```
                    DDDD    EEEEE  RRRR     000     SSSS    AAA
                    D   D   E      R   R   0   0   S       A   A
                    D   D   E      R   R   0   0   S       A   A
                    D   D   EEEE   RRRR    0   0    SSS    A   A
                    D   D   E      R R     0   0       S   AAAAA
                    D   D   E      R  R    0   0       S   A   A
                    DDDD    EEEEE  R   R    000    SSSS    A   A

IIIIII    BBBBBBBB      000000    XX      XX
IIIIII    BBBBBBBB      000000    XX      XX
  II      BB      BB   00    00   XX      XX
  II      BB      BB   00    00   XX      XX
  II      BB      BB   00    00     XX  XX
  II      BB      BB   00    00     XX  XX
  II      BBBBBBBB     00    00       XX
  II      BBBBBBBB     00    00       XX
  II      BB      BB   00    00     XX  XX
  II      BB      BB   00    00     XX  XX
  II      BB      BB   00    00   XX      XX      ....
  II      BB      BB   00    00   XX      XX      ....
IIIIII    BBBBBBBB      000000    XX      XX      ....
IIIIII    BBBBBBBB      000000    XX      XX      ....

MM      MM    CCCCCCCC  RRRRRRRR        ;;;;      666666
MM      MM    CCCCCCCC  RRRRRRRR        ;;;;      666666
MMMM  MMMM  CC          RR      RR      ;;;;    66
MMMM  MMMM  CC          RR      RR      ;;;;    66
MM  MM  MM  CC          RR      RR              66
MM  MM  MM  CC          RR      RR              66
MM      MM  CC          RRRRRRRR        ;;;;    66666666
MM      MM  CC          RRRRRRRR        ;;;;    66666666
MM      MM  CC          RR  RR          ;;;;    66      66
MM      MM  CC          RR  RR          ;;;;    66      66
MM      MM  CC          RR    RR          ;;    66      66
MM      MM  CC          RR    RR          ;;    66      66
MM      MM    CCCCCCCC  RR      RR      ;;        666666
MM      MM    CCCCCCCC  RR      RR      ;;        666666

                    DDDD    EEEEE  RRRR     000     SSSS    AAA
                    D   D   E      R   R   0   0   S       A   A
                    D   D   E      R   R   0   0   S       A   A
                    D   D   EEEE   RRRR    0   0    SSS    A   A
                    D   D   E      R R     0   0       S   AAAAA
                    D   D   E      R  R    0   0       S   A   A
                    DDDD    EEEEE  R   R    000    SSSS    A   A
```

V3.73

VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS
VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS
VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS

VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS
VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS
VAX/VMS DEROSA IBOX 10-AUG-1984 15:54 LPA0: 10-AUG-1984 15:54 LAUREL:[DEROSA.IBOX.MICROCODE]IBOX.MCR;6 VAX/VMS

```
                DDDD    EEEEE   RRRR     000     SSSS    AAA
                D   D   E       R   R    0   0   S       A   A
                D   D   E       R   R    0   0   S       A   A
                D   D   EEEE    RRRR     0   0    SSS    A   A
                D   D   E       R R      0   0       S   AAAAA
                D   D   E       R  R     0   0       S   A   A
                DDDD    EEEEE   R   R     000    SSSS    A   A

      IIIIII    BBBBBBBB      000000    XX      XX
      IIIIII    BBBBBBBB      000000    XX      XX
        II      BB      BB    00  00    XX      XX
        II      BB      BB    00  00    XX      XX
        II      BB      BB    00  00      XX  XX
        II      BB      BB    00  00      XX  XX
        II      BBBBBBBB      00  00        XX
        II      BBBBBBBB      00  00        XX
        II      BB      BB    00  00      XX  XX
        II      BB      BB    00  00      XX  XX
        II      BB      BB    00  00    XX      XX      ....
        II      BB      BB    00  00    XX      XX      ....
      IIIIII    BBBBBBBB      000000    XX      XX      ....
      IIIIII    BBBBBBBB      000000    XX      XX      ....

MM      MM    CCCCCCCC  RRRRRRRR      ;;;;      666666
MM      MM    CCCCCCCC  RRRRRRRR      ;;;;      666666
MMMM  MMMM  CC          RR      RR    ;;;;    66
MMMM  MMMM  CC          RR      RR    ;;;;    66
MM  MM  MM  CC          RR      RR            66
MM  MM  MM  CC          RR      RR            66
MM      MM  CC          RRRRRRRR      ;;;;    66666666
MM      MM  CC          RRRRRRRR      ;;;;    66666666
MM      MM  CC          RR  RR        ;;;;    66      66
MM      MM  CC          RR  RR        ;;;;    66      66
MM      MM  CC          RR    RR        ;;    66      66
MM      MM  CC          RR    RR        ;;    66      66
MM      MM    CCCCCCCC  RR      RR    ;;        666666
MM      MM    CCCCCCCC  RR      RR    ;;        666666

                DDDD    EEEEE   RRRR     000     SSSS    AAA
                D   D   E       R   R    0   0   S       A   A
                D   D   E       R   R    0   0   S       A   A
                D   D   EEEE    RRRR     0   0    SSS    A   A
                D   D   E       R R      0   0       S   AAAAA
                D   D   E       R  R     0   0       S   A   A
                DDDD    EEEEE   R   R     000    SSSS    A   A
```

; Table of Contents

; IBOXDEF.MIC

```
                        ;1      .TITLE  "V3.73  --  VENUS Ibox Microcode"
                        ;2      .NOLIST
                        ;9      .LIST
                        ;10
                        ;11     .TOC    "IBOXDEF.MIC"
                        ;12     .TOC    "Revision 11.7"
                        ;13     .TOC    "22 March 1984"
                        ;14
                        ;15     .CREF                   ; Cref everything
                        ;16     .LIST
                        ;17     .RTOL                   ; Number bits right to left
                        ;18     .HEXADECIMAL            ; Hex numbering
                        ;19
          ;20   .SCODE                  ; This definition is needed to cause MICRO2, the fine
          ;21   .WIDTH/1                ; program that it is, to put identifiers into the .ULD
          ;22   A/=<0:0>                ; file!!
          ;23     FOO=1                 ;
          ;24
                        ;25     .ICODE                  ; Ibox microcode is in I-memory
                        ;26     .WIDTH/51               ; 51 (decimal) bits in Ibox microword
                        ;27     .ALLMEMFIELDS           ; Force all memory symbols into .ULD file
                        ;28     .RANDOM                 ; Random uaddress allocation and constraints
                        ;29
    ;30     .NOBIN
    ;31     .TOC    "       Revision History"
    ;32
    ;33     ;Rev    Date            Explanation
    ;34     ;---    ----            -----------
    ;35     ; 11    10-Jun-83       Fixed UMISC/, CTX CTL/, and Scoreboard documentation.  Added: New CTX CTL/ encoding for forcing
    ;36     ;                       word context for memory references.  This is for the new utrap/SII timing fix.
    ;37     ;                       Added: New UMISC2/ encodings to correct MOVL [R0J(R1), R0 scoreboard problem.  Uword width is now
    ;38     ;                       51 bits.
    ;39     ;                       Added: new "string mode" encoding to UMISC2/ for possible/probable later use by hardware.  This
    ;40     ;                       enables hardware to delete a state-machine.
    ;41     ;                       Added: new UMISC2/ microfield encoding FORCE.SET.SB.ENA.OPBUS.PARITY.
    ;42     ;                       Deleted: the "string mode" encoding in UMISC2/.  Added UTRAP CIP encoding to UMISC2/ to fix the
    ;43     ;                       INDEX IFORK CYCLE vs. Unaligned operand microtrap bug.
    ;44     ; 10    7-Jan-83        Deleted DMUX CTL/ field.
    ;45     ;                       Added SET FB OP MEM REQ encodings to UMISC/ microfield.
    ;46     ;                       Added UMISC2/ microfield (back up to 50 bits).
    ;47     ;                       Changed effects of OPVALID/GPR.IMM to fix RAF problem in S^#literal and (PC)+.
    ;48     ;                       Also, we only generate OPVALID if the operand is -RAF now.
    ;49     ;                       Also, we loop on setting RAF until Ebox asserts Ibox.Eerror to cause us to Istall.
    ;50     ;                       Computed Branch change: delete IFORK CTL/COMPBR.F.
    ;51     ;                       Added new encoding to UMISC2/ field: ENA.OPBUS.PARITY.
    ;52     ;  9    6-Sep-82        Added DMUX CTL/ field.
    ;53     ;                       Documented ucode restriction in CTX CTL/ ufield.
    ;54     ;                       Clarified documentation of IMD loading.
    ;55     ;  8    11-Mar-82       Rip out Compatibility Mode, make some minor changes to ufields as the design progresses.  Reassign
```

```
;56     ;                        ufield locations.  Add explicit GPR macros for R. Glackemeyer.  Add new CPC source encoding in the
;57     ;                        Bmux select field.  Added uMisc encoding to clear CPC valid.
;58     ;  7    19-Dec-81         Major Revision to bring definition up to the new Ibox design.  Much documentation was removed,
;59     ;                        since the new Ibox Level II & III specs. are more complete than their predecessors.
;60     ;  6    2-June-81         Major revision to bring micromachine definition up to date with the Ibox microword spec (by A.
;61     ;                        Helenius, originally from this file).  Many fields are changed/added/deleted/modified.  Hooks to
;62     ;                        support flushes are added.  UMISC/13.FLUSH.COND now conditionally loads CPC from VA if Branch_True.
;63     ;  5    10-November-80    Minor documentation fix.  Changed ABCMUXCTL/ default encoding.
;64     ;  4    31-October-80     First version considered stable and usable for coding all of Ibox microcode.  Added .DEFAULT
;65     ;                        conventions.  Added validity checks on branch combinations and meager checks on memory request
;66     ;                        encodings.
;67     ;  3    16-October-80     More changes in the misc. field encodings.  RLUG Unwind Done signal is now under microword control.
;68     ;  2    13-October-80     Results of reviews with A. Helenius, T. Knight, and C. Liu.  Changes to "WRITE.VA.READY" generation.
;69     ;                        The "WRITE.DONE" signal disappears.  Misc. encodings and microword length are changed.
;70     ;  1    2-October-80      Cleaned up and updated the information & encodings of most of the fields.  Added the UMISC field.
;71     ;                        Split the Ibox effort into three files: IBOXDEF, IBOXMACRO, and IBOXCODE.
;72     ;  0    25-August-1980    Start of History.  Initial definition of Ibox microword.  File is formatted for macro area and
;73     ;                        microcode area.
```

```
;74      .PAGE   "       Ibox Microword Definition - Dummy Fields"
;75
;76      ;
;77      ; This is used to implement the explicit GPR macros for Rich G.  It is
;78      ; not expected that the system microcode will use this.
;79      ;
;80
;81      GPRREG/=<103:100>
;82        R0=0
;83        R1=1
;84        R2=2
;85        R3=3
;86        R4=4
;87        R5=5
;88        R6=6
;89        R7=7
;90        R8=8
;91        R9=9
;92        RA=0A
;93        RB=0B
;94        RC=0C
;95        RD=0D
;96        RE=0E
;97        RF=0F
;98
;99      GPRREG32/=<103:102>
;100     GPRREG10/=<101:100>
;101
```

```
;102     .PAGE   "       Ibox Microword Definition - Real Fields"
;103
;104     ;
;105     ; Next microaddress
;106     ;
;107
;108     NEXT/=<7:0>,.NEXTADDRESS
;109     N0/=<0:0>                       ; Used in UBEN/ validity checks
;110     N10/=<1:0>                      ; Ditto
;111     N20/=<2:0>                      ; "
;112     N30/=<3:0>                      ; "
;113     N32/=<3:2>                      ; "
;114
;115
;116
;117     ;
;118     ; Ibox branch enable.  The outputs of the BEN MUXes are OR'd with NEXT<7:0>.
;119     ;
;120     ;
;121     ;    Encoding   UBEN MUX<3>     UBEN MUX<2>     UBEN MUX<1>     UBEN MUX<0>
;122     ;    ----------  -----------     -----------     -----------     -----------
;123     ;        0           0               0               0               0
;124     ;        1           0           Quad+Octa       DRAM MEM<1>     DRAM MEM<0>
;125     ;        2       RLOG first bit  DRAM CTX<2>     DRAM CTX<1>     DRAM CTX<0>
;126     ;        3           0               0               0               0
;127     ;
;128
;129     UBEN/=<9:8>,.DEFAULT=<UBEN/NOBEN>
;130       NOBEN=0                                       ;No branch (NOP).
;131       Q+O.ACCESS=1,.VALIDITY=<.EQLE<N20/>, 0]>      ;(Q+O) and Access type
;132       ACCESS=1,.VALIDITY=<.EQLE<N20/>, 4]>
;133       RLOG_FB=2,.VALIDITY=<.EQLE<N30/>, 7]>         ;Rlog First Bit
;134       CONTEXT=2,.VALIDITY=<.EQLE<N30/>, 8]>         ;Operand Context
;135       QUAD.OCTA=2,.VALIDITY=<.EQLE<N30/>, 9]>       ;If we know operand
;136                                                     ; is Quad or Octa
;137
;138
;139     ;
;140     ; Ifork Control Field.  This decides whether we should do an Ifork.  If we
;141     ; decide to do an Ifork, the UBEN/ and NEXT/ fields are ignored.
;142     ;
;143
;144     IFORK CTL/=<12:10>,.DEFAULT=<IFORK CTL/NOP>
;145       NOP=0                                 ;No Ifork
;146       MEMORY=1                              ;Ifork if Asrc+(Read*BWL)
;147       REGISTER=2                            ;Ifork if Vsrc+Write+((Read+Modify)*BWL)
;148       IMMEDIATE=3                           ;Ifork if READ*BWL
;149       QUAD=4                                ;Ifork if Quad
;150       IFORK=6,.VALIDITY=<.EQLE<UBEN/>,<UBEN/NOBEN>1>        ;Ifork DAMMIT
```

```
;151    .PAGE
;152
;153    ;
;154    ; Microtrap Control Field.  This enables a microtrap in the NEXT cycle
;155    ; based on unalignment at the IVA BUS in this cycle.
;156    ;
;157
;158    UTRAP CTL/=<14:13>,.DEFAULT=<UTRAP CTL/INHIBIT>
;159      INHIBIT=0                                 ;No unal. traps in the next cycle.
;160      INDIRECT=1,.VALIDITY=<.ANDE.ORE.EQLE<CTX CTL/>, <CTX CTL/INDIRECT>],
;161                                 .EQLE<CTX CTL/>, <CTX CTL/AUTOINC.DEF>]],
;162                               .EQLE<MCF/>, <MCF/READ_V_RCHK>]]>
;163                                 ;Trap next cycle for Unal. INDIRECT fetches
;164      OPERAND=2,.VALIDITY=<.ANDE.EQLE<IFORK CTL/>,<IFORK CTL/MEMORY>],
;165                               .EQLE<UBEN/>,<UBEN/Q+0.ACCESS>],
;166                               .EQLE<MCF/>,<MCF/CONDITIONAL>],
;167                               .EQLE<OPVALID/>,<OPVALID/MEMORY>]]>
;168                                 ;Trap next cycle for Unal. (Rn) or
;169                                 ; any other unal. operand.
;170
;171
;172    ;
;173    ; Control of the 2-input adder inputs is dictated by the AMUX SEL/ and BMUX SEL/
;174    ; microfields.
;175    ;
;176
;177    AMUX SEL/=<17:15>,.DEFAULT=<AMUX SEL/ZERO>
;178      ZERO=0
;179      AMUXPC=1                          ; Amux PC from IDP module
;180      FOUR=2                            ; +4
;181      NEGFOUR=3                         ; -4
;182      GPR=4                             ; Something selected via GPRSEL field - take from
;183                                        ; WBUS if wbus match.
;184      GPR.LS1=5                         ; Ditto, x2, stall 1 cycle if Wbus match
;185      GPR.LS2=6                         ; Ditto, x4, stall 1 cycle if Wbus match
;186      WBUS=7                            ; Take from Wbus (used during flush.wbus utrap)
;187
;188    BMUX SEL/=<20:18>,.DEFAULT=<BMUX SEL/HOLDVA>
;189      ZERO=0
;190      VA=1
;191      ADDERCTX=2                        ; Adder context as dictated by CTX.CTL field
;192      CPC=3                             ; CPC from IDP module.  This is only used for
;193                                        ; string continue flushes.
;194      DMUX.IBF=4                        ; Ibuffer data from Dmux
;195      DMUX.IMD=5                        ; IMD data from Dmux
;196      HOLDVA=7                          ; ** Not a Bmux select function - inhibit
;197                                        ; clocking of the VA LATCH.
```

```
;198    .PAGE
;199
;200    ;
;201    ; Context Control for Adder, Memory References, and Rlog entries.
;202    ; Note: the default for this field must be either CTX CTL/DRAM or CTX CTL/INDIRECT.
;203    ; See the ucode restrictions list at the end of this file.
;204    ;
;205
;206    CTX CTL/=<23:21>,.DEFAULT=<CTX CTL/DRAM>
;207
;208    ;
;209    ;                                   - - Contexts (in Bytes) - -
;210    ; Encoding              Adder               Memory Ref.     RLOG Entries
;211    ;
;212      DRAM=0                ;DRAM CTX           DRAM CTX        DRAM CTX
;213      INDIRECT=1            ;4                  4               4
;214      WORD=2                ;x                  2               x
;215      UNWIND=3              ;-RLOG CTX          x               x
;216      AUTODEC=4             ;-DRAM CTX          DRAM CTX        -DRAM CTX
;217      AUTOINC.DEF=5         ;-4                 4               x
;218
;219
;220
;221
;222    ;
;223    ; GPR address control.
;224    ;
;225
;226    GPR SEL/=<26:24>,.DEFAULT=<GPR SEL/ISTREAM>
;227      ISTREAM=0             ; GPRADD is extracted from Istream<b1> in this cycle.
;228      INDEX=1               ;Index register is being addressed by a saved Index
;229                            ; register number
;230      PREVADD+1=2           ;GPR address is previous GPR address incremented by
;231                            ; one.  Used to handle multiprecision operands.
;232      PREVADD=3             ;GPR address is previous GPR address.  Used to
;233                            ; handle autoincrement mode.
;234      RLOG_UNWIND=4         ;Asserted during RLOG unwind.
;235      EXPLICIT=5            ;GPRADD<3:2> <== CTX.CTL<1:0>
;236                            ; GPRADD<1:0> <== UNPACK.CTL<1:0>
```

```
;237    .PAGE
;238
;239    ;
;240    ; If this is set, the RLOG entry is pushed with a valid GPR number and
;241    ; context entry, and a Wbus cycle is requested.  If this is clear, an entry
;242    ; is pushed on the stack if this is the FIRST Ifork for this opcode,
;243    ; and the Wbus is conditionally requested if doing an RLOG unwind
;244    ; (GPR SEL = 3).
;245    ;
;246
;247    WBUSREQ/=<27:27>,.DEFAULT=<WBUSREQ/COND>
;248      COND=0                ; Hardware decides
;249      ON=1                  ; Force Rlog entry & Wbus request
;250
;251
;252    ;
;253    ; This marks the beginning of an Rn specifier flow for scoreboarding
;254    ; and killing OP WRT purposes.
;255    ;
;256
;257    REG MODE/=<28:28>,.DEFAULT=<REG MODE/NO>
;258      NO=0                  ; This is not Rn specifier
;259      YES=1                 ; Yes this is an Rn specifier
;260
;261
;262    ;
;263    ; OPVALID control.
;264    ;
;265    ; OPVALID will actually be sensed at least 2 cycles from now,
;266    ; even for Rn and short literal fill.
;267    ;
;268
;269    OPVALID/=<30:29>,.DEFAULT=<OPVALID/NOP>
;270      NOP=0                 ;Do not generate an OPVALID signal.
;271      GPR.IMM=1             ;Generate OPVALID if
;272                            ; READ + ((MODIFY+VSRC)*Rmode)
;273      MEMORY=2              ;Generate OPVALID if
;274                            ; ASRC+((READ+MODIFY)*Aligned)
;275      FORCE=3               ;Generate an OPVALID unconditionally.
;276
;277
;278    ;
;279    ; Unpacker Control.  This specifies how ID register data is driven onto
;280    ; the OPBUS.
;281    ;
;282
;283    UNPACK CTL/=<32:31>,.DEFAULT=<UNPACK CTL/SIGN>
;284      NOP=0                 ; Do not unpack
;285      SIGN=1                ; Sign extend if -Asrc
;286      LITERAL=2             ; Unpack as short literal according to DRAM CTX and
;287                            ; TYPE fields
```

```
;288    .PAGE .
;289
;290    ;
;291    ; Memory control field.  This initiates memory references, sometimes
;292    ; contionally based on Dram data.
;293    ;
;294
;295    MCF/=<36:33>,.DEFAULT=<MCF/NOP>
;296      CONDITIONAL=0          ; READ.RCHK if DRAM MEM = READ
;297                             ; READ.WCHK if DRAM MEM = MODIFY
;298                             ; otherwise, NOP
;299      NOP=1                  ;No op-fetch port memory request.
;300      WRITE_V_NOPAGE=2,.VALIDITY=<.EQLL<UMISC/>,<UMISC/WRITE.VA.READY>1>
;301                             ;Initiate a WRITE virtual, no page check.  No
;302                             ; cross-page or alignment checks are done.
;303      WRITE_V_NOPAGE.2ND=3   ;Initiate a WRITE virtual, no page check.  No
;304                             ; cross-page or alignment checks are done.
;305      WRITE_V_WCHK=5,.VALIDITY=<.EQLL<UMISC/>,<UMISC/WRITE.VA.READY>1>
;306                             ;Initiate a WRITE virtual, checking for write access.
;307                             ; If no write access, a microtrap will be generated
;308                             ; for Ebox microcode.
;309      READ_V_WCHK=8          ;Initiate a READ virtual, checking for write access.
;310      READ_V_RCHK=0A
;311                             ;Initiate a READ virtual, checking for read access.
;312      READ_V_RCHK.2ND=0B
;313                             ;Initiate a READ virtual, checking for read access.
;314      READ_V_NOPAGE=0C
;315                             ;Initiate a READ virtual, no page check.
;316      READ_V_NOPAGE.2ND=0D
;317                             ;Initiate a READ virtual, no page check.
;318      IB.FILL.OP=0E          ;Make a memory request using VA as the virtual address,
;319                             ; and send the corresponding data to the IBUFFER, if
;320                             ; the OPPORT status in Mbox is no problem.  If problem
;321                             ; occurs, set IBFPORT problem bit.
;322      IB.FILL.IBF=0F         ;Make a memory request using VA as the virtual address,
;323                             ; and send the corresponding data to the IBUFFER, if
;324                             ; the IBUFFER port status in Mbox is no problem.  If
;325                             ; problem occurs, set IBFPORT problem bit.
```

```
;326     .PAGE
;327
;328     ;
;329     ; The UMISC field is for misc. encodings in the microword.
;330     ;
;331
;332     UMISC/=<40:37>,.DEFAULT=<UMISC/NOP>
;333       NOP=0                     ;No misc. signal generation
;334       WRITE.VA.READY=1,.VALIDITY=<.AND.ORE.EQLE<MCF/>,<MCF/WRITE_V_NOPAGE>],
;335                                    .EQLE<MCF/>,<MCF/WRITE_V_WCHK>]],
;336                                 .EQLE<CYCLE ID/>, <CYCLE ID/NORMAL>]]>
;337                               ;Stall until Ebox asserts OP.WRITE command.
;338       IB.FLUSH.LD.CPC=2       ;Used during flush operations to: a) VIBA_VA+4,
;339                               ; b) alter CPC/ESA/ISA valid bits, c) CPC_VA, d) clear
;340                               ; some signals in Ibox control logic, e) set CPC VALID.
;341       IB.FLUSH=3              ;Used during flush operations to: A) VIBA_VA+4,
;342                               ; b) alter ESA/ISA valid bits, and c) clear some
;343                               ; signals in Ibox control logic.
;344       CONDCOMP.BRANCH=4       ;A stall enable used during conditional and computed branches to
;345                               ; stall until the micromachine receives CC VALID from the Ebox.
;346       IB.FLUSH.COND=5         ;Used during conditional branch operations.  If BRANCH-
;347                               ; TRUE this initiates all the operations listed for
;348                               ; "IB.FLUSH.LD.CPC".  If NOT BRANCH_TRUE, "Op-port
;349                               ; cancel" is issued to the cache to abort the target
;350                               ; address fetch.  Used in second cycle of branch microflow.
;351       READ.IMD=6              ;Access an indirect address in IMD.  It will enable ISTALL
;352                               ; until OP MD RESP is detected.  When MD RESP is received,
;353                               ; this encoding causes the IMD latches in the IBUF MCAs to load the
;354                               ; IMD latches in the DBS slices.
;355       INH.SB.CHECK=7          ;Inhibits the assertion of a stall condition due to a
;356                               ; scoreboard hit.  Used to handle the reading of the
;357                               ; second longword of a 'quad & modify' type GPR operand.
;358       RAF=8                   ;Generate Reserved Addressing Mode Fault.
;359       DIAG.DONE=9             ;This causes a "diagnostic done"-handshake signal to
;360                               ; go to the Ebox branch enable logic.
;361       DIAG.RESET=0A           ;Reset some Ibox control logic.
;362       BMUX.CHK.CTX=0B         ;This encoding will cause the partiy check at the output
;363                               ; of the BMUX LATCH to be DRAM CTX dependent.  Used to
;364                               ; handle immediate data being passed through the BMUX.
;365       POPSTACK=0C             ;Pop the microaddress stack and use it as the next uPC.
;366       CLEAR.CPCVALID=0D       ;This clears the CPC VALID flag.  It is used during flush and macro-branch
;367                               ; microflows.
;368       COND.FA.OP.MEM.REQ=0E   ;Asserts FA OP MEM REQ IFF Dram Ref = -Asrc (i.e., if Dram Ref <> 0).
;369                               ; Since Dram Ref = 0 for Asrc and Vsrc operands, this is set FA OP MEM REQ IFF
;370                               ; Dram Ref = -(Asrc + Vsrc).  See the end of this file for more info on FA
;371                               ; OP MEM REQ.
;372       FA.OP.MEM.REQ=0F        ;Asserts FA OP MEM REQ unconditionally.
```

```
;373    .PAGE
;374
;375    ;
;376    ; Field to identify the current cycle
;377    ;
;378
;379    CYCLE ID/=<42:41>,.DEFAULT=<CYCLE ID/NORMAL>
;380      NORMAL=0              ;Non-Ifork cycle
;381      INDEXFORK=1           ;Index Ifork entry microword
;382      BOAFORK=2             ;BOA Ifork entry microword
;383      BEANTOWN=3            ;Vanilla flavored Ifork entry microword
;384
;385
;386    ;
;387    ; Used in microtrap routines to override ISTALL due to either the Ebox
;388    ; Ibox.Stop command or the DRAM SUSPEND bit.
;389    ;
;390
;391    UNSTALL/=<43:43>,.DEFAULT=<UNSTALL/NORMAL>
;392       NORMAL=0             ;Observe all stall signals in the NEXT microcycle.
;393       UNSTALL=1            ;Unstall if Istall due to the above 2 conditions.
;394
;395
;396    ;
;397    ; Used in DIAGNOSTIC MODE ONLY
;398    ;
;399
;400    IB HOLD/=<44:44>,.DEFAULT=<IB HOLD/NOHOLD>
;401      NOHOLD=0              ;Allow IBUFFER to do its thing
;402      HOLD=1                ;Prevent IBUFFER from shifting
;403
;404    DIAG UNSTALL/=<45:45>,.DEFAULT=<DIAG UNSTALL/NOP>
;405      NOP=0                 ;Allow normal stall effects
;406      SUPER=1               ;Override any existing Ibox stall condition
;407
;408
;409    ;
;410    ; The MARK bit: if set, CPU clocks stall as of the next T0ish.
;411    ;
;412
;413    MARK/=<46:46>,.DEFAULT=<MARK/OFF>
;414       OFF=0                ;Normal state
;415       ON=1                 ;Micro-breakpoint encoding.
```

```
;416    .PAGE
;417
;418    ;
;419    ; This field is for additional miscellaneous encodings in the Ibox uword.
;420    ;
;421
;422    UMISC2/=<49:47>,.DEFAULT=<UMISC2/NOP>
;423       NOP=0                    ;No signal generation
;424       INDEX.IMM=1              ;Assert "Index immediate mode" bit.  Also, this generates
;425                                ; the INHIBIT SET SB VALID signal (the same signal generated
;426                                ; by UMISC2/INHIBIT.SET.SB.VALID).
;427       ENA.OPBUS.PARITY=2,.Validity=<.ORE     .AndE.EqlE<Amux Sel/>, <Amux Sel/Zero>],
;428                                                  .EqlE<Bmux Sel/>, <Bmux Sel/Dmux.Ibf>]],
;429                                              .AndE.EqlE<Amux Sel/>, <Amux Sel/Zero>],
;430                                                  .EqlE<Bmux Sel/>, <Bmux Sel/Dmux.Imd>]],
;431                                              .AndE.EqlE<Bmux Sel/>, <Bmux Sel/Zero>],
;432                                                  .EqlE<Amux Sel/>, <Amux Sel/GPR>]]       ]>
;433                                ;Assert "ENA OP PAR CHK" signal to enable OPBUS parity
;434                                ; checking.  See the microcode restrictions at the end
;435                                ; of this file for more information.
;436       FORCE.SET.SB.VALID=4     ;Force the validation of a scoreboard entry iff one is
;437                                ; pending for the optimized destination of the current macroinstruction.
;438       INHIBIT.SET.SB.VALID=5,.Validity=<.AndE.EqlE<Reg Mode/>,<Reg Mode/NO>],
;439                                                .EqlE<Cycle Id/>,<Cycle Id/BOAFORK>]]>
;440                                ;Inhibit the validation of a scoreboard entry.  This can only be
;441                                ; This is intended to inhibit the too-early entry of an optimized
;442                                ; GPR# into the scoreboard.
;443       FORCE.SET.SB.ENA.OPBUS.PARITY=6,.Validity=<.ORE     .AndE.EqlE<Amux Sel/>, <Amux Sel/Zero>],
;444                                                  .EqlE<Bmux Sel/>, <Bmux Sel/Dmux.Ibf>]],
;445                                              .AndE.EqlE<Amux Sel/>, <Amux Sel/Zero>],
;446                                                  .EqlE<Bmux Sel/>, <Bmux Sel/Dmux.Imd>]],
;447                                              .AndE.EqlE<Bmux Sel/>, <Bmux Sel/Zero>],
;448                                                  .EqlE<Amux Sel/>, <Amux Sel/GPR>]]       ]>
;449                                ; This performs a FORCE.SET.SB.VALID and an ENA.OPBUS.PARITY.  It performs
;450                                ; everything a "2" and "4" encoding in the microfield do, together.
;451       UTRAP.CIP=7              ;Identifies a microword in a read*word or Read*Longword unaligned microtrap.  This is used
;452                                ; to preserve INDEX IFORK CYCLE through an unaligned microtrap.
;453
;454
;455
;456    ;
;457    ; Microword parity.  Currently, odd parity is used.  Even parity would be
;458    ; .NOTing the .PARITY qualifier
;459    ;
;460
;461    PAR0/=<31:0>                          ; The areas for parity field generation
;462    PAR1/=<49:32>
;463    UPAR/=<50:50>,.DEFAULT=<.PARITYE<PAR0/>,<PAR1/>]>
```

```
;464    .PAGE  "       Misc. Notes"
;465
;466    ;
;467    ;
;468    ; The VA LATCH loads from the datapath adder in every cycle (t3ish) unless the BMUX SEL/ microfield says
;469    ; to hold the VA.  Note that holding the VA is either forcing it to hold via BMUX SEL/ (which is the
;470    ; microword default for that field, so saying nothing gets you what you want), or saying VA_VA, which is
;471    ; slightly more obscure but included in the listing since this is a free country.
;472    ;
;473    ; Note that saying VA_VA also effectively freezes the contents of the OP VA latch whereas the Bmux field
;474    ; does not.  This affects memory references.
;475    ;
;476    ; Total number of used bits in the Ibox microword: 51 out of a total of 52 possible bits.
;477    ;
;478    ; FA OP MEM REQ is a signal asserted at SET FA FULL time if IFORK VALID * ( this operand will require at
;479    ; least 1 memory reference ).  This signal goes into a state machine on ICB.  The state machine keeps this
;480    ; asserted until the next Ebox FORK, which clears it (along with master reset or flush).  The Ebox latches
;481    ; the value of this ICB signal at FORK time and uses it to determine which port status code bits to look
;482    ; at if -OPVALID.  If set, it looks at the OP PORT status line from the Mbox which indicates an operand
;483    ; problem.  If clear, it looks at the IBUFFER port status line.  We therefore want to set this (via our 2
;484    ; UMISC/ encodings) for all INDIRECT memory specifiers and for all Direct memory specifiers which are not
;485    ; (Asrc + Vsrc).  Note that we must do this in the Ifork vector.
;486    ;
;487    ; The UMISC2/INDEX.IMM signal is generated during every cycle of (PC)+[Rn] mode in which we fetch the
;488    ; useless immediate-mode data from the Ibuffer.  Note that this is done before any other work for this
;489    ; specifier.  This goes to the Ebox and may only be set at FORK time; it may be cleared anytime.  Since
;490    ; Istall doesn't affect this bit, it will be generated every cycle until we finish processing emptying the
;491    ; Ibuffer.  This particular UMISC2/ encoding also generates the INHIBIT SET SB VALID signal.
;492    ;
;493    ; UMISC2/INHIBIT.SET.SB.VALID will inhibit the entering of a new SB entry into the scoreboard.  It must only
;494    ; be used during the IFORK CYCLE of every specifier that is a BDA.  It can be set after that cycle but this isn't
;495    ; necessary (It has to be set only once).  UMISC2/FORCE.SET.SB.VALID will force the validation of a new SB entry
;496    ; sitting at the input to the scoreboard iff OPT SAV (the saved version of OPTIMIZED) is TRUE.
;497    ;
```

```
;498    .PAGE   "       Ibox Microcode Restrictions"
;499
;500    ;
;501    ; Intra-word restrictions can be handled by MICRO2 validity checks.  This list is for inter-word restrictions which
;502    ; must be remembered by the lowly microcoder.
;503    ;
;504    ; This list will change with time as we discover new things about the Ibox design.
;505    ;
;506    ; (1)   Reading Memory for indirect addresses must be done in the following
;507    ;       sequence:
;508    ;
;509    ;       (A) Initiate the READ with the address being in VA by the END of
;510    ;           this cycle.
;511    ;
;512    ;       (B) Assert UMISC/READ.IMD, via the STALL UNTIL MD RESP macro.
;513    ;
;514    ;       (C) Assert BMUX SEL/BMUX.IMD via some macro.
;515    ;
;516    ; (2)   The Bmux Sel/ field encoding HOLDVA, which is not a Bmux select
;517    ;       function but rather inhibits loading of the VA LATCH, does not
;518    ;       inhibit loading of the OP VA LATCH.  Hence if you wish to correctly
;519    ;       generate a virtual address to reference memory you must load VA
;520    ;       with itself, i.e. use the VA_VA macro.
;521    ;
;522    ; (3)   The following microwords MUST clear CPC VALID via the CLEAR.CPCVALID encoding
;523    ;       in the uMisc/ microfield:
;524    ;
;525    ;       (a)     The first microword (on the Ifork) in the Unconditional Branch microflow.
;526    ;       (b)     The first microword (on the Ifork) in the Conditional Branch microflow.
;527    ;       (c)     The first microword (on the Ifork) in the Computed Branch microflow.
;528    ;       (d)     The first microword (in the microtrap vector) in the Flush.Wbus.&.Load.CPC
;529    ;               microflow.
;530    ;
;531    ;       The purpose of this encoding is to prevent the Ebox from storing a restart address
;532    ;       (in the event of a trap) which is incorrect, due to the fact that the CPC is not
;533    ;       yet valid.  CPC may not be valid if the Mbox does not issue PA ACK to the first
;534    ;       memory request of a flush.
;535    ;
;536    ; (4)   The default encoding for the CTX CTL/ microfield must be either CTX CTL/DRAM (=0) or
;537    ;       CTX CTL/INDIRECT (=1).  Reason: the carry-in into the datapath adder is CTX CTL<2> along
;538    ;       with RLOG context decode, and this occurs even though the Bmux is not selecting context.
;539    ;       Hence, although the adder is not sourcing the context input, a carry in (used in subtraction)
;540    ;       would occur.  Hence, the ctx ctl field must default to DRAM or INDIRECT, which does not generate
;541    ;       this carry-in signal.
;542    ;
;543    ; (5)   All Ifork vector entries which process indirect memory specifiers must force the assertion of
;544    ;       FA OP MEM REQ via the macro "FORCE FA OP MEM REQ".  This must be done in the Ifork.
;545    ;
;546    ; (6)   All Ifork vector entries which process non-indirect memory specifiers must conditionally
;547    ;       assert FA OP MEM REQ via the macro "CONDITIONAL FA OP MEM REQ".  This must be done in the
;548    ;       Ifork.  This forces FA OP MEM REQ IFF Dram Ref = -Asrc (which translates into -(Asrc + Vsrc)).
;549    ;
```

```
;550    .Page
;551
;552    ;
;553    ; (7)   For the (PC)+[Rn] specifier, every microword which fetches a byte, word, or longword of useless Ibuffer
;554    ;       data must assert the UMISC2/IND%X.IMM signal.  This tells the Ebox to look at the Ibuffer port status
;555    ;       lines from the Mbox even though FA OP MEM REQ may be set.
;556    ;
;557    ; (8)   We must not assert OPVALID if the operand specifier is a Reserved Addressing Mode fault (RAF) condition.
;558    ;
;559    ; (9)   When an RAF is discovered, the microcode must jump-dot setting RAF until Ebox brings up Ibox.error, which in
;560    ;       turn causes us to Istall.
;561    ;
;562    ; (A)   The UMISC2/ encoding "ENA.OPBUS.PARITY" must be asserted in microwords which do the following:
;563    ;
;564    ;       (
;565    ;       ( (AMUX SEL/ = ZERO) .AND. ( (BMUX SEL/ = DMUX.IBF) .OR. (BMUX SEL/ = DMUX.IMD) ) )
;566    ;                                                  .OR.
;567    ;                       ( (BMUX SEL/ = ZERO) .AND. (AMUX SEL/ = GPR) )
;568    ;                                                                                  )
;569    ;                                                  .AND.
;570    ;       ( We might/are be setting OPVALID for data avaiable in the ID register (not a memory operand). )
;571    ;
;572    ;       The problem is that data loaded into ID doesn't have good parity unless it is created from the above sources.
;573    ;       The hardware takes the ENA.OPBUS.PARITY encoding and qualifies it with ENA LOAD ID to determine whether to
;574    ;       enable or disable opbus parity checking (the signal name is ENA OP PAR CHK).
;575    ;
;576    ;       This signal, via the ENABLE OPBUS PARITY CHECKING macro, must be asserted in every microword in which we
;577    ;       could be loading VA (and therefore ID) with the operand, and iff it is from one of the above sources.
;578    ;
;579    ;       I considered making this a .DEFAULT thing in MICRO2 so that it would be done automatically, but felt that
;580    ;       there might be situations in Ibox microdiagnostics which would not want the default.
;581    ;
;582    ; (B)   All Ifork vector entries must have a CYCLE ID/ encoding which is appropriate
;583    ;       even if the operand is an RAF.  This is because CYCLE ID/ injects into alot of
;584    ;       logic, notable ISA loading & ISA valid logic.  For example, the Ifork entry for
;585    ;       [Rn][Rn] must have CYCLE ID/BOAFORK.
;586    ;
;587    ; (C)   UMISC/INH.SB.CHECK must be asserted in the following microwords:
;588    ;
;589    ;       (a)     The second microwords of (Rn)+, @(Rn)+, [Rn](Rn)+ (the cycle after the BOAFORK), and [Rn]@(Rn)+ (the cycle
;590    ;               after the BOAFORK)microflows.  Reason: If we are supposed to take a SB HIT STALL, we would have done so
;591    ;               during the IFORK CYCLE microword since that microword sources the autoinc. (or BOA autoinc.) GPR.  If we
;592    ;               are executing at the uword after the IFORK CYCLE, we clearly didn't take the SB HIT STALL.  But if the
;593    ;               current macroinstruction is being optimized, and the destination GPR# = the autoinc. GPR#, then the dest.
;594    ;               GPR is entered into the scoreboard and we could take an unnecessary, and lethal, SB HIT STALL.  Using the
;595    ;               UMISC/ encoding to inhibit SB HIT STALLS prevents this from happening.
;596    ;
;597    ;       (b)     When reading the second longword of a (Modify * Quad * Rn) operand.  The reason for this is slightly
;598    ;               obscure, but essentially the SB logic gets confused by a Quad Scoreboarded Modify Rn operand.
;599    ;
```

```
;600    .Page
;601
;602    ;
;603    ; (D)   UMISC2/INHIBIT.SET.SB.VALID must be set in the Ifork Cycle microwords of all BOA operand specifiers (except for
;604    ;       [Rn](PC)+ specifiers.  For this specifier, the Index Immediate drain encoding asserts the INHIBIT SET SB VALID
;605    ;       signal).  This inhibits the premature entering of the optimized destination of the current macroinstruction, if
;606    ;       one is present, into the scoreboard and thereby setting the stage for a spurious SB hit on the source index GPR.
;607    ;
;608    ; (E)   UMISC2/FORCE.SET.SB.VALID must be set in the microword which finally loads VA with the operand target address
;609    ;       IFF the microflow in question came from a BOA Ifork Cycle microword which issued a UMISC2/INDEX.IMM or a
;610    ;       UMISC2/INHIBIT.SET.SB.VALID.
;611    ;
;612    ; (F)   UMISC2/FORCE.SET.SB.ENA.OPBUS.PARITY must be set in the 2nd microword (i.e., the one immediately following the
;613    ;       Ifork cycle) of the Read * Rn * (Quad*Octa) operand specifier flow.  This is needed to fix the MOVQ R0, R1
;614    ;       scoreboard bug, which was a flaw when the R1 specifier was optimized.
;615    ;
;616    ;       The hardware will inhibit the entering of R1&R2 (via the Quad context) into the scoreboard during the Ifork
;617    ;       cycle.  The microword which processes the 2nd GPR will assert this encoding, which causes OPBUS parity checking
;618    ;       to be performed as well as a conditional loading of the scoreboard iff the instruction was optimized.  This
;619    ;       should not hurt Octa operands since they are never optimized.
;620    ;
;621    ; (10)  UMISC2/UIRAP.CIP must be set in the microwords which process a Read*Word or Read*Longword unaligned operand.
;622    ;       Reason: If the operand prior to an indexed-BOA combination specifier is one of these 2 types of unaligned
;623    ;       operands, the hw will forget that we microtrapped out of an index ifork cycle.  When the microtrap microcode
;624    ;       does return to the Ifork, we will go to CB, but the Ifork microaddress for the BOA may have been cooked up
;625    ;       as though the specifier was not a BOA!  This signal forces the Index ifork cycle state machine to
;626    ;       remember the INDEX IFORK CYCLE encoding and therefore always cook up the BOA Ifork address properly.
;627    ;
```

```
;628    .PAGE   "IBOXMACRO.MIC"
;629    .TOC    "Revision 8.3"
;630    .TOC    "21 March 1984"
;631
;632    .TOC    "       Revision History"
;633
;634    ;Rev    Date            Explanation
;635    ;---    ----            -----------
;636    ;  8    30-Nov-83       Added: INHIBIT SET SCOREBOARD VALID, FORCE SET SCOREBOARD VALID, and
;637    ;                      STRING MODE macros.
;638    ;                      Added: Explicit WORD context macro for the new utrap/SII
;639    ;                      timing fix.
;640    ;                      Added: FORCE SET SB VALID & ENA OPBUS PARITY CHECKING macro.
;641    ;                      Removed: STRING MODE macro.  Added: UTRAP CIP macro.
;642    ;  7    7-Jan-83        Deleted "Will Drain..." macros.
;643    ;                      Added FORCE FA OP MEM REQ and CONDITIONAL FA OP MEM REQ
;644    ;                      macros.
;645    ;                      Added "INDEX IMMEDIATE DRAIN" macro to facilitate the (PC)+[Rn]
;646    ;                      specifier processing.
;647    ;                      Renamed macro for OPVALID/GPR.IMM.
;648    ;                      Deleted IFORK IF COMP BR FALSE macro, renamed WAIT FOR IRD
;649    ;                      into WAIT FOR CC VALID.
;650    ;                      Added ENABLE OPBUS PARITY CHECKING macro.
;651    ;                      Add: Since GPR SEL/<2> does not force an INHIBIT SB STALL
;652    ;                      in the hardware, I added UMISC/INH.SB.CHECK to all macros
;653    ;                      which use GPR SEL/RLOG_UNWIND or GPR SEL/EXPLICIT.
;654    ;                      Remove: above MACRO construction.  Hardware will fix the
;655    ;                      problem.
;656    ;  6    7-Sep-82        Added Clear CPC Valid macro.  Added "Will Drain..."
;657    ;                      macros.
;658    ;  5    11-Mar-82       Updated.  Added VA_VA.  Added explicit GPR macros.
;659    ;                      Added macros to source CPC for string continue
;660    ;                      flushes.
;661    ;  4    22-Dec-81       Updated to reflect new Ibox design.
;662    ;  3    26-May-81       Updated to reflect new IBOXDEF.MIC file.  Added
;663    ;                      flushes and Conditional Branch flows.
;664    ;  2    10-November-80  Removed unused macro.  Changed macros to Ebox format
;665    ;                      (No change in functionality, just difference in
;666    ;                      appearence).
;667    ;  1    31-October-80   Macro file completed.  First rev. based on reasonably
;668    ;                      stable hardware definition.
;669    ;  0    13-October-80   Start of History.
```

```
;670    .PAGE   "       Ibox Microcode Macros - Diagnostic"
;671
;672    ;
;673    ; Explicit GPR moves and shifts
;674    ;
;675
;676    MOV REG VA              "AMUX SEL/GPR, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;677                             GPRREG/@1, CTX CTL/<GPRREG32/>,
;678                             UNPACK CTL/<GPRREG10/>"
;679    MOV REG VA REQ WBUS     "AMUX SEL/GPR, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;680                             GPRREG/@1, CTX CTL/<GPRREG32/>, WBUSREQ/ON,
;681                             UNPACK CTL/<GPRREG10/>"
;682    MOV 2(REG) VA           "AMUX SEL/GPR.LS1, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;683                             GPRREG/@1, CTX CTL/<GPRREG32/>,
;684                             UNPACK CTL/<GPRREG10/>"
;685    MOV 4(REG) VA           "AMUX SEL/GPR.LS2, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;686                             GPRREG/@1, CTX CTL/<GPRREG32/>,
;687                             UNPACK CTL/<GPRREG10/>"
;688    MOV 2(REG) VA REQ WBUS  "AMUX SEL/GPR.LS1, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;689                             GPRREG/@1, CTX CTL/<GPRREG32/>, WBUSREQ/ON,
;690                             UNPACK CTL/<GPRREG10/>"
;691    MOV 4(REG) VA REQ WBUS  "AMUX SEL/GPR.LS2, BMUX SEL/ZERO, GPR SEL/EXPLICIT,
;692                             GPRREG/@1, CTX CTL/<GPRREG32/>, WBUSREQ/ON,
;693                             UNPACK CTL/<GPRREG10/>"
```

```
;694     .PAGE   "       Ibox Microcode Macros - System"
;695
;696     ;
;697     ; General micromachine control - Gotos, NOP, Ifork control, branches
;698     ;
;699
;700     BRANCH ON (Q+0) AND ACCESS      "UBEN/Q+0.ACCESS"
;701     BRANCH ON ACCESS                "UBEN/ACCESS"
;702     BRANCH ON RLOG FIRST BIT        "UBEN/RLOG_FB"
;703     BRANCH ON CONTEXT               "UBEN/CONTEXT"
;704     BRANCH ON QUAD OR OCTA          "UBEN/QUAD.OCTA"
;705     GOTO []                         "NEXT/@1"
;706     IFORK                           "IFORK CTL/IFORK"
;707     IFORK IF A+R*BWL                "IFORK CTL/MEMORY"
;708     IFORK IF QUAD                   "IFORK CTL/QUAD"
;709     IFORK IF R*BWL                  "IFORK CTL/IMMEDIATE"
;710     IFORK IF V+W+RM*BWL             "IFORK CTL/REGISTER"
;711     RETURN                          "UMISC/POPSTACK"
;712
;713
;714     ;
;715     ; Memory functions - All operations are VIRTUAL
;716     ;
;717                                     "MCF/CONDITIONAL"
;718     COND MEM REQ                    "MCF/IB.FILL.IBF"
;719     IBUF REQ FROM IBF-PORT          "MCF/IB.FILL.OP"
;720     IBUF REQ FROM OP-PORT           "MCF/READ_V_RCHK"
;721     READ                            "MCF/READ_V_RCHK.2ND"
;722     READ.2ND                        "MCF/READ_V_WCHK"
;723     READ.WCHK                       "MCF/READ_V_NOPAGE"
;724     READ.NOPAGE                     "MCF/READ_V_NOPAGE.2ND"
;725     READ.NOPAGE.2ND                 "UMISC/READ.IMD"
;726     STALL UNTIL IMD MDRESP          "MCF/WRITE_V_WCHK"
;727     WRITE                           "MCF/WRITE_V_NOPAGE"
;728     WRITE.NOPAGE                    "MCF/WRITE_V_NOPAGE.2ND"
;729     WRITE.NOPAGE.2ND
;730
;731
;732     ;
;733     ; Signal generation to the Ebox
;734     ;
;735
;736     CONDITIONAL FA OP MEM REQ       "UMISC/COND.FA.OP.MEM.REQ"
;737     ENABLE OPBUS PARITY CHECKING    "UMISC2/ENA.OPBUS.PARITY"
;738     FORCE FA OP MEM REQ             "UMISC/FA.OP.MEM.REQ"
;739     INDEX IMMEDIATE DRAIN           "UMISC2/INDEX.IMM"
;740     OPVALID                         "OPVALID/FORCE"
;741     OPVALID IF R+(MV*RN)            "OPVALID/GPR.IMM"
;742     OPVALID IF A+RM*AL              "OPVALID/MEMORY"
;743     RESERVED ADDRESSING FAULT       "UMISC/RAF"
;744     WRITE VA READY                  "UMISC/WRITE.VA.READY"
```

```
;745    .PAGE
;746
;747    ;
;748    ; General macros
;749    ;
;750
;751    IGPR_IGPR+CTX                   "AMUX SEL/GPR, BMUX SEL/ADDERCTX,
;752                                     WBUSREQ/ON"
;753    IGPR_IGPR+4                     "AMUX SEL/GPR, BMUX SEL/ADDERCTX,
;754                                     WBUSREQ/ON, CTX CTL/INDIRECT"
;755    VA_0                            "AMUX SEL/ZERO, BMUX SEL/ZERO"
;756    VA_CPC                          "AMUX SEL/ZERO, BMUX SEL/CPC"
;757    VA_AMUXPC-CTX                   "AMUX SEL/AMUXPC, BMUX SEL/ADDERCTX,
;758                                     CTX CTL/AUTODEC"
;759    VA_AMUXPC+IBUFFER DATA          "AMUX SEL/AMUXPC, BMUX SEL/DMUX.IBF"
;760    VA_GPR(PREVADD+1)               "AMUX SEL/GPR, BMUX SEL/ZERO, GPR SEL/PREVADD+1"
;761    VA_IBUFFER DATA                 "AMUX SEL/ZERO, BMUX SEL/DMUX.IBF"
;762    VA_IGPR                         "AMUX SEL/GPR, BMUX SEL/ZERO, GPR SEL/ISTREAM"
;763    VA_IGPR(PREVADD)-4              "AMUX SEL/GPR, BMUX SEL/ADDERCTX,
;764                                     GPR SEL/PREVADD, CTX CTL/AUTOINC.DEF"
;765    VA_IGPR(PREVADD)-CTX            "AMUX SEL/GPR, BMUX SEL/ADDERCTX,
;766                                     GPR SEL/PREVADD, CTX CTL/AUTODEC"
;767    VA_IGPR+IBUFFER DATA            "AMUX SEL/GPR, BMUX SEL/DMUX.IBF"
;768    VA_IMD                          "AMUX SEL/ZERO, BMUX SEL/DMUX.IMD"
;769    VA_VA                           "AMUX SEL/ZERO, BMUX SEL/VA"
;770    VA_VA+INDEX GPR                 "AMUX SEL/GPR, BMUX SEL/VA, GPR SEL/INDEX"
;771    VA_VA+2(INDEX GPR)              "AMUX SEL/GPR.LS1, BMUX SEL/VA, GPR SEL/INDEX"
;772    VA_VA+4                         "AMUX SEL/FOUR, BMUX SEL/VA"
;773    VA_VA+4(INDEX GPR)              "AMUX SEL/GPR.LS2, BMUX SEL/VA, GPR SEL/INDEX"
;774    VA_VA-4                         "AMUX SEL/NEGFOUR, BMUX SEL/VA"
;775    VA_IGPR_IGPR-CTX                "AMUX SEL/GPR, BMUX SEL/ADDERCTX,
;776                                     CTX CTL/AUTODEC, WBUSREQ/ON"
;777    VA_WBUS                         "AMUX SEL/WBUS, BMUX SEL/ZERO"
```

```
;778    .PAGE
;779
;780    ;
;781    ; Context definition macros
;782    ;
;783
;784    CONTEXT IS -4 4 X               "CTX CTL/AUTOINC.DEF"
;785    CONTEXT IS 4 4 4                "CTX CTL/INDIRECT"
;786    CONTEXT IS X 2 X                "CTX CTL/WORD"
;787
;788
;789    ;
;790    ; Cycle identification
;791    ;
;792
;793    INDEX CYCLE                     "CYCLE ID/INDEXFORK"
;794    BOA CYCLE                       "CYCLE ID/BOAFORK"
;795    VANILLA CYCLE                   "CYCLE ID/BEANTOWN"
;796    VANILLA RMODE CYCLE             "CYCLE ID/BEANTOWN, REG MODE/YES"
;797
;798
;799    ;
;800    ; OPBUS Unpacking
;801    ;
;802
;803    NO OPBUS UNPACK                 "UNPACK/NOP"
;804    LITERAL UNPACK OPBUS            "UNPACK CTL/LITERAL"
;805
;806
;807    ;
;808    ; Flush & Conditional Branch Macros
;809    ;
;810
;811    FLUSH & LOAD CPC                "UMISC/IB.FLUSH.LD.CPC"
;812    FLUSH                           "UMISC/IB.FLUSH"
;813    WAIT FOR CC VALID               "UMISC/CONDCOMP.BRANCH"
;814    COND IBUF FLUSH & LOAD CPC      "UMISC/IB.FLUSH.COND"
;815
;816
;817    ;
;818    ; Misc.
;819    ;
;820
;821    BMUX CHECK WITH CONTEXT                             "UMISC/BMUX.CHK.CTX"
;822    CLEAR CPC VALID                                     "UMISC/CLEAR.CPCVALID"
;823    ENABLE TRAP IF UNAL IND                             "UTRAP CTL/INDIRECT"
;824    ENABLE TRAP IF UNAL OP                              "UTRAP CTL/OPERAND"
;825    FORCE SET SCOREBOARD VALID                          "UMISC2/FORCE.SET.SB.VALID"
;826    FORCE SET SB VALID & ENA OPBUS PARITY CHECKING      "UMISC2/FORCE.SET.SB.ENA.OPBUS.PARITY"
;827    HOLD IBUFFER                                        "IB HOLD/HOLD"
;828    INHIBIT SET SCOREBOARD VALID                        "UMISC2/INHIBIT.SET.SB.VALID"
;829    INHIBIT STALLS                                      "UNSTALL/UNSTALL"
;830    MARK                                                "MARK/ON"
;831    NO SCOREBOARD CHECK                                 "UMISC/INH.SB.CHECK"
;832    UNWIND RLOG ENTRY                                   "AMUX SEL/GPR, BMUX SEL/ADDERCTX, CTX CTL/UNWIND,
```

```
;833                                          GPP SEL/RLOG_UNWIND, UNSTALL/UNSTALL"
;834    UTRAP CIP                            "UMISC2/UTRAP.CIP"
```

```
;835    .PAGE   "IBOXCODE.MIC"
;836    .TOC    "Revision 13"
;837    .TOC    "22 March 1984"
;838
;839    .TOC    "      Revision History"
;840
;841    ;Rev    Date            Explanation
;842    ;---    ----            -----------
;843    ;
;844    ; 13    22-Mar-84       Removed: The String mode jump-dot loop.  It looks like hw will never take advantage of it.
;845    ;                      Added: UTRAP CIP macro to the Unaligned * Read * (W+L) microwords.
;846    ; 12    21-Nov-83       Added: String Mode Jump-dot loop after an Ebox Flush and hold CPC
;847    ;                      utrap.  This replaces a hw state machine on ICA which remembered that
;848    ;                      the Ibox is in string mode.
;849    ;                      Added: INHIBIT SET SCOREBOARD VALID and FORCE SET SCOREBOARD VALID macros
;850    ;                      to the BOA IFORK CYCLE uwords of all indexed specifiers except for
;851    ;                      [Rn](PC)+, and to the uword which finally loads VA with the target address
;852    ;                      for all indexed specifiers.
;853    ;                      Added: New utrap scheme for Unal*Read*(W+L) operands which
;854    ;                      is part of the SII timing fix.
;855    ;                      Added: The microwords for all 3 types of branch instructions which issue
;856    ;                      the "F" request to the Mbox now issue the UNSTALL/UNSTALL (INHIBIT STALLS
;857    ;                      macro) bit.  This allows CPC to be set valid when the PA ACK from the "E"
;858    ;                      request comes back from the Mbox even if the Ebox has said STOP IBOX
;859    ;                      REFERENCES.
;860    ;                      Clarified some documentation.
;861    ;                      Added: FORCE SET SB VALID & ENA OPBUS PARITY CHECKING to the 2nd microword of
;862    ;                      Read * Rn * (Q+Q) microflow.
;863    ; 11    14-Oct-83       Fixed: RAF index specifiers had an Cycle ID = Vanilla.  Since cycle ID =
;864    ;                      INDEX causes ISA to get loaded, this had the effect of loading ISA twice
;865    ;                      for one specifier during an RAF.  As a result, Tryggve's exception
;866    ;                      microcode pushed the wrong PC on the stack if he read ISA (which would be
;867    ;                      the case if the first operand of an instruction was for example
;868    ;                      S^#literal[Rn]).  Now, RAF index classes of operands have cycle ID = BOA,
;869    ;                      which allows DRAM and IBUF PE latch loading and doesn't clobber ISA.
;870    ;                      Corrected Misc. documentation typos.
;871    ;                      Fixed: The second uwords of (Rn)+, @(Rn)+, [Rn](Rn)+, and [Rn]@(Rn)+ will
;872    ;                      now assert INHIBIT.SB.CHECK.  This fixes the problem of an optimized
;873    ;                      instruction with the destination GPR = to the autoinc. GPR, which is source
;874    ;                      in the cycle following the IFORK CYCLE microword.  Since that GPR is
;875    ;                      sourced in the IFORK CYCLE microword, if we should stall then we would have
;876    ;                      at the IFORK CYCLE.
;877    ; 10    17-Jun-83       Fixed an unbelievably gross and disgusting bug: All Quad
;878    ;                      and Octa writes (for both Write and Modify operands) were
;879    ;                      doing write.wchk writes for the 2nd - nth longwords, for
;880    ;                      both unaligned and aligned flows.  This caused spurious
;881    ;                      page boundry check microtraps if a Q + O datum was
;882    ;                      sufficiently close to the end of a page.
;883    ;                      Removed "HOLD IBUFFER" from the Master Reset microwords.
;884    ;                      This prevents loading of the double buffer, which in
;885    ;                      turn prevents power-up bad parity from being cleared.
;886    ;                      [ This might not be a problem, but could be. ]
;887    ;  9    14-Oct-82       Added microword at location BF to handle returning from
;888    ;                      a Read*WL unaligned second reference to an Ifork microword.
;889    ;                      Fixed: Octa unaligned Modify flow was decrementing the VA
```

```
;890    ;                       by 4 once too often.
;891    ;                       Deleted: "Will Drain Dmux if C]" macros from the (PC)+ flows.
;892    ;                       Added "FA OP MEM REQ" macros to assert FA OP MEM REQ to all
;893    ;                       appropriate Ifork entry microwords.
;894    ;                       Added: the INDEX IMMEDIATE DRAIN macro to the uwords in the (PC)+[Rn]
;895    ;                       flow which drain the useless Ibuffer data from the Ibuf.
;896    ;                       Changed: OPVALID generation for (PC)+ and S^#literal.
;897    ;                       Changed: we now continuously set RAF until Istall comes up.
;898    ;                       Changed over to new conditional/computed branch scheme.
;899    ;                       Added OP8US parity enable to appropriate microwords.
;900    ;  8    7-Sep-82        Added: "Clear CPC Valid" to the first uword of: (a)
;901    ;                       on Ifork, Unconditional Branches, Conditional Branches,
;902    ;                       and Computed Branches; (b) in the microtrap vector,
;903    ;                       the Flush from Wbus & load CPC flow.
;904    ;                       Added: "Will Drain Dmux if C]" macros to the (PC)+
;905    ;                       flows.
;906    ;                       Fixed: s^#literal was not doing an Ifork if read*bwl
;907    ;                       on the Ifork word.
;908    ;                       Added CYCLE ID/ encodings to all the R.A.F. entries in
;909    ;                       the Ifork vector.  Reason: the latching of DRAM and Ibuf
;910    ;                       parity errors is contingent upon the assertion of Ifork
;911    ;                       Cycle, which is generated from the CYCLE ID/ encodings.
;912    ;  7    10-May-82       Fixed: context on Istream request during flushes and
;913    ;                       conditional branches.
;914    ;                       Fixed: was not ensuring OP VA LATCH had correct data.
;915    ;                       Merged computed and unconditional branch flows,
;916    ;                       saved 1 microword.
;917    ;                       Changed: microcode to source different CPC during
;918    ;                       String Flushes.
;919    ;  6    11-Mar-82       Rip out CM.  Minor changes to existing design.
;920    ;  5    22-Dec-81       Yet another new Ibox design.
;921    ;  4    26-May-81       Massive update to mirror current hardware design.
;922    ;  3    5-November-80   Changed macros.  Minor documentation changes.
;923    ;  2    31-October-80   First version based on stable hardware definition.
;924    ;                       All microflows coded.
;925    ;  1    5-October-80    More flows.
;926    ;  0    19-September-80 Start of History.  First simple flows.
```

```
;927     .PAGE   "       General Description"
;928
;929     ;************************************************************************
;930     ;
;931     ; Each IFORK signal signifies a completed evaluation of an operand specifier.  If that specifier was not an
;932     ; Index-mode specifier, the IFORK also signifies a completed servicing of the operand.  There is a 1-to-1
;933     ; correspondence between IFORKing and a modification of the DRAM Execution Point Counter, with the exception
;934     ; of [Rn] mode where EPC is not modified.
;935     ;
;936     ; This file contains the entirety of Ibox microcode.  Two other files are necessary: IBOXDEF.MIC and
;937     ; IBOXMACRO.MIC.  They respectively contain the hardware definition and microcode macros for this microcode.
;938     ;
;939     ; The microcode structure is broken into individual routines starting in a microword vector called "Ifork".
;940     ; Each of these routines handles a particular operand specifier:
;941     ;
;942     ;
;943     ;       Ifork ----v----------v--------------v--- . . . -----v
;944     ;             ^   v          v              v                v
;945     ;             ^   v          v              v                v
;946     ;             ^  ----------  ----------   ----------      ----------
;947     ;             ^  | spc.   |  | spc.   |   | spc.   |      | spc.   |
;948     ;             ^  | routine|  | routine |   | routine |      | routine|
;949     ;             ^  ----------  ----------   ----------      ----------
;950     ;             ^     v           v            v                v
;951     ;             ^     v           v            v                v
;952     ;             ^     v           v            v                v
;953     ;             '<<<<<<<<<'<<<<<<<<<<<'<<<<<<<<<<<<'<<<<<<<<<<<<<<<'
;954     ;
;955     ;
;956     ;
;957     ; The Ibox micromachine supports a microtrap mechanism to divert the microflow to a trap service routine.
;958     ; There are 2 types of microtraps: those that are reactions to Ebox commands and those that are reactions to
;959     ; unaligned memory addresses during operand specifier handling.  The Ebox-command microtraps do not return to
;960     ; the point of interruption.  The Unalignment microtraps either do or do not return.
;961     ;
;962     ;************************************************************************
                        ;963    .BIN
                        ;964
                        ;965
                        ;966    .TOC    "       Region Assignments"
                        ;967
                        ;968    ;************************************************************************
                        ;969    ; These mnemonics define the regions of memory for forks, utrap vectors,
                        ;970    ; and anything else that must be .REGIONed.
                        ;971    ;************************************************************************
                        ;972
                        ;973    .SET/IFORKLOW=0C0               ; Ifork Vector
                        ;974    .SET/IFORKHIGH=0FF
                        ;975
                        ;976    .SET/MAINLOW=000                ; Utrap vector (addresses are forced) and the
                        ;977    .SET/MAINHIGH=0BF               ; bulk of Ibox ucode
```

```
                        ;978    .PAGE   "       The Ifork Vector"
                        ;979
                        ;980    ;*****************************************************************
                        ;981    ;
                        ;982    ; Flow(s):
                        ;983    ;       This vector contains the first microwords of every operand specifier routine.
                        ;984    ;
                        ;985    ;       Normally, bit <5> is asserted if this is a BOA cycle and bit <4> is asserted if
                        ;986    ;       this operand specifier uses the PC.  However, special encodings exist for the
                        ;987    ;       various types of branch instructions.
                        ;988    ;
                        ;989    ;       Unused locations in the Ifork are filled with Jump-dots.
                        ;990    ;
                        ;991    ;*****************************************************************
                        ;992
                        ;993    .REGION/IFORKLOW,IFORKHIGH
                        ;994
                        ;995
                        ;996    0CF:    ;-------------------------------; Conditional Branch
                        ;997            CLEAR CPC VALID, VANILLA CYCLE, ;
                        ;998            VA_AMUXPC+IBUFFER DATA,         ; VA_target address, ***do not stall
I 0CF, 0,07A2,8010,802E ;999            GOTO EIB.B.CONDCOMP.BRJ         ; on CC VALID here***
                        ;1000
                        ;1001   0FF:    ;-------------------------------; Unconditional Branch
                        ;1002           CLEAR CPC VALID, VANILLA CYCLE, ;
                        ;1003           CONTEXT IS 4 4 4,               ;
                        ;1004           VA_AMUXPC+IBUFFER DATA,         ; VA_target address, make first ref.
                        ;1005           IBUF REQ FROM OP-PORT,          ;
I 0EF, 0,07BC,8030,8038 ;1006           GOTO EIB.B.UNCOND.BRJ           ;
                        ;1007
                        ;1008   0FF:    ;-------------------------------; Computed Branches
                        ;1009           VANILLA CYCLE,                  ;
                        ;1010           VA_AMUXPC+IBUFFER DATA,         ; VA_target address
                        ;1011           CLEAR CPC VALID,                ;
I 0FF, 0,07A2,8010,802E ;1012           GOTO EIB.B.CONDCOMP.BRJ         ;
                        ;1013
                        ;1014   0CA:    ;-------------------------------; Rn
                        ;1015           VA_IGPR, VANILLA RMODE CYCLE,   ; Ifork if this can be done in 1
                        ;1016           OPVALID IF R+(MV*RN),           ; cycle, continue if Asrc or
                        ;1017           IFORK IF V+W+RM*BWL,            ; (read*modify)*QO
                        ;1018           ENABLE OPBUS PARITY CHECKING,   ;
                        ;1019           BRANCH ON (Q+Q) AND ACCESS,     ;
I 0CA, 1,0602,8002,0910 ;1020           GOTO EIB.NM.RNJ                 ;
                        ;1021
                        ;1022   0DA:    ;-------------------------------; PC (UNPREDICTABLE)
                        ;1023           RESERVED ADDRESSING FAULT,      ;
                        ;1024           VANILLA CYCLE,                  ; To allow Ibuf and Dram PE latching
I 0DA, 0,0702,801C,0010 ;1025           GOTO EIB.B.JDOT.RAFJ            ;
```

```
                        ;1026  .PAGE
                        ;1027
                        ;1028  0C9:    ;--------------------------------; (Rn)
                        ;1029          VA_IGPR, COND MEM REQ,           ;
                        ;1030          ENABLE OPBUS PARITY CHECKING,    ;
                        ;1031          OPVALID IF A+RM*AL,              ;
                        ;1032          CONDITIONAL FA OP MEM REQ,       ;
                        ;1033          ENABLE TRAP IF UNAL OP,          ; Trap next cycle if unaligned add.,
                        ;1034          VANILLA CYCLE,                   ; if this can't be done in 1 cycle
                        ;1035          BRANCH ON (Q+0) AND ACCESS,      ; goto common fetch routine
                        ;1036          IFORK IF A+R*BWL,                ;
I 0C9, 1,07C0,C002,4540 ;1037          GOTO [IB.NM.FET]                 ;
                        ;1038
                        ;1039  0D9:    ;--------------------------------; (PC) [UNPREDICTABLE]
                        ;1040          RESERVED ADDRESSING FAULT,       ;
                        ;1041          VANILLA CYCLE,                   ; To allow Ibuf and Dram PE latching
I 0D9, 0,0702,801C,0010 ;1042          GOTO [IB.B.JDOT.RAF]             ;
                        ;1043
                        ;1044  0C7:    ;--------------------------------; (Rn)+
                        ;1045          VANILLA CYCLE,                   ; Done in 2 cycles
                        ;1046          CONDITIONAL FA OP MEM REQ,       ;
                        ;1047          IGPR_IGPR+CTX,                   ;
I 0C7, 0,07C2,880A,0040 ;1048          GOTO [IB.NM.AINC]                ;
                        ;1049
                        ;1050  0D7:    ;--------------------------------; (PC)+
                        ;1051          VANILLA CYCLE, IFORK IF R*BWL,   ;
                        ;1052          ENABLE OPBUS PARITY CHECKING,    ;
                        ;1053          VA_IBUFFER DATA,                 ;
                        ;1054          HMUX CHECK WITH CONTEXT,         ;
                        ;1055          OPVALID IF R+(MV*RN),            ; Set opvalid if Read.
                        ;1056          BRANCH ON (Q+0) AND ACCESS,      ;
I 0D7, 5,0762,A010,0D20 ;1057          GOTO [IB.NM.IMM]                 ;
                        ;1058
                        ;1059  0C6:    ;--------------------------------; @(Rn)+
                        ;1060          VANILLA CYCLE,                   ; Done in 2 cycles
                        ;1061          IGPR_IGPR+4,                     ; bump up the GPR, get the ind. add.
                        ;1062          FORCE FA OP MEM REQ,             ;
I 0C6, 4,07B2,8B2A,0041 ;1063          GOTO [IB.NM.AINCDEF]             ;
                        ;1064
                        ;1065  0D6:    ;--------------------------------; @(PC)+
                        ;1066          VANILLA CYCLE,                   ;
                        ;1067          VA_IBUFFER DATA, COND MEM REQ,   ; VA_address from Istream
                        ;1068          CONDITIONAL FA OP MEM REQ,       ;
                        ;1069          ENABLE OPBUS PARITY CHECKING,    ;
                        ;1070          ENABLE TRAP IF UNAL OP,          ;
                        ;1071          OPVALID IF A+RM*AL,              ;
                        ;1072          IFORK IF A+R*BWL,                ;
                        ;1073          BRANCH ON (Q+0) AND ACCESS,      ;
I 0D6, 1,07C0,C010,4540 ;1074          GOTO [IB.NM.FET]                 ;
```

```
                        ;1075  .PAGE
                        ;1076
                        ;1077  0C8:    ;--------------------------------; -(Rn)
                        ;1078          VANILLA CYCLE,                   ;
                        ;1079          VA_IGPR_IGPR-CTX, COND MEM REQ,  ;
                        ;1080          ENABLE TRAP IF UNAL OP,          ;
                        ;1081          CONDITIONAL FA OP MEM REQ,       ;
                        ;1082          OPVALID IF A+RM*AL,              ;
                        ;1083          IFORK IF A+R*BWL,                ;
                        ;1084          BRANCH ON (Q+0) AND ACCESS,      ;
I 0C8, 0,07C0,C88A,4540 ;1085          GOTO [IB.NM.FET]                 ;
                        ;1086
                        ;1087  0D8:    ;--------------------------------; -(PC) [UNPREDICTABLE]
                        ;1088          RESERVED ADDRESSING FAULT,       ;
                        ;1089          VANILLA CYCLE,                   ; To allow Ibuf and Dram PE latching
I 0D8, 0,0702,801C,0010 ;1090          GOTO [IB.B.JDOT.RAF]             ;
                        ;1091
                        ;1092  0C5:    ;--------------------------------; B^D(Rn)
                        ;1093          VANILLA CYCLE, COND MEM REQ,     ;
                        ;1094          VA_IGPR+IBUFFER DATA,            ;
                        ;1095          CONDITIONAL FA OP MEM REQ,       ;
                        ;1096          ENABLE TRAP IF UNAL OP,          ;
                        ;1097          OPVALID IF A+RM*AL,              ;
                        ;1098          IFORK IF A+R*BWL,                ;
                        ;1099          BRANCH ON (Q+0) AND ACCESS,      ;
I 0C5, 0,07C0,C012,4540 ;1100          GOTO [IB.NM.FET]                 ;
                        ;1101
                        ;1102  0D5:    ;--------------------------------; B^D(PC)
                        ;1103          VANILLA CYCLE, COND MEM REQ,     ;
                        ;1104          CONDITIONAL FA OP MEM REQ,       ;
                        ;1105          VA_AMUXPC+IBUFFER DATA,          ;
                        ;1106          ENABLE TRAP IF UNAL OP,          ;
                        ;1107          OPVALID IF A+RM*AL,              ;
                        ;1108          IFORK IF A+R*BWL,                ;
                        ;1109          BRANCH ON (Q+0) AND ACCESS,      ;
I 0D5, 0,07C0,C010,C540 ;1110          GOTO [IB.NM.FET]                 ;
                        ;1111
                        ;1112  0C3:    ;--------------------------------; W^D(Rn)
                        ;1113          VANILLA CYCLE, COND MEM REQ,     ;
                        ;1114          VA_IGPR+IBUFFER DATA,            ;
                        ;1115          CONDITIONAL FA OP MEM REQ,       ;
                        ;1116          ENABLE TRAP IF UNAL OP,          ;
                        ;1117          OPVALID IF A+RM*AL,              ;
                        ;1118          IFORK IF A+R*BWL,                ;
                        ;1119          BRANCH ON (Q+0) AND ACCESS,      ;
I 0C3, 0,07C0,C012,4540 ;1120          GOTO [IB.NM.FET]                 ;
```

```
                        ;1121  .PAGE
                        ;1122
                        ;1123  0D3:    ;-------------------------------; W^D(PC)
                        ;1124          VANILLA CYCLE, COND MEM REQ,    ;
                        ;1125          VA_AMUXPC+IBUFFER DATA,         ;
                        ;1126          ENABLE TRAP IF UNAL OP,         ;
                        ;1127          CONDITIONAL FA OP MEM REQ,      ;
                        ;1128          OPVALID IF A+RM*AL,             ;
                        ;1129          IFORK IF A+R*BWL,               ;
                        ;1130          BRANCH ON (Q+0) AND ACCESS,     ;
I 0D3, 0,07C0,C010,C540 ;1131          GOTO CIB.NM.FET]                ;
                        ;1132
                        ;1133  0C1:    ;-------------------------------; L^D(Rn)
                        ;1134          VANILLA CYCLE, COND MEM REQ,    ;
                        ;1135          VA_IGPR+IBUFFER DATA,           ;
                        ;1136          CONDITIONAL FA OP MEM REQ,      ;
                        ;1137          ENABLE TRAP IF UNAL OP,         ;
                        ;1138          OPVALID IF A+RM*AL,             ;
                        ;1139          IFORK IF A+R*BWL,               ;
                        ;1140          BRANCH ON (Q+0) AND ACCESS,     ;
I 0C1, 0,07C0,C012,4540 ;1141          GOTO CIB.NM.FET]                ;
                        ;1142
                        ;1143  0D1:    ;-------------------------------; L^D(PC)
                        ;1144          VANILLA CYCLE, COND MEM REQ,    ;
                        ;1145          VA_AMUXPC+IBUFFER DATA,         ;
                        ;1146          ENABLE TRAP IF UNAL OP,         ;
                        ;1147          OPVALID IF A+RM*AL,             ;
                        ;1148          CONDITIONAL FA OP MEM REQ,      ;
                        ;1149          IFORK IF A+R*BWL,               ;
                        ;1150          BRANCH ON (Q+0) AND ACCESS,     ;
I 0D1, 0,07C0,C010,C540 ;1151          GOTO CIB.NM.FET]                ;
                        ;1152
                        ;1153  0C4:    ;-------------------------------; @B^D(Rn)
                        ;1154          VANILLA CYCLE,                  ;
                        ;1155          VA_IGPR+IBUFFER DATA, READ,     ;
                        ;1156          FORCE FA OP MEM REQ,            ;
                        ;1157          CONTEXT IS 4 4 4,               ;
                        ;1158          ENABLE TRAP IF UNAL IND,        ;
I 0C4, 0,07F4,8032,2059 ;1159          GOTO CIB.NM.INDIRECT.FET]       ;
                        ;1160
                        ;1161  0D4:    ;-------------------------------; @B^D(PC)
                        ;1162          VANILLA CYCLE,                  ;
                        ;1163          VA_AMUXPC+IBUFFER DATA, READ,   ;
                        ;1164          FORCE FA OP MEM REQ,            ;
                        ;1165          CONTEXT IS 4 4 4,               ;
                        ;1166          ENABLE TRAP IF UNAL IND,        ;
I 0D4, 0,07F4,8030,A059 ;1167          GOTO CIB.NM.INDIRECT.FET]       ;
```

```
                        ;1168  .PAGE
                        ;1169
                        ;1170  0C2:    ;--------------------------------; @W^D(Rn)
                        ;1171          VANILLA CYCLE,                   ;
                        ;1172          VA_IGPR+IBUFFER DATA, READ,      ;
                        ;1173          FORCE FA OP MEM REQ,             ;
                        ;1174          CONTEXT IS 4 4 4,                ;
                        ;1175          ENABLE TRAP IF UNAL IND,         ;
I 0C2, 0,07F4,8032,2059 ;1176          GOTO [IB.NM.INDIRECT.FET]        ;
                        ;1177
                        ;1178  0D2:    ;--------------------------------; @W^D(PC)
                        ;1179          VANILLA CYCLE,                   ;
                        ;1180          FORCE FA OP MEM REQ,             ;
                        ;1181          VA_AMUXPC+IBUFFER DATA, READ,    ;
                        ;1182          CONTEXT IS 4 4 4,                ;
                        ;1183          ENABLE TRAP IF UNAL IND,         ;
I 0D2, 0,07F4,8030,A059 ;1184          GOTO [IB.NM.INDIRECT.FET]        ;
                        ;1185
                        ;1186  0C0:    ;--------------------------------; @L^D(Rn)
                        ;1187          VANILLA CYCLE,                   ;
                        ;1188          FORCE FA OP MEM REQ,             ;
                        ;1189          VA_IGPR+IBUFFER DATA, READ,      ;
                        ;1190          CONTEXT IS 4 4 4,                ;
                        ;1191          ENABLE TRAP IF UNAL IND,         ;
I 0C0, 0,07F4,8032,2059 ;1192          GOTO [IB.NM.INDIRECT.FET]        ;
                        ;1193
                        ;1194  0D0:    ;--------------------------------; @L^D(PC)
                        ;1195          VANILLA CYCLE,                   ;
                        ;1196          FORCE FA OP MEM REQ,             ;
                        ;1197          VA_AMUXPC+IBUFFER DATA, READ,    ;
                        ;1198          CONTEXT IS 4 4 4,                ;
                        ;1199          ENABLE TRAP IF UNAL IND,         ;
I 0D0, 0,07F4,8030,A059 ;1200          GOTO [IB.NM.INDIRECT.FET]        ;
                        ;1201
                        ;1202  0CC:    ;--------------------------------; S^#literal
                        ;1203          VANILLA CYCLE, IFORK IF R*BWL,   ;
                        ;1204          VA_IBUFFER DATA,                 ;
                        ;1205          ENABLE OPBUS PARITY CHECKING,    ;
                        ;1206          OPVALID IF R+(MV*RN),            ;
                        ;1207          LITERAL UNPACK OPBUS,            ;
                        ;1208          BRANCH ON (Q+0) AND ACCESS,      ;
I 0CC, 5,0603,2010,0D30 ;1209          GOTO [IB.NM.LIT]                 ;
                        ;1210
                        ;1211  0DC:    ;--------------------------------; S^#literal
                        ;1212          VANILLA CYCLE, IFORK IF R*BWL,   ;
                        ;1213          OPVALID IF R+(MV*RN),            ;
                        ;1214          ENABLE OPBUS PARITY CHECKING,    ;
                        ;1215          VA_IBUFFER DATA,                 ;
                        ;1216          LITERAL UNPACK OPBUS,            ;
                        ;1217          BRANCH ON (Q+0) AND ACCESS,      ;
I 0DC, 5,0603,2010,0D30 ;1218          GOTO [IB.NM.LIT]                 ;
```

```
                        ;1219  .PAGE
                        ;1220
                        ;1221          ;-------------------------------; [Rn]
I 0CB, 4,0202,801C,18D8 ;1222  0CB:    INDEX CYCLE, IFORK              ;
                        ;1223
                        ;1224  0DB:    ;-------------------------------; [PC]   [fatal]
                        ;1225          RESERVED ADDRESSING FAULT,      ; It is ok to make this VANILLA CYCLE,
                        ;1226          VANILLA CYCLE,                  ; To allow Ibuf and Dram PE latching
I 0DB, 0,0702,801C,0010 ;1227          GOTO [IB.B.JDOT.RAF]            ;
                        ;1228
                        ;1229  0EB:    ;-------------------------------; index [Rn] [fatal]
                        ;1230          RESERVED ADDRESSING FAULT,      ;
                        ;1231          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0EB, 4,0502,801C,0010 ;1232          GOTO [IB.B.JDOT.RAF]            ;
                        ;1233
                        ;1234  0FB:    ;-------------------------------; index [PC] [fatal]
                        ;1235          RESERVED ADDRESSING FAULT,      ;
                        ;1236          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0FB, 4,0502,801C,0010 ;1237          GOTO [IB.B.JDOT.RAF]            ;
                        ;1238
                        ;1239  0EA:    ;-------------------------------; index Rn   [fatal]
                        ;1240          RESERVED ADDRESSING FAULT,      ;
                        ;1241          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0EA, 4,0502,801C,0010 ;1242          GOTO [IB.B.JDOT.RAF]            ;
                        ;1243
                        ;1244  0FA:    ;-------------------------------; Index PC [UNPREDICTABLE]
                        ;1245          RESERVED ADDRESSING FAULT,      ;
                        ;1246          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0FA, 4,0502,801C,0010 ;1247          GOTO [IB.B.JDOT.RAF]            ;
                        ;1248
                        ;1249  0E9:    ;-------------------------------; index (Rn)
                        ;1250          BOA CYCLE, VA_IGPR,             ; VA_Boa address, split on context to
                        ;1251          BRANCH ON CONTEXT,              ; do index multiplication
                        ;1252          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1253          CONDITIONAL FA OP MEM REQ,      ;
I 0E9, 6,85C2,8002,0288 ;1254          GOTO [IB.NM.INDEX]              ;
                        ;1255
                        ;1256  0F9:    ;-------------------------------; Index (PC) [UNPREDICTABLE]
                        ;1257          RESERVED ADDRESSING FAULT,      ;
                        ;1258          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0F9, 4,0502,801C,0010 ;1259          GOTO [IB.B.JDOT.RAF]            ;
                        ;1260
                        ;1261  0E8:    ;-------------------------------; index -(Rn)
                        ;1262          BOA CYCLE, VA_IGPR_TGPR-CTX,    ;
                        ;1263          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1264          CONDITIONAL FA OP MEM REQ,      ;
                        ;1265          BRANCH ON CONTEXT,              ;
I 0E8, 2,85C2,888A,0288 ;1266          GOTO [IB.NM.INDEX]              ;
                        ;1267
                        ;1268  0F8:    ;-------------------------------; Index -(PC) [UNPREDICTABLE]
                        ;1269          RESERVED ADDRESSING FAULT,      ;
                        ;1270          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0F8, 4,0502,801C,0010 ;1271          GOTO [IB.B.JDOT.RAF]            ;
```

```
                        ;1272  .PAGE
                        ;1273
                        ;1274  0E7:   ;-------------------------------; index (Rn)+
                        ;1275         BOA CYCLE, IGPR_IGPR+CTX,       ;
                        ;1276         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1277         CONDITIONAL FA OP MEM REQ,      ;
I 0E7, 6,85C2,880A,0049 ;1278         GOTO [IB.NM.INDEX.AINC]         ;
                        ;1279
                        ;1280  0F7:   ;-------------------------------; index (PC)+
                        ;1281         GOTO [IB.NM.INDEXPCPOP],        ; Drain the Istream of the useless
                        ;1282         BOA CYCLE, VA_IBUFFER DATA,     ; immediate data, see if we are done
                        ;1283         BRANCH ON CONTEXT,              ; draining it all.
                        ;1284         CONDITIONAL FA OP MEM REQ,      ;
I 0F7, 0,85C2,8010,0278 ;1285         INDEX IMMEDIATE DRAIN           ;
                        ;1286
                        ;1287  0E6:   ;-------------------------------; index @(Rn)+
                        ;1288         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1289         BOA CYCLE, IGPR_IGPR+4,         ;
                        ;1290         FORCE FA OP MEM REQ,            ;
I 0E6, 6,85C2,882A,004A ;1291         GOTO [IB.NM.INDEX.AINCDEF]      ;
                        ;1292
                        ;1293  0F6:   ;-------------------------------; index @(PC)+
                        ;1294         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1295         BOA CYCLE,                      ;
                        ;1296         VA_IBUFFER DATA,                ; VA_4 bytes from the Ibuffer as the BOA
                        ;1297         CONDITIONAL FA OP MEM REQ,      ;
                        ;1298         BRANCH ON CONTEXT,              ;
I 0F6, 6,85C2,8010,0288 ;1299         GOTO [IB.NM.INDEX]              ;
                        ;1300
                        ;1301  0E5:   ;-------------------------------; index B^D(Rn)
                        ;1302         BOA CYCLE,                      ; VA_ sext(Ibuffer byte) + Rn as the BOA
                        ;1303         VA_IGPR+IBUFFER DATA,           ;
                        ;1304         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1305         CONDITIONAL FA OP MEM REQ,      ;
                        ;1306         BRANCH ON CONTEXT,              ;
I 0E5, 2,85C2,8012,0288 ;1307         GOTO [IB.NM.INDEX]              ;
                        ;1308
                        ;1309  0F5:   ;-------------------------------; index B^D(PC)
                        ;1310         BOA CYCLE,                      ;
                        ;1311         VA_AMUXPC+IBUFFER DATA,         ; VA_sext(Ibuff. byte) + CPC to make
                        ;1312         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1313         BRANCH ON CONTEXT,              ; the BOA
                        ;1314         CONDITIONAL FA OP MEM REQ,      ;
I 0F5, 2,85C2,8010,8288 ;1315         GOTO [IB.NM.INDEX]              ;
                        ;1316
                        ;1317  0E3:   ;-------------------------------; index W^D(Rn)
                        ;1318         BOA CYCLE,                      ; VA_ sext (Ibuffer word) + Rn to make
                        ;1319         VA_IGPR+IBUFFER DATA,           ; the BOA
                        ;1320         BRANCH ON CONTEXT,              ;
                        ;1321         INHIBIT SET SCOREBOARD VALID,   ;
                        ;1322         CONDITIONAL FA OP MEM REQ,      ;
I 0E3, 2,85C2,8012,0288 ;1323         GOTO [IB.NM.INDEX]              ;
```

```
                        ;1324  .PAGE
                        ;1325
                        ;1326  0F3:    ;-------------------------------; index W^D(PC)
                        ;1327          BOA CYCLE,                      ;
                        ;1328          VA_AMUXPC+IBUFFER DATA,         ; VA_sext(Ibuff. word) + CPC to make the
                        ;1329          BRANCH ON CONTEXT,              ; BOA
                        ;1330          CONDITIONAL FA OP MEM REQ,      ;
                        ;1331          INHIBIT SET SCOREBOARD VALID,   ;
I 0F3, 2,85C2,8010,8288 ;1332          GOTO DIB.NM.INDEX]              ;
                        ;1333
                        ;1334  0E1:    ;-------------------------------; index L^D(Rn)
                        ;1335          BOA CYCLE,                      ; VA_ Ibuffer l.w. + Rn to make the BOA
                        ;1336          VA_IGPR+IBUFFER DATA,           ;
                        ;1337          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1338          BRANCH ON CONTEXT,              ;
                        ;1339          CONDITIONAL FA OP MEM REQ,      ;
I 0E1, 2,85C2,8012,0288 ;1340          GOTO DIB.NM.INDEX]              ;
                        ;1341
                        ;1342  0F1:    ;-------------------------------; index L^D(PC)
                        ;1343          BOA CYCLE,                      ;
                        ;1344          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1345          VA_AMUXPC+IBUFFER DATA,         ; VA_ (Ibuff. l.w.) + CPC to make the
                        ;1346          BRANCH ON CONTEXT,              ; BOA
                        ;1347          CONDITIONAL FA OP MEM REQ,      ;
I 0F1, 2,85C2,8010,8288 ;1348          GOTO DIB.NM.INDEX]              ;
                        ;1349
                        ;1350  0E4:    ;-------------------------------; index @B^D(Rn)
                        ;1351          VA_IGPR+IBUFFER DATA, READ,     ; VA_sext(ibuff. byte) + GPR to make the
                        ;1352          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1353          ENABLE TRAP IF UNAL IND,        ;
                        ;1354          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1355          BOA CYCLE, FORCE FA OP MEM REQ, ;
I 0E4, 2,85F4,8032,204C ;1356          GOTO DIB.NM.INDEX.INDIRECT]     ;
                        ;1357
                        ;1358  0F4:    ;-------------------------------; index @B^D(PC)
                        ;1359          BOA CYCLE, FORCE FA OP MEM REQ, ;
                        ;1360          VA_AMUXPC+IBUFFER DATA, READ,   ; VA_ sext(ibuff. byte) + CPC to make the
                        ;1361          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1362          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1363          ENABLE TRAP IF UNAL IND,        ;
I 0F4, 2,85F4,8030,A04C ;1364          GOTO DIB.NM.INDEX.INDIRECT]     ;
                        ;1365
                        ;1366  0E2:    ;-------------------------------; index @W^D(Rn)
                        ;1367          BOA CYCLE, FORCE FA OP MEM REQ, ;
                        ;1368          VA_IGPR+IBUFFER DATA, READ,     ; VA_sext(ibuff. word) + GPR to make the
                        ;1369          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1370          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1371          ENABLE TRAP IF UNAL IND,        ;
I 0E2, 2,85F4,8032,204C ;1372          GOTO DIB.NM.INDEX.INDIRECT]     ;
```

```
                        ;1373  .PAGE
                        ;1374
                        ;1375  0F2:    ;--------------------------------; index @W^D(PC)
                        ;1376          BOA CYCLE, FORCE FA OP MEM REQ, ;
                        ;1377          VA_AMUXPC+IBUFFER DATA, READ,   ; VA_ sext(ibuff. word) + CPC to make the
                        ;1378          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1379          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1380          ENABLE TRAP IF UNAL IND,        ;
I 0F2, 2,85F4,8030,A04C ;1381          GOTO [IB.NM.INDEX.INDIRECT]     ;
                        ;1382
                        ;1383  0E0:    ;--------------------------------; index @L^D(Rn)
                        ;1384          BOA CYCLE, FORCE FA OP MEM REQ, ;
                        ;1385          VA_IGPR+IBUFFER DATA, READ,     ; VA_(ibuff. l.w.) + GPR to make the
                        ;1386          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1387          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1388          ENABLE TRAP IF UNAL IND,        ;
I 0E0, 2,85F4,8032,204C ;1389          GOTO [IB.NM.INDEX.INDIRECT]     ;
                        ;1390
                        ;1391  0F0:    ;--------------------------------; index @L^D(PC)
                        ;1392          BOA CYCLE, FORCE FA OP MEM REQ, ;
                        ;1393          VA_AMUXPC+IBUFFER DATA, READ,   ; VA_ (ibuff. l.w.) + CPC to make the
                        ;1394          CONTEXT IS 4 4 4,               ; indirect add. of the BOA
                        ;1395          INHIBIT SET SCOREBOARD VALID,   ;
                        ;1396          ENABLE TRAP IF UNAL IND,        ;
I 0F0, 2,85F4,8030,A04C ;1397          GOTO [IB.NM.INDEX.INDIRECT]     ;
                        ;1398
                        ;1399  0EC:    ;--------------------------------; index S^#literal   [fatal]
                        ;1400          RESERVED ADDRESSING FAULT,      ;
                        ;1401          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0EC, 4,0502,801C,0010 ;1402          GOTO [IB.B.JDOT.RAF]            ;
                        ;1403
                        ;1404  0FC:    ;--------------------------------; index S^#literal   [fatal]
                        ;1405          RESERVED ADDRESSING FAULT,      ;
                        ;1406          BOA CYCLE,                      ; To allow Ibuf and Dram PE latching
I 0FC, 4,0502,801C,0010 ;1407          GOTO [IB.P.JDOT.RAF]            ;
```

```
                        ;1408  .PAGE
                        ;1409
                        ;1410  ;*****************************************************************************
                        ;1411  ; These words fill in the Unused locations in Ifork.  These are here only
                        ;1412  ; for debugging purposes; these can be deleted after prototype debug.
                        ;1413  ;*****************************************************************************
                        ;1414
                        ;1415
                        ;1416          ;--------------------------------;
I 0CD, 4,0002,801C,00CD ;1417  X:      GOTO [X]                         ;
                        ;1418
                        ;1419          ;--------------------------------;
I 0CE, 4,0002,801C,00CE ;1420  XX:     GOTO [XX]                        ;
                        ;1421
                        ;1422          ;--------------------------------;
I 0DD, 0,0002,801C,00DD ;1423  XXX:    GOTO [XXX]                       ;
                        ;1424
                        ;1425          ;--------------------------------;
I 0DE, 0,0002,801C,00DE ;1426  XXXX:   GOTO [XXXX]                      ;
                        ;1427
                        ;1428          ;--------------------------------;
I 0DF, 4,0002,801C,00DF ;1429  XXXXX:  GOTO [XXXXX]                     ;
                        ;1430
                        ;1431          ;--------------------------------;
I 0ED, 0,0002,801C,00ED ;1432  XXXXXX: GOTO [XXXXXX]                    ;
                        ;1433
                        ;1434          ;--------------------------------;
                        ;1435  XXXXXXX:                                 ;
I 0EE, 0,0002,801C,00EE ;1436          GOTO [XXXXXXX]                   ;
                        ;1437
                        ;1438          ;--------------------------------;
                        ;1439  XXXXXXXX:                                ;
I 0FD, 4,0002,801C,00FD ;1440          GOTO [XXXXXXXX]                  ;
                        ;1441
                        ;1442          ;--------------------------------;
                        ;1443  XXXXXXXXX:                               ;
I 0FE, 4,0002,801C,00FE ;1444          GOTO [XXXXXXXXX]                 ;
```

```
;1445  .PAGE   "       Microtraps"
;1446  .TOC    "               Unalignment Microtraps"
;1447
;1448  ;*************************************************************************
;1449  ; There are three types of unaligned microtraps:
;1450  ;
;1451  ; 1) Indirect reference.  Cycle "A" does a READ for an indirect address.  Cycle
;1452  ;                        "B" will not complete if the OP.VA LATCH in cycle "A"
;1453  ;                        detected an unaligned address.  We trap to 0C:,
;1454  ;                        which completes the second reference and POPs back
;1455  ;                        to the interrupted microflow.
;1456  ;
;1457  ; 2) Operand reference, Read*WL.
;1458  ;
;1459  ;                        Cycle "A" does a READ for an operand, and the reference
;1460  ;                        is a READ with word or longword context and is UNALIGNED.
;1461  ;                        In the same cycle, a conditional IFORK command says
;1462  ;                        Ifork if A+R*BWL.  Cycle "A" will cause an Ifork, and
;1463  ;                        cycle "B" will be a microword on the Ifork vector.  We
;1464  ;                        will microtrap out of cycle "B" to 1 of 2 microwords: we go to
;1465  ;                        0D: if the unaligned operand context is LONG, and we trap to
;1466  ;                        08: if the unaligned operand context is WORD.  Both of those
;1467  ;                        microwords complete the second reference.
;1468  ;
;1469  ;                        The address pushed on the microstack is:
;1470  ;
;1471  ;                        IF (Ifork Valid) then (Push the address of microword "B")
;1472  ;                              ELSE (Push BF);
;1473  ;
;1474  ;                        When we return from the second-reference, we will return
;1475  ;                        to location BF if Ifork Valid was not true at microtrap
;1476  ;                        time, or to the Ifork microword if Ifork Valid was true.
;1477  ;
;1478  ; 3) Operand reference, not (Read*BWL)
;1479  ;
;1480  ;                        The same microword "A" in trap #2 above, except that this
;1481  ;                        is not a READ reference*BWL.  Cycle "A"'s conditional
;1482  ;                        Ifork fails and a BRANCH ON (Q+0) AND ACCESS takes us to
;1483  ;                        a number of possible cycle "B"s, each of which handles
;1484  ;                        a particular type of specifier.  We will trap out of
;1485  ;                        cycle "B" and use the low 3 bits of address "B" to
;1486  ;                        index into the microtrap vector, locations 00: to 07:.
;1487  ;                        The microtrap flow which handles the particular type
;1488  ;                        of trap will NOT return to the interrupted microflow.
;1489  ;
;1490  ; Entry Point(s):
;1491  ;       Where?  For What?                           Pop to Old Flow?
;1492  ;       ------  ----------------------------        ----------------
;1493  ;       0C:     Indirect unaligned addresses.       Yes.
;1494  ;       0D:     Read*LONG unaligned operands        Sometimes.
;1495  ;       08:     Read*WORD unaligned operands        Sometimes.
;1496  ;       00-07:  Not(read*(W+L)) unaligned operands  No.
;1497  ;
;1498  ; Input:
;1499  ;       VA LATCH contains the address which caused the unalignment trap.
```

```
;1500  ;*****************************************************************************
```

```
                        ;1501  .PAGE
                        ;1502
                        ;1503  .REGION/MAINLOW,MAINHIGH
                        ;1504
                        ;1505
                        ;1506  ;*****************************************************************
                        ;1507  ; Indirect Unalignment trap comes here.
                        ;1508  ;*****************************************************************
                        ;1509
                        ;1510          ;-------------------------------;
I 00C, 0,0196,8005,000D ;1511  0C:     VA_VA+4, READ.2ND, RETURN       ;
                        ;1512
                        ;1513
                        ;1514
                        ;1515  ;*****************************************************************
                        ;1516  ; Operand Unalignment trap IFF the catylyst reference was a READ*LONG.
                        ;1517  ;*****************************************************************
                        ;1518
                        ;1519          ;-------------------------------;
                        ;1520  0D:     VA_VA+4, READ.2ND, OPVALID,     ; If we are from an Index Ifork cycle,
                        ;1521          UTRAP CIP,                      ; remember to use Index mode in the
I 00D, 3,8196,8025,0008 ;1522          CONTEXT IS 4 4 4, RETURN        ; calculation of the BOA Ifork address.
                        ;1523
                        ;1524
                        ;1525  ;*****************************************************************
                        ;1526  ; Operand Unalignment trap IFF the catylyst reference was a READ*WORD
                        ;1527  ;*****************************************************************
                        ;1528
                        ;1529          ;-------------------------------;
                        ;1530  08:     VA_VA+4, READ.2ND, OPVALID,     ;
                        ;1531          UTRAP CIP,                      ; Ditto as for utrap to 0D:.
I 008, 3,8196,8045,00BF ;1532          CONTEXT IS X 2 X, RETURN        ;
                        ;1533
                        ;1534
                        ;1535
                        ;1536  ;*****************************************************************
                        ;1537  ; As explained on the previous page, Iff -Ifork Valid at READ*(W+L) utrap time
                        ;1538  ; the hardware pushes BF onto the 1-deep microaddress stack (instead of the
                        ;1539  ; address of the word being microtrapped).  When the ucode executes a RETURN,
                        ;1540  ; it will goto this microword, which will do an IFORK command so that we
                        ;1541  ; try to get valid Ifork data into the double buffer.
                        ;1542  ;*****************************************************************
                        ;1543
                        ;1544          ;-------------------------------;
I 0BF, 4,0002,801C,1802 ;1545  0BF:    IFORK                           ;
```

```
                    ;1546  .PAGE
                    ;1547
                    ;1548  ;*************************************************************************
                    ;1549  ; Operand Unalignment traps come here IFF the catylyst reference was
                    ;1550  ; not (READ*WL).
                    ;1551  ;
                    ;1552  ; The low 3 bits of the trap routine vector address are the
                    ;1553  ; low 3 bits of the BRANCH ON (Q+D) AND ACCESS target address.
                    ;1554  ; These are the start addresses for the various micro-routines:
                    ;1555  ;
                    ;1556  ;       Unaligned Condition             Trap Vector Address
                    ;1557  ;       -------------------             -------------------
                    ;1558  ;       WL*Write                        010
                    ;1559  ;       WL*Modify                       011
                    ;1560  ;       (Q+D)*Read                      101
                    ;1561  ;       (Q+D)*Write                     110
                    ;1562  ;       (Q+D)*Modify                    111
                    ;1563  ;*************************************************************************
                    ;1564
                    ;1565  ;*************************************************************************
                    ;1566  ; WL*Write starts here.  No reference has occurred yet.
                    ;1567  ;*************************************************************************
                    ;1568
                    ;1569          ;-------------------------------;
                    ;1570  02:     WRITE, VA_VA, WRITE VA READY,   ;
I 002, 0,002A,8004,002B ;1571      GOTO [IR.UT.U.LASTWRITE]        ;
                    ;1572
                    ;1573
                    ;1574  ;*************************************************************************
                    ;1575  ; WL*Modify starts here.  The first read reference has occurred.
                    ;1576  ;*************************************************************************
                    ;1577
                    ;1578          ;-------------------------------;
                    ;1579  03:     VA_VA+4, OPVALID,               ;
I 003, 4,001A,E005,0000 ;1580      READ.NOPAGE.2ND                 ;
                    ;1581
                    ;1582
                    ;1583          ;-------------------------------;
                    ;1584          VA_VA-4, WRITE VA READY,        ;
                    ;1585          WRITE.NOPAGE,                   ;
I 000, 4,0024,8005,802B ;1586      GOTO [IR.UT.U.LASTWRITE]        ;
```

```
                        ;1587  .PAGE
                        ;1588
                        ;1589  ;*****************************************************************************
                        ;1590  ; (Q+0)*Read starts here.  The first Read reference has occurred.
                        ;1591  ;*****************************************************************************
                        ;1592
                        ;1593          ;--------------------------------;
I 005, 0,0016,E005,0001 ;1594  05:     VA_VA+1, READ.2ND, OPVALID      ; L.W. #1 is done
                        ;1595
                        ;1596
                        ;1597          ;--------------------------------;
                        ;1598          READ, BRANCH ON QUAD OR OCTA,   ;
I 001, 0,0014,8004,021B ;1599          VA_VA                           ;
                        ;1600
                        ;1601
                        ;1602  =1011   ;1011--------------------------; Quad (Octa finishes l.w.#4 here too)
                        ;1603  IB.UT.U.QQ.R:                           ;
                        ;1604          VA_VA+4, OPVALID, READ.2ND,     ; All done
I 01B, 0,0016,E005,181F ;1605          IFORK                           ;
                        ;1606
                        ;1607
                        ;1608          ;1111--------------------------; Octa
I 01F, 0,0016,E005,0004 ;1609          VA_VA+4, OPVALID, READ.2ND      ; L.W.#2 is done
                        ;1610
                        ;1611          ;--------------------------------;
I 004, 4,0014,8004,0011 ;1612          VA_VA, READ                     ; Start on #3
                        ;1613
                        ;1614          ;--------------------------------;
I 011, 4,0016,E005,0012 ;1615          VA_VA+4, READ.2ND, OPVALID      ;
                        ;1616
                        ;1617          ;--------------------------------;
I 012, 4,0014,8004,001B ;1618          VA_VA, READ, GOTO [IB.UT.U.QQ.R]; Start on #4, finish
```

```
                        ;1619  .PAGE
                        ;1620
                        ;1621  ;*****************************************************************
                        ;1622  ; (Q+0)*Write starts here.  No reference has occurred yet.
                        ;1623  ;
                        ;1624  ; The (Q+0)*Modify unaligned flow joins at IB.UT.U.QQ.MODWRI: to write
                        ;1625  ; back the MODIFY operand to memory.
                        ;1626  ;
                        ;1627  ; (WL)*Write and (WL)*Modify unaligned flows join at IB.UT.U.LASTWRITE:
                        ;1628  ; to do the last operand write.
                        ;1629  ;*****************************************************************
                        ;1630
                        ;1631          ;-------------------------------;
I 006, 4,002A,8004,0013 ;1632  06:     WRITE, WRITE VA READY, VA_VA    ;
                        ;1633
                        ;1634
                        ;1635          ;-------------------------------;
                        ;1636  IB.UT.U.QQ.MODWRI:                      ;
I 013, 4,0006,8005,0016 ;1637          VA_VA+4, WRITE.NOPAGE.2ND       ; l.w. #1 is done
                        ;1638
                        ;1639          ;-------------------------------;
                        ;1640          WRITE.NOPAGE, WRITE VA READY,   ; Part 1, l.w. #2
I 016, 0,0024,8004,022B ;1641          VA_VA, BRANCH ON QUAD OR OCTA   ;
                        ;1642
                        ;1643  =1011   ;1011----------------------------; Quad (Octa finishes l.w.#4 here too,
                        ;1644  IB.UT.U.LASTWRITE:                      ; and WL*write+modify flows)
                        ;1645          VA_VA+4, WRITE.NOPAGE.2ND,      ; Done!
I 02B, 4,0006,8005,182F ;1646          IFORK                           ;
                        ;1647
                        ;1648          ;1111----------------------------; Octa
I 02F, 0,0006,8005,0018 ;1649          VA_VA+4, WRITE.NOPAGE.2ND       ; L.w.#2 is done
                        ;1650
                        ;1651
                        ;1652  ;*****************************************************************
                        ;1653  ; Octa unaligned * Write, continue writing the operand to memory.  We have
                        ;1654  ; done the first 2 longwords.
                        ;1655  ;*****************************************************************
                        ;1656
                        ;1657          ;-------------------------------;
                        ;1658          WRITE.NOPAGE, WRITE VA READY,   ;
I 018, 0,0024,8004,0019 ;1659          VA_VA                           ;
                        ;1660
                        ;1661          ;-------------------------------;
I 019, 4,0006,8005,001A ;1662          VA_VA+4, WRITE.NOPAGE.2ND       ;
                        ;1663
                        ;1664          ;-------------------------------;
                        ;1665          WRITE.NOPAGE, WRITE VA READY,   ; Start on #4, finish above in
I 01A, 4,0024,8004,002B ;1666          VA_VA, GOTO [IB.UT.U.LASTWRITE] ; Quad microword
```

```
                        ;1667  .PAGE
                        ;1668
                        ;1669  ;*************************************************************************
                        ;1670  ; (Q+0)*Modify starts here.  The first read reference has occurred.
                        ;1671  ;
                        ;1672  ; We join the (Q+0)*Write unaligned flow at IB.UT.U.QQ.MODWRI: to write
                        ;1673  ; back the operand to memory.
                        ;1674  ;*************************************************************************
                        ;1675
                        ;1676          ;-------------------------------;
                        ;1677  07:      VA_VA+4, READ.NOPAGE.2ND,       ;
I 007, 0,001A,E005,001C ;1678          OPVALID                         ;
                        ;1679
                        ;1680          ;-------------------------------;
                        ;1681          READ, BRANCH ON QUAD OR OCTA,   ;
I 01C, 4,0014,E004,023B ;1682          VA_VA                           ;
                        ;1683
                        ;1684
                        ;1685  =1011    ;1011---------------------------; Quad (Octa finishes reading l.w.#4 here
                        ;1686  IB.UT.U.QQ.M:                            ; too)
                        ;1687          VA_VA+4, READ.NOPAGE.2ND,       ; Done with reading, now back down the
                        ;1688          OPVALID, BRANCH ON QUAD OR OCTA,; VA and join the write flow
I 03B, 0,001A,E005,024B ;1689          GOTO [IB.UT.U.QQ.M.WRI]         ;
                        ;1690
                        ;1691          ;1111---------------------------; Octa
                        ;1692          VA_VA+4, READ.NOPAGE.2ND,       ; L.w.#2 is done
I 03F, 4,001A,E005,0021 ;1693          OPVALID, GOTO [IB.UT.U.OCTAM]   ;
                        ;1694
                        ;1695
                        ;1696
                        ;1697  =1011    ;1011---------------------------; Quad (and Octa, for 3rd VA decrement)
                        ;1698  IB.UT.U.QQ.M.WRI:                        ;
                        ;1699          VA_VA-4,                        ; Now back down the VA to get ready for
I 04B, 0,0002,8005,801E ;1700          GOTO [IB.UT.U.QQ.M.WRI2]        ; the write sequence
                        ;1701
                        ;1702          ;1111---------------------------; Octa
I 04F, 0,0002,8005,801D ;1703          VA_VA-4                         ;
                        ;1704
                        ;1705          ;-------------------------------;
                        ;1706          VA_VA-4,                        ;
I 01D, 0,0002,8005,804B ;1707          GOTO [IB.UT.U.QQ.M.WRI]         ;
                        ;1708
                        ;1709
                        ;1710
                        ;1711          ;-------------------------------;
                        ;1712  IB.UT.U.QQ.M.WRI2:                       ;
                        ;1713          VA_VA-4, WRITE VA READY,        ; VA will now point to first longword,
                        ;1714          WRITE.NOPAGE,                   ; start the write sequence in the MODIFY
I 01E, 0,0024,8005,8013 ;1715          GOTO [IB.UT.U.QQ.MODWRI]        ; *(Q+0) unaligned microcode
```

```
                        ;1716  .PAGE
                        ;1717
                        ;1718  ;*****************************************************************************
                        ;1719  ; Octa*Modify unaligned continues here.  We have read the first 2 longwords.
                        ;1720  ; Continue with longword #3.
                        ;1721  ;*****************************************************************************
                        ;1722
                        ;1723          ;--------------------------------;
                        ;1724  IB.UT.U.OCTAM:                           ;
I 021, 4,0014,8004,0028 ;1725          VA_VA, READ                     ;
                        ;1726
                        ;1727          ;--------------------------------;
                        ;1728          VA_VA+4, READ.NOPAGE.2ND,       ; Reading #3 is done.
I 028, 0,001A,8005,0029 ;1729          OPVALID                         ;
                        ;1730
                        ;1731          ;--------------------------------;
                        ;1732          READ, GOTO CIB.UT.U.QQ.MJ,      ; Start on #4, finish in Quad modify
I 029, 0,0014,8004,0038 ;1733          VA_VA                           ; microword above.
```

```
;1734  .PAGE  "                Ebox Command Microtraps"
;1735
;1736  ;*************************************************************************
;1737  ;
;1738  ; The Ebox Command microtraps are:
;1739  ;
;1740  ; Loc.  Name             Used When?                 Function
;1741  ; ----  ---------------  -------------------------  -------------------------------
;1742  ; 09:   FLUSH WB         Start a String Fetch       Take an address from WBUS and
;1743  ;                                                   start fetching string data (no
;1744  ;                                                   operand proc.)
;1745  ; 0A:   FLUSH WB&LD CPC  Start a new macroflow      Take an address from WBUS and
;1746  ;                                                   start the Ibox from it.
;1747  ; 0B:   FLUSH CPC        Terminate String Fetch     Take the address locked in CPC and
;1748  ;                                                   resume normal operand processing.
;1749  ; 0C:   RLOG Unwind      Rolling back an inst.      Unwind the RLOG to the first
;1750  ;                                                   occurance of a set "First Bit" in
;1751  ;                                                   an RLOG entry.
;1752  ;
;1753  ;       These flows will disable stalls due to SUSPEND or the Ibox.Stop command.
;1754  ;       After servicing the microtrap condition we goto a jump-dot microword for
;1755  ;       RLOG unwinds and the start string fetch, and back to Ifork for start
;1756  ;       new macroflow and terminate string fetch.
;1757  ;
;1758  ; Entry Locations:
;1759  ;       See above chart.
;1760  ;
;1761  ;*************************************************************************
```

```
                        ;1762  .PAGE
                        ;1763
                        ;1764  ;******************************************************************************
                        ;1765  ; Unwind RLOG microtrap service.
                        ;1766  ;******************************************************************************
                        ;1767
                        ;1768          ;-------------------------------;
                        ;1769  0E:      INHIBIT STALLS,                 ;
                        ;1770          UNWIND RLOG ENTRY,              ;
I 00E, 4,0802,846A,0257 ;1771          BRANCH ON RLOG FIRST BIT        ;
                        ;1772
                        ;1773
                        ;1774  =0111    ;0111-------------------------;
                        ;1775  IB.UT.RLOG:                            ;
                        ;1776          INHIBIT STALLS,                 ;
                        ;1777          UNWIND RLOG ENTRY,              ;
                        ;1778          BRANCH ON RLOG FIRST BIT,       ;
I 057, 4,0802,846A,0257 ;1779          GOTO [IB.UT.RLOG]               ;
                        ;1780
                        ;1781          ;1111-------------------------; This Jump-dot loop is used for other
                        ;1782  IB.B.JDOT:                             ; things, BUT NOT FOR RAF LOOPING.
I 05F, 0,0002,801C,005F ;1783          GOTO [IB.B.JDOT]                ;
                        ;1784
                        ;1785
                        ;1786
                        ;1787
                        ;1788  ;******************************************************************************
                        ;1789  ; Flush From WBUS & Load CPC microtrap service.
                        ;1790  ;******************************************************************************
                        ;1791
                        ;1792          ;-------------------------------;
                        ;1793  0A:      INHIBIT STALLS, VA_WBUS,        ;
                        ;1794          CLEAR CPC VALID,                ;
                        ;1795          CONTEXT IS 4 4 4,               ;
I 00A, 0,09BC,8023,802A ;1796          IBUF REQ FROM OP-PORT           ; Initiate first newstream reference
                        ;1797
                        ;1798
                        ;1799          ;-------------------------------;
                        ;1800  IB.UT.F.LDCPC:                         ;
                        ;1801          VA_VA+4, INHIBIT STALLS,        ;
                        ;1802          CONTEXT IS 4 4 4,               ;
                        ;1803          IBUF REQ FROM IBF-PORT,         ;
I 02A, 4,085E,8025,1809 ;1804          FLUSH & LOAD CPC, IFORK         ;
```

```
                        ;1805  .PAGE
                        ;1806
                        ;1807  ;*****************************************************************************
                        ;1808  ; Flush from WBUS & Hold CPC microtrap service.
                        ;1809  ;*****************************************************************************
                        ;1810
                        ;1811          ;-------------------------------;
                        ;1812  09:      INHIBIT STALLS, VA_WBUS,        ;
                        ;1813          CONTEXT IS 4 4 4,               ;
I 009, 4,081C,8023,802C ;1814          IBUF REQ FROM OP-PORT           ;
                        ;1815
                        ;1816          ;-------------------------------;
                        ;1817          VA_VA+4, FLUSH,                 ; Inc. string address & load VIBA, invalidate
                        ;1818          IBUF REQ FROM IBP-PORT,         ; IBUFFER, initiate second reference
                        ;1819          CONTEXT IS 4 4 4,               ;
                        ;1820          INHIBIT STALLS,                 ;
I 02C, 0,087E,8025,005F ;1821          GOTO [IB.B.JDOT]                ;
                        ;1822
                        ;1823
                        ;1824
                        ;1825  ;*****************************************************************************
                        ;1826  ; Flush from CPC microtrap service (End String fetch).
                        ;1827  ;*****************************************************************************
                        ;1828
                        ;1829          ;-------------------------------;
                        ;1830  03:      INHIBIT STALLS, VA_CPC,         ;
                        ;1831          CONTEXT IS 4 4 4,               ;
                        ;1832          IBUF REQ FROM OP-PORT,          ;
I 00B, 0,081C,802C,002A ;1833          GOTO [IB.UT.F.LDCPC]            ;
```

```
                        ;1834  .PAGE  "                Master Reset Microtrap"
                        ;1835
                        ;1836  ;*************************************************************************
                        ;1837  ; This word executes when the MASTER RESET signal is asserted by the Console.
                        ;1838  ;*************************************************************************
                        ;1839
                        ;1840          ;--------------------------------;  Invalidate the Ibuffer, don't stall,
I 00F, 4,0862,801C,002D ;1841  0F:     FLUSH, INHIBIT STALLS            ;  *** allow ibuffer shift but NOTE that
                        ;1842                                           ;  no Ibuf memory references can occur
                        ;1843                                           ;  since Master Reset forces no IB mem.
                        ;1844                                           ;  ref. ***
                        ;1845
                        ;1846          ;--------------------------------;
                        ;1847  IB.UT.M.FREEZE:                          ;
I 02D, 0,0002,801C,002D ;1848          GOTO [IB.UT.M.FREEZE]            ;
```

```
                        ;1849  .PAGE   "       Branches"
                        ;1850
                        ;1851  ;*****************************************************************
                        ;1852  ;
                        ;1853  ;       Conditional Branches (e.g. BEQ) and Unconditional Branches (e.g.
                        ;1854  ;       BRB, BRW) are handled differently.  Conditional Branches calculated
                        ;1855  ;       the target address on Ifork and goto IB.B.COND.BR:, which will
                        ;1856  ;       activate a stall term until CC VALID.  When this cycle unsuspends it
                        ;1857  ;       fires off the first reference.  Then in the third cycle a test is
                        ;1858  ;       made on the PSL condition codes, if branch false the first reference
                        ;1859  ;       is aborted and the second reference & associated hardware reset does
                        ;1860  ;       not occur.  UNCONDITIONAL branches know that the branch will occur,
                        ;1861  ;       so the Ifork microword calculates the target address and fires the
                        ;1862  ;       first reference, and the second microword below fires the second
                        ;1863  ;       reference and goes to IFORK.
                        ;1864  ;
                        ;1865  ;       Computed branches work identically as conditional branches.  The branch
                        ;1866  ;       /nobranch test is made on the CC's.
                        ;1867  ;
                        ;1868  ; Entry Point(s):
                        ;1869  ;       IB.B.CONDCOMP.BR:
                        ;1870  ;       IB.B.UNCOND.BR:
                        ;1871  ;
                        ;1872  ; Input:
                        ;1873  ;       VA has the target address.
                        ;1874  ;
                        ;1875  ; Output:
                        ;1876  ;       Branch_true             Target stream will be fetched.
                        ;1877  ;       not(branch_true)        Abort the initial target stream reference and
                        ;1878  ;                               continue with the current Istream.
                        ;1879  ;
                        ;1880  ;*****************************************************************
                        ;1881
                        ;1882
                        ;1883          ;-------------------------------;
                        ;1884  IB.B.CONDCOMP.BR:                      ;
                        ;1885          CONTEXT IS 4 4 4, VA_VA,       ;
                        ;1886          IBUF REQ FROM OP-PORT,         ; Stall here until CC VALID, fire first
I 02E, 4,009C,8024,0031 ;1887          WAIT FOR CC VALID              ; reference
                        ;1888
                        ;1889          ;-------------------------------;
                        ;1890          VA_VA+4, IBUF REQ FROM IBF-PORT,; Abort this and previous Mbox command
                        ;1891          COND IBUF FLUSH & LOAD CPC,    ; if not branch_true
                        ;1892          CONTEXT IS 4 4 4, IFORK,       ;
I 031, 4,088E,8025,1838 ;1893          INHIBIT STALLS                 ;
                        ;1894
                        ;1895
                        ;1896          ;-------------------------------;
                        ;1897  IB.B.UNCOND.BR:                        ;
                        ;1898          VA_VA+4, IBUF REQ FROM IBF-PORT,;
                        ;1899          CONTEXT IS 4 4 4,              ;
                        ;1900          FLUSH & LOAD CPC, IFORK,       ;
I 038, 0,085E,8025,1810 ;1901          INHIBIT STALLS                 ;
```

```
                    ;1902  .PAGE  "      Rn"
                    ;1903
                    ;1904  ;*****************************************************************
                    ;1905  ;
                    ;1906  ; Flow(s):
                    ;1907  ;       Rmode operands, Asrc or (Quad+Octa)*(Read+Modify), continue here.
                    ;1908  ;       All other Rn operands were handled in 1 cycle on Ifork.
                    ;1909  ;
                    ;1910  ; Description:
                    ;1911  ;       If Asrc, we set RAF bit.  If -Asrc, we finish passing the next
                    ;1912  ;       1 or 3 GPRs to the Ebox.
                    ;1913  ;
                    ;1914  ; Entry Point(s):
                    ;1915  ;       IB.NM.RN: branching on  (Q+0) AND ACCESS.
                    ;1916  ;
                    ;1917  ; Input:
                    ;1918  ;       The GPR address latch has the next GPR address.
                    ;1919  ;
                    ;1920  ;*****************************************************************
                    ;1921
                    ;1922
                    ;1923  ;*****************************************************************
                    ;1924  ; Since the Ifork uword did an Ifork if Vsrc + Write + ((R+M)*BWL),
                    ;1925  ; some branch target words are not used.
                    ;1926  ;*****************************************************************
                    ;1927
                    ;1928  =000    ;000--------------------------; Asrc*BWL
                    ;1929  IB.NM.RN:                             ;
                    ;1930  IB.B.JDOT.RAF:                        ; All RAFs loop here, keeping the RAF bit
                    ;1931          RESERVED ADDRESSING FAULT,    ; in the IBE register set.
I 010, 0,0102,801C,0010 ;1932          GOTO [IB.B.JDOT.RAF]          ;
                    ;1933
                    ;1934          ;100--------------------------; Asrc*(Q+0)
                    ;1935  =100    RESERVED ADDRESSING FAULT,    ;
I 014, 0,0102,801C,0010 ;1936          GOTO [IB.B.JDOT.RAF]          ;
                    ;1937
                    ;1938          ;101--------------------------; Read*(Q+0)
                    ;1939          FORCE SET SB VALID & ENA OPBUS PARITY CHECKING, ;
                    ;1940          OPVALID, VA_GPR(PREVADD+1),   ;
                    ;1941          IFORK IF QUAD,                ; We are done if Quad Rn operand
I 015, 3,0002,8202,1039 ;1942          GOTO [IB.NM.RN.OCTA]          ;
                    ;1943
                    ;1944  =111    ;111--------------------------; Modify*(Q+0)
                    ;1945          OPVALID, VA_GPR(PREVADD+1),   ;
                    ;1946          ENABLE OPBUS PARITY CHECKING, ;
                    ;1947          NO SCOREBOARD CHECK,          ;
                    ;1948          IFORK IF QUAD,                ; We are done if Quad Rn operand
I 017, 1,00B2,8202,1039 ;1949          GOTO [IB.NM.RN.OCTA]          ;
                    ;1950  =
```

```
                        ;1951  .PAGE
                        ;1952
                        ;1953          ;-------------------------------; Here to finish Octa*Rn Read+mod.
                        ;1954  IB.MM.RN.OCTA:                          ; specifier
                        ;1955          ENABLE OPBUS PARITY CHECKING,   ;
I 039, 1,0002,8202,003A ;1956          OPVALID, VA_GPR(PREVADD+1)      ;
                        ;1957
                        ;1958          ;-------------------------------;
                        ;1959          ENABLE OPBUS PARITY CHECKING,   ;
                        ;1960          OPVALID, VA_GPR(PREVADD+1),     ;
I 03A, 5,0002,8202,1820 ;1961          IPDRK                           ;
```

Bur 2 Q/3 0/5

```
                        ;1962  .PAGE   "      (PC)+"
                        ;1963
                        ;1964  ;*****************************************************************
                        ;1965  ; Native Mode (PC)+ specifier.  The Ifork microword handled Read*BWL
                        ;1966  ; types of this specifier in one cycle.
                        ;1967  ;
                        ;1968  ; Entry Point(s):
                        ;1969  ;       IB.NM.IMM:, branching on (Q+0) AND ACCESS.  The Ifork microword
                        ;1970  ;       did the following:
                        ;1971  ;
                        ;1972  ;       VA_IBUFFER DATA, OPVALID IF R+(MV*RN),
                        ;1973  ;       IFORK IF R*BWL, BRANCH ON (Q+0) AND ACCESS,
                        ;1974  ;       GOTO [IB.NM.IMM]
                        ;1975  ;
                        ;1976  ; Input:
                        ;1977  ;       If BWL*read, we have sent the read data to the Ebox.
                        ;1978  ;       If (Q+0)*read, we have send the first longword of data to the Ebox.
                        ;1979  ;*****************************************************************
                        ;1980
                        ;1981
                        ;1982  =000    ;000--------------------------; Asrc*BWL
                        ;1983  IB.NM.IMM:                           ;
I 020, 0,0002,E088,9822 ;1984          VA_AMUXPC-CTX, OPVALID, IFORK  ;
                        ;1985
                        ;1986  =010    ;010--------------------------; Write*BWL
                        ;1987          RESERVED ADDRESSING FAULT,     ;
I 022, 0,0102,801C,0010 ;1988          GOTO [IB.B.JDOT.RAF]           ;
                        ;1989
                        ;1990          ;011--------------------------; Modify*BWL
                        ;1991          RESERVED ADDRESSING FAULT,     ;
I 023, 0,0102,801C,0010 ;1992          GOTO [IB.B.JDOT.RAF]           ;
                        ;1993
                        ;1994          ;100--------------------------; Asrc*(Q+0)
                        ;1995          VA_IBUFFER DATA,               ; Swallow 1 or 3 more longwords from
                        ;1996          BRANCH ON QUAD OR OCTA,        ; the ibuffer
I 024, 0,0002,8010,026B ;1997          GOTO [IB.NM.IMM.QO.ASRC]       ;
                        ;1998
                        ;1999          ;101--------------------------; Read*(Q+0)
                        ;2000          VA_IBUFFER DATA, OPVALID,      ; We are done if Quad
                        ;2001          ENABLE OPBUS PARITY CHECKING,  ;
                        ;2002          IFORK IF QUAD,                 ;
I 025, 1,0002,E010,103C ;2003          GOTO [IB.NM.IMM.OCTA.READ]     ;
                        ;2004
                        ;2005          ;110--------------------------; Write*(Q+0)
                        ;2006          RESERVED ADDRESSING FAULT,     ;
I 026, 0,0102,801C,0010 ;2007          GOTO [IB.B.JDOT.RAF]           ;
                        ;2008
                        ;2009          ;111--------------------------; Modify*(Q+0)
                        ;2010          RESERVED ADDRESSING FAULT,     ;
I 027, 0,0102,801C,0010 ;2011          GOTO [IB.B.JDOT.RAF]           ;
```

```
                        ;2012  .PAGE
                        ;2013
                        ;2014  ;*************************************************************************
                        ;2015  ; Read * Octa immediate mode operands continue here.
                        ;2016  ;*************************************************************************
                        ;2017
                        ;2018          ;-------------------------------;
                        ;2019  IB.NM.IMM.OCTA.READ:                    ;
                        ;2020          ENABLE OPBUS PARITY CHECKING,   ;
I 03C, 1,0002,E010,003D ;2021          VA_IBUFFER DATA, OPVALID        ;
                        ;2022
                        ;2023          ;-------------------------------;
                        ;2024          ENABLE OPBUS PARITY CHECKING,   ;
I 03D, 1,0002,E010,186B ;2025          VA_IBUFFER DATA, OPVALID, IFORK ;
                        ;2026
                        ;2027
                        ;2028
                        ;2029  ;*************************************************************************
                        ;2030  ; Asrc * (Quad+Octa) immediate mode operands continue here.
                        ;2031  ;
                        ;2032  ; =1011 is Quad.
                        ;2033  ; =1111 is Octa.
                        ;2034  ;*************************************************************************
                        ;2035
                        ;2036          ;1011-----------------------------;
                        ;2037  =1011                                     ;
                        ;2038  IB.NM.IMM.QO.ASRC:                       ; Quad and Octa Immediate Asrc's finish
I 06B, 0,0002,E088,986F ;2039          VA_AMUXPC-CTX, OPVALID, IFORK   ; here
                        ;2040
                        ;2041          ;1111-----------------------------;
I 06F, 4,0002,8010,003E ;2042          VA_IBUFFER DATA                 ;
                        ;2043
                        ;2044          ;-------------------------------;
                        ;2045          VA_IBUFFER DATA,                ;
I 03E, 4,0002,8010,006B ;2046          GOTO [IB.NM.IMM.QO.ASPC]        ;
```

```
                        ;2047  .PAGE   "       (Rn)+ and @(Rn)+"
                        ;2048
                        ;2049  ;*****************************************************************
                        ;2050  ;
                        ;2051  ; (Rn)+ and @(Rn)+ specifiers are handled in at least 2 cycles.  The Ifork
                        ;2052  ; uword loaded GPR with GPR + context for (Rn)+ and GPR + 4 for @(Rn)+.
                        ;2053  ; These microwords load VA with the right thing and join the general fetch
                        ;2054  ; flows.
                        ;2055  ;
                        ;2056  ; Entry Point(s):
                        ;2057  ;       IB.NM.AINC:
                        ;2058  ;       IB.NM.AINCDEF:          if @(Rn)+
                        ;2059  ;
                        ;2060  ; Input:
                        ;2061  ;       GPR specified by IGPR (now PREVADD) has the original contents + context
                        ;2062  ;       or 4.
                        ;2063  ;
                        ;2064  ;*****************************************************************
                        ;2065
                        ;2066
                        ;2067          ;-------------------------------;
                        ;2068  IB.NM.AINC:                              ;
                        ;2069          VA_IGPR(PREVADD)-CTX,           ;
                        ;2070          COND MEM REQ,                   ;
                        ;2071          OPVALID IF A+RM*AL,             ;
                        ;2072          ENABLE TRAP IF UNAL OP,         ; Trap next cycle if unaligned add.,
                        ;2073          BRANCH ON (Q+0) AND ACCESS,     ; goto common fetch routine if needed
                        ;2074          NO SCOREBOARD CHECK,            ;
                        ;2075          IFORK IF A+R*BWL,               ;
I 040, 4,00E0,C38A,4540 ;2076          GOTO CIB.NM.FET]                ;
                        ;2077
                        ;2078
                        ;2079          ;-------------------------------;
                        ;2080  IB.NM.AINCDEF:                           ;
                        ;2081          VA_IGPR(PREVADD)-4,             ;
                        ;2082          NO SCOREBOARD CHECK,            ;
                        ;2083          READ, CONTEXT IS -4 4 X,        ;
                        ;2084          ENABLE TRAP IF UNAL IND,        ; Trap next cycle if unaligned add.
I 041, 0,00F4,83AA,2059 ;2085          GOTO CIB.NM.INDIRECT.FET]       ;
```

```
                        ;2086  .PAGE   "      Literal Specifiers"
                        ;2087
                        ;2088  ;*****************************************************************************
                        ;2089  ;
                        ;2090  ; Short Literal operands come here if they are not Read*BWL.
                        ;2091  ;
                        ;2092  ; Entry Point(s):
                        ;2093  ;       IB.NM.LIT: branching on (Q+O) AND ACCESS.
                        ;2094  ;
                        ;2095  ; Input:
                        ;2096  ;       The first longword of a Quad+Octa literal was queued to Ebox
                        ;2097  ;       on Ifork.
                        ;2098  ;
                        ;2099  ;*****************************************************************************
                        ;2100
                        ;2101  =000    ;000-------------------------------; Asrc*BWL
                        ;2102  IB.NM.LIT:                                  ;
                        ;2103          RESERVED ADDRESSING FAULT,          ;
I 030, 0,0102,801C,0010 ;2104          GOTO [IB.B.JDOT.RAF]                ;
                        ;2105
                        ;2106  =010    ;010-------------------------------; Write*BWL
                        ;2107          RESERVED ADDRESSING FAULT,          ;
I 032, 0,0102,801C,0010 ;2108          GOTO [IB.B.JDOT.RAF]                ;
                        ;2109
                        ;2110          ;011-------------------------------; Modify*BWL
                        ;2111          RESERVED ADDRESSING FAULT,          ;
I 033, 0,0102,801C,0010 ;2112          GOTO [IB.B.JDOT.RAF]                ;
                        ;2113
                        ;2114          ;100-------------------------------; Asrc*(Q+O)
                        ;2115          RESERVED ADDRESSING FAULT,          ;
I 034, 0,0102,801C,0010 ;2116          GOTO [IB.B.JDOT.RAF]                ;
                        ;2117
                        ;2118          ;101-------------------------------; Read*(Q+O)
                        ;2119          VA_0, OPVALID, IFORK IF QUAD,       ;
I 035, 0,0002,E000,1044 ;2120          GOTO [IB.NM.LIT.OCTAREAD]           ;
                        ;2121
                        ;2122          ;110-------------------------------; Write*(Q+O)
                        ;2123          RESERVED ADDRESSING FAULT,          ;
I 036, 0,0102,801C,0010 ;2124          GOTO [IB.B.JDOT.RAF]                ;
                        ;2125
                        ;2126          ;111-------------------------------; Modify*(Q+O)
                        ;2127          RESERVED ADDRESSING FAULT,          ;
I 037, 0,0102,801C,0010 ;2128          GOTO [IB.B.JDOT.RAF]                ;
                        ;2129
                        ;2130
                        ;2131  ;*****************************************************************************
                        ;2132  ; Octa*Read Literal mode specifiers continue here
                        ;2133  ;*****************************************************************************
                        ;2134
                        ;2135          ;-------------------------------;
                        ;2136  IB.NM.LIT.OCTAREAD:                        ;
I 044, 4,0002,E000,0048 ;2137          VA_0, OPVALID                       ; Longword #3
                        ;2138
                        ;2139          ;-------------------------------;
I 048, 0,0002,E000,1849 ;2140          VA_0, OPVALID, IFORK                ; Longword #4, done
```

```
                        ;2141  .PAGE  "       Indexed - Indexed (Rn)+, @(Rn)+ Second Cycle"
                        ;2142
                        ;2143  ;*****************************************************************
                        ;2144  ;
                        ;2145  ; All (Rn)+ and @(Rn)+ specifiers are decoded in 2 cycles, incl. the indexed.
                        ;2146  ; versions.  The Ifork uword loaded IGPR with IGPR + Context or IGPR + 4,
                        ;2147  ; depending on the specifier.  These microwords below decrement IGPR and
                        ;2148  ; load the result into VA.
                        ;2149  ;
                        ;2150  ; Entry Point(s):
                        ;2151  ;       IB.NM.INDEX.AINC:
                        ;2152  ;       IB.NM.INDEX.AINCDEF:
                        ;2153  ;
                        ;2154  ; Input:
                        ;2155  ;       IGPR(PREVADD) will get us the incremented value of IGPR.
                        ;2156  ;
                        ;2157  ;*****************************************************************
                        ;2158
                        ;2159          ;-------------------------------; index (Rn)+
                        ;2160  IB.NM.INDEX.AINC:                       ;
                        ;2161          NO SCOREBOARD CHECK,            ;
                        ;2162          VA_IGPR(PREVADD)-CTX,           ;
                        ;2163          BRANCH ON CONTEXT,              ;
I 049, 0,00E2,838A,0288 ;2164          GOTO [IB.NM.INDEX]              ;
                        ;2165
                        ;2166          ;-------------------------------; index @(Rn)+
                        ;2167  IB.NM.INDEX.AINCDEF:                    ;
                        ;2168          NO SCOREBOARD CHECK,            ;
                        ;2169          VA_IGPR(PREVADD)-4,             ;
                        ;2170          CONTEXT IS -4 4 X, READ,        ;
                        ;2171          ENABLE TRAP IF UNAL IND,        ;
I 04A, 4,00F4,83AA,204C ;2172          GOTO [IB.NM.INDEX.INDIRECT]     ;
```

```
                        ;2173  .PAGE  "      Indexed - Indirect BOA Fetch"
                        ;2174
                        ;2175  ;*****************************************************************
                        ;2176  ;
                        ;2177  ; Indexed Indirect specifiers (@(Rn)+, @x^D(Rn), etc.) must fetch the indirect
                        ;2178  ; BOA from memory before joining the common Index-multiplier microcode.
                        ;2179  ;
                        ;2180  ; Description:
                        ;2181  ;       The Ifork microword initiated the indirect fetch and loaded
                        ;2182  ;       the GPR if indirect autoincrement mode.
                        ;2183  ;
                        ;2184  ;
                        ;2185  ; Entry Point(s):
                        ;2186  ;       IP.NM.INDEX.INDIRECT: directly, to wait for IMD MDRESP on the indirect
                        ;2187  ;       fetch.
                        ;2188  ;
                        ;2189  ; Output:
                        ;2190  ;       VA will contain the BOA when we branch to IB.NM.INDEX:.
                        ;2191  ;
                        ;2192  ;*****************************************************************
                        ;2193
                        ;2194
                        ;2195          ;-------------------------------;
                        ;2196  IB.NM.INDEX.INDIRECT:                   ;
I 04C, 0,00C2,801C,004D ;2197          STALL UNTIL IMD MDRESP          ;
                        ;2198
                        ;2199          ;-------------------------------;
                        ;2200          VA_IMD, BRANCH ON CONTEXT,      ;
I 04D, 0,0002,8014,0288 ;2201          GOTO [IB.NM.INDEX]              ;
```

```
                        ;2202  .PAGE   "       Index (PC)+ Ibuffer Draining"
                        ;2203
                        ;2204  ;*****************************************************************
                        ;2205  ; [Rn](PC)+ mode must drain useless immediate-mode data from the Ibuffer.  The
                        ;2206  ; Ifork microword did a VA_IBUFFER DATA which drained 1, 2, or 4 bytes.
                        ;2207  ;
                        ;2208  ; Entry Point(s):
                        ;2209  ;       IB.NM.INDEXPCPOP: branching on context.
                        ;2210  ;       =1000 is Byte   =1001 is Word   =1010 is Longword
                        ;2211  ;       =1011 is Quad   =1100 is Octa
                        ;2212  ;
                        ;2213  ; Input:
                        ;2214  ;       The Istream has been fixed if context = BYTE, WORD, or LONGWORD.
                        ;2215  ;       If the context is QUAD or OCTA, only the first longword of the
                        ;2216  ;       immediate-mode data has been swallowed by the microcode.
                        ;2217  ;
                        ;2218  ; Output:
                        ;2219  ;       VA will contain AMUX PC - CONTEXT, which will be the address of the
                        ;2220  ;       immediate mode data which we have eaten out of the Istream.
                        ;2221  ;
                        ;2222  ;       The immediate mode data in the Ibuffer will have been disgarded.
                        ;2223  ;
                        ;2224  ;       We will branch to IB.NM.INDEX: after loading VA with the BOA.
                        ;2225  ;*****************************************************************
                        ;2226
                        ;2227
                        ;2228  =1000   ;1000-------------------------; Byte context, IBUF is ok
                        ;2229  IB.NM.INDEXPCPOP:                     ;
                        ;2230          VA_AMUXPC-CTX,                ; VA _ BOA, don't have to branch since
                        ;2231          BRANCH ON CONTEXT,            ; we know this is BYTE, but it doesn't
I 078, 4,0002,8088,8288 ;2232          GOTO [IB.NM.INDEX]            ; hurt
                        ;2233
                        ;2234          ;1001-------------------------; Word context, ditto
                        ;2235          VA_AMUXPC-CTX,                ; VA _ BOA, don't have to branch since
                        ;2236          BRANCH ON CONTEXT,            ; we know this is WORD, but it doesn't
I 079, 4,0002,8088,8288 ;2237          GOTO [IP.NM.INDEX]            ; hurt
                        ;2238
                        ;2239          ;1010-------------------------; Longword context, ditto
                        ;2240  IB.NM.INDEXPCPOP.LQO:                 ;
                        ;2241          VA_AMUXPC-CTX,                ; VA _ BOA, Quad and Octa use this word
                        ;2242          BRANCH ON CONTEXT,            ; too so branch on context
I 07A, 4,0002,8088,8288 ;2243          GOTO [IB.NM.INDEX]            ;
                        ;2244
                        ;2245          ;1011-------------------------; Quad context, eat 4 more Ibuff bytes
                        ;2246  IB.NM.INDEXPCPOP.QO:                  ; (Octa uses this to eat last longword)
                        ;2247          VA_IBUFFER DATA,              ;
                        ;2248          INDEX IMMEDIATE DRAIN,        ;
I 07B, 0,8002,8010,007A ;2249          GOTO [IB.NM.INDEXPCPOP.LQO]   ;
                        ;2250
                        ;2251          ;1100-------------------------; Octa context, eat 12 more Ibuff bytes
                        ;2252          INDEX IMMEDIATE DRAIN,        ;
I 07C, 4,8002,8010,004E ;2253          VA_IBUFFER DATA               ;
                        ;2254  =
```

```
                        ;2255  .PAGE
                        ;2256
                        ;2257          ;--------------------------------;
                        ;2258          INDEX IMMEDIATE DRAIN,           ;
                        ;2259          VA_IBUFFER DATA,                 ; Third longword is done, handle last
I 04E, 4,8002,8010,007B ;2260          GOTO EIB.MM.INDEXPCPOP.003       ; longword with Quad flow
```

```
                    ;2261  .PAGE  "      Index Multiplier Code"
                    ;2262
                    ;2263  ;*****************************************************************
                    ;2264  ;
                    ;2265  ; After the particular Ifork Index microflow has loaded VA with the Base
                    ;2266  ; Operand Address (BOA), it branches to here on DRAM context.  This code multiplies
                    ;2267  ; the Index value appropriately, then branches to IB.NM.FET:, which is the
                    ;2268  ; common fetch-phase code.
                    ;2269  ;
                    ;2270  ; Description:
                    ;2271  ;       The last word to complete BOA fetching did the following:
                    ;2272  ;
                    ;2273  ;       BRANCH ON CONTEXT,
                    ;2274  ;       GOTO [IB.NM.INDEX]
                    ;2275  ;
                    ;2276  ; Entry Point(s):
                    ;2277  ;       IB.NM.INDEX:, branching on context.
                    ;2278  ;
                    ;2279  ;       =1000 is byte index.                    =1001 is word index.
                    ;2280  ;       =1010 is longword index.                =1011 is quad index.
                    ;2281  ;       =1100 is octa index.
                    ;2282  ;
                    ;2283  ; Input:
                    ;2284  ;       VA contains the BOA and the Index Register in the Ibox datapath contains
                    ;2285  ;       the number of the index register ([Rn]).
                    ;2286  ;
                    ;2287  ; Output:
                    ;2288  ;       VA will finally contain the address of the operand, at which point we will
                    ;2289  ;       do a conditional memory request and branch to common ucode which
                    ;2290  ;       handles the fetch-phase of operand handling.
                    ;2291  ;
                    ;2292  ;*****************************************************************
                    ;2293
                    ;2294  =1000   ;1000-------------------------; Byte index context
                    ;2295  IB.NM.INDEX:                          ;
                    ;2296          VA_VA+INDEX GPR, COND MEM REQ,  ;
                    ;2297          ENABLE TRAP IF UNAL OP,         ;
                    ;2298          OPVALID IF A+RM*AL,             ;
                    ;2299          IFORK IF A+R*BWL,               ;
                    ;2300          BRANCH ON (Q+0) AND ACCESS,     ;
                    ;2301          FORCE SET SCOREBOARD VALID,     ;
I 088, 6,0000,C106,4540 ;2302          GOTO [IB.NM.FET]                ;
                    ;2303
                    ;2304          ;1001-------------------------; Word index context
                    ;2305          VA_VA+2(INDEX GPR),             ;
                    ;2306          ENABLE TRAP IF UNAL OP,         ;
                    ;2307          OPVALID IF A+RM*AL,             ;
                    ;2308          IFORK IF A+R*BWL, COND MEM REQ, ;
                    ;2309          FORCE SET SCOREBOARD VALID,     ;
                    ;2310          BRANCH ON (Q+0) AND ACCESS,     ;
I 089, 2,0000,C106,C540 ;2311          GOTO [IB.NM.FET]                ;
```

```
                        ;2312  .PAGE
                        ;2313
                        ;2314          ;1010--------------------------; Longword index context
                        ;2315  IB.NM.INDEX.LQO:                       ; or last cycle of Quad or Octa
                        ;2316          VA_VA+4(INDEX GPR),            ; Index cooking..
                        ;2317          ENABLE TRAP IF UNAL OP,        ;
                        ;2318          OPVALID IF A+RM*AL,            ;
                        ;2319          IFORK IF A+R*BWL, COND MEM REQ, ;
                        ;2320          FORCE SET SCOREBOARD VALID,    ;
                        ;2321          BRANCH ON (Q+O) AND ACCESS,    ;
I 08A, 2,0000,C107,4540 ;2322          GOTO [IB.NM.FET]               ;
                        ;2323
                        ;2324          ;1011--------------------------; Quad index context
                        ;2325  IB.NM.INDEX.QO:                        ; Quad takes 2 cycles and Octa takes
                        ;2326          VA_VA+4(INDEX GPR),            ; 4 cycles.
I 08B, 0,0002,8107,008A ;2327          GOTO [IB.NM.INDEX.LQO]         ;
                        ;2328
                        ;2329          ;1100--------------------------; Octa index context
I 08C, 4,0002,8107,0050 ;2330          VA_VA+4(INDEX GPR)             ;
                        ;2331  =
                        ;2332          ;------------------------------;
                        ;2333          VA_VA+4(INDEX GPR),            ;
I 050, 4,0002,8107,0088 ;2334          GOTO [IB.NM.INDEX.QO]          ;
                        ;2335
```

```
                        ;2336  .PAGE   "       Fetch Phase Common Routine"
                        ;2337
                        ;2338  ;*****************************************************************
                        ;2339  ;
                        ;2340  ; After calculating the VA of an operand in memory, most memory operand
                        ;2341  ; specifier flows branch to this code on a (Q+O) AND ACCESS branch.
                        ;2342  ; The word branching to here did the following:
                        ;2343  ;
                        ;2344  ;       COND MEM REQ,
                        ;2345  ;       OPVALID IF A+RM*AL,
                        ;2346  ;       ENABLE TRAP IF UNAL OP,
                        ;2347  ;       BRANC ON (Q+O) AND ACCESS,
                        ;2348  ;       IFORK IF A+R*BWL,
                        ;2349  ;       GOTO [IB.NM.FET]
                        ;2350  ;
                        ;2351  ; Entry Point(s):
                        ;2352  ;       IB.NM.FET:.
                        ;2353  ;
                        ;2354  ; Input:
                        ;2355  ;       VA has the operand address, and a conditional memory request has
                        ;2356  ;       been requested of the hardware.
                        ;2357  ;
                        ;2358  ;*****************************************************************
                        ;2359
                        ;2360  =000
                        ;2361  IB.NM.FET:
                        ;2362  =010    ;010-----------------------------; Write*BWL
                        ;2363          WRITE VA READY, WRITE, IFORK,    ;
I 042, 4,002A,8004,1843 ;2364          VA_VA                            ;
                        ;2365
                        ;2366          ;011-----------------------------; Modify*BWL
                        ;2367          WRITE VA READY, WRITE.NOPAGE,    ;
I 043, 0,0024,8004,1845 ;2368          IFORK, VA_VA                     ;
                        ;2369
                        ;2370  =101    ;101-----------------------------; Read*QO
                        ;2371          VA_VA+4, OPVALID, READ,          ; Get second longword, we are done if
                        ;2372          IFORK IF QUAD,                   ; this is a Quad datam
I 045, 4,0014,E005,1056 ;2373          GOTO [IB.NM.FET.OCTAREAD]        ;
                        ;2374
                        ;2375          ;110-----------------------------; Write*QO
                        ;2376          VA_VA, WRITE VA READY, WRITE,    ; Write first longword, see if this is
                        ;2377          GOTO [IB.NM.FET.QO.WRITE],       ; Quad
I 046, 0,002A,8004,02A8 ;2378          BRANCH ON QUAD OR OCTA           ;
                        ;2379
                        ;2380          ;111-----------------------------; Modify*QO
                        ;2381          VA_VA+4, OPVALID, READ.NOPAGE,   ; Get second longword, are we Quad?
                        ;2382          BRANCH ON QUAD OR OCTA,          ;
I 047, 0,0018,E005,029B ;2383          GOTO [IB.NM.FET.QO.MODIFY]       ;
```

```
                        ;2384  .PAGE
                        ;2385
                        ;2386  ;*****************************************************************************
                        ;2387  ; Modify * (Quad + Octa) continues here.
                        ;2388  ;
                        ;2389  ; =1011 is Quad
                        ;2390  ; =1111 is Octa
                        ;2391  ;*****************************************************************************
                        ;2392
                        ;2393  =1011   ;1011-----------------------------;
                        ;2394  IB.NM.FET.QO.MODIFY:                      ;
                        ;2395          VA_VA-4, WRITE VA READY,          ; Ok, write first longword of dest.
                        ;2396          WRITE.NOPAGE,                     ; and link up with Quad. Write operand
I 09B, 0,0024,8005,80A8 ;2397          GOTO [IB.NM.FET.QO.WRITE]         ; below.
                        ;2398
                        ;2399          ;1111-----------------------------;
I 09F, 4,0018,E005,0051 ;2400          VA_VA+4, OPVALID, READ.NOPAGE     ; Fetch third longword
                        ;2401
                        ;2402          ;-------------------------------;
I 051, 4,0018,E005,0052 ;2403          VA_VA+4, OPVALID, READ.NOPAGE     ; Fetch fourth longword
                        ;2404
                        ;2405          ;-------------------------------;
I 052, 0,0002,8005,8053 ;2406          VA_VA-4                           ; Now bump VA down to first l.w.
                        ;2407
                        ;2408          ;-------------------------------;
I 053, 4,0002,8005,8054 ;2409          VA_VA-4                           ;
                        ;2410
                        ;2411          ;-------------------------------;
                        ;2412          VA_VA-4, WRITE VA READY,          ; Begin to write Octa dest. and link
                        ;2413          WRITE.NOPAGE,                     ; up with Octa dest. below
I 054, 4,0024,8005,80AF ;2414          GOTO [IB.NM.FET.OCTAWRITE]        ;
                        ;2415
                        ;2416
                        ;2417
                        ;2418  ;*****************************************************************************
                        ;2419  ; Write* (Quad + Octa) continues here
                        ;2420  ;
                        ;2421  ; =011 is Quad
                        ;2422  ; =111 is Octa
                        ;2423  ;*****************************************************************************
                        ;2424
                        ;2425  =1011   ;1011-----------------------------;
                        ;2426  IB.NM.FET.QO.WRITE:                       ;
                        ;2427          WRITE VA READY, WRITE.NOPAGE,     ; Write second longword, we are done, or
I 0AB, 0,0024,8005,18AF ;2428          VA_VA+4, IFORK                    ; 4th Octa longword
                        ;2429
                        ;2430          ;1111-----------------------------;
                        ;2431  IB.NM.FET.OCTAWRITE:                      ;
                        ;2432          WRITE VA READY, WRITE.NOPAGE,     ; Octa, write second longword
I 0AF, 0,0024,8005,0055 ;2433          VA_VA+4                           ;
                        ;2434
                        ;2435          ;-------------------------------;
                        ;2436          WRITE VA READY, WRITE.NOPAGE,     ;
                        ;2437          VA_VA+4,                          ; third, and link up with quad for 4th
I 055, 4,0024,8005,00AB ;2438          GOTO [IB.NM.FET.QO.WRITE]         ;
```

```
                        ;2439  .PAGE
                        ;2440
                        ;2441  ;******************************************************************************
                        ;2442  ; Read * Octa continues here
                        ;2443  ;******************************************************************************
                        ;2444
                        ;2445          ;--------------------------------;
                        ;2446  IB.NM.FET.OCTAREAD:                      ;
I 056, 4,0014,E005,0058 ;2447          VA_VA+4, OPVALID, READ           ;
                        ;2448
                        ;2449          ;--------------------------------;
I 058, 0,0014,E005,1859 ;2450          VA_VA+4, OPVALID, READ, IFORK    ; we are done!
```

```
                        ;2451  .PAGE   "      Indirect Address Fetch"
                        ;2452
                        ;2453  ;*****************************************************************
                        ;2454  ;
                        ;2455  ; Indirect specifiers come here to get the indirect address into VA and
                        ;2456  ; proceed with the fetch phase of the operand.  Specifically the specifiers
                        ;2457  ; which branch to here are: @(Rn)+, @x^D(Rn), @x^D(PC).
                        ;2458  ;
                        ;2459  ; Entry Point(s):
                        ;2460  ;       IB.NM.INDIRECT.FET:
                        ;2461  ;
                        ;2462  ; Input:
                        ;2463  ;       The memory request for the indirect address was fired off on
                        ;2464  ;       Ifork.
                        ;2465  ;
                        ;2466  ;*****************************************************************
                        ;2467
                        ;2468
                        ;2469          ;-------------------------------;
                        ;2470  IB.NM.INDIRECT.FET:                     ;
I 059, 0,00C2,801C,005A ;2471          STALL UNTIL IMD MDRESP          ; Wait for the data to become valid
                        ;2472
                        ;2473          ;-------------------------------;
                        ;2474          VA_IMD, ENABLE TRAP IF UNAL OP, ; Now start fetch phase of operand
                        ;2475          COND MEM REQ,                   ;
                        ;2476          ENABLE OPBUS PARITY CHECKING,   ;
                        ;2477          OPVALID IF A+RM*AL,             ;
                        ;2478          IFORK IF A+R*BWL,               ;
                        ;2479          BRANCH ON (Q+0) AND ACCESS,     ;
I 05A, 1,0000,C014,4540 ;2480          GOTO [IB.NM.FET]                ;
                        ;2481
                        ;2482
                        ;2483  ;*************************************************************************
                        ;2484  ;
                        ;2485  ; End of VENUS Ibox Microcode
                        ;2486  ;
                        ;2487  ;*************************************************************************
```

Cross Reference Listing - Field Names and Defined Values

```
        EXPLICIT                235 #
        INDEX                   228 #   2296    2305    2316    2326    2330    2333
        ISTREAM                 227 #   999     1006    1012    1015    1025    1029    1042    1048    1057    1063
                                1074    1085    1090    1100    1110    1120    1131    1141    1151    1159    1167    1176
                                1184    1192    1200    1209    1218    1222    1227    1232    1237    1242    1247    1250
                                1259    1266    1271    1278    1285    1291    1299    1307    1315    1323    1332    1340
                                1348    1356    1364    1372    1381    1389    1397    1402    1407    1417    1420    1423
                                1426    1429    1432    1436    1440    1444    1511    1522    1532    1545    1571    1580
                                1586    1594    1599    1605    1609    1612    1615    1618    1632    1637    1641    1646
                                1649    1659    1662    1666    1678    1682    1689    1693    1700    1703    1707    1715
                                1725    1729    1733    1783    1796    1804    1814    1821    1833    1841    1848    1887
                                1893    1901    1932    1936    1984    1988    1992    1997    2003    2007    2011    2021
                                2025    2039    2042    2046    2104    2108    2112    2116    2120    2124    2128    2137
                                2140    2197    2201    2232    2237    2243    2249    2253    2260    2364    2368    2373
                                2378    2383    2397    2400    2403    2406    2409    2414    2428    2433    2438    2447
                                2450    2471    2480
        PREVADD                 232 #   2069    2081    2162    2169
        PREVADD+1               230 #   1940    1945    1956    1960
        RLOG_UNWIND             234 #   1770    1777
GPRREG                          81 #
        R0                      82 #
        R1                      83 #
        R2                      84 #
        R3                      85 #
        R4                      86 #
        R5                      87 #
        R6                      88 #
        R7                      89 #
        R8                      90 #
        R9                      91 #
        RA                      92 #
        RB                      93 #
        RC                      94 #
        RD                      95 #
        RE                      96 #
        RF                      97 #
GPRREG10                        100 #
GPRREG32                        99 #
IB_HOLD                         400 #
        HOLD                    402 #
        NOHOLD                  401 #   999     1006    1012    1020    1025    1037    1042    1048    1057    1063
                                1074    1085    1090    1100    1110    1120    1131    1141    1151    1159    1167    1176
                                1184    1192    1200    1209    1218    1222    1227    1232    1237    1242    1247    1254
                                1259    1266    1271    1278    1285    1291    1299    1307    1315    1323    1332    1340
                                1348    1356    1364    1372    1381    1389    1397    1402    1407    1417    1420    1423
                                1426    1429    1432    1436    1440    1444    1511    1522    1532    1545    1571    1580
                                1586    1594    1599    1605    1609    1612    1615    1618    1632    1637    1641    1646
                                1649    1659    1662    1666    1678    1682    1689    1693    1700    1703    1707    1715
                                1725    1729    1733    1771    1779    1783    1796    1804    1814    1821    1833    1841
                                1848    1887    1893    1901    1932    1936    1942    1949    1956    1961    1984    1988
                                1992    1997    2003    2007    2011    2021    2025    2039    2042    2046    2076    2085
                                2104    2108    2112    2116    2120    2124    2128    2137    2140    2164    2172    2197
                                2201    2232    2237    2243    2249    2253    2260    2302    2311    2322    2327    2330
                                2334    2364    2368    2373    2378    2383    2397    2400    2403    2406    2409    2414
                                2428    2433    2438    2447    2450    2471    2480
```

Cross Reference Listing - Field Names and Defined Values

```
IFORK CTL                144 #
        IFORK            150 #  1222  1545  1605  1646  1804  1892  1900  1961  1984  2025
                         2039   2140  2363  2368  2428  2450
        IMMEDIATE        148 #  1051  1203  1212
        MEMORY           146 #  1036  1037  1072  1074  1083  1085  1098  1100  1108  1110
                         1118   1120  1129  1131  1139  1141  1149  1151  2075  2076  2299  2302
                         2308   2311  2319  2322  2478  2480
        NOP              145 #  999   1006  1012  1025  1042  1048  1063  1090  1159  1167
                         1176   1184  1192  1200  1227  1232  1237  1242  1247  1254  1259  1266
                         1271   1278  1285  1291  1299  1307  1315  1323  1332  1340  1348  1356
                         1364   1372  1381  1389  1397  1402  1407  1417  1420  1423  1426  1429
                         1432   1436  1440  1444  1511  1522  1532  1571  1580  1586  1594  1599
                         1609   1612  1615  1618  1632  1637  1641  1649  1659  1662  1666  1678
                         1682   1689  1693  1700  1703  1707  1715  1725  1729  1733  1771  1779
                         1783   1796  1814  1821  1833  1841  1848  1887  1932  1936  1956  1988
                         1992   1997  2007  2011  2021  2042  2046  2085  2104  2108  2112  2116
                         2124   2128  2137  2164  2172  2197  2201  2232  2237  2243  2249  2253
                         2260   2327  2330  2334  2378  2383  2397  2400  2403  2406  2409  2414
                         2433   2438  2447  2471
        QUAD             149 #  1941  1948  2002  2119  2372
        REGISTER         147 #  1017
MARK                     413 #
        OFF              414 #  999   1006  1012  1020  1025  1037  1042  1048  1057  1063
                         1074   1085  1090  1100  1110  1120  1131  1141  1151  1159  1167  1176
                         1184   1192  1200  1209  1218  1222  1227  1232  1237  1242  1247  1254
                         1259   1266  1271  1278  1285  1291  1299  1307  1315  1323  1332  1340
                         1348   1356  1364  1372  1381  1389  1397  1402  1407  1417  1420  1423
                         1426   1429  1432  1436  1440  1444  1511  1522  1532  1545  1571  1580
                         1586   1594  1599  1605  1609  1612  1615  1618  1632  1637  1641  1646
                         1649   1659  1662  1666  1678  1682  1689  1693  1700  1703  1707  1715
                         1725   1729  1733  1771  1779  1783  1796  1804  1814  1821  1833  1841
                         1848   1887  1893  1901  1932  1936  1942  1949  1956  1961  1984  1988
                         1992   1997  2003  2007  2011  2021  2025  2039  2042  2046  2076  2085
                         2104   2108  2112  2116  2120  2124  2128  2137  2140  2164  2172  2197
                         2201   2232  2237  2243  2249  2253  2260  2302  2311  2322  2327  2330
                         2334   2364  2368  2373  2378  2383  2397  2400  2403  2406  2409  2414
                         2428   2433  2438  2447  2450  2471  2480
        ON               415 #
MCF                      295 #
        CONDITIONAL      296 #  1029  1037  1067  1074  1079  1085  1093  1100  1103  1110
                         1113   1120  1124  1131  1134  1141  1144  1151  2070  2076  2296  2302
                         2308   2311  2319  2322  2475  2480
        IB.FILL.IBF      322 #  1803  1818  1890  1898
        IB.FILL.OP       318 #  1005  1796  1814  1832  1886
        NOP              299 #  999   1012  1020  1025  1042  1048  1057  1063  1090  1209
                         1218   1222  1227  1232  1237  1242  1247  1254  1259  1266  1271  1278
                         1285   1291  1299  1307  1315  1323  1332  1340  1348  1402  1407  1417
                         1420   1423  1426  1429  1432  1436  1440  1444  1545  1700  1703  1707
                         1771   1779  1783  1841  1848  1932  1936  1942  1949  1956  1961  1984
                         1988   1992  1997  2003  2007  2011  2021  2025  2039  2042  2046  2104
                         2108   2112  2116  2120  2124  2128  2137  2140  2164  2197  2201  2232
                         2237   2243  2249  2253  2260  2327  2330  2334  2406  2409  2471
        READ_V_NOPAGE    314 #  2381  2400  2403
        READ_V_NOPAGE.2ND 316 #  1580  1677  1687  1692  1728
        READ_V_RCHK      310 #  1155  1159  1163  1167  1172  1176  1181  1184  1189  1192
```

```
                        1197    1200    1351    1356    1360    1364    1368    1372    1377    1381    1385    1389
                        1393    1397    1598    1612    1618    1681    1725    1732    2083    2085    2170    2172
                        2371    2447    2450
    READ_V_RCHK.2ND     312 #   1511    1520    1530    1594    1604    1609    1615
    READ_V_WCHK         309 #
    WRITE_V_NOPAGE      300 #   1571    1585    1586    1632    1640    1641    1658    1659    1665    1666
                        1714    1715    2364    2367    2368    2378    2396    2397    2413    2414    2427    2428
                        2432    2433    2436    2438
    WRITE_V_NOPAGE.2ND  303 #   1637    1645    1649    1662
    WRITE_V_WCHK        305 #   1570    1571    1586    1632    1632    1641    1659    1666    1715    2363
                        2364    2368    2376    2378    2397    2414    2428    2433    2438
N0                      109 #
N10                     110 #
N20                     111 #
N30                     112 #
N32                     113 #
NEXT                    108 #
    IB.B.CONDCOMP.BR    999     1012    1884 #
    IB.B.JDOT           1782 #  1783    1821
    IB.B.JDOT.RAF       1025    1042    1090    1227    1232    1237    1242    1247    1259    1271    1402
                        1407    1930 #  1932    1936    1988    1992    2007    2011    2104    2108    2112    2116
                        2124    2128
    IB.B.UNCOND.BR      1006    1897 #
    IB.NM.AINC          1048    2068 #
    IB.NM.AINCDEF       1063    2080 #
    IB.NM.FET           1037    1074    1085    1100    1110    1120    1131    1141    1151    2076    2302
                        2311    2322    2361 #  2480
    IB.NM.FET.OCTAREAD  2373    2446 #
    IB.NM.FET.OCTAWRITE 2414    2431 #
    IB.NM.FET.QO.MODIFY 2383    2394 #
    IB.NM.FET.QO.WRITE  2377    2397    2426 #  2438
    IB.NM.IMM           1057    1983 #
    IB.NM.IMM.OCTA.READ 2003    2019 #
    IB.NM.IMM.QO.ASRC   1997    2038 #  2046
    IB.NM.INDEX         1254    1266    1299    1307    1315    1323    1332    1340    1348    2164    2201
                        2232    2237    2243    2295 #
    IB.NM.INDEX.AINC    1278    2160 #
    IB.NM.INDEX.AINCDEF 1291    2167 #
    IB.NM.INDEX.INDIRECT 1356   1364    1372    1381    1389    1397    2172    2196 #
    IB.NM.INDEX.LQO     2315 #  2327
    IB.NM.INDEX.QO      2325 #  2334
    IB.NM.INDEXPCPOP    1281    2229 #
    IB.NM.INDEXPCPOP.LQO 2240 # 2249
    IB.NM.INDEXPCPOP.QO 2246 #  2260
    IB.NM.INDIRECT.FET  1159    1167    1176    1184    1192    1200    2085    2470 #
    IB.NM.LIT           1209    1218    2102 #
    IB.NM.LIT.OCTAREAD  2120    2136 #
    IB.NM.RN            1020    1929 #
    IB.NM.RN.OCTA       1942    1949    1954 #
    IB.UT.F.LDCPC       1800 #  1833
    IB.UT.M.FREEZE      1847 #  1848
    IB.UT.RLOG          1775 #  1779
    IB.UT.U.LASTWRITE   1571    1586    1644 #  1666
    IB.UT.U.OCTAM       1693    1724 #
    IB.UT.U.QO.M        1686 #  1732
```

Cross Reference Listing - Field Names and Defined Values

Cross Reference Listing - Field Names and Defined Values

Cross Reference Listing - Macro Names

Cross Reference Listing - Expression Names

; Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
I 000	1586	1599	1571:	1560:	1612	1594:	1632:	1678:
I 008	1532:	1814:	1796:	1833:	1511:	1522:	1771:	1841:
I 010	1932=	1615	1618	1637	1936=	1942=	1641	1949=
I 018	1659	1662	1666	1605=	1682	1707	1715	1609=
I 020	1984=	1725	1988=	1992=	1997=	2003=	2007=	2011=
I 028	1729	1733	1804	1646=	1821	1848	1887	1649=
I 030	2104=	1893	2108=	2112=	2116=	2120=	2124=	2128=
I 038	1901	1956	1961	1689=	2021	2025	2046	1693=
I 040	2076	2085	2364=	2368=	2137	2373=	2378=	2383=
I 048	2140	2164	2172	1700=	2197	2201	2260	1703=
I 050	2334	2403	2406	2409	2414	2438	2447	1779=
I 058	2450	2471	2480					1783=
I 060 - 067 Unused								
I 068				2039=				2042=
I 070 - 077 Unused								
I 078	2232=	2237=	2243=	2249=	2253=			
I 080 - 087 Unused								
I 088	2302=	2311=	2322=	2327=	2330=			
I 090 - 097 Unused								
I 098				2397=				2400=
I 0A0 - 0A7 Unused								
I 0A8				2428=				2433=
I 0B0 - 0B7 Unused								
I 0B8								1545:
I 0C0	1192:	1141:	1176:	1120:	1159:	1100:	1063:	1048:
I 0C8	1085:	1037:	1020:	1222:	1209:	1417	1420	999:
I 0D0	1200:	1151:	1184:	1131:	1167:	1110:	1074:	1057:
I 0D8	1090:	1042:	1025:	1227:	1218:	1423	1426	1429
I 0E0	1389:	1340:	1372:	1323:	1356:	1307:	1291:	1278:
I 0E8	1266:	1254:	1242:	1232:	1402:	1432	1436	1006:
I 0F0	1397:	1348:	1381:	1332:	1364:	1315:	1299:	1285:
I 0F8	1271:	1259:	1247:	1237:	1407:	1440	1444	1012:

	Words not in bounds
IBOXDEF	0
IBOXMACRO	0
IBOXCODE	173
Used	173
Remaining	

Total microwords used in memory I: 173
Total microwords remaining in memory I: 0
Highest address used in memory I: 0FF (hex)

```
; The files used in this assembly are:
IBOXDEF.MIC
IBOXMACRO.MIC
IBOXCODE.MIC


Pass 1 warnings:     0   Pass 2 warnings:     0
        Unsplit binary Lines:     0
Pass 1 errors:       0   Pass 2 errors:       0
```