12		1	
11		1	
10		1	
9		1	
8		1	
7		1	
6		1	
5		1	
4		1	
3		1	
2		1	
1		1	

LIST / LIST / SHEET	STRAN / STRANA / PAGE	IZDAJA / IZDANJE / ISSUE	LIST / LIST / SHEET	STRAN / STRANA / PAGE	IZDAJA / IZDANJE / ISSUE	LIST / LIST / SHEET	STRAN / STRANA / PAGE	IZDAJA / IZDANJE / ISSUE

57

DELOVNA KOPIJA

TESTING PROCEDURE

	Priimek in ime	Podpis	Gradivo	Odstopi netol. mer	Toplotna obdelava	Površ. zaščita	Pripadnost
							TRIGLAV
Konstr.	Žalar D.		Izdaja 1				
Projekt.	Žalar D.		Znak				
Pregled.	Žalar D.		Št. obv. 30-044				
Števil.	J. Bojanič		Datum 15.2.89				
Stand.	B. Maležič		Podpis				

Iskra Delta
proizvodnja računalniških sistemov in inženiring, p.o.

Naziv: PVT CPU-J1.1

List	Stran	J	K	Identifikacijska številka dok.
1				8477044

Namembnost kopije: PREDPIS PREIZ/EN

Arhiv

Merilo	Sekcija	Namesto identifikacijske številke 1 2 d.
		19351044

VME CPU-J11 TESTING PROCEDURE

For testing procedure it is necessary to instal testing eproms TSTL (84 767 044) on IC30 and TSTH (84 769 044) on IC29.

The testing procedure consists of four testing routines and a down line loader program. Routines are executed sequentially in loop. Description of test routines follows:

1. ROM CHECKSUM

The ROM checksum routine calculates word checksum of TSTROM data. TSTROM data are on I/O page in physical address space from 17773000 to 17773777. This routine prompts with a ROM message. If there is a checksum error, then an error message ERR occurs and the testing procedure is started again from the beginning. If there is no error, then testing proceeds with the console output routine.

2. CONSOLE OUTPUT

The console output routine outputs a string of ASCII characters on console port. It prompts with CON message. The output string is 1234560123456789. This routine does not use the interrupt mechanism. Testing proceeds with the console input routine.

3. CONSOLE INPUT (ECHO)

The console input routine asks for input string of ASCII characters. It prompts with CIN message. Then it outputs the ENQ ASCII character and waits for input string. Input string must be equal to 12345678. If not, then an error message ERR occurs and testing procedure is started again from the beginning. If there is no input response, then time out message TOT occurs and testing proceeds with memory routine as if there were no error.

4. MEMORY

The memory routine sequentially writes random data into local memory from physical address 0 to 157777 (56kB). Then it compares to valid data. It prompts with RAM message. If there is a data or parity error, then error message ERR occurs and testing procedure is started again from the beginning. Complete test lasts approximately 12 minutes. Then if there is no error, the testing proceeds with loader program.

5. DOWN LINE LOADER

The down line loader program is not used for testing but for loading larger tests into memory.

The loader program prompts with message LDR and ready command (ASCII ^A). Then it waits to start the load command (ASCII ^C).

When the start load command is received, the loader answers with LDR and ready command and waits for input record.

If there is no response, then a time out message TOT occurs and the testing procedure is started again from the beginning. If there is a record checksum error, then negative acknowledge error message NAK with retransmit command (ASCII ^D) occurs and the loader waits for retransmission.

The load and start address must be in range from 0 to 157777, else error message ERR occurs and the testing procedure is started again from the beginning.

The record must be organized as follows:

LLAAAATTDD....DDCC

LL	record length in bytes
AAAA	load or start address
TT	record type
DD	data bytes
CC	record checksum mod 256

record type 00 is data record
record type 01 is command record (start command)

All input data are in HEX ASCII format. All non-HEX input data (except command characters) are ignored.

NOTE: If trap 4 (I/O bus time out, non-existent memory....) occurs any time during testing procedure, then error message ERR appears and the testing procedure is started again from the beginning.

```
                                .ENABL  AMA
                                .ENABL  LC
                        ;*****************************************
                        ;*
                        ;*              VME CPU J11 Module Test V1.0
                        ;*
                        ;*              ROM Checksum
                        ;*              Console Output
                        ;*              Console Input (Echo)
                        ;*              RAM (64kB-8kB)
                        ;*              Load Diagnostic
                        ;*
                        ;*****************************************

        000000'                 X=.
        173000'                 .=X+173000

        000004          T4      =       4               ; non-existent memory trap
        000114          T114    =       114             ; parity error trap


173000  000005          START:  RESET

173002  012706  173242'         MOV     #ERR,SP         ; initialize stack

173006  010637  000004          MOV     SP,@#T4         ; initialize non-
173012  012737  000340  000006  MOV     #340,@#T4+2     ; existent memory trap

173020  010637  000114          MOV     SP,@#T114       ; initialize
173024  012737  000340  000116  MOV     #340,@#T114+2   ; parity error trap
```

```
                        ;*****************************************
                        ;*
                        ;*              ROM Checksum
                        ;*
                        ;*****************************************
                        ;
                        ; R0 message pointer           R4 unmodified
                        ; R1 ROM word length counter   R5 return address
                        ; R2 ROM address pointer       SP unmodified
                        ; R3 checksum register

        173000          ROMADD  =       173000          ; ROM start address
        001000          ROMLEN  =       1000            ; ROM length (bytes)

173032  012700  173626'         MOV     #ROMMSG,R0      ; set output message pointer
173036  012705  173044'         MOV     #ROM,R5         ; set return address
173042  000412                  BR      SLU             ; output message

173044  012702  173000  ROM:    MOV     #ROMADD,R2      ; set start ROM address
173050  012701  000400          MOV     #ROMLEN/2,R1    ; set ROM length in words

173054  005003                  CLR     R3              ; set initial checksum word
173056  062203          10$:    ADD     (R2)+,R3        ; add data word to checksum
173060  077102                  SOB     R1,10$          ; until end of ROM
173062  001067                  BNE     ERR             ; if checksum not equal zero
                                                        ; then ROM checksum error

173064  012705  173110'         MOV     #ECHO,R5        ; set return address
```

```
                        ;*****************************************
                        ;*
                        ;*              Console Output
                        ;*
                        ;*****************************************
                        ;
                        ; R0 message pointer           R4 unmodified
                        ; R1 message length counter    R5 return address
                        ; R2 unmodified                SP unmodified
                        ; R3 unmodified

        177564          XCSR    =       177564          ; transmitter control/status register
        177566          XBUF    =       177566          ; transmitter data register

173070  112001          SLO:    MOVB    (R0)+,R1        ; get message length

173072  105737  177564  10$:    TSTB    @#XCSR          ; transmitter empty?
173076  100375                  BPL     10$             ; if not, wait
173100  112037  177566          MOVB    (R0)+,@#XBUF    ; put character
173104  077106                  SOB     R1,10$          ; until end of message

173106  000115                  JMP     (R5)            ; return
```

```
                        ;*****************************************
                        ;*
                        ;*              Console Input (Echo)
                        ;*
                        ;*****************************************
                        ;
                        ; R0 message pointer           R4 unmodified
                        ; R1 message length counter    R5 return address
                        ; R2 time constant             SP unmodified
                        ; R3 time constant
                        
        177560          RCSR    =       177560          ; receiver control/status register
        177562          RBUF    =       177562          ; receiver data regiter
        004000          TIM     =       4000            ; time constant
                        
173110  112001          ECHO:   MOVB    (R0)+,R1        ; get input length
                        
173112  012702  004000          MOV     #TIM,R2         ; set time constant
173116  010203          10$:    MOV     R2,R3           ; set time constant
                        
173120  105737  177560  20$:    TSTB    @#RCSR          ; receiver full?
173124  100007                  BPL     40$             ; if not, wait
173126  122037  177562          CMPB    (R0)-,@#RBUF    ; test for valid input
173132  001043                  BNE     ERR             ; if not, error
173134  077107                  SOB     R1,20$          ; until end of message
                        
173136  012705  173170'  30$:    MOV     #RAM,R5         ; set return address
173142  000752                  BR      SLU             ; output message
                        
173144  077313          40$:    SOB     R3,20$          ; decrement time constant
173146  077215                  SOB     R2,10$          ; decrement time constant
                        
173150  012700  173757'          MOV     #TOTMSG,R0      ; set tine out message
173154  012705  173162'          MOV     #50$,R5         ; set return address
173160  000743                  BR      SLU             ; output message
                        
173162  012700  173732'  50$:    MOV     #RAMMSG,R0      ; set ram message
173166  000763                  BR      30$             ; set return address
                                                        ; output message
```

```
                        ;****************************************
                        ;*
                        ;*              RAM Write/Compare
                        ;*
                        ;****************************************
                        ;
                        ; R0 unmodified                R4 write=0/verify=1 flagg
                        ; R1 data register            R5 return address
                        ; R2 pattern modifier register  SP unmodified
                        ; R3 address register

177746                  MEMPAR  =       177746          ; memory parity register address
160000                  RAMLEN  =       160000          ; memory length

001000                  ENPAR   =       1000            ; parity enable bit

                                .ENABL  LSB
173170 052737 001000 177746 RAM:    BIS     #ENPAR,@#MEMPAR ; enable local memory parity

173176 005002                       CLR     R2              ; set initial pattern modifier
173200 005004                       CLR     R4              ; set write/verify flagg to write

173202 005003           10$:        CLR     R3              ; set initial address
173204 022703 000004    20$:        CMP     #4,R3           ; non-existent memory trap?
173210 001403                       BEQ     30$             ; if yes, branch
173212 022703 000114                CMP     #114,R3         ; parity error trap?
173216 001001                       BNE     40$             ; if not, branch
173220 022323           30$:        CMP     (R3)+,(R3)+     ; skip trap
173222 010301           40$:        MOV     R3,R1           ; copy address
173224 000301                       SWAB    R1              ; swap high and low byte of address
173226 074301                       XOR     R3,R1           ; xor high and low byte of address
173230 074201                       XOR     R2,R1           ; xor with pattern modifier

173232 005704                       TST     R4              ; test for write/verify flagg
173234 001407                       BEQ     50$             ; branch to write pass

173236 120123                       CMPB    R1,(R3)+        ; verify data
173240 001411                       BEQ     60$             ; if equal, continue

173242 012700 173747'   ERR:    MOV     #ERRMSG,R0      ; set error message
173246 012705 173000'   LOOP:   MOV     #START,R5       ; set start return address
173252 000706                   BR      SLU             ; output message

173254 110123           50$:    MOVB    R1,(R3)+        ; write data
173256 022703 160000            CMP     #RAMLEN,R3      ; end of memory bank?
173262 001350                   BNE     20$             ; no, continue

173264 005104           60$:    COM     R4              ; complement write/verify flagg
173266 001345                   BNE     10$             ; branch to verify pass
173270 105202                   INCB    R2              ; next pattern modifier
173272 001343                   BNE     10$             ; branch to write pass

173274 012705 173452'           MOV     #LDR,R5         ; set return address
173300 000673                   BR      SLU             ; output ready message for load
                                .DSABL  LSB
```

```
                        ;*****************************************
                        ;*
                        ;*              Load Diagnostic
                        ;*
                        ;*****************************************
                        ;
                        ; R0 character register        R4 record checksum (low byte)
                        ; R1 time constant                high nibble buffer (high byte)
                        ; R2 load/transfer address     R5 return address
                        ; R3 data byte counter (low byte)
                        ;    time constant (high byte) SP subroutines stack pointer
                        ;
                        ;
                                .ENABL  LSB
173302  100063          10$:    BPL     LDR             ; wait for start load command
173304  012700  173740' 20$:    MOV     #LDRMSG,R0      ; set ready message for load
173310  012705  173316' 30$:    MOV     #LOA,R5         ; set return address
173314  000665                  BR      SLU             ; output ready message for load
                        ;
173316  005004          LOA:    CLR     R4              ; initialize checksum
173320  012705  173326'         MOV     #40$,R5         ; set return address
173324  000520                  BR      INBYT           ; get data byte count
173326  010003          40$:    MOV     R0,R3           ; R3 is data byte counter
173330  012705  173336'         MOV     #50$,R5         ; set return address
173334  000514                  BR      INBYT           ; get high byte of address
173336  010002          50$:    MOV     R0,R2           ; R2 is address pointer
173340  012705  173346'         MOV     #60$,R5         ; set return address
173344  000510                  BR      INBYT           ; get low byte of address
173346  000302          60$:    SWAB    R2              ; set high byte of address
173350  150002                  BISB    R0,R2           ; set low byte of address
173352  012705  173360'         MOV     #70$,R5         ; set return address
173356  000503                  BR      INBYT           ; get record type byte
173360  005700          70$:    TST     R0              ; test for record type
173362  001414                  BEQ     90$             ; data record
173364  077052                  SOB     R0,ERR          ; if not equal error else command record
173366  012705  173374'         MOV     #80$,R5         ; set return address
173372  000475                  BR      INBYT           ; get checksum byte
173374  001023          80$:    BNE     120$            ; if not equal, checksum error
173376  020227  160000          CMP     R2,#RAMLEN      ; check for valid transfer address
173402  103317                  BHIS    ERR             ; if out of range then error
173404  012700  173740'         MOV     #LDRMSG,R0      ; set ready message for end load
173410  010205                  MOV     R2,R5           ; set return address as start address
173412  000626                  BR      SLU             ; output message and start execution
                        ;
173414  012705  173422' 90$:    MOV     #100$,R5        ; set return address
173420  000462                  BR      INBYT           ; get data byte
173422  020227  160000  100$:   CMP     R2,#RAMLEN      ; check for valid RAM address
173426  103305                  BHIS    ERR             ; if out of range then error
173430  110022                  MOVB    R0,(R2)+        ; load data byte into memory
173432  077310                  SOB     R3,90$          ; decrement byte count
173434  012705  173442'         MOV     #110$,R5        ; set return address
173440  000452                  BR      INBYT           ; get checksum byte
173442  001720  110$:   BEQ     20$             ; if checksum ok then get next record
173444  012700  173767' 120$:   MOV     #NAKMSG,R0      ; else negative acknowledge
173450  000717                  BR      SCS             ; and retry
                        ;
173452  012706  173302' IDP:    MOV     #10$,SP         ; set return address and get character
                                .DSABL  LSB
```

```
                        ;****************************************
                        ;*
                        ;*              Loader Subroutines
                        ;*
                        ;****************************************

173456  012701  010000  INCH:   MOV     #2*TIM,R1       ; set time constant
173462  000303                  SWAB    R3              ; save byte counter
173464  105003                  CLRB    R3              ; set time constant
173466  105737  177560  10$:    TSTB    @#RCSR          ; receiver full?
173472  100027                  BPL     50$             ; if not, wait
173474  113700  177562          MOVB    @#RBUF,R0       ; get character
173500  022700  000003          CMP     #3,R0           ; start load character ^C ?
173504  001002                  BNE     20$             ; no, continue
173506  000270                  SEN                     ; yes, set negativ flagg
173510  000417                  BR      40$             ; return
173512  162700  000071  20$:    SUB     #'9,R0          ; convert ascii to hex:
173516  003002                  BGT     30$
173520  062700  000007          ADD     #7,R0
173524  062700  000002  30$:    ADD     #2,R0
173530  002752                  BLT     INCH            ; if less then zero ignore not hex
173532  042700  000040          BIC     #40,R0          ; for lower case
173536  020027  000017          CMP     R0,#15.
173542  003345                  BGT     INCH            ; if greater then ignore not hex
173544  105003                  CLRB    R3              ; clean byte counter
173546  000303                  SWAB    R3              ; restore byte counter
173550  000116          40$:    JMP     (SP)            ; return
173552  105303          50$:    DECB    R3              ; decrement time constant
173554  001344                  BNE     10$             ; until zero
173556  077135                  SOB     R1,10$          ; decrement time constant
173560  012700  173757'         MOV     #TOTMSG,R0      ; set time out message
173564  000630                  BR      LOOP            ; start testing again
173566  012706  173574'  INBYT:  MOV     #10$,SP         ; set return address
173572  000731                  BR      INCH            ; get high nibble
173574  072027  000014  10$:    ASH     #12.,R0         ; shift in high byte, high nibble
173600  050004                  BIS     R0,R4           ; save in R4 high byte
173602  012706  173610'         MOV     #20$,SP         ; set return address
173606  000723                  BR      INCH            ; get low nibble
173610  000304          20$:    SWAB    R4              ; allign high nibble
173612  150400                  BISB    R4,R0           ; merge, high byte is zero
173614  000304                  SWAB    R4              ; recover old checksum
173616  060004                  ADD     R0,R4           ; calculate new checksum
173620  042704  177400          BIC     #177400,R4      ; clear high byte of checksum
173624  000115                  JMP     (R5)            ; return
```

```
                                ;*****************************************
                                ;*
                                ;*              Output Messages
                                ;*
                                ;*****************************************

173626    034                   ROMMSG: .BYTE   <SLUMSG-ROMMSG-1>
173627    015     012     126           .ASCII  <15><12>/VKE CPU J11 Test V1.0/
173632    115     105     040
173635    103     120     125
173640    040     112     061
173643    061     040     124
173646    145     163     164
173651    040     126     061
173654    056     060
173656    015     012     122           .ASCII  <15><12>/RON/
173661    117     115
173663    035                   SLUMSG: .BYTE   <ECHMSG-SLUMSG-1>
173664    015     012     103           .ASCII  <15><12>/COT/
173667    117     124
173671    015     012     061           .ASCII  <15><12>/12345601234567 89/
173674    062     063     064
173677    065     066     060
173702    061     062     063
173705    064     065     066
173710    067     070     071
173713    015     012     103           .ASCII  <15><12>/CIN/<5>              ; answerback
173716    111     116     005
173721    010                   ECHMSG: .BYTE   <RAMMSG-ECHMSG-1>
173722    061     062     063           .ASCII  /12345678/
173725    064     065     066
173730    067     070
173732    005                   RAMMSG: .BYTE   <LDRMSG-RAMMSG-1>
173733    015     012     122           .ASCII  <15><12>/RAM/
173736    101     115
173740    006                   LDRMSG: .BYTE   <ERRMSG-LDRMSG-1>
173741    015     012     114           .ASCII  <15><12>/LDR/<1>              ; load ^A
173744    104     122     001
173747    007                   ERRMSG: .BYTE   <TOTMSG-ERRMSG-1>
173750    015     012     105           .ASCII  <15><12>/ERR/<0><0>
173753    122     122     000
173756    000
173757    007                   TOTMSG: .BYTE   <NAKMSG-TOTMSG-1>
173760    015     012     124           .ASCII  <15><12>/TOT/<0><0>
173763    117     124     000
173766    000
173767    006                   NAKMSG: .BYTE   <MSG-NAKMSG-1>
173770    015     012     116           .ASCII  <15><12>/NAK/<4>              ; negative ack ^D
173773    101     113     004
173776                          MSG:

                  173776'               .=.-173776
173776  113306                  ROMCHK: .WORD   113306                        ROM checksum

                                ;*****************************************

                  173000'               .END    START
```


Izdaja	1					List	Stran	J	K	Identifikacijska številka dok
Št. obvestila	30-044					11				84772044
IskraDelta proizvodnja računalniških sistemov in inženiring, p.o.						Arhiv				Namesto identifikacijske številke 124
										19351044

aero 82-3407-13-8

```
.MAIN.  MACRO 2.00   25-Mar-87 09:45  Page 8-1
Symbol Table

ECHMSG  173721R     LDR     173452R     RAM     173170R     ROMCHK  173776R     TOTMSG  173757R
ECHO    173110R     LDRMSG  173740R     RAMLEN= 160000      ROMLEN= 001000      TTI4    = 000114
ENPAR = 001000      LOA     173316R     RAMMSG  173732R     ROMMSG  173626R     TX4     = 000004
ERR     173242R     LOOP    173246R     RBUF    = 177562    SLU     173070R     X       = 000000
ERRMSG  173747R     MEMPAR= 177746      RCSR    = 177560    SLUMSG  173663R     XBUF    = 177566
INBYT   173566R     MSG     173776R     ROM     173044R     START   173000R     XCSR    = 177564
INCR    173456R     NAKMSG  173767R     ROMADD= 173000      TIM   = 004000

. ABS.  000000     000
        174000     001
Errors detected:  0

Virtual memory used:  466 words  ( 2 pages)
Dynamic memory:  3086 words  ( 11 pages)
Elapsed time:  00:00:09
,TSTROM=TSTROM
```