```
SSSSSSSSSSSS  YYY        YYY  SSSSSSSSSSSS  LLL                    000000000      AAAAAAAAA
SSSSSSSSSSSS  YYY        YYY  SSSSSSSSSSSS  LLL                    000000000      AAAAAAAAA
SSSSSSSSSSSS  YYY        YYY  SSSSSSSSSSSS  LLL                    000000000      AAAAAAAAA
SSS           YYY        YYY  SSS           LLL                  000      000   AAA       AAA
SSS           YYY        YYY  SSS           LLL                  000      000   AAA       AAA
SSS           YYY        YYY  SSS           LLL                  000      000   AAA       AAA
SSS              YYY  YYY     SSS           LLL                  000      000   AAA       AAA
SSS              YYY  YYY     SSS           LLL                  000      000   AAA       AAA
SSS              YYY  YYY     SSS           LLL                  000      000   AAA       AAA
  SSSSSSSS          YYY          SSSSSSSS   LLL                  000      000   AAA       AAA
  SSSSSSSS          YYY          SSSSSSSS   LLL                  000      000   AAA       AAA
  SSSSSSSS          YYY          SSSSSSSS   LLL                  000      000   AAA       AAA
        SSS         YYY                SSS  LLL                  000      000   AAAAAAAAAAAAAAA
        SSS         YYY                SSS  LLL                  000      000   AAAAAAAAAAAAAAA
        SSS         YYY                SSS  LLL                  000      000   AAA       AAA
        SSS         YYY                SSS  LLL                  000      000   AAA       AAA
        SSS         YYY                SSS  LLL                  000      000   AAA       AAA
        SSS         YYY                SSS  LLL                  000      000   AAA       AAA
SSSSSSSSSSSS        YYY  SSSSSSSSSSSS  LLLLLLLLLLLLLLL          000000000      AAA       AAA
SSSSSSSSSSSS        YYY  SSSSSSSSSSSS  LLLLLLLLLLLLLLL          000000000      AAA       AAA
SSSSSSSSSSSS        YYY  SSSSSSSSSSSS  LLLLLLLLLLLLLLL          000000000      AAA       AAA
```

```
MM        MM    CCCCCCCC   FFFFFFFFFF   77777777       999999        000000
MM        MM    CCCCCCCC   FFFFFFFFFF   77777777       999999        000000
MMMM    MMMM   CC          FF                  77   99        99   00        00
MMMM    MMMM   CC          FF                  77   99        99   00        00
MM  MM    MM   CC          FF                  77   99        99   00      0000
MM   MM   MM   CC          FF                  77   99        99   00      0000
MM        MM   CC          FFFFFFFF           77    99999999   00   00    00
MM        MM   CC          FFFFFFFF           77    99999999   00   00    00
MM        MM   CC          FF               77             99   0000      00
MM        MM   CC          FF               77             99   0000      00
MM        MM   CC          FF             77               99   00        00
MM        MM   CC          FF             77               99   00        00
MM        MM    CCCCCCCC   FF            77            999999        000000
MM        MM    CCCCCCCC   FF            77            999999        000000
```

```
LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II             SS
LL               II             SS
LL               II             SS
LL               II             SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

```
 1    {
 2    { Version:        'V04-000'
 3    {
 4    {*********************************************************************
 5    {*                                                                   *
 6    {*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                         *
 7    {*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.          *
 8    {*   ALL RIGHTS RESERVED.                                            *
 9    {*                                                                   *
10    {*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
11    {*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH LICENSE  AND WITH THE   *
12    {*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
13    {*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
14    {*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
15    {*   TRANSFERRED.                                                     *
16    {*                                                                   *
17    {*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
18    {*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
19    {*   CORPORATION.                                                     *
20    {*                                                                   *
21    {*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
22    {*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.          *
23    {*                                                                   *
24    {*                                                                   *
25    {*********************************************************************
26
27    {++
28    {
29    { FACILITY:  VAX/VMS CPU-dependent Code Macro Libraries
30    {
31    { ABSTRACT:
32    {
33    {      This file contains the SDL source for 11/790 machine check frame
34    {      definitions.
35    {
36    { ENVIRONMENT:
37    {
38    {      n/a
39    {
40    {--
41    {
42    {
43    { AUTHOR: Wayne Cardoza        CREATION DATE:  01-Nov-1982
44    {
45    { MODIFIED BY:
46    {
47    {      V03-011 WMC0008          Wayne Cardoza          23-Jul-1984
48    {              Still more spec changes.
49    {
50    {      V03-010 WMC0007          Wayne Cardoza          08-Jul-1984
51    {              Assorted spec changes.
52    {
53    {      V03-009 WMC0006          Wayne Cardoza          30-May-1983
54    {              Minor changes and corrections.
55    {
56    {      V03-008 WMC0005          Wayne Cardoza          22-FEB-1983
57    {              Spec changes to MSTAT1, MSTAT2, MDECC
```

```
58    {
59    {         V03-007 WMC0004           Wayne Cardoza         08-Feb-1983
60    {             Rearrange EHSR
61    {
62    {         V03-006 WMC0003           Wayne Cardoza         20-Dec-1982
63    {             Separate PAMM code from cache bit
64    {
65    {         V03-005 WMC0002           Wayne Cardoza         24-Nov-1982
66    {             Add the VMS type code definitions.
67    {
68    {         V03-004 WMC0001           Wayne Cardoza         14-Nov-1982
69    {             Changes to MDECC, MSTAT1
70    {--
71
```

```
72    module $MCF790DEF;
73
74    aggregate MCF790 structure prefix MCF790$;
75        SIZE longword unsigned;                /* size in bytes of frame
76        EHSR_OVERLAY union;         /* error handling status register
77            EHSR longword unsigned; /* entire register
78            EHSR_BITS structure;
79                EHSR_OVERLAY_1 union;
80                    MCHK_CODE byte unsigned; /* VMS puts a code here
81                    EHSR_BITS_1 structure;
82                        SERV_TYPE bitfield mask length 3; /* VMS service type
83                        FILLTA bitfield length 1 fill prefix MCF790 tag $$;
84                        RSRC_REM bitfield mask; /* Resource removed from service
85                        SBIA bitfield mask; /* full SBIA log follows
86                        SBIA_ERR bitfield mask; /* SBIA error summary included
87                        MBOX_1D bitfield mask; /* MBOX 1D error included
88                    end EHSR_BITS_1;
89                end EHSR_OVERLAY_1;
90                TRAP_VEC bitfield mask length 8; /* trap vector
91                FILLT bitfield length 1 fill prefix MCF790 tag $$;
92                AUTO_SHUT bitfield mask;  /* Severe error flag
93                MEAR_SAV bitfield mask;   /* meaningful to microcode
94                ICS bitfield mask;        /* ICS correction
95                IDRAM bitfield mask;      /* IDRAM correction
96                FDRAM bitfield mask;      /* FDRAM correction
97                FBACS bitfield mask;      /* FBACS correction
98                FBMCS bitfield mask;      /* FBMCS correction
99                IBOX_GPR bitfield mask;   /* IBOX GPR correction
100               EBOX_SPBA bitfield mask;  /* EBOX SP B to A
101               EBOX_SPAB bitfield mask;  /* EBOX SP A to B
102               FBOX_SP bitfield mask;    /* FBOX SP correction
103               FBOX bitfield mask;       /* FBOX service
104               VMS_ENT bitfield mask;    /* VMS entered
105               EHM_ENT bitfield mask;    /* EHM entered
106               MBOX bitfield mask;       /* MBOX service
107           end EHSR_BITS;
108       end EHSR_OVERLAY;
109       EVMQSAV longword unsigned; /* virtual address - EBOX port requests
110       EBCS_OVERLAY union;         /* EBOX control status register
111           EBCS longword unsigned; /* entire register
112           EBCS_BITS structure;
113               EBCS_OVERLAY_1 union;
114                   EBCS_BITS_2 structure;
115                       FILL2 bitfield fill prefix MCF790 tag $$;
116                       IO_RD bitfield mask;         /* IO read abort
```

```
117                    MEM_WRT bitfield mask;          /* memory write abort
118                    STA_MOD bitfield mask;          /* state modified abort
119                    EB_ABT bitfield mask;           /* EBOX abort
120                    FILL3 bitfield length 3 fill prefix MCF790 tag $$;
121                    WBUS_CHK bitfield mask;         /* WBUS to EDP error
122                    EDP_PE bitfield mask;           /* EBOX data path parity error
123                    USTK_PE bitfield mask;          /* EBOX microstack
124                    ECS_PE bitfield mask;           /* EBOX control store
125                    EMCR_PE bitfield mask;          /* EBOX memory control RAM
126                    IBOX_ERR bitfield mask;         /* IBOX hardware error
127                    MBOX_INT bitfield mask;         /* MBOX interrupt request
128                    MBOX_FE bitfield mask;          /* MBOX fatal error
```

```
129                end EBCS_BITS_2;
130                EBCS_BITS_3 structure;
131                    FILL2A bitfield fill prefix MCF790 tag $$;
132                    ABORTS bitfield mask length 4;
133                    FILL3A bitfield length 3 fill prefix MCF790 tag $$;
134                    DIAG_ERR bitfield mask;         /* diagnostic error flag
135                end EBCS_BITS_3;
136            end EBCS_OVERLAY_1;
137            FILL4 bitfield length 4 fill prefix MCF790 tag $$;
138            PME bitfield mask;                  /* performance measurement enable
139            FILL5 bitfield length 6 fill prefix MCF790 tag $$;
140            ICS_EF bitfield mask;          /* IBOX control store error
141            IDRAM_EF bitfield mask;        /* IBOX dispatch RAM error
142            FBMCS_EF bitfield mask;        /* FBOX FBM control store error
143            FBACS_EF bitfield mask;        /* FBOX FBA control store error
144            FDRAM_EF bitfield mask;        /* FBOX dispatch RAM error
145        end EBCS_BITS;
146    end EBCS_OVERLAY;
147    EDPSR_OVERLAY union;         /* EBOX data path status register
148        EDPSR longword unsigned;/* entire register
149        EDPSR_BITS structure;
150            B_RAM_PE bitfield mask;        /* scratchpad to BMUX error
151            A_WBUS_PE bitfield mask;       /* WBUS to AMUX error
152            A_RAM_PE bitfield mask;        /* scratchpad to AMUX error
153            OPER_CHK bitfield mask;        /* operand parity error
154            FILL51 bitfield fill prefix MCF790 tag $$;
155            RSLT_CHK bitfield mask;        /* result parity error
156            B_OPBUS bitfield mask;         /* OPBUS to BMUX error
157            B_WBUS bitfield mask;          /* WBUS to BMUX error
158            EDP_MISC bitfield mask;        /* misc source parity error
159            FILL6 bitfield length 2 fill prefix MCF790 tag $$;
160            WREG bitfield mask;            /* W register parity error
161            VMQ_BYTE bitfield mask length 4; /* VMQ byte in error
162            FILL7 bitfield length 8 fill prefix MCF790 tag $$;
163            AMX_BYTE bitfield mask length 4; /* AMUX byte in error
164            BMX_BYTE bitfield mask length 4; /* BMUX byte in error
165        end EDPSR_BITS;
166    end EDPSR_OVERLAY;
167    CSLINT_OVERLAY union;        /* console/interrupt register
168        CSLINT longword unsigned; /* entire register
169        CSLINT_BITS structure;
170            CADR bitfield mask length 6; /* console bus address
171            CWRT bitfield mask;           /* console bus write
172            CCLK bitfield mask;           /* console bus clock
173            CDAT bitfield mask length 8; /* console bus data
174            IPR bitfield mask length 4; /* interrupt priority request level
175            INT_SRC bitfield mask;        /* IPR due to internal source
176            IOA bitfield mask length 2; /* I/O adapter with highest IPR
```

```
177            CSL_TTX bitfield mask;          /* console terminal transmit
178            CSL_TRX bitfield mask;          /* console terminal receive
179            CSL_RL bitfield mask;           /* console RL
180            INT_TMR bitfield mask;          /* interval timer interrupt
181            INT_MBOX bitfield mask;         /* MBOX interrupt
182            CPU_PF bitfield mask;           /* CPU powerfail interrupt
183            CSL_HP bitfield mask;           /* console halt pending
184        end CSLINT_BITS;
185    end CSLINT_OVERLAY;
186    IBESR_OVERLAY union:          /* IBOX error/status register
```

```
187        IBESR longword unsigned; /* entire register
188        IBESR_BITS structure;
189            FILL8 bitfield length 8 fill prefix MCF790 tag $$;
190            UOP_SEL bitfield mask length 2; /* OP BUS data source
191            SRC_IMD bitfield mask;          /* OP BUS source was IMD
192            UTPR bitfield mask length 3; /* processor port causing microtrap
193            FILL9 bitfield length 7 fill prefix MCF790 tag $$;
194            ICS_PE bitfield mask;           /* IBOX control store parity error
195            IDRAM_PE bitfield mask;         /* DRAM
196            IAMUX_PE bitfield mask;         /* AMUX whren GPR selected
197            RLOG_PE bitfield mask;          /* unwinding RLOG
198            IBUF_PE bitfield mask;          /* error on byte-1, byte-0, or R-mode finder
199            IBMUX_PE bitfield mask;         /* output of ALU BMUX
200            RSV_MODE bitfield mask;         /* reserved mode
201            IWBUS_PE bitfield mask;         /* WBUS error detected by IBOX
202            IAMUX_EC bitfield mask length 2; /*
203        end IBESR_BITS;
204    end IBESR_OVERLAY;
205    EBXWD1 longword unsigned;      /* EBOX write data 1
206    EBXWD2 longword unsigned;      /* EBOX write data 2
207    IVASAV longword unsigned;      /* virtual address for OP port requests
208    VIBASAV longword unsigned;     /* virtual address of next IBUF port request
209    ESASAV longword unsigned;      /* PC during EBOX execution and result storage
210    ISASAV longword unsigned;      /* PC of instruction OP port working on
211    CPC longword unsigned;         /* PC of instruction evaluated in IBUFFER
212    MSTAT1_OVERLAY union;          /* MBOX status register 1
213        MSTAT1 longword unsigned;           /* entire register
214        MSTAT1_BITS structure;
215            CSH_DAT_BW bitfield mask;    /* datapath parity error on byte write
216            ARR_CYCL bitfield mask;      /* error detected on array refill to cache
217            CSH_ERR bitfield mask;       /* indicates which cache had the error
218            CSH_DAT_NBW bitfield mask;   /* datapath parity error, non byte write
219            WRT_DAT_PE bitfield mask length 4; /* MDBUS parity error on write data
220            TB_TAG_PE bitfield mask;     /* error on address tag
221            TB_A_PE bitfield mask;       /* error on PTE
222            TB_B_PE bitfield mask;       /* error on PTE
223            TB_VAL_PE bitfield mask;     /* error in valid bit
224            CSH_HIT bitfield mask length 4; /* cache hit/miss history
225            AB_ADPT bitfield mask length 2; /* ABUS adapter in error
226            AB_CYCL bitfield mask;       /* ABUS cycle in error
227            AB_ADR_PE bitfield mask;     /* ABUS physical address in error
228            AB_CM_PE bitfield mask;      /* ABUS cntrl/mask parity error
229            AB_DAT_PE bitfield mask;     /* ABUS data parity error
230            CPR_PE_A bitfield mask;      /* cycle parameter RAM error (A)
231            CPR_PE_B bitfield mask;      /* cycle parameter RAM error (B)
232            WDCNT bitfield mask length 2; /* longword in error
233            CYCLE_TYP bitfield mask length 4; /* MBOX cycle type
234            DEST_CP bitfield mask length 2; /* port being serviced
235        end MSTAT1_BITS;
236    end MSTAT1_OVERLAY;
```

```
237        MSTAT2_OVERLAY union;          /* MBOX status register 2
238           MSTAT2 longword unsigned; /* entire register
239           MSTAT2_BITS structure;
240              FILL95 bitfield length 1 fill prefix MCF790 tag $$;
241              MBOX_LCK bitfield mask;     /* error while lock asserted
242              CP_IO_BUF bitfield mask;    /* error on CPU to IO request
243              NXM bitfield mask;          /* non-existent memory
244              CSH_W bitfield mask;        /* selected cache entry was modified
```

```
245              CSH_TAG_W bitfield mask;    /* error in cache written bit
246              CSH_TAG_PE bitfield mask;   /* error in cache tag
247              MUL_ERR bitfield mask;      /* multiple MBOX errors
248              SBIA_STAT bitfield mask length 6; /* SBIA diagnostic status
249              AB_BAD_DAT bitfield mask;   /* ABUS bad data flag received
250              SBIA_CPBW bitfield mask;    /* SBIA error was on CP byte write
251              PAMM_DATA bitfield mask length 4; /* PAMM code
252              PAMM_CACHE bitfield mask;   /* PAMM cache disable bit
253           end MSTAT2_BITS;
254        end MSTAT2_OVERLAY;
255        MDECC_OVERLAY union;          /* MBOX data ECC register
256           MDECC longword unsigned; /* entire register
257           MDECC_BITS structure;
258              ECC_DIAG bitfield mask length 8; /* force errors
259              FILL115 bitfield length 1 fill prefix MCF790 tag $$;
260              SYNDRM bitfield mask length 6; /* error data syndrome
261              PAR_INV bitfield mask;      /* indicates parity is being inverted
262              FILL11 bitfield length 3 fill prefix MCF790 tag $$;
263              ADR_PE bitfield mask;       /* data address parity error
264              DBL_BIT bitfield mask;      /* double bit error
265              SNG_ERR bitfield mask;      /* single bit error
266              BAD_DATA bitfield mask;     /* bad data flag
267              DATA_MUL bitfield mask;     /* multiple errors
268           end MDECC_BITS;
269        end MDECC_OVERLAY;
270        MERG longword unsigned;       /* MBOX error generator register
271        CSHCTL_OVERLAY union;         /* MBOX cache control register
272           CSHCTL longword unsigned; /* entire register
273           CSHCTL_BITS structure;
274              CSH_0_ENB bitfield mask;    /* cache 0 enable
275              CSH_1_ENB bitfield mask;    /* cache 1 enable
276              FRC_HIT bitfield mask;      /* force cache hit
277              FRC_MISS bitfield mask;     /* force cache miss
278           end CSHCTL_BITS;
279        end CSHCTL_OVERLAY;
280        MEDR longword unsigned;       /* data word used during error
281        MEAR longword unsigned;       /* physical address in latch during error
282        FBXERR_OVERLAY union;         /* FBOX error register
283           FBXERR longword unsigned;     /* entire register
284           FBXERR_BITS structure;
285              FBOX_ERR bitfield;          /* There is an error - rest of bits valid
286              FILL12 bitfield length 1 fill prefix MCF790 tag $$;
287              TEST bitfield mask;         /* error during self test
288              FILL13 bitfield length 11 fill prefix MCF790 tag $$;
289              DATA_TYP bitfield mask length 2; /* data type during error
290              FILL14 bitfield length 1 fill prefix MCF790 tag $$;
291              FBOX_GPR bitfield mask;     /* error reading scratchpad
292              FBOX_SLF bitfield mask;     /* error during self test
293              FBOX_DRAM bitfield mask;    /* DRAM parity error
294              FBOX_FBA_CS bitfield mask;  /* error in adder control store
295              FBOX_FBM_CS bitfield mask;  /* error in multiplier control store
296           end FBXERR_BITS;
```

```
297        end FBXERR_OVERLAY;
298        CSES longword unsigned;        /* control store error status register
299        PC longword unsigned;
300        PSL longword unsigned;
301
302                                       /* MBOX cycle types
```

```
303        constant(
304            NOP,
305            READ_REG,                       /* read register
306            WRITE_REG,                      /* write register
307            WRITE_BAK,                      /* write back
308            ABUS_ORT,                       /* ABUS array write
309            DATA_COR,                       /* data correction
310            CLR_CSH,                        /* clear cache
311            TB_PROBE,                       /* TB probe
312            ABUS,                           /* ABUS
313            CP_REFL,                        /* CP refill
314            INVAL_TB,                       /* invalidate TB
315            TB_CYCLE,                       /* TB cycle
316            CP_BYT_WRT,                     /* CP byte write
317            CP_WRT,                         /* CP write
318            CP_READ,                        /* CP read
319            ABUS_REFL                       /* ABUS refill
320        ) equals 0 increment 1 prefix MCF790 tag $C;
321
322                                       /* DEST CP (port) codes
323        constant(
324            IBF_PORT_0,                     /* IBUF port
325            OP_PORT,                        /* OP fetch port
326            EBOX_PORT,                      /* EBOX port
327            IBF_PORT_3                      /* IBUF port
328        ) equals 0 increment 1 prefix MCF790 tag $C;
329
330                                       /* VMS machine check service codes
331        constant(
332            FBOX,                           /* FBOX
333            EBOX,                           /* EBOX
334            IBOX,                           /* IBOX
335            MBOX_FE                         /* MBOX fatal error
336        ) equals 1 increment 1 prefix MCF790 tag $C;
337
338    end MCF790;
339    end_module $MCF790DEF;
```
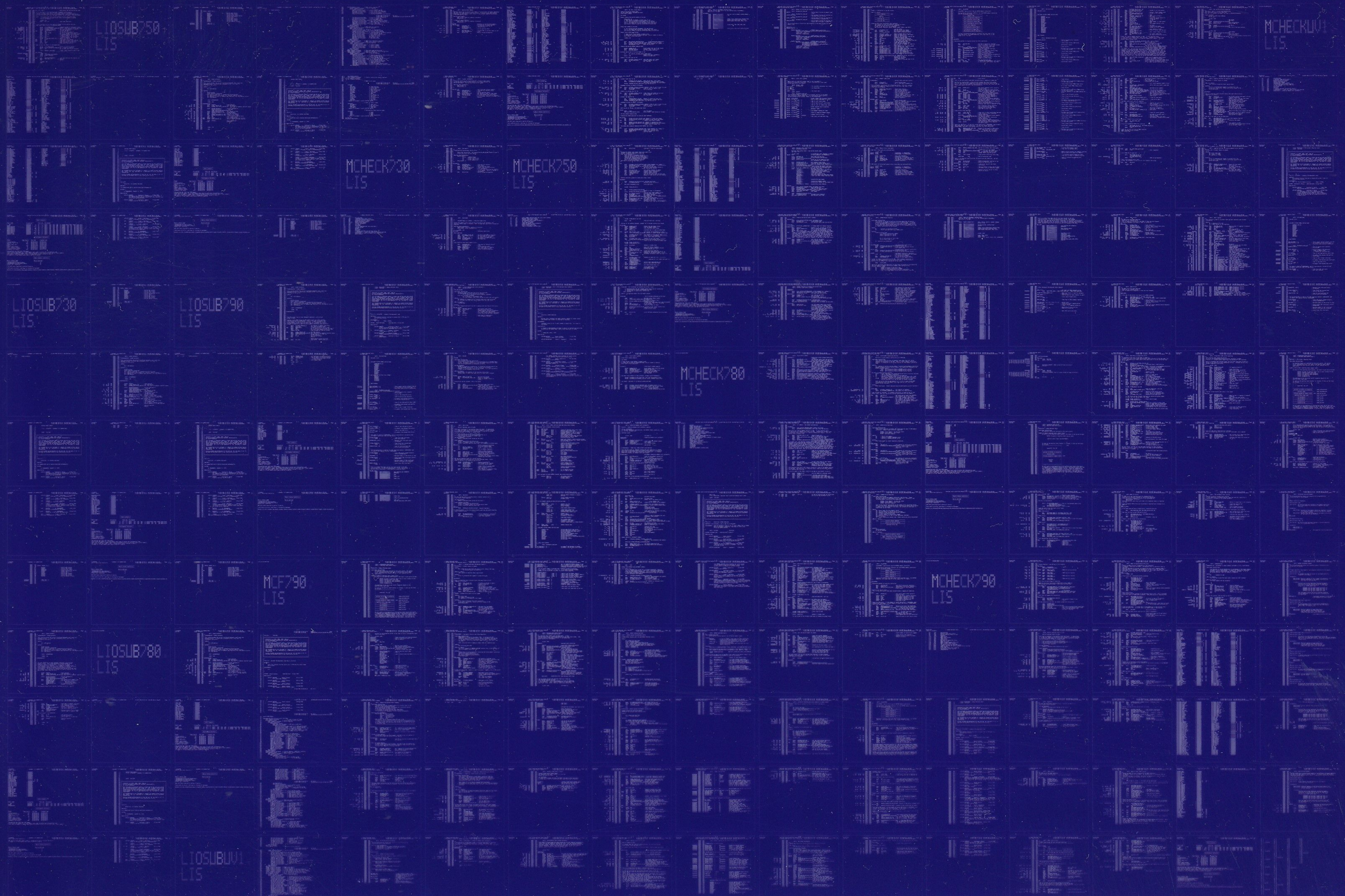
LIOSUB750
LIS

MCHECKUV1
LIS

MCHECK730
LIS

MCHECK750
LIS

LIOSUB730
LIS

LIOSUB790
LIS

MCHECK780
LIS

MCF790
LIS

MCHECK790
LIS

LIOSUB780
LIS

LIOSUBUV1
LIS