        :FX9JC
5627 044300 000764                BR     5$         ; KEEP LOOKING FOR ACK OR NACK. :FX9JC
5628 044302 005037 001166      7$:   CLR    $MSGTYPE ; SAY "ARBITER'S MAILBOX HAS BEEN ... :FX9JC
5629                                     ; ... UPDATED"           :FX9JC
5630 044306 000002                8$:   RTI          ; ... AND WE'RE DONE.    :FX9JC
5631
5632 044310 000000      CHKSUM: .WORD 0
5633 044312 000000      CNTZ:   .WORD 0      ;COUNTER FOR "+Z"'S      :FXD

```

5635

.SBTTL ERROR HANDLER ROUTINE

```

;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;AND GO TO ERHAN ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1      HALT ON ERROR
;*SW09=1      LOOP ON ERROR
;*CALL
;*          ERROR      N          ;;ERROR-EMT AND N=ERROR ITEM NUMBER

```

```

044314          $ERROR:
044314 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
044316 105237 001103  7$:  INCB          $ERFLG          ;;SET THE ERROR FLAG
044322 001775          BEQ          7$          ;;DON'T LET THE FLAG GO TO ZERO
044324 013777 001102 134610  MOV          $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
044332 005237 001112          INC          $ERTTL          ;;INC THE ERROR COUNT
044336 011637 001116          MOV          (SP),$ERRPC          ;;GET ADDRESS OF ERROR INSTRUCTION
044342 162737 000002 001116  SUB          #2,$ERRPC
044350 117737 134542 001114  MOVB         @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
044356 021627 001002          CMP          (SP),#1002          ;;IF RETURN PC LESS THAN 1002
044362 101046          BHI          12$          ;;ERROR IS ILLEGAL TRAP

;;PROCESS UNEXPECTED TRAP OR INTERRUPT
044364 016637 000004 001116  MOV          4(SP),$ERRPC          ;;GET PC AT TIME OF FALSE TRAP
044372 162737 000002 001116  SUB          #2,$ERRPC          ;;ADJUST PC
044400 104401 044444          TYPE          ,10$          ;;TYPE HEADER
044404 013746 001116          MOV          $ERRPC,-(SP)          ;;SAVE $ERRPC FOR TYPEOUT
044410 104402          TYPOC          ;;GO TYPE -OCTAL ASCII(ALL DIGITS)
044412 104401 044452          TYPE          ,11$
044416 162716 000004          SUB          #4,(SP)          ;;GET FALSE TRAP VECTOR ADDR
044422 011637 001116          MOV          (SP),$ERRPC
044426 013746 001116          MOV          $ERRPC,-(SP)          ;;SAVE $ERRPC FOR TYPEOUT
044432 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
044434 104401 001163          TYPE          ,CRLF
044440 022626          CMP          (SP)+,(SP)+          ;;POP FALSE TRAP VECTOR PC&ADDR
044442 000420          BR          20$
044444          200          120          103 10$: .ASCIZ <200>'PC= '
044452          040          040          125 11$: .ASCIZ ' UNEXPECTED TRAP TO '
          .EVEN

044500          12$: JSR          PC,ERHAN          ;;GO TO USER ERROR ROUTINE
044500 004737 044630          20$: CMPB         #APTENV,$ENV          ;;RUNNING IN APT MODE
044504          BNE          2$          ;;NO SKIP APT ERROR REPORT
044504 122737 000001 001206  MOVB         $ITEMB,21$          ;;SET ITEM NUMBER AS ERROR NUMBER
044512 001007          JSR          PC,$ATY4          ;;REPORT FATAL ERROR TO APT
044514 113737 001114 044526  .BYTE          0          21$:
044522 004737 042552          .BYTE          0
044526          000          BR          22$          ;;APT ERROR LOOP
044527          000          TST          @SWR          ;;HALT ON ERROR
044530 000777          BPL          3$          ;;SKIP IF CONTINUE
044532 005777 134402          2$: HALT          ;;HALT ON ERROR!
044536 100002          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
044540 000000          BIT          #BIT09,@SWR          ;;LOOP ON ERROR SWITCH SET?
044542 104410          BEQ          4$          ;;BR IF NO
044544 032777 001000 134366 3$: MOV          $LPERR,(SP)          ;;FUDGE RETURN FOR LOOPING
044552 001402
044554 013716 001110

```

```

044560 005737 001160      4$:   TST   $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
044564 001402              BEQ   5$              ;;BR IF NONE
044566 013716 001160      MOV   $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
044572              5$:   CMP   ##ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
044600 022737 040116 000042 BNE   6$              ;;BRANCH IF NO
044602 001001              HALT                ;;YES
044604 000000              6$:   RTI                ;;RETURN
044604 000002              ;//////////////////// OVERLAY PATCH //////////////////////
5636              ;
5637              ;
5638              ; PATCH $ERROR TO UPDATE ARBITERS MAIL-BOX IF NECESSARY
5639              ; BEFORE TAKING THE APT-ERROR-STALL.
5640              ;
5641              ;
5642              SVPC=.
5643 044530 000426      . =22$
5644              BR      222$
5645              ; BACK UP TO THE "APT-ERROR-STALL"...
5646 044606 017746 134502 222$:  MOV   @#CSR,-(SP)    ; ...AND CHANGE IT.
5647 044612 000005      RESET
5648              ; =SVPC
5649 044614 012777 002000 134472 MOV   #2000,@#CSR   ; SAVE PRIMARY CSR (FOR DEBUG)...
5650 044622 104416      A$MAIL
5651 044624 000777      BR      2$          ; ...AND RESET TO ENSURE THERE AREN'T...
5652 044626 000741      BR      2$          ; ...ANY Q-BUS INTERRUPTS LEFT HANGING.
                    ; RE-ARM TRAP ON Q-BUS RESET.
                    ; UPDATE ARBITERS $MAIL...
                    ; ...AND STALL HERE !!
                    ;FX8JC

```

D13

KXJ11 CA FUNCTIONAL TEST
 ERROR HANDLER ROUTINE

MACRO Y05.02 Thursday 03-Apr 86 14:11 Page 158

```

5654
5655
5656
5657
5658
5659
5660 044630 010046
5661 044632 113700 001114
5662 044636 006300
5663 044640 005260 051034
5664 044644 042737 100000 001112
5665 044652 032777 020000 134260
5666 044660 001037
5667
5668 044662 005046
5669 044664 113716 001114
5670 044670 104401 044764 104403 1#:
5671 044700 113746 001102
5672 044704 104401 044774 104403
5673 044714 013746 001116
5674 044720 104401 045006 104402
5675 044726 104401 001163
5676 044732 005737 044742
5677 044736 001402
5678 044740 104401
5679 044742 000000
5680 044744 104401
5681 044746 000000
5682 044750 004737
5683 044752 000000
5684 044754 104401 001163
5685
5686 044760 012600
5687 044762 000207
5688
5689 044764 200 105 122 EH1: .ASCIZ <CRLF>/ERROR /
5690 044774 040 111 116 EH2: .ASCIZ / IN TEST /
5691 045006 040 101 124 EH3: .ASCIZ / AT PC /
5692 .EVEN

```

```

;
; ERROR SIGNATURE HANDLER.
; LOG THE NUMBER OF OCCURENCES OF EACH ERROR IN "ELOG".
; IF SWR<13>=0, PRINT A COMMON HEADER AND A BASIC ERROR MESSAGE
; AND FINISH UP IN ONE OF SEVERAL SIGNATURE FORMAT SUBROUTINES.
;
ERHAN: MOV RO,-(SP) ; SAVE RO.
        MOVB $ITEMB,RO ; GET ERROR NUMBER...
        ASL RO ; ...SHIFT UP TO WORD OFFSET...
        INC ELOG(RO) ; ...AND TALLY THIS ERROR.
        BIC #BIT15,$ERTTL ; KEEP TOTAL ERROR COUNT POS.
        BIT #BIT13,@SWR ; REPORTS INHIBITED ??
        BNE ERXIT ; BR IF SO.

        CLR -(SP)
        MOVB $ITEMB,(SP) ; PUSH ERROR #.
        TYPE ,EH1 ,TYPOS ,3!WLZ ; PRINT ERROR NUMBER
        MOVB $TSTNM,-(SP) ; PUSH TEST #
        TYPE ,EH2 ,TYPOS ,2!WLZ ; PRINT TEST NUMBER
        MOV $ERRPC,-(SP) ; PUSH PC
        TYPE ,EH3 ,TYPOC ; PRINT PC.
        TYPE , $CRLF
        TST EMPRE
        BEQ .+6
        TYPE
EMPRES: 0 ; PRINT ERROR PREFIX (IF ANY)...
        TYPE
EMADR: 0 ; ...THE BASIC ERROR MESSAGE...
        CALL @ (PC)+
ESADR: 0 ; ...AND THE FORMATTED SIGNATURE.
        TYPE , $CRLF

ERXIT: MOV (SP)+,RO ; RESTORE...
        RTS PC ; ...AND RETURN TO $ERROR.

```

E13

```

5694
5695
5696
5697
5698
5699
5700 045016 032777 040000 134114 ESUMRY: BIT #BIT14,BSWR
5701 045024 001051 BNE 4$ ; DON'T IF SUPPRESSED...
5702 045026 005737 001112 TST $ERTTL
5703 045032 001446 BEQ 4$ ; ...OR IF THERE AREN'T ANY ERRORS.
5704
5705 045034 104401 045152 TYPE ,ESUM1 ; PRINT HEADING.
5706 045040 012700 051034 MOV #ELOG,R0
5707 045044 012701 000225 MOV #ELOGN,R1
5708 045050 005710 1$: TST (R0) ; ERROR N HAPPEN ??
5709 045052 001434 BEQ 3$ ; BR IF NOT
5710 045054 011046 MOV (R0),-(SP) ; PUSH NUMBER OF HAPPENINGS.
5711 045056 010002 MOV R0,R2 ; ERROR LOC...
5712 045060 162702 051034 SUB #ELOG,R2 ;... - BASE ...
5713 045064 006202 ASR R2 ;... / 2 = ERROR NUMBER.
5714 045066 052702 104000 BIS #ERROR,R2 ; MAKE <EMT+N>.
5715 045072 012703 002456 MOV #TRAPE,R3
5716 045076 022302 2$: CMP (R3)+,R2 ; FIND THE ERROR CALL...
5717 045100 001376 BNE 2$
5718 045102 005743 TST -(R3) ;... 2 = ERROR PC.
5719 045104 010346 MOV R3, -(SP) ; PUSH ERROR PC.
5720 045106 042702 104000 BIC #ERROR,R2
5721 045112 010246 MOV R2, -(SP) ; PUSH ERROR NUMBER.
5722
5723 045114 104401 045203 104403 TYPE ,TAB09 ,TYPOS ,3!WLZ ; PRINT ERROR NUMBER...
5724 045124 104401 046131 104402 TYPE ,SPAC4 ,TYPOC ;...ERROR PC...
5725 045132 104401 046131 104405 TYPE ,SPAC4 ,TYPDS ;...AND NUMBER OF HAPPENINGS.
5726 045140 104401 001163 TYPE ,CRLF
5727 045144 005720 3$: TST (R0)+
5728 045146 077140 SOB R1,1$
5729 045150 000207 4$: RETURN
5730
5731 045152 200 011 ESUM1: .ASCII <CRLF><11>
5732 045154 105 122 122 .ASCIZ /ERROR PC TIMES/<CRLF>
5733 ; NNN PPPPPP N
5734 045203 011 040 000 TAB09: .ASCIZ <11><40>
5735 .EVEN

```



```

5737
5738
5739
5740 045206
5741 045206 000207
5742
5743 045210 104401 001163
5744 045214 004737 045230
5745 045220 000442
5746 045222 162737 000002 001120
5747 045230 013746 001120
5748 045234 104401 045713 104402
5749 045242 000207
5750
5751 045244 104401 001163
5752 045250 004737 045256
5753 045254 000424
5754 045256 004737 037436
5755 045262 013746 037540
5756 045266 013746 037536
5757 045272 005766 000002
5758 045276 100003
5759 045300 062766 100000 000002
5760 045306 006116
5761 045310 104401 045704
5762 045314 104403 000403
5763 045320 104403 000405
5764 045324 000207
5765 045326 013746 001126
5766 045332 013746 001124
5767 045336 104401 001163
5768 045342 104401 045725 104402
5769 045350 104401 045735 104402
5770 045356 000207
5771 045360
5772 045360 013746 001126
5773 045364 013746 001122
5774 045370 013746 001124
5775 045374 013746 001120
5776 045400 104401 001163
5777 045404 104401 045747 104402
5778 045412 104401 046136 104402
5779 045420 104401 001163
5780 045424 104401 045767 104402
5781 045432 104401 046136 104402
5782 045440 000207
5783
5784 045442 013746 001120
5785 045446 013746 001122
5786 045452 104403 000403
5787 045456 104401 045006 104402
5788 045464 000207
5789
5790 045466 013746 001126
5791 045472 013746 001124
5792 045476 104401 001163
5793 045502 104401 046007 104405

```

; SUBROUTINES TO ASSEMBLE AND PRINT VARIOUS ERROR SIGNATURES.
;
EF1:
NOSIG: RETURN ; SOME HAVE NO SIGNATURE AT ALL.
;
EF2: TYPE , \$CRLF ; A 16 BIT ADDRESS WITH GOOD/BAD DATA.
CALL LA16
BR EF4
LA16M2: SUB #2, \$GDADR ; FOR POST INCR ADDRESSES.
LA16: MOV \$GDADR, -(SP) ; A 16 BIT (LOCAL) ADDRESS.
TYPE , ADDR , TYPOC
RETURN
;
EF3: TYPE , \$CRLF ; A 22 BIT ADDRESS WITH GOOD/BAD DATA.
CALL GA22
BR EF4
GA22: CALL VA2GA ; CONVERT FROM A 16 BIT VIRTUAL (R1).
MOV GALO, (SP)
MOV GAHI, -(SP)
TST 2(SP) ; ADDR<15> SET ??
BPL 1\$; SKIP IF NOT, "C" = 0...
ADD #BIT15, 2(SP) ; ...ELSE CLEAR IT, "C" = 1.
1\$: ROL (SP)
TYPE , GADDRS
TYPOS , 3!WLZ ; PRINT ADDR<21:15>
TYPOS , 5!WLZ ; PRINT ADDR<14:00>
RETURN
EF4: MOV \$BDDAT, -(SP) ; GOOD/BAD DATA.
MOV \$GDDAT, -(SP)
TYPE , \$CRLF
TYPE , EXPD , TYPOC
TYPE , RECD , TYPOC
RETURN
EF5: ; UNUSED.
EF6: MOV \$BDDAT, -(SP) ; SRC-ADDR-DATA/DST-ADDR-DATA.
MOV \$BDADR, -(SP)
MOV \$GDDAT, -(SP)
MOV \$GDADR, -(SP)
TYPE , \$CRLF
TYPE , SADR , TYPOC ; SOURCE ADDRESS...
TYPE , COMMA , TYPOC ; ...AND DATA.
TYPE , \$CRLF
TYPE , DADR , TYPOC ; DEST ADDRESS...
TYPE , COMMA , TYPOC ; ...AND DATA.
RETURN
EF7: MOV \$GDADR, -(SP) ; BAD TRAP -- PC...
MOV \$BDADR, -(SP) ; ...AND VECTOR.
TYPOS , 3!WLZ ; PRINT VECTOR...
TYPE , EH3 , TYPOC ; ...AND PC AT TIME OF TRAP.
RETURN
EF8: MOV \$BDDAT, -(SP) ; INTERRUPT SEQUENCE INCOMPLETE.
MOV \$GDDAT, -(SP)
TYPE , \$CRLF
TYPE , IEXPD , TYPOS ; EXPECTED THIS MANY...

```

5794 045510 104401 045735 104405 TYPE ,RECD ,TYPOS ;...GOT THIS MANY.
5795 045516 000207 RETURN
5796
5797 045520 013746 001122 EF9: MOV $BDADR, (SP) ; INTERRUPT SEQUENCE (VECTOR) ERROR.
5798 045524 013746 001120 MOV $GDADR, -(SP)
5799 045530 104401 001163 TYPE , $CRLF
5800 045534 104401 046032 104403 TYPE ,VEXPD ,TYPOS ,3!WLZ ; EXPECTED VECTOR...
5801 045544 104401 045735 104403 TYPE ,RECD ,TYPOS ,3!WLZ ;...RECEIVED VECTOR.
5802 045554 000207 RETURN
5803
5804 045556 013746 001126 EF10: MOV $BDDAT, -(SP) ; 7201 OR 8036 BYTES RECEIVED...
5805 045562 013746 001124 MOV $GDDAT, (SP) ;...AND TRANSMITTED.
5806 045566 104401 001163 TYPE , $CRLF
5807 045572 104401 046051 104405 TYPE ,BXMTD ,TYPDS ; SENT THIS MANY...
5808 045600 104401 045735 104405 TYPE ,RECD ,TYPDS ;...AND GOT THESE BACK.
5809 045606 000207 RETURN
5810 045610 013746 001132 EF11: MOV $BDDAT+4, -(SP) ; 7201 EXTERNAL OR SPECIAL STATUS.
5811 045614 013746 001130 MOV $BDDAT+2, -(SP)
5812 045620 104401 001163 TYPE , $CRLF
5813 045624 104401 046067 104403 TYPE ,OSRO ,TYPOS ,3!WLZ ; OBSERVED SRO...
5814 045634 104401 046075 104403 TYPE ,OSR1 ,TYPOS ,3!WLZ ;...AND SR1.
5815 045644 000207 RETURN
5816
5817 045646 013746 001130 EF12: MOV $BDDAT+2, -(SP) ; 7201 MODEM STATUS SRO(B)...
5818 045652 013746 001126 MOV $BDDAT, -(SP) ;...AND SRO(A).
5819 045656 104401 001163 TYPE , $CRLF
5820 045662 104401 046105 104403 TYPE ,OSROA ,TYPOS ,3!WLZ ; OBSERVED SRO(A)...
5821 045672 104401 046116 104403 TYPE ,OSROB ,TYPOS ,3!WLZ ;...AND SRO(B).
5822 045702 000207 RETURN
5823
;
5824 ; BITS AND PIECES USED BY THE FORMATTERS.
5825 ;
5826 045704 107 114 117 GADDRS: .ASCII /GLOBAL /
5827 045713 101 104 104 ADDR: .ASCIZ /ADDRESS: /
5828 045725 105 130 120 EXPD: .ASCIZ /EXP'D: /
5829 045735 040 040 122 RECD: .ASCIZ / REC'D: /
5830 045747 123 122 103 SADR: .ASCIZ /SRC ADDR,DATA: /
5831 045767 104 123 124 DADR: .ASCIZ /DST ADDR,DATA: /
5832 046007 111 116 124 IEXPD: .ASCIZ /INTERRUPTS EXP'D: /
5833 046032 126 105 103 VEXPD: .ASCIZ /VECTOR EXP'D: /
5834 046051 102 131 124 BXMTD: .ASCIZ /BYTES XMT'D: /
5835 046067 123 122 060 OSRO: .ASCIZ /SRO: /
5836 046075 040 040 123 OSR1: .ASCIZ / SR1: /
5837 046105 123 122 060 OSROA: .ASCIZ /SRO(A): /
5838 046116 040 040 123 OSROB: .ASCIZ / SRO(B): /
5839 046131 040 040 SPAC4: .ASCII / /
5840 046133 040 SPAC2: .ASCII / /
5841 046134 040 000 SPACE: .ASCIZ / /
5842 046136 054 000 COMMA: .ASCIZ /,/

```

```

5844
5845 ; BASIC ERROR MESSAGES.
5846 ;
5847 046140 104 103 112 DCJDM: .ASCIZ /DCJ11 BASE INSTRUCTION TESTS/
5848 046175 123 105 114 STF: .ASCIZ /SELF-TEST FAILURE/
5849 ;
5850 046217 125 116 105 UNXT: .ASCIZ /UNEXPECTED TRAP THRU VECTOR /
5851 046254 102 125 123 CSRE: .ASCIZ /BUS TIME-OUT AT KXCSR /
5852 ;
5853 ;
5854 046303 102 125 123 CLK1: .ASCIZ /BUS TIME-OUT AT LINE CLOCK /
5855 046337 103 114 117 CLK2: .ASCIZ /CLOCK INTERRUPT NOT MASKED AT LEVEL 6/
5856 046405 103 114 117 CLK3: .ASCIZ /CLOCK INTERRUPT NOT RECEIVED/
5857 046442 103 101 116 CLK4: .ASCIZ /CAN'T DISABLE CLOCK INTERRUPT/
5858 ;
5859 046500 102 125 123 DPR1: .ASCIZ /BUS TIME-OUT AT LOCAL DPR /
5860 046533 104 120 122 DPR2: .ASCIZ /DPR LOCAL WRITE-READ ERROR/
5861 046566 124 111 115 DPR3: .ASCIZ /TIME-OUT (EOP) ON Q-BUS READ AT DPR /
5862 046633 116 117 116 DPR4: .ASCIZ /NON-ZERO DATA RETURNED ON Q-BUS READ/
5863 046700 103 117 115 DPR5: .ASCIZ /COMMAND INTERRUPT (DPRO) DIDN'T HAPPEN (OR DOESN'T WORK)/
5864 046771 116 117 040 DPR6: .ASCII /NO /
5865 046774 124 111 115 DPR7: .ASCIZ /TIME-OUT (EOP) ON Q-BUS WRITE TO DPR /
5866 047042 111 116 124 DPR8: .ASCIZ /INTERRUPT RECEIVED WITH DPR DISABLED/
5867 .ODD
5868 047107 104 120 122 DPR9: .ASCIZ /DPRXX INTERRUPT NOT RECEIVED/
5869 047144 101 122 102 DPR10: .ASCIZ /ARBITER WRITE TO DPR (WORD 15) FAILS/
5870 ;
5871 ;CONSOLE SERIAL LINE ERROR MESSAGES
5872 ;
5873 047211 130 115 124 SLIX: .ASCIZ /XMTR /
5874 047217 122 103 126 SLIR: .ASCIZ /RCVR /
5875 047225 111 116 124 SLI2: .ASCIZ /INTERRUPT NOT MASKED AT LEVEL 4/
5876 047265 111 116 124 SLI3: .ASCIZ /INTERRUPT NOT RECEIVED/
5877 047314 122 105 121 SLI4: .ASCIZ /REQUEST STILL ACTIVE AFTER "IACK"/
5878 047356 122 105 103 SLI5: .ASCIZ /RECEIVED DATA INCORRECT/
5879 ;
5880 047406 102 125 123 NEC1: .ASCIZ \BUS TIME-OUT AT SYNC/ASYN (OR I8254) \
5881 047455 070 060 060 NEC2: .ASCIZ /800HZ INTERRUPT (8254 TIMER 2) NOT RECEIVED/
5882 047531 103 110 101 NECCM: .ASCIZ /CHAN X (ASYN) /
5883 047536 NECC= NECCM+5
5884 047541 NECM= NECCM+8.
5885 047551 103 110 101 NECDM: .ASCIZ /CHAN A (SYNC-DMA) /
5886 047574 104 101 124 NEC3: .ASCIZ /DATA TRANSFER INCOMPLETE/
5887 047625 074 105 117 NEC3A: .ASCIZ /<EOF-SDLC> NOT RECEIVED/
5888 047356 NEC4= SLI5 ; RECEIVED DATA INCORRECT.
5889 047655 125 116 105 NEC5: .ASCIZ \UNEXPECTED EXTERNAL/ERROR INTERRUPT\
5890 047721 115 117 104 NECCM: .ASCIZ /MODEM CONTROL ERROR/<CRLF>
5891 047746 123 124 101 NEC6: .ASCIZ /STATUS INTERRUPT NOT RECEIVED/
5892 050004 123 124 101 NEC7: .ASCIZ /STATUS WRONG WITH <RTS> SET/
5893 050040 123 124 101 NEC8: .ASCIZ \STATUS WRONG WITH <TT108/2> IN CSRA SET\
5894 050110 123 124 101 NEC9: .ASCIZ /STATUS WRONG WITH <TERM-IN-SERV> IN CSRA SET/
5895 050165 123 124 101 NEC10: .ASCIZ /STATUS WRONG WITH ALL CONTROLS CLEAR/
5896 ;
5897 050232 102 125 123 PIO1: .ASCIZ /BUS TIME-OUT AT PIO /
5898 050257 120 111 117 PIO2: .ASCIZ /PIO RESET STATE INCORRECT/
5899 050311 120 111 117 PITN: .ASCIZ /PIO TIMER N /
5900 050323 PITX= .-3
5901 050326 116 117 124 PIO3: .ASCIZ /NOT RUNNING/

```

113

KXJ11-CA FUNCTIONAL TEST
ERROR HANDLER ROUTINE

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 163

5902	050342	116	105	126	PI04:	.ASCIZ	/NEVER STOPS/
5903	050356	111	116	124	PI05:	.ASCIZ	/INTERRUPT NOT MASKED AT LEVEL 4/
5904	050416	111	116	124	PI06:	.ASCIZ	/INTERRUPT NOT RECEIVED/
5905	050445	120	111	117	PORTBA:	.ASCIZ	/PIO PORT B=>A /
5906	050464	120	111	117	PORTAB:	.ASCIZ	/PIO PORT A=>B /
5907	050503	104	101	124	PI07:	.ASCIZ	/DATA TRANSFER INCOMPLETE/
5908	047356				PI08=	SLI5	; DATA INCORRECT.
5910						.EVEN	

5911

```

.SBTTL BINARY TO ASCII DECIMAL ENCODER
;*****
; CONVERT BINARY TO SIX CHARACTER ASCII DECIMAL STRING.
; RANGE: -32768. TO +32767. (SIGNED) OR 0 TO 65535. (UNSIGNED).
;
; CALL: MOV NUM,RO      ; OR SDECIMAL (SIGNED).
;       CALL DECIMAL   ; RO POINTS TO STRING ADDRESS ON RETURN.
;       ...
;
DECIMAL: MOV #177,-(SP) ; UNSIGNED, PUSH A NULL.
          402
SDECIMAL: MOV #'',(SP) ; SIGNED, PUSH A + SIGN.
          MOV #177,(PC)+ ; NULL LEAD 0'S.
LZS:      0
          MOV (SP),8$ ; SET 1ST CHAR (NULL OR +).
          MOV RO,(SP) ; COPY NUMBER TO THE STACK...
          MOV #8$,RO ; ...AND USE RO FOR THE STRING POINTER.
          CMPB (RO),#177 ; CHECK SIGN BYTE...
          BEQ 1$ ; ...AND BR IF UNSIGNED.
          TST (SP) ; IS NUMBER POS ??
          BPL 1$ ; YUP.
          NEG (SP) ; NOPE, MAKE ABS VALUE...
          MOVB #'-',(RO) ; ...AND CHANGE THE SIGN TO (-).
          JSR R5,2$ ; CONVERT BINARY TO ASCII
          10000. ; 10+4
          1000. ; 10+3
          100. ; 10+2
          10. ; 10+1
          1. ; 10+0
          TST (SP)+ ; FIX STACK...
          MOV #8$,RO ; ...RETURN POINTER TO CALLER.
          RETURN

2$:      INC RO ; POINT TO NEXT DIGIT...
          MOVB #'0,(RO) ; ...AND SET IT TO 0.
3$:      CMP 2(SP),(R5)
          BLO 4$
          INCB (RO) ; ACCUMULATE THIS DIGIT.
          SUB (R5),2(SP)
          BR 3$
4$:      CMP (R5)+,#1 ; LAST DIGIT ??
          BEQ 6$ ; ALL DONE IF SO.
          TST LZS ; NO, STILL NULLING ??
          BEQ 2$ ; LOOP IF NOT.
          CMPB (RO),#'0 ; YES, GOT A 0 ??
          BNE 5$ ; BR IF NOT (1ST NZ FOUND).
          MOVB LZS,(RO) ; YES, REPLACE IT...
          402 ; ...AND SKIP NEXT.
5$:      CLR LZS ; 1ST NON-ZERO, CLEAR LZ SWITCH.
          BR 2$ ; LOOP FOR 5 DIGITS.
6$:      RTS R5 ; RETURN.

8$:      .BLKB 6 ; STRING BUFFER...
          .WORD 0 ; ...PROPERLY TERMINATED.

```

```

050534 012746 000177
050540 000402
050542 012746 000053
050546 012727 000177
050552 000000
050554 011637 050720
050560 010016
050562 012700 050720
050566 121027 000177
050572 001405
050574 005716
050576 100003
050600 005416
050602 112710 000055
050606 004537 050634
050612 023420
050614 001750
050616 000144
050620 000012
050622 000001
050624 005726
050626 012700 050720
050632 000207

050634 005200
050636 112710 000060
050642 026615 000002
050646 103404
050650 105210
050652 161566 000002
050656 000771
050660 022527 000001
050664 001414
050666 005737 050552
050672 001760
050674 121027 000060
050700 001003
050702 113710 050552
050706 000402
050710 005037 050552
050714 000747
050716 000205

050720
050726 000000

```

```

5913 ;THE FOLLOWING ARE ARBITER PARAMETERS NEEDED BY THE K2
5914 050730 000000 .MSIZE: .WORD 0 ;; TOTAL MEMORY SIZE (K WORDS).
5915 050732 000000 .LSTPG: .WORD 0 ;; PAGE ADDRESS (PAF) OF LAST 1K PAGE...
5916 ;;...OR ZERO IF MMU NOT AVAILABLE.
5917 050732 .MMU= .LSTPG ;; MMU AVAILABLE FLAG.
5918 050734 000000 .LSTAD: .WORD 0 ;; LAST VIRTUAL ADDRESS IN LAST PAGE...
5919 ;;...OR LAST ADDRESS UNDER 28K (IF NO MMU).
5920 050736 000020 .ABUSW: .WORD 16. ;; ADDRESS BUS WIDTH, 16, 18, OR 22.
5921 050740 000000 IOP.UT: .WORD 0 ;; UNIT UNDER TEST FOR MULTIPLE UNITS
5922 050742 000000 IOP.ID: .WORD 0 ;; IOP ID NUMBER
5923 050744 000000 KG.IOP: .WORD 0 ;; 1 - KNOWN GOOD IOP PRESENT
5924 050746 000000 IOPCNT: .WORD 0 ;; NUMBER OF IOPS INDICATED BY ARBITER SNIFF ROUTINE
5926 050750 000000 $HILIM: .WORD 0 ;
5927 050752 000000 LOWLIM: .WORD 0 ;; LOWEST QBUS ADDRESS THAT SHARED MEMORY CAN START AT
5928 050754 000000 MAXBLK: .WORD 0 ;; MAXIMUM NUMBER OF 8KB BLOCKS OF SHARED MEMORY THAT
5929 ; CAN RESIDE ON THE QBUS
5930 050756 177777 QBE22: .WORD -1 ;; ONE INDICATES THAT A 22 BIT QBUS EXERCISER IS
5931 ; RESIDENT IN THE SYSTEM AT ADDRESS 170020
5932 050760 177777 QBE18: .WORD 1 ;; ONE INDICATES THAT AN 18 BIT QBUS EXERCISER IS
5933 ; RESIDENT IN THE SYSTEM AT ADDRESS 170000
5934 050762 000000 KDJ$: .WORD 0 ;
5935 050764 000000 J11$: .WORD 0 ;
5936 050766 000000 F11$: .WORD 0 ; \
5937 050770 000000 T11$: .WORD 0 ; >NON ZERO IN ONE OF THESE INDICATES ARBITER CPU TYPE
5938 050772 000000 L11$: .WORD 0 ; /
5939 050774 $PATCH: .BLKW 20 ; AREA FOR PATCHES

```

L13

5942
5943
5944
5945
5946
5947
5948
5949 000225
5950 051034
5951 051506 051506
5952

```
;  
; EVERY THING BEFORE THIS IS LOADED INTO THE TARGET K2  
; NOW ALL THE REST IS THE ARBITERS REMOTE LOADER AND VT    NONE OF WHICH  
; NEED BE COPIED DOWN TO THE SLAVE UNIT.  
;  
; DEFINE A SPACE FOR THE "ERROR LOG".  
;  
ELOGN= $EN+1  
ELOG:  .BLKW   ELOGN   ; ERROR LOG (1WORD/ERROR).  
LASTAD: .       ; LAST LOAD IMAGE ADDRESS.  
;
```

```

5954 ;*****
5955 ;THIS STUFF RUNS FIRST TO ESTABLISH WHAT TYPE OF ARBITER CPU IS
5956 ;PRESENT AND HOW MUCH MEMORY IS AVAILABLE (IMPORTANT FOR SHARED
5957 ;MEMORY TESTS).
5958 ;THE FOLLOWING STUFF OVERLAYS THE BUFFER AREA.
5959 ;IF THE TEST IS RUN IN THE ARBITER THE FOLLOWING
5960 ;STUFF MAY BE OVER WRITTEN
5961 ;*****
5962 051510 012706 001100 $$ARB: MOV #1100,R6 ;SETUP THE STACK
5963 051514 012737 000176 001140 MOV #176,@#SWR ;POINTER TO SOFTWARE SWITCH REGISTER
5964 051522 012737 000176 001142 MOV #176,@#SWR+2 ;AND DISPLAY REGISTER
5965 051530 032777 004000 127402 BIT #BIT11,@SWR ;STANDALONE TEST?
5966 051536 001402 BEQ 10$ ;NOT DOING STAND ALONE IOP; GO TO
5967 ;ARBITER SETUP ROUTINES
5968 051540 000137 056702 JMP REMOTE ;DOING STANDALONE TEST; SKIP OVER THE
5969 ;ARBITER STUFF
5970 051544 012737 000340 177776 10$: MOV #340,@#PS ;PSW - SET KERNEL MODE
5971 051552 004737 052160 JSR PC,ARBTR ;FIND OUT WHAT TYPE OF ARBITER WE ARE
5972 051556 004737 052574 JSR PC,.SIZE ;FIND OUT HOW MANY KWORDS OF ARBITER
5973 ;MEMORY THERE ARE
5974 051562 004737 053412 JSR PC,SNIFFR ;FIND OUT HOW MANY AND WHAT TYPE OF IOPS
5975 ;ARE IN THE SYSTEM
5976 051566 005737 001244 TST $DEVM ;IS THERE AT LEAST ONE IOP?
5977 051572 001003 BNE 1$ ;
5978 051574 104401 052006 TYPE ,NORESP ;INFORM OPERATOR THAT NO IOPS WERE FOUND
5979 051600 000000 HALT
5980 051602 004737 054070 1$: JSR PC,QBEX ;ARE THERE Q BUS EXERCISERS?
5981 051606 004737 054202 JSR PC,SHRMEM ;GET INFORMATION FOR SHARED MEMORY TESTS
5982 ;SUCH AS AMOUNT OF EXISTING MEMORY AND THE
5983 ;NUMBER OF ADDRESS BITS SUPPORTED
5984 051612 004737 055310 JSR PC,SHOSYS ;NOW SHOW WHAT WHAT IOPS ARE PRESENT
5985 ;-----
5986 ;NOW SYNC UP THE IOP UNIT BEING TESTED WITH THE IOP ID NUMBER WHICH
5987 ;IS DETERMINED BY THE SETTING OF THE ID SWITCH ON THE KXJ11 MODULE
5988
5989 051616 $AGAIN: ;COMES HERE THE FIRST TIME THROUGH AND AFTER IT HAS TESTED ALL
5990 ;THE IOPS IN THE SYSTEM.
5991
5992 051616 012737 000001 050740 MOV #1,IOP.UT ;IOP UNDER TEST
5993 051624 012704 001256 MOV #DDW2,R4 ;POINT TO FIRST IOP TABLE ENTRY SET UP BY SNIFFR
5994 051630 012737 000002 050742 MOV #2,IOP.ID ;FIRST IOP ID NUMBER; 0 AND 1 ARE STANDALONE
5995 051636 005714 TRYNXT: TST (R4) ;IS THERE AN IOP HERE?
5996 051640 001531 BEQ NXT ;GO TRY THE NEXT POSITION IF NOT.
5997
5998 ;YUP
5999 051642 013702 050740 MOV IOP.UT,R2
6000 051646 006302 R2 ;MOVE IT LEFT ONE POSITION
6001 051650 016237 056732 056730 MOV SLUADR(R2),SLU2 ;GET THE CORRECT SLU
6002 051656 012737 160000 052002 MOV #160000,IOPADR ;GET BASE ADDRESS OF IOPS
6003 051664 020427 001272 CMP R4,#DDW8 ;IS THE IOP ID NUMBER > 7
6004 051670 002403 BLT 1$
6005 051672 062737 015000 052002 ADD #15000,IOPADR ;YES, SHIFT TO UPPER RANGE
6006 051700 013701 050742 1$: MOV IOP.ID,R1 ;GET IOP ID NUMBER
6007 051704 072127 000005 ASH #5,R1 ;SHIFT IT OVER
6008 051710 060137 052002 ADD R1,IOPADR ;
6009 051714 032714 000002 BIT #BIT01,(R4) ;IS IOP IN HIGH RANGE OR LOW RANGE?
6010 051720 001403 BEQ 2$ ; 1=LOW 2=HI

```



```

6011 051722 052737 002000 052002      BIS      #2000,IOPADR      ;
6012 051730 010437 052004      2$:     MOV      R4,POINT  ;SAVE R4 IN LOCATION POINT
6013 051734 005737 052000      TST      FRESH          ;IS THIS A NEW LOAD OF THE DIAGNOSTIC?
6014 051740 001011                BNE      3$             ;BRANCH IF NOT
6015                                ;
6016                                ;ELSE LOAD THE DIAGNOSTIC INTO THE TARGET IOP
6017                                ;
6018 051742 013746 050742      MOV      IOP.ID,-(SP)
6019 051746 104401 052136 104405      TYPE   .LDING,TYPDS
6020 051754 104401 001163      TYPE   .$CRLF
6021 051760 000137 056702      JMP     REMOTE
6022                                ;
6023                                ;IF THE DIAGNOSTIC HAS ALREADY BEEN LOADED INTO ALL IOPS THEN JUST RESTART
6024                                ;
6025 051764 013700 052002      3$:     MOV      IOPADR,R0      ;GET ADDRESS OF IOP TO RESUME TESTING
6026 051770 005060 000030      CLR     30(R0)         ;WRITE TO TPR12 TO KICK IT OFF
6027 051774 000137 056770      JMP     LINK           ;LINK UP WITH THE IOP SERIAL LINE
6028                                ;
6029 052000 000000                FRESH:  .WORD   0         ;ZERO INDICATES THAT ALL IOPS ARE NOT LOADED
6030 052002 000000                IOPADR: .WORD   0         ;HOLD VIRTUAL ADDRESS OF IOP BEING TESTED
6031 052004 000000                POINT:  .WORD   0         ;HOLDS DEVICE DESCRIPTOR WORD ADDRSS
6032                                ;
6033 052006      200      116      117 NORESP: .ASCIZ <CRLF>/NO IOPS INSTALLED/
6034                                .EVEN
6035 052032 012706 001100      NEXT1: MOV     #1100,SP      ;SET UP THE STACK
6036 052036 005037 001166      CLR     $MAIL          ;
6037 052042 005037 001170      CLR     $MAIL+2        ; CLEAN UP APT MAIL BOX
6038 052046 005037 001172      CLR     $MAIL+4        ;
6039 052052 005037 001102      CLR     $TSTNM         ;ZERO OUT THE TEST NUMBER
6040 052056 005237 050740      INC     IOP.UT          ;IOP UNDER TEST (NOT ID NUMBER)
6041 052062 023737 050740 050746      CMP     IOP.UT,IOPCNT  ;IS IOP UT / NUMBER OF IOPS IN SYSTEM?
6042 052070 003005                BGT     2$             ;
6043 052072 013704 052004      MOV     POINT,R4       ;RETRIEVE VALUE OF DEVICE DESCRIPTOR WORD
6044 052076 020427 001310      CMP     R4,#$DDW15     ; DID WE HIT THE TOP?
6045 052102 001010                BNE     NXT            ;
6046 052104 005237 052000      2$:     INC     FRESH     ;INDICATE THAT ITS NOT A NEW LOAD
6047 052110 005737 052000      TST     FRESH          ;DON'T LET IT INCREMENT TO ZERO
6048 052114 001002                BNE     1$             ;
6049 052116 005237 052000      INC     FRESH          ;
6050 052122 000635                1$:     BR      $AGAIN
6051 052124 062704 000002      NXT:   ADD     #2,R4     ;POINT TO NEXT POSSIBLE IOP
6052 052130 005237 050742      INC     IOP.ID         ;NEXT IOP ID NUMBER
6053 052134 000640                BR      TRYNEXT
6054 052136      200      114      117 LDING:  .ASCIZ <CRLF>/LOADING IOP ID# /
6055                                .EVEN

```

```

6057 052160          ARBITR: ;ROUTINE TO FIND OUT WHAT KIND OF ARBITER CPU IS IN THE SYSTEM
6058 052160 104401 052450      TYPE      ,ARBTYP
6059 052164 013746 000004      MOV      @#4,-(SP)      ;PUSH OLD VECTOR ONTO STACK
6060 052170 013746 000006      MOV      @#6,-(SP)      ;
6061 052174 012737 052250 000004      MOV      #5,@#4      ;LOAD NEW HANDLER ADDRESS
6062 052202 012737 000340 000006      MOV      #PR7,@#6      ;
6063 052210 013700 177750      MOV      @#177750,R0      ;MOVE MAINTENANCE REGISTER TO R0
6064          ;NON-KDJ11 WILL TIMEOUT ON ACCESS TO MAINT. REG
6065 052214 042700 177417      BIC      #177417,R0      ;CLEAR THE NON-CPU ID BITS
6066 052220 022700 000020      CMP      #20,R0      ;IS THE ARBITER CPU A KDJ11?
6067 052224 001014          BNE      10$      ;IF ITS NOT A KDJ BRANCH
6068          ; ELSE
6069 052226 052737 000015 177746      BIS      #15,@#CCR      ; FORCE CACHE MISSES
6070 052234 104401 052477      TYPE      ,KDJ11A
6071 052240 012737 000001 050762      MOV      #1,@#KDJ$      ;MOVE A ONE TO KDJ TO INDICATE THE TYPE OF ARBITER
6072 052246 000403          BR      10$      ;
6073 052250 005037 050762      5$: CLR      @#KDJ$      ; NOT A KDJ11
6074 052254 022626          CMP      (SP)+,(SP)+      ;RESTORE STACK
6075 052256 012637 000006      10$: MOV      (SP)+,@#6      ;RESTORE VECTORS
6076 052262 012637 000004      MOV      (SP)+,@#4      ;
6077 052266 005737 050762      TST      @#KDJ$      ;IF THE ARBITER CPU IS A KDJ11
6078 052272 001402          BEQ      7$      ;
6079 052274 000137 052446          JMP      25$      ; THEN SKIP THE REST OF THIS
6080 052300 013746 000010      7$: MOV      @#10,-(SP)      ;SAVE CONTENTS OF VECTOR
6081 052304 013746 000012      MOV      @#12,-(SP)      ;
6082 052310 012737 052422 000010      MOV      #15,@#10      ;LOAD IN NEW VECTOR
6083 052316 012737 000340 000012      MOV      #PR7,@#12      ;
6084 052324 000007          MFPT          ;LSI 11 WILL EXECUTE A RESERVED INSTRUCTION TRAP
6085 052326 022700 000005      CMP      #J11,R0      ;IS IT A DCJ11 BASED CPU OTHER THAN KDJ11?
6086 052332 001411          BEQ      11$      ; BRANCH IF YES
6087 052334 022700 000004      CMP      #T11,R0      ;IS IT A DCT11 BASED CPU?
6088 052340 001414          BEQ      12$      ; BRANCH IF YES
6089 052342 022700 000003      CMP      #F11,R0      ;IS IT A DCF11 BASED CPU?
6090 052346 001417          BEQ      13$      ; BRANCH IF YES
6091 052350 104401 052556          TYPE      ,UNKNWN      ;ARBITER CPU TYPE UNKNOWN
6092 052354 000430          BR      20$      ;
6093 052356 012737 000001 050764 11$: MOV      #1,@#J11$      ;
6094 052364 104401 052510          TYPE      ,DCJ11
6095 052370 000422          BR      20$      ;
6096 052372 012737 000001 050770 12$: MOV      #1,@#T11$      ;INDICATE T11 BASED CPU
6097 052400 104401 052534          TYPE      ,DCT11
6098 052404 000414          BR      20$      ;
6099 052406 012737 000001 050766 13$: MOV      #1,@#F11$      ;INDICATE AN F11 BASED CPU
6100 052414 104401 052524          TYPE      ,DCF11
6101 052420 000406          BR      20$      ;
6102 052422          ;GOT HERE VIA A RESERVED INSTRUCTION TRAP WHICH INDICATES AN LSI11 CPU
6103 052422 012737 000001 050772 15$: MOV      #1,@#L11$      ;INDICATE AN LSI11 PROCESSOR
6104 052430 104401 052545          TYPE      ,LSI11
6105 052434 022626          CMP      (SP)+,(SP)+      ;RESTORE STACK
6106 052436 012637 000012      20$: MOV      (SP)+,@#12      ;RESTORE VECTORS
6107 052442 012637 000010      MOV      (SP)+,@#10      ;
6108 052446 000207      25$: RTS      PC
6109          ;MESSAGES
6110 052450          015      012      101      ARBTYP: .ASCIZ <CR><LF>/ARBITER CPU TYPE IS /
6111 052477          101      040      113      KDJ11A: .ASCIZ /A KDJ11A/
6112 052510          104      103      112      DCJ11: .ASCIZ /DCJ11 BASED/
6113 052524          101      040      113      DCF11: .ASCIZ /A DCF11/

```

C14

KXJ11-CA FUNCTIONAL TEST
BINARY TO ASCII DECIMAL ENCODER

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 170

6114	052534	101	116	040	DCT11:	.ASCIZ	/AN SBC11/
6115	052545	101	116	040	LSI11:	.ASCIZ	/AN LSI11/
6116	052556	125	116	104	UNKN11:	.ASCIZ	/UNDETERMINED/
6117						.EVEN	

```

6119
6120
6121
6122
6123
6124
6125
6126
6127 052574 005037 177572
6128 052600 012737 000020 050736
6129 052606 013746 000004
6130 052612 013746 000006
6131 052616 013746 000114
6132 052622 013746 000116
6133 052626 010605
6134
6135 052630 012737 000116 000114
6136 052636 012737 000002 000116
6137 052644 012737 052672 000004
6138 052652 012737 000340 000006
6139 052660 005737 177572
6140 052664 000240
6141 052666 000240
6142 052670 000416
6143
6144 052672 012737 052716 000004 1$:
6145 052700 005002
6146 052702 005003
6147 052704 005712 2$:
6148 052706 062702 004000
6149 052712 005203
6150 052714 000773
6151 052716 162702 000002 3$:
6152 052722 005001
6153 052724 000514
6154
6155 052726 012701 172340
6156 052732 012702 172300
6157 052736 012703 000010
6158 052742 005000
6159 052744 010021
6160 052746 012722 077406 1$:
6161 052752 062700 000200
6162 052756 077306
6163 052760 012741 177600
6164 052764 005041
6165 052766 005003
6166
6167 052770 012737 053046 000004
6168 052776 005737 172516
6169 053002 012737 000020 172516
6170 053010 012737 053064 000004
6171 053016 052737 000001 177572
6172 053024 012711 010000
6173 053030 023737 000004 140004
6174 053036 001013
6175

```

```

.SBTTL 16/18/22 BIT MEMORY SIZER
;*****
;: MEMORY SIZER FREELY ADAPTED FROM ".SIZE" IN SYSMAC.SML (C3).
;: THIS ROUTINE MUST RESIDE WITHIN THE FIRST 24K (0-137776).
;: ALL GENERAL REGISTERS ARE USED BUT NOT SAVED.
;: MEMORY PARITY ERRORS (IF ANY) ARE IGNORED.
;:
;:
;:
;:SIZE: CLR @MMRO ;:ENSURE THAT MEMORY MANAGEMENT IS OFF!!!
MOV #16,..ABUSW ;: ASSUME 16 BIT ADDRESSING.
MOV @#4,-(SP) ;: SAVE BUS-ERROR...
MOV @#6,-(SP)
MOV @#114,-(SP) ;:...AND PARITY VECTORS.
MOV @#116,-(SP)
MOV SP,R5 ;: SAVE STACK POINTER IN R5.
MOV #116,@#114 ;: IGNORE PARITY ERRORS.
MOV #RTI,@#116
MOV #1$,@#4 ;: SIZE USING "1$" IF KT ISN'T THERE.
MOV #340,@#6
TST MMRO
NOP
NOP ;: FILLER FOR T11.
BR .SIZKT ;: OTHERWISE, SIZE USING THE KT.
MOV #3$,@#4 ;: NO KT -- SET TRAP CATCHER.
CLR R2 ;: FIRST ADDRESS (0).
CLR R3
TST (R2) ;: SIZE FROM 0 UP...
ADD #4000,R2 ;:...IN 1K STEPS...
INC R3
BR 2$ ;:...UNTIL WE TRAP.
SUB #2,R2 ;: R2 = LAST VIRTUAL ADDRESS.
CLR R1 ;: R1 = 0 (PAF DOESN'T APPLY).
BR .SIZXIT ;: RETURN.
.SIZKT: MOV #KIPARO,R1 ;: 1ST "PAR" ADDRESS...
MOV #KIPDRO,R2 ;: ...AND IT'S "PDP"...
MOV #8,R3 ;:...AND THERE ARE 8 OF EACH.
CLR R0 ;: 1ST PAGE IS ZERO.
MOV RO,(R1)+ ;: SET PAR'S = 0, 4K, 8K ... 28K.
MOV #77406,(R2)+ ;: SET PDR'S = 4K, EX-UP, READ/WRITE.
ADD #200,R0 ;: SET NEXT PAGE PAF (+4K)...
SOB R3,1$ ;: ...AND LOOP UNTIL ALL LOADED.
MOV #177600,-(R1) ;: PAR7 IS THE I/O PAGE (PAF).
CLR -(R1) ;: PAR6 WILL DO THE SIZING.
CLR R3 ;: R3 WILL COUNT THE K'S.
MOV #2$,@#4 ;: TRAP TO 2$ IF NO 22 BIT SUPPORT.
TST MMR3 ;: 22 BITS SUPPORTED ??
MOV #20,MMR3 ;: MUST BE, SET 22 BIT MODE...
MOV #3$,@#4 ;:...TRAP TO 3$ IF 22 BIT ADDRESS IS NXM.
BIS #BIT00,@MMRO ;:***** KT ON *****
MOV #10000,(R1) ;: SET PAR6 AT START OF 22 BIT LAND...
CMP @#4,@#140004 ;:...AND LOOK FOR WRAP-AROUND.
BNE 4$ ;: WE'RE HERE IF IT DIDN'T TRAP...
;:...BRANCH IF IT DIDN'T WRAP-AROUND.

```

E14

KXJ11 CA FUNCTIONAL TEST
16/18/22 BIT MEMORY SIZER

```

6176 053040 005037 172516          CLR      MMR3          ;; WRAP-AROUND -- MUST BE 18 BITS ONLY.
6177 053044 000401                    BR      6$
6178 053046 022626                    CMP      (SP)+,(SP)+  ;FIX STACK
6179 053050 012737 000022 050736 6$:  MOV      #18,..ABUSW  ;; SET THE BUS WIDTH - 18.
6180 053056 012704 007600                    MOV      #7600,R4    ;; SET SIZER LIMIT = 124K.
6181 053062 000406                    BR      5$
6182 053064 022626                    CMP      (SP)+,(SP)+  ;FIX STACK
6183 053066 012737 000026 050736 4$:  MOV      #22,..ABUSW  ;; WE HAVE A REAL 22 BIT ADDRESS SPACE.
6184 053074 012704 177600                    MOV      #177600,R4  ;; SET SIZER LIMIT = 2048K.
6185 053100                    5$:
6186 053100 012737 053130 000004 .SIZMEM: MOV    #2$,@#4    ;; TRAP TO 2$ WHEN DONE SIZING.
6187 053106 005011                    CLR      (R1)        ;; SET PAR6 AT 1ST PAGE (0).
6188 053110 005737 140000                    1$:  TST      140000     ;; SIZE USING PAR6 (+0)...
6189 053114 062711 000040                    ADD      #40,(R1)    ;;...IN 1K STEPS.
6190 053120 005203                    INC      R3
6191 053122 021104                    CMP      (R1),R4    ;; REACHED LIMIT ??
6192 053124 103771                    BLO     1$          ;; LOOP IF NOT.
6193 053126 000401                    BR      3$
6194 053130 022626                    2$:  CMP      (SP)+,(SP)+  ;FIX THE STACK
6195 053132 011100                    3$:  MOV      (R1),R0    ;; DONE, SAVE FINAL PAR6...
6196 053134 012711 001400                    MOV      #1400,(R1)  ;;...AND RESET IT TO BANK 6 (24K).
6197 053140 005037 177572                    CLR      @MMR0      ;;***** KT OFF *****
6198 053144 010001                    MOV      R0,R1      ;; RECOVER SIZING RESULT.
6199 053146 162701 000040                    SUB      #40,R1     ;; R1 = LAST 1K PAGE (PAF).
6200 053152 012702 003776                    MOV      #3776,R2   ;; R2 = LAST ADDR IN THAT PAGE.
6201
6202 053156 010137 050732          .SIZXIT: MOV    R1,.LSTPG  ;; RETURN LAST PAGE (PAF)...
6203 053162 010237 050734                    MOV      R2,.LSTAD  ;;...LAST VIRTUAL ADDRESS...
6204 053166 010337 050730                    MOV      R3,.MSIZE  ;;...AND TOTAL MEMORY SIZE (K).
6205 053172 013746 050730                    MOV      .MSIZE,-(SP) ;; REPORT MEMORY SIZE
6206 053176 104401 001163                    TYPE    ,#CRLF
6207 053202 104405                    TYPE    TYP0S
6208 053204 104401 053353                    TYPE    ,KWMEM      ;; MEMORY SIZE MESSAGE
6209 053210 022737 000026 050736  CMP      #22,..ABUSW  ;; AFTER ALL THAT, ARE WE SUPPORTING 22 BIT
6210                                     ;; ADDRESSING?
6211 053216 001003                    BNE     1$          ;;
6212 053220 104401 053312                    TYPE    ,ADR22     ;; YES, TYPE MESSAGE INDICATING 22 BIT ADDRESS
6213 053224 000411                    BR      2$
6214 053226 022737 000022 050736 1$:  CMP      #18,..ABUSW  ;; HOW ABOUT 18 BIT ADDRESSES?
6215 053234 001003                    BNE     3$
6216 053236 104401 053305                    TYPE    ,ADR18     ;; NOPE, TYPE MESSAGE INDICATING NO 22 BIT
6217                                     ;;...SUPPORT
6218 053242 000402                    BR      2$
6219 053244 104401 053300                    3$:  TYPE    ,ADR16
6220 053250 104401 053317                    2$:  TYPE    ,ADRSUP
6221 053254 010506                    MOV      R5,SP      ;; RECOVER OUR STACK POINTER...
6222 053256 012637 000116                    MOV      (SP)+,@#116 ;;...AND THE ERROR VECTORS.
6223 053262 012637 000114                    MOV      (SP)+,@#114
6224 053266 012637 000006                    MOV      (SP)+,@#6
6225 053272 012637 000004                    MOV      (SP)+,@#4
6226 053276 000207                    RETURN
6227 053300          200          061          066  ADR16: .ASCIZ <CRLF>/16 /
6228 053305          200          061          070  ADR18: .ASCIZ <CRLF>/18 /
6229 053312          200          062          062  ADR22: .ASCIZ <CRLF>/22 /
6230 053317          102          111          124  ADRSUP: .ASCIZ /BIT ADDRESSES ARE SUPPORTED/
6231 053353          113          127          040  KWMEM: .ASCIZ /KW OF ARBITER MEMORY INSTALLED/
6232                                     .EVEN

```

```

6233
6234 053412
6235
6236 053412 105737 001207
6237 053416 100516
6238 053420 013746 000004
6239 053424 013746 000006
6240 053430 012737 053766 000004
6241 053436 012737 000340 000006
6242 053444 005037 001244
6243 053450 012701 000004
6244 053454 012703 000006
6245
6246
6247
6248
6249 053460 012704 001256
6250
6251 053464 012737 000001 001412
6252 053472 012705 160100
6253 053476 004737 053734
6254
6255 053502 012703 000010
6256 053506 012705 175400
6257 053512 004737 053734
6258
6259 053516 005737 050770
6260 053522 001003
6261 053524 005737 050772
6262 053530 001415
6263
6264
6265
6266 053532 012737 000002 001412 10#:
6267 053540 012705 162100
6268 053544 004737 053734
6269
6270 053550 012703 000010
6271 053554 012705 177400
6272 053560 004737 053734
6273
6274 053564 012637 000006
6275 053570 012637 000004
6276 053574 032737 100000 001244
6277 053602 001440
6278 053604 104401 053776 8#:
6279 053610 104411
6280 053612 012600
6281 053614 120027 000131
6282 053620 001411
6283 053622 120027 000015
6284 053626 001406
6285 053630 120027 000116
6286 053634 001423
6287 053636 104401 001162
6288 053642 000760
6289 053644 052777 010000 125266 9#:

```

```

;NOW FIND OUT HOW MANY AND WHAT ADDRESS THE IOPS ARE AT IN THE SYSTEM
;
TSTB @#ENVH ;SHOULD WE SIZE?
BMI 12#
MOV @#4,(SP) ;SAVE VECTOR
MOV @#6,-(SP)
MOV #5,@#4 ;LOAD NXM TRAP HANDLER
MOV #PR7,@#6
CLR $DEVH ;CLEAR THE DEVICE MAP
MOV #4,R1 ;LOAD $DEVH WITH VALUE IN R1
MOV #6,R3 ;LOOP CONTROL
;
;SINCE IOP WITH AN ID OF 0 OR 1 HAS ITS INTERFACE TO THE QBUS DISABLED
;WE'LL SKIP OVER THOSE POSITIONS IN THE DEVICE DESCRIPTOR WORDS.
;
MOV #DDW2,R4 ;START OF DEVICE DESCRIPTOR WORDS
; IN E TABLE
;
MOV #1,@#TSTLOC
MOV #160100,R5 ;BASE ADDRESS FOR IOP ID# 2 OR OVER.
JSR PC,1#
;
MOV #10,R3
MOV #175400,R5 ;BASE ADDRESS FOR IOP ID OF 8 AND OVER
JSR PC,1#
;
TST T11# ;IS THE ARBITER T11 BASED
; OR
; LSI11 BASED?
; IF NOT THEN SKIP THE NEXT FEW LINES
; AS IT GETS HAIRY WITH MMU REGISTERS
; IN F11 AND J11 ARBITERS, OCCUPYING
; THE SAME I/O ADDRESSES.
;
MOV #2,@#TSTLOC
MOV #162100,R5 ;BASE ADDRESS OF LOW RANGE IOPS
JSR PC,1#
;
MOV #10,R3
MOV #177400,R5 ;BASE ADDRESS FOR IOP ID OF 8 AND OVER
JSR PC,1#
;
11#: MOV (SP)+,@#6 ;RESTORE VECTORS
MOV (SP)+,@#4
BIT #BIT15,$DEVH ;IS THERE AN IOP ID = 15?
BEQ 15#
8#: TYPE ,KGIOPQ
RDCHR
MOV (SP)+,R0 ;GET CHARACTER
CMPB RO,#'Y ;IS RESPONSE YES?
BEQ 9#
CMPB RO,#15 ;IS RESPONSE THE DEFAULT?
BEQ 9#
CMPB RO,#'N ;IS RESPONSE NO
BEQ 15#
TYPE ,#QUES
BR 8# ;ASK AGAIN
BIS #BIT12,@SWR ;SET BIT IN SWR

```

```

6290 053652 000404          BR      7$
6291
6292          ;
6293          ;COMES HERE IN EITHER STANDALONE OR APT MODE
6294 053654 032777 010000 125256 12$: BIT      #BIT12,@SMR          ;IS IOP ID# 15 TO BE USED AS A
6295                                     ;...KNOWN GOOD DEVICE
6296 053662 001410          BEQ      15$          ;
6297 053664 042737 100000 001244 7$: BIC      #BIT15,#DEVM        ;YES, CLEAR IT OUT OF DEVICE MAP
6298 053672 005037 001310          CLR      $DDW15          ; AND DEVICE DESCRIPTOR WORD
6299 053676 012737 177777 050744 15$: MOV      #1,@#KG.IOP      ;INDICATE A KNOWN GOOD IOP IS PRESENT
6300 053704 013701 001244          MOV      $DEVM,R1          ;LOAD THE FINISHED DEVICE MAP INTO R1
6301 053710 012702 000020          MOV      #16.,R2          ;INIT R2 AS A COUNTER
6302 053714 005037 050746          CLR      IOPCNT          ;CLEAR THE IOP COUNTER
6303 053720 006001          13$: ROR      R1              ;ROTATE DEVICE MAP
6304 053722 103002          BCC      14$          ;BRANCH IF LOW ORDER BIT WAS NOT SET
6305 053724 005237 050746          INC      IOPCNT          ;INCREMENT IOP COUNTER IF LOW ORDER BIT
6306                                     ;... WAS SET
6307 053730 077205          14$: SOB      R2,13$          ;LOOP CONTROL
6308 053732 000207          RTS      PC              ;RETURN TO MAINLINE CODE
6309
6310          ;
6311          ;-----
6312          ;SUBROUTINE
6313          ;CALLED BY SNIFFR
6314 053734 005715          1$: TST      (R5)          ;IF TIME-OUT OCCURS THEN NO IOP AT
6315 053736 000240          NOP          ;... THIS ADDRESS
6316 053740 013714 001412          MOV      @#TSTLOC,(R4)      ;DATA IN WORD INDICATES IOP PRESENT
6317                                     ;1=LOW RANGE ADDRESS, 2=HI RANGE ADDRESS
6318 053744 050137 001244          2$: BIS      R1,@#$DEVM        ;SET BIT IN DEVICE MAP IN ETABLE
6319 053750 062704 000002          ADD      #2,R4          ;POINT TO NEXT IOP TABLE ENTRY
6320 053754 006301          ASL      R1
6321 053756 062705 000040          ADD      #40,R5          ;GET NEXT IOP ADDRESS
6322 053762 077314          SOB      R3,1$
6323 053764 000207          RTS      PC              ;RETURN TO SNIFFR
6324
6325          ;-----NXM TRAP HANDLER
6326 053766 005014          5$: CLR      (R4)          ;ZERO BYTE INDICATES NO IOP AT ADDRESS
6327 053770 012716 053750          MOV      #2$,(SP)        ;PUT NEW RETURN ADDRESS ON STACK
6328 053774 000002          RTI          ;RETURN
6329
6330 053776 200 111 123 KGIOPQ: .ASCIZ <CRLF>#IS IOP ID #15 TO BE USED AS A KNOWN GOOD IOP? (Y/N) <Y>#
6331          .EVEN
6332
6333 054070          QBEX: ;SUBROUTINE TO DETERMINE IF Q BUS EXERCISER IS RESIDENT IN SYSTEM
6334          ;-----
6335 054070 013746 000004          MOV      @#4,-(SP)        ;SAVE VECTORS ON THE STACK
6336 054074 013746 000006          MOV      @#6,-(SP)        ;
6337 054100 012737 054172 000004          MOV      #20$,@#4        ;ANTICIPATE TIMEOUT
6338 054106 012737 000340 000006          MOV      #PR7,@#6        ;
6339 054114 005737 170000          TST      @#170000        ;LOOK FOR RESPONSE
6340 054120 102404          BVS      10$          ;V BIT SET INDICATES TIMEOUT
6341 054122 012737 000001 050760          MOV      #1,@#QBE18      ;18 BIT QBUS EXERCISER RESIDENT
6342 054130 000402          BR      11$
6343 054132 005037 050760          10$: CLR      @#QBE18        ;NO 18 BIT QBE
6344 054136 005737 170020          11$: TST      @#170020        ;LOOK FOR RESPONSE
6345 054142 102404          BVS      15$          ;V BIT SET INDICATES TIMEOUT
6346 054144 012737 000001 050756          MOV      #1,@#QBE22      ;22 BIT QBUS EXERCISER RESIDENT

```

H14

KXJ11-CA FUNCTIONAL TEST
16/18/22 BIT MEMORY SIZER

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 175

```

6347 054152 000402          BR      16$
6348 054154 005037 050756 15$: CLR    @QBE22      ;NO 22 BIT QBE
6349 054160 012637 000006 16$: MOV    (SP)-,@#6   ;RESTORE VECTORS
6350 054164 012637 000004      MOV    (SP)-,@#4
6351 054170 000207          RTS     PC
6352
6353          ;INTERRUPT HANDLER FOR TIMEOUT ON ACCESS TO Q BUS EXERCISER
6354
6355 054172 052766 000002 000002 20$: BIS    #BIT01,2(SP) ;GOT HERE ON TIMEOUT. SET THE V BIT IN
6356          ; RETURN PSW TO INDICATE TIMEOUT OCCURRED.
6357 054200 000002          RTI
6358
6359 054202          ;SHRMEM: ;FIRST LET'S FIND OUT HOW MUCH SHARED MEMORY SPACE IS AVAILABLE
6360          ;ON THE Q BUS.
6364 054202 022737 000026 050736      CMP    #22..,ABUSW   ;IS THIS A 22 BIT SYSTEM?
6365 054210 001410          BEQ    10$
6366 054212 022737 000022 050736      CMP    #18..,ABUSW   ;IS THIS AN 18 BIT SYSTEM?
6367 054220 001410          BEQ    15$
6368 054222 012737 001600 050750      MOV    #1600,$HILIM ;
6369 054230 000407          BR     20$
6370 054232 012737 177600 050750 10$: MOV    #177600,$HILIM ;
6371 054240 000403          BR     20$
6372 054242 012737 007600 050750 15$: MOV    #7600,$HILIM ;
6373 054250 013737 050732 050752 20$: MOV    @#.LSTPG,@#LOWLIM ;
6374 054256 062737 000040 050752      ADD    #40,@#LOWLIM  ;MOVE LAST PAGE TO LOWLIMO
6375 054264 032737 000177 050752      BIT    #177,@#LOWLIM ;MAKE IT POINT TO FIRST NXM ADDR
6376          ;CHECK AND MAKE SURE FIRST NXM
6377          ;PAGE IS A 8KB BOUNDARY.
6377 054272 001406          BEQ    5$
6378 054274 062737 000200 050752      ADD    #200,@#LOWLIM ;IF NOT THEN MAKE IT THE NEXT HIGHER
6379          ;8KB BOUNDARY
6380 054302 042737 000177 050752      BIC    #177,@#LOWLIM ;LOWLIM HOLDS THE FIRST NXM ADDRESS
6381          ;ON AN 8KB BOUNDARY
6382 054310 023737 050752 050750 5$: CMP    @#LOWLIM,$HILIM ;MAKE SURE WE HAVE ENOUGH ROOM
6383 054316 103405          BLO    6$
6384 054320 005037 050754          CLR    MAXBLK
6385          ;NO ROOM FOR SHARED MEMORY. CLEAR
6386 054324 104401 054475          TYPE   ,NOSHMR      ; MAXBLK
6387 054330 000424          BR     7$
6388 054332 013737 050750 050754 6$: MOV    $HILIM,MAXBLK ;CALCULATE THE AMOUNT OF
6389 054340 163737 050752 050754      SUB    @#LOWLIM,MAXBLK ; Q-BUS MEMORY SPACE USABLE
6390 054346 013701 050754          MOV    MAXBLK,R1    ; FOR SHARED MEMORY
6391 054352 005000          CLR    R0
6392 054354 071027 000200          DIV   #200,R0
6393 054360 010037 050754          MOV    R0,MAXBLK
6394          ;MAXIMUM NUMBER OF QBUS 8KB BLOCKS
6395 054364 013746 050754          MOV    MAXBLK,-(SP) ;AVAILABLE FOR SHARED MEMORY
6396 054370 104401 001163          TYPE   ,#CRLF      ;REPORT AMOUNT OF SHARED MEMORY
6397 054374 104405          TYPDS
6398 054376 104401 054420          TYPE   ,SHMAMT
6399 054402 012737 054530 000404 7$: MOV    #ARB404,@#404
6400 054410 012737 000340 000406      MOV    #PR7,@#406
6401 054416 000207          RTS     PC
6402 054420          040      070      113 SHMAMT: .ASCIZ / 8KB BLOCKS OF SHARED MEMORY SPACE AVAILABLE/
6403 054475          200      116      117 NOSHMR: .ASCIZ <CRLF>/NO ROOM FOR SHARED MEMORY/

```



```

6405                                     ;THIS ROUTINE IS ENTERED WHEN THE K2 UNDER TEST SIGNALS THE ARBITER
6406                                     ;THAT IT IS READY TO ACCEPT THE DATA NEEDED TO CONFIGURE THE SHARED
6407                                     ;MEMORY.
6408
6409 054530 005737 055304      ARB404: TST      FIRST      ;SEE IF THIS IS THE FIRST TIME THRU
6410 054534 001034              BNE      2$
6411 054536 013700 052002      MOV      IOPADR,R0
6412 054542 016037 000012 055306  MOV      12(R0),MAX.WT      ;GET MAXIMUM WRITE VALUE
6413 054550 005037 055274      CLR      NBLKS             ;INIT NBLKS
6414 054554 013737 050754 055300  MOV      MAXBLK,REMBLK     ;INIT REMAINING BLOCKS
6415 054562 013737 050752 055276  MOV      LOWLIM,QBSTRT     ;INIT STARTING ADDRESS OF Q BUS NXM
6416 054570 012737 177777 002426  MOV      #-1,@#TRAP4X
6417 054576 012737 055164 000400  MOV      #ARB400,@#400
6418 054604 012737 000340 000402  MOV      #PR7,@#402
6419 054612 012737 054726 000374  MOV      #ARB374,@#374
6420 054620 012737 000340 000376  MOV      #PR7,@#376
6421 054626 005237 055304      2$:      INC      FIRST
6422 054632 022737 000100 055300  CMP      #64.,REMBLK      ;WHILE REMBLK ARE > 64
6423 054640 002007              BGE      3$
6424 054642 162737 000100 055300  SUB      #64.,REMBLK      ;SUBTRACT THE 64 BLOCKS TO BE TESTED
6425 054650 012737 000100 055302  MOV      #64.,CURBLK      ;INIT THE NUMBER OF BLOCKS OF
6426                                     ; QBUS MEMORY TO BE ALLOCATED
6427 054656 000405              BR      4$
6428 054660 013737 055300 055302 3$:      MOV      REMBLK,CURBLK
6429 054666 005037 055300      CLR      REMBLK
6430 054672              4$:
6431 054672 005237 055274      AR.404: INC      NBLKS      ;INCREMENT NUMBER OF BLOCKS TO BE
6432                                     ;ALLOCATED
6433 054676 013700 052002      MOV      IOPADR,R0
6434 054702 013760 055274 000026  MOV      NBLKS,26(R0)     ;GET BASE ADDRESS OF IOP BEING TESTED
6435                                     ;LOAD NUMBER OF BLOCKS TO BE SHARED
6436 054710 013760 055276 000030  MOV      QBSTRT,30(R0)   ;INTO K2 TPR11
6437                                     ;LOAD STARTING Q BUS ADDRESS INTO
6438 054716 012701 000200      MOV      #200,R1 ;****
6439 054722 077101              SOB      R1, .           ;****
6440 054724 000002              RTI

```

```

6451 054726          ARB374: ;WRITE THE SHARED MEMORY THAT HAS BEEN ALLOCATED TO THE QBUS
6452                ;
6521 054726          30$: ;CHECK THAT THE BOUNDARYS ARE BEING HONORED
6522 054726 013701 055276      MOV   QBSTRT,R1          ;STARTING QBUS ADDRESS OF SHARED MEM
6523 054732 010137 172352      MOV   R1,@#KIPAR5      ;
6524 054736 010103          MOV   R1,R3
6525 054740 162703 000200      SUB   #200,R3          ;LAST NXM BEFORE SHARED MEMORY
6526 054744 010337 172346      MOV   R3,@#KIPAR3      ;
6527 054750 013700 055274      MOV   NBLKS,R0         ;
6528 054754 062701 000200      31$: ADD   #200,R1          ;CALCULATE FIRST NXM AFTER SHARED
6529                ;...MEMORY
6530 054760 077003          SOB   R0,31$           ;LOOP CONTROL
6531 054762 010137 172350      MOV   R1,@#KIPAR4      ;
6532 054766 162701 000200      SUB   #200,R1          ;LAST SHARED MEMORY BLOCK
6533 054772 010137 172354      MOV   R1,@#KIPAR6      ;
6534 054776 012702 055134      MOV   #32$,R2         ;
6535 055002 013746 000004      MOV   @#4,-(SP)        ;SAVE VECTOR
6536 055006 012737 055072 000004  MOV   #33$,@#4
6537 055014 052737 000020 172516  BIS   #20,@#172516     ;ENABLE 22 BIT ADDRESSES
6538 055022 052737 000001 177572  BIS   #1,@#MMR0       ;TURN ON MMU
6539 055030 023727 055304 000001  CMP   FIRST,#1        ;
6540 055036 001003          BNE   37$
6541 055040 062702 000002      ADD   #2,R2
6542 055044 000402          BR    38$
6543 055046 005732          37$: TST   @ (R2)+          ;SHOULD TIMEOUT
6544 055050 000413          BR    34$
6545 055052 005732          38$: TST   @ (R2)+          ;HIGH NXM BOUNDARY SHOULD TIMEOUT
6546 055054 000411          BR    34$
6547 055056 012737 055114 000004  MOV   #35$,@#4
6548 055064 005732          TST   @ (R2)+          ;SHOULD NOT TIME OUT
6549 055066 005732          TST   @ (R2)+          ;
6550 055070 000425          BR    36$
6551                ;
6552 055072 062716 000002      33$: ADD   #2,(SP)          ;BUMP RETURN ADDRESS
6553 055076 000002          RTI
6554                ;
6555                34$:
6556 055100          MOV   IOPADR,R0
6557 055100 013700 052002          MOV   #ARBER,10(R0)   ;SHARED MEMORY OUT OF BOUNDS ;+++
6558 055104 012760 000001 000010  BR    40$
6559 055112 000420          ;
6560                35$:
6561 055114          MOV   IOPADR,R0
6562 055114 013700 052002          MOV   #ARBER,10(R0)   ;SHARED MEMORY TIMED-OUT ;+++
6563 055120 012760 000002 000010  MOV   #40$,(SP)      ;BAIL OUT AFTER ERROR
6564 055126 012716 055154          RTI
6565 055132 000002          ;
6566                32$:
6567 055134 077776          .WORD 77776          ;LOW BOUNDARY NXM
6568 055136 100000          .WORD 100000         ;HIGH BOUNDARY NXM
6569 055140 120000          .WORD 120000         ;FIRST SHARED MEM
6570 055142 157776          .WORD 157776         ;LAST SHARED MEM
6571                ;
6572 055144 012637 000004      36$: MOV   (SP)+,@#4
6573 055150 000137 055164      JMP   ARB400
6574                ;
6575 055154 012637 000004      40$: MOV   (SP)+,@#4          ;RESTORE VEC
6576 055160 000137 055234      JMP   EX400          ;BAIL OUT AFTER ERROR

```

```

6574 055164          ARB400: ;NOW DETERMINE IF ALL BLOCKS HAVE BEEN TESTED. IF NOT THEN
6575                                     ;INITIALIZE THE VALUES NEEDED TO CONTINUE.
6576
6577 055164 023737 055274 055302      CMP      NBLKS,CURBLK      ;DID WE TEST ALL THE BLOCKS IN THE
6578                                     ; CURRENT 256KW SEGMENT?
6579 055172 001012          BNE      2$
6580 055174 005037 055274          CLR      NBLKS
6581 055200 005737 055300          CLR      NBLKS
6582 055204 001407          TST      REMBLK
6583                                     BEQ      25$
6584 055206 062737 020000 055276      ADD      #20000,QBSTRT  ;CLEAR NUMBER OF BLOCKS TO BE ALLOCATED
6585                                     ;HAVE ALL BLOCKS BEEN TESTED?
6586 055214 000137 054530          JMP      ARB400        ;IF YES THEN BAIL OUT
6587                                     ;ELSE
6588 055220 000137 054672          ;2$: JMP      AR.404    ; GET NEXT STARTING QBUS ADDRESS
6589                                     ;
6590 055224 013700 052002          ;25$: MOV     IOPADR,R0  ; OF NEXT 64 4KW BLOCKS
6591 055230 005060 000010          CLR     10(R0)
6592 055234 005037 177572          EX400: CLR     @#MMRO  ;DO IT AGAIN
6593 055240 005000          CLR     R0
6594 055242 012702 000007          MOV     #7,R2
6595 055246 012701 172340          MOV     #KIPAR0,R1
6596 055252 010021          ;1$: MOV     R0,(R1)+
6597 055254 062700 000200          ADD     #200,R0
6598 055260 077204          SOB     R2,1$
6599 055262 005037 002426          CLR     TRAP4X
6600 055266 005037 055304          CLR     FIRST
6601 055272 000002          RTI
6602
6603
6604 055274 000000          NBLKS: .WORD 0
6605 055276 000000          QBSTRT: .WORD 0
6606 055300 000000          REMBLK: .WORD 0
6607 055302 000000          CURBLK: .WORD 0
6608 055304 000000          FIRST: .WORD 0
6609 055306 000000          MAX.WT: .WORD 0
;SIGNAL K2 THAT TEST IS DONE
;MMU OFF
;INIT R0
;SET UP FOR COUNT
;INIT R1
;INIT KIPARS
;POINT TO NEXT PAGE
;LOOP CONTROL
;
;DONE WITH TEST

```

```

6611
6612 055310          SHOSYS: ;NOW SHOW THE SYSTEM IOPS ON THE ARBITER'S CONSOLE
6613
6614 055310 105737 001207      TSTB   @#$ENVM           ;SHOULD WE SIZE?
6615 055314 100444          BMI     33$
6616 055316 012704 001256      MOV     #$DDW2,R4        ;POINT TO FIRST IOP TABLE ENTRY
6617 055322 012702 000006      MOV     #6,R2           ;LOOP CONTROL
6618 055326 012737 002000 001412  MOV     #2000,@#TSTLOC  ;BIT SET FOR HI RANGE
6619 055334 012737 160100 055510  MOV     #160100,22$     ;BASE ADDRESS OF IOPS
6620 055342 012737 000002 050742  MOV     #2,IOP.ID       ;FIRST IOP ID NUMBER
6621 055350 004737 055430      JSR     PC,40$
6622
6623 055354 012702 000010          MOV     #8,R2           ;LOOP CONTROL
6624 055360 012737 175400 055510  MOV     #175400,22$     ;BASE ADDRESS OF IOPS WITH ID# 8 OR OVER
6625 055366 004737 055430      JSR     PC,40$
6626
6627 055372 022737 000001 050760  CMP     #1,@#QBE18      ;IS THERE AN 18 BIT QBE IN THE SYSTEM
6628 055400 001004          BNE     31$
6629 055402 104401 055536      TYPE   ,32$
6630 055406 104401 055551      TYPE   ,35$
6631 055412 022737 000001 050756 31$:  CMP     #1,@#QBE22      ;IS THERE A 22 BIT QBE IN THE SYSTEM?
6632 055420 001002          BNE     33$
6633 055422 104401 055544      TYPE   ,34$
6634 055426 000207          RTS     PC
6635
6636 055430 005714          40$:  TST     (R4)
6637 055432 001430          BEQ     30$             ;IF ZERO THEN NO IOP PRESENT
6638 055434 132714 000001      BITB   #1,(R4)         ;WHAT IS THE ADDRESS RANGE
6639 055440 001004          BNE     41$
6640 055442 053737 001412 055510  BIS     @#TSTLOC,22$    ;SET HI RANGE FOR PRINT OUT
6641 055450 000403          BR      42$
6642 055452 043737 001412 055510 41$:  BIC     @#TSTLOC,22$    ;SET LO RANGE FOR PRINT OUT
6643 055460          42$:
6644 055460 104401 055605          1$:  TYPE   ,7$
6645 055464 013746 050742      MOV     IOP.ID,-(SP)
6646 055470 104405          TYPDS
6647 055472 104401 055620      TYPE   ,8$
6648 055476 104401 055627      TYPE   ,10$
6649 055502 104401 055641      3$:  TYPE   ,12$
6650 055506 012746          MOV     (PC)+,-(SP)
6651 055510 000000          22$:  .WORD  0
6652 055512 104402          TYPOC
6653 055514 062704 000002      30$:  ADD     #2,R4
6654 055520 005237 050742          INC     IOP.ID
6655 055524 062737 000040 055510  ADD     #40,22$
6656 055532 077242          SOB    R2,40$
6657 055534 000207          RTS     PC
6658
6659 055536 015 012 061 32$:  .ASCIZ <CR><LF>/18 /
6660 055544 015 012 062 34$:  .ASCII <CR><LF>/22 /
6661 055551 102 111 124 35$:  .ASCIZ /BIT QBUS EXERCISER RESIDENT/
6662 055605 015 012 111 7$:  .ASCIZ <CR><LF>/IOP ID# /
6663 055620 040 111 123 8$:  .ASCIZ / IS A /
6664 055627 113 130 112 10$:  .ASCIZ /KXJ11-CA /
6665          ;11$:  .ASCIZ /KXT11-CA /
6666 055641 040 101 124 12$:  .ASCIZ / AT ADDRESS /
6667          ;20$:  .ASCIZ / IS PRESENT BUT NOT AVAILABLE/

```

```

        .EVEN
        ;
        IOPV02: IOP$TR,160100
        IOPV03: IOP$TR,160140
        IOPV04: IOP$TR,160200
        IOPV05: IOP$TR,160240
        IOPV06: IOP$TR,160300
        IOPV07: IOP$TR,160340
        IOPV08: IOP$TR,175400
        IOPV09: IOP$TR,175440
        IOPV10: IOP$TR,175500
        IOPV11: IOP$TR,175540
        IOPV12: IOP$TR,175600
        IOPV13: IOP$TR,175640
        IOPV14: IOP$TR,175700
        IOPV15: IOP$TR,175740

        IOP$T: MOV     R1, (SP)           ;SAVE GPRS
                MOV     RO, (SP)
                MOV     @4(SP),R1       ;GET LOW RANGE ADDRESS OF TPRO FROM
                ; INTERRUPTING IOP.
        MOV     R1,RO
        ASH     #-4,RO                 ;DETERMINE WHICH IOP INTERRUPTED
        BIC     #+C36,RO               ;MASK OFF BITS <4:1>
        ADD     #+DDW0,RO              ;GET DESCRIPTOR WORD
        BITB    #BIT00,(RO)           ; IS THE IOP IN THE HIGH RANGE?
        BNE     1$,
        BIS     #2000,R1               ;BOOST IT UP TO HIGH RANGE
        1$:    MOV     12(R1),RO       ;GO FIND OUT WHAT IT WANTS TO DO
                ; FROM THE DATA IN TPRS
                ; MAKE SURE ITS NOT ODD
                ; UH OH, ITS ODD; BAIL OUT
                ; RO POINTS TO THE ROUTINE
        BIT     #1,RO
        BNE     20$,
        ADD     #25$,RO
        JSR     PC,@(RO)
        20$:   CLR     10(R1)
                ;INFORM K2 THAT WE ARE DONE
                ;RESTORE REGISTERS
        MOV     (SP)+,RO
        MOV     (SP)+,R1
        CMP     (SP)+,(SP)+           ;FAKE A RETURN FROM TRAP
        RTI
        25$:   .WORD   NOTST, WT.LCK, TST.ST, ARBTPR, Q$HIT

        ;TEST THAT SHARED MEMORY CAN BE ACCESSED FROM THE QBUS USING THE LOCK
        ;INSTRUCTIONS
        ;
        WT.LCK: MOV     @+KIPARS,-(SP)   ;SAVE VALUE IN KIPARS
                MOV     RO,-(SP)         ;SAVE REGISTERS
                MOV     R1,-(SP)
                MOV     R2,-(SP)
        MOV     LOWLIM,@+KIPARS         ;POINT TO SHARED MEMORY
        BIS     #1,@+MPRO              ;TURN ON MMU
        CLR     RO                       INITIALIZE RO
        MOV     #120000,R2
        ;INIT 4KW BLOCK OF SHARED MEMORY WITH PATTERN 000000 TO 007776
        1$:    MOV     #4096.,R1
                ;
                .WORD   7322           ;***TEST WRTLCK INSTRUCTION
                ;
                ADD     #2,RO
                SOB     R1,1$          ;LOOP CONTROL
  
```

```

6725 056126 042737 000001 177572      BIC    #1,@MMIO      ;OFF THE MMU
6726 056134 012602                    MOV    (SP)+,R2      ;
6727 056136 012601                    MOV    (SP)+,R1      ;
6728 056140 012600                    MOV    (SP)+,R0      ;
6729 056142 012637 172352            MOV    (SP)+,@KIPARS ;RESTORE KIPARS
6730 056146 000207                    RTS    PC
6731
6732
6733 056150 013746 172352            ;NOW TEST THE TSTSET LOCK INSTRUCTION
TST.ST: MOV    @KIPARS, (SP)          ;SAVE VALUE IN KIPARS
6734 056154 010046                    MOV    RO,-(SP)      ;SAVE REGISTERS
6735 056156 010146                    MOV    R1,-(SP)
6736 056160 010246                    MOV    R2,-(SP)
6737 056162 013737 050752 172352    MOV    LOWIM,@KIPARS ;POINT TO SHARED MEMORY
6738 056170 052737 000001 177572    BIS    #1,@MMIO      ;TURN ON MMU
6739 056176 012701 010000            MOV    #4096,R1      ;
6740 056202 012702 120000            MOV    #120000,R2    ;
6741
6742
6743
6744
6745
6746
6747
6748
6749 056206 007222                    4$:   .WORD    7222      ;****TEST TSTSET INSTRUCTION
6750 056210 077102                    SOB    R1,4$         ;LOOP CONTROL
6751 056212 042737 000001 177572    BIC    #1,@MMIO      ;OFF THE MMU
6752 056220 012602                    MOV    (SP)+,R2      ;
6753 056222 012601                    MOV    (SP)+,R1      ;
6754 056224 012600                    MOV    (SP)+,R0      ;
6755 056226 012637 172352            MOV    (SP)+,@KIPARS ;RESTORE KIPARS
6756 056232 000207                    RTS    PC
6757
6758 056234
6759
6760 056234 013746 000110            ARBTPR: ;LOOP ON READING TPRO UNTIL IT TURNS TO ZERO. THE LOCAL J11
6761 056240 013746 000112            ;WILL ALSO BE ACCESSING THE TPR FILE, MAKING SURE NO NXMS OCCUR
6762 056244 012737 056272 000110    MOV    @#110,-(SP)   ;SAVE VECTOR
6763 056252 012737 000340 000112    MOV    @#112,-(SP)
6764
6765 056260 106427 000000            MOV    #2,@#110     ;NEW HANDLER
6766 056264 005711                    MOV    #340,@#112
6767 056266 001376                    ;R1 POINTS TO TPRO OF THE KXJ11 UNDER TEST
6768 056270 000401                    MTPS    #0           ;LOWER INTERRUPT LEVEL
6769
6770 056272 022626                    1$:   TST    (R1)      ;IS R1 ZERO YET?
6771
6772 056274 012637 000112            BNE    1$           ;MUST BE ZERO
6773 056300 012637 000110            BR     3$           ;COMES HERE ONLY IF KXJ11 UNDER TEST HAS A FAILURE AND INTERRUPTS VIA 110
6774 056304 000207                    2$:   CMP    (SP)+,(SP)+ ;FIX STACK
6775
6776 056306
6777
6778 056306 010046                    ;NORMAL EXIT
6779 056310 010146                    3$:   MOV    (SP)+,@#112 ;AND VECTORS
6780 056312 013746 000004            MOV    (SP)+,@#110  ;
6781 056316 013746 000006            RTS    PC
Q$HIT: ;THIS ROUTINE WILL KEEP ON ACCESSING THE SHARED MEMORY CAUSING Q HITS
;ON THE KXJ11-CA.
MOV    RO,-(SP)      ;SAVE REGISTERS
MOV    R1,-(SP)
MOV    @4,-(SP)      ;SAVE NXM VECTOR
MOV    @#6,-(SP)

```

```

6782 056322 013746 000110      MOV      @#110,-(SP)      ;SAVE VECTOR
6783 056326 013746 172352      MOV      @#KIPARS,-(SP)  ;SAVE KIPARS
6784 056332 012737 056422 000110  MOV      #2,@#110        ;NEW HANDLER
6785 056340 012737 056412 000004  MOV      #6,@#4          ;NEW NXM HANDLER
6786 056346 012737 000340 000006  MOV      #PR7,@#6
6787 056354 013737 050752 172352  MOV      LOWLIM,@#KIPARS ;PUT LOW END OF SHARED MEM IN KIPARS
6788 056362 052737 000001 177572  BIS      #1,@#MMRO      ;TURN ON SHARED MEMORY
6789 056370 106427 000000      MTPS     #0              ;ALLOW INTERRUPTS
6790 056374 012700 120000      1$:     MOV      #120000, R0
6791 056400 012701 010000      MOV      #4096.,R1
6792 056404 005220      5$:     INC      (R0).        ;READ/MOD/WRITE SHARED MEMORY
6793 056406 077102      SOB      R1,5$
6794 056410 000771      BR       1$
6795 056412
6796
6797 056412 012777 040000 173362  6$:     ;COMES HERE IF SHARED MEMORY TIME-OUT OCCURS. IF THIS HAPPENS
;THE DEVICE UNDER TEST IS HUNG. RESET IT AND LINK UP
6798 056420 000000      MOV      #40000,@IOPADR ;RESET THE IOP UNDER TEST
6799
6800 056422      2$:     ;COMES HERE WHEN K2 INTERRUPTS ARBITER AFTER TESTING IS DONE
6801 056422 012716 056430      MOV      #3,(SP)        ;ALTER RETURN ADDRESS
6802 056426 000002      RTI      ;FIX UP STACK AFTER INTERRUPT
6803 056430 012637 172352      3$:     MOV      (SP)+,@#KIPARS ;RESTORE IT ALL
6804 056434 012637 000110      MOV      (SP)+,@#110
6805 056440 012637 000006      MOV      (SP)+,@#6      ;RESTORE VECTORS
6806 056444 012637 000004      MOV      (SP)+,@#4
6807 056450 012601      MOV      (SP)+,R1
6808 056452 012600      MOV      (SP)+,R0
6809 056454 042737 000001 177572  BIC      #BIT00,@#MMRO   ;TURN OFF MMU
6810 056462 000207      RTS      PC
6811
6812 056464 000207      NOTST:   RTS      PC
6813
6814
6815 056466 056474      KG#T04: .WORD    KG#T06
6816 056470 100000      .WORD    <QBMEM!UP>
6817 056472 000002      KG#T05: .WORD    2
6818 056474      KG#T06: ;CHAIN TO TRANSFER FROM SHARED MEMORY TO KNOWN GOOD IOP
6819 056474 001602      <CARA!CARB!COP!MODE>
6820 056476 000000      .WORD    0              ;SHARED MEMORY SEGMENT
6821 056500 000000      .WORD    0              ;SHARED OFFSET
6822 056502 000000 020000      <K2MEM!UP>, 20000
6823 056506 010000      4096.
6824 056510 000030 100140      <SWRQ!HM>,<CTC!TRW!INTLV>
6825
6826 056514      KG#T07: ;CHAIN TO TRANSFER FROM KNOWN GOOD IOP TO SHARED MEMORY
6827 056514 001602      <CARA!CARB!COP!MODE>
6828 056516 000000      .WORD    0              ;SHARED MEMORY SEGMENT
6829 056520 000000      .WORD    0
6830 056522 000000 030000      <K2MEM!UP>, 30000
6831 056526 004000      2048.
6832 056530 000030 000160      <SWRQ!HM>,<TRW!INTLV!FLIP>

```

```

6836 .SBTTL      Q BUS INTERRUPT (QIR) SERVICE ROUTINES
6837 ; *****
6838 ; * NOTE: THE Q-BUS INTERRUPT SERVICE ROUTINES HAVE BEEN REVISED TO *
6839 ; * COMPARE THE STACK POINTER WITH 1100. IF THE STACK POINTER IS *
6840 ; * HIGHER THAN 1100, WE ASSUME APT IS IN THE LOAD PROCESS AND THE *
6841 ; * ROUTINE WILL SEND A BREAK TO THE IOP BEFORE RETURNING TO THE *
6842 ; * LOADER. IF APT IS NOT LOADING, THE INTERRUPTS WILL *
6843 ; * BE SERVICED. JEANELL CUNNINGHAM, 02 MAY 1984. *
6844 ; *****
6845 ;
6846 ; THESE ROUTINES ARE EXECUTED IN THE ARBITER PROCESSOR
6847 ; IN RESPONSE TO ANY OF THREE Q-REQUESTS:
6848 ;
6849 ; 1. (QIRV) -- DO NOTHING, JUST DISMISS.
6850 ; 2. (QIRV1) -- EXECUTE BUS-RESET AND DISMISS.
6851 ; 3. (QIRV2) -- WRITE LAST WORD IN NON-ARBITER DUAL-PORT-RAM(S).
6852 ;
6853 QREQ2: CMP     SP, #1100      ; IS APT LOADING? (SP GREATER THAN 1100?)
6854       BHI     OFFSBC        ; IF SO, DON'T DO THIS
6855       MOV     RO, -(SP)     ; SAVE RO.
6856       MOV     @#4, -(SP)   ; SAVE ERROR VECTOR...
6857       MOV     @#6, -(SP)   ; ...CHANGE IT...
6858       MOV     #1$, @#4
6859       MOV     #340, @#6
6860       BR     4$            ; ...AND SKIP NEXT 2.
6861       BIS     #2, 2(SP)    ; ON BUS-ERROR, SET "V"...
6862       RTI
6863 ;
6864 ; ACCESS (READ-MOD-WRITE) WORD 15 IN Q-BUS DPR WINDOW.
6865 ;
6866 4$:   MOV     IOPADR, RO    ; LAST DPR WORD INDEX.
6867 2$:   COM     36(RO)       ; DPR<15> CPU
6868       NOP
6869 3$:   MOV     (SP)+, @#6    ; RESTORE ERROR VECTOR.
6870       MOV     (SP)+, @#4
6871       MOV     (SP)+, RO    ; RESTORE RO.
6872       BR     SAYYES       ; SEND 'Y OVER SERIAL LINE AND DISMISS. ;FX8JC
6873 ;
6874 QREQ1: CMP     SP, #1100    ; IS APT LOADING? (SP GREATER THAN 1100?)
6875       BHI     OFFSBC        ; IF SO, SEND BREAK TO IOP.
6876       RESET
6877       BR     SAYYES       ; BUS-RESET.
6878 ; SEND 'Y OVER SERIAL LINE AND DISMISS. ;FX8JC
6879 QREQ:  CMP     SP, #1100    ; IS APT LOADING? (SP GREATER THAN 1100?)
6880       BHI     OFFSBC        ; IF SO, SEND BREAK TO IOP.
6881       BR     SAYYES       ; SEND 'Y OVER SERIAL LINE AND DISMISS. ;FX8JC
6882 ;
6883 OFFSBC: RTI                ; DISMISS INTERRUPT.
6884 ;
6885 ;
6886 SAYYES: MOV     R1, -(SP)    ; SAVE R1. ;FX8JC
6887         MOV     SLU2, R1    ; POINT TO SECOND SERIAL LINE. ;FX8JC
6888 1$:   TSTB    4(R1)        ; WAIT FOR READY. ;FX8JC
6889         BPL     1$          ; ;FX8JC
6890         MOVB   #'Y, 6(R1)   ; SEND A 'Y FOR "YES, I DID IT". ;FX8JC
6891         MOV     (SP)+, R1    ; RESTORE R1. ;FX8JC
6892         RTI                ; AND DISMISS INTERRUPT. ;FX8JC
  
```


D15

```

6895
6896
6897
6898
6899
6900
6901
6902
6903
6904
6905
6906      001100
6907
6908
6909
6910 056702 000472
6911 056704 013700 056730
6912 056710 043760 056764 000004
6913 056716 053760 056766 000004
6914 056724 000461
6915
6916 056726 177560
6917 056730 176500
6918 056732 000000
6919 056734 176500
6920 056736 176510
6921 056740 176520
6922 056742 176530
6923 056744 176540
6924 056746 176550
6925 056750 176560
6926 056752 176570
6927 056754 176600
6928 056756 176610
6929 056760 176620
6930 056762 176630
6931
6932 056764 177400
6933 056766 164000
6934
6935
6936
6937
6938
6939 056770 012706 001100
6940 056774 106427 000000
6941 057000 004737 057672
6942 057004 103015
6943 057006 120027 000032
6944 057012 001003
6945 057014 004737 057774
6946 057020 000767
6947 057022 120027 000100
6948 057026 001002
6949 057030 004737 057646
6950 057034 004737 057646
6951 057040 004737 057702

```

```

.SBTTL REMOTE LOADER AND VIRTUAL TERMINAL (V4.0).
;*****
; SUBROUTINES TO LINK TO AND CONTROL A TARGET SBC 11 CPU.
; TARGET MACHINE MUST BE CAPABLE OF EXECUTING IT'S OWN MICRO-CDT
; (MACRO-ODT IN T11) AND MUST HAVE AT LEAST 16KW RAM AT ABSOLUTE 0.
;
; *****
; USP CHANGED FROM BUF2 TO 1100 TO PREVENT TOTAL WIPE-OUT OF PROGRAM
; WHEN STACK RUNS AWAY. JEANELL CUNNINGHAM, 30 APRIL 1984.
; *****

USP=1100
; ENTRY FROM 200G (AUTO-START UNDER XXDP OR APT).
;
; REMOTE: BR      LOAD      ; NOP TO ENABLE SPEED CHANGER.      ;FX8JC
;          MOV     SLU2,R0   ; ***
;          BIC     SPDMSK,4(R0) ; *** IF SPEED PROGRAMMABLE
;          BIS     SPDSET,4(R0) ; *** SET SPEED = 9600 (OR AS NEEDED).
;          BR      LOAD

;
; SLU1: 177560      ; 177560 = CONSOLE.
; SLU2: .WORD      176500 ; DEFAULT SLU2
; SLUADR: .WORD    0
;          176500      ; 176500 = MXV11 PORT 1..DLV11-J PORT 0
;          176510      ; 176510 = DLV11-J PORT 1
;          176520      ; 176520 = DLV11-J PORT 2
;          176530
;          176540
;          176550
;          176560
;          176570
;          176600
;          176610
;          176620
;          176630
;          ...175610 = DLV11-E, OR F OR WHATEVER.
;          SPDMASK: 177400 ; PBR BIT MASK (IF DC319 DLART, USE 72).
;          SPDSET: 164000 ; SET PBR 9600 (IF DC319 DLART, USE 52).
;
; LINK (VIRTUAL TERMINAL) MODE.
; WHEN ECHOING ODT'S PROMPT, CONVERT @ TO @@.
; IN ORDER TO DISTINGUISH SBC HALT (@@) FROM MASTER HALT (@).
;
; LINK:  MOV     #USP,SP   ; RESET STACK POINTER.
;        MTPS   #0        ; LOWER CPU.
1$:     CALL    SBCIN
;        BCC    4$
;        CMPB   RO,#32    ; SBC'S MAILBOX COMMAND <+Z> ??
;        BNE   2$        ; SKIP IF NOT...
;        CALL   UPAMB     ; ...ELSE, UPDATE ARBITERS COPY.
;        BR    1$
2$:     CMPB   RO,#'@     ; SBC'S ODT PROMPT ??
;        BNE   3$        ; SKIP IF NOT...
;        CALL   TTYOUT    ; ...ELSE, ECHO IT TWICE (@@).
3$:     CALL   TTYOUT
4$:     CALL   TTYIN

```

E15

6952	057044	103355		BCC	1\$	
6953	057046	120027	000033	CMPB	RO,#33	; <ESC> ??
6954	057052	001003		BNE	5\$; SKIP IF NOT...
6955	057054	004737	057522	CALL	BRK	;...ELSE, SEND BREAK (SBC HALT).
6956	057060	000747		BR	1\$	
6957	057062	004737	057636	CALL	SBCOUT	
6958	057066	000744	5\$:	BR	1\$; LOOP FOREVER.

```

6960
6961
6962
6963 057070 106427 000340
6964 057074 027727 121000 000002
6965 057102 001406
6966 057104 012737 002542 000100
6967 057112 012737 000300 000102
6968 057120 012737 043002 000034
6969 057126 012737 000340 000036
6970
6971 057134 012706 001100
6972 057140 104401 060430
6973 057144 004737 057522
6974 057150 004737 057736
6975 057154 004737 060264
6976 057160 004737 060242
6977
6978
6979
6980
6981
6982
6983 057164 012746 077416
6984 057170 012746 060550
6985 057174 012746 000125
6986 057200 016602 000004
6987 057204 004737 057406
6988 057210 000407
6989 057212 017602 000002
6990 057216 062766 000002 000002
6991 057224 004737 057414
6992 057230 004737 057744
6993
6994 057234 005316
6995 057236 100365
6996 057240 005002
6997 057242 004737 057422
6998 057246 004737 057736
6999 057252 022626
7000 057254 012602
7001 057256 004737 060276
7002
7003
7004
7005
7006 057262 012702 001100
7007 057266 004737 060304
7008
7009 057272 012700 000120
7010 057276 004737 057636
7011 057302 004737 057752
7012
7013
7014
7015
7016 057306 004737 057672

; REMOTE LOADER - SET UP THE ARBITER AND GET IN TOUCH WITH THE SBC.
;
LOAD:  MTPS    #PR7          ; RAISE CPU.
      CMP     @100,#RTI      ; BEVNT DISMISSED ??
      BEQ     1$            ; SKIP IF SO.
      MOV     #DISMISS,@#100 ; NO, SET TO DISMISS...
      MOV     #300,@#102    ; ...AT PRI 6.
1$:    MOV     #TRAP,@#34    ; SET TRAP VECTOR...
      MOV     #340,@#36     ; ...FOR SYSMAC "TYPE" CALLS.

RELOAD: MOV     #USP,SP      ; RESET STACK POINTER.
      TYPE    ,WAIT1        ; "WAIT..."
      CALL    BRK           ; BREAK TO THE SBC...
      CALL    WAIT00        ; ...AND WAIT FOR HIS @.
      CALL    SETPSW        ; RAISE PSW.
      CALL    SETREGS      ; SET MMU AND MAP-RAM OFF.

;
; OK, LOAD AND START THE IMAGE LOADER VIA SBC'S ODT.
; LOAD SEQUENCE REQUIRES 21. CHARS PER WORD TRANSFERRED.
;   ADDR / DATA <SP> DATA <LF>
; WHICH YIELDS AN EFFECTIVE LOAD RATE OF 45. WORDS/SEC AT 9600 BAUD.
;
      MOV     #LSTRT,-(SP)   ; PUSH TARGET LOCAL STARTING ADDRESS...
      MOV     #LOAD,-(SP)   ; ...LOADER SOURCE ADDRESS...
      MOV     #LSIZE,-(SP)  ; ... AND WORD COUNT.
      MOV     4(SP),R2      ; TARGET ADDRESS => R2.
      CALL    OPNADR        ; XMIT ADDRESS<SLASH>.
      BR      2$
1$:    MOV     @2(SP),R2     ; NEXT DATA WORD => R2.
      ADD     #2,2(SP)      ; ... AND BUMP POINTER.
      CALL    DATALF        ; XMIT DATA<LF>.
2$:    CALL    WAIT40       ; WAIT FOR <SP> AFTER EACH...
      ; ... ADDRESS IS OPENED IN THE SBC.
      DEC     (SP)         ; BUMP WORD COUNT...
      BPL    1$           ; ... AND LOOP 'TIL ALL WORDS GONE.
      CLR    R2
      CALL    DATACR      ; TERMINATE DATA SEQ WITH <CR>...
      CALL    WAIT00       ; ... AND WAIT FOR ODT READY.
      CMP    (SP)+,(SP)+   ; OK, DISCARD WC AND SRC POINTER.
      MOV    (SP)+,R2      ; POP TARGET LOCAL START ADDR => R2...
      CALL    SETPC        ; ...SET HIS PC = (R2)...

; *****
; THESE TWO LINES ADDED TO INSURE VALID STACK POINTER IN SBC BEFORE SENDING
; PROCEED COMMAND.  JEANELL CUNNINGHAM, 7-JUN-84
;
      MOV     #USP,R2      ; PUT USP INTO R2 TO SEND TO SBC
      CALL    SETSP       ; ... SET SBC SP TO (R2)
; *****
; ... AND XMIT THE P(ROCEED) COMMAND.
      MOV     #'P,RO
      CALL    SBCOUT
      CALL    WAITP       ; WAIT FOR P(ROCEED) ECHO.

;
; NOW, THE TARGET MACHINE IS EXECUTING THE IMAGE LOADER.
; IF KXT, HE'LL TRY A DMA LOAD FIRST.
;
3$:    CALL    SBCIN      ; WAIT FOR SOMETHING TO HAPPEN.
    
```

```

7017 057312 103375          BCC      3$
7018 057314 120027 000033  CMPB    R0,#33      ; IF <33> DMA LOAD WAS SUCCESSFUL...
7019 057320 001426          BEQ     8$           ; ...SAY SO AND GET OUT.
7020 057322 120027 000034  CMPB    R0,#34      ; IF <34> DMA FAILED OR NOT SUPPORTED...
7021 057326 001402          BEQ     4$           ; ...START A SERIAL LOAD.
7022 057330 000000          HALT
7023 057332 000700          BR      RELOAD      ; W-T F !!!
7024
7025 057334 005003          4$:    CLR     R3           ; INIT 1KW MARKER...
7026 057336 005002          CLR     R2           ; ...AND ADDRESS POINTER.
7027 057340 112200          5$:    MOVB   (R2)+,R0
7028 057342 004737 057636  CALL    SBCOUT      ; XFER A BYTE.
7029 057346 005203          INC     R3           ; BUMP WORD COUNT.
7030 057350 032703 003777  BIT     #3777,R3    ; 1KW DONE ??
7031 057354 001002          BNE    6$           ; SKIP IF NOT.
7032 057356 104401 060444  TYPE    ,KMARK      ; ELSE, MARK IT.
7033 057362 020227 051507  6$:    CMP     R2,#LASTAD+1
7034 057366 101764          BLOS   5$           ; LOOP 'TIL DONE.
7035 057370 004737 057672  7$:    CALL    SBCIN   ; WAIT FOR DONE ACKNOWLEDGE <34>.
7036 057374 103375          BCC     7$
7037
7038 057376 104401 060450  8$:    TYPE    ,LOADED ; "SBC LOADED"
7039 057402 000137 056770  JMP     LINK        ; ENTER LINK MODE.
7040
7041          ; SUBROUTINE TO TRANSMIT ASCII ADDRESS AND/OR DATA FROM R2.
7042          ; XMIT ADDR<SLASH> TO OPEN ADDRESS FOR INPUT.
7043          ; XMIT DATA<LF> TO STORE DATA, AND OPEN NEXT ADDRESS.
7044          ; XMIT DATA<CR> TO CLOSE WORD AND TERMINATE DATA SEQUENCE.
7045
7046 057406 012746 000057  OPNADR: MOV    #' /,-(SP) ; TERMINATE WITH SLASH...
7047 057412 000402          402      ; ...OR...
7048 057414 012746 000012  DATA LF: MOV   #12,-(SP) ; ...LINE FEED...
7049 057420 000402          402      ; ...OR...
7050 057422 012746 000015  DATA CR: MOV   #15,-(SP) ; ...RETURN.
7051 057426 012703 000006  MOV     #6,R3       ; IN ANY CASE, XMIT 6 DIGITS...
7052 057432 012704 000001  MOV     #1,R4       ; ...THE MSD BEING JUST 1 BIT...
7053 057436 000402          402
7054 057440 012704 000003  1$:    MOV     #3,R4   ; ...AND 3 BITS EACH FOR THE REST..
7055 057444 005001          CLR     R1           ; ASSEMBLE IN R1.
7056 057446 006302          2$:    ASL     R2
7057 057450 006101          ROL     R1
7058 057452 077403          SOB    R4,2$        ; ROTATE DTGIT TO R1<2:0>...
7059 057454 062701 000060  ADD     #'0,R1      ; ...MAKE ASCII.
7060 057460 010100          MOV     R1,R0
7061 057462 004737 057636  CALL    SBCOUT      ; XMIT NEXT CHAR.
7062 057466 004737 057672  3$:    CALL    SBCIN
7063 057472 103375          BCC     3$
7064 057474 120001          CMPB   R0,R1       ; ECHO CORRECT ??
7065 057476 001404          BEQ    4$           ; BR IF SO.
7066 057500 104401 060526  TYPE    ,EKOERR     ; ELSE, "ASCII ECHO ERROR"...
7067 057504 000137 057134  JMP     RELOAD      ; ...AND RETRY.
7068 057510 077325          4$:    SOB    R3,1$
7069 057512 012600          MOV    (SP)+,R0    ; XMIT TERMINATOR.
7070 057514 004737 057636  CALL    SBCOUT
7071 057520 000207          RETURN           ; AND RETURN.
7072
7073          ; SUBROUTINE TO SEND "BREAK" TO THE SBC.

```

```

7074
7075 057522 010546          ; BRK:  MOV    R5,-(SP)      ;SAVE OLD R5 VALUE ON STACK      ;HE808
7076 057524 013705 056730  MOV    SLU2,R5
7077 057530 116500 000002  MOV    2(R5),R0      ; DUMMY READ (CLEAR RECEIVER).    ;FX9JC
7078 057534 052765 000001 000004  BIS    #1,4(R5)      ; SET "BREAK"...
7079 057542 112765 000060 000006  MOV    #'0,6(R5)    ; TRANSMIT TWO CHARACTERS FOR TIMING. ;FX8JC
7080 057550 105765 000004      11$:  TSTB  4(R5)        ; WAIT FOR READY.                  ;FX8JC
7081 057554 100375      BPL    11$           ;
7082 057556 112765 000060 000006  MOV    #'0,6(R5)    ; SEND SECOND.                    ;FX8JC
7083 057564 105765 000004      22$:  TSTB  4(R5)        ; WAIT FOR READY.                  ;FX8JC
7084 057570 100375      BPL    22$           ;
7085 057572 005001      CLR    R1            ; WAIT VALUE                        ;FX8JC
7086 057574 105715      1$:  TSTB  (R5)         ; \
7087 057576 100412      BMI    3$           ; >...AND WAIT FOR A RESPONSE.    ;
7088 057600 077103      SOB   R1,1$        ; /
7097 057602 104401 060514  TYPE   ,HUNG        ; NO LUCK, TELL THE MAN !!
7098 057606 026627 000002 057070  CMP    2(SP),#LOAD  ; CALLED FROM LOAD ??
7099 057614 103403      BLO   3$           ; JUST RETURN IF NOT...
7100 057616 012766 057134 000002  MOV    #RELOAD,2(SP) ; ...ELSE, RETRY.
7101 057624 042765 000001 000004  3$:  BIC    #1,4(R5)    ; ENSURE "BREAK" IS OFF.
7102 057632 012605      MOV    (SP)+,R5     ; RESTORE R5                        ;;HE808
7103 057634 000207      RETURN
7104
7105          ; CHARACTER OUTPUT TO TARGET SBC OR TTY:.
7106
7107 057636 010546          SBCOUT: MOV    R5,-(SP)      ;SAVE OLD R5 VALUE                ;HE808
7108 057640 013705 056730  MOV    SLU2,R5
7109 057644 000403      BR    TTYZ
7110 057646 010546          TTYOUT: MOV    R5,-(SP)      ;SAVE OLD R5 VALUE                ;HE808
7111 057650 013705 056726  MOV    SLU1,R5
7112 057654 105765 000004  TTYZ:  TSTB  4(R5)
7113 057660 100375      BPL    TTYZ
7114 057662 110065 000006  MOV    R0,6(R5)     ; OUTPUT FROM R0...
7115 057666 012605      MOV    (SP)+,R5     ; ...RESTORE R5...                ;HE808
7116 057670 000207      RETURN              ; ...AND RETURN.
7117
7118          ; CHARACTER INPUT FROM TARGET SBC OR KBD:.
7119
7120 057672 010546          SBCIN:  MOV    R5,-(SP)      ;SAVE OLD R5 ON STACK             ;HE808
7121 057674 013705 056730  MOV    SLU2,R5
7122 057700 000403      BR    TTYX
7123 057702 010546          TTYIN:  MOV    R5,-(SP)      ;HE808
7124 057704 013705 056726  MOV    SLU1,R5
7125 057710 105715  TTYX:  TSTB  (R5)
7126 057712 100007      BPL    1$           ; NO CHAR ("C" IS CLEAR).
7127 057714 105715      TSTB  (R5)         ; TEST IT AGAIN IN CASE IT SET ON BREAK.;FXBJC
7128 057716 100005      BPL    1$           ; YES ... WAS SET ON A BREAK, SO RETURN.;FXBJC
7129 057720 116500 000002  MOV    2(R5),R0     ; VALID CHAR SO INPUT TO R0...
7130 057724 042700 177600  BIC    #+C177,R0   ; ...STRIP...
7131 057730 000261      SEC
7132 057732 012605      1$:  MOV    (SP)+,R5     ; ...SET "C"
7133 057734 000207      RETURN              ; RESTORE STACK                    ;HE808
7134
7135          ; SUBROUTINES TO WAIT FOR SPECIFIC RESPONSE FROM SBC.
7136
7137 057736 012746 000100  WAIT00: MOV    #'@,-(SP)    ; WAIT FOR ODT PROMPT <100>.
7138 057742 000402      402

```

```

7139 057744 012746 000040      WAIT40: MOV      #40,-(SP)      ; WAIT FOR SPACE CHAR <040>.
7140 057750 000402              402
7141 057752 012746 000120      WAITP: MOV      #'P,-(SP)      ; WAIT FOR P(ROCEED) ECHO <120>.
7142 057756 004737 057672      1$: CALL      SBCIN
7143 057762 103375              BCC      1$
7144 057764 120016              CMPB     RO,(SP)
7145 057766 001373              BNE      1$
7146 057770 005726              2$: TST      (SP)+
7147 057772 000207              RETURN
7148
7149      ; SUBROUTINE TO UPDATE ARBITERS COPY OF THE APT MAILBOX.
7150      ; CALLED IN LINK MODE IF/WHEN <+Z> IS RECEIVED FROM SBC.
7151
7152      ;+
7153      ; UPAMB REVISED FOR KXTFX9 TO INCORPORATED CHECKSUMS.
7154      ;-
7155 057774 010546      UPAMB: MOV      R5,-(SP)      ;SAVE R5
7156 057776 013705 056730      MOV      SLU2,R5
7157 060002 105765 000004      TSTB     4(R5)
7158 060006 100372              BPL      UPAMB
7159 060010 112765 000032 000006      MOV      #32,6(R5)
7160 060016 012700 060214      MOV      #TMPMAI,RO
7161 060022 012701 000021      MOV      #17.,R1
7162 060026 005037 060240      CLR      UPERR
7163 060032 005002      1$: CLR      R2
7164 060034 105715      2$: TSTB     (R5)
7165 060036 100402              BMI      3$
7166 060040 077203              SOB      R2,2$
7167 060042 000451              BR       7$
7168 060044 016502 000002      3$: MOV      2(R5),R2
7169 060050 100002              BPL      4$
7170 060052 005237 060240      INC      UPERR
7171 060056 110220      4$: MOV      R2,(RO)+
7172 060060 077114              SOB      R1,1$
7173      ; NOW CHECK FOR ERROR AND COMPUTE THE CHECKSUM.
7174 060062 005737 060240      TST      UPERR
7175 060066 001037              BNE      7$
7176 060070 012701 000020      MOV      #16.,R1
7177 060074 012704 060214      MOV      #TMPMAI,R4
7178 060100 005037 060236      CLR      ARBSUM
7179 060104 112403      5$: MOV      (R4)+,R3
7180 060106 042703 177400      BIC      #+C377,R3
7181 060112 060337 060236      ADD      R3,ARBSUM
7182 060116 106337 060236      ASLB     ARBSUM
7183 060122 005537 060236      ADC      ARBSUM
7184 060126 077112              SOB      R1,5$
7185 060130 123737 060236 060234      CMPB     ARBSUM,RCDSUM
7186 060136 001013              BNE      7$
7187 060140 012700 060214      MOV      #TMPMAI,RO
7188 060144 012701 001166      MOV      #MAIL,R1
7189 060150 012702 000010      MOV      #8.,R2
7190 060154 012021      6$: MOV      (RO)+,(R1)+
7191 060156 077202              SOB      R2,6$
7192 060160 012700 000006      MOV      #ACK,RO
7193 060164 000402              BR       8$
7194 060166 012700 000025      7$: MOV      #NACK,RO
7195 060172 005037 060240      8$: CLR      UPERR
    
```

```

7196 060176 105765 000004      9$:  TSTB    4(R5)      ; WAIT FOR TRANSMIT READY.      ;FX9JC
7197 060202 100375              BPL      9$          ;                               ;FX9JC
7198 060204 110065 000006      MOVB    R0,6(R5)    ; SEND ACK OR NACK.           ;FX9JC
7199 060210 012605              MOV     (SP)+,R5    ; RESTORE R5                   ;
7200 060212 000207              RETURN                   ; AND RETURN TO LINK ROUTINE.  ;FX9JC
7201
7202 060214              TMPMAI: .BLKW    8.    ; SCRATCH PAD MAILBOX.        ;FX9JC
7203 060234 000000              RCDSUM: .WORD    0    ; LOW BYTE IS RECEIVED CHECKSUM. ;FX9JC
7204 060236 000000              ARBSUM: .WORD    0    ; ACCUMULATOR FOR COMPUTING CHECKSUM. ;FX9JC
7205 060240 000000              UPERR:  .WORD    0    ; ERROR FLAG FOR OVERRUNS.     ;FX9JC

```

```

7206
7207 ; SUBROUTINES TO SET MMU, MAP-RAM, PSW, AND PC
7208 ; IN PREPARATION FOR LOADING/STARTING THE SBC.
7209 ;
7210

```

```

7211 060242 012701 060370      .ENABL LSB
SETREGS: MOV     #0MM,R1      ; MMU...
7212 060246 005002              CLR     R2            ; ...DISABLED.
7213 060250 004737 060310      CALL    1$
7214 060254 012701 060402      MOV     #OKX,R1       ; PRIMARY CSR...
7215 060260 005002              CLR     R2            ; ...DISABLED (MAP-RAM OFF).
7216 060262 000412              BR      1$
7217 060264 012701 060414      SETPSW: MOV     #OPS,R1 ; PSW...
7218 060270 012702 000340      MOV     #340,R2       ; ...PRI 7.
7219 060274 000405              BR      1$
7220 060276 012701 060420      SETPC:  MOV     #OPC,R1 ; PC (SET IN R2 BY CALLER).
7221 060302 000402              BR      1$
7222 060304 012701 060424      SETSP:  MOV     #OSP,R1 ; SP (SET IN R2 BY CALLER)

```

```

7223
7224 ;*****
7225 ; INSURE THAT "ODT" ECHOES THE CHARACTERS CORRECTLY. ;FX8JC

```

```

7226 060310 112103              1$:  MOVB    (R1)+,R3   ; SAVE BYTE IN R3 FOR COMPARE.  ;FX8JC
7227 060312 001414              BEQ     3$            ; IF ZERO, STRING IS DONE.     ;FX8JC
7228 060314 110300              MOVB    R3,R0         ; BYTE TO R0 FOR THE CALL.     ;FX8JC
7229 060316 004737 057636      CALL    SBCOUT        ;                               ;FX8JC
7230 060322 004737 057672      2$:  CALL    SBCIN        ;                               ;FX8JC
7231 060326 103375              BCC     2$            ;                               ;FX8JC
7232 060330 120003              CMPB   R0,R3         ; ECHO CORRECT?                ;FX8JC
7233 060332 001766              BEQ     1$            ; YES, CONTINUE ...           ;FX8JC
7234 060334 104401 060526      TYPE   ,EKOERR       ; ELSE, "ASCII ECHO ERROR" ... ;FX8JC
7235 060340 000137 057134      JMP     RELOAD        ; ... AND GO TRY IT ALL AGAIN. ;FX8JC
7236 060344 004737 057672      3$:  CALL    SBCIN        ; OTHERWISE, WAIT FOR <SP> ... ;FX8JC
7237 060350 120027 000040      CMPB   R0,#40        ;                               ;FX8JC
7238 060354 001373              BNE     3$            ;                               ;FX8JC
7239 060356 004737 057422      CALL   DATACR        ; ... AND SET VALUE FROM R2.   ;FX8JC
7240 060362 004737 057736      4$:  CALL   WAIT00       ; WAIT FOR @...                ;FX8JC
7241 060366 000207              RETURN                   ; ... AND RETURN.             ;FX8JC

```

```

7242
7243 .DSABL LSB
7244 ;
7245 ; TEXT FOR MANIPULATING REGISTERS UNDER "ODT".
7246 ;

```

```

7247 060370      067      067      067      067      OMM:  .ASCIZ '77777572/' ; OPEN MMU SRO F/J.
7248 060402      067      067      067      067      OKX:  .ASCIZ '77777530/' ; OPEN K2CSR0 F/J.
7249 060414      122      123      057      OPS:  .ASCIZ 'RS/' ; OPEN PSW.
7250 060420      122      067      057      OPC:  .ASCIZ 'R7/' ; OPEN PC.
7251 060424      122      066      057      OSP:  .ASCIZ 'R6/' ; OPEN SP.

```

```

7252 ; *****

```

15

7253				:	
7254				:	DIRECTIONS AND STATUS TEXT.
7255				:	
7256	060430	015	012	040	WAIT1: .ASCIZ <CR><LF>' WAIT...'
7257	060444	113	056	007	KMARK: .ASCIZ 'K.'<7>
7258	060450	056	056	111	LOADED: .ASCII '..IOP LOADED'<7><CRLF>
7259	060466	200	040	040	.ASCIZ <CRLF>' USE <ESC> TO HALT'<CRLF>
7260	060514	111	117	120	HUNG: .ASCIZ 'IOP HUNG'<CRLF>
7261	060526	101	123	103	EKOERR: .ASCIZ 'ASCII ECHO ERROR'<CRLF>
7262					.EVEN


```

7264 ;*****
7265 ; IMAGE LOADER FOR THE TARGET SBC.
7266 ; THE ASCII (00T) LOADER STUFFS THIS ROUTINE INTO THE SBC
7267 ; AT THE TOP OF 16K, AND STARTS IT RUNNING AT PRI 7.
7268 ;
7269 ;.DSABL AMA ; MAKE IT RELATIVE PIC.
7270 ;
7271 060550 010706 ;$LOAD: MOV PC,SP
7272 060552 005746 TST -(SP) ; SET A STACK...
7273 060554 C10600 MOV SP,RO
7274 060556 005040 1$: CLR -(RO) ;...AND CLEAR FROM HERE DOWN.
7275 060560 005700 TST RO
7276 060562 001375 BNE 1$
7277 ;
7278 ; DMA LOADER.
7279 ;
7280 ;$DLOAD: MOV $$SLOAD--6,RO ; \
7281 060564 012700 000144 ADD PC,RO ; > SET A TRAP CATCHER...
7282 060570 060700 MOV RO,@#4 ; / ...IN CASE CHIP IS DEAD.
7283 060572 010037 000004 MOV #PR7,@#6
7284 060576 012737 000340 000006 MOV #340,@#K2CSRC ;IS IOP IN STANDALONE?
7285 060604 032737 000340 177524 BIT $SLOAD ;...THEN JUST DO SERIAL LOAD
7286 060612 001451 BEQ @#CMDR ; CHIP RESET.
7287 060614 005037 174454 CLR @#115 @#MMR ; SET MASTER MODE...
7288 060620 012737 000115 174470 MOV #40!@#1,@#CMDR ;...CLEAR IE'S...
7289 060626 012737 000074 174454 CLR @#CHA1H
7290 060634 005037 174446 MOV #2!--6,RO ; \
7291 060640 012700 000050 ADD PC,RO ; > SET CHAIN ADDRESS.
7292 060644 060700 MOV RO,@#CHA1L ; /
7293 060646 010037 174442 MOV #240,@#CMDR ; START 'EM UP...
7294 060652 012737 000240 174454 MOV #IP,@#STAT1 ;...AND WAIT FOR DONE.
7295 060660 032737 020000 174456 1$: BIT 1$
7296 060666 001774 BEQ @#STAT1,RO ; SAVE STATUS...
7297 060670 013700 174456 MOV @#CMDR ;...AND RESET CHIP.
7298 060674 005037 174454 CLR @TC,RO ; TERMINAL COUNT ??
7299 060700 032700 000001 BIT $SLOAD ; NO, DMA FAILED.
7300 060704 001414 BEQ @33,@#177566 ; YES, SEND DMA DONE <+(>...
7301 060706 112737 000033 177566 MOVB $START ;...AND THAT'S ALL THERE IS TO IT !!
7302 060714 000431 BR
7303 ;
7304 060716 001602 2$: 1602 ; CHAIN LOAD ARA, ARB, WC, AND CH MODE...
7305 060720 100000 000000 100000, 0 ;...SRC -- Q-BUS ZERO...
7306 060724 000000 000000 0, 0 ;...DST -- LOCAL ZERO...
7307 060730 024644 <LASTAD+2>/2 ;...RANGE -- 0-LASTAD...
7308 060732 000030 001340 30, 1340 ;...AND START (SOFT-REQ).

```

```

7310 ;
7311 ; SERIAL (BYTE) LOADER.
7312 ;
7313 ; EFFECTIVE LOAD RATE IS 480. WORDS/SEC AT 9600 BAUD.
7314 ;
7315 060736 005000 ;$LOAD: CLR R0 ; SERIAL LOAD FROM 0 TO LASTAD.
7316 060740 113701 177562 MOVB @#177562,R1 ; DUMMY READ.
7317 060744 112737 000034 177566 MOVB #34,@#177566 ; SEND READY MARK <+>.
7318 060752 105737 177560 1$: TSTB @#177560 ; NOW, WAIT FOR...
7319 060756 100375 BPL 1$
7320 060760 113720 177562 MOVB @#177562,(R0)+ ;...AND STORE EACH BYTE...
7321 060764 020027 051507 CMP R0,#LASTAD+1
7322 060770 101770 BLOS 1$ ;... 'TIL DONE.
7323 060772 112737 000034 177566 MOVB #34,@#177566 ; SEND DONE MARK <+>.
7324 ;
7325 ; NOW ONE WAY OR T'OTHER, THAT SUCKER'S LOADED.
7326 ; DELAY A SKOSH AND START IT RUNNING.
7327 ;
7328 061000 005000 ;$START: CLR R0
7329 061002 077001 SOB R0 ; DELAY.
7330 061004 013737 000206 000202 MOV @#206,@#202 ; DELETE "REMOTE" START ADDRESS.
7331 061012 012707 000200 MOV #200,PC ; S T A R T . . .
7332 061016 000000 1$: HALT ; . . . O R H A L T
7333 061020 000776 BR 1$
7334 .EVEN ;FX8JC
7335 $LSIZE= <.-$LOAD>/2 ; IMAGE LOADER SIZE (IN WORDS)...
7336 $LSTRT= 100000-<.-$LOAD> ;...AND LOCAL STARTING ADDRESS.
7337 $LSTRT= $LSTRT-110 ; *** LOWER TO CLEAR T'S ODT STACK ***
7338 ;*****
7339 061022 BUFR1: .BLKW 2048.
7340 071022 BUFR2: .BLKW 2048.
7341 000001 .END
    
```

ABASE = 000000	APRIOR= 000000	BIT13 = 020000	CH2STA 021512	DC.RCV 011750
ABORT = 002506	APTCU= 000040	BIT14 = 040000	CIE = 100000	DC.XMT 011526
ACDW1 = 000000	APTENV= 000001	BIT15 = 100000	CKBYTE 014032	DDISP = 177570
ACDW2 = 000000	APTSIZ= 000200	BIT2 = 000004	CKSUM 010712	DECIMA 050534
ACK = 000006	APTSP0= 000100	BIT3 = 000010	CKSWR = 104410	DIN = 000400
ACKEN = 020000	ARBDIS 036330	BIT4 = 000020	CKTBL 013772	DISMIS 002542
ACPUOP= 000000	ARBER = 000002	BIT5 = 000040	CLIE1 = 000060	DISPLA 001142
AD = 000020	ARBTR 052160	BIT6 = 000100	CLIE2 = 000061	DISPRE 000174
ADDRS 045713	ARBLMG 035776	BIT7 = 000200	CLIP1 = 000054	DMAC1 020102
ADDW0 = 000000	ARBSUM 060236	BIT8 = 000400	CLIP2 = 000055	DMAC2 020133
ADDW1 = 000000	ARBSUP 016703	BIT9 = 001000	CLK1 046303	DMAC3 020170
ADDW10= 000000	ARBTPR 056234	BLKER 032740	CLK2 046337	DMAC4 020221
ADDW11= 000000	ARBTYP 052450	BOP = 000020	CLK3 046405	DMAC5 020252
ADDW12= 000000	ARB374 054726	BPTVEC= 000014	CLK4 046442	DMAC6 020303
ADDW13= 000000	ARB400 055164	BRK 057522	CM 032114	DMAC7 020336
ADDW14= 000000	ARB404 054530	BRQ = 040000	CMC = 040000	DMAEND 015742
ADDW15= 000000	AR.404 054672	BRGEN = 010000	CMDR = 174454	DMAV 000214
ADDW2 = 000000	ASWREG= 000000	BUFR1 061022	CNFGER 016604	DMAV1 000214
ADDW3 = 000000	ATESTN= 000000	BUFR2 071022	CNFG01 016536	DMAV2 000220
ADDW4 = 000000	AUNIT = 000000	BUSHOG= 000040	CNTNU 032006	DMA1 020042
ADDW5 = 000000	AUSWR = 000000	BUSREL= 000100	CNTZ 044312	DMA2 020413
ADDW6 = 000000	AVECT1= 000000	BUSTO 011030	COMMA 046136	DMA3 020451
ADDW7 = 000000	AVECT2= 000000	BXMTD 046051	COP = 000200	DMA4 020513
ADDW8 = 000000	A\$MAIL= 104416	BYT = 000000	COP1 = 174462	DMA5 020562
ADDW9 = 000000	A\$MAI 044046	BYTES 037576	COP2 = 174460	DOUT = 000600
ADEVCT= 000000	A.SET 022500	B1 = 000040	COUNT 001364	DOWN = 000010
ADEVM = 000000	A16 = 040000	B2 = 000100	CPEREG= 177766	DPREND 032122
ADRBIT 016760	A17 = 100000	B4 = 000140	CPUER 003556	DPRV12 000134
ADRSUP 053317	A4800 022476	CA = 004000	CPUTST= 000000	OPRV4 000120
ADR16 053300	BADBAL 016572	CANT 012436	CR = 000015	DPRV8 000124
ADR18 053305	BADBIT 016654	CARA = 001000	CRLF = 000200	DPR1 046500
ADR22 053312	BARA = 000100	CARAH1 037042	CSRADR 004715	DPR10 047144
AD1621 017036	BARB = 000040	CARAH1= 174432	CSRDAT 004731	DPR11 027371
AENV = 000000	BAUDSE 022244	CARALO 037044	CSRE 046254	DPR2 046533
AENVM = 000000	BDQXFR 020601	CARAL1= 174412	CSRER 004662	DPR3 046566
AFATAL= 000000	BEVNT 000100	CARB = 000400	CTC = 100000	DPR4 046633
AGAIN 003064	BFLAG 001352	CARBH1= 174422	CTCC = 000040	DPR5 046700
AI 026050	BI 026064	CARBL1= 174402	CTCS = 000024	DPR6 046771
AI.B0 = 164377	BIACKV= 000130	CCR = 177746	CTMS = 000070	DPR7 046774
ALLCTR 001374	BIEN = 002000	CEOP = 020000	CTRL = 000000	DPR8 047042
AMADR1= 000000	BIT 016776	CHA = 000004	CTTC = 000054	DPR9 047107
AMADR2= 000000	BITRAP 003042	CHAD = 000001	CTV = 000010	DR = 000000
AMADR3= 000000	BITO = 000001	CHAIN1 014042	CURBLK 055302	DSTART 000204
AMADR4= 000000	BIT00 = 000001	CHAIN2 014064	C.CLR 022550	DSWR = 177570
AMAMS1= 000000	BIT01 = 000002	CHAIN3 014104	C1FLAG 001404	DTCEXP 014204
AMAMS2= 000000	BIT02 = 000004	CHAIN4 014124	C2FLAG 001406	DTCTO 005631
AMAMS3= 000000	BIT03 = 000010	CHAINS 014144	DADR 045767	DTCXM1 015214
AMAMS4= 000000	BIT04 = 000020	CHAIN6 014164	DATACR 057422	DTCXM2 015264
AMSGAD= 000000	BIT05 = 000040	CHAIH = 174446	DATALF 057414	DTCXM3 015334
AMSGLG= 000000	BIT06 = 000100	CHAIL = 174442	DATIO = 000200	DTCXM4 015403
AMSGTY= 000000	BIT07 = 000200	CHAZH = 174444	DATPR 005046	DTCXM5 015612
AMTYP1= 000000	BIT08 = 000400	CHAZL = 174440	DCF11 052524	DTCXM6 015651
AMTYP2= 000000	BIT09 = 001000	CHB = 0000005	DCJDM 046140	DTCXM7 015700
AMTYP3= 000000	BIT1 = 000002	CHEKSU 010514	DCJ11 052510	DTCXM8 015452
AMTYP4= 000000	BIT10 = 002000	CHKSUM 044310	DCOUNT 001372	DTCXM9 015532
AQ.BI = 165400	BIT11 = 004000	CHNG10 015014	DCT11 052534	DTCX01 014754
APASS = 000000	BIT12 = 010000	CHX = 0000005	DC.DAT 012222	D100 037564

D200	037562	F11	= 000003	IRQ7	= 000020	KIPDR4=	172310	MODE	= 000002
D50	037566	F11#	050766	IRX	023742	KIPDR5=	172312	NAC	= 010000
EF1	045206	GADDRS	045704	IR12	031142	KIPDR6=	172314	NACK	= 000025
EF10	045556	GAHI	037536	IR4	031126	KIPDR7=	172316	NATIVE	037256
EF11	045610	GALO	037540	IR8	031134	KMARK	060444	NBLKS	055274
EF12	045646	GA22	045256	ISEX	023754	KMEM	053353	NECC	= 047536
EF2	045210	GD#IOP	021014	ITC	= 001000	KXINIT	002250	NECCM	047531
EF3	045244	GO	= 000001	ITX	023716	K2CSR=	177520	NECCM	047551
EF4	045326	GTSWR	= 104407	IV	= 000004	K2CSR8=	177522	NECM	= 047541
EF5	045360	HALTRP	002664	IXIT	024024	K2CSR9=	177524	NECMH	047721
EF6	045360	HD	= 000000	I#SWR	= 104414	K2CSR0=	177530	NECV	000070
EF7	045442	HIPAR	007472	I#SWR	043076	K2CSR1=	177526	NEC.AS	022560
EF8	045466	HIQB	= 175400	JMP\$AY	016726	K2CSR2=	177534	NEC.DM	023264
EF9	045520	HLTFLG	001350	J11	= 000005	K2CSR3=	177536	NEC.EN	024536
EH1	044764	HM	= 000010	J11REG	003230	K2CSR4=	177540	NEC.EX	024546
EH2	044774	HOG	= 000000	J11#	050764	K2CSR5=	003620	NEC.I	023674
EH3	045006	HOLD	= 000020	KDJ#	050762	K2CIO	= 040000	NEC.MD	024034
EKOERR	060526	HT	= 000011	KDJ11A	052477	K2MEM	= 000000	NEC.SY	022572
ELOG	051034	HUNG	060514	KDPAR0=	172360	K2QIR	= 177532	NECC01	022610
ELOGN	= 000225	HZ800	022474	KDPAR1=	172362	K2RAM	005652	NEC1	047406
EMADR	044746	IDERR	031156	KDPAR2=	172364	LA	032106	NEC10	050165
EMPRE	044742	IEOP	= 000200	KDPAR3=	172366	LASTAD	051506	NEC2	047455
EMTVEC	= 000030	IEXPD	046007	KDPAR4=	172370	LA16	045230	NEC3	047574
ENDIRQ	031174	IMC	= 000400	KDPAR5=	172372	LA16M2	045222	NEC3A	047625
ENDLMG	037160	INIT	002046	KDPAR6=	172374	LCLMEM	037050	NEC4	= 047356
ENDPIO	026046	INIT0	017016	KDPAR7=	172376	LDING	052136	NEC5	047655
ENDRAM	011232	INTLV	= 000140	KDPDR0=	172320	LF	= 000012	NEC6	047746
ENDSHM	033056	INTVEC	036434	KDPDR1=	172322	LINK	056770	NEC7	050004
ENDSLU	012504	INT120	032772	KDPDR2=	172324	LMGH1	036436	NEC8	050040
EOP	= 000002	INT134	032574	KDPDR3=	172326	LMGH2	036503	NEC9	050110
EOPER	020661	IOPADR	052002	KDPDR4=	172330	LMGH3	036544	NEXT1	052032
EOPTBL	014654	IOPCNT	050746	KDPDR5=	172332	LMGH4	036571	NLZ	= 000000
ERHAN	044630	IOPEND	021754	KDPDR6=	172334	LMGH5	036631	NMIEND	002732
ERRCNT	001370	IOPVEX	000410	KDPDR7=	172336	LN	= 000006	NMITRP	002544
ERROR	= 104000	IOPV02	055656	KGIOPI	053776	LOAD	057070	NORESP	052006
FRRVEC	= 000004	IOPV03	055662	KGNR	021756	LOADED	060450	NOSHMR	054475
ERXIT	044760	IOPV04	055666	KG#T01	022054	LOCTPR	004740	NOSIG	045206
ESADR	044752	IOPV05	055672	KG#T02	022074	LOOPIN	001376	NOTST	056464
ESUMRY	045016	IOPV06	055676	KG#T03	022114	LOQB	= 160000	NXT	052124
ESUM1	045152	IOPV07	055702	KG#T04	056466	LOWLIM	050752	NXTIOP	000254
EXPD	045725	IOPV08	055706	KG#T05	056472	LOWPAR	007316	OFFSBC	056652
EXPDAT	001360	IOPV09	055712	KG#T06	056474	LQ2HI	= 015400	OKX	060402
EXTMEM	= 000000	IOPV10	055716	KG#T07	056514	LSI11	052545	OMM	060370
EX400	055234	IOPV11	055722	PG.ER	022016	LZS	050552	OOPS	017004
FALCON	000140	IOPV12	055726	KG.IOP	050744	L11#	050772	OPC	060420
FATALB	010626	IOPV13	055732	KIPAR0=	172340	MAXBLK	050754	OPNADR	057406
FATALR	010704	IOPV14	055736	KIPAR1=	172342	MAXWRT	032572	OPS	060414
FATALW	010634	IOPV15	055742	KIPAR2=	172344	MAX.WT	055306	OSP	060424
FATAL1	010640	IOP#TR	= 104417	KIPAR3=	172346	MCC	= 000002	OSRO	046067
FIN22	017106	IOP#T	055746	KIPAR4=	172350	MIC	= 000000	OSROA	046105
FIRST	055304	IOP.ID	050742	KIPAR5=	172352	MIR	= 174470	OSROB	046116
FLAG	001366	IOP.UT	050740	KIPAR6=	172354	MIRDEF	= 000135	OSR1	046075
FLIP	= 000020	IOTVEC	= 000020	KIPAR7=	172356	MIR0	= 177572	PACONV	011120
FLT0	037612	IP	= 020000	KIPDR0=	172300	MIR3	= 172516	PACS	= 000020
FLT1	037602	IRQ4	= 000002	KIPDR1=	172302	MIRU	001462	PADB	= 000032
FLT10	037600	IRQ5	= 000004	KIPDR2=	172304	MIRUTST	= 000000	PAHS	= 000102
FRESH	052000	IRQ6	= 000010	KIPDR3=	172306	MIRVEC	= 000250	PAMS	= 000100

PAR	001576	PSW	= 177776	RDLIN	= 104412	SETSP	060304	SSRCH2	= 000103
PARCHK	006752	PWRFAI	= 000000	RDOCT	= 104413	SHLOCK	035564	STACK	= 001100
PARDEL	006750	PWRV	000024	READO	017060	SHMAMT	054420	START	001622
PARE	010733	PWRVEC	= 000024	RECD	045735	SHMXFR	034762	STAT	= 177774
PARERR	011174	QA	032102	RECDAT	001362	SHMXIT	035226	STAT1	= 174456
PATMSK	= 000010	QADR	037114	RELOAD	057134	SHM1	033076	STAT2	= 174454
PAV	= 000004	QBCHN2	017670	REMBLK	055300	SHM2	033144	STBB	= 000005
PBCS	= 000022	QBCHN3	017716	REMOTE	056702	SHOSYS	055310	STBOT	= 001000
PBDB	= 000034	QBCHN4	017744	REOP	= 002000	SHRMEM	054202	STF	= 046175
PBHS	= 000122	QBFIN	021000	RESTAR	001622	SIPARO	= 172240	STFUNL	= 000014
PBMS	= 000120	QBEGO	017646	RESVEC	= 000010	SIPAR1	= 172242	STKLMT	= 177774
PBV	= 000006	QBETST	017106	RFLAG	C01344	SIPAR2	= 172244	STQBIO	037046
PCDB	= 000036	QBETO1	017770	RL	= 000004	SIPAR3	= 172246	STWW	= 000004
PCDD	= 000014	QBETO2	020006	RMC	= 004000	SIPAR4	= 172250	SUMCHK	037274
PDPDTC	036764	QBETO3	020024	RPRI	= 000002	SIPAR5	= 172252	SVPC	= 044606
PDR	001562	QBEX	054070	RQIO	031722	SIPAR6	= 172254	SWM	= 000002
PEVNT	000104	QBEXER	017456	RR	= 000002	SIPAR7	= 172256	SWR	001140
PIOAV	000160	QBE1	= 170000	RSTART	000200	SIPDR0	= 172200	SWREG	000176
PIOBV	000164	QBE18	050760	RSTRAP	002614	SIPDR1	= 172202	SWROPT	043152
PIOD	025430	QBE2	= 170020	RTC	= 010000	SIPDR2	= 172204	SWRQ	= 000020
PIOD1	025470	QBE22	050756	RVEC	= 000000	SIPDR3	= 172206	SWO	= 000001
PIOD2	025560	QBLOAD	037142	R6	= 000006	SIPDR4	= 172210	SW00	= 000001
PIOD3	025616	QBMEM	= 100000	R7	= 000007	SIPDR5	= 172212	SW01	= 000002
PIOT	025072	QBSTRT	055276	SACKER	020716	SIPDR6	= 172214	SW02	= 000004
PIOT1	025166	QBUSIO	= 140000	SACKTO	020627	SIPDR7	= 172216	SW03	= 000010
PIO1	050232	QDPR	027750	SADR	045747	SKP1	= 000401	SW04	= 000020
PIO2	050257	QHIT	035252	SAVSP1	001400	SKP2	= 000402	SW05	= 000040
PIO3	050326	QIEND	027114	SAVSP2	001402	SKP3	= 000403	SW06	= 000100
PIO4	050342	QIOC	032100	SAYYES	056654	SLIR	047217	SW07	= 000200
PIO5	050356	QIOCC	032120	SBCIN	057672	SLIX	047211	SW08	= 000400
PIO6	050416	QIR	= 177532	SBCOUT	057636	SLI2	047225	SW09	= 001000
PIO7	050503	QIRER	004700	SBM	= 000003	SLI3	047265	SW1	= 000002
PIO8	= 047356	QIRV	000144	SCCCH1	= 000240	SLI4	047314	SW10	= 002000
PIRQ	= 177772	QIRV1	000150	SCCCH2	= 000241	SLI5	047356	SW11	= 004000
PIRQV	000240	QIRV2	000154	SCOPE	= 000004	SLOC00	001354	SW12	= 010000
PIRQVE	= 000240	QIT1	026356	SDECIM	050542	SLOC01	001356	SW13	= 020000
PIRTRP	002406	QI1	027132	SDPAR0	= 172260	SLUADR	056732	SW14	= 040000
PIRVEC	000240	QI2	027157	SDPAR1	= 172262	SLU1	056726	SW15	= 100000
PITN	050311	QI3	027210	SDPAR2	= 172264	SLU1T0	005573	SW2	= 000004
PITX	= 050323	QI4	027257	SDPAR3	= 172266	SLU2	056730	SW3	= 000010
POINT	052004	QI5	027323	SDPAR4	= 172270	SLU2T0	005612	SW4	= 000020
PORTAB	050464	QREQ	056642	SDPAR5	= 172272	SNGL	= 000000	SW5	= 000040
PORTBA	050445	QREQ1	056630	SDPAR6	= 172274	SNIFFR	053412	SW6	= 000100
PRAME	010762	QREQ2	056534	SDPAR7	= 172276	SNM	= 000000	SW7	= 000200
PRGSIZ	= 000123	QHIT	056306	SDPDR0	= 172220	SOURCE	001410	SW8	= 000400
PROMCK	037412	Q18ADR	037052	SDPDR1	= 172222	SPACE	046134	SW9	= 001000
PROMLO	010450	Q18BEX	017442	SDPDR2	= 172224	SPAC2	046133	YSER	003571
PRTYV	000114	Q1822	017412	SDPDR3	= 172226	SPAC4	046131	S.SET	022520
PRO	= 000000	Q22ADR	037076	SDPDR4	= 172230	SPDMSK	056764	TAB09	045203
PR1	= 000040	Q22BEX	017424	SDPDR5	= 172232	SPDSET	056766	TBIACK	000130
PR2	= 000100	RAME	011006	SDPDR6	= 172234	SRCH88	= 000017	TBITVE	= 000014
PR3	= 000140	RAMTYT	011063	SDPDR7	= 172236	SRCHMM	= 000012	TBUF	= 000006
PR4	= 000200	RBUF	= 000002	SETIE1	= 000062	SRO	= 177572	TC	= 000001
PR5	= 000240	RCDSUM	060234	SETIE2	= 000063	SR1	= 177574	TCS	= 000004
PR6	= 000300	RCS	= 000000	SETPC	060276	SR2	= 177576	TDMA1	013140
PR7	= 000340	RDB	= 177776	SETPSW	060264	SR3	= 172516	TDMA2	013234
PS	= 177776	RDCMR	= 104411	SETREG	060242	SSRCH1	= 000102	TDPR1	027522

TDP2	027662	TST6	013026	UDPDR7=	177636	\$CID	001336	\$ERTTL	001112
TIPV1	030124	TST7	015756	UIPAR0=	177640	\$CKSMR	041610	\$ESCAP	001160
TIPV2	030470	TTYIN	057702	UIPAR1=	177642	\$CLK	001332	\$ETABL	001206
TIPV3	030720	TTYOUT	057646	UIPAR2=	177644	\$CMTAG	001100	\$ETEND	001312
TKVEC	000060	TTYX	057710	UIPAR3=	177646	\$CM3	000000	\$FATAL	001170
TMPMAI	060214	TTYZ	057654	UIPAR4=	177650	\$CNTLG	042345	\$FFLG	043000
TNLEDS	037542	TVEC	000004	UIPAR5=	177652	\$CNTLU	042340	\$FILLC	001156
TOEH	011110	TYPBN	104406	UIPAR6=	177654	\$CPTYP	001312	\$FILLS	001155
TOPMEM	005650	TYPDS	104405	UIPAR7=	177656	\$CPUOP	001214	\$GDADR	001120
TPRCHK	021622	TYPE	104401	UIPDR0=	177600	\$CRLF	001163	\$GDDAT	001124
TPRDER	005554	TYPOC	104402	UIPDR1=	177602	\$CSR	001314	\$GET42	040106
TPRER	005532	TYPON	104404	UIPDR2=	177604	\$CSRA	001334	\$GTSWR	041660
TPRI	000006	TYPOS	104403	UIPDR3=	177606	\$CSRB	001336	\$HD	000003
TPRNXM	031600	T\$NAME	104415	UIPDR4=	177610	\$CSRC	001340	\$HIBTS	001000
TPRTO	031710	T\$NAM	043740	UIPDR5=	177612	\$DBLK	041252	\$HILIM	050750
TPROTR	001346	T10.1	022244	UIPDR6=	177614	\$DDW0	001252	\$HIOCT	042532
TPRO0	175000	T10.2	022500	UIPDR7=	177616	\$DDW1	001254	\$ICNT	001104
TPRO1	175002	T10.3	024034	UNEXPI	003056	\$DDW10	001276	\$INTAG	001135
TPRO2	175004	T11	000004	UNKNMN	052556	\$DDW11	001300	\$ITEMB	001114
TPRO3	175006	T11\$	050770	UNXHLT	003015	\$DDW12	001302	\$LEDS	001340
TPRO4	175010	T11.1	025072	UNXRES	002734	\$DDW13	001304	\$LF	001164
TPRO5	175012	T11.2	025430	UNXT	046217	\$DDW14	001306	\$LFLG	042777
TPRO6	175014	T13.1	027522	UNXTPR	002762	\$DDW15	001310	\$LOAD	060550
TPRO7	175016	T13.2	030124	UNXTRP	002416	\$DDW2	001256	\$LPADR	001106
TPRO8	175020	T13.3	031174	UP	000000	\$DDW3	001260	\$LPERR	001110
TPRO9	175022	T15.1	033274	UPAMB	057774	\$DDW4	001262	\$LSIZE	000125
TPR10	175024	T15.2	035252	UPERR	060240	\$DDW5	001264	\$LSTRT	077416
TPR11	175026	T15.3	035564	USP	001100	\$DDW6	001266	\$MADR1	001220
TPR12	175030	T2.1	003466	VARC	037625	\$DDW7	001270	\$MADR2	001224
TPR13	175032	T2.10	005364	VA2GA	037436	\$DDW8	001272	\$MADR3	001230
TPR14	175034	T2.2	003620	VEXPD	046032	\$DDW9	001274	\$MADR4	001234
TPR15	175036	T2.3	004214	WAITP	057752	\$DEVCT	001176	\$MAIL	001166
TPVEC	000064	T2.4	004740	WAIT00	057736	\$DEVN	001244	\$MAMS1	001216
TRAPE	002456	T2.5	005046	WAIT1	060430	\$DLOAD	060564	\$MAMS2	001222
TRAPVE	000034	T2.6	005220	WAIT40	057744	\$DMA	001324	\$MAMS3	001226
TRAP4X	002426	T2.7	005316	WC	032112	\$DMPAR	007674	\$MAMS4	001232
TRBB	000001	T3.1	006044	WLZ	000400	\$DNARD	010314	\$MBADR	001002
TRFUNL	000010	T3.2	006376	WQIO	031732	\$DOAGN	040126	\$MFLG	042776
TRP.4	010540	T3.3	006546	WRD	000001	\$DPR	001320	\$MNEW	042363
TRTVEC	000014	T3.4	006752	WTLK	036667	\$DPRQ	001322	\$MSGAD	001202
TRWW	000000	T3.5	007316	WT.LCK	056056	\$DTBL	041242	\$MSGLG	001204
TRYNXT	051636	T3.6	007472	XCVR	000140	\$EN	000224	\$MSGTY	001166
TSET	036726	UDPARO=	177660	XDB	000002	\$ENDAD	040116	\$MSWR	042352
TSTLOC	001412	UDPARI=	177662	\$AGAIN	051616	\$ENDCT	037662	\$MTYP1	001217
TST.ST	056150	UDPAR2=	177664	\$APTHD	001000	\$ENULL	040132	\$MTYP2	001223
TST1	003160	UDPAR3=	177666	\$ATYC	042560	\$ENV	001206	\$MTYP3	001227
TST10	022134	UDPAR4=	177670	\$ATY1	042534	\$ENVN	001207	\$MTYP4	001233
TST11	024602	UDPAR5=	177672	\$ATY3	042542	\$EOP	037632	\$NULL	001154
TST12	026220	UDPAR6=	177674	\$ATY4	042552	\$EOPCT	037654	\$NWTST	000001
TST13	027412	UDPAR7=	177676	\$AUTOB	001134	\$EOP	040165	\$OCNT	041530
TST14	032124	UDPDR0=	177620	\$BASE	001242	\$EOPID	040136	\$OMODE	041532
TST15	033174	UDPDR1=	177622	\$BDADR	001122	\$EOPP	040155	\$OVER	040432
TST16	037176	UDPDR2=	177624	\$BDDAT	001126	\$ERFLG	001103	\$PASS	001174
TST2	003404	UDPDR3=	177626	\$BIN	041606	\$ERMAX	001115	\$PASTM	001006
TST3	005652	UDPDR4=	177630	\$CDW1	001246	\$ERROR	044314	\$PATCH	050774
TST4	011312	UDPDR5=	177632	\$CDW2	001250	\$ERRPC	001116	\$PIO	001330
TST5	012506	UDPDR6=	177634	\$CHARC	041032	\$ERRTB	001312	\$PNUM	000010

E16

KXJ11-CA FUNCTIONAL TEST
Symbol table

MACRO Y05.02 Thursday 03 Apr 86 14:11 Page 198

\$QDLY 001342	\$SVLAD 040376	\$TRAP 043002	\$TYPOS 041306	\$\$TDX 041262
\$QIR 001316	\$SVPC = 001000	\$TRAP2 043024	\$UNIT 001200	\$OFILL 041531
\$QUES 001162	\$SWR = 101400	\$TRP = 000020	\$UNITM 001010	\$4OCAT 000000
\$RDCHR 042072	\$SWREG 001210	\$TRPAD 043036	\$UPPAR 010046	.ABUSW 050736
\$ROLIN 042222	\$SWRMK= 000200	\$TSTM 001004	\$UPWRD 010176	.LSTAD 050734
\$RDOCT 042374	\$SW08T 040446	\$TSTNM 001102	\$USWR 001212	.LSTPG 050732
\$RDSZ = 000010	\$TCID 001340	\$TTYIN 042330	\$VECT1 001236	.MMU = 050732
\$RTNAD 040130	\$TESTN 001172	\$TYPBN 041534	\$VECT2 001240	.MSIZE 050730
\$SCOPE 040206	\$TKB 001146	\$TYPDS 041036	\$XOFF = 000023	.SIZE 052574
\$SETUP= 000127	\$TKS 001144	\$TYPE 040502	\$XON = 000021	.SIZKT 052726
\$SLOAD 060736	\$TN = 000017	\$TYPEC 040714	\$XTSTR 040230	.SIZME 053100
\$SL2 001326	\$TPB 001152	\$TYPEX 041034	\$\$ARB 051510	.SIZXI 053156
\$START 061000	\$TPFLG 001157	\$TYPOC 041332	\$\$GET4= 000000	.\$X = 001000
\$STUP = 177777	\$TPS 001150	\$TYPON 041346	\$\$SW08= 000017	

. ABS. 101022 000 (RW,I,GBL,ABS,OV)R)
000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 422
Work file writes: 347
Size of work file: 57336 Words (224 Pages)
Size of core pool: 19402 Words (74 Pages)
Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:03:03.58
CZKXAA,CZKXAA/CR/-SP=MACMAC.MLB/ML,CZKXAA