```
;;; -*-Mode:LISP;Package:CC;Base:8.;Fonts:CPTFONT-*-
;;; Cadr diagnositics

(DEFUN ROT32 (NUM AMT)
  (LOGAND 37777777777
	  (COND ((< AMT 30) (+ (ASH NUM AMT) (LDB (+ (LSH (- 40 AMT) 6) AMT) NUM)))
		(T (DPB (LDB (- 40 AMT) NUM)
			(+ (LSH AMT 6) (- 40 AMT))
			(ASH NUM (- AMT 40)))))))

(DEFMACRO ONES-COMPLEMENT (X)  ;Can't use LOGXOR with -1 on bignums!
  `(BOOLE 14 0 ,X))
```

```
;;; Function for scoping.  Stop when a key hit.  Only forms which evaluate their
;;; arguments allowed here.

(DEFUN CC-LOOP (FORM)
  (DO ((FCN (CAR FORM))
       (ARGS (MAPCAR #'EVAL (CDR FORM))))
      ((KBD-TYI-NO-HANG))
    (APPLY FCN ARGS)))

(DECLARE (SPECIAL CC-LOW-LEVEL-FLAG CC-DIAG-TRACE))
(DEFVAR ALL-DATA-PATHS
        '(CC-TEST-IR-DP CC-TEST-PC-DP CC-TEST-MD-DP CC-TEST-VMA-DP
        CC-TEST-M-MEM-DP CC-TEST-A-MEM-DP
        CC-TEST-PP-DP CC-TEST-PI-DP CC-TEST-PDL-DP CC-TEST-Q-DP CC-TEST-C-MEM-DP
        CC-TEST-LC-DP CC-TEST-A-PASS-DP CC-TEST-M-PASS-DP
        CC-TEST-ALU-SHIFT-LEFT-DP CC-TEST-ALU-SHIFT-RIGHT-DP
        CC-TEST-UNIBUS-MAP-DP CC-TEST-BUSINT-BUFFERS-DP))
(DEFVAR ALL-MEMORIES
      '( (M-MEM ,RAMMO 32. 5.)
         (A-MEM ,RAAMO 32. 10.)
         (PDL-BUFFER ,RAPBO 32. 10.)
         (C-MEM ,RACMO 48. 14.)
         (D-MEM ,RADMO 16. 11.)
;NOTE, CAN'T TEST BIT 16 OF D-MEM (R-BIT)
         (SPC ,RAUSO 19. 5.)
         (LEVEL-1-MAP ,RAM10 5. 11.)
         (LEVEL-2-MAP ,RAM20 24. 10.)
         (UNIBUS-MAP ,RAUBMO 16. 4) ))

(defmacro deftest (function-name defun-args test-name &body forms)
  `(progn 'compile
          (putprop ',function-name ,test-name 'test-name)
          (defun ,function-name ,defun-args ,@forms)))

(DEFMACRO RUN-TEST-FUNCTION (FUNCTION-NAME &REST ARGS)
  `(PROGN 'COMPILE
          (format t "~2&Running ~A test~%" (get ',function-name 'test-name))
          (,FUNCTION-NAME ,@ARGS)))

;;; Toplevel machine checking
(DEFUN CC-TEST-MACHINE (&OPTIONAL (C-MEM-BANKS-TO-TEST 3))
  (LET ((CC-LOW-LEVEL-FLAG 'VERY))
    (IF (AND (EQ DBG-ACCESS-PATH 'SERIAL)
             (BOUNDP 'SERIAL-STREAM))
        (FUNCALL SERIAL-STREAM ':CLEAR-INPUT))
    (FORMAT T "~&For best results, ground -TPTSE, 1C07-09 on CMEM boards
Resetting machine")
    (DBG-RESET)          ;Forcibly reset the whole machine
    (CC-RESET-MACH)                ;NOW SET TO THE CORRECT MODE
    (RUN-TEST-FUNCTION CC-TEST-DATA-PATHS ALL-DATA-PATHS)
    (RUN-TEST-FUNCTION CC-FAST-ADDRESS-TESTS ALL-MEMORIES)
    (RUN-TEST-FUNCTION CC-FAST-ADDRESS-TEST-C-MEM-BANKS C-MEM-BANKS-TO-TEST)
    (RUN-TEST-FUNCTION CC-TEST-SPC-POINTER)
    (RUN-TEST-FUNCTION CC-TEST-SHIFTER-LOGIC)
    (RUN-TEST-FUNCTION CC-TEST-OA-REGS)
    (RUN-TEST-FUNCTION CC-TEST-DISPATCH)
    (RUN-TEST-FUNCTION CC-TEST-CLOCK)
    NIL))

(DEFTEST CC-TEST-SHIFTER-LOGIC () "Shifter logic"
  (CC-TEST-MASK-LEFT)
  (CC-TEST-MASK-RIGHT)
  (CC-TEST-MASKER)
  (CC-TEST-SHIFTER)
  (CC-TEST-LC-AFFECTS-SHIFT))

(DEFTEST CC-TEST-DATA-PATHS (DATA-PATH-LIST) "Data paths"
   (MAPC #'(LAMBDA (FUNCTION)
             (FORMAT T "~&~4TRunning ~A.~%" FUNCTION)
             (APPLY FUNCTION NIL))
         DATA-PATH-LIST))

(DEFtest CC-FAST-ADDRESS-TESTS (MEMORIES-LIST) "Fast address"
  (SEND TERMINAL-IO ':TYO #\CR)
  (MAPC #'(LAMBDA (X)
            (FORMAT T "~&~4TFast address test ~A~%" (CAR X))
            (APPLY 'CC-FAST-ADDRESS-TEST X))
        MEMORIES-LIST))

(DEFtest CC-GROSS-DATA-TESTS (MEMORIES-LIST) "Gross data"
   (LET ((CC-LOW-LEVEL-FLAG 'VERY))
     (FORMAT T "Resetting machine . . .")
     (DBG-RESET)                  ;Forcibly reset the whole machine
     (CC-RESET-MACH)              ;Now set to the correct mode
     (SEND TERMINAL-IO ':TYO #\CR)
     (MAPC #'(LAMBDA (X)
               (FORMAT T "~&~4TGross data test ~A~%" (CAR X))
            (APPLY 'CC-GROSS-DATA-TEST X))
        MEMORIES-LIST)))

(DEFUN CC-OTHER-TESTS NIL
  (RUN-TEST-FUNCTION CC-TEST-PC-INCREMENTER)
  (RUN-TEST-FUNCTION CC-TEST-SPY-IR)
  (RUN-TEST-FUNCTION CC-TEST-INCREMENTER)
  (RUN-TEST-FUNCTION CC-TEST-ARITH-COND-JUMP)
  (RUN-TEST-FUNCTION CC-GROSS-DATA-TESTS ALL-MEMORIES)
  (RUN-TEST-FUNCTION CC-ADDRESS-TEST-A-MEM)
  (RUN-TEST-FUNCTION CC-TEST-M-MEM-ADR)
  (RUN-TEST-FUNCTION CC-TEST-A-MEM-ADR)
  (RUN-TEST-FUNCTION CC-TEST-PDL-ADR))

;Test each 4K separately since they have separate address drivers
(DEFtest CC-FAST-ADDRESS-TEST-C-MEM-BANKS (&OPTIONAL (NBANKS 4)) "C-MEM Banks Fast Address"
  (DOTIMES (BANK NBANKS)
    (CC-FAST-ADDRESS-TEST
      (FORMAT NIL "CMEM-BANK ~A" BANK) (+ RACMO (* BANK 10000)) 48. 12.)))
```

```
;;; Toplevel data path tests

(DEFUN CC-TEST-IR-DP ()
  (CC-TEST-DATA-PATH "Unibus → DEBUG-IR → IR → SPY1 → Unibus" RAIR 48.))

(DEFUN CC-TEST-PC-DP ()
  (CC-TEST-DATA-PATH "Unibus → DEBUG-IR → IR(Jump) → PC → SPY2 → Unibus" RAPC 14.))

(DEFUN CC-TEST-MD-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → SPY1 → Unibus" RAMD 32.))

(DEFUN CC-TEST-VMA-DP ()
  (CC-TEST-DATA-PATH "(Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD
 → MF → (m) → ALU → Obus → VMAS → VMA → MF → (m)
 → ALU → Obus (M00 and M01 prints) → Unibus" RAVMA
32.))

(DEFUN CC-TEST-M-MEM-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → L → M-MEM → MLATCH → (m) → ALU → Obus → SPY1 →
 Unibus"
 RAMMO 32.))

(DEFtest CC-TEST-M-MEM (&OPTIONAL (V1 0) (V2 -1) (ADR 1)) "M-MEM"
  (until-key
    (CC-WRITE-M-MEM ADR V1)
    (CC-READ-M-MEM ADR)
    (CC-WRITE-M-MEM ADR V2)
    (CC-READ-M-MEM ADR)))

(DEFUN CC-TEST-A-MEM-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → L → A-MEM → ALATCH → (a) → ALU → Obus → SPY1 →
 Unibus"
 RAAMO 32.))

(DEFtest CC-TEST-A-MEM (&OPTIONAL (V1 0) (V2 -1) (ADR 1)) "A-MEM"
  (until-key
    (CC-WRITE-A-MEM ADR V1)
    (CC-READ-A-MEM ADR)
    (CC-WRITE-A-MEM ADR V2)
    (CC-READ-A-MEM ADR)))

(DEFtest CC-TEST-A-MEM-ADDRESSES NIL "A-MEM Addresses"
  (WITHOUT-INTERRUPTS
    (DO ((ADR 1 (LSH ADR 1)))
        ((KBD-TYI-NO-HANG))
      (IF (> ADR 1000) (SETQ ADR 1))
      (CC-WRITE-A-MEM ADR 0)
      (CC-WRITE-A-MEM ADR -1))))

(DEFUN CC-TEST-PP-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → PP → MF → (m) → ALU → Obus → SPY1 → Unibus"
                     RAPP 10.))

(DEFUN CC-TEST-PI-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → PI → MF → (m) → ALU → Obus → Unibus"
                     RAPI 10.))

(DEFUN CC-TEST-PDL-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → L → PDL-Buffer → PLATCH → (m) → ALU → Obus →
 Unibus"
                     RAPBO 32.))

(DEFtest CC-TEST-PDL-ADDRESSES () "PDL Addresses"
  (DO ((BIT 1 (IF (> BIT 1000) 1 (LSH BIT 1))))
      ((KBD-TYI-NO-HANG))
    (CC-R-D (+ RAPBO BIT) 0)))

(DEFUN CC-TEST-Q-DP ()
  (CC-TEST-DATA-PATH "Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → ALU →
 Q → MF → (m) → ALU → Obus → Unibus"
                     RAQ 32.))

(DEFUN CC-TEST-C-MEM-DP ()
  (CC-TEST-DATA-PATH
"*FIRST* Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
     → ALU → Obus (M00 and M01 prints) → L  → M-MEM
*THEN* Unibus → Xbus (MEM bus on 1A-J1 and 1B-J1) → MDS → MD → MF → (m)
     → ALU → Obus (M00 and M01 prints) → L  → A-MEM
*AND FINALLY* A-MEM & M-MEM → IWR → C-MEM → IR(Jump) → Unibus" RACMO 48.))

(DEFtest CC-TEST-C-MEM (&OPTIONAL (V1 0) (V2 -1) (ADR 0)) "C-MEM"
  (UNTIL-KEY
    (CC-WRITE-C-MEM ADR V1)
    (CC-WRITE-C-MEM ADR V2)))

(DEFUN CC-TEST-LC-DP ()
  (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-INT-CNTRL 1_29.)   ;SET LC BYTE MODE
  (CC-TEST-DATA-PATH "Unibus → Xbus → MD → MF → (m)
 → ALU → Obus (M00 and M01 prints) → LC → MF → M → ALU → Obus" RALC 26.))

(DEFUN CC-TEST-A-PASS-DP ()
  (CC-TEST-DATA-PATH " → L → APASS → A → ALU" '(CC-A-PASS-HANDLER) 32.))

(DEFUN CC-TEST-M-PASS-DP ()
  (CC-TEST-DATA-PATH " → L → MPASS → MF → M → ALU" '(CC-M-PASS-HANDLER) 32.))

(DEFUN CC-TEST-ALU-SHIFT-LEFT-DP ()
  (CC-TEST-DATA-PATH "MD,Q(31) → ALU-SHIFT-LEFT-1" '(CC-ALU-SHIFT-LEFT-HANDLER) 32.))

(DEFUN CC-TEST-ALU-SHIFT-RIGHT-DP ()
  (CC-TEST-DATA-PATH "MD → M+M → ALU-SHIFT-RIGHT-1" '(CC-ALU-SHIFT-RIGHT-HANDLER) 32.))

(DEFUN CC-TEST-UNIBUS-MAP-DP ()
  (CC-TEST-DATA-PATH "Unibus Map" RAUBMO 16.))

;Read and write Xbus location 0 through all 16 Unibus buffers
(DEFUN CC-TEST-BUSINT-BUFFERS-DP ()
  (COND ((EQ SPY-ACCESS-PATH 'BUSINT)
         (DO DBG-UNIBUS-MAP-NUMBER 0 (1+ DBG-UNIBUS-MAP-NUMBER) (= DBG-UNIBUS-MAP-NUMBER 20)
           (CC-TEST-DATA-PATH
             (FORMAT NIL "Unibus → Buffer ~O → Xbus loc 0 → Buffer ~O → Unibus"
                         DBG-UNIBUS-MAP-NUMBER DBG-UNIBUS-MAP-NUMBER)
             200000 32.)))))
```

```
(DEFUN CC-A-PASS-HANDLER (OP DATA)
  (SELECTQ OP
    (WRITE-READ
     (CC-WRITE-MD DATA)          ;PUT VALUE INTO THE MRD REGISTER
     (CC-EXECUTE  ;NOTE NO WRITE, JUST PUT IT IN IR
       CONS-IR-M-SRC CONS-M-SRC-MD       ;MOVE IT TO DESIRED PLACE
       CONS-IR-ALUF CONS-ALU-SETM
       CONS-IR-OB CONS-OB-ALU
       CONS-IR-A-MEM-DEST (+ CONS-A-MEM-DEST-INDICATOR 0))
     (CC-EXECUTE (EXECUTOR CC-EXECUTE-LOAD-DEBUG-IR)
                 CONS-IR-A-SRC 0                          ;PUT IT ONTO THE OBUS
                 CONS-IR-ALUF CONS-ALU-SETA
                 CONS-IR-OB CONS-OB-ALU)
     (CC-DEBUG-CLOCK)    ;EXECUTE THE WRITE, LOAD IR WITH THE READ
     (LET ((ACTUAL (CC-READ-OBUS)))     ;READ BACK THE DATA VIA THE PASS AROUND PATH
       (COND ((AND CC-DIAG-TRACE (NOT (= ACTUAL DATA)))
              (FORMAT T "~%A-PASS WROTE ~O READ ~O" DATA ACTUAL)))
       ACTUAL))
    (OTHERWISE (FERROR NIL "Unknown A-Pass Handler Operation"))))

(DEFUN CC-M-PASS-HANDLER (OP DATA)
  (SELECTQ OP
    (WRITE-READ
      (CC-WRITE-MD DATA)                    ;PUT VALUE INTO THE MRD REGISTER
      (CC-EXECUTE  ;NOTE NO WRITE, JUST PUT IT IN IR
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-M-MEM-DEST 0)               ;ADR
      (CC-EXECUTE (EXECUTOR CC-EXECUTE-LOAD-DEBUG-IR)
                  CONS-IR-M-SRC 0        ;PUT IT ONTO THE OBUS
                  CONS-IR-ALUF CONS-ALU-SETM
                  CONS-IR-OB CONS-OB-ALU)
      (CC-DEBUG-CLOCK)  ;EXECUTE THE WRITE, LOAD IR WITH THE READ
      (LET ((ACTUAL (CC-READ-OBUS)))    ;READ BACK THE DATA VIA THE PASS AROUND PATH
        (COND ((AND CC-DIAG-TRACE (NOT (= ACTUAL DATA)))
               (FORMAT T "~%M-PASS WROTE ~O READ ~O" DATA ACTUAL)))
        ACTUAL))
    (OTHERWISE (FERROR NIL "UNKNOWN OP"))))

(DEFUN CC-ALU-SHIFT-LEFT-HANDLER (OP DATA)
  (SELECTQ OP
    (WRITE-READ
     (CC-WRITE-Q (ASH (LOGAND DATA 1) 31.))     ;low bit to high bit of Q
     (CC-WRITE-MD (ASH DATA -1))
     (CC-EXECUTE                                ;NOTE NO WRITE, JUST PUT IT IN IR
       CONS-IR-M-SRC CONS-M-SRC-MD
       CONS-IR-ALUF CONS-ALU-SETM
       CONS-IR-OB CONS-OB-ALU-LEFT-1)
     (LET ((ACTUAL (CC-READ-OBUS)))
       (COND ((AND CC-DIAG-TRACE (NOT (= ACTUAL DATA)))
              (FORMAT T "~%ALU-LEFT WROTE ~O READ ~O" DATA ACTUAL)))
       ACTUAL))
    (OTHERWISE (FERROR NIL "UNKNOWN OP"))))

(DEFUN CC-ALU-SHIFT-RIGHT-HANDLER (OP DATA)
  (SELECTQ OP
    (WRITE-READ
     (CC-WRITE-MD DATA)
     (CC-EXECUTE
       CONS-IR-M-SRC CONS-M-SRC-MD
       CONS-IR-ALUF CONS-ALU-M+M
       CONS-IR-OB CONS-OB-ALU-RIGHT-1)
     (LET ((ACTUAL (CC-READ-OBUS)))
       (COND ((AND CC-DIAG-TRACE (NOT (= ACTUAL DATA)))
              (FORMAT T "~%ALU-RIGHT WROTE ~O READ ~O" DATA ACTUAL)))
       ACTUAL))
    (OTHERWISE (FERROR NIL "UNKNOWN OP"))))

;;; Numeric list operations

(DEFMACRO NUMERIC-LIST-DELQ (N L)
   `(SETQ ,L (DELQ ,N ,L)))

(DEFUN NUMERIC-LIST-MEMQ (N L)
  (DO ((L L (CDR L)))
      ((NULL L) NIL)
    (AND (= (CAR L) N)
         (RETURN L))))

(DEFUN NUMERIC-LIST-UNION (L1 L2)
  (DO ((L L1 (CDR L))
       (R L2))
      ((NULL L) R)
    (OR (NUMERIC-LIST-MEMQ (CAR L) R)
        (SETQ R (CONS (CAR L) R)))))
```

```
(DEFUN NUMERIC-LIST-INTERSECTION (L1 L2)
  (DO ((L L1 (CDR L))
       (R NIL))
      ((NULL L) R)
    (AND (NUMERIC-LIST-MEMQ (CAR L) L2)
         (SETQ R (CONS (CAR L) R)))))

(DEFUN NUMERIC-LIST-DIFFERENCE (L1 L2)
  (DO ((L L1 (CDR L))
       (R NIL))
      ((NULL L) R)
    (OR (NUMERIC-LIST-MEMQ (CAR L) L2)
        (SETQ R (CONS (CAR L) R)))))
```

```
;;; Data path internals

(DEFUN CC-WRITE-AND-READ (REGADR DATA &OPTIONAL (MASK 37777777777))
  (COND ((ATOM REGADR)
         (CC-R-D REGADR DATA)
         (LET ((ACTUAL (CC-R-E REGADR)))
           (COND ((AND CC-DIAG-TRACE (NOT (ZEROP (LOGAND (LOGXOR ACTUAL DATA) MASK))))
                  (FORMAT T "~&Reg address ~O, wrote ~O, read ~O" REGADR DATA ACTUAL)))
           ACTUAL))
        (T (FUNCALL (CAR REGADR) 'WRITE-READ DATA))))

(DEFVAR CC-SUSPECT-BIT-LIST) ; Must be bound around CC-PRINT-BIT-LIST

;RETURNS T IF IT WORKS, PRINTS MESSAGE AND RETURNS NIL IF IT IS BUSTED.
(DEFUN CC-TEST-DATA-PATH (MESSAGE REGADR NBITS)
  (LET ((CC-LOW-LEVEL-FLAG 'VERY)
        (TEM)
        (CC-SUSPECT-BIT-LIST NIL)
        (ZEROS 0)
        (ONES (SUB1 (LOGDPB 1 (+ (LSH NBITS 6) 0001) 0))))
    (COND ((= (SETQ TEM (CC-WRITE-AND-READ REGADR ZEROS ONES))
              (CC-WRITE-AND-READ REGADR ONES ONES))
           (CC-BARF-ABOUT-DATA-PATH MESSAGE REGADR)
           (FORMAT T "~&~4TCan't affect it, erroneous value is ~O~%" TEM)
           NIL)
          (T (LET ((BITS-NOT-ONE (CC-TEST-DATA-PATH-FLOATING-BITS REGADR NBITS ZEROS))
                   (BITS-NOT-ZERO (CC-TEST-DATA-PATH-FLOATING-BITS REGADR NBITS ONES)))
               (COND ((AND (NULL BITS-NOT-ONE) (NULL BITS-NOT-ZERO)     ;NO ERROR
                           (NULL CC-SUSPECT-BIT-LIST))
                      T)
                     (T
                      (LET ((ERRONEOUS-BITS     ;BITS THAT LOSE, TEST FOR SHORTING
                              (NUMERIC-LIST-UNION BITS-NOT-ONE BITS-NOT-ZERO)))
                        (LET ((STUCK-AT-ZERO
                                (NUMERIC-LIST-DIFFERENCE BITS-NOT-ONE BITS-NOT-ZERO))
                              (STUCK-AT-ONE
                                (NUMERIC-LIST-DIFFERENCE BITS-NOT-ZERO BITS-NOT-ONE)))
                          (CC-BARF-ABOUT-DATA-PATH MESSAGE REGADR)
                          (CC-PRINT-BIT-LIST "Bits stuck at zero: " STUCK-AT-ZERO)
                          (CC-PRINT-BIT-LIST "Bits stuck at one: " STUCK-AT-ONE)
                          (AND (= (LENGTH ERRONEOUS-BITS) 2)  ;MAYBE THEY'RE SHORTED TOGETHER
                               (CC-TEST-DATA-PATH-SHORTED-BIT REGADR NBITS
                                                              (CAR ERRONEOUS-BITS)))
                          NIL)
                        (CC-PRINT-BIT-LIST "The following bits are also suspected of being losers:"
                                           CC-SUSPECT-BIT-LIST)

))))))))

;RETURN LIST OF BIT NUMBERS WHICH WON'T SET DIFFERENT FROM THE OTHERS.
;ALSO SETS CC-SUSPECT-BIT-LIST TO BITS WHICH ARE NOTICED TO
;BE LOSING WHILE TESTING DIFFERENT BITS.
;NOTE THE NEED TO DO BIGNUM ARITHMETIC.

(DEFUN CC-TEST-DATA-PATH-FLOATING-BITS (REGADR NBITS BACKGROUND)
  ;FIRST, DETERMINE SENSE OF BIT LOOKING FOR
  (LET ((BACK-BIT (COND ((ZEROP BACKGROUND) 0) (T 1)))
        (SET-BIT (COND ((ZEROP BACKGROUND) 1) (T 0)))
        (MASK (1- (LOGDPB 1 (+ (LSH NBITS 6) 0001) 0))))
    (DO ((BITNO 0 (1+ BITNO))
         (BITPOS 0001 (+ BITPOS 0100))
         (READBACK)
         (ERROR-LIST NIL))
        ((>= BITNO NBITS) ERROR-LIST)
      (SETQ READBACK (CC-WRITE-AND-READ REGADR (LOGDPB SET-BIT BITPOS BACKGROUND) MASK))
      (DO ((I 0 (1+ I))
           (PPSS 0001 (+ PPSS 0100))
           (BIT))
          ((>= I NBITS))
        (SETQ BIT (LOGLDB PPSS READBACK))
        (COND ((= I BITNO)
               (OR (= SET-BIT BIT)
                   (PUSH I ERROR-LIST)))
              (T (OR (= BACK-BIT BIT)
                     (CC-FINGER-SUSPECT-BIT I))))))))

(DEFUN CC-FINGER-SUSPECT-BIT (BITNO)
  (OR (NUMERIC-LIST-MEMQ BITNO CC-SUSPECT-BIT-LIST)
      (SETQ CC-SUSPECT-BIT-LIST (CONS BITNO CC-SUSPECT-BIT-LIST))))

;GIVEN A BIT WHICH FAILS, TRY TO PROVE IT IS SHORTED TO SOME OTHER BIT.
;PRINT OUT THE RESULTS AND OUGHT TO REMOVE FROM SUSPECT LIST.              *******
;NOTE THAT FOR NON-COMPLEMENTED TRI-STATE DATA PATHS, 1 SHORTED TO 0 GIVES 0,
;THUS IN THE NORMAL TEST SHORTED BITS LOOK STUCK AT ZERO.
;THIS ONLY TESTS WITH ONES.
(DEFUN CC-TEST-DATA-PATH-SHORTED-BIT (REGADR NBITS BITNO)
  (DO ((BAD-BIT (LOGDPB 1 (+ (LSH BITNO 6) 0001) 0))
        (I 0 (1+ I))
        (TEST-BIT 0001 (+ TEST-BIT 100))
        (BASE 10.)
        (*NOPOINT T)
        (LOSING-BITS NIL))
       ((>= I NBITS)
        (COND ((= (LENGTH LOSING-BITS) 1)
               (NUMERIC-LIST-DELQ (CAR LOSING-BITS) CC-SUSPECT-BIT-LIST)
               (FORMAT T "~&~4TBit ~D is shorted  to bit ~D~%" BITNO (CAR LOSING-BITS)))
              (T
               (FORMAT T "~&~4TBit ~D has problems, can't isolate.~%" BITNO)
               (CC-PRINT-BIT-LIST "Seems as if shorted to bits " LOSING-BITS))))
     (LET ((BOTH-BITS (LOGDPB 1 TEST-BIT BAD-BIT)))
       (COND ((= I BITNO))          ;OF COURSE IT'S SHORTED TO ITSELF!
             ((= BOTH-BITS (CC-WRITE-AND-READ REGADR BOTH-BITS))
              (PUSH I LOSING-BITS))))))

(DEFUN CC-BARF-ABOUT-DATA-PATH (MESSAGE REGADR)
  (FORMAT T "~%~4TTesting register addresses ~O,~%~8TData path is ~A." REGADR MESSAGE))

(DEFUN CC-PRINT-BITS (WD)
  (LET ((CC-SUSPECT-BIT-LIST NIL))  ;KLUDGE
    (CC-PRINT-BIT-LIST NIL (CC-WRONG-BITS-LIST 0 WD (HAULONG WD)))))

(DEFUN CC-PRINT-BIT-LIST (MESSAGE BITLIST)
  (COND (BITLIST
         (IF MESSAGE (SEND TERMINAL-IO ':STRING-OUT MESSAGE))
         (DO ((L (SORT BITLIST #'LESSP) (CDR L))
              (COMMA NIL T)
              (LASTVALUE -2 (CAR L))
              (RANGE-END NIL)
              (RANGE-START))
             ((NULL L)
              (AND RANGE-END
                   (IF (= (1+ RANGE-START) RANGE-END)
                       (FORMAT T ", ~D" RANGE-END))
                   (FORMAT T "-~D" RANGE-END)))
           (COND ((= (CAR L) (1+ LASTVALUE))
                  (OR RANGE-END (SETQ RANGE-START LASTVALUE))
                  (SETQ RANGE-END (CAR L)))
                 (T
                  (AND RANGE-END
                       (IF (= (1+ RANGE-START) RANGE-END)
                           (FORMAT T ", ~D" RANGE-END)
                         (FORMAT T "-~D" RANGE-END)))
                  (SETQ RANGE-END NIL)
                  (AND COMMA (SEND TERMINAL-IO ':STRING-OUT ", "))
                  (FORMAT:ONUM (CAR L)))))
         (SETQ CC-SUSPECT-BIT-LIST
               (NUMERIC-LIST-DIFFERENCE CC-SUSPECT-BIT-LIST BITLIST))
         (SEND TERMINAL-IO ':TYO #\CR))))

;;; CADR ADDRESS TESTS THAT RUN IN THE MACHINE

(DECLARE (SPECIAL CC-MODE-REG CC-DIAG-TRACE))

(COMMENT TEST LOOP STORERS)

;WRITE A-MEMORY, LC HAS ADDRESS SHIFTED INTO DESTINATION FIELD,
;VMA IS ADDED TO LC EACH TIME AROUND THE LOOP, STOP VIA THE STATISTICS COUNTER,
;MD HAS VALUE TO BE STORED, Q-R GETS ADDED TO MD EACH TIME AROUND THE LOOP.
;TO DO THE ADDITIONS WE NEED SOMETHING IN A-MEM.  WE CAUSE IT TO COME
;IN FROM THE PASS-AROUND PATH SO AS NOT TO TRUST THE MEMORY!
;0:     ((OA-REG-LOW) LC)
;1:     ((A-MEM) MD STAT-BIT) ;HALT HERE WHEN DONE
;2:     ((1777@A) Q-R)
;3:     ((MD) ADD MD 1777@A)
;4:     ((1777@A) VMA)
;5:     ((LC) ADD LC 1777@A)
;6:     (JUMP 0)

;THIS VERSION FILLS IT ALL ALTHOUGH IT COULD HAVE MORE PARAMETERS
;BASHES 0@M AS USUAL
;WRONG VALUE IN 0@A BECAUSE THE CODE BASHES 0@M AS IT RUNS AND A=M
(DEFUN CC-FILL-A-MEM (VALUE VALUE-INC UPWARDS-P)
  (CC-EXECUTE (W-C-MEM 0)
        CONS-IR-M-SRC CONS-M-SRC-LC
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-OA-LOW)
  (CC-EXECUTE (W-C-MEM 1)
        CONS-IR-STAT-BIT 1
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-A-MEM-DEST CONS-A-MEM-DEST-INDICATOR)
  (CC-EXECUTE (W-C-MEM 2)
        CONS-IR-M-SRC CONS-M-SRC-Q
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-A-MEM-DEST CONS-A-MEM-DEST-1777)
  (CC-EXECUTE (W-C-MEM 3)
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-A-SRC 1777
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-ADD
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-MD)
  (CC-EXECUTE (W-C-MEM 4)
        CONS-IR-M-SRC CONS-M-SRC-VMA
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-A-MEM-DEST CONS-A-MEM-DEST-1777)
  (CC-EXECUTE (W-C-MEM 5)
        CONS-IR-M-SRC CONS-M-SRC-LC
        CONS-IR-A-SRC 1777
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-ADD
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-LC)
  (CC-EXECUTE (W-C-MEM 6)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-JUMP-ADDR 0
        CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
        CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 7)       ;SO HAS GOOD PARITY
        CONS-IR-OP CONS-OP-JUMP)
  (CC-WRITE-STAT-COUNTER -1024.) ;STOP AFTER WRITING 1024. LOCATIONS
  (COND (UPWARDS-P
         (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC 0)       ;FIRST ADDRESS, SHIFTED OVER
         (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-VMA 1_14.)  ;ADDRESS INCREMENT (MAGIC NUMBER)
         (CC-WRITE-Q VALUE-INC)
         (CC-WRITE-MD VALUE))
        (T
         (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC 1777_14.);FIRST ADDRESS, SHIFTED OVER
         (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-VMA -1_14.) ;ADDRESS INCREMENT (MAGIC NUMBER)
         (CC-WRITE-Q (- VALUE-INC))
         (CC-WRITE-MD (+ VALUE (* 2000 VALUE-INC)))))
  (CC-RUN-TEST-LOOP 0))

(DECLARE (SPECIAL SPY-MODE SPY-CLK SPY-FLAG-1) (FIXNUM (SPY-READ FIXNUM)))

(DEFUN CC-RUN-TEST-LOOP (ADR)
  (CC-WRITE-PC ADR)
  (CC-NOOP-CLOCK)               ;FIRST INSTRUCTION TO IR
  (CC-CLOCK)                    ;CLOCK AGAIN
  (SPY-WRITE SPY-MODE (LOGIOR CC-MODE-REG 10))  ;ENABLE STAT HALT
  (SPY-WRITE SPY-CLK 1) ;TAKE OFF
  (DO () ((ZEROP (BOOLE 1 4000 (SPY-READ SPY-FLAG-1))))
    (COND ((KBD-TYI-NO-HANG) (BREAK CC-RUN-TEST-LOOP))
          (T (PROCESS-SLEEP 15. "Await Stat Halt"))))    ;AWAIT STAT HALT
  )
```

```
;SCAN A-MEMORY, LC HAS ADDRESS SHIFTED INTO SOURCE FIELD,
;VMA IS ADDED TO LC EACH TIME AROUND THE LOOP, STOP VIA THE STATISTICS COUNTER,
;MD HAS VALUE TO BE CHECKED FOR, Q-R GETS ADDED TO MD EACH TIME AROUND THE LOOP.
;TO DO THE ADDITIONS WE NEED SOMETHING IN A-MEM.  WE CAUSE IT TO COME
;IN FROM THE PASS-AROUND PATH SO AS NOT TO TRUST THE MEMORY!
;WE BASH 0@A SINCE IT LOSES ANYWAY.
;HALT BY GOING INTO A LOOP WITH STAT-BIT ON IF COMPARE FAILS, GOOD DATA IN MD,
;BAD DATA IN 0@M.
;0:     ((OA-REG-HIGH) LC)
;1:     ((0@M) 0@A STAT-BIT) ;HALT HERE WHEN DONE, C(A) TO 0@A, 0@M, L
;2:     (JUMP-NOT-EQUAL MD 0@A 10)
;3:     ((0@A) Q-R)
;4:     ((MD) ADD MD 0@A)
;5:     ((0@A) VMA)
;6:     ((LC) ADD LC 0@A)
;7:     (JUMP 0)
;10:    (JUMP 10 STAT-BIT)      ;HALT HERE IF ERROR

;SCAN OUT A-MEMORY FROM 2@A THROUGH 1777@A, RETURN A LIST OF MISMATCHES
;IN THE FORM ((ADDR GOOD BAD) ...)
;BASHES 0@M AS USUAL.  0@A IS KNOWN TO BE BAD.
;WRITING INTO CONTROL MEMORY BASHES 1@A, SO WE DON'T SCAN THAT EITHER.
(DEFUN CC-SCAN-A-MEM (VALUE VALUE-INC)
  (CC-EXECUTE (W-C-MEM 0)
        CONS-IR-M-SRC CONS-M-SRC-LC
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-OA-HIGH)
  (CC-EXECUTE (W-C-MEM 1)
        CONS-IR-STAT-BIT 1
        CONS-IR-A-SRC 0
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETA
        CONS-IR-M-MEM-DEST 0)
  (CC-EXECUTE (W-C-MEM 2)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-A-SRC 0
        CONS-IR-JUMP-ADDR 10
        CONS-IR-JUMP-COND CONS-JUMP-COND-M-NEQ-A
        CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 3)
        CONS-IR-M-SRC CONS-M-SRC-Q
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-A-MEM-DEST CONS-A-MEM-DEST-INDICATOR)
  (CC-EXECUTE (W-C-MEM 4)
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-A-SRC 0
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-ADD
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-MD)
  (CC-EXECUTE (W-C-MEM 5)
        CONS-IR-M-SRC CONS-M-SRC-VMA
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-A-MEM-DEST CONS-A-MEM-DEST-INDICATOR)
  (CC-EXECUTE (W-C-MEM 6)
        CONS-IR-M-SRC CONS-M-SRC-LC
        CONS-IR-A-SRC 0
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-ADD
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-LC)
  (CC-EXECUTE (W-C-MEM 7)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-JUMP-ADDR 0
        CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
        CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 10)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-JUMP-ADDR 10
        CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
        CONS-IR-N 1
        CONS-IR-STAT-BIT 1)
  (DO ((ADDRESS 2)      ;LOOP REPEATS EACH TIME MACHINE HALTS
       (LOC) (GOOD) (BAD)
       (ERRORS NIL))
      (())
    (DECLARE (FIXNUM ADDRESS))
    (CC-WRITE-STAT-COUNTER (- ADDRESS 1024.))    ;NUMBER OF LOCATIONS YET TO SCAN
    (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC (LSH ADDRESS 6))      ;FIRST ADDRESS, SHIFTED OVER
    (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-VMA 1_6) ;ADDRESS INCREMENT (MAGIC NUMBER)
    (CC-WRITE-Q VALUE-INC)
    (CC-WRITE-MD (+ VALUE (* VALUE-INC ADDRESS)))
    (CC-RUN-TEST-LOOP 0)                        ;RUN UNTIL DONE OR ERROR
    (AND (= (CC-READ-PC) 3)                     ;NORMAL HALT, DONE
          (RETURN (NREVERSE ERRORS)))
     (SETQ ADDRESS (1+ (logand 7777
                               (ASH (CC-READ-M-MEM CONS-M-SRC-LC)
                                -6)))  ;NEXT ADDRESS TO DO
           LOC (1- ADDRESS)
           GOOD (CC-READ-M-MEM CONS-M-SRC-MD)
           BAD (CC-READ-M-MEM 0))
     (AND CC-DIAG-TRACE (PRINT (LIST 'LOC LOC 'GOOD GOOD 'BAD BAD)))
     (SETQ ERRORS (CONS (LIST LOC GOOD BAD) ERRORS))))

(DEFtest CC-ADDRESS-TEST-A-MEM () "A-MEM"
  (DO ((SHIFT 0 (1+ SHIFT))
       (ADDEND (+ 1 (LSH 1 10.) (LSH 1 20.) (LSH 1 30.))
               (+ ADDEND ADDEND))
       (TEM))
      ((= SHIFT 10.))
    (DECLARE (FIXNUM SHIFT ADDEND))
    (CC-FILL-A-MEM 0 ADDEND T)                    ;FILL UPWARDS WITH ADDRESS
    (COND ((SETQ TEM (CC-SCAN-A-MEM 0 ADDEND))   ;SCAN FOR ERRORS
           (CC-FILL-A-MEM 0 ADDEND NIL)           ;GOT ERROR, FILL DOWNWARDS
           (CC-ADDRESS-TEST-ANALYZE TEM (CC-SCAN-A-MEM 0 ADDEND) SHIFT NIL))) ;TELL RESULTS
    (CC-FILL-A-MEM -1 (- ADDEND) T)               ;FILL UPWARDS WITH COMPLEMENT OF ADDRESS
    (COND ((SETQ TEM (CC-SCAN-A-MEM -1 (- ADDEND)))      ;SCAN FOR ERRORS
           (CC-FILL-A-MEM -1 (- ADDEND) NIL)     ;GOT ERROR, FILL DOWNWARDS
           (CC-ADDRESS-TEST-ANALYZE TEM (CC-SCAN-A-MEM -1 (- ADDEND))
                                    SHIFT T)))))         ;TELL RESULTS

;THIS COULD BE MUCH HAIRIER
(DEFUN CC-ADDRESS-TEST-ANALYZE (UPWARD-ERRORS DOWNWARD-ERRORS SHIFT COMPLEMENT-P)
  SHIFT COMPLEMENT-P ; Not used
  (DO ((L (NCONC UPWARD-ERRORS DOWNWARD-ERRORS) (CDR L))
       (ADDRESS-AND -1)
       (ADDRESS-IOR 0)
       (DATA-BITS-IN-ERROR 0))
      ((NULL L)
       (FORMAT T "Address AND ~O , address IOR ~O, data bits in error ~O~%"
               ADDRESS-AND ADDRESS-IOR DATA-BITS-IN-ERROR))
    (DECLARE (FIXNUM ADDRESS-AND ADDRESS-IOR DATA-BITS-IN-ERROR)) ;NOT TESTING C-MEM
    (SETQ ADDRESS-AND (LOGAND (CAAR L) ADDRESS-AND)
          ADDRESS-IOR (LOGIOR (CAAR L) ADDRESS-IOR)
          DATA-BITS-IN-ERROR (LOGIOR (LOGXOR (CADAR L) (CADDAR L)) DATA-BITS-IN-ERROR))))

;Fast address test writes zeros and ones into 2 locations
;whose addresses differ in 1 bit, checks for interference.
;This detects address bits stuck at zero or one for some data
;bits, but does not detect adjacent address bits shorted together.
(DEFtest CC-FAST-ADDRESS-TEST (MEM-NAME REGADR N-DATA-BITS N-ADDRESS-BITS) "Fast Address"
  (DECLARE (FIXNUM REGADR N-DATA-BITS N-ADDRESS-BITS))
  (DO ((N (COND ((EQ MEM-NAME 'C-MEM) 2)  ;C-MEM MAY NOT BE A POWER OF 2. CROCK.
                (T 4))
          (1- N))
       (PHASE 0 (1+ PHASE))
       (ONES (SUB1 (EXPT 2 N-DATA-BITS)))
       (ADR-MASK (1- (EXPT 2 N-ADDRESS-BITS)))
       (ZEROS 0))
      ((= N 0))
    (DO ((BITNO 0 (1+ BITNO))
         (GOOD1 (COND ((EVENP PHASE) ZEROS) (T ONES)))
         (GOOD2 (COND ((EVENP PHASE) ONES) (T ZEROS)))
         (BAD1)
         (BAD2)
         (BAD3)
         (OTHER-LOC)
         (K)
         (CC-SUSPECT-BIT-LIST))
        ((= BITNO N-ADDRESS-BITS))
      (SETQ K (+ REGADR (COND ((< PHASE 2)
                               (LSH 1 BITNO))
                              (T (LOGXOR ADR-MASK (LSH 1 BITNO))))))
      (SETQ OTHER-LOC (COND ((< PHASE 2) REGADR)
                            (T (+ REGADR ADR-MASK))))
      (CC-R-D K GOOD2)
      (COND ((NOT (EQUAL (SETQ BAD2 (CC-R-E K)) GOOD2))
             (FORMAT T "~4T~A loc ~O" MEM-NAME (- K REGADR))
             (CC-PRINT-BIT-LIST " fails in data bits "
                                (CC-WRONG-BITS-LIST GOOD2 BAD2 N-DATA-BITS))))
      (CC-R-D OTHER-LOC GOOD1)                   ;Deposit in loc 0 second for A & M's sake
      (COND ((NOT (EQUAL (SETQ BAD1 (CC-R-E OTHER-LOC)) GOOD1))
             (FORMAT T "~4T~A LOC ~O" MEM-NAME (- OTHER-LOC REGADR))
             (CC-PRINT-BIT-LIST " fails in data bits "
                                (CC-WRONG-BITS-LIST GOOD1 BAD1 N-DATA-BITS))))
      (COND ((NOT (EQUAL (SETQ BAD3 (CC-R-E K)) GOOD2))
             (FORMAT T "~A address bit ~D (~O and ~O)"
                     MEM-NAME BITNO (- K REGADR) (- OTHER-LOC REGADR))
             (CC-PRINT-BIT-LIST (IF (EVENP PHASE) " fails storing 1's then 0 in data bits "
                                    " fails storing 0 then 1's in data bits ")
                                (CC-WRONG-BITS-LIST GOOD2 BAD3 N-DATA-BITS)))))))
```

```
(DEFUN CC-QUIET-ADDRESS-TEST (MEM-NAME REGADR N-DATA-BITS N-ADDRESS-BITS)
  (DECLARE (FIXNUM REGADR N-DATA-BITS N-ADDRESS-BITS))
  (DO ((N (COND ((EQ MEM-NAME 'C-MEM) 2)  ;C-MEM MAY NOT BE A POWER OF 2. CROCK.
                (T 4))
          (1- N))
       (PHASE 0 (1+ PHASE))
       (ONES (SUB1 (EXPT 2 N-DATA-BITS)))
       (ADR-MASK (1- (EXPT 2 N-ADDRESS-BITS)))
       (ZEROS 0))
      ((= N 0))
    (DO ((BITNO 0 (1+ BITNO))
         (GOOD (COND ((EVENP PHASE) ONES) (T ZEROS)))
         (OTHER-LOC)
         (K)
         (CC-SUSPECT-BIT-LIST))
        ((= BITNO N-ADDRESS-BITS))
      (SETQ K (+ REGADR (COND ((< PHASE 2)
                               (LSH 1 BITNO))
                              (T (LOGXOR ADR-MASK (LSH 1 BITNO))))))
      (SETQ OTHER-LOC (COND ((< PHASE 2) REGADR)
                            (T (+ REGADR ADR-MASK))))
      (CC-R-D K GOOD))))

;Test all bits of memory for ability to retain 0's, 1's. Then try 0's in
; even addresses, 1's in odd ones.
(DEFUN CC-GROSS-DATA-TEST (MEM-NAME REGADR N-DATA-BITS N-ADDRESS-BITS
                           &OPTIONAL (MAX-ERRORS 5.) &AUX CC-SUSPECT-BIT-LIST)
 (*CATCH 'EXIT
  (DO ((N 3 (1- N))
       (ONES (SUB1 (EXPT 2 N-DATA-BITS)))
       (ZEROS 0)
       (HIADR (+ REGADR (EXPT 2 N-ADDRESS-BITS)))
       (ERRORS 0))
      ((= N 0))
    (DO ((ADR REGADR (+ ADR 2))
         (EVEN-DATA (COND ((= N 2) ZEROS)
                          ((= N 1) ONES)
                          (T ZEROS)))
         (ODD-DATA (COND ((= N 2) ZEROS)
                         ((= N 1) ONES)
                         (T ONES))))
        ((>= ADR HIADR)
         (DO ((ADR REGADR (+ ADR 2))
              (TEM))
             ((>= ADR HIADR))
           (COND ((NOT (= (SETQ TEM (CC-R-E ADR)) EVEN-DATA))
                  (FORMAT T "~%Wrote ~O in locn ~O of ~O, read ~S losing bits "
                          EVEN-DATA (- ADR REGADR) MEM-NAME TEM)
                  (CC-PRINT-BIT-LIST NIL (CC-WRONG-BITS-LIST EVEN-DATA TEM N-DATA-BITS))
                  (COND ((> (SETQ ERRORS (1+ ERRORS)) MAX-ERRORS)
                         (*THROW 'EXIT NIL)))))
           (COND ((NOT (= (SETQ TEM (CC-R-E (1+ ADR))) ODD-DATA))
                  (FORMAT T "~%Wrote ~O in locn ~O of ~O, read ~S losing bits"
                          ODD-DATA (1+ (- ADR REGADR)) MEM-NAME TEM)
                  (CC-PRINT-BIT-LIST NIL (CC-WRONG-BITS-LIST ODD-DATA TEM N-DATA-BITS))
                  (COND ((> (SETQ ERRORS (1+ ERRORS)) MAX-ERRORS)
                         (*THROW 'EXIT NIL)))))))
      (CC-R-D ADR EVEN-DATA)
      (CC-R-D (1+ ADR) ODD-DATA)))))

(DEFUN CC-WRONG-BITS-LIST (GOOD BAD N-DATA-BITS)
  (DO ((BITNO 0 (1+ BITNO))
       (PPSS 0001 (+ 100 PPSS))
       (L NIL))
      ((= BITNO N-DATA-BITS) L)
    (OR (= (LOGLDB PPSS GOOD) (LOGLDB PPSS BAD))
        (SETQ L (CONS BITNO L)))))

(DEFtest CC-TEST-SPC-POINTER () "SPC Pointer"
  (PROG (USP READ GOOD)
        (SETQ USP (CC-READ-MICRO-STACK-PTR))
        (DOTIMES (C 32.)
          (CC-EXECUTE (WRITE)
                      CONS-IR-M-SRC CONS-M-SRC-MD        ;PUSH IT
                      CONS-IR-ALUF CONS-ALU-SETM
                      CONS-IR-OB CONS-OB-ALU
                      CONS-IR-FUNC-DEST CONS-FUNC-DEST-MICRO-STACK-PUSH)
          (SETQ READ (CC-READ-MICRO-STACK-PTR))
          (COND ((NOT (= (SETQ GOOD (LOGAND 37 (+ (1+ C) USP))) READ))
                 (FORMAT T "~%SPC PTR INCREMENT FAILED, WAS ~O, SHOULD BE ~O" READ GOOD))))
        (SETQ USP (CC-READ-MICRO-STACK-PTR))
        (DOTIMES (C 32.)
           (CC-EXECUTE (WRITE)
                       CONS-IR-M-SRC CONS-M-SRC-MICRO-STACK-POP
                       CONS-IR-ALUF CONS-ALU-SETM
                       CONS-IR-OB CONS-OB-ALU)
           (SETQ READ (CC-READ-MICRO-STACK-PTR))
           (COND ((NOT (= (SETQ GOOD (LOGAND 37 (- USP (1+ C)))) READ))
                  (FORMAT T "~%SPC PTR DECREMENT FAILED, WAS ~O, SHOULD BE ~O" READ GOOD))))
        ))

(DEFUN CC-WRITE-ZERO-SPC (&OPTIONAL (V 0))
  (DO ()((KBD-TYI-NO-HANG))
    (CC-WRITE-MD V)      ;GET DATA INTO MRD
    (CC-EXECUTE (WRITE)
                CONS-IR-M-SRC CONS-M-SRC-MD      ;PUSH IT
                CONS-IR-ALUF CONS-ALU-SETM
                CONS-IR-OB CONS-OB-ALU
                CONS-IR-FUNC-DEST CONS-FUNC-DEST-MICRO-STACK-PUSH)))
```

```
;;; CADR SHIFTER TEST                                              -*-LISP-*-

(DECLARE (FIXNUM SPY-IR-LOW (SPY-READ FIXNUM))
         (NOTYPE (SPY-WRITE FIXNUM FIXNUM))
         (SPECIAL SPY-IR-LOW)
         (*EXPR SPY-READ SPY-WRITE))

(DEFtest CC-TEST-SPY-IR () "Spy IR"
  (DOLIST (PART '(SPY-IR-HIGH SPY-IR-MED SPY-IR-LOW))
    (DOLIST (BACKGROUND '(0 177777))
      (DO ((I 0 (1+ I))
           (BIT 1 (ASH BIT 1)))
          ((≥ I 16.))
        (LET ((PATTERN (LOGXOR BIT BACKGROUND)))
         (SPY-WRITE (SYMEVAL PART) PATTERN)
         (CC-NOOP-DEBUG-CLOCK)
         (LET ((ACTUAL (SPY-READ (SYMEVAL PART))))
           (OR (= ACTUAL PATTERN)
               (FORMAT T "~&SPY-IR - Wrote: ~O, Read: ~O" PATTERN ACTUAL))))))))

(DEFMACRO ADD2L (ITEM LIST)
  `(OR (NUMERIC-LIST-MEMQ ,ITEM ,LIST)
       (SETQ ,LIST (CONS ,ITEM ,LIST))))


(DECLARE (SPECIAL CC-SUSPECT-BIT-LIST))
(DEFtest CC-TEST-SHIFTER () "Shifter"
  "Algorithm is to shift floating ones and zeros with all possible shifts.
Record bits that failed at shifter input, at shifter output, between
the two shifter stages, and also which shift counts fail.  Note that
if the masker proms aren't plugged in, selecting the 32-bit-wide byte
will work anyway due to pullups.  Prom problems will show up as failure
of particular bits at the shifter output, you can try unplugging the
offending prom.  To reduce randomness we bring 0 in
on the A-source.  This is now written so that it works whether or
not proms are present, it addresses 0 in the right mask which is all 1's
and 37 in the left mask which is also all 1's."
  (CC-WRITE-A-MEM 2 0)
  (DO ((INPUT-ERRONEOUS-ZEROS NIL)
       (MIDDLE-ERRONEOUS-ZEROS NIL)
       (OUTPUT-ERRONEOUS-ZEROS NIL)
       (INPUT-ERRONEOUS-ONES NIL)
       (MIDDLE-ERRONEOUS-ONES NIL)
       (OUTPUT-ERRONEOUS-ONES NIL)
       (ERRONEOUS-SHIFT-COUNTS NIL)
       (CC-SUSPECT-BIT-LIST NIL)
       (BITNO 0 (1+ BITNO))) ;THE FLOATING BIT
      ((= BITNO 32.)
       (TERPRI)
       (CC-PRINT-BIT-LIST "Shift counts with erroneous bits: " ERRONEOUS-SHIFT-COUNTS)
       (CC-PRINT-BIT-LIST "M bits with erroneous zeros: " INPUT-ERRONEOUS-ZEROS)
       (CC-PRINT-BIT-LIST "SA bits with erroneous zeros: " MIDDLE-ERRONEOUS-ZEROS)
       (CC-PRINT-BIT-LIST "R bits with erroneous zeros: " OUTPUT-ERRONEOUS-ZEROS)
       (CC-PRINT-BIT-LIST "M bits with erroneous ones: " INPUT-ERRONEOUS-ONES)
       (CC-PRINT-BIT-LIST "SA bits with erroneous ones: " MIDDLE-ERRONEOUS-ONES)
       (CC-PRINT-BIT-LIST "R bits with erroneous ones: " OUTPUT-ERRONEOUS-ONES))
    (DO ((BACKGROUND 37777777777 0))  ;FIRST FLOATING ZEROS, THEN FLOATING ONES
        (())
      (DECLARE (FIXNUM BACKGROUND))
      (CC-WRITE-MD (LOGXOR BACKGROUND (ASH 1 BITNO)))  ;SHIFTER INPUT
      (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE       ;INST TO SHIFT BY 0 INTO IR
                  CONS-IR-A-SRC 2
                  CONS-IR-M-SRC CONS-M-SRC-MD
                  CONS-IR-BYTL-1 37
                  CONS-IR-MROT 0
                  CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-LDB) ;LDB = SR, NOT MR
      (DO ((MROT 0 (1+ MROT))
           (BAD)
           (CORRECT-IR (SPY-READ SPY-IR-LOW) (1+ CORRECT-IR))
           (GOOD (LOGXOR BACKGROUND (ASH 1 BITNO)) ;EXPECTED OUTPUT
                 (ROT32 GOOD 1)))
          ((= MROT 32.))
        (DECLARE (FIXNUM MROT GOOD BAD))
        (COND ((NOT (= (SETQ BAD (CC-READ-OBUS)) GOOD)) ;HA! AN ERROR, STASH STUFF AWAY
               (IF-FOR-LISPM
                (COND (CC-DIAG-TRACE
                       (FORMAT T "~&Rot: ~O, Bit: ~O, Good: ~O, Bad: ~O, Reread: ~O"
                               MROT (ASH 1 BITNO) GOOD BAD (CC-READ-OBUS))) ))
               (ADD2L MROT ERRONEOUS-SHIFT-COUNTS)
               (DO ((J 0 (1+ J))                  ;BITS OF OUTPUT
                    (GOOD GOOD (ASH GOOD -1))
                    (BAD BAD (ASH BAD -1)))
                   ((= J 32.))
                 (OR (= (LOGAND 1 GOOD) (LOGAND 1 BAD))
                     (COND ((ZEROP (LOGAND 1 GOOD))  ;AN ERRONEOUS ONE
                                (ADD2L J OUTPUT-ERRONEOUS-ONES)
                                (ADD2L (LOGAND (- J MROT) 37) INPUT-ERRONEOUS-ONES)
                                (ADD2L (LOGAND (- J (LOGAND MROT -4)) 37) MIDDLE-ERRONEOUS-ONES))
                               (T
                                (ADD2L J OUTPUT-ERRONEOUS-ZEROS)
                                (ADD2L (LOGAND (- J MROT) 37) INPUT-ERRONEOUS-ZEROS)
                                (ADD2L (LOGAND (- J (LOGAND MROT -4)) 37) MIDDLE-ERRONEOUS-ZEROS)
                                ))))))
           (SPY-WRITE SPY-IR-LOW (1+ (SPY-READ SPY-IR-LOW)))        ;INCREMENT MROT FIELD
           (CC-NOOP-DEBUG-CLOCK)
           (LET ((ACTUAL-IR (SPY-READ SPY-IR-LOW)))        ;Did the IR get written correctly?
             (COND ((NOT (= (1+ CORRECT-IR) ACTUAL-IR))
                    (FORMAT T "~&Debug IR -  Correct: ~O, Read back: ~O"
                              (1+ CORRECT-IR) ACTUAL-IR)))))
         (AND (ZEROP BACKGROUND) (RETURN NIL)))))

;; With the shift data paths known to work, read out all elements of the left
;; mask and verify that they contain the correct contents.  We continue to
;; select location 0 of the right mask, which is all 1's.
;; It may be helpful to pull out the right-mask proms at this stage.
(DEFUN CC-TEST-MASK-LEFT ()
  (CC-WRITE-A-MEM 1 0)
  (CC-WRITE-M-MEM 2 37777777777)
  ((LAMBDA (TEM)
      (DECLARE (FIXNUM TEM))
      (SETQ TEM (CC-READ-A-MEM 1))
      (OR (= 0 TEM)
          (ERROR '|in 1@A - should be 0| TEM 'FAIL-ACT))
      (SETQ TEM (CC-READ-M-MEM 2))
      (OR (= 37777777777 TEM)
          (ERROR '|in 2@M - should be 37777777777| TEM 'FAIL-ACT))
      (DO ((BYTL-1 0 (1+ BYTL-1))
           (GOOD 1 (1+ (ASH GOOD 1))))
          ((= BYTL-1 32.))
        (DECLARE (FIXNUM BYTL-1 GOOD))
        (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE
                    CONS-IR-A-SRC 1
                    CONS-IR-M-SRC 2
                    CONS-IR-BYTL-1 BYTL-1
                    CONS-IR-MROT 0
                    CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-LDB)  ;LDB = SR, NO MR
        (SETQ TEM (CC-READ-OBUS))
        (AND (≠ TEM GOOD)
             (FORMAT T "~%BYTL-1=~O, MROT=~O, Left Mask=~O, should be ~O~%"
                     BYTL-1 MROT TEM GOOD))))
;Paren above, far right should match with →(lambda (tem)
   0))

;; With the shift data paths and the left mask known to work, read out
;; all locations of the right mask and verify that they are correct.
;; Here we hold the left mask at all 1's, which incidentally tests its
;; address adder.
(DEFUN CC-TEST-MASK-RIGHT ()
  (CC-WRITE-A-MEM 1 0)
  (CC-WRITE-M-MEM 2 37777777777)
  ((LAMBDA (TEM)
      (DECLARE (FIXNUM TEM))
      (SETQ TEM (CC-READ-A-MEM 1))
      (OR (= 0 TEM)
          (ERROR '|in 1@A - should be 0| TEM 'FAIL-ACT))
      (SETQ TEM (CC-READ-M-MEM 2))
      (OR (= 37777777777 TEM)
          (ERROR '|in 2@M - should be 37777777777| TEM 'FAIL-ACT))
      (DO ((MROT 0 (1+ MROT)) ;right mask address
           (BYTL-1 37 (1- BYTL-1)) ;keeps the left mask address = 37
           (GOOD 37777777777 (LOGXOR GOOD (ASH 1 MROT))))
          ((= MROT 32.))
        (DECLARE (FIXNUM MROT BYTL-1 GOOD))
        (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE
                    CONS-IR-A-SRC 1
                    CONS-IR-M-SRC 2
                    CONS-IR-BYTL-1 BYTL-1
                    CONS-IR-MROT MROT
                    CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)  ;MR, NO SR
        (SETQ TEM (CC-READ-OBUS))
        (AND (≠ TEM GOOD)
             (FORMAT T "~%BYTL-1=~O, MROT=~O, Right Mask=~O, should be ~O~%"
                     BYTL-1 MROT TEM GOOD))))
;Paren above, far right should match with →(lambda (tem)
   0))

;; Verify that the masker works.  This finds things like broken wires on
;; the mask inputs to the 9S42's.
;; The somewhat simple-minded algorithm is to make the masker select all M
;; and make sure no bits from A get OR'ed in, then select all A and make sure
;; no bits from M get OR'ed in.
(DEFtest CC-TEST-MASKER () "Masker"
  (LET ((CC-SUSPECT-BIT-LIST NIL))
    (CC-WRITE-A-MEM 1 37777777777)
    (CC-WRITE-M-MEM 2 0)
    (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE
                CONS-IR-A-SRC 1
                CONS-IR-M-SRC 2
                CONS-IR-BYTL-1 37
                CONS-IR-MROT 0
                CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)
    (CC-PRINT-BIT-LIST "Erroneous A bits coming through masker:"
                       (CC-WRONG-BITS-LIST 0 (CC-READ-OBUS) 32.))
    (LET ((RH 0) (LH 0))
      (DECLARE (FIXNUM LH RH))
      (CC-WRITE-A-MEM 1 0)
      (CC-WRITE-M-MEM 2 37777777777)
      (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE       ;Select A in the right half
                  CONS-IR-A-SRC 1
                  CONS-IR-M-SRC 2
                  CONS-IR-BYTL-1 17
                  CONS-IR-MROT 20
                  CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)
      (SETQ RH (CC-READ-OBUS))
      (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE       ;Select A in the left half
                  CONS-IR-A-SRC 1
                  CONS-IR-M-SRC 2
                  CONS-IR-BYTL-1 17
                  CONS-IR-MROT 0
                  CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)
      (SETQ LH (CC-READ-OBUS))
      (CC-PRINT-BIT-LIST "Erroneous M bits coming through masker:"
                         (CC-WRONG-BITS-LIST 0 (DPB (LDB 2020 LH) 2020 RH) 32.)))))

;; With the normal shift and mask logic known to work, test LC-modification.
;; Things to test are whether both halfwords and all 4 bytes properly mung
;; the MROT field.  Doesn't currently test whether automatic fetching.
;; Does test LC incrementing.  Eventually that should be tested.
;; Should test LC → VMA data path.
(DEFUN CC-TEST-LC-AFFECTS-SHIFT ()
  (CC-WRITE-A-MEM 1 0)
  (CC-WRITE-M-MEM 2 37777777777)
  (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-INT-CNTRL 1_29.) ;Put machine in byte mode
  (DO ((LC 1 (1+ LC))
       (LC-READBACK (+ 1_31. 1_29. 1) (1+ LC-READBACK)) ;Needfetch, Byte Mode, 1
       (GOOD 377 (ASH GOOD 8))
       (TEM))
      ((= LC 5))
    (DECLARE (FIXNUM LC LC-READBACK GOOD TEM))
    (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC LC) ;Select byte (initially rightmost, LC=current+1)
    (SETQ TEM (CC-READ-M-MEM CONS-M-SRC-LC))
    (IF (≠ TEM LC-READBACK)
        (FORMAT T "~&~4TWrong value in LC, is ~O, but should be ~O" TEM LC-READBACK))
    (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE
                CONS-IR-A-SRC 1
                CONS-IR-M-SRC 2
                CONS-IR-BYTL-1 7
                CONS-IR-MROT 0
                CONS-IR-MF 3
                CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)  ;MR, NO SR
    (SETQ TEM (CC-READ-OBUS))
    (IF (≠ TEM GOOD)
        (FORMAT T "~&~4T LC=~O (byte mode), shifter output=~O, should be ~O"
                LC-READBACK TEM GOOD)))
  (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-INT-CNTRL 0_29.) ;Put machine in word mode
  (DO ((LC 2 (+ LC 2))
       (LC-READBACK (+ 1_31. 2) (+ LC-READBACK 2)) ;Needfetch, no Byte Mode, 2 (=1 wd)
       (GOOD 177777 (ASH GOOD 16.))
       (TEM))
      ((= LC 4))
    (DECLARE (FIXNUM LC LC-READBACK GOOD TEM))
    (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC LC) ;Select halfword (initially rightmost, LC=current+1)
    (SETQ TEM (CC-READ-M-MEM CONS-M-SRC-LC))
    (IF (≠ TEM LC-READBACK)
        (FORMAT T "~%Wrong value in LC, is ~O, but should be ~O" TEM LC-READBACK))
    (CC-EXECUTE CONS-IR-OP CONS-OP-BYTE
                CONS-IR-A-SRC 1
                CONS-IR-M-SRC 2
                CONS-IR-BYTL-1 17
                CONS-IR-MROT 0
                CONS-IR-MF 3
                CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-SELECTIVE-DEPOSIT)  ;MR, NO SR
    (SETQ TEM (CC-READ-OBUS))
    (AND (NOT (= TEM GOOD))
         (FORMAT T "~%LC=~O (halfword mode), shifter output=~O, should be ~O"
                 LC-READBACK TEM GOOD)))
  (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-INT-CNTRL 1_29.) ;Put machine in byte mode
  (DOTIMES (B 24.)
    (LET ((GOOD (ASH 1 B))
          (TEM NIL))
      (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-LC (1- GOOD))
```

```
       (CC-SAVE-MICRO-STACK)
       (SETQ CC-SAVED-MICRO-STACK-PTR 0)
       (AS-1 40000 CC-MICRO-STACK 0)
       (CC-RESTORE-MICRO-STACK)
       (CC-EXECUTE (WRITE)
             CONS-IR-OP CONS-OP-JUMP
             CONS-IR-R 1
             CONS-IR-JUMP-COND CONS-JUMP-COND-UNC)
       (COND ((NOT (= (SETQ TEM (LOGAND 77777777 (CC-READ-M-MEM CONS-M-SRC-LC)))
                      GOOD))
              (FORMAT T "~%LC failed to increment properly good ~O, bad ~O" GOOD TEM)))))
)

;;; CADR DISPATCH TEST

;; Fill all of D memory with its own address, and no RPN bits
(DEFUN CC-FILL-D-MEM-W-ADR ()
  (DO ((I 0 (1+ I)))
      ((= I 2048.))
    (DECLARE (FIXNUM I))
    (CC-WRITE-D-MEM I I)))

;; Read back all possible bytes with MROT=0, make sure right address
;; comes back into the PC.  Here we always use a disp addr of 0.
(DEFtest CC-TEST-DISPATCH (&aux tem) "Dispatch"
  (CC-FILL-D-MEM-W-ADR)
  (DO ((BYTL 0 (1+ BYTL))
       (MXVAL 1 (* MXVAL 2))
       (OK-CNT 0)
       (ERR-CNT 0))
      ((= BYTL 8.)
       (COND ((NOT (ZEROP ERR-CNT))
              (FORMAT T "~%~S TRIALS OK" OK-CNT))))
    (DECLARE (FIXNUM BYTL MXVAL))
    (DO ((VAL 0 (1+ VAL))
         (PC))
        ((= VAL MXVAL))
      (DECLARE (FIXNUM VAL PC))
      (CC-WRITE-MD (- VAL MXVAL)) ;Turn on extra bits to detect improper masking
      (CC-EXECUTE CONS-IR-OP CONS-OP-DISPATCH   ;Execute a dispatch
                  CONS-IR-M-SRC CONS-M-SRC-MD
                  CONS-IR-DISP-BYTL BYTL
                  CONS-IR-DISP-ADDR 0)
              ;At this point the disp is in IR but has not yet been executed.
     (CC-CLOCK)                          ;Clock it so PC loads from disp mem
     (SETQ PC (CC-READ-PC))
     (IF (= PC VAL)    ; Read the right location
         (INCF OK-CNT)
       (INCF ERR-CNT)  ; Read the wrong location [Else clauses]
       (FORMAT T "~%Dispatch error, BYTL=~O, M=~O, DPC=~O, but should be ~O"
               BYTL
               (LOGAND 37777777777 (- VAL MXVAL))
               PC
               VAL))))
  (cc-write-d-mem 0 (dpb 1 cons-disp-p-bit (dpb 1 cons-disp-n-bit 0)))
  (FORMAT T "~& Testing out stack (USP).~%")
  (do ((cnt 0 (1+ cnt))
       (adr 0 (lsh 1 cnt)))
      ((= adr 20000))
    (cc-execute (w-c-mem adr)
                cons-ir-op cons-op-dispatch
                cons-ir-disp-lpc 1
                cons-ir-disp-bytl 0
                cons-ir-disp-addr 0)
    (cc-save-micro-stack)
    (setq cc-saved-micro-stack-ptr 0)
    (as-1 -1 cc-micro-stack 0)
    (as-1 -1 cc-micro-stack 1)
    (cc-restore-micro-stack)
    (cc-write-pc adr)
    (cc-noop-clock)     ;dispatch inst to IR
    (cc-clock)          ;execute it
    (cc-noop-clock)     ;write spc
    (cc-save-micro-stack)
    (cond ((not (= cc-saved-micro-stack-ptr 1))
           (format t "~%Dispatch push failed to advance USP ~O" cc-saved-micro-stack-ptr)))
    (cond ((not (= (setq tem (ar-1 cc-micro-stack 1)) adr))
           (format t "~%Dispatch push own address at adr ~O pushed ~O instead" adr tem)))
    (cc-execute (w-c-mem adr)
                cons-ir-op cons-op-dispatch
                cons-ir-disp-bytl 0
                cons-ir-disp-addr 0)
    (cc-save-micro-stack)
    (setq cc-saved-micro-stack-ptr 0)
    (as-1 -1 cc-micro-stack 0)
    (as-1 -1 cc-micro-stack 1)
    (cc-restore-micro-stack)
    (cc-write-pc adr)
    (cc-noop-clock)      ;dispatch inst to IR
    (cc-clock)           ;execute it
    (cc-noop-clock)      ;write spc
    (cc-save-micro-stack)
    (cond ((not (= cc-saved-micro-stack-ptr 1))
           (format t "~%Dispatch push failed to advance USP ~O" cc-saved-micro-stack-ptr)))
    (cond ((not (= (setq tem (ar-1 cc-micro-stack 1)) (1+ adr)))
           (format t "~%Dispatch next address at adr ~O pushed ~O instead" adr tem))))
)
```

```
(DEFVAR CC-DIAG-TRACE NIL "Value of T prints all errors as they occur.")

(DECLARE (FIXNUM I J K M N NBITS BITNO REGADR PPSS SHIFT RELAD)
         (SPECIAL CC-SUSPECT-BIT-LIST CC-DIAG-TRACE CC-TEST-ADR-BARFED))


(DEFtest CC-TEST-M-MEM-ADR () "M-MEM Address"
  (CC-TEST-ADR "M-MEM" RAMMO 32. 32. 1))   ;COMMENT, REGADR, WIDTH, # REGISTERS, INITIAL
                         ;RELATIVE TEST ADR; M 0 DOESNT WIN SINCE IT GETS CLOBBERED BY
                                      ;CC-R-D WHEN WRITING THE MD.

(DEFtest CC-TEST-A-MEM-ADR () "A-MEM Address"
  (CC-TEST-ADR "A-MEM" RAAMO 32. 1024. 1)) ;LIKEWISE, A 0 LOSES.

(DEFtest CC-TEST-PDL-ADR () "PDL Address"
  (CC-TEST-ADR "PDL-BUFFER" RAPBO 32. 1024. 0))
```

```
;Data test, using progressive shifts of the address and complement of address as data
(DEFUN CC-TEST-ADR (MESSAGE REGADR NBITS NREG IRELAD)
 (COND ((<= NBITS 36.)  ;FOR SPEED, FIXNUM CASE IS SEPARATE
  (DO ((PHASE NIL (NOT PHASE))
       (I 0 (IF PHASE (1+ I) I))
       (ONES (1- (ASH 1 NBITS)))
       (SHIFT)
       (ACTUAL)
       (CC-TEST-ADR-BARFED NIL)
       (ERRORS 0 0)
       (ADDRESS-LENGTH (HAULONG NREG)))
      ((= I NBITS))
    (DECLARE (FIXNUM I SHIFT ONES ACTUAL))      ;This won't win for c-mem,
                                                ; but its sooo slow otherwise
    (SETQ SHIFT (IF PHASE (- NBITS ADDRESS-LENGTH I) I))
    (FORMAT T "~&Data is address shifted ~D places" SHIFT)
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (CC-R-D (+ REGADR RELAD) (LOGAND ONES (ASH RELAD SHIFT))))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (COND ((NOT (= (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                       (LOGAND ONES (ASH RELAD SHIFT))))
               (SETQ ERRORS (1+ ERRORS))
               (CC-TEST-ADR-BARF MESSAGE RELAD (LOGAND ONES (ASH RELAD SHIFT)) ACTUAL))))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (CC-R-D (+ REGADR RELAD) (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (COND ((NOT (= (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                       (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))))
               (SETQ ERRORS (1+ ERRORS))
               (CC-TEST-ADR-BARF MESSAGE RELAD (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))
                                 ACTUAL))))
;IF THERE WERE ERRORS, GO THRU THE OPPOSITE DIRECTION TO ATTEMPT TO DETERMINE
; THE HIGH ADR THAT LOST.
    (COND ((NOT (ZEROP ERRORS))
           (FORMAT T "~&Scanning down, same parameters~%")
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
               (CC-R-D (+ REGADR RELAD) (LOGAND ONES (ASH RELAD SHIFT))))
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
               (COND ((NOT (= (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                              (LOGAND ONES (ASH RELAD SHIFT))))
                      (SETQ ERRORS (1+ ERRORS))
                      (CC-TEST-ADR-BARF MESSAGE RELAD
                                        (LOGAND ONES (ASH RELAD SHIFT)) ACTUAL))))
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
               (CC-R-D (+ REGADR RELAD) (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))))
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD NREG)
               (COND ((NOT (= (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                              (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))))
                      (SETQ ERRORS (1+ ERRORS))
                      (CC-TEST-ADR-BARF MESSAGE RELAD
                                        (LOGAND ONES (ONES-COMPLEMENT (ASH RELAD SHIFT)))
                                        ACTUAL))))
           (TERPRI)))
))
 (T  ;NON-FIXNUM CASE
  (DO ((SHIFT 0 (1+ SHIFT))
       (SHIFTMPY 1 (PLUS SHIFTMPY SHIFTMPY))
       (ONES (DIFFERENCE (DPB 1 (+ (LSH NBITS 6) 0001) 0) 1))
       (ACTUAL)
       (CC-TEST-ADR-BARFED NIL)
       (ERRORS 0 0))
      ((= SHIFT NBITS))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (CC-R-D (+ REGADR RELAD) (TIMES RELAD SHIFTMPY)))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (COND ((NOT (EQUAL (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                           (TIMES RELAD SHIFTMPY)))
               (SETQ ERRORS (1+ ERRORS))
               (CC-TEST-ADR-BARF MESSAGE RELAD (TIMES RELAD SHIFTMPY) ACTUAL))))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (CC-R-D (+ REGADR RELAD) (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))))
    (DO RELAD IRELAD (1+ RELAD) (= RELAD NREG)
        (COND ((NOT (EQUAL (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                           (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))))
               (SETQ ERRORS (1+ ERRORS))
               (CC-TEST-ADR-BARF MESSAGE RELAD (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))
                                 ACTUAL))))
;IF THERE WERE ERRORS, GO THRU THE OPPOSITE DIRECTION TO ATTEMPT TO DETERMINE
; THE HIGH ADR THAT LOST.
    (COND ((NOT (ZEROP ERRORS))
           (PRINC "SCANNING DOWN, SAME PARAMETERS")
           (TERPRI)
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
               (CC-R-D (+ REGADR RELAD) (TIMES RELAD SHIFTMPY)))
           (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
               (COND ((NOT (EQUAL (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                                  (TIMES RELAD SHIFTMPY)))
```

```
                    (SETQ ERRORS (1+ ERRORS))
                    (CC-TEST-ADR-BARF MESSAGE RELAD (TIMES RELAD SHIFTMPY) ACTUAL))))
          (DO RELAD (1- NREG) (1- RELAD) (< RELAD IRELAD)
              (CC-R-D (+ REGADR RELAD) (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))))
          (DO RELAD (1- NREG) (1- RELAD) (< RELAD NREG)
              (COND ((NOT (EQUAL (SETQ ACTUAL (CC-R-E (+ REGADR RELAD)))
                                 (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))))
                    (SETQ ERRORS (1+ ERRORS))
                    (CC-TEST-ADR-BARF MESSAGE RELAD
                                       (DIFFERENCE ONES (TIMES RELAD SHIFTMPY))
                                       ACTUAL))))
          (TERPRI)))
))))

(DEFUN CC-TEST-ADR-BARF (MESSAGE RELAD GOOD BAD)
  (AND (NOT CC-TEST-ADR-BARFED)
       (SETQ CC-TEST-ADR-BARFED T)
       (FORMAT T "~&Error while address-testing: ~A~%" MESSAGE))
  (FORMAT T "Relative address: ~O; wrote ~O; read ~O~%" GOOD RELAD BAD))

(DEFUN CC-ASSURE-C-MEM-ZERO (&OPTIONAL (START 0)(END 20000))
  (DO ((ADR START (1+ ADR))
       (C-MEM-CONTENTS)
       (C-MEM-HIGH) (C-MEM-MEDIUM) (C-MEM-LOW)
       (HIGH-BAD-AND 177777)
       (MEDIUM-BAD-AND 177777)
       (LOW-BAD-AND 177777)
       (HIGH-BAD-OR 0)
       (MEDIUM-BAD-OR 0)
       (LOW-BAD-OR 0)
       (BAD-ADDRESS-AND 177777)
       (BAD-ADDRESS-OR 0))
      ((>= ADR END) (FORMAT T "~%AND of non-zero locations: ~O~%OR of non-zero locations: ~O
AND of bad addresses: ~O~%OR of bad address: ~O"
                            (+ (ASH HIGH-BAD-AND 40) (ASH MEDIUM-BAD-AND 20) LOW-BAD-AND)
                            (+ (ASH HIGH-BAD-OR 40) (ASH MEDIUM-BAD-OR 20) LOW-BAD-OR)
                            BAD-ADDRESS-AND
                            BAD-ADDRESS-OR))
      (COND ((NOT (ZEROP (SETQ C-MEM-CONTENTS (CC-READ-C-MEM ADR))))
             (SETQ BAD-ADDRESS-AND (LOGAND BAD-ADDRESS-AND ADR)
                   BAD-ADDRESS-OR (LOGIOR BAD-ADDRESS-OR ADR)
                   C-MEM-HIGH (LDB 4020 C-MEM-CONTENTS)
                   C-MEM-MEDIUM (LDB 2020 C-MEM-CONTENTS)
                   C-MEM-LOW (LDB 0020 C-MEM-CONTENTS))
             (SETQ HIGH-BAD-AND (LOGAND HIGH-BAD-AND C-MEM-HIGH)
                   MEDIUM-BAD-AND (LOGAND MEDIUM-BAD-AND C-MEM-MEDIUM)
                   LOW-BAD-AND (LOGAND LOW-BAD-AND C-MEM-LOW)

                   HIGH-BAD-OR (LOGIOR HIGH-BAD-OR C-MEM-HIGH)
                   MEDIUM-BAD-OR (LOGIOR MEDIUM-BAD-OR C-MEM-MEDIUM)
                   LOW-BAD-OR (LOGIOR LOW-BAD-OR C-MEM-LOW))))))


;CC-ZERO-C-MEM defined in LMCONS;ZERO
(DEFUN CC-ZERO-C-MEM-CONTINUOUS ()
  (CC-EXECUTE (W-C-MEM 0)
        CONS-IR-OP CONS-OP-BYTE
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-A-SRC 1
        CONS-IR-BYTE-FUNC CONS-BYTE-FUNC-DPB
        CONS-IR-MROT 12.
        CONS-IR-BYTL-1 13.
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-OA-LOW)
  (CC-EXECUTE (W-C-MEM 1)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-A-SRC 1 ;VALUE TO WRITE (HIGH)
        CONS-IR-M-SRC 1 ;VALUE TO WRITE (LOW)
        CONS-IR-JUMP-ADDR 0
        CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
        CONS-IR-R 1
        CONS-IR-P 1
        CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 2)
        CONS-IR-STAT-BIT 1
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-M+1
        CONS-IR-FUNC-DEST CONS-FUNC-DEST-MD)
  (CC-EXECUTE (W-C-MEM 3)
        CONS-IR-OP CONS-OP-JUMP
        CONS-IR-JUMP-ADDR 0
        CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
        CONS-IR-N 1)
  (UNTIL-KEY
      (CC-WRITE-STAT-COUNTER -16380.) ;STOP AFTER WRITING 16K-4 LOCATIONS
      (CC-WRITE-M-MEM 1 0)
      (CC-WRITE-MD 4) ;STARTING AT 4
      (CC-RUN-TEST-LOOP 0))
```

```
)

(DEFUN CC-TEST-C-MEM-PARITY-CHECKER NIL
  (DO ((BIT 0 (1+ BIT))
       (QUAN))
      ((= BIT 47.))
    (CC-WRITE-C-MEM 0 (SETQ QUAN (ASH 1 BIT)))
    (CC-EXECUTE (WRITE)
                CONS-IR-OP CONS-OP-JUMP ;DO JUMP INSTRUCTION TO DESIRED PLACE
                CONS-IR-JUMP-ADDR 0
                CONS-IR-JUMP-COND CONS-JUMP-COND-UNC)
       (COND ((NOT (= QUAN (CC-READ-IR)))
              (FORMAT T "~%~WROTE ~O READ ~O" QUAN (CC-READ-IR))))
       (CC-NOOP-CLOCK)
       (COND ((NOT (ZEROP (LOGLDB 501 (SPY-READ SPY-FLAG-1))))
              (FORMAT T "~%parity checker failed BIT ~D." BIT)))))

(DEFUN CC-MEM-TEST-LOOP (ADR &OPTIONAL WRITE-DATA READ-ALSO)
  (COND (WRITE-DATA
         (DO ((WORD)) ((KBD-TYI-NO-HANG) (PHYS-MEM-READ ADR))
             (AND WORD (RETURN-ARRAY WORD))
             (PHYS-MEM-WRITE ADR WRITE-DATA)
             (AND READ-ALSO (SETQ WORD (PHYS-MEM-READ ADR)))))
        (T
         (DO ((WORD)) ((KBD-TYI-NO-HANG) WORD)
             (AND WORD (RETURN-ARRAY WORD))
             (SETQ WORD (PHYS-MEM-READ ADR))))))

(DEFUN CC-MEM-ZERO (FROM TO)
  (DO ((ADR FROM (1+ ADR)))
      ((OR (KBD-TYI-NO-HANG) (> ADR TO)) ADR)
      (PHYS-MEM-WRITE ADR 0)))

;;; Perform a read or write, check specified status bits.
(DEFUN DC-CLP-NXM (&AUX STATUS)
  (DO () ((KBD-TYI-NO-HANG) STATUS)
      (PHYS-MEM-WRITE DC-CLP-ADR 400000)
      (PHYS-MEM-WRITE DC-CMD-ADR 0)
      (PHYS-MEM-WRITE DC-START-ADR 0)
      (DO () ((LDB-TEST 0001 (SETQ STATUS (PHYS-MEM-READ DC-STS-ADR)))))))


(DEFUN CC-MEM-FILL (FROM TO &OPTIONAL (WORD 0) (FUNCTION (FUNCTION 1+)))
  (DO ((ADR FROM (1+ ADR))
       (WORD WORD (FUNCALL FUNCTION WORD)))
      ((OR (KBD-TYI-NO-HANG) (> ADR TO)) ADR)
      (PHYS-MEM-WRITE ADR WORD)))

(DEFUN CC-MEM-FILL-CHECK (FROM TO &OPTIONAL (WORD 0) (FUNCTION (FUNCTION 1+)))
  (DO ((ADR FROM (1+ ADR))
       (MEM-WORD 0)
       (WORD WORD (FUNCALL FUNCTION WORD)))
      ((OR (KBD-TYI-NO-HANG) (> ADR TO)) ADR)
      (OR (= (SETQ MEM-WORD (PHYS-MEM-READ ADR)) WORD)
          (FORMAT T "Compare error: Adr=~O, is ~O but should be ~O~%" ADR MEM-WORD WORD))))

(DEFUN CC-MEM-TEST-ONE-WORD-TO-DISK (ADR &OPTIONAL (WORD 0)
                                         PRINT-FLAG (FUNCTION (FUNCTION 1+)))
  (DO ((CORE-PAGE (// ADR 400))
       (WORD WORD (FUNCALL FUNCTION WORD)))
      ((KBD-TYI-NO-HANG) WORD)
      (AND PRINT-FLAG (PRINC WORD) (PRINC " "))
      (PHYS-MEM-WRITE ADR WORD)
      (CC-DISK-WRITE 1 CORE-PAGE 1)))

(DEFUN CC-MEM-READ-DISK (ADR)
  (CC-DISK-READ 1 (// ADR 400) 1))

(DEFUN CC-DISK-REPEAT-OP (CORE-PAGE &OPTIONAL SLEEP-TIME
                                    ERROR-PRINT-FLAG (FCN CC-DISK-WRITE-FCN))
  (PHYS-MEM-WRITE 12 (LSH CORE-PAGE 8))
  (DO ((STATUS))
      ((KBD-TYI-NO-HANG))
      (AND SLEEP-TIME (PROCESS-SLEEP SLEEP-TIME) "CC Disk Operation")
      (PHYS-MEM-WRITE (+ CC-DISK-ADDRESS 0) FCN) ;Store command, does reset
      (PHYS-MEM-WRITE (+ CC-DISK-ADDRESS 1) 12)  ;Store CLP
      (SETQ CC-DISK-LAST-CMD FCN CC-DISK-LAST-CLP 12)
      (PHYS-MEM-WRITE (+ CC-DISK-ADDRESS 2) 1)  ;Disk adr: always track 0, head 0, sector 1
      (PHYS-MEM-WRITE (+ CC-DISK-ADDRESS 3) 0)   ;Start transfer
      (DO () ((NOT (ZEROP (LDB 0001 (SETQ STATUS (PHYS-MEM-READ CC-DISK-ADDRESS)))))))
      (COND ((AND ERROR-PRINT-FLAG
                  (NOT (ZEROP (LOGAND STATUS 47777560))))
             ; ERROR BITS: INTERNAL PARITY, NXM, MEM PAR, HEADER COMPARE,
             ; HEADER ECC, ECC HARD, ECC SOFT, READ OVERRUN, WRITE OVERRUN,
             ; START-BLOCK ERR, TIMEOUT, SEEK ERR, OFF LINE, OFF CYL, FAULT,
             ;    NO SEL, MUL SEL
             (CC-DISK-ANALYZE)))))
```

```
;; MAP FIRST 256K VIRTUAL MEMORY TO PHYSICAL MEMORY
(DEFUN CC-LOAD-STRAIGHT-MAP (&OPTIONAL (PAGE-OFFSET 0))
  (DO ((L-2 0 (1+ L-2)))
      ((≥ L-2 1024.))
      (CC-WRITE-LEVEL-2-MAP L-2 (+ 60000000 L-2 PAGE-OFFSET)))
  (DO ((L-1 0 (1+ L-1)))
      ((≥ L-1 40))
      (CC-WRITE-LEVEL-1-MAP L-1 L-1)))

(DEFMACRO CC-MEMORY-BANK (VMA)
  `(LDB 1612 ,VMA))

(DEFUN CC-PARITY-SWEEP-INFO (PHYS-ADR-LIST
                             &OPTIONAL FIX-SINGLE-BIT-ERRORS (PRINT-AREA-SYMBOL T))
  (DO ((L PHYS-ADR-LIST (CDR L))
       (PHYS-ADR) (VIRT-ADR) (AREA-NUMBER) (AREA-SYMBOL) (CORE) (DISK))
      ((NULL L) NIL)
    (SETQ PHYS-ADR (CAR L) VIRT-ADR (QF-VIRT-ADR-OF-PHYS-ADR PHYS-ADR))
    (COND ((NULL VIRT-ADR)
           (FORMAT T "~%~O: not found in PHT" PHYS-ADR))
          ((= (LDB 1020 VIRT-ADR) %PHT-DUMMY-VIRTUAL-ADDRESS)
           (FORMAT T "~%~O: which is a free page of core" PHYS-ADR))
          (T
           (SETQ AREA-NUMBER (QF-AREA-NUMBER-OF-POINTER VIRT-ADR)
                 AREA-SYMBOL
                 (COND (PRINT-AREA-SYMBOL
                        (READLIST (CC-Q-EXPLODE (QF-MEM-READ (+ (QF-INITIAL-AREA-ORIGIN
                                                                  'AREA-NAME)
                                                                AREA-NUMBER))))))
           (FORMAT T "~%~O: Virtual adr ~O, Area ~S " PHYS-ADR VIRT-ADR AREA-SYMBOL)
           (FORMAT T " Core copy ~O, Disk copy ~O  bits:"
                   (SETQ CORE (QF-MEM-READ VIRT-ADR))
                   (SETQ DISK (QF-MEM-READ-DISK-COPY VIRT-ADR)))
           (CC-PRINT-BITS (LOGXOR CORE DISK))
           (IF (AND FIX-SINGLE-BIT-ERRORS
                    T  ;(SINGLE-BIT-P (LOGXOR CORE DISK))
                    )
               (PROGN (FORMAT T "~%Fixing locn ~o to ~o" phys-adr disk)
                      (PHYS-MEM-WRITE PHYS-ADR DISK)))))))

(DEFUN CC-PARITY-SWEEP (&OPTIONAL (NUMBER-OF-MEMORIES 2)
                                  VERBOSE-P FIX-ERRORS-P
                                  (FIRST-ADDRESS 0)
                        &AUX (C-MEM-SAVE-LIST '(17000 17001 17002 17003)))
  (LET ((SAVED-CONTROL-MEMORY (MAPCAR #'CC-READ-C-MEM C-MEM-SAVE-LIST))
        (CURRENT-DATA-LOGAND) (CURRENT-DATA-LOGIOR)
        (CURRENT-ADR-LOGAND) (CURRENT-ADR-LOGIOR)
        (CURRENT-BANK (CC-MEMORY-BANK FIRST-ADDRESS))
        (ERROR-FLAG NIL) (MEM-SIZE (LSH NUMBER-OF-MEMORIES 16.))
        (BAD-LOCS))
    (DBG-RESET)                          ;TEMPORARY KLUDGE?  JUST IN CASE MACHINE IS HUNG
    (CC-RESET-MACH)
    (SPY-WRITE SPY-MODE 44)              ;Prom disable, errhalt
    (CC-FAST-LOAD-STRAIGHT-MAP)
    (CC-EXECUTE (W-C-MEM 17000)
                CONS-IR-OP CONS-OP-ALU
                CONS-IR-M-SRC CONS-M-SRC-VMA
                CONS-IR-OB CONS-OB-ALU
                CONS-IR-ALUF CONS-ALU-M+1
                CONS-IR-FUNC-DEST CONS-FUNC-DEST-VMA-START-READ)
    (CC-EXECUTE (W-C-MEM 17001)
                CONS-IR-STAT-BIT 1)      ;DELAY (NO PAGE FAULT EXPECTED)
    (CC-EXECUTE (W-C-MEM 17002)
                CONS-IR-M-SRC CONS-M-SRC-MD
                CONS-IR-OB CONS-OB-ALU
                CONS-IR-ALUF CONS-ALU-SETM)      ;DEST M-GARBAGE
    (CC-EXECUTE (W-C-MEM 17003)
                CONS-IR-OP CONS-OP-JUMP
                CONS-IR-JUMP-ADDR 17000
                CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
                CONS-IR-N 1)
    (SETQ MEM-SIZE (- MEM-SIZE FIRST-ADDRESS))
    (CC-WRITE-STAT-COUNTER (MINUS MEM-SIZE))
    (CC-WRITE-FUNC-DEST CONS-FUNC-DEST-VMA (1- FIRST-ADDRESS))
    (SETQ CURRENT-DATA-LOGAND -1  CURRENT-DATA-LOGIOR 0
          CURRENT-ADR-LOGAND -1  CURRENT-ADR-LOGIOR 0)
    (DO () (NIL)
      (CC-RUN-TEST-LOOP-W-ERROR-HALTS 17000)
      (LET ((VMA (CC-READ-M-MEM CONS-M-SRC-VMA))
            (MD (CC-READ-M-MEM CONS-M-SRC-MD)))
        (COND ((≠ (CC-MEMORY-BANK VMA) CURRENT-BANK)
               (COND (ERROR-FLAG
                      (CC-PRINT-BANK-AS-BOARD-AND-BANK CURRENT-BANK)
                      (FORMAT T "~&Address LOGAND=~O, Address LOGIOR=~O, Data LOGAND=~O, Data LOGIOR=~O~%"
                              (LOGAND CURRENT-ADR-LOGAND (1- (ASH 1 24.)))
                              CURRENT-ADR-LOGIOR
                              (LOGAND CURRENT-DATA-LOGAND (1- (ASH 1 32.)))
                              CURRENT-DATA-LOGIOR)))
```

```
              (SETQ CURRENT-DATA-LOGAND -1  CURRENT-DATA-LOGIOR 0
                    CURRENT-ADR-LOGAND -1  CURRENT-ADR-LOGIOR 0
                    CURRENT-BANK (CC-MEMORY-BANK VMA)
                    ERROR-FLAG NIL)))
       (COND ((ZEROP (CC-READ-STAT))
              (RETURN T))
             (T (AND VERBOSE-P
                     (FORMAT T "~%VMA: ~O MD: ~O" VMA MD))
                (SETQ ERROR-FLAG T
                      CURRENT-ADR-LOGAND (LOGAND CURRENT-ADR-LOGAND VMA)
                      CURRENT-ADR-LOGIOR (LOGIOR CURRENT-ADR-LOGIOR VMA)
                      CURRENT-DATA-LOGAND (LOGAND CURRENT-DATA-LOGAND MD)
                      CURRENT-DATA-LOGIOR (LOGIOR CURRENT-DATA-LOGAND MD)
                      BAD-LOCS (CONS VMA BAD-LOCS))
                (AND FIX-ERRORS-P
                     (PHYS-MEM-WRITE VMA (PHYS-MEM-READ VMA)))))))
    (DOLIST (LOC C-MEM-SAVE-LIST)
      (CC-WRITE-C-MEM LOC (CAR SAVED-CONTROL-MEMORY))
      (SETQ SAVED-CONTROL-MEMORY (CDR SAVED-CONTROL-MEMORY)))
    BAD-LOCS))

(DEFUN CC-PRINT-BANK-AS-BOARD-AND-BANK (BANK)
  (LET ((BOARD (FIX (// BANK 4))))
    (FORMAT T "~&Bank ~O, which is Bank ~O of Board ~O  (based from zero)~%"
          BANK (- BANK (* BOARD 4)) BOARD)
    T))


(DEFUN CC-RUN-TEST-LOOP-W-ERROR-HALTS (ADR)
  (CC-WRITE-PC ADR)
  (CC-NOOP-CLOCK)                ;FIRST INSTRUCTION TO IR
  (CC-CLOCK)                     ;CLOCK AGAIN
  (SPY-WRITE SPY-MODE 54)  ;ENABLE STAT HALT, PROM DISABLE, ERR HALT
  (SPY-WRITE SPY-CLK 1) ;TAKE OFF
  (DO () ((BIT-TEST 6000 (LOGXOR 4000 (SPY-READ SPY-FLAG-1))))
    (PROCESS-SLEEP 15. "Await Stat Halt"))          ;AWAIT STAT HALT
  )
```

```
;;; Function for testing and adjusting the clock

(DEFVAR cc-adjust-clock-array)

(deftest cc-test-clock () "Clock"
  (or (variable-boundp cc-adjust-clock-array)
      (setq cc-adjust-clock-array (*array nil 'fixnum 8.))) ; Should be changed
  ;These first two are to get everything paged in
  (cc-measure-clock 0)
  (cc-measure-clock 4)
  (doTIMES (I 8.)
    (ASET (cc-measure-clock i) CC-ADJUST-CLOCK-array i))
  (SEND TERMINAL-IO ':LINE-OUT "
Speed   ILong     Pin   Actual  Nominal")
  (do ((i 0 (1+ i))
       (pins '(5D08-6 5D08-3 5D08-16 5D08-14 5D08-5 5D08-4 5D08-17 5D08-15) (cdr pins))
       (nominals '(235. 180. 180. 160. 235. 220. 210. 200.) (cdr nominals)))
      ((= i 8))
    (format t "  ~D     ~:[No~;Yes~]    ~A         ~D      ~D~%"
            (logand 3 i) (> i 3) (car pins) (AREF cc-adjust-clock-array i)
            (car nominals)))
  (FORMAT T "~%Also, scope clock at 5A10-11; width of low phase should be about 75 ns
~%~4TNote that the standard cycle-time (5D08-16) has been increased to
~4T180ns from 170ns for greater reliability.~%"))

;Returns period in nanoseconds
(DEFUN CC-MEASURE-CLOCK (SPEED-ILONG &AUX START-TIME END-TIME)
  (CC-WRITE-MD 0)  ;Will count cycles
  (cond ((< speed-ilong 4)
         (cc-execute cons-ir-m-src cons-m-src-md
                     cons-ir-ob cons-ob-alu
                     cons-ir-aluf cons-alu-M+1
                     cons-ir-func-dest cons-func-dest-md))
        (t
         (cc-execute cons-ir-ilong 1
                     cons-ir-m-src cons-m-src-md
                     cons-ir-ob cons-ob-alu
                     cons-ir-aluf cons-alu-M+1
                     cons-ir-func-dest cons-func-dest-md)))
  (spy-write spy-mode (logand 3 speed-ilong)) ;Set speed, clear errstop, etc.
  (spy-write spy-clk 11)        ;Set RUN and DEBUG
  (let ((low (%unibus-read 764120))  ;Hardware synchronizes if you read this one first
        (high (%unibus-read 764122)))
    (setq start-time (dpb high 2007 low)))
  (process-sleep 60. "Measure Clock")
  (spy-write spy-clk 10)        ;Clear RUN, but leave DEBUG set
  (spy-write spy-mode 0)        ;Dont leave that random speed in there.  The cc-read-m-mem
                                ; may cause randomness if you do.
  (let ((low (%unibus-read 764120))  ;Hardware synchronizes if you read this one first
        (high (%unibus-read 764122)))
    (setq end-time (dpb high 2007 low)))
  (// (* (cond ((> end-time start-time) (- end-time start-time))
               (t (+ (- end-time start-time) 1_23.)))
         1000.)
     (cc-read-m-mem cons-m-src-md)))
```

```
;;; Testing of instruction-modification paths.  The general methodology is
;;; to execute an instruction which has an OA destination,
;;; then read back the IR.  With one side of the IOB or-gates held low we
;;; test the bits on the other side.  First we put the OA-modifying instruction into
;;; the IR, then we put the desired value for the I lines into the DEBUG-IR
;;; then do a DEBUG-CLOCK.
(DEFtest CC-TEST-OA-REGS () "OA Registers"
  (CC-TEST-OA-REG "OA-REG-LOW" CONS-FUNC-DEST-OA-LOW 0 26. 1 0)
  (CC-TEST-OA-REG "OA-REG-LOW" CONS-FUNC-DEST-OA-LOW 0 26. 0 1)
  (CC-TEST-OA-REG "OA-REG-HIGH" CONS-FUNC-DEST-OA-HIGH 26. 22. 1 0)
  (CC-TEST-OA-REG "OA-REG-HIGH" CONS-FUNC-DEST-OA-HIGH 26. 22. 0 1))

;;; Float a 1 bit through and complain about wrong 1's or 0's
;;; Conceivably could float 0's also.
(DEFUN CC-TEST-OA-REG (MESSAGE DEST FIRST-IR-BIT N-BITS IR-BIT M-BIT)
  (DO ((N N-BITS (1- N))
       (IR-BIT (ASH IR-BIT FIRST-IR-BIT) (ASH IR-BIT 1))
       (M-BIT M-BIT (ASH M-BIT 1))
       (BITNO 0 (1+ BITNO))
       (GOOD) (BAD)
       (CC-SUSPECT-BIT-LIST NIL))
      ((ZEROP N))
    (CC-WRITE-MD M-BIT)
    (CC-EXECUTE
        CONS-IR-M-SRC CONS-M-SRC-MD
        CONS-IR-OB CONS-OB-ALU
        CONS-IR-ALUF CONS-ALU-SETM
        CONS-IR-FUNC-DEST DEST)
    (CC-WRITE-DIAG-IR IR-BIT)
    (CC-DEBUG-CLOCK)
    ;; IR should now have OR of M-BIT and IR-BIT
    (SETQ GOOD (LOGIOR IR-BIT (ASH M-BIT FIRST-IR-BIT))
          BAD (CC-READ-IR))
    (COND ((NOT (= GOOD BAD))
           (FORMAT T "~&~4T~A failure: IR-BIT is ~D, M-BIT is ~D " MESSAGE IR-BIT M-BIT)
           (COND ((ZEROP IR-BIT)
                  (FORMAT T "OB has 1 in bit ~D" BITNO)
                  (IF (NOT (ZEROP FIRST-IR-BIT))
                      (FORMAT T " (= ~D)" (+ BITNO FIRST-IR-BIT)))
                  (SEND TERMINAL-IO ':STRING-OUT ", I"))
                 (T (FORMAT T "I has 1 in bit ~D, OB" (+ BITNO FIRST-IR-BIT))))
           (IF (ZEROP BAD)
               (SEND TERMINAL-IO ':LINE-OUT " has zero.  IR got zero")
             (CC-PRINT-BIT-LIST " has zero.  1-bits in IR: "
                                (CC-WRONG-BITS-LIST 0 BAD 48.)))))))

(DEFVAR CC-RANDOM-DATA-ARRAY NIL)
(DEFVAR CC-RANDOM-DATA-ARRAY-COMPLEMENTED NIL)

;This one takes a while.  Run it when you are out to lunch.
(DEFtest CC-C-MEM-BLOCK-ADDRESS-TEST (&OPTIONAL (ISA 0)) "C-MEM Block Address"
   (COND ((NULL CC-RANDOM-DATA-ARRAY)
          (SETQ CC-RANDOM-DATA-ARRAY (MAKE-ARRAY NIL ART-Q 400))
          (SETQ CC-RANDOM-DATA-ARRAY-COMPLEMENTED (MAKE-ARRAY NIL ART-Q 400))
          (DO I 0 (1+ I) (= I 400)
              (AS-1 (LOGXOR (AS-1 (DPB (RANDOM 200000)
                                        4020
                                        (DPB (RANDOM 200000)
                                             2020
                                             (RANDOM 200000)))
                                   CC-RANDOM-DATA-ARRAY
                                   I)
                             77777777777777777)
                    CC-RANDOM-DATA-ARRAY-COMPLEMENTED
                    I))))
   (*CATCH 'BLOCK-TEST
        (DO SA ISA (+ SA 400) (= SA 40000)
            (CC-CMB-TEST SA))))

(DEFUN CC-CMB-TEST (SA)
   (CC-CMB-WRITE-BLOCK SA CC-RANDOM-DATA-ARRAY)
   (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY)))
          (FORMAT T "~%400 wd block at ~O doesnt retain data" SA))
         (T (CC-CMB-ZAP SA 0 SA 0)
            (CC-CMB-ZAP SA (+ SA 400) 40000 0)
            (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY)))
                   (FORMAT T "~%400 wd block at ~O changed by writing 0's elsewhere" SA)))
            (CC-CMB-ZAP SA 0 SA -1)
            (CC-CMB-ZAP SA (+ SA 400) 40000 0)
            (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY)))
                   (FORMAT T "~%400 wd block at ~O changed by writing 1's elsewhere" SA)))
            (CC-CMB-WRITE-BLOCK SA CC-RANDOM-DATA-ARRAY-COMPLEMENTED)
            (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY-COMPLEMENTED)))
                   (FORMAT T "~%400 wd block at ~O doesn't retain (complemented) data" SA)))
            (CC-CMB-ZAP SA 0 SA 0)
            (CC-CMB-ZAP SA (+ SA 400) 40000 0)
            (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY-COMPLEMENTED)))
                 (FORMAT T "~%400 wd block at ~O changed by writing 0's elsewhere (COM)"
                         SA)))
          (CC-CMB-ZAP SA 0 SA -1)
          (CC-CMB-ZAP SA (+ SA 400) 40000 0)
          (COND ((NOT (ZEROP (CC-CMB-TEST-BLOCK SA CC-RANDOM-DATA-ARRAY-COMPLEMENTED)))
                 (FORMAT T "~%400 wd block at ~O changed by writing 1's elsewhere(COM)"
                         SA))))))

(DEFUN CC-CMB-ZAP (SA FROM TO DATA)
  (COND ((KBD-TYI-NO-HANG)
         (FORMAT T "~%WAS TESTING BLOCK AT ~S" SA)
         (*THROW 'BLOCK-TEST NIL)))
  (CC-WRITE-A-MEM 1 (LOGLDB 4020 DATA)) ;1@A GETS HIGH 16 BITS
  (CC-WRITE-M-MEM 0 (DPB (LDB 2020 DATA) 2020 (LDB 0020 DATA))) ;0@M GETS LOW 32 BITS
  (DO I FROM (1+ I) (NOT (< I TO))
     (CC-EXECUTE (WRITE)
            CONS-IR-OP CONS-OP-JUMP   ;EXECUTE MAGIC FLAVOR OF JUMP INSTRUCTION
            CONS-IR-JUMP-ADDR I
            CONS-IR-P 1               ;R+P=WRITE C MEM
            CONS-IR-R 1
            CONS-IR-A-SRC 1
            ;CONS-IR-M-SRC 0
            CONS-IR-JUMP-COND CONS-JUMP-COND-UNC)))


(DEFUN CC-CMB-WRITE-BLOCK (SA ARY)
   (DO I 0 (1+ I) (= I 400)
      (CC-WRITE-C-MEM (+ SA I) (AR-1 ARY I))))

(DEFUN CC-CMB-TEST-BLOCK (SA ARY &AUX (ERRS 0) RES)
   (DO ((I 0 (1+ I)))
      ((OR (= I 400)
           (AND (NULL CC-DIAG-TRACE)
                (NOT (ZEROP ERRS))))
       ERRS)
     (COND ((NOT (= (SETQ RES (CC-READ-C-MEM (+ SA I))) (AR-1 ARY I)))
            (SETQ ERRS (1+ ERRS))
            (COND (CC-DIAG-TRACE
                   (FORMAT T "~%ADR:~O READ ~O, SHOULD BE ~O" (+ I SA) RES (AR-1 ARY I))))))))


;ALU TESTS

(DEFtest CC-TEST-INCREMENTER () "Incrementer"
 (DO ((BIT 0 (1+ BIT))
      (DAT)
      (RES))
     ((= BIT 32.))
   (CC-WRITE-M-MEM 1 (1- (SETQ DAT (ASH 1 BIT))))
   (CC-EXECUTE
     CONS-IR-OP CONS-OP-ALU
     CONS-IR-M-SRC 1
     CONS-IR-OB CONS-OB-ALU
     CONS-IR-ALUF CONS-ALU-M+1)
   (COND ((NOT (= (SETQ RES (CC-READ-OBUS)) DAT))
          (FORMAT T "~%Incrementing bit ~D, got ~o instead of ~o" BIT RES DAT)))))

(DEFtest CC-TEST-ARITH-COND-JUMP () "Arithmetic Conditional Jump"
  (DO ((BIT 0 (1+ BIT))
       (DAT))
      ((= BIT 31.))
    (SETQ DAT (ASH 1 BIT))
    (CC-WRITE-M-MEM 1 DAT)
    (CC-WRITE-M-MEM 2 (1- DAT))
    (CC-WRITE-M-MEM 3 (MINUS DAT))
    (CC-WRITE-M-MEM 4 (MINUS (1- DAT)))
    (DO ((I 1 (1+ I)))
        ((= I 4))
      (CC-TEST-JUMP-INTERNAL I I CONS-JUMP-COND-M=A "M=A" T)
      (CC-TEST-JUMP-INTERNAL I I CONS-JUMP-COND-M<A "M<A" NIL)
      (CC-TEST-JUMP-INTERNAL I I CONS-JUMP-COND-M>A "M>A" NIL)
      (CC-TEST-JUMP-INTERNAL I I CONS-JUMP-COND-M<=A "M<=A" T)
      (CC-TEST-JUMP-INTERNAL I I CONS-JUMP-COND-M>=A "M>=A" T))
    (CC-TEST-JUMP-1 2 1)
    (CC-TEST-JUMP-1 3 4)))

(DEFUN CC-TEST-JUMP-1 (LESS MORE)
  (CC-TEST-JUMP-INTERNAL LESS MORE CONS-JUMP-COND-M<A "M<A" T)
  (CC-TEST-JUMP-INTERNAL MORE LESS CONS-JUMP-COND-M<A "M<A" NIL)
  (CC-TEST-JUMP-INTERNAL LESS MORE CONS-JUMP-COND-M>A "M>A" NIL)
  (CC-TEST-JUMP-INTERNAL MORE LESS CONS-JUMP-COND-M>A "M>A" T))

(DEFUN CC-TEST-JUMP-INTERNAL (M-ADR A-ADR JUMP-COND STRING SHOULD-JUMP
                              &AUX NPC JCOND WILL-JUMP ERR)
  (CC-WRITE-PC 0)
  (CC-EXECUTE
    CONS-IR-OP CONS-OP-JUMP
    CONS-IR-M-SRC M-ADR
```

```
      CONS-IR-A-SRC A-ADR
      CONS-IR-JUMP-COND JUMP-COND
      CONS-IR-JUMP-ADDR 777)
  (SETQ JCOND (LDB 0201 (SPY-READ SPY-FLAG-2)))
  (SETQ WILL-JUMP (NOT (OR (AND (NOT (ZEROP JCOND)) (ZEROP (LDB 0601 JUMP-COND)))
                           (AND (ZEROP JCOND) (NOT (ZEROP (LDB 0601 JUMP-COND)))))))
  (COND ((EQ WILL-JUMP SHOULD-JUMP)
         (FORMAT T "~%JCOND incorrect before clock")  ;note! dont believe this error too much.
         (SETQ ERR T)))
  (CC-CLOCK)
  (SETQ NPC (CC-READ-PC))
  (COND ((NOT (= NPC (COND (SHOULD-JUMP 777) (T 2))))
         (FORMAT T "~%JUMP FAILED: M=~O, A=~O, COND ~A, NPC=~O"
                 (CC-READ-M-MEM M-ADR)
                 (CC-READ-A-MEM A-ADR)
                 STRING
                 NPC))
        (ERR (FORMAT T "~%Actual jump OK: M=~O, A=~O, COND ~A, NPC=~O"
                 (CC-READ-M-MEM M-ADR)
                 (CC-READ-A-MEM A-ADR)
                 STRING
                 NPC))))

;Use this to try to find slow ALU bits with a scope.
(DEFUN CC-ALU-SPEED-TEST (&OPTIONAL (A-VALUE 0) (M-VALUE 0) (A-REG 2) (M-REG 30))
  (PROG (CH FROB-M)
        (CC-STOP-MACH)
        (CC-EXECUTE (W-C-MEM 100)
            CONS-IR-SPARE-BIT 1         ;for scope trigger
            CONS-IR-OP CONS-OP-JUMP
            CONS-IR-A-SRC A-REG
            CONS-IR-M-SRC M-REG
            CONS-IR-JUMP-COND CONS-JUMP-COND-M=A
            CONS-IR-N 1
            CONS-IR-JUMP-ADDR 200)
        (CC-EXECUTE (W-C-MEM 101) )
        (CC-EXECUTE (W-C-MEM 102)
            CONS-IR-OP CONS-OP-JUMP
            CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
            CONS-IR-N 0
            CONS-IR-JUMP-ADDR 100)
        (CC-EXECUTE (W-C-MEM 103)
   ;        CONS-IR-OP CONS-OP-ALU
   ;        CONS-IR-M-SRC 1
   ;        CONS-IR-M-MEM-DEST 1
   ;        CONS-IR-OB CONS-OB-ALU
   ;        CONS-IR-ALUF CONS-ALU-M+1
                    )

        (CC-EXECUTE (W-C-MEM 200)
            CONS-IR-OP CONS-OP-JUMP
            CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
            CONS-IR-N 0
            CONS-IR-JUMP-ADDR 100)
        (CC-EXECUTE (W-C-MEM 201)
   ;        CONS-IR-OP CONS-OP-ALU
   ;        CONS-IR-M-SRC 3
   ;        CONS-IR-M-MEM-DEST 3
   ;        CONS-IR-OB CONS-OB-ALU
   ;        CONS-IR-ALUF CONS-ALU-M+1
                    )
    L   (CC-WRITE-A-MEM A-REG A-VALUE)
        (CC-WRITE-M-MEM M-REG M-VALUE)
        (SETQ CH (CC-RUN-LOOP 100))
        (COND ((MEMQ CH '(#/a #/A))
               (SETQ FROB-M NIL))
              ((MEMQ CH '(#/m #/M))
               (SETQ FROB-M T)))
        (COND (FROB-M
               (COND ((= CH #/+) (SETQ M-VALUE (1+ M-VALUE)))
                     ((= CH #/←) (SETQ M-VALUE (ASH M-VALUE 1)))
                     ((= CH #/→) (SETQ M-VALUE (ASH M-VALUE -1)))
                     ((OR (= CH #/z) (= CH #/Z)) (SETQ M-VALUE 0))))
              (T
               (COND ((= CH #/+) (SETQ A-VALUE (1+ A-VALUE)))
                     ((= CH #/←) (SETQ A-VALUE (ASH A-VALUE 1)))
                     ((= CH #/→) (SETQ A-VALUE (ASH A-VALUE -1)))
                     ((OR (= CH #/z) (= CH #/Z)) (SETQ A-VALUE 0)))))
        (FORMAT T "~%M-VALUE = ~O, A-VALUE = ~O" M-VALUE A-VALUE)
        (GO L)
))

(DEFUN CC-RUN-LOOP (ADR &AUX CH)
  (CC-WRITE-PC ADR)
  (CC-NOOP-CLOCK)                  ;FIRST INSTRUCTION TO IR
  (CC-CLOCK)                       ;CLOCK AGAIN
  (SPY-WRITE SPY-CLK 1) ;TAKE OFF
  (DO () ((SETQ CH (KBD-TYI-NO-HANG)))
    (PROCESS-SLEEP 15. "Running Loop"))
  (CC-STOP-MACH)
  CH)

(DEFtest CC-TEST-PC-INCREMENTER NIL "PC Incrementer"
  (DOTIMES (B 14.)
    (CC-TEST-PC-INCREMENT (1- (LSH 1 B))))
  (DOTIMES (B 13.)
    (CC-TEST-PC-INCREMENT (- (LSH 1 (1+ B)) 2))))

(DEFUN CC-TEST-PC-INCREMENT (VAL)
  (CC-WRITE-PC VAL)
  (CC-NOOP-DEBUG-CLOCK)
  (COND ((NOT (= (CC-READ-PC) (1+ VAL)))
         (FORMAT T "~% PC of ~O incremented to ~O" VAL (CC-READ-PC)))))

(DEFUN CC-TEST-USTACK-TO-PC  (N)
  (LET ((USP (CC-READ-MICRO-STACK-PTR))
        (VAL))
    (CC-WRITE-MD N)      ;GET DATA INTO MRD
    (CC-EXECUTE (WRITE)
             CONS-IR-M-SRC CONS-M-SRC-MD        ;PUSH IT
             CONS-IR-ALUF CONS-ALU-SETM
             CONS-IR-OB CONS-OB-ALU
             CONS-IR-FUNC-DEST CONS-FUNC-DEST-MICRO-STACK-PUSH)
    (CC-EXECUTE
             CONS-IR-OP CONS-OP-JUMP
             CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
             CONS-IR-R 1)
    (CC-CLOCK)
    (SETQ VAL (CC-READ-PC))
    (COND ((NOT (= USP (CC-READ-MICRO-STACK-PTR)))
           (FORMAT T "~%USP ~O BEFORE PUSH, POP; ~O AFTER"
                   USP  (CC-READ-MICRO-STACK-PTR))))
    VAL))

(DECLARE (SPECIAL SPY-OPC SPY-OPC-CONTROL))

(DEFUN CC-TEST-OPC-TRIAL (N &AUX TEM)
  (DOTIMES (C 8)
    (CC-WRITE-PC (+ N C)))
  (DOTIMES (C 8)
    (SETQ TEM (SPY-READ SPY-OPC))
    (COND ((NOT (= TEM (+ N C)))
           (FORMAT T "~%OPC #~D, Wroto ~O ;read ~O" C (+ N C) TEM)))
    (SPY-WRITE SPY-OPC-CONTROL 2)        ;CLOCK OPCS
    (SPY-WRITE SPY-OPC-CONTROL 0)))

(DEFUN CC-PRINT-OPCS-LOOP NIL
  (DO () (())
    (PRINT (SPY-READ SPY-OPC))
    (SPY-WRITE SPY-OPC-CONTROL 2)        ;CLOCK OPCS
    (SPY-WRITE SPY-OPC-CONTROL 0)))

(DEFUN CC-SETUP-DIVIDE-TEST ()
  "Load C-MEM with divide routine...
Divide two numbers.  This routine taken from UCADR 108.
Dividend in 22, divisor in 23 (same values as M-1 and M-2 for randomness).
Quotient In Q-R, remainder 22.
Clobbers 1000@A.  Zeros 2@M, 2@A"
  (CC-WRITE-M-MEM 2 0)
  (CC-EXECUTE (W-C-MEM 0)          ;HALT . in 0
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-MF CONS-MF-HALT
      CONS-IR-JUMP-ADDR 0
      CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
      CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 6))       ;a couple of no-ops to get started by
  (CC-EXECUTE (W-C-MEM 7))
  (CC-EXECUTE (W-C-MEM 10)       ;(JUMP-GREATER-OR-EQUAL-XCT-NEXT M-1 A-ZERO DIV1)
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-M-SRC 22
      CONS-IR-A-SRC 2
      CONS-IR-JUMP-ADDR 13
      CONS-IR-JUMP-COND CONS-JUMP-COND-M>=A
      CONS-IR-N 0)
  (CC-EXECUTE (W-C-MEM 11)       ; ((A-TEM1 Q-R) M-1)
      CONS-IR-M-SRC 22
      CONS-IR-A-MEM-DEST (+ CONS-A-MEM-DEST-INDICATOR 1000)
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-ALUF CONS-ALU-SETM
      CONS-IR-Q CONS-Q-LOAD)
  (CC-EXECUTE (W-C-MEM 12)       ;((Q-R) SUB M-ZERO A-TEM1)
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 1000
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-ALUF CONS-ALU-SUB
      CONS-IR-Q CONS-Q-LOAD)
  (CC-EXECUTE (W-C-MEM 13)      ;DIV1   ((M-1) DIVIDE-FIRST-STEP M-ZERO A-2)
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 23
      CONS-IR-OB CONS-OB-ALU-LEFT-1
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-DFSTEP
      CONS-IR-Q CONS-Q-LEFT)
  (CC-EXECUTE (W-C-MEM 14)      ;DIV1A  (JUMP-IF-BIT-SET (BYTE-FIELD 1 0) Q-R DIVIDE-BY-ZERO)
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-M-SRC CONS-M-SRC-Q
      CONS-IR-JUMP-COND 0       ;test bit 0
      CONS-IR-JUMP-ADDR 0
      CONS-IR-P 1
      CONS-IR-N 1)
  (DOTIMES (C 31.)              ;((M-1) DIVIDE-STEP M-1 A-2)
    (CC-EXECUTE (W-C-MEM (+ C 15))
      CONS-IR-M-SRC 22
      CONS-IR-A-SRC 23
      CONS-IR-OB CONS-OB-ALU-LEFT-1
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-DSTEP
      CONS-IR-Q CONS-Q-LEFT))
  (CC-EXECUTE (W-C-MEM (+ 15 31.))      ;((M-1) DIVIDE-LAST-STEP M-1 A-2)
      CONS-IR-M-SRC 22
      CONS-IR-A-SRC 23
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-DSTEP
      CONS-IR-Q CONS-Q-LEFT)
  (CC-EXECUTE (W-C-MEM (+ 16 31.))      ;(JUMP-LESS-OR-EQUAL-XCT-NEXT M-ZERO A-TEM1 DIV2)
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 1000
      CONS-IR-JUMP-ADDR (+ 3 16 31.)
      CONS-IR-JUMP-COND CONS-JUMP-COND-M<=A
      CONS-IR-N 0)
  (CC-EXECUTE (W-C-MEM (+ 17 31.))      ;((M-1) DIVIDE-REMAINDER-CORRECTION-STEP M-1 A-2)
      CONS-IR-M-SRC 22
      CONS-IR-A-SRC 23
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-RSTEP)
  (CC-EXECUTE (W-C-MEM (+ 20 31.))      ;((M-1) SUB M-ZERO A-1)
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 22
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-SUB)
  (CC-EXECUTE (W-C-MEM (+ 21 31.))      ;DIV2   ((A-TEM1) XOR M-2 A-TEM1)
      CONS-IR-M-SRC 23
      CONS-IR-A-SRC 1000
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-A-MEM-DEST (+ CONS-A-MEM-DEST-INDICATOR 1000)
      CONS-IR-ALUF CONS-ALU-XOR)
  (CC-EXECUTE (W-C-MEM (+ 22 31.))      ;(POPJ-LESS-OR-EQUAL M-ZERO A-TEM1)
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 1000
      CONS-IR-JUMP-COND CONS-JUMP-COND-M<=A
      CONS-IR-R 1
      CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM (+ 23 31.))      ;(POPJ-AFTER-NEXT (A-TEM1) Q-R)
      CONS-IR-POPJ 1
      CONS-IR-M-SRC CONS-M-SRC-Q
      CONS-IR-A-MEM-DEST (+ CONS-A-MEM-DEST-INDICATOR 1000)
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-ALUF CONS-ALU-SETM)
  (CC-EXECUTE (W-C-MEM (+ 24 31.))      ;((Q-R) SUB M-ZERO A-TEM1)
      CONS-IR-M-SRC 2
      CONS-IR-A-SRC 1000
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-ALUF CONS-ALU-SUB
      CONS-IR-Q CONS-Q-LOAD)
  ;calling routine loop
  ;1000@a TEM,  1001@A dividend 1002@a divisor 1003@a correct remainder
  ;1@M counts errors.
  (CC-EXECUTE (W-C-MEM 100)
      CONS-IR-A-SRC 1001
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-M-MEM-DEST 22
      CONS-IR-ALUF CONS-ALU-SETA)
  (CC-EXECUTE (W-C-MEM 101)
      CONS-IR-A-SRC 1002
      CONS-IR-OB CONS-OB-ALU
      CONS-IR-M-MEM-DEST 23
      CONS-IR-ALUF CONS-ALU-SETA)
  (CC-EXECUTE (W-C-MEM 102)
      CONS-IR-OP CONS-OP-JUMP
```

```
       CONS-IR-JUMP-ADDR 10
       CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
       CONS-IR-P 1
       CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 103)
       CONS-IR-STAT-BIT 1
       CONS-IR-OP CONS-OP-JUMP
       CONS-IR-JUMP-ADDR 100
       CONS-IR-M-SRC 22
       CONS-IR-A-SRC 1003
       CONS-IR-JUMP-COND CONS-JUMP-COND-M=A
       CONS-IR-N 1)
  (CC-EXECUTE (W-C-MEM 104)
       CONS-IR-OP CONS-OP-JUMP
       CONS-IR-JUMP-ADDR 100
       CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
       CONS-IR-N 0)
  (CC-EXECUTE (W-C-MEM 105)
       CONS-IR-M-SRC 1
       CONS-IR-OB CONS-OB-ALU
       CONS-IR-M-MEM-DEST 1
       CONS-IR-ALUF CONS-ALU-M+1)
 )

(defun display-registers-for-debug-divide-test ()
  (cond ((variable-boundp display-registers-for-debug-divide-test-flag)
         (format T "~%A-MEM 1001 dividend  ~A    " (cc-read-a-mem 1001))
         (format T "A-MEM 1002 divisor   ~A~%" (cc-read-a-mem 1002))
         (format T "A-MEM 1003 rem       ~A    " (cc-read-a-mem 1003))
         (format T "M-MEM 1    count     ~A" (cc-read-m-mem 1))
         (format T "~%M-1                  ~A    " (cc-read-m-mem #o22))
         (format T "A-2                 ~A" (cc-read-A-mem #o23))
         (format T "~%Output Bus           ~A" (cc-read-obus)))))

;(setq  display-registers-for-debug-divide-test-flag T)

;first arg of NIL says use values in machine.
(DEFUN CC-DIVIDE-TEST-LOOP (&OPTIONAL (DIVIDEND (RANDOM 37777777))
                                      (DIVISOR (RANDOM 37777777)))
  (LET ((REM (IF DIVIDEND (\ DIVIDEND DIVISOR))))
    (CC-WRITE-M-MEM 1 0)                              ;error count
    (IF (NUMBERP DIVIDEND)
        (PROGN (CC-WRITE-A-MEM 1001 DIVIDEND)
               (CC-WRITE-A-MEM 1002 DIVISOR)
               (CC-WRITE-A-MEM 1003 REM)))
    (CC-WRITE-STAT-COUNTER -40000.) ;times around loop
    (CC-RUN-TEST-LOOP 100)
    (CC-READ-M-MEM 1))
)

(DEFUN CC-DIVIDE-SAVE-STATE NIL
  (LIST (CC-READ-A-MEM 1001) (CC-READ-A-MEM 1002) (CC-READ-A-MEM 1003)))

(DEFUN CC-DIVIDE-RESTORE-STATE (STATE)
  (CC-WRITE-A-MEM 1001 (CAR STATE))
  (CC-WRITE-A-MEM 1002 (CADR STATE))
  (CC-WRITE-A-MEM 1003 (CADDR STATE)))

(DEFUN CC-DIVIDE-COMPARE-STATE (STATE &AUX TEM)
  (IF (NOT (= (SETQ TEM (CC-READ-A-MEM 1001)) (CAR STATE)))
      (FORMAT T "~%1001 CLOBBERED FROM ~S TO ~S" TEM (CAR STATE)))
  (IF (NOT (= (SETQ TEM (CC-READ-A-MEM 1002)) (CADR STATE)))
      (FORMAT T "~%1002 CLOBBERED FROM ~S TO ~S" TEM (CADR STATE)))
  (IF (NOT (= (SETQ TEM (CC-READ-A-MEM 1003)) (CADDR STATE)))
      (FORMAT T "~%1003 CLOBBERED FROM ~S TO ~S" TEM (CADDR STATE))))

(comment
(DEFUN CC-DIVIDE-TEST-LOOP-STATE NIL
  (LIST (CC-READ-A-MEM 1001) (CC-READ-A-MEM 1002))) )

(DEFUN CC-DIVIDE-RESTORE-STATE-AND-DIAGNOSE (S)
  (DBG-RESET)
  (CC-RESET-MACH)
  (CC-ZERO-ENTIRE-MACHINE)
  (CC-SETUP-DIVIDE-TEST)
  (APPLY (FUNCTION CC-DIVIDE-TEST-LOOP) S)
  (CC-DIVIDE-DIAGNOSE))

(DEFUN CC-DIVIDE-TEST ()
  (DO ((TEM)) (())
    (IF (NOT (ZEROP (SETQ TEM (CC-DIVIDE-TEST-LOOP))))
        (RETURN TEM))))


;use this if divide works at ultra slow speed and fails at normal speed.  Args
; that fail should already be loaded as per above test loop.
;Running at ultra slow speed, this builds a table output-bus versus PC.
```

```
;Then, running at normal speed, it samples machine and tries to find the
;lowest PC where output bus has wrong thing.
(DEFUN CC-DIVIDE-DIAGNOSE ()
  (PROG (HIST PC OBUS INST TEM LOWEST-PC LOWEST-PC-OBUS GOOD-COMPARISONS BAD-COMPARISONS)
        (CC-SET-SPEED 0)
        (CC-COLON-START 100)
        (DOTIMES (C 1000)
          (CC-STOP-MACH)
          (SETQ PC (CC-READ-PC) OBUS (CC-READ-OBUS) INST (CC-READ-IR))
          (IF (NOT (= (LDB CONS-IR-OP INST) CONS-OP-JUMP))
              (IF (SETQ TEM (ASSQ PC HIST))
                  (IF (NOT (= (CDR TEM) OBUS))
                      (COMMENT (PROGN (FORMAT T "~%Multiple values observed at PC ~O, ~O ~O "
                                              PC OBUS (CDR TEM))
                              (CC-PRINT-BITS (LOGXOR OBUS (CDR TEM))))))
                  (SETQ HIST (CONS (CONS PC OBUS) HIST))))
          (SPY-WRITE SPY-CLK 1))          ;continue
        (CC-STOP-MACH)
        (CC-SET-SPEED 2)         ;normal
        (SPY-WRITE SPY-CLK 1)
        (SETQ GOOD-COMPARISONS 0 BAD-COMPARISONS 0)
        (DOTIMES (C 1000)
          (CC-STOP-MACH)
          (SETQ PC (CC-READ-PC) OBUS (CC-READ-OBUS) INST (CC-READ-IR))
          (IF (NOT (= (LDB CONS-IR-OP INST) CONS-OP-JUMP))
              (IF (SETQ TEM (ASSQ PC HIST))
                  (IF (NOT (= OBUS (CDR TEM)))
                      (PROGN (SETQ BAD-COMPARISONS (1+ BAD-COMPARISONS))
                             (IF (OR (NULL LOWEST-PC)
                                     (< PC LOWEST-PC))
                                 (SETQ LOWEST-PC PC LOWEST-PC-OBUS OBUS)))
                      (SETQ GOOD-COMPARISONS (1+ GOOD-COMPARISONS)))))
          (SPY-WRITE SPY-CLK 1))
        (CC-STOP-MACH)
        (IF LOWEST-PC
            (PROGN (FORMAT T "~%Lowest PC at error ~O, OBUS ~O, should be ~O"
                           LOWEST-PC LOWEST-PC-OBUS (CDR (ASSQ LOWEST-PC HIST)))
                   (FORMAT T "~%bits wrong ")
                   (CC-PRINT-BITS (LOGXOR LOWEST-PC-OBUS (CDR (ASSQ LOWEST-PC HIST))))))
        (FORMAT T "~%Length of HIST ~S, good comps ~S, bad comps ~S"
                (LENGTH HIST) GOOD-COMPARISONS BAD-COMPARISONS)
  ))

;THIS DOESNT SEEM TO WORK JUST YET.
(DEFUN CC-PDL-BUFFER-PUSH-POP-CHECK ()
  (DBG-RESET)
  (CC-RESET-MACH)
  (CC-EXECUTE (W-C-MEM 100)
      CONS-IR-FUNC-DEST CONS-FUNC-DEST-PDL-BUFFER-PUSH)
  (CC-EXECUTE (W-C-MEM 101)
      CONS-IR-M-SRC CONS-M-SRC-C-PDL-BUFFER-POINTER-POP)
  (CC-EXECUTE (W-C-MEM 102)
      CONS-IR-OP CONS-OP-JUMP
      CONS-IR-JUMP-ADDR 100
      CONS-IR-JUMP-COND CONS-JUMP-COND-UNC
      CONS-IR-N 1)
  (LET ((PP 1777) PC RPP INCR IR)
    (CC-WRITE-PDL-BUFFER-POINTER PP)
    (CC-SET-SPEED 2)
    (CC-COLON-START 100)
    (DOTIMES (C 1000)
      (CC-STOP-MACH)
      (SETQ PC (CC-READ-PC)
            IR (CC-READ-IR)
            RPP (CC-READ-PDL-BUFFER-POINTER))
      (SETQ INCR (CDR (ASSQ PC '((100 . 0) (101 . 1) (102 . 0) (103 . 0)))))
      (IF (NULL INCR)
          (FORMAT T "~%PC was random ~S" PC)
          (IF (NOT (= (LOGAND 1777 (+ PP INCR)) RPP))
              (FORMAT T "~%PP Wrong, was ~O, should be ~O" RPP (LOGAND 1777 (+ PP INCR)))))
      (CC-WRITE-IR IR)
      (CC-WRITE-PC PC)
      (CC-CLOCK)
      (SPY-WRITE SPY-CLK 1))    ;CONTINUE
    ))
```

```
;;; Keyboard Tests
(DEFVAR KEY-BITS
        '((#/4 11)
          (#\PLUS-MINUS 21)
          (#\NETWORK 42)
          (#\MACRO 100)
          (#/C 164)))

(DEFVAR *TEST-LOCAL-KEYBOARD* NIL)
(DEFUN KEYBOARD-DBG-READ (ADR)
  (IF *TEST-LOCAL-KEYBOARD* (%UNIBUS-READ ADR) (DBG-READ ADR)))

(DEFUN KEYBOARD-DBG-WRITE (ADR DATA)
  (IF *TEST-LOCAL-KEYBOARD* (%UNIBUS-WRITE ADR DATA) (DBG-WRITE ADR DATA)))

(DEFUN TEST-IO-KEYBOARD ()
  (KEYBOARD-DBG-READ 764100)                          ;Clear out keyboard
  (IF (LDB-TEST 0501 (KEYBOARD-DBG-READ 764112))
      (FORMAT T "~&Keyboard ready did not clear when read"))
  (DOLIST (L KEY-BITS)
    (APPLY 'TEST-KEY L))
  )

(DEFUN TEST-KEY (KEY VALUE)
  (FORMAT T "~&Hold down the ~:C key on the debugee and then type space on this keyboard."
          KEY)
  (SEND STANDARD-INPUT ':TYI)
  (LET ((READ-KEY (KEYBOARD-DBG-READ 764100)))
    (IF (≠ READ-KEY VALUE)
        (FORMAT T "Keyboard should have been ~O and was ~O" VALUE READ-KEY))))

(DEFUN CC-TEST-IO-BOARD (&OPTIONAL (*TEST-LOCAL-KEYBOARD* *TEST-LOCAL-KEYBOARD*))
  (FORMAT T "~&Testing Time of day clock")
  (CHECK-ANDS-AND-OR 764120 16. 1000. "Time of day")
  ;; Enable remote mouse
  (KEYBOARD-DBG-WRITE 764112 1)
  (FORMAT T
          "~&Testing mouse Y direction, roll mouse upwards for a while
and then type space")
  (CHECK-ANDS-AND-OR 764104 12. NIL "Mouse Y position")
  (FORMAT T
          "~&Testing mouse X direction, roll mouse sideways for a while
and then type space")
  (CHECK-ANDS-AND-OR 764106 12. NIL "Mouse X position")
  (FORMAT T "~&Testing console beeper, should be beeping")
  (LOOP DO (KEYBOARD-DBG-READ 764110) UNTIL (SEND STANDARD-INPUT ':TYI-NO-HANG))
  (FORMAT T "~&Testing Chaosnet interface")
  (LET ((CHAOS:CHATST-USE-DEBUG (NOT *TEST-LOCAL-KEYBOARD*)))
    (CHAOS:CHATST)))

(DEFUN CHECK-ANDS-AND-OR (ADDR BITS ITERATION NAME)
  (LET* ((MASK (1- (^ 2 BITS)))
         (AND MASK)
         (OR 0))
    (DO ((I 0 (1+ I))
         (RES))
        ((IF (NULL ITERATION)
             (SEND STANDARD-INPUT ':TYI-NO-HANG)
             (≥ I ITERATION)))
      (SETQ RES (LOGAND MASK (KEYBOARD-DBG-READ ADDR))
            OR (LOGIOR OR RES)
            AND (LOGAND AND RES)))
    (IF (OR (≠ AND 0) (≠ OR MASK))
        (FORMAT T "~&Bits in the ~A register not changing.~% LOGAND=~O LOGIOR=~O"
                NAME AND OR))))

(DEFCONST *SERIAL-IO-TESTS*
          '((((:BAUD 1200.) (:PARITY :ODD)
             (:NUMBER-OF-DATA-BITS 7) (:NUMBER-OF-STOP-BITS 2))
            ((:BAUD 9600.) (:PARITY :EVEN)
             (:NUMBER-OF-DATA-BITS 8) (:NUMBER-OF-STOP-BITS 1))))

(DEFUN TEST-SERIAL-IO ()
  (LET ((STREAM NIL))
    (UNWIND-PROTECT
      (PROGN
        (SETQ STREAM (SI:MAKE-SERIAL-STREAM
                       ':NUMBER-OF-STOP-BITS 1
                       ':PARITY ':ODD))
        (DOLIST (PROP '(:CHECK-PARITY-ERRORS :CHECK-OVER-RUN-ERRORS :CHECK-FRAMING-ERRORS))
          (FUNCALL STREAM ':PUT PROP T))
        (FORMAT T "~&Testing serial I/O using /"remote loop back/" in the UART.")
        (UNWIND-PROTECT
          (PROGN
            (FUNCALL STREAM ':PUT ':LOCAL-LOOP-BACK T)
            (TEST-SERIAL-IO-SERIES STREAM *SERIAL-IO-TESTS*))
          (FUNCALL STREAM ':PUT ':LOCAL-LOOP-BACK NIL))
        (FORMAT T "~2&Attach a loop-back plug; type N if you don't want to do this test,
or any other character to run the test.")
        (LET ((CHAR (SEND STANDARD-INPUT ':TYI)))
          (COND ((NOT (CHAR-EQUAL #/N CHAR))
                 (FORMAT T "~&Testing extra EIA-RS-232 bits.")
                 (TEST-SERIAL-IO-EIA-RS-232-BITS STREAM)
                 (TEST-SERIAL-IO-SERIES STREAM *SERIAL-IO-TESTS*)))))
      (CLOSE STREAM))))

(DEFVAR *SERIAL-IO-ERROR-COUNT*)
(DEFCONST *SERIAL-IO-ERROR-LIMIT* 5)

(DEFUN TEST-SERIAL-IO-SERIES (STREAM SERIES)
  (DOLIST (TEST SERIES)
    (LET ((BASE 10.)
          (FIRST T)
          (*SERIAL-IO-ERROR-COUNT* 0))
      (FORMAT T "~&")
      (DOLIST (CLAUSE TEST)
        (LET ((NAME (FIRST CLAUSE))
              (VALUE (SECOND CLAUSE)))
          (IF (NOT FIRST)
              (FORMAT T "; "))
          (SETQ FIRST NIL)
          (FORMAT T "~S = ~S" NAME VALUE)
          (FUNCALL STREAM ':PUT NAME VALUE)))
      (TEST-SERIAL-IO-CHARS STREAM))))

(DEFCONST *SERIAL-IO-TIMEOUT* 60.)

(DEFUN TEST-SERIAL-IO-CHARS (STREAM)
  (DOTIMES (SENT-CHAR (^ 2 (FUNCALL STREAM ':GET ':NUMBER-OF-DATA-BITS)))
    (FUNCALL STREAM ':TYO SENT-CHAR)
    (COND ((PROCESS-WAIT-WITH-TIMEOUT "Serial In" *SERIAL-IO-TIMEOUT* STREAM ':LISTEN)
           (LET ((GOT-CHAR (FUNCALL STREAM ':TYI)))
             (COND ((NOT (= SENT-CHAR GOT-CHAR))
                    (FORMAT T "~&Error: sent ~O and got back ~O (both octal)~%"
                            SENT-CHAR GOT-CHAR)
                    (INCF *SERIAL-IO-ERROR-COUNT*)
                    (COND ((< *SERIAL-IO-ERROR-COUNT* *SERIAL-IO-ERROR-LIMIT*)
                           (FORMAT T "~&Status of serial I//O line:~%")
                           (SI:SERIAL-STATUS)))))))
           (T
             (FORMAT T "~&Error: timed out waiting for character ~O (octal)~%"
                     SENT-CHAR)))))

;;; Unfortunately, you can't read back clear-to-send (the LM-2 Serial I/O
;;; documentation is wishful thinking).
(DEFUN TEST-SERIAL-IO-EIA-RS-232-BITS (STREAM)
  (LOOP FOR SET IN '(:DATA-TERMINAL-READY :DATA-TERMINAL-READY)
        FOR GET IN '(:DATA-SET-READY       :CARRIER-DETECT)
        DO
        (FUNCALL STREAM ':PUT SET NIL)
        (IF (NOT (NULL (FUNCALL STREAM ':GET GET)))
            (FORMAT T "~&Error: Sent zero on ~S and got one on ~S.~%" SET GET))
        (FUNCALL STREAM ':PUT SET T)
        (IF (NULL (FUNCALL STREAM ':GET GET))
            (FORMAT T "~&Error: Sent one on ~S and got zero on ~S.~%" SET GET)))
  ;; Fix world.
  (FUNCALL STREAM ':PUT ':REQUEST-TO-SEND T)
  (FUNCALL STREAM ':PUT ':DATA-TERMINAL-READY T))
```

```
;;; -*- Mode: Lisp; Package: CADR; Base: 8 -*-
;;; Routines for hacking the pseudo-debugger          -*-LISP-*-

;;; The following are the active locations:
;;; 766100  Reads or writes the debuggee-Unibus location addressed by the registers below.
;;; 766114  (Write only) Contains bits 1-16 of the debuggee-Unibus address
;;;         to be accessed.  Bit 0 of the address is always zero.
;;; 766110  (Write only) Contains additional modifier bits, as follows.
;;;         These bits are reset to zero when the debuggee's Unibus is reset.
;;;         1  Bit 17 of the debuggee-Unibus address.
;;;         2  Resets the debuggee's Unibus and bus interface.  Write a 1 here then write a 0.
;;;       * 4  Timeout inhibit.  This turns off the NXM timeout for all Xbus and Unibus cycles
;;;            done by the debuggee's bus interface (not just those by the debugger).
;;; 766104  (Read only) These contain the status for bus cycles executed on the debuggee's
;;;         busses.  These bits are cleared by writing into location 766044 (Error Status)
;;;         on the debuggee's Unibus.  They are not cleared by power up.
;;;         1  Xbus NXM Error.  Set when an Xbus cycle times out for lack of response.
;;;         2  Xbus Parity Error.  Set when an Xbus read receives a word with bad parity,
;;;            and the Xbus ignore-parity line was not asserted.  Parity Error is also set
;;;            by Xbus NXM Error.
;;;         4  CADR Address Parity Error.  Set when an address received from the processor
;;;            has bad parity.
;;;        10  Unibus NXM Error.  Set when a Unibus cycle times out for lack of response.
;;;        20  CADR Parity Error.  Set when data received from the processor has bad parity.
;;;        40  Unibus Map Error.  Set when an attempt to perform an Xbus cycle through the
;;;            Unibus map is refused because the map specifies invalid or write-protected.
;;;        The remaining bits are random (not necessarily zero).

(DEFVAR SERIAL-STREAM)          ;when DBG-ACCESS-PATH = SERIAL

(DECLARE (SPECIAL DBG-NXM-INHIBIT DBG-ACCESS-PATH DBG-SERIAL-HIGH-BIT DBG-HOST
                  DBG-CHAOS-STRING DBG-CHAOS-16 DBG-UNIQUE-ID))
(SETQ DBG-NXM-INHIBIT NIL
      DBG-ACCESS-PATH 'BUSINT                    ;Possible values: BUSINT, SERIAL, CHAOS
      DBG-SERIAL-HIGH-BIT -1
      DBG-HOST NIL
      DBG-CHAOS-STRING (MAKE-ARRAY NIL 'ART-STRING
                                   (* 2 CHAOS:MAX-DATA-WORDS-PER-PKT) NIL '(2))
      DBG-CHAOS-16 (MAKE-ARRAY NIL 'ART-16B  CHAOS:MAX-DATA-WORDS-PER-PKT DBG-CHAOS-STRING)
      DBG-UNIQUE-ID NIL)

;;; Reset the state of the internal variables
(DEFUN DBG-CC-RESET ()
  (SETQ DBG-SERIAL-HIGH-BIT -1
        DBG-UNIQUE-ID NIL)
  (COND ((AND (BOUNDP 'SERIAL-STREAM) SERIAL-STREAM)
         ;; Set the serial line back to 300. baud. since it resets
         ;;itself when reset via the hardware
         (FUNCALL SERIAL-STREAM ':PUT ':BAUD 300.)
         (FUNCALL SERIAL-STREAM ':CLEAR-INPUT))))

;;; Read a location on the debuggee's Unibus
(DEFUN DBG-READ (ADR &OPTIONAL (CHAOS-DBG-TYPE 'DATA))
  (SELECTQ DBG-ACCESS-PATH
    (SERIAL
      (DBG-UPDATE-HIGH-BIT ADR)
      (FORMAT #'CC-SERIAL-STREAM "~O//" (LSH ADR -1))
      (READ #'CC-SERIAL-STREAM))
    (BUSINT
      (%UNIBUS-WRITE 766110 (+ (LSH ADR -17.) (COND (DBG-NXM-INHIBIT 4) (T 0))))
      (%UNIBUS-WRITE 766114 (LSH ADR -1))
      (%UNIBUS-READ 766100))
    (CHAOS
      (LET ((PKT (DBG-CHAOS CHAOS-DBG-TYPE ADR)))
        (PROG1 (AREF PKT CHAOS:FIRST-DATA-WORD-IN-PKT)
               (CHAOS:RETURN-PKT PKT))))
    (OTHERWISE (FERROR NIL "~A is illegal DBG-ACCESS-PATH" DBG-ACCESS-PATH))))

;;; Write a location on the debuggee's Unibus
(DEFUN DBG-WRITE (ADR VAL &OPTIONAL (CHAOS-DBG-TYPE 'DATA))
  (SETQ VAL (LOGAND VAL 177777))
  (SELECTQ DBG-ACCESS-PATH
    (SERIAL
      (DBG-UPDATE-HIGH-BIT ADR)
      (FORMAT #'CC-SERIAL-STREAM "~O:~O:" (LSH ADR -1) VAL))
    (BUSINT
      (%UNIBUS-WRITE 766110 (+ (LSH ADR -17.) (COND (DBG-NXM-INHIBIT 4) (T 0))))
      (%UNIBUS-WRITE 766114 (LSH ADR -1))
      (%UNIBUS-WRITE 766100 VAL))
    (CHAOS
      (DBG-CHAOS CHAOS-DBG-TYPE ADR VAL))
    (OTHERWISE (FERROR NIL "~A is illegal DBG-ACCESS-PATH" DBG-ACCESS-PATH)))
  T)

;;; Reset the debuggee's Unibus
(DEFUN DBG-RESET ()
  (SETQ CC-UNIBUS-MAP-TO-MD-OK-FLAG NIL)
  (SELECTQ DBG-ACCESS-PATH
    (SERIAL
      (FORMAT #'CC-SERIAL-STREAM "2S")
      (FORMAT #'CC-SERIAL-STREAM "~OS" (+ (ABS DBG-SERIAL-HIGH-BIT)
                                          (COND (DBG-NXM-INHIBIT 4) (T 0)))))
    (BUSINT (%UNIBUS-WRITE 766110 2)
            (%UNIBUS-WRITE 766110 (COND (DBG-NXM-INHIBIT 4) (T 0))))
    (CHAOS
     (DBG-CHAOS 'RESET 0 0))
    (OTHERWISE (FERROR NIL "~A is illegal DBG-ACCESS-PATH" DBG-ACCESS-PATH)))
  T)

;;; Print the error status bits
(DEFUN DBG-PRINT-STATUS ()
   (CC-PRINT-SET-BITS (SELECTQ DBG-ACCESS-PATH
                          (SERIAL (FUNCALL #'CC-SERIAL-STREAM ':TYO #/R)
                                  (READ #'CC-SERIAL-STREAM))
                          (BUSINT (%UNIBUS-READ 766104))
                          (CHAOS (LET ((PKT (DBG-CHAOS 'STATUS 0)))
                                   (PROG1 (AREF PKT CHAOS:FIRST-DATA-WORD-IN-PKT)
                                          (CHAOS:RETURN-PKT PKT))))
                          (OTHERWISE
                            (FERROR NIL "~A is illegal DBG-ACCESS-PATH" DBG-ACCESS-PATH)))
                        '(XBUS-NXM-ERR XBUS-PARITY-ERR CADR-ADDRESS-PARITY-ERR
                          UNIBUS-NXM-ERR CADR-DATA-PARITY-ERR UNIBUS-MAP-ERR
                          NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)))

;;; Reset the error status
(DEFUN DBG-RESET-STATUS ()
  (SELECTQ DBG-ACCESS-PATH
    (SERIAL)
    (BUSINT (DBG-WRITE 766044 0))
    (CHAOS (DBG-CHAOS 'STATUS 0 0))))

;;; Setup the high bit of the SERIAL debugger correctly
(DEFUN DBG-UPDATE-HIGH-BIT (ADR &AUX (HIGH (LDB 2101 ADR)))
  (OR (= HIGH DBG-SERIAL-HIGH-BIT)
      (FORMAT #'CC-SERIAL-STREAM "~OS" (+ (SETQ DBG-SERIAL-HIGH-BIT HIGH)
                                          (COND (DBG-NXM-INHIBIT 4) (T 0))))))
;;; Dummy stream for SERIAL I/O
(DEFVAR CC-SERIAL-TRACE NIL)
(DEFVAR CC-SERIAL-LAST-DIRECTION NIL)

(DEFPROP CC-SERIAL-STREAM T 'SI:IO-STREAM-P)

(DEFSELECT CC-SERIAL-STREAM
  (:TYO (CHAR)
  ;; Don't do this at load time since it doesn't work if the machine doesn't have
  ;; the serial interface hardware
        (OR (BOUNDP 'SERIAL-STREAM)
            (cc-setup-serial-stream))
        (COND (CC-SERIAL-TRACE
               (COND ((NOT (EQ CC-SERIAL-LAST-DIRECTION 'OUTPUT))
                      (SETQ CC-SERIAL-LAST-DIRECTION 'OUTPUT)
                      (FORMAT TERMINAL-IO " OUTPUT ")))
               (FUNCALL TERMINAL-IO ':TYO CHAR)))
        (FUNCALL SERIAL-STREAM ':TYO (CHARACTER-ODD-PARITY CHAR)))
  (:TYI (&OPTIONAL IGNORE)
        (OR (BOUNDP 'SERIAL-STREAM)
            (cc-setup-serial-stream))
        (DO ((CHAR (FUNCALL SERIAL-STREAM ':TYI) (FUNCALL SERIAL-STREAM ':TYI)))
            (NIL)
  ;         (OR (ODDP (CHARACTER-PARITY CHAR))
  ;             (FERROR NIL "BAD PARITY RECEIVED - ~O" CHAR))
          (COND (CC-SERIAL-TRACE
               (COND ((NOT (EQ CC-SERIAL-LAST-DIRECTION 'INPUT))
                      (SETQ CC-SERIAL-LAST-DIRECTION 'INPUT)
                      (FORMAT TERMINAL-IO " INPUT ")))
               (FUNCALL TERMINAL-IO ':TYO CHAR)))
          (SELECTQ (SETQ CHAR (LOGAND CHAR 177))
            (7 (FERROR NIL "ERRONEOUS COMMAND RECEIVED BY DEBUGGER"))
            (10 (FUNCALL SERIAL-STREAM ':TYO 33)
                (FERROR NIL "DEBUGGER GOT PARITY ERROR, RESETTING DEBUGGER"))
            (15)
            (T (RETURN CHAR)))))
  (:STRING-OUT (STRING &OPTIONAL (START 0) END)
        (OR (BOUNDP 'SERIAL-STREAM)
            (cc-setup-serial-stream))
        (OR END (SETQ END (ARRAY-ACTIVE-LENGTH STRING)))
        (COND (CC-SERIAL-TRACE
               (COND ((NOT (EQ CC-SERIAL-LAST-DIRECTION 'OUTPUT))
                      (SETQ CC-SERIAL-LAST-DIRECTION 'OUTPUT)
                      (FORMAT TERMINAL-IO " OUTPUT ")))
               (FUNCALL TERMINAL-IO ':STRING-OUT STRING START END)))
        (DO ((IDX START (1+ IDX)))
            ((>= IDX END))
          (FUNCALL SERIAL-STREAM ':TYO (CHARACTER-ODD-PARITY (AREF STRING IDX))))))
```

```
(defun cc-setup-serial-stream nil
  (SETQ SERIAL-STREAM (SI:MAKE-SERIAL-STREAM ':PARITY NIL ':NUMBER-OF-DATA-BITS 8
                                             ':BAUD 1200.)))
(defun cc-serial-set-speed (baud)
  (let ((num-baud (find-position-in-list baud '(50. 75. 110. 134. 150. 300. 600.
                                               1200. 1800. 2000. 2400. 3600.
                                               4800. 7200. 9600. 19200.))))
    (format #'cc-serial-stream "~OA" num-baud)
    (funcall serial-stream ':put ':baud baud)))

(DEFUN CHARACTER-PARITY (CHAR &AUX (PARITY 0))
  (DOTIMES (I 8.)
    (SETQ PARITY (LOGXOR CHAR PARITY))
    (SETQ CHAR (LSH CHAR -1)))
  (LOGAND PARITY 1))

(DEFUN CHARACTER-ODD-PARITY (CHAR)
  (DPB (LOGXOR 1 (CHARACTER-PARITY (LOGAND CHAR 177))) 0701 CHAR))

;;; DBG-CHAOS: Take a debug cycle over the Chaos net
;;; First arg is type of cycle (DATA, STATUS, RESET, ANALOG, INTERNAL-8748,EXTERNAL-8748)
;;;  second is address, third is value
(DEFUN DBG-CHAOS (TYPE ADR &OPTIONAL DATA
                  &AUX PKT (TIMEOUT (COND (DBG-NXM-INHIBIT 4) (T 0))))
  (SETQ TIMEOUT (LOGIOR TIMEOUT (LDB 2101 ADR)))
  (COND ((NULL DBG-HOST)
         (FORMAT QUERY-IO "~&Chaos host to debug? ")
         (SETQ DBG-HOST (CHAOS:ADDRESS-PARSE (READLINE QUERY-IO)))))
  (COND ((= (ARRAY-LEADER DBG-CHAOS-STRING 0) 2)
         (COND ((OR (NULL DBG-UNIQUE-ID) (> DBG-UNIQUE-ID 370))
                (ASET #/$ DBG-CHAOS-STRING 0)
                (ASET #\SPACE DBG-CHAOS-STRING 1)
                (ASET 1_8. DBG-CHAOS-16 1)                  ;0 Unique ID, Reset
                (STORE-ARRAY-LEADER 4 DBG-CHAOS-STRING 0)
                (CHAOS:RETURN-PKT (CHAOS:SIMPLE DBG-HOST DBG-CHAOS-STRING))
                (STORE-ARRAY-LEADER 2 DBG-CHAOS-STRING 0)
                (SETQ DBG-UNIQUE-ID 0)))
         (SETQ DBG-UNIQUE-ID (1+ DBG-UNIQUE-ID))
         (ASET (+ 1_8. DBG-UNIQUE-ID) DBG-CHAOS-16 1)
         (ASET #/$ DBG-CHAOS-STRING 0)
         (ASET #\SPACE DBG-CHAOS-STRING 1)
         (STORE-ARRAY-LEADER 4 DBG-CHAOS-STRING 0)))
  (SELECTQ TYPE
    (RESET (SETQ TIMEOUT (LOGIOR TIMEOUT DATA))
           (SETQ TYPE 120))
    (DATA (SETQ TYPE (IF DATA 200 000)))
    (ANALOG (SETQ TYPE 040)                         ; ADR specifies which channel
            (SETQ ADR (LOGIOR 400 (LSH ADR 1))))
    (STATUS (SETQ TYPE (IF DATA 240 040)
                  ADR 2))
    (DEBUGGER-HIBERNATE (SETQ TYPE 040)
                        (SETQ ADR 200))
    (INTERNAL-8748 (SETQ TYPE (IF DATA 300 100))  ; DATA specifies address
                   (SETQ ADR (LSH ADR 1)))
    (EXTERNAL-8748 (SETQ TYPE (IF DATA 340 140))
                   (SETQ ADR (LSH ADR 1)))
    (OTHERWISE (FERROR NIL "Unknown request type ~S" TYPE)))
  (LET ((WORD (AREF DBG-CHAOS-16 1))
        (PTR))
    (SETQ PTR (1- (LSH WORD -8.)))
    (ASET (+ (LSH TYPE 8.) TIMEOUT) DBG-CHAOS-16 (+ 2 (* PTR 3)))
    (ASET (LSH ADR -1) DBG-CHAOS-16 (+ 3 (* PTR 3)))
    (AND DATA (ASET (LOGAND DATA 177777) DBG-CHAOS-16 (+ 4 (* PTR 3))))
    (ASET (DPB (SETQ PTR (+ PTR 2)) 1010 WORD) DBG-CHAOS-16 1)
    (COND ((OR (> (+ PTR 3) (// CHAOS:MAX-DATA-WORDS-PER-PKT 3))
               (NOT (BIT-TEST TYPE 200)))
           ;; Conservative, or a read
           (STORE-ARRAY-LEADER (+ 4 (* (1- PTR) 6)) DBG-CHAOS-STRING 0)
           (SETQ PKT (CHAOS:SIMPLE DBG-HOST DBG-CHAOS-STRING))
           (AND (BIT-TEST TYPE 200) (CHAOS:RETURN-PKT PKT))
           (STORE-ARRAY-LEADER 2 DBG-CHAOS-STRING 0)
           PKT))))

(DEFUN DBG-CHAOS-WRITE-FROB ()
  (ASET 340_8. DBG-CHAOS-16 2)
  (ASET 060 DBG-CHAOS-16 3)
  (ASET 525252 DBG-CHAOS-16 4)
  (DO () (())
    (ERRSET
      (PROGN
        (SETQ DBG-UNIQUE-ID (1+ DBG-UNIQUE-ID))
        (ASET (+ 2_8 DBG-UNIQUE-ID) DBG-CHAOS-16 1)
        (CHAOS:RETURN-PKT (CHAOS:SIMPLE DBG-HOST DBG-CHAOS-STRING)))
      NIL)))

(DEFUN DBG-ANALOG ()
   (DOLIST (X '(0 1 2 3 4 5 6 7))
     (DBG-WRITE (LOGIOR 20 X) 0 'EXTERNAL-8748)
(DBG-WRITE 30 -1 'EXTERNAL-8748)
(PRINT (LDB 0010 (DBG-READ 40 'EXTERNAL-8748)))))
```

```
;;; Higher-level operations

;;; The Unibus map is 16 words at 766140.  It consists of 14 address bits, write-ok, and valid
;;; It controls locations 140000-177777 (2000 byte locations per page).
(DEFUN DBG-READ-UNIBUS-MAP (LOC)
  (DBG-READ (+ 766140 (* 2 LOC))))

(DEFUN DBG-WRITE-UNIBUS-MAP (LOC VAL)
  (SETQ CC-UNIBUS-MAP-TO-MD-OK-FLAG NIL)        ;Caprine necrophilia
  (DBG-WRITE (+ 766140 (* 2 LOC)) VAL))

(DEFUN READ-UNIBUS-MAP (LOC)
  (%UNIBUS-READ (+ 766140 (* 2 LOC))))

(DEFUN WRITE-UNIBUS-MAP (LOC VAL)
  (%UNIBUS-WRITE (+ 766140 (* 2 LOC)) VAL))

;This run as warm initialization.  In the PDP11 slave case, it
; assures there will be no collision with PDP11 memory.
(DEFUN DBG-CLEAR-UNIBUS-MAP () "clear debugee's unibus map"
  (DOTIMES (L 16.)
    (DBG-WRITE-UNIBUS-MAP L 0)))

;(ADD-INITIALIZATION "clear unibus map" '(CLEAR-UNIBUS-MAP) '(:SYSTEM))
;CLEAR-UNIBUS-MAP called from SI:LISP-REINITIALIZE.  Do it very early to
; avoid screwwing ETHERNET code.
(DEFUN CLEAR-UNIBUS-MAP ()  "clear this machine's unibus map"
  (DOTIMES (L 16.)
    (WRITE-UNIBUS-MAP L 0)))

;; Returns unibus location mapped into specified xbus location
(DEFUN DBG-SETUP-UNIBUS-MAP (LOC XBUS-LOC)
  (DBG-WRITE-UNIBUS-MAP LOC (+ 140000 (LDB 1016 XBUS-LOC)))
  (+ 140000 (* LOC 2000) (* 4 (LOGAND 377 XBUS-LOC))))

(DEFUN SETUP-UNIBUS-MAP (LOC XBUS-LOC)
  (WRITE-UNIBUS-MAP LOC (+ 140000 (LDB 1016 XBUS-LOC)))
  (+ 140000 (* LOC 2000) (* 4 (LOGAND 377 XBUS-LOC))))

(DEFUN DBG-PRINT-UNIBUS-MAP ()
  (DO ((LOC 0 (1+ LOC))
       (CONTENTS))
      ((= LOC 20))
    (SETQ CONTENTS (DBG-READ-UNIBUS-MAP LOC))
    (PRINT LOC)
    (PRIN1-THEN-SPACE (COND ((ZEROP (LDB 1701 CONTENTS)) 'NOT-VALID) (T 'VALID)))
    (PRIN1-THEN-SPACE (COND ((ZEROP (LDB 1601 CONTENTS)) 'READ-ONLY) (T 'WRITE-OK)))
    (PRIN1 (ASH (LOGAND 37777 CONTENTS) 8))))

;;; Routines to read and write the Xbus using Unibus map location 17

(DEFVAR DBG-UNIBUS-MAP-NUMBER 17)       ;This can be changed by diagnostics

(DEFUN DBG-READ-XBUS (XBUS-LOC)
  (LET ((UBUS-LOC (DBG-SETUP-UNIBUS-MAP DBG-UNIBUS-MAP-NUMBER XBUS-LOC))
        (RES NIL))
    (SETQ RES (DBG-READ UBUS-LOC))
    (LOGDPB (DBG-READ (+ UBUS-LOC 2)) 2020 RES)))

(DEFUN DBG-WRITE-XBUS (XBUS-LOC VAL)
  (LET ((UBUS-LOC (DBG-SETUP-UNIBUS-MAP DBG-UNIBUS-MAP-NUMBER XBUS-LOC)))
    (DBG-WRITE UBUS-LOC (LOGLDB 0020 VAL))
    (DBG-WRITE (+ UBUS-LOC 2) (LDB 2020 VAL))))
```

```
;;; Accessing the interrupt-control and so forth registers
;;; 766040  Interrupt control
;;;               1  Disable Interrupt Grant
;;;               2  Local Enable (read only)
;;;            1774  Interrupt Vector of last interrupt
;;;            2000  Enable Unibus Interrupts
;;;            4000  Interrupt Stops Grants
;;;           30000  Interrupt level (0,4,5,6)
;;;           40000  Xbus interrupt (read only)
;;;          100000  Unibus Interrupt
;;;                  Only bits masked by 36001 can be written at this address.
;;; 766042  Interrupt control 2 (write only)
;;;                  Writes bits masked by 101774 of the above register.
;;; 766044  Error register (writing clears)  (see 764542 above)

(DEFUN DBG-PRINT-INTERRUPT-STATUS ()
  (LET ((INTC (DBG-READ 766040)))
    (TERPRI)
    (CC-PRINT-SET-BITS INTC
                       '( DISABLE-INT-GRANT LOCAL-ENABLE NIL NIL NIL NIL NIL NIL NIL NIL
                         ENABLE-UNIBUS-INT INT-STOPS-GRANTS NIL NIL XBUS-INT UNIBUS-INT ))
    (PRINC '| LEVEL=|)
    (PRIN1 (LOGLDB 1402 INTC))
    (PRINC '| VECTOR=|)
    (PRIN1 (LOGAND 1774 INTC)))
)
```