```
/*
 * asmdef.h
 *
 * Assembler Hacks
 *
 * Jeffrey Mogul                    15 May 1981
 *      after code written by Luis Trabb-Pardo
 */

/* branch to a label */
#define branch(a) asm (" .text;");\
                  asm("  jmp a")

/* push a word (short) */
#define wpush(a) asm (" .text");\
        asm("   movw a,sp@-")

/* push a register */
#define rpush(a) asm (" .text");\
        asm("   movl a,sp@-")

/* pop a register */
#define rpop(a) asm ("  .text");\
        asm("   movl sp@+,a")

/* define an entry in a vector of longs (e.g., exception vector) */
#define excv(a) asm("   .text")\
                asm("   .long a")

/*
 * exclink - used to push a word describing an exception,
 *      then jump to ".enter".  The word contains two bytes,
 *      which form a two-character message.
 */
#define exclink(n,a,b) asm("    .text");\
        asm("n: movw    #'a*256.+'b,sp@-");\
        asm("   jra     .enter")

/*
 * exclinkn - like exclink but pushes a numeric code
 */
#define exclinkn(n,x) asm("     .text");\
        asm("n: movw    #'x,sp@-");\
        asm("   jra     .enter")

/* return from interrupt/exception */
#define endsrv asm("    rte")

/* label a point for assembler jumps */
#define label(a)        asm("a:")

/* label a point globally for assembler jumps */
#define LabelGbl(a)     asm("   .globl a"); asm ("a:");

/* set processor interrupt level */
#define setINTLV(a)\
 {extern unsigned short sr; sr= (unsigned short)(0x2000+(a<<8));}

/* Register save area allocation */
#define _reg_d0         0
#define _reg_a0         8
#define _reg_ss         15
```

```
#define _reg_us         16
#define _reg_sr         17
#define _reg_pc         18
#define _reg_MX         18
asm("_disp_us=64.");

/* Auxiliary functions to save and restore things */
/* Allocation of saved registers on the stack:
        sp@(72.)...     PC
        sp@(70.)...     SR
        sp@(68.)...      0
        sp@(64.)...     US
        sp@(62.)...     SS
        sp@(58.)...     A6
                .......
        sp@(32.)...     A0
        sp@(28.)...     D7
                .......
        sp@........     D0
*/

/* The Monitor sees the Register Save Area as 19 long parameters:
        monitor(r_pc, r_sr, r_us, r_ss,
                r_a6, r_a5, r_a4, r_a3, r_a2, r_a1, r_a0,
                r_d7, r_d6, r_d5, r_d4, r_d3, r_d2, r_d1, r_d0)
*/

/* Storing the exception cause after an exception: */
#define ExcCause(a)  asm("       .text");\
        asm("   movw    #a.,sp@-");\

/* Saving registers after an exception: */
#define saveregs asm("  .text");\
        asm("   subql   #4.,sp");\
        asm("   moveml  #/FFFF,sp@-");\
        asm("   movl    usp,a0");\
        asm("   movl    a0,sp@(_disp_us)")

/* Restoring registers after a call to the monitor: */
#define restregs        asm("    .text");\
        asm("   movl    sp@(_disp_us),a0");\
        asm("   movl    a0,usp");\
        asm("   moveml  sp@+,#/FFFF");\
        asm("   addql   #6.,sp")

/*
 * Save Access Address etc. after BusErr or AdrErr; the assumption
 * is that 0 contains the address of the global data region, and
 * the first two longs of global data store the top two longs from
 * the exception stack frame.
 */
#define SAVEACCESS      \
        asm("   movl    a0,4.");        /* temporarily store a0 */\
        asm("   movl    0.,a0");        /* get global base pointer */\
        asm("   movl    sp@+,a0@+");    /* pop first long */\
        asm("   movl    sp@+,a0@");     /* pop second long */\
        asm("   movl    4.,a0");        /* restore a0 */

/*
 * Flush Access Address etc. after BusErr or AdrErr; we don't
 * need them and cannot rte until they are gone
 */
#define FLUSHACCESS     asm("   addql #8.,sp"); /* pop two longs */
```

```
/*
 * Opcode definitions -
 *      The monitor occasionally stuffs some opcodes into
 *      memory.  These are defined here.
 */

#define OPC_2NOP        0x4E714E71      /* TWO nops */
#define OPC_NOP_RTS     0x4E714E75      /* nop followed by an rts */
#define OPC_TRAP1       0x4E41          /* trap 1 */
#define OPC_2UN1        0xFFFFFFFF      /* TWO "unimplemented 1111"s */
```

```
/*
 * buserr.h
 *
 * definitions for structure showing details of Bus Error
 * or Address Error on MC68000
 *
 * Jeffrey Mogul                15 June 1981
 */

struct BusErrInfo {
        unsigned int :11;               /* padding */
        unsigned int Read:1;            /* Read = 1, Write = 0; */
        unsigned int Instr:1;           /* FALSE if reference was a fetch */
        unsigned int FuncCode:3;        /* "Function Code */
        unsigned short AccAddrLo;       /* Access Address */
        unsigned short AccAddrHi;
        unsigned short InstrReg;        /* Instruction Register */
};

/*
 * values in FuncCode field
 */
#define BERR_USERDATA   1
#define BERR_USERPROG   2
#define BERR_SUPERDATA  5
#define BERR_SUPERPROG  6
#define BERR_INTACK     7
```

```
/*
 * globram.h
 *
 * [illegible]
 *
 * Jeffrey Mogul        15 May 1981
 *      created from code written by Luis Trabb-Pardo
 *
 * This defines a data structure that lives in RAM and is
 * used by the ROM monitor for its own purposes.
 *
 * There are two ways of accessing these data; the most
 * straightforward way is to say \140\140GlobPtr->xxx''; however,
 * if a single routine makes more than 3 accesses a global datum
 * (measured staticly), code size can be improved by first
 * loading GlobPtr into a register -- i.e.,
 *      register GlobDes *gp = GlobPtr;
 */

#include "sunmon.h"

/*
 * things whose size does not actually matter are made into
 * ints; this removes the "extw; extl" sequences seen all over.
 */
typedef struct {
        union  {        /* MUST be first in Global Data */
            struct      {       /* how bus error info appears on stack */
                long os_l1;
                long os_l2;
            } OnStack;
            struct BusErrInfo Data;     /* structured according to contents */
        } BE;
        char linbuf[BUFSIZE+2]; /* data input buffer */
        char *lineptr;          /* next "unseen" char in linbuf */
        int linesize;           /* size of input line in linbuf */

        int recount;            /* count of S-records loaded */

        int debounce;           /* used in debounced Abort switch */

        short BreakVal;         /* instruction replaced by Break trap */
        word *BreakAdx;         /* address of current BPT */
        int BrkInPrg;           /* flag: true if we are handling BPT */

        int RefrCnt;            /* pseudo-clock counter */

        long MemorySize;        /* size of memory in bytes */
        int CurContext;         /* current context register contents */
        char EndTransp;         /* escape character for transparent mode */
        int ConChan;            /* Control channel (A_chan or B_chan) */
        int EchoOn;             /* true if input should be echoed */
        int FrameBuffer;        /* true if frame buffer is console */
                } GlobDes;

#define GlobBase (*((GlobDes **)0))     /* set to point at base of
                                         * monitor global data
                                         */

#define GlobPtr ((GlobDes *)STRTDATA)
```

```
/*
 * m68vectors.h
 *
 * Defines addresses for MC68000 exception vectors
 *      Addresses are of type "long *"
 *
 * Jeffrey Mogul        Stanford University     22 April 1981
 *
 */

#define EVEC_RESETSSP   ((long *)0x000)
#define EVEC_RESETPC    ((long *)0x004)
#define EVEC_BUSERR     ((long *)0x008)
#define EVEC_ADDERR     ((long *)0x00C)
#define EVEC_ILLINST    ((long *)0x010)
#define EVEC_DIVZERO    ((long *)0x014)
#define EVEC_CHK        ((long *)0x018)
#define EVEC_TRAPV      ((long *)0x01C)
#define EVEC_PRIV       ((long *)0x020)
#define EVEC_TRACE      ((long *)0x024)
#define EVEC_LINE1010   ((long *)0x028)
#define EVEC_LINE1111   ((long *)0x02C)

/*
 * vectors 0x30 - 0x5C are "reserved to Motorola")
 */

#define EVEC_SPURINT    ((long *)0x060)
#define EVEC_LEVEL1     ((long *)0x064)
#define EVEC_LEVEL2     ((long *)0x068)
#define EVEC_LEVEL3     ((long *)0x06C)
#define EVEC_LEVEL4     ((long *)0x070)
#define EVEC_LEVEL5     ((long *)0x074)
#define EVEC_LEVEL6     ((long *)0x078)
#define EVEC_LEVEL7     ((long *)0x07C)

#define EVEC_TRAP0      ((long *)0x080)
#define EVEC_TRAP1      ((long *)0x084)
#define EVEC_TRAP2      ((long *)0x088)
#define EVEC_TRAP3      ((long *)0x08C)
#define EVEC_TRAP4      ((long *)0x090)
#define EVEC_TRAP5      ((long *)0x094)
#define EVEC_TRAP6      ((long *)0x098)
#define EVEC_TRAP7      ((long *)0x09C)
#define EVEC_TRAP8      ((long *)0x0A0)
#define EVEC_TRAP9      ((long *)0x0A4)
#define EVEC_TRAPA      ((long *)0x0A8)
#define EVEC_TRAPB      ((long *)0x0AC)
#define EVEC_TRAPC      ((long *)0x0B0)
#define EVEC_TRAPD      ((long *)0x0B4)
#define EVEC_TRAPE      ((long *)0x0B8)
#define EVEC_TRAPF      ((long *)0x0BC)

/*
 * vectors 0xC0 - 0xFC are "reserved to Motorola"
 */

/*
 * vectors 0x100 - 0x3FC are User Interrupt Vectors
 */
#define EVEC_USERINT(v) ((long *)(0x100 + ((v)<<2)))

#define EVEC_LASTVEC    ((long *)(0x3FC))
```

```
#define EVEC_AFTER      ((long *)(0x400))
#define NUM_EVECS       256     /* number of exception vectors */
```

```
#include "nec7201.h"

/*  Address definitions for the nec7201                    */

#define A_chan  0
#define B_chan  4

#define ContReg(ch)     *(char*)(0x600002+ch/**/_chan)
#define StatReg(ch)     *(char*)(0x600002+ch/**/_chan)  /* Read only */
#define TxHldReg(ch)    *(char*)(0x600000+ch/**/_chan)  /* Write only */
#define RxHldReg(ch)    *(char*)(0x600000+ch/**/_chan)  /* Read only */


/*  Programming the NEC PD7201 for the     M68000          */
/*      Assumes External clock frequency 16 * baud rate */

/* Setting a control register */
#define NECset(ch,r,v) \
        ContReg(ch)= (char)r;\
        ContReg(ch)= (char)v

/* Reset Tx Interrupt */
#define UAtxres(ch)\
        ContReg(ch)= (char)NECtxres

/* Reset External Latch */
#define NECresxt(ch)\
        ContReg(ch)= NECptres(0)

/* WRITE REGISTER 1 */
/*#define NECINR1               (NECtxint|NECrxina) */
#define NECINR1 0       /* no interrupts */

/* WRITE REGISTER 2 */
#define NECINR2         4       /* could also be zero, but remember clrb...*/

/* WRITE REGISTER 3 */
#define NECINR3         (NECrxena|NECrx8bt|NECautoe)

/* WRITE REGISTER 4 */
#define NECINR4         (NECrx1sb|NEC16clk)

/* WRITE REGISTER 5 */
#define NECINR5         (NECtxrts|NECtxena|NECtx8bt|NECdtr)
#define UAtxena(ch)     NECset(ch,5,NECINR5)
#define UAtxdis(ch)     NECset(ch,5,(NECtxrts|NECtx8bt|NECdtr))


#define NECinseq char __NECIN__[]= { \
        (char)NECchres, (char)NECchres,\
        (char)2, (char)(NECINR2),\
        (char)4,(char)(NECINR4),\
        (char)3,(char)(NECINR3),\
        (char)5,(char)(NECINR5),\
        (char)1,(char)(NECINR1)\
}

                                /* Expand NECinseq within scope reach of
                                   NECin expansion.
                                   A (char*) chadx must be provided
                                */
#define NECin(chadx) \
        for (chadx=__NECIN__; chadx<&__NECIN__[sizeof(__NECIN__)];) {\
```

```
                ContReg(A)= *chadx;\
                ContReg(B)= *(chadx++);\
        }

                                        /* initUART uses a (char*) chadx */
#define initUART(chadx) \
        TCSetGrp(TIMAClk,TMODuart,TFRQuart);\
        TCSetGrp(TIMBClk,TMODuart,TFRQuart);\
        TCArmCnt(TSCountA+TSCountB);\
        NECin(chadx);\
        TxHldReg(B)= (char)'\0';\
        TxHldReg(A) = '\0'

                                        /* A good approx. to freq is TFRQuart*k */
#define speedUART(ch,freq) \
        TCSetGrp(TIM/**/ch/**/Clk,TMODuart,freq);\
        TCArmCnt(TSCount/**/ch)

#define ReadUART(ch) RxHldReg(ch)

#define WriteUART(a,ch) TxHldReg(ch)= a

#define WriteBusy(a) \
                {while((char)(StatReg(A)&NECtxrdy)==0); TxHldReg(A)= a;}

#define UARTstat(ch) StatReg(ch)
```

```
/*
 * pcmap.h
 *
 * Macros and definitions for                          memory mapping
 *
 * Jeffrey Mogul        20 March 1981
 *
 * 8 July 1981 (jcm) -- changed Multibus Memory map to cover ca. 1mb
 * 7 May 1981 (jcm) -- revised for PC board version
 * 6 May 1981 (jcm) -- added mapping for Multibus Memory space
 *
 *       from code written by Luis Trabb-Pardo
 */

/*
 * address space allocations
 */
#define ADRSPC_RAM      0x0             /* Read/Write, segmented memory */
#define ADRSPC_PROM0    0x200000        /* ROM #0, boot space */
#define ADRSPC_PROM1    0x400000        /* ROM #1 */
#define ADRSPC_UART     0x600000        /* onboard UART */
#define ADRSPC_TIMER    0x800000        /* onboard Timer */
#define ADRSPC_PAGEMAP  0xA00000        /* base of page table */
#define ADRSPC_SEGMAP   0xC00000        /* base of segment table */
#define ADRSPC_CONTEXT  0xE00000        /* context register */

#define ADRSPC_SIZE     0x200000        /* size of each address space */


/*
 * Memory is divided into 16 contexts; the current context is
 * defined by the context register.
 */

/*
 * Context Register -- only 4 high order bits (of 16) count
 */
#define CONTEXTREG (*(unsigned short *)ADRSPC_CONTEXT)

#define SETCONTEXT(x) CONTEXTREG = ((x)<<12)

/*
 * Surprise! to read the context register, look at the
 * high 4 bits of any segment map entry. (?)
 */
#define GETCONTEXT      ((*(unsigned short *)ADRSPC_SEGMAP) >> 12)

#define NUMCONTEXTS     16
```

```
/*

   A memory address is

        struct {
                unsigned reserved:3;            -- not currently used, MBZ
                unsigned SegInContext:6;        -- segment within contex
                unsigned PageInSeg:4;           -- page within segment
                unsigned ByteInPage:11;         -- byte within page
                }       MemAddress;

   The segment number is computed as <context><SegInContext>, yielding
   a 10-bit segment table index.

   A segment table entry is

        struct {
                unsigned CurContext:4;          -- Current context (Read only)
                unsigned SegProt_User:1;        -- allows User access
                unsigned SegProt_Write:1;       -- allows Write access
                unsigned SegProt_Read:1;        -- allows Read access
                unsigned SegProt_Exec:1;        -- allows Execute access
                unsigned reserved:2;            -- not currently used, MBZ
                unsigned PageNumHi:6;           -- high bits of Page number
                }       SegTabEntry;

   The virtual page number is computed as <PageNumHi><PageInSeg>,
   yielding a 10-bit page table index.

   A page table entry is

        struct {
                unsigned PageUsed:1;            -- "Used" flag
                unsigned PageDirty:1;           -- "Dirty" bit
                unsigned PageSpace:2;           -- physical page space
                unsigned PageFrameNumber:12;    -- physical page index
                }       PageTabEntry;

   The physical memory address is computed as <PageFrameNumber><ByteInPage>,
   yielding a 23-bit byte address.

*/
```

```
/*
 * Segment map constants/macros
 */
#define SEGMAPBASE      ADRSPC_SEGMAP   /* base of segment map */
#define CONTEXTSIZE     0x40            /* 64. segments per context */
#define SEGMAPSIZE      0x400           /* 16*64 segments */

/* return address of segment map entry #<seg> */
#define SEGMAPADR(seg) ((short *)(SEGMAPBASE+((seg)<<15)))

/* set segment table entry #<seg> to <entry> */
#define SETSEGMAP(seg,entry) *SEGMAPADR(seg) = (entry)

/* get segment table entry #<seg> */
#define GETSEGMAP(seg) (*SEGMAPADR(seg))

/* segment protection bits -- can be set independently */
#define SEGPRO_EXEC     0x100
#define SEGPRO_READ     0x200
#define SEGPRO_WRITE    0x400
#define SEGPRO_USER     0x800
#define SEGPRO_ALL      (SEGPRO_EXEC|SEGPRO_READ|SEGPRO_WRITE|SEGPRO_USER)

/*
 * Page map constants/macros
 */
#define PAGEMAPBASE     ADRSPC_PAGEMAP  /* base of page map */
#define PAGEMAPSIZE     0x400           /* 1024. pages total */
#define PAGESIZE        0x800           /* 2048. bytes per page */

/* return address of page map entry #<page> */
#define PAGEMAPADR(page) ((short *)(PAGEMAPBASE+((page)<<11)))

/* set page table entry #<page> to <entry> */
#define SETPAGEMAP(page,entry) *PAGEMAPADR(page) = (entry)

/* get page table entry #<page> */
#define GETPAGEMAP(page) (*PAGEMAPADR(page))

/* page "address space modes" -- mutually exclusive */
#define PGSPC_MEM       0x0000          /* on-board memory */
#define PGSPC_NXM       0x1000          /* invalid page */
#define PGSPC_MBMEM     0x2000          /* multibus memory */
#define PGSPC_MBIO      0x3000          /* multibus I/O */

/*
 * Initial mapping of memory:
 *
 * We need an initial mapping of memory that will suffice
 * to get the monitor started and allow memory to be sized;
 * we make virtual memory map directly onto physical memory
 * for all 16 contexts.  We leave things in context 0.
 */
#define INITMAP\
{\
        register i;\
        register j;\
        for (j = 0; j < NUMCONTEXTS; j++) {\
            SETCONTEXT(j);\
            for (i = 0; i < CONTEXTSIZE; i++) {\
                SETSEGMAP(i, (i|SEGPRO_ALL));\
            }\
```

```
        }\
        SETCONTEXT(0);\
        for (i = 0; i < PAGEMAPSIZE; i++) {\
                SETPAGEMAP(i,(i|PGSPC_MEM));\
        }\
}

/*
   Boot State:

In boot state (the state of the system after reset) the address
        0XXXXX          reads and executes from address 2XXXXX
                        writes onto address 0XXXXX,
in this way it is possible to initialize RAM just after reset.
Also, all interrupts including normally non-maskable ones are
disabled.

Note that CLR instructions should not be used on RAM space in
boot state; this is because the CLR instructions do a read cycle
before writing.

To exit boot state, perform a write operation into the
PROM0 address space.

*/

#define EXITBOOT *(unsigned short *)ADRSPC_PROM0 = 1
```

```
/*
 * MULTIBUS MEMORY AND I/O MAPPING
 */

/*
 * Multibus I/O mapping
 *
 * By convention, MultiBus I/O is mapped at the top of the
 * mappable address space.
 */

#define IOLOWPAGE       0x3E0   /* first MultiBus page */
#define IOLOWADR        0x1F0000        /* IOLOWPAGE << 11 */
#define NUMIOPAGES      0x020   /* number of MultiBus pages */

/*
 * MultiBus(x) returns the mapped address for MultiBus address x
 */
#define MultiBus(x)     (IOLOWADR+(x))

/*
 * initIOMAP initializes the top pages of the map to conform to
 * the convention described above
 */
#define initIOMAP  \
{\
        register i;\
        for (i=0; i<NUMIOPAGES; i++)\
                SETPAGEMAP((i+IOLOWPAGE),(i|PGSPC_MBIO));\
}

/*
 * Multibus memory mapping
 *
 * By convention, MultiBus memory is mapped just below
 * Multibus I/O in the mappable address space.
 */

#define MBMEMLOWPAGE    0x200   /* first MultiBus memory page */
#define MBMEMLOWADR     0x100000        /* MBMEMLOWPAGE << 11 */
#define NUMMBMEMPAGES   0x1e0   /* number of MultiBus memory pages */

/*
 * MultiBusMem(x) returns the mapped address for MultiBus memory address x
 */
#define MultiBusMem(x)  (MBMEMLOWADR+(x))

/*
 * initMBMEMMAP initializes the top pages of the map to conform to
 * the convention described above
 */
#define initMBMEMMAP  \
{\
        register i;\
        for (i=0; i<NUMMBMEMPAGES; i++)\
                SETPAGEMAP((i+MBMEMLOWPAGE),(i|PGSPC_MBMEM));\
}

#if NUMIOPAGES == NUMMBMPAGES
/*
 * InitMultiBusMap initializes the maps for both Multibus memory and
 * I/O; the loop-jamming of initMBMEMMAP and initIOMAP saves
```

```
 * code in the ROM monitor.  This works because NUMIOPAGES ==
 * NUMMBMEMPAGES.
 */
#define InitMultiBusMap  \
{\
        register i;\
        for (i=0; i<NUMIOPAGES; i++) {\
                SETPAGEMAP((i+MBMEMLOWPAGE),(i|PGSPC_MBMEM));\
                SETPAGEMAP((i+IOLOWPAGE),(i|PGSPC_MBIO));\
        }\
}

#else
/* do each initialization; cannot jam loops */
#define InitMultiBusMap \
        initMBMEMMAP;\
        initIOMAP;
#endif
```

```
#include "amd9513.h"

/* Timer addresses on the     system */

#define ADTimCom        *(unsigned short*)0x800002      /* for Commands */
#define ADTimReg        *(unsigned short*)0x800000      /* for Registers */

/* Input frequency is 4MHz */

/*  Channel Assignment is */

#define TIMRefresh      3       /* for refresh task */
#define TSCountR        TSCount3
#define TIMAClk         4       /* Ch.A clock */
#define TSCountA        TSCount4
#define TIMBClk         5       /* Ch.B clock */
#define TSCountB        TSCount5
#define TIMInter        2       /* Interrupt Clock, Level 6 */
#define TSInter         TSCount2
#define TIMClock        1       /* Counter Clock */
#define TSClock         TSCount1

#define ResTIMER \
        TCReset;                /* Master Reset */\
        TCLdCnt(TSAll);         /* Reset All counters */\
        TC16Bus                 /* Enter 16 bit Bus mode */

/* Uart Timing:  the timer functions in "MODE D"... */
#define TMODuart (unsigned short)(TCMFall+TCMNoGa+TCMF1+TCMDiSpG+\
                        TCMRldLd+TCMCntRp+TCMBinCt+TCMDwnCt+TCMTCTog)
        /* with a square wave of 16 * 9615.4Hz */
#define TFRQuart 13

/* Refresh Timing: also "MODE D" */
#define TMODrefr TMODuart
        /* frequency: 500Hz */
#define TFRQrefr 8000

#define initREFR \
        TCSetGrp(TIMRefresh,TMODrefr,TFRQrefr);\
        TCArmCnt(TSCountR)

/* Counter clock Timing:  it counts at any of the available frequencies */
#define TMODcclk (unsigned short)(TCMFall+TCMNoGa+TCMDiSpG+\
                        TCMRldLd+TCMCntRp+TCMBinCt+TCMUpCt+TCMOutLo)

/* Setting up the counting clock: the parameter 'freq' must be
        1 to 5 to select F1 to F5 */
#define TCSetClk(freq) \
        TCLoDaPo(TSModeRg+TIMClock);\
        TRLoad(TMODcclk+TCMF/**/freq);\
        TCDisDP;\
        TCLoDaPo(TSHoldRg+TIMClock)

/* Reading the counting clock */
#define TIMRdClk        (TCSavCnt(TSAll),ADTimReg)

/* Resetting the counting clock */
#define TIMClkRs        TCLdArCnt(TSClock)
```

```
/*
 * sunuart.h
 *
 * Defines semi-hardware-independent constants/macros for
 * sun on-board UART.
 *
 * Jeffrey Mogul
 *      after "necuart.h" by Luis Trabb-Pardo
 *

#include "nec7201.h"

/*  Address definitions for the nec7201 on the SUN board */

/* Channel definitions */

#define A_chan  0
#define B_chan  4

#define ContReg(ch)     *(char*)(0x600002+ch)
#define StatReg(ch)     *(char*)(0x600002+ch)   /* Read only */
#define TxHldReg(ch)    *(char*)(0x600000+ch)   /* Write only */
#define RxHldReg(ch)    *(char*)(0x600000+ch)   /* Read only */


/*  Programming the NEC PD7201 for the SUN M68000 board:  */
/*      Assumes External clock frequency 16 * baud rate */

/*
 * Set a NEC uart control register;
 *      ch == channel
 *      r == register #
 *      v == value
 */
#define NECset(ch,r,v) \
        ContReg(ch)= (char)r;\
        ContReg(ch)= (char)v

/*
 * Reset External/Status Interrupts
 */
#define UART_RESXT(ch)  ContReg(ch) = NECrexst

extern  char UARTInitSeq[];
#define INITBEGIN       &UARTInitSeq[0]
#define INITEND         &UARTInitSeq[sizeof(UARTInitSeq)]

/*
 * initialize both uarts; Aspeed and Bspeed are indices into
 * UARTSpeedTable[]
 *
 * It's not clear why this should be a macro (saves a few bytes)
 */

#define INITUART(Aspeed, Bspeed) {\
        register char* p;\
        TCSetGrp(TIMAClk, TMODuart, TFRQuart);\
        TCSetGrp(TIMBClk, TMODuart, TFRQuart);\
        TCArmCnt(TSCountA + TSCountB);\
        for (p = INITBEGIN; p < INITEND;) {\
            ContReg(A_chan) = *p;\
            ContReg(B_chan) = *p++;\
        }\
```

```
        TxHldReg(A_chan) = '\0';\
        TxHldReg(B_chan) = '\0';\
    }

#define ReadUART(ch) RxHldReg(ch)

#define UARTstat(ch) StatReg(ch)
```

```
/*
 * sunbug.h
 *
 * Definitions for SunBug debugger
 *
 * Jeffrey Mogul                30 March 1981
 */

/*
 * Several Well-known Pup sockets should be defined in pupconstants.h;
 * these are SUNBUGSERVER, SUNBUGUSER, and SUNBOOTUSER.
 */
#define SUNBUGSERVER 0500
#define SUNBUGUSER   0501
#define SUNBOOTUSER  0502

/*
 * SunBug Pup types:
 *      even types go from Server to User
 *      odd types go from User to Server
 */

#define SUNBUG_RETURN   0200    /* return from trap */
#define SUNBUG_TRAP     0201    /* I trapped */
#define SUNBUG_READMEM  0202    /* read memory */
#define SUNBUG_RMACK    0203    /* acks read memory */
#define SUNBUG_WRITEMEM 0204    /* write memory */
#define SUNBUG_WMACK    0205    /* acks write memory */
#define SUNBUG_READREG  0206    /* read register */
#define SUNBUG_RRACK    0207    /* acks read register */
#define SUNBUG_WRITEREG 0210    /* write register */
#define SUNBUG_WRACK    0211    /* acks write register */
#define SUNBUG_RESET    0212    /* reset Sun */
#define SUNBUG_ERROR    0213    /* error */
#define SUNBUG_BOOTYOURSELF     0214    /* Force Bootload */
#define SUNBUG_BOOTDONE 0215    /* Bootload is done */

/*
 * Error codes for SUNBUG_ERROR
 */
#define SBERR_BADCMD    (-1)    /* bad command */
#define SBERR_RESET     (-2)    /* RESET done */
```

```
/*
 * sunemt.h
 *
 * emulator trap definitions for the  -  MC68000 ROM monitor
 *
 * Jeffrey Mogul                    22 May 1981
 */

/*
 * Negative EMT type codes are reserved to supervisor mode
 */

#define SUPERONLY(x)    ((x) < 0)

#define SU(x)   (-(x))

#define EMT_TICKS       0       /* ticks() */
#define EMT_PUTCHAR     1       /* putchar(x) */
#define EMT_VERSION     2       /* version() */
#define EMT_GETCHAR     3       /* getchar() */
#define EMT_GETMEMSIZE  4       /* getmemsize */
#define EMT_SETECHO     5       /* setecho(0/1) */

#define EMT_SETSEGMAP   SU(1)   /* setsegmap(cxt,segno,entry) */
#define EMT_GETSEGMAP   SU(2)   /* getsegmap(cxt,segno) */
#define EMT_GETCONTEXT  SU(3)   /* getcontext() */
#define EMT_SETCONTEXT  SU(4)   /* setcontext(x) */
```

```
/*
 * sunmon.h
 *
 * Header file for     MC68000 ROM Monitor
 *
 * Created:
 *      Jeffrey Mogul                   14 May 1981
 *      out of the ashes of code originally written by Luis Trabb-Pardo
 */

#ifndef SUNMONDEFINED
#define SUNMONDEFINED

#include "buserr.h"
#include "m68vectors.h"

/*
 * Monitor version number: MAJVERSION.MINVERSION
 *      (This could be in an automatically generated subfile?)
 */
#define MAJVERSION      0
#define MINVERSION      8

/*
 * C language "improvements"
 */
#define then
#define true    1
#define false   0
#define loop    for(;;)

typedef int     typroc();       /* change this */
typedef short   word;
typedef long    longword;
typedef char    byte;

/*
 * Various memory layout parameters
 */
#define INITSP  0x1000          /* initial stack pointer, for C */
#define INITSPa 1000            /* for asm() use */

#define USERCODE 0x1000         /* starting address for user programs */


/*
 * RAM Refresh
 *
 * The current hardware does not support the use of "User Interrupt
 * Vectors"; therefore, the RAM Refresh routine starts where they
 * lie.
 */
#define RAMREFR ((typroc*)EVEC_USERINT(0))      /* address of RAM
                                                 * refresh routine
                                                 */

#ifdef TIRAM            /* TI's rams are weird */
#define RAMREFOPS 64    /* operations to do refresh */
#define RAMREFTIME MS_1 /* time between refreshes */
#else
#define RAMREFOPS 128   /* operations to do refresh */
#define RAMREFTIME MS_2 /* time between refreshes */
#endif TIRAM
```

```
/*
 * STRTDATA is the starting address of monitor global data
 */
#define STRTDATA ( ((int)RAMREFR) + (RAMREFOPS*sizeof(short)) )

/*
 * Size of line input buffer - this should be at least 80,
 * to leave room for a full-length S-record.
 */
#define BUFSIZE 80

/*
 * These are the "erase" and "kill" characters for
 * input processing; they can be changed.
 */
#define CERASE1 '\b'    /* backspace */
#define CERASE2 0x7F    /* delete */
#define CKILL   '\025'  /* ctrl/U */

/*
 * These are the default mode-changing characters
 */
#define CENDTRANSP      '\036'  /* end transparent mode, ctrl/^ */
#define CSTLD           '\\'    /* Start Load */

#define UPCASE  0x5F    /* mask to force upper case letters */
#define NOPARITY 0x7F   /* mask to strip off parity */

/*
 * Exception cause codes
 *
 * necmon passes one of these to monitor() [in bmon] on special
 * exceptions.
 *
 * These definitions get used in asm()'s and thus must not have
 * trailing tabs!  (or more than one leading tab)
 */
#define EXC_RESET       0
                        /* Reset switch hit (or soft reset) */
#define EXC_ABORT       1
                        /* Abort switch hit */
#define EXC_BREAK       2
                        /* Breakpoint trap taken */
#define EXC_EXIT        3
                        /* Exit trap taken */
#define EXC_TRACE       4
                        /* Trace trap taken */
#define EXC_EMT 5
                        /* Emulator trap */
#define EXC_BUSERR      6
                        /* Bus Error */
#define EXC_ADRERR      7
                        /* Address Error */

#endif SUNMONDEFINED
```

```
/*
 * necmon.c
 *
 * "kernel" of monitor
 *
 *
 * present state due to Jeffrey Mogul                    18 May 1981
 *      version 2 (pc board) support added               19 May 1981
 *      fixed bug in initialization (Bus Errors)          7 Aug 1981
 */

#include "sunmon.h"
#include "asmdef.h"
#include "globram.h"
#include "necuart.h"
#include "timer.h"
#include "pcmap.h"
#ifdef  FRAMEBUF
#include "framebuf.h"
#endif  FRAMEBUF

/* shorthand */
#define excvmake(ptr, routine) {int routine(); *(ptr) = (long)routine;}

/* Exception Vector */

        /* we need a "mini" exception vector in ROM, just to
         * get started.  This MUST be the first data in the ROM!
         */
        excv(/INITSPa); /* excv(/INITSP) doesn't work here */
        excv(startmon);

/* Exception Links */
        exclink(inter10,I,I);
        exclink(inter14,Z,D);
        exclink(inter18,C,h);
        exclink(inter1C,T,V);
        exclink(inter20,P,r);
        exclink(inter28,U,0);
        exclink(inter2C,U,1);
        exclink(interUN,U,n);
        exclink(interL1,L,1);
        exclink(interL2,L,2);
        exclink(interL3,L,3);
        exclink(interL4,L,4);
        exclink(interL5,L,5);
        exclink(interL6,L,6);
        exclink(interTR,T,r);
label(.enter);
        branch(interrupt);

NECinseq;

main()
{
        register char *p;
        register long *et;
        register GlobDes *gp;
        register long msize;

label(REFRsrv);
        rpush(d0);
        TCClrOut(TIMRefresh);
```

```
        RAMREFR();

#if     RAMREFTIME == MS_2
        GlobPtr->RefrCnt += 2;
#else   /* RAMREFTIME = MS_1 */
        (GlobPtr->RefrCnt)++;
#endif

        /* Abort switch - use GlobPtr->debounce to detect change of state */
#ifdef  ABORTSWITCH
        if (StatReg(A)&NECsync)
#else
        if (StatReg(A)&NECbreak)
#endif  ABORTSWITCH
            GlobPtr->debounce = true;
        else if (GlobPtr->debounce) {   /* changed state! */
            GlobPtr->debounce = false;
            NECresxt(A);
            rpop(d0);
            ExcCause(EXC_ABORT);
            goto ENTERmon;
        }
        rpop(d0);
        NECresxt(A);
endsrv;

label(startmon);
{
        /* FIRST! reset Timers, to avoid being interrupted */
        ResTIMER;

        /* Next, set up memory context and mapping */
        INITMAP;        /* this must be done to provide a stack
                         * for subsequent calls (e.g., mapmem())
                         * also, leaves us in context 0
                         */
        EXITBOOT;       /* Exit boot state (allows memory reads from RAM) */

        excvmake(EVEC_BUSERR,BEcatch);  /* ignore Bus Errors for now */

        msize = mapmem()<<11;           /* map the memory "right" */

        /*
         * Write all of memory in order to initialize parity bits.
         * This can't be a "clr", since that instruction reads(!)
         * before writing.  The use of OPC_2UN1 means that random
         * jumps will cause immediately "U1" traps.
         */
        for (et = 0; et < (long *)msize; ) *et++ = OPC_2UN1;

        gp = GlobPtr;

        gp->MemorySize = msize; /* record for future reference */
        gp->CurContext = 0;     /* ditto */

#ifdef  FRAMEBUF
        /* determine presence of frame buffer -
         * this uses BEcatch which clears memory location #0 on
         * a bus error
         */
        excvmake(EVEC_BUSERR,BEcatch);  /* catch Bus Errors */
        *(long*)0 = 1;
        *(long*)GXUnit0Base = 0;        /* touch frame buffer */
```

```
        gp->FrameBuffer = *(long*)0;
#endif  FRAMEBUF

        initUART(p);
        initREFR;

        /* make all vectors "undefined" */
        for (et = EVEC_TRAPF; et ;)
            excvmake(et--,interUN);

        et = EVEC_BUSERR;       /* skip first two vectors */

        excvmake(et++,BusError);
        excvmake(et++,AddrError);
        excvmake(et++,inter10);
        excvmake(et++,inter14);
        excvmake(et++,inter18);
        excvmake(et++,inter1C);
        excvmake(et++,inter20);
        excvmake(et++,TraceTrap);
        excvmake(et++,inter28);
        excvmake(et++,inter2C);

        et = EVEC_LEVEL1;
        excvmake(et++,interL1);
        excvmake(et++,interL2);
        excvmake(et++,interL3);
        excvmake(et++,interL4);
        excvmake(et++,interL5);
        excvmake(et++,interL6);
        excvmake(et++,REFRsrv);
        while (et <= EVEC_TRAPF)
                excvmake(et++,interTR);
        excvmake(EVEC_TRAP1,TRAPbrek);
        excvmake(EVEC_TRAPE,TRAPexit);
        excvmake(EVEC_TRAPF,TRAPemu);


        /*
         * Create RAM Refresh subroutine -
         *      OPC_2NOP is a long coding two nops, OPC_NOP_RTS is a long
         *      coding an nop followed by an rts.  We fill all
         *      but the last long with OPC_2NOP, then tack on an OPC_NOP_RTS.
         */
        for (et = (long*)RAMREFR;
                et < (long*)((int)RAMREFR + ((RAMREFOPS-2)*sizeof(short)));
                 *(et++) = OPC_2NOP);
        *et = OPC_NOP_RTS;

        GlobBase = GlobPtr;             /* store ptr to globals for users */

        gp->debounce = false;

#ifdef BREAKTRAP
        gp->BrkInPrg = 0;       /* no break in progress */
        gp->BreakAdx = 0;       /* initialize break address */
#endif BREAKTRAP

LabelGbl(SoftReset);    /* jump here to do a soft reset */
        rpush(#0.);     /* "PC" -- maybe should be something useful? */
        wpush(sr);      /* Fake Exception Stack */
        ExcCause(EXC_RESET);
};
```

```
ENTERmon:
        saveregs;
        setINTLV(7);
        monitor();
        restregs;
        endsrv;


label(TRAPbrek);
        ExcCause(EXC_BREAK);
        goto ENTERmon;

unusedlabel1:
label(TRAPexit);
        ExcCause(EXC_EXIT);
        goto ENTERmon;

unusedlabel2:
label(interrupt);
                                goto ENTERmon;

unusedlabel3:
label(TraceTrap);
        ExcCause(EXC_TRACE);
        goto ENTERmon;

unusedlabel4:
label(TRAPemu);
        ExcCause(EXC_EMT);
        goto ENTERmon;

unusedlabel5:
label(BusError)
        SAVEACCESS;             /* save Error Access Address */
        ExcCause(EXC_BUSERR);
        goto ENTERmon;

unusedlabel6:
label(AddrError)
        SAVEACCESS;             /* save Error Access Address */
        ExcCause(EXC_ADRERR);
        goto ENTERmon;

unusedlable7:
label(BEcatch)
        FLUSHACCESS;            /* remove access address from stack */
#ifdef  FRAMEBUF
        *(long*)0 = 0;          /* signal that BE occured */
#endif  FRAMEBUF
        endsrv;
}
```

```
/*
 * bmon.c
 *
 * 68K VERY BASIC MONITOR
 *
 * original author: Luis Trabb Pardo
 * largely re-written by Jeffrey Mogul  April, 1981
 */

#include "sunmon.h"
#include "asmdef.h"
#include "globram.h"
#include "necuart.h"
#include "statreg.h"
#include "pcmap.h"

#define isnum(c) ((c>='0')&&(c<='9'))

/*
 * read the next input line into a global buffer
 */
getline()
{
        register char *p = GlobPtr->linbuf;
        register int linelength = 0;
        register int erase;     /* number of characters to erase */

        GlobPtr->lineptr = p;   /* reset place-in-line pointer */

        while ( (*p = getchar()) != '\r') {
                erase = 0;
                if (*p == CERASE1) {
                    putchar(' ');       /* un-echo the \b */
                    erase = 1;  /* erase one character */
                }
                else if (*p == CERASE2) /* <del> key */
                    erase = 1;
                else if (*p == CKILL)
                    erase = linelength; /* erase all character on line */
                else if (linelength < BUFSIZE) {    /* ok to accept chars */
                    p++;
                    linelength++;
                /* here we could ding for line-too-long */
                }
                while(linelength && erase--) {
                        p--;
                        linelength--;
                        message("\b \b");
                    }
        }

        *(++p)= '\0';   /* the ++ protects the \r to make getone() work */
        GlobPtr->linesize = linelength;
}
```

```
/*
 * gets one character from the input buffer; returns are
 * converted to nulls
 */
char getone()
{
        return ((*(GlobPtr->lineptr)=='\r')?
                '\0':
                 *((GlobPtr->lineptr)++));
}

/*
 * indicates the next character that getone() would return;
 * does NOT convert \r to \0.
 */
char peekchar()
{
        return(*(GlobPtr->lineptr));
}

/*
 *Prints message mess on the controlling terminal.
 */
message(mess)
register char *mess;
{
        for(;*mess!='\0';putchar(*(mess++)));
}

char chardigs[]="0123456789ABCDEF";

/*
 * printhex prints rightmost <digs> hex digits of <val>
 */
printhex(val,digs)
register long val;
register int digs;
{
        register int i;

        i = ((--digs)&7)<<2;    /* digs == 0 => print 8 digits */

        for (; i >= 0; i-=4)
                putchar(chardigs[(val>>i)&0xF]);
}

/*
 * prints a message, prints the l hex digits of v, prints a
 * ?, and reads a new line into the buffer.
 */
queryval (m,v,l)
char *m;
longword v;
int l;
{
        message(m);
        printhex(v,l);
        message("? ");
        getline();
}
```

```
/*
 * Returns the hex value of a char or -1 if the char is not hex
 */
int ishex(ch)
char ch;
{
        register char c= ch;

        if (isnum(c))           then return(c-'0');
        if (c>='a'&&c<='f')     then return(c-'a'+10);
        if (c>='A'&&c<='F')     then return(c-'A'+10);
        return(-1);
}

/* get one hex-coded byte */
gethexbyte()
{
        register int v = 0;

        v = ishex(getone())<<4;
        return(v|ishex(getone()));
}

/* get a hex number */
longword getnum()
{
        register longword v = 0;
        register int hexval;

        while ((hexval= ishex(peekchar()))>=0) {
                        v= (v<<4)| hexval;
                        getone();
        }
        return(v);
}


/* Display/change one memory or map item
 * returns true iff another item coming.
 */
openitem(num,adrs,digs,dsize)
register int num;       /* either address or map number */
register word *adrs;    /* object address */
register int digs;      /* digits in address or map number */
int dsize;              /* 1 == byte, 2 == short, 4 == long */
{
        register char c;

        printhex(num,digs);
        if (dsize == 2)
           queryval(": ",*adrs,4);
        else    /* dsize == 1 */
           queryval(": ",*((char*)adrs),2);
        /* dsize == 4 not implemented */
        c = peekchar();
        if (ishex(c)>=0) {
                if (dsize == 2)
                    *adrs = getnum();
                else    /* dsize == 1 */
                    *((char *)adrs) = getnum();
                /* dsize == 4 not implemented */
                return(1);
```

```
                }
        else if (c1='\r')
                return(0);
}
```

```
#define SREC_DATA       '2'     /* Data-carrying S-record */
#define SREC_TRAILER    '8'     /* Trailer S-record */

/*
 * get an S-record.  Returns character used for handshaking.
 * pcadx is set to address part of S-record for non-data records.
 */
char getrecord(pcadx)
longword *pcadx;
{
        register GlobDes *gp = GlobPtr;
        register int bytecount;
        register int x;
        register longword adx;
        char rtype;
        int w;
        short checksum;
        char *savptr;

        printhex(gp->recount,2);

        /*
         * Format of 'S' record:
         *      S<type><count><address><datum>...<datum><checksum>
         *        digit  byte   3bytes  byte      byte    byte
         */

        rtype = getone();
        if ( (rtype != SREC_DATA) && (rtype != SREC_TRAILER))
            return('T');

        bytecount = gethexbyte();
        /*
         * checksum = 0; checksum += bytecount
         */
        checksum = bytecount;

        /* check to see that the bytecount agrees with line length */
        if ( (bytecount < 3) || (gp->linesize != (bytecount<<1)+4))
            return('L');

        /* accumulate address - 3 bytes */
        adx = 0;
        for (w = 0; w++ < 3; ) {
                adx = (adx<<8)|(x = gethexbyte());
                checksum += x;
                bytecount--;
        }

        /* verify checksum */
        savptr = gp->lineptr;   /* save line ptr for rescanning */
        for (x = bytecount; x > 0 ; x--)
                checksum += gethexbyte();

        if ((++checksum)&0xFF)
            return('K');
```

```
        /* checksum worked - now rescan input and read data */
        if (rtype == SREC_DATA) {
                gp->lineptr = savptr;
                while (--bytecount) {   /* predecrement to ignore checksum */
                        *(char *)adx++ = (char)gethexbyte();
                }
        } else *pcadx = adx;

        (gp->recount)++;

#ifdef MORESPACE
        if (!(gp->recount&0x3))
            busyouta('.');  /* let user know that good records are coming */
#endif MORESPACE

        return('Y');
}

/* Register names */
char _reg_nam[]=
"D0\0D1\0D2\0D3\0D4\0D5\0D6\0D7\0A0\0A1\0A2\0A3\0A4\0A5\0A6\0SS\0US\0SR\0PC";

openreg(r,radx)
word r;
longword *radx;
        /* Display reg r, get new value (if any)
        If right answer:increment r and repeat until last reg
        */
{
        register char c;

        r+=(getnum()&0xF);      /* only use last hex digit */
        radx+=r;
        for(;r<=_reg_MX;r++,radx++)  {
                message(&_reg_nam[r+r+r]);
                queryval(": ",*radx,8);
                c = peekchar();
                if (ishex(c) >= 0) {
                    if (r != _reg_ss)   /* You toucha SS, I breaka you face */
                        *radx = getnum();
                }
                else if (c != '\r') break;
        }
}

doload()
        /* Sends the rest of the line to channel B, and
        enters load mode.  To avoid a "shouting match" with
        the host, we don't echo until we see a CSTLD.
        */
{
        register GlobDes *gp = GlobPtr;

        setmode('B');
        gp->EchoOn = 0;
        gp->linbuf[0] = (char)'\r';
        message(gp->linbuf);
        gp->recount = 0;
        while (getchar() != CSTLD);
        gp->EchoOn = 1;
}
```

```
#ifdef BREAKTRAP
dobreak()
        /* Sets up a break point */
{
        register GlobDes *gp = GlobPtr;
        register word *bp= gp->BreakAdx;
        register word *bpa;

        if (bp && (!gp->BrkInPrg) && (*bp != OPC_TRAP1))
            /* oops? got bashed */
            bp = 0;

        queryval("Break: ",(int)bp,6);

        if (peekchar()!='\r') then {
                gp->BrkInPrg = 0;       /* clear any pending BPT */
                bpa= (word*)getnum();
                if (bp) then *bp = gp->BreakVal;
                gp->BreakVal= *bpa;
                if (bpa) then *bpa= OPC_TRAP1;
                gp->BreakAdx= bpa;
        };
}

/*
 * TakeBreak - handle breakpoint trap
 */
TakeBreak(pcp)
register long *pcp;     /* pointer to user's PC */
{
        register GlobDes *gp = GlobPtr;

        message("\nBreak at ");
        *pcp -= 2;              /* back up to broken word */
        printhex(*pcp,6);
        putchar('\n');
        gp->BrkInPrg++;         /* remember that we are breaking */
        *(gp->BreakAdx) = gp->BreakVal; /* restore old instruction */
}
#endif BREAKTRAP

/*
 * Sets terminal operation mode: A, B, or T
 */
setmode(c)
register char c;
{
        register GlobDes *gp = GlobPtr;
        register char former = gp->ConChan;

        gp->EchoOn = 1;
        if (c=='A') {
                gp->ConChan = A_chan;
                /* if already on channel A, don't putchar('\n') */
                if (former != A_chan)
                    putchar('\n');
        } else if (c=='B') then {
                gp->ConChan = B_chan;
        } else if (c=='T') then {
                transparent();
                putchar('\n');
#ifdef MORESPACE
```

```
        } else message("Unknown mode\n");
#else
        };
#endif MORESPACE
}

/*#ifdef MORESPACE*/
#ifndef NETBOOT
/*
 * give a little help
 */
givehelp()
{
        message("Commands:\nAn,Dn,Mn - open A, D, or Map reg #n\n");
        message("E a - examine word @a\nO a - examine byte @a\n");
        message("G [a] - go\nC - continue\nB - breakpoint\n");
        message("R - open proc regs\nI T - transparent mode\n");
        message("L <cmd> - download with <cmd>\nX - set escape\n");
}
#endif NETBOOT
/*#endif MORESPACE*/
```

```
#ifdef NETBOOT
#define MAXBOOTTRIES 5
/*
 * get and setup a program via the network.  Keeps trying
 * until it succeeds or MAXBOOTTRIES is exceeded.
 */
netfetch(bfname)
register char *bfname;
{
        register int bflen = -1;
        register int tries = 0;

        while ((bflen <= 0) && (tries < MAXBOOTTRIES)) {
            if (tries++) putchar('.');
            bflen = getbootfile(1,bfname,USERCODE,0);
        }
        if (bflen > 0)
            return(setup(USERCODE,bflen));
        else {
            message("Timeout\n");
            return(0);
        }
}
#endif NETBOOT
```

```
monitor(        r_d0, r_d1, r_d2, r_d3, r_d4, r_d5, r_d6, r_d7,
                r_a0, r_a1, r_a2, r_a3, r_a4, r_a5, r_a6,
                r_ss, r_us, r_sr, r_pc)
long    r_d0, r_d1, r_d2, r_d3, r_d4, r_d5, r_d6, r_d7,
        r_a0, r_a1, r_a2, r_a3, r_a4, r_a5, r_a6,
        r_ss, r_us, r_sr, r_pc;

        /* This mini-monitor does only a few things:
                a <octal dig>   open A regs
                b               set breakpoint
                c [<address>]   continue [at <address>]
                d <octal dig>   open D regs
                e <hex number>  open address (as word)
                f <filename>    netload file but don't start
                g [<address>]   start (call) [at <address>]
                h               help
                i [A|B|T]       terminal operation mode A,B or T
                k               reset monitor stack
                l <UNIX cmd>    get in LOAD mode
                m <hex number>  open segment map
                n <filename>    netload file and start it
                o <hex number>  open address (as byte)
                p <hex number>  open page map
                r               open PC,SR,SS,US
                s               load S-record
                x <char>        set transparent-mode escape to <char>
        */
{
        register GlobDes *gp = GlobPtr;
        register char c;
        register int goadx;
        long cause;     /* must NOT be in register */
        int (*calladx)();       /* ditto */

        cause = r_sr>>16;

        if (cause != EXC_EMT)   /* don't reset UART for EMT! */
            setmode('A');               /* revert to normal UART operation */
                /* probably we should save mode and restore it later? */

        switch(cause) { /* what caused entry into monitor? */

        case EXC_RESET:
                /* we define the name as a macro to cause the
                 * version numbers to be substituted.
                 */
#ifdef NETBOOT
#define IDENT(X,Y) "Sun Network Monitor, Version X.Y - 0x"
#else
#define IDENT(X,Y) "Sun Monitor, Version X.Y - 0x"
#endif NETBOOT
                message(IDENT(MAJVERSION,MINVERSION));
                printhex(gp->MemorySize,5);
                message(" bytes of memory\n");
                r_us = gp->MemorySize;  /* reset User Stack pointer */
                gp->EndTransp = CENDTRANSP;     /* default escape */
                break;

        case EXC_ABORT:
                message("\nAbort");
                goto PCPrint;
```

```
        case EXC_BREAK:
#ifdef  BREAKTRAP
                TakeBreak(&r_pc);
                break;
#else
                message("BREAK\n");
                break;
#endif  BREAKTRAP

        case EXC_EXIT:
                break;

        case EXC_TRACE:
#ifdef  BREAKTRAP
                if (gp->BrkInPrg) {
                    /* this is single step past broken instruction */
                    r_sr &= ~SR_TRACE;
                    gp->BrkInPrg = 0;
                    if ((*(gp->BreakAdx)) == gp->BreakVal) {
                        /* good - not bashed since we set it */
                        *(gp->BreakAdx) = OPC_TRAP1;    /* reset break */
                    }
                    return;     /* in either case, return to user code */
                }
#endif  BREAKTRAP
                message("\nTrace trap");
                goto PCPrint;

        case EXC_EMT:   /* emulator trap */
#ifdef EMULATE
                r_d0 = Emulate(&r_pc,r_sr,r_us);
#else
                r_d0 = -1;      /* return -1 if no emulator */
#endif EMULATE
                return;

        case EXC_BUSERR:
                message("Bus");
                goto AccAddPrint;

        case EXC_ADRERR:
                message("Address");
AccAddPrint:    message(" Error, addr: ");
                printhex(*(long*)(&(gp->BE.Data.AccAddrLo)),8);
PCPrint:        message(" at ");
                printhex(r_pc,6);
                putchar('\n');
                break;

        default: cause = r_sr&0xFFFF0000;
                message("\nException: ");
                message((char *)&cause);    /* high word of r_sr is message */
                putchar('\n');
                break;
        }
```

```
        r_sr &= 0xFFFF;         /* clear extraneous high word */

        loop {
                message(">");
                getline();

                while(c= getone())
                    if (c >= '0') break;

                while (peekchar() == ' ')
                    getone();   /* remove any spaces before argument */

                if (c == '\0') continue;

                switch(c&UPCASE) {      /* slight hack, force upper case */

                case 'A':       /* open A register */
                        openreg(_reg_a0,&r_d0);
                        break;
#ifdef BREAKTRAP
                case 'B':       /* set breakpoint */
                        dobreak();
                        break;
#endif BREAKTRAP
                case 'C':       /* Continue */
                        if(goadx = getnum())
                            then r_pc = goadx;
#ifdef BREAKTRAP
                        else if (gp->BrkInPrg) {
                            /* breakpoint trap taken */
                            r_sr |= SR_TRACE;   /* set trace trap */
                        }
#endif BREAKTRAP
                        return;
                case 'D':       /* open D register */
                        openreg(_reg_d0,&r_d0);
                        break;
                case 'E':       /* open memory (short word) */
                        for(goadx = getnum()&0x0FFFFFE; ;goadx+=2)
                            if (!openitem(goadx,(word *)goadx,6,2)) break;
                        break;
#ifdef NETBOOT
                case 'F':       /* fetch */
                        gp->linbuf[gp->linesize] = 0;
                        r_pc = netfetch(gp->lineptr);
                        break;
#endif NETBOOT
                case 'G':       /* Go */
                        if (!(*((long*)&calladx) = getnum()))
                                *((long*)&calladx) = r_pc;
                        calladx();
                        break;
#ifdef MORESPACE
                case 'H':       /* Help */
#ifdef NETBOOT
                        *((long*)&calladx) = netfetch("monhelp2");
                        if (calladx) calladx();
                        break;
#else
                        givehelp();
#endif NETBOOT
                        break;
```

```
#endif MORESPACE
```

```
                case 'I':       /* change mode */
                        setmode(getone()&UPCASE);       /* force upper case */
                        break;
                case 'K':       /* reset stack */
                        {extern int sp;
                        sp = INITSP;
                        branch(SoftReset);
                        }
                        break;
                case 'L':       /* enter Load mode */
                        doload();
                        break;
                case 'M':       /* open segment map entry */
                        for (goadx = getnum()&(CONTEXTSIZE-1);
                                goadx < CONTEXTSIZE ;goadx++)  {
                            message("SegMap ");
                            if (!openitem(goadx,(word *)SEGMAPADR(goadx),4,2))
                                break;
                        }
                        break;
#ifdef NETBOOT
                case 'N':       /* net-boot a file */
                        gp->linbuf[gp->linesize] = 0;
                        *((long*)&calladx) = netfetch(gp->lineptr);
                        if (calladx) calladx();
                        break;
#endif NETBOOT
#ifdef MORESPACE
                case 'O':       /* open memory (byte) */
                        for(goadx = getnum()&0x0FFFFFF; ;goadx++)
                            if (!openitem(goadx,(word *)goadx,6,1)) break;
                        break;
#endif MORESPACE
                case 'P':       /* set page map */
                        for (goadx = getnum()&(PAGEMAPSIZE-1);
                                goadx < PAGEMAPSIZE ;goadx++)  {
                            message("PageMap ");
                            if (!openitem(goadx,(word*)PAGEMAPADR(goadx),4,2))
                                break;
                        }
                        break;
#ifdef FIXME    /* orphan! set context reg */
                        setPID(getnum());
                        break;
#endif FIXME
                case 'R':       /* open registers */
                        openreg(_reg_ss,&r_d0);
                        break;
                case 'S':       /* accept S record */
                        putchar(getrecord(&r_pc));
                        break;
#ifdef TESTFROB
                case 'T':       /* test new frob */
                        test();
                        break;
#endif TESTFROB
                case 'X':       /* set transparent-mode escape */
                        gp->EndTransp = getone();
                        break;
                default:
                        message("What?\n");
                        break;
```

```
            }       /* end of switch */
        }
}
```

```
/*
 * busyio.c
 *
 * busywait I/O module
 *
 * Jeffrey Mogul                1 May 1981
 */

#include "sunmon.h"
#include "asmdef.h"
#include "globram.h"
#include "necuart.h"

busyouta(x)
register char x;
{
        while((char)(UARTstat(A)&NECtxrdy)==0); /* wait for ready */
        TxHldReg(A) = x;
}


busyoutb(x)
register char x;
{
        while((char)(UARTstat(B)&NECtxrdy)==0); /* wait for ready */
        TxHldReg(B) = x;
}

putchar(x)
register char x;
{
        register int (*outptr)();

        if (GlobPtr->ConChan == A_chan)
            outptr = busyouta;
        else
            outptr = busyoutb;

        outptr(x);

        x &= NOPARITY;  /* strip parity bit */

        if (x == '\r') outptr('\n');
        if (x == '\n') outptr('\r');
}

char getchar()
{
        register char c;
        register long rp;

        rp = 0x600002 + GlobPtr->ConChan;       /* rp -> control reg */

        while (!( (*(char*)rp) & NECrxrdy));    /* busy wait for char */
        rp -= 2;                                /* rp -> data reg */
        c = (*(char*)rp)&NOPARITY;              /* get char, strip parity */

        if (GlobPtr->EchoOn)
            putchar(c);

        return(c);
}
```

```
/*
 * transparent mode - cross connect UARTs A and B.
 * Since it is quite possible that the two UARTs are running
 * at different speeds, there is a potential flow control
 * problem here.  We leave this up to the user to solve;
 * this routine runs at the speed of the slower side.
 */
transparent()
{
        register char c;
        register int EscPending = 0;    /* true if we just saw an escape */

        loop {
            /* move char from Host to here */
            if (UARTstat(B)&NECrxrdy)
                busyouta(ReadUART(B));

            /* has user typed a character? */
            if (UARTstat(A)&NECrxrdy) {
                c = ReadUART(A);        /* get the char */
                if (EscPending)
                    if ((c&UPCASE) == 'C') {
                        break;  /* esc-c: end transparent mode */
                    }
                    else EscPending = 0;        /* clear flag */
                else if ((c&NOPARITY)==(GlobPtr->EndTransp)) {
                        EscPending++;
                        continue;
                }
                busyoutb(c);    /* send char to host */
            }
        }
}
```

```
/*
 * emulate.c
 *
 * Emulator trap handler
 */

#ifdef EMULATE
#include "sunmon.h"
#include "globram.h"
#include "pcmap.h"
#include "sunemt.h"
#include "statreg.h"

#define ARG_TRAPTYPE    0       /* position of trap type on arg list */
#define ARG_ARG1        1       /* position of argument 1 */
#define ARG_ARG2        2       /* position of argument 2 */
#define ARG_ARG3        3       /* position of argument 3 */

Emulate(pcadr,usrsr,usrsp)
register long *pcadr;   /* address of saved pc */
register short usrsr;   /* user's status register */
register long *usrsp;   /* user stack pointer */
{
        register int traptype;
        register int traparg1;
        register int traparg2;
        register int traparg3;
        register GlobDes *gp = GlobPtr;

        if (usrsr&SR_SUPERVISOR) {      /* super-mode trap */
            usrsp = &(pcadr[1]);        /* arguments follow saved pc */
            }

        traptype = usrsp[ARG_TRAPTYPE];
        traparg1 = usrsp[ARG_ARG1];     /* this might be a bug! suppose
                                         * traptype is at bottom of stack! */
        traparg2 = usrsp[ARG_ARG2];
        traparg3 = usrsp[ARG_ARG3];

        /* Check if a user-mode trap is reserved to supervisor mode */

        if ( (SUPERONLY(traptype)) && ((usrsr&SR_SUPERVISOR) == 0) ) {
                /* nasty, nasty */
                return(-1);     /* return failure */
        }

        switch (traptype) {

        case EMT_PUTCHAR:
                putchar(traparg1);
                break;

        case EMT_VERSION:
                return((MAJVERSION*0x100)+MINVERSION);
                break;

        case EMT_GETCHAR:
                return(getchar());
                break;

        case EMT_SETECHO:
                gp->EchoOn = traparg1;
                return(0);
```

```
        case EMT_SETSEGMAP:
                /* this must be done in registers since changing
                 * the context may lose us our ram!
                 */
                traptype = gp->CurContext;      /* remember current
                                                 * context (traptype
                                                 * is free) */
                SETCONTEXT(traparg1);
                SETSEGMAP(traparg2,traparg3);
                SETCONTEXT(traptype);
                return(0);
                break;

        case EMT_GETSEGMAP:
                /* this must be done in registers since changing
                 * the context may lose us our ram!
                 */
                traptype = gp->CurContext;      /* remember current
                                                 * context (traptype
                                                 * is free) */
                SETCONTEXT(traparg1);
                traparg3 = GETSEGMAP(traparg2);
                SETCONTEXT(traptype);
                return(traparg3);
                break;

        case EMT_GETCONTEXT:
                return(gp->CurContext);
                break;

        case EMT_SETCONTEXT:
                gp->CurContext = traparg1;
                SETCONTEXT(traparg1);
                return(0);
                break;

        case EMT_GETMEMSIZE:
                return(gp->MemorySize);
                break;

        case EMT_TICKS:
                return(gp->RefrCnt);
                break;

        default:
                return(-1);
                break;
        }
}
#endif EMULATE
```

```
 1                                                          .data
 2                                                  _disp_us=64.
 3                                                          .text
 4     00 0000  0000  1000                                  .long /1000
 5     00 0004  0000  00BC                                  .long startmon
 6     00 0008                                      inter10:
 7     00 0008  3F3C  4949                                  movw    #'I*256.+'I,sp@
 8     00 000C  6054                                        jra     .enter
 9     00 000E                                      inter14:
10     00 000E  3F3C  5A44                                  movw    #'Z*256.+'D,sp@
11     00 0012  604E                                        jra     .enter
12     00 0014                                      inter18:
13     00 0014  3F3C  4368                                  movw    #'C*256.+'h,sp@
14     00 0018  6048                                        jra     .enter
15     00 001A                                      inter1C:
16     00 001A  3F3C  5456                                  movw    #'T*256.+'V,sp@
17     00 001E  6042                                        jra     .enter
18     00 0020                                      inter20:
19     00 0020  3F3C  5072                                  movw    #'P*256.+'r,sp@
20     00 0024  603C                                        jra     .enter
21     00 0026                                      inter28:
22     00 0026  3F3C  5530                                  movw    #'U*256.+'0,sp@
23     00 002A  6036                                        jra     .enter
24     00 002C                                      inter2C:
25     00 002C  3F3C  5531                                  movw    #'U*256.+'1,sp@
26     00 0030  6030                                        jra     .enter
27     00 0032                                      interUN:
28     00 0032  3F3C  556E                                  movw    #'U*256.+'n,sp@
29     00 0036  602A                                        jra     .enter
30     00 0038                                      interL1:
31     00 0038  3F3C  4C31                                  movw    #'L*256.+'1,sp@
32     00 003C  6024                                        jra     .enter
33     00 003E                                      interL2:
34     00 003E  3F3C  4C32                                  movw    #'L*256.+'2,sp@
35     00 0042  601E                                        jra     .enter
36     00 0044                                      interL3:
37     00 0044  3F3C  4C33                                  movw    #'L*256.+'3,sp@
38     00 0048  6018                                        jra     .enter
39     00 004A                                      interL4:
40     00 004A  3F3C  4C34                                  movw    #'L*256.+'4,sp@
41     00 004E  6012                                        jra     .enter
42     00 0050                                      interL5:
43     00 0050  3F3C  4C35                                  movw    #'L*256.+'5,sp@
44     00 0054  600C                                        jra     .enter
45     00 0056                                      interL6:
46     00 0056  3F3C  4C36                                  movw    #'L*256.+'6,sp@
47     00 005A  6006                                        jra     .enter
48     00 005C                                      interTR:
49     00 005C  3F3C  5472                                  movw    #'T*256.+'r,sp@
50     00 0060  4E71                                        jra     .enter
51     00 0062                                      .enter:
52                                                   .text;
53     00 0062  4EF9  0000  0330                            jmp     interrupt
54                                                          .globl  __NECIN__
55     00 0068                                      __NECIN__:
56     00 0068  1818                                        .word   6168
57     00 006A  0204                                        .word   516
58     00 006C  0444                                        .word   1092
59     00 006E  03E1                                        .word   993
60     00 0070  05EA                                        .word   1514
61     00 0072  0100                                        .word   256
62                                                          .text
63                                                          .globl  main
```

```
 64    00 0074                                          main:
 65    00 0074  4E56  FFE4                                  link    a6,#-_F1
 66    00 0078  48EE  38E4  FFE4                            moveml  #_S1,a6@(-_F1
 67                                                     | A1 = 8
 68    00 007E                                          REFRsrv:
 69                                                         .text
 70    00 007E  2F00                                        movl    d0,sp@-
 71    00 0080  33FC  FFE3  0080  0002                      movw    #65507,838861(
 72    00 0088  4EB9  0000  0100                            jbsr    256
 73    00 008E  54B9  0000  0274                            addql   #2,628
 74    00 0094  1039  0060  0002                            movb    6291458,d0
 75    00 009A  4880                                        extw    d0
 76    00 009C  48C0                                        extl    d0
 77    00 009E  0280  0000  0080                            andl    #128,d0
 78    00 00A4  6764                                        jeq     .L14
 79    00 00A6  23FC  0000  0001  0000  0266                movl    #1,614
 80    00 00B0                                          .L15:
 81    00 00B0  201F                                        movl    sp@+,d0
 82    00 00B2  13FC  0010  0060  0002                      movb    #16,6291458
 83    00 00BA  4E73                                        rte
 84    00 00BC                                          startmon:
 85    00 00BC  33FC  FFFF  0080  0002                      movw    #65535,838861(
 86    00 00C4  33FC  FF5F  0080  0002                      movw    #65375,838861(
 87    00 00CC  33FC  FFEF  0080  0002                      movw    #65519,838861(
 88    00 00D4  4285                                        clrl    d5
 89    00 00D6  6054                                        jra     .L20
 90    00 00D8                                          .L20001:
 91    00 00D8  2005                                        movl    d5,d0
 92    00 00DA  720C                                        moveq   #12,d1
 93    00 00DC  E3A0                                        asll    d1,d0
 94    00 00DE  33C0  00E0  0000                            movw    d0,14680064
 95    00 00E4  4286                                        clrl    d6
 96    00 00E6                                          .L23:
 97    00 00E6  0C86  0000  0040                            cmpl    #64,d6
 98    00 00EC  6C3C                                        jge     .L18
 99    00 00EE  2006                                        movl    d6,d0
100    00 00F0  0080  0000  0F00                            orl     #3840,d0
101    00 00F6  2206                                        movl    d6,d1
102    00 00F8  740F                                        moveq   #15,d2
103    00 00FA  E5A1                                        asll    d2,d1
104    00 00FC  0681  00C0  0000                            addl    #12582912,d1
105    00 0102  2041                                        movl    d1,a0
106    00 0104  3080                                        movw    d0,a0@
107    00 0106  5286                                        addql   #1,d6
108    00 0108  60DC                                        jra     .L23
109    00 010A                                          .L14:
110    00 010A  4AB9  0000  0266                            tstl    614
111    00 0110  679E                                        jeq     .L15
112    00 0112  42B9  0000  0266                            clrl    614
113    00 0118  13FC  0010  0060  0002                      movb    #16,6291458
114    00 0120  201F                                        movl    sp@+,d0
115    00 0122  3F3C  0001                                  movw    #1.,sp@-
116    00 0126  6000  01D8                                  jra     .L17
117    00 012A                                          .L18:
118    00 012A  5285                                        addql   #1,d5
119    00 012C                                          .L20:
120    00 012C  0C85  0000  0010                            cmpl    #16,d5
121    00 0132  6DA4                                        jlt     .L20001
122    00 0134  4279  00E0  0000                            clrw    14680064
123    00 013A  4286                                        clrl    d6
124    00 013C  601A                                        jra     .L26
125    00 013E                                          .L20003:
126    00 013E  2006                                        movl    d6,d0
```

```
127     00 0140  0080  0000  0000                    orl     #0,d0
128     00 0146  2206                                movl    d6,d1
129     00 0148  740B                                moveq   #11,d2
130     00 014A  E5A1                                asll    d2,d1
131     00 014C  0681  00A0  0000                    addl    #10485760,d1
132     00 0152  2041                                movl    d1,a0
133     00 0154  3080                                movw    d0,a0@
134     00 0156  5286                                addql   #1,d6
135     00 0158                                  .L26:
136     00 0158  0C86  0000  0400                    cmpl    #1024,d6
137     00 015E  6DDE                                jlt     .L20003
138     00 0160  33FC  0001  0020  0000              movw    #1,2097152
139     00 0168  4EB9  0000  0000                    jbsr    mapmem
140     00 016E  720B                                moveq   #11,d1
141     00 0170  E3A0                                asll    d1,d0
142     00 0172  2E00                                movl    d0,d7
143     00 0174  287C  0000  0000                    movl    #0,a4
144     00 017A  6006                                jra     .L30
145     00 017C                                  .L20005:
146     00 017C  28FC  FFFF  FFFF                    movl    #-1,a4@+
147     00 0182                                  .L30:
148     00 0182  B9C7                                cmpl    d7,a4
149     00 0184  65F6                                jcs     .L20005
150     00 0186  267C  0000  0200                    movl    #512,a3
151     00 018C  2747  0078                          movl    d7,a3@(120)
152     00 0190  42AB  007C                          clrl    a3@(124)
153     00 0194  33FC  FF04  0080  0002              movw    #65284,838861C
154     00 019C  33FC  0B22  0080  0000              movw    #2850,8388608
155     00 01A4  33FC  000D  0080  0000              movw    #13,8388608
156     00 01AC  33FC  FF05  0080  0002              movw    #65285,838861C
157     00 01B4  33FC  0B22  0080  0000              movw    #2850,8388608
158     00 01BC  33FC  000D  0080  0000              movw    #13,8388608
159     00 01C4  33FC  FF38  0080  0002              movw    #65336,838861C
160     00 01CC  2A7C  0000  0068                    movl    #__NECIN__,a5
161     00 01D2  600C                                jra     .L33
162     00 01D4                                  .L20007:
163     00 01D4  13D5  0060  0002                    movb    a5@,6291458
164     00 01DA  13DD  0060  0006                    movb    a5@+,6291462
165     00 01E0                                  .L33:
166     00 01E0  BBFC  0000  0074                    cmpl    #__NECIN__+12,
167     00 01E6  65EC                                jcs     .L20007
168     00 01E8  4239  0060  0004                    clrb    6291460
169     00 01EE  4239  0060  0000                    clrb    6291456
170     00 01F4  33FC  FF03  0080  0002              movw    #65283,8388610
171     00 01FC  33FC  0B22  0080  0000              movw    #2850,8388608
172     00 0204  33FC  1F40  0080  0000              movw    #8000,8388608
173     00 020C  33FC  FF24  0080  0002              movw    #65316,8388610
174     00 0214  287C  0000  00BC                    movl    #188,a4
175     00 021A                                  .L20009:
176     00 021A  288C  0000  0032                    movl    #interUN,a4@
177     00 0220  598C                                subql   #4,a4
178     00 0222  B9FC  0000  0000                    cmpl    #0,a4
179     00 0228  66F0                                jne     .L20009
180     00 022A  287C  0000  0008                    movl    #8,a4
181     00 0230  28FC  0000  033E                    movl    #BusError,a4@+
182     00 0236  28FC  0000  035A                    movl    #AddrError,a4@
183     00 023C  28FC  0000  0008                    movl    #inter10,a4@+
184     00 0242  28FC  0000  000E                    movl    #inter14,a4@+
185     00 0248  28FC  0000  0014                    movl    #inter18,a4@+
186     00 024E  28FC  0000  001A                    movl    #inter1C,a4@+
187     00 0254  28FC  0000  0020                    movl    #inter20,a4@+
188     00 025A  28FC  0000  0332                    movl    #TraceTrap,a4@
189     00 0260  28FC  0000  0026                    movl    #inter28,a4@+
```

```
190     00 0266  28FC  0000  002C                        movl    #inter2C,a4@+
191     00 026C  287C  0000  0064                        movl    #100,a4
192     00 0272  28FC  0000  0038                        movl    #interL1,a4@+
193     00 0278  28FC  0000  003E                        movl    #interL2,a4@+
194     00 027E  28FC  0000  0044                        movl    #interL3,a4@+
195     00 0284  28FC  0000  004A                        movl    #interL4,a4@+
196     00 028A  28FC  0000  0050                        movl    #interL5,a4@+
197     00 0290  28FC  0000  0056                        movl    #interL6,a4@+
198     00 0296  28FC  0000  007E                        movl    #REFRsrv,a4@+
199     00 029C  6006                                    jra     .L55
200     00 029E                                  .L20011:
201     00 029E  28FC  0000  005C                        movl    #interTR,a4@+
202     00 02A4                                  .L55:
203     00 02A4  B9FC  0000  00BC                        cmpl    #188,a4
204     00 02AA  63F2                                    jls     .L20011
205     00 02AC  23FC  0000  0324  0000  0084            movl    #TRAPbrek,132
206     00 02B6  23FC  0000  032A  0000  00B8            movl    #TRAPexit,184
207     00 02C0  23FC  0000  0338  0000  00BC            movl    #TRAPemu,188
208     00 02CA  287C  0000  0100                        movl    #256,a4
209     00 02D0  6006                                    jra     .L63
210     00 02D2                                  .L20013:
211     00 02D2  28FC  4E71  4E71                        movl    #1316048497,a4
212     00 02D8                                  .L63:
213     00 02D8  B9FC  0000  01FC                        cmpl    #508,a4
214     00 02DE  65F2                                    jcs     .L20013
215     00 02E0  28BC  4E71  4E75                        movl    #1316048501,a4
216     00 02E6  23FC  0000  0200  0000  0000            movl    #512,0
217     00 02F0  42AB  0066                              clrl    a3@(102)
218                                              .globl  SoftReset
219     00 02F4                                  SoftReset:
220                                              .text
221     00 02F4  2F3C  0000  0000                        movl    #0.,sp@-
222     00 02FA  40E7                                    movw    sr,sp@-
223     00 02FC  3F3C  0000                              movw    #0.,sp@-
224     00 0300                                  .L17:
225     00 0300  598F                                    subql   #4.,sp
226     00 0302  48E7  FFFF                              moveml  #/FFFF,sp@-
227     00 0306  4E68                                    movl    usp,a0
228     00 0308  2F48  0040                              movl    a0,sp@(_disp_us
229     00 030C  46FC  2700                              movw    #9984,sr
230     00 0310  4EB9  0000  0000                        jbsr    monitor
231     00 0316  206F  0040                              movl    sp@(_disp_us),a
232     00 031A  4E60                                    movl    a0,usp
233     00 031C  4CDF  FFFF                              moveml  sp@+,#/FFFF
234     00 0320  5C8F                                    addql   #6.,sp
235     00 0322  4E73                                    rte
236     00 0324                                  TRAPbrek:
237     00 0324  3F3C  0002                              movw    #2.,sp@-
238     00 0328  60D6                                    jra     .L17
239     00 032A                                  TRAPexit:
240     00 032A  3F3C  0003                              movw    #3.,sp@-
241     00 032E  60D0                                    jra     .L17
242     00 0330                                  interrupt:
243     00 0330  60CE                                    jra     .L17
244     00 0332                                  TraceTrap:
245     00 0332  3F3C  0004                              movw    #4.,sp@-
246     00 0336  60C8                                    jra     .L17
247     00 0338                                  TRAPemu:
248     00 0338  3F3C  0005                              movw    #5.,sp@-
249     00 033C  60C2                                    jra     .L17
250     00 033E                                  BusError:
251     00 033E  23C8  0000  0004                        movl    a0,4.
252     00 0344  2079  0000  0000                        movl    0.,a0
```

```
253     00 034A  20DF                          movl    sp@+,a0@+
254     00 034C  209F                          movl    sp@+,a0@
255     00 034E  2079  0000  0004              movl    4.,a0
256     00 0354  3F3C  0006                    movw    #6.,sp@-
257     00 0358  60A6                          jra     .L17
258     00 035A                        AddrError:
259     00 035A  23C8  0000  0004              movl    a0,4.
260     00 0360  2079  0000  0000              movl    0.,a0
261     00 0366  20DF                          movl    sp@+,a0@+
262     00 0368  209F                          movl    sp@+,a0@
263     00 036A  2079  0000  0004              movl    4.,a0
264     00 0370  3F3C  0007                    movw    #7.,sp@-
265     00 0374  608A                          jra     .L17
266     00 0376  4E75                          rts
267                                    _F1 = 28
268                                    _S1 = 14564
269                                    M1 = 0
270                                        .data
271     00 0378                        .end
```

```
 1                                                          .data
 2                                                  _disp_us=64.
 3                                                          .text
 4                                                          .globl  getline
 5      00 0000                                     getline:
 6      00 0000  4E56  FFF4                                 link    a6,#-_F1
 7      00 0004  48EE  20C0  FFF4                           moveml  #_S1,a6@(-_F1
 8                                                  | A1 = 8
 9      00 000A  2A7C  0000  0208                           movl    #520,a5
10      00 0010  4287                                       clrl    d7
11      00 0012  23CD  0000  025A                           movl    a5,602
12      00 0018  6052                                       jra     .L14
13      00 001A                                     .L20000:
14      00 001A  4286                                       clrl    d6
15      00 001C  0C15  0008                                 cmpb    #8,a5@
16      00 0020  6612                                       jne     .L16
17      00 0022  2F3C  0000  0020                           movl    #32,sp@-
18      00 0028  4EB9  0000  0000                   7       jbsr    putchar
19      00 002E  588F                                       addql   #4,sp
20      00 0030                                     .L20001:
21      00 0030  7C01                                       moveq   #1,d6
22      00 0032  601C                                       jra     .L24
23      00 0034                                     .L16:
24      00 0034  0C15  007F                                 cmpb    #127,a5@
25      00 0038  67F6                                       jeq     .L20001
26      00 003A                                     .L19:
27      00 003A  0C15  0015                                 cmpb    #21,a5@
28      00 003E  6604                                       jne     .L21
29      00 0040  2C07                                       movl    d7,d6
30      00 0042  600C                                       jra     .L24
31      00 0044                                     .L21:
32      00 0044  0C87  0000  0050                           cmpl    #80,d7
33      00 004A  6C04                                       jge     .L24
34      00 004C  528D                                       addql   #1,a5
35      00 004E  5287                                       addql   #1,d7
36      00 0050                                     .L23:
37      00 0050                                     .L22:
38      00 0050                                     .L20:
39      00 0050                                     .L18:
40      00 0050                                     .L24:
41      00 0050  4A87                                       tstl    d7
42      00 0052  6718                                       jeq     .L14
43      00 0054  2006                                       movl    d6,d0
44      00 0056  5386                                       subql   #1,d6
45      00 0058  4A80                                       tstl    d0
46      00 005A  6710                                       jeq     .L14
47      00 005C  538D                                       subql   #1,a5
48      00 005E  5387                                       subql   #1,d7
49                                                          .text
50      00 0060  2F3C  0000  09C4                           movl    #.L27,sp@-
51      00 0066  616E                                       jbsr    message
52      00 0068  588F                                       addql   #4,sp
53      00 006A  60E4                                       jra     .L24
54      00 006C                                     .L25:
55      00 006C                                     .L14:
56      00 006C  4EB9  0000  0000                   7       jbsr    getchar
57      00 0072  1A80                                       movb    d0,a5@
58      00 0074  0C00  000D                                 cmpb    #13,d0
59      00 0078  66A0                                       jne     .L20000
60      00 007A                                     .L15:
61      00 007A  528D                                       addql   #1,a5
62      00 007C  4215                                       clrb    a5@
63      00 007E  23C7  0000  025E                           movl    d7,606
```

```
 64     00 0084                                          .L12:
 65     00 0084  4CEE  20C0  FFF4                           moveml  a6@(-_F1),#838
 66     00 008A  4E5E                                       unlk    a6
 67     00 008C  4E75                                       rts
 68                                                     _F1 = 12
 69                                                     _S1 = 8384
 70                                                     | M1 = 4
 71                                                         .globl  getone
 72     00 008E                                         getone:
 73     00 008E  4E56  0000                                 link    a6,#-_F2
 74     00 0092  48EE  0000  0000                           moveml  #_S2,a6@(-_F2)
 75                                                     | A2 = 8
 76     00 0098  2079  0000  025A                           movl    602,a0
 77     00 009E  0C10  000D                                 cmpb    #13,a0@
 78     00 00A2  6604                                       jne     .L10000
 79     00 00A4  4280                                       clrl    d0
 80     00 00A6  6014                                       jra     .L29
 81     00 00A8                                         .L10000:
 82     00 00A8  2039  0000  025A                           movl    602,d0
 83     00 00AE  52B9  0000  025A                           addql   #1,602
 84     00 00B4  2040                                       movl    d0,a0
 85     00 00B6  1010                                       movb    a0@,d0
 86     00 00B8  4880                                       extw    d0
 87     00 00BA  48C0                                       extl    d0
 88     00 00BC                                         .L10001:
 89     00 00BC                                         .L29:
 90     00 00BC  4E5E                                       unlk    a6
 91     00 00BE  4E75                                       rts
 92                                                     _F2 = 0
 93                                                     _S2 = 0
 94                                                     | M2 = 0
 95                                                         .text
 96                                                         .globl  peekchar
 97     00 00C0                                         peekchar:
 98     00 00C0  4E56  0000                                 link    a6,#-_F3
 99     00 00C4  48EE  0000  0000                           moveml  #_S3,a6@(-_F3)
100                                                     | A3 = 8
101     00 00CA  2079  0000  025A                           movl    602,a0
102     00 00D0  1010                                       movb    a0@,d0
103     00 00D2                                         .L31:
104     00 00D2  4E5E                                       unlk    a6
105     00 00D4  4E75                                       rts
106                                                     _F3 = 0
107                                                     _S3 = 0
108                                                     | M3 = 0
109                                                         .text
110                                                         .globl  message
111     00 00D6                                         message:
112     00 00D6  4E56  FFFC                                 link    a6,#-_F4
113     00 00DA  48EE  2000  FFFC                           moveml  #_S4,a6@(-_F4)
114     00 00E0  2A6E  0008                                 movl    a6@(8),a5
115                                                     | A4 = 12
116     00 00E4                                         .L35:
117     00 00E4  4A15                                       tstb    a5@
118     00 00E6  6712                                       jeq     .L32
119     00 00E8                                         .L33:
120     00 00E8  101D                                       movb    a5@+,d0
121     00 00EA  4880                                       extw    d0
122     00 00EC  48C0                                       extl    d0
123     00 00EE  2F00                                       movl    d0,sp@-
124     00 00F0  4EB9  0000  0000                       7   jbsr   putchar
125     00 00F6  588F                                       addql   #4,sp
126     00 00F8  60EA                                       jra     .L35
```

```
127     00 00FA                                         .L34:
128     00 00FA                                         .L32:
129     00 00FA  4CEE  2000  FFFC                           moveml  a6@(-_F4),#8192
130     00 0100  4E5E                                       unlk    a6
131     00 0102  4E75                                       rts
132                                                     _F4 = 4
133                                                     _S4 = 8192
134                                                     | M4 = 4
135                                                         .data
136                                                         .globl  chardigs
137     00 08F8                                         chardigs:
138     00 08F8  3031                                       .word   12337
139     00 08FA  3233                                       .word   12851
140     00 08FC  3435                                       .word   13365
141     00 08FE  3637                                       .word   13879
142     00 0900  3839                                       .word   14393
143     00 0902  4142                                       .word   16706
144     00 0904  4344                                       .word   17220
145     00 0906  4546                                       .word   17734
146     00 0908  0000                                       .word   0
147                                                         .text
148                                                         .globl  printhex
149     00 0104                                         printhex:
150     00 0104  4E56  FFF4                                 link    a6,#-_F5
151     00 0108  48EE  00E0  FFF4                           moveml  #_S5,a6@(-_F5)
152     00 010E  2E2E  0008                                 movl    a6@(8),d7
153     00 0112  2C2E  000C                                 movl    a6@(12),d6
154                                                     | A5 = 16
155     00 0116  5386                                       subql   #1,d6
156     00 0118  2006                                       movl    d6,d0
157     00 011A  0280  0000  0007                           andl    #7,d0
158     00 0120  E580                                       asll    #2,d0
159     00 0122  2A00                                       movl    d0,d5
160     00 0124                                         .L41:
161     00 0124  4A85                                       tstl    d5
162     00 0126  6D26                                       jlt     .L38
163     00 0128  2007                                       movl    d7,d0
164     00 012A  EAA0                                       asrl    d5,d0
165     00 012C  0280  0000  000F                           andl    #15,d0
166     00 0132  0680  0000  08F8                           addl    #chardigs,d0
167     00 0138  2040                                       movl    d0,a0
168     00 013A  1010                                       movb    a0@,d0
169     00 013C  4880                                       extw    d0
170     00 013E  48C0                                       extl    d0
171     00 0140  2F00                                       movl    d0,sp@-
172     00 0142  4EB9  0000  0000                       7   jbsr    putchar
173     00 0148  588F                                       addql   #4,sp
174     00 014A                                         .L39:
175     00 014A  5985                                       subql   #4,d5
176     00 014C  60D6                                       jra     .L41
177     00 014E                                         .L40:
178     00 014E                                         .L38:
179     00 014E  4CEE  00E0  FFF4                           moveml  a6@(-_F5),#224
180     00 0154  4E5E                                       unlk    a6
181     00 0156  4E75                                       rts
182                                                     _F5 = 12
183                                                     _S5 = 224
184                                                     | M5 = 4
185                                                         .text
186                                                         .globl  queryval
187     00 0158                                         queryval:
188     00 0158  4E56  0000                                 link    a6,#-_F6
189     00 015C  48EE  0000  0000                           moveml  #_S6,a6@(-_F6)
```

```
190                                                     | A6 = 20
191     00 0162  2F2E  0008                                movl    a6@(8),sp@-
192     00 0166  6100  FF6E                                jbsr    message
193     00 016A  588F                                      addql   #4,sp
194     00 016C  2F2E  0010                                movl    a6@(16),sp@-
195     00 0170  2F2E  000C                                movl    a6@(12),sp@-
196     00 0174  618E                                      jbsr    printhex
197     00 0176  508F                                      addql   #8,sp
198                                                        .text
199     00 0178  2F3C  0000  09C8                          movl    #.L44,sp@-
200     00 017E  6100  FF56                                jbsr    message
201     00 0182  588F                                      addql   #4,sp
202     00 0184  6100  FE7A                                jbsr    getline
203     00 0188                                         .L43:
204     00 0188  4E5E                                      unlk    a6
205     00 018A  4E75                                      rts
206                                                     _F6 = 0
207                                                     _S6 = 0
208                                                     | M6 = 8
209                                                        .globl  ishex
210     00 018C                                         ishex:
211     00 018C  4E56  FFFC                                link    a6,#-_F7
212     00 0190  48EE  0080  FFFC                          moveml  #_S7,a6@(-_F7)
213                                                     | A7 = 12
214     00 0196  1E2E  000B                                movb    a6@(11),d7
215     00 019A  0C07  0030                                cmpb    #48,d7
216     00 019E  6D14                                      jlt     .L47
217     00 01A0  0C07  0039                                cmpb    #57,d7
218     00 01A4  6E0E                                      jgt     .L47
219     00 01A6  1007                                      movb    d7,d0
220     00 01A8  4880                                      extw    d0
221     00 01AA  48C0                                      extl    d0
222     00 01AC  0480  0000  0030                          subl    #48,d0
223     00 01B2  6036                                      jra     .L46
224     00 01B4                                         .L47:
225     00 01B4  0C07  0061                                cmpb    #97,d7
226     00 01B8  6D14                                      jlt     .L48
227     00 01BA  0C07  0066                                cmpb    #102,d7
228     00 01BE  6E0E                                      jgt     .L48
229     00 01C0  1007                                      movb    d7,d0
230     00 01C2  4880                                      extw    d0
231     00 01C4  48C0                                      extl    d0
232     00 01C6  0480  0000  0057                          subl    #87,d0
233     00 01CC  601C                                      jra     .L46
234     00 01CE                                         .L48:
235     00 01CE  0C07  0041                                cmpb    #65,d7
236     00 01D2  6D14                                      jlt     .L49
237     00 01D4  0C07  0046                                cmpb    #70,d7
238     00 01D8  6E0E                                      jgt     .L49
239     00 01DA  1007                                      movb    d7,d0
240     00 01DC  4880                                      extw    d0
241     00 01DE  48C0                                      extl    d0
242     00 01E0  0480  0000  0037                          subl    #55,d0
243     00 01E6  6002                                      jra     .L46
244     00 01E8                                         .L49:
245     00 01E8  70FF                                      moveq   #-1,d0
246     00 01EA                                         .L46:
247     00 01EA  4CEE  0080  FFFC                          movemT  a6@(-_F7),#128
248     00 01F0  4E5E                                      unlk    a6
249     00 01F2  4E75                                      rts
250                                                     _F7 = 4
251                                                     _S7 = 128
252                                                     | M7 = 0
```

```
253                                                     .text
254                                                     .globl  gethexbyte
255     00 01F4                                 gethexbyte:
256     00 01F4  4E56  FFFC                             link    a6,#-_F8
257     00 01F8  48EE  0080  FFFC                       moveml  #_S8,a6@(-_F8
258                                                 | A8 = 8
259     00 01FE  4287                                   clrl    d7
260     00 0200  6100  FE8C                             jbsr    getone
261     00 0204  4880                                   extw    d0
262     00 0206  48C0                                   extl    d0
263     00 0208  2F00                                   movl    d0,sp@-
264     00 020A  6180                                   jbsr    ishex
265     00 020C  588F                                   addql   #4,sp
266     00 020E  E980                                   asll    #4,d0
267     00 0210  2E00                                   movl    d0,d7
268     00 0212  6100  FE7A                             jbsr    getone
269     00 0216  4880                                   extw    d0
270     00 0218  48C0                                   extl    d0
271     00 021A  2F00                                   movl    d0,sp@-
272     00 021C  6100  FF6E                             jbsr    ishex
273     00 0220  588F                                   addql   #4,sp
274     00 0222  8087                                   orl     d7,d0
275     00 0224                                 .L51:
276     00 0224  4CEE  0080  FFFC                       moveml  a6@(-_F8),#128
277     00 022A  4E5E                                   unlk    a6
278     00 022C  4E75                                   rts
279                                             _F8 = 4
280                                             _S8 = 128
281                                             | M8 = 4
282                                                     .text
283                                                     .globl  getnum
284     00 022E                                 getnum:
285     00 022E  4E56  FFF8                             link    a6,#-_F9
286     00 0232  48EE  00C0  FFF8                       moveml  #_S9,a6@(-_F9)
287                                                 | A9 = 8
288     00 0238  4287                                   clrl    d7
289     00 023A  600C                                   jra     .L54
290     00 023C                                 .L20003:
291     00 023C  2007                                   movl    d7,d0
292     00 023E  E980                                   asll    #4,d0
293     00 0240  8086                                   orl     d6,d0
294     00 0242  2E00                                   movl    d0,d7
295     00 0244  6100  FE48                             jbsr    getone
296     00 0248                                 .L20002:
297     00 0248                                 .L54:
298     00 0248  6100  FE76                             jbsr    peekchar
299     00 024C  4880                                   extw    d0
300     00 024E  48C0                                   extl    d0
301     00 0250  2F00                                   movl    d0,sp@-
302     00 0252  6100  FF38                             jbsr    ishex
303     00 0256  588F                                   addql   #4,sp
304     00 0258  2C00                                   movl    d0,d6
305     00 025A  6CE0                                   jge     .L20003
306     00 025C                                 .L55:
307     00 025C  2007                                   movl    d7,d0
308     00 025E                                 .L53:
309     00 025E  4CEE  00C0  FFF8                       moveml  a6@(-_F9),#192
310     00 0264  4E5E                                   unlk    a6
311     00 0266  4E75                                   rts
312                                             _F9 = 8
313                                             _S9 = 192
314                                             | M9 = 4
315                                                     .text
```

```
316                                                     .globl  openitem
317     00 0268                                 openitem:
318     00 0268  4E56  FFF0                             link    a6,#-_F10
319     00 026C  48EE  20E0  FFF0                       moveml  #_S10,a6@(-_F1
320     00 0272  2E2E  0008                             movl    a6@(8),d7
321     00 0276  2A6E  000C                             movl    a6@(12),a5
322     00 027A  2C2E  0010                             movl    a6@(16),d6
323                                             | A10 = 24
324     00 027E  2F06                                   movl    d6,sp@-
325     00 0280  2F07                                   movl    d7,sp@-
326     00 0282  6100  FE80                             jbsr    printhex
327     00 0286  508F                                   addql   #8,sp
328     00 0288  0CAE  0000  0002  0014                 cmpl    #2,a6@(20)
329     00 0290  6614                                   jne     .L58
330                                                     .text
331     00 0292  2F3C  0000  0004                       movl    #4,sp@-
332     00 0298  3015                                   movw    a5@,d0
333     00 029A  48C0                                   extl    d0
334     00 029C  2F00                                   movl    d0,sp@-
335     00 029E  2F3C  0000  09CB                       movl    #.L59,sp@-
336     00 02A4  6014                                   jra     .L20005
337     00 02A6                                 .L58:
338     00 02A6  2F3C  0000  0002                       movl    #2,sp@-
339     00 02AC  1015                                   movb    a5@,d0
340     00 02AE  4880                                   extw    d0
341     00 02B0  48C0                                   extl    d0
342     00 02B2  2F00                                   movl    d0,sp@-
343     00 02B4  2F3C  0000  09CE                       movl    #.L61,sp@-
344     00 02BA                                 .L20005:
345     00 02BA  6100  FE9C                             jbsr    queryval
346     00 02BE                                 .L20004:
347     00 02BE  DFFC  0000  000C                       addl    #12,sp
348     00 02C4                                 .L60:
349     00 02C4  6100  FDFA                             jbsr    peekchar
350     00 02C8  1A00                                   movb    d0,d5
351     00 02CA  1005                                   movb    d5,d0
352     00 02CC  4880                                   extw    d0
353     00 02CE  48C0                                   extl    d0
354     00 02D0  2F00                                   movl    d0,sp@-
355     00 02D2  6100  FEB8                             jbsr    ishex
356     00 02D6  588F                                   addql   #4,sp
357     00 02D8  4A80                                   tstl    d0
358     00 02DA  6D1C                                   jlt     .L62
359     00 02DC  0CAE  0000  0002  0014                 cmpl    #2,a6@(20)
360     00 02E4  6608                                   jne     .L63
361     00 02E6  6100  FF46                             jbsr    getnum
362     00 02EA  3A80                                   movw    d0,a5@
363     00 02EC  6006                                   jra     .L64
364     00 02EE                                 .L63:
365     00 02EE  6100  FF3E                             jbsr    getnum
366     00 02F2  1A80                                   movb    d0,a5@
367     00 02F4                                 .L64:
368     00 02F4  7001                                   moveq   #1,d0
369     00 02F6  6008                                   jra     .L57
370     00 02F8                                 .L62:
371     00 02F8  0C05  000D                             cmpb    #13,d5
372     00 02FC  6702                                   jeq     .L57
373     00 02FE  4280                                   clrl    d0
374     00 0300                                 .L65:
375     00 0300                                 .L57:
376     00 0300  4CEE  20E0  FFF0                       moveml  a6@(-_F10),#84
377     00 0306  4E5E                                   unlk    a6
378     00 0308  4E75                                   rts
```

```
379                                                    _F10 = 16
380                                                    _S10 = 8416
381                                                    | M10 = 12
382                                                        .globl  getrecord
383     00 030A                                        getrecord:
384     00 030A  4E56  FFE4                                link    a6,#-_F11
385     00 030E  48EE  20E0  FFE4                          moveml  #_S11,a6@(-_F1
386                                                    | A11 = 12
387     00 0314  2A7C  0000  0200                          movl    #512,a5
388     00 031A  2F3C  0000  0002                          movl    #2,sp@-
389     00 0320  2F2D  0062                                movl    a5@(98),sp@-
390     00 0324  6100  FDDE                                jbsr    printhex
391     00 0328  508F                                      addql   #8,sp
392     00 032A  6100  FD62                                jbsr    getone
393     00 032E  1D40  FFFF                                movb    d0,a6@(-1)
394     00 0332  0C2E  0032  FFFF                          cmpb    #50,a6@(-1)
395     00 0338  670E                                      jeq     .L68
396     00 033A  0C2E  0038  FFFF                          cmpb    #56,a6@(-1)
397     00 0340  6706                                      jeq     .L68
398     00 0342  7054                                      moveq   #84,d0
399     00 0344  6000  00AE                                jra     .L67
400     00 0348                                        .L68:
401     00 0348  6100  FEAA                                jbsr    gethexbyte
402     00 034C  2E00                                      movl    d0,d7
403     00 034E  3D47  FFF8                                movw    d7,a6@(-8)
404     00 0352  0C87  0000  0003                          cmpl    #3,d7
405     00 0358  6D0E                                      jlt     .L10002
406     00 035A  2007                                      movl    d7,d0
407     00 035C  E380                                      asll    #1,d0
408     00 035E  5880                                      addql   #4,d0
409     00 0360  222D  005E                                movl    a5@(94),d1
410     00 0364  B280                                      cmpl    d0,d1
411     00 0366  6706                                      jeq     .L69
412     00 0368                                        .L10002:
413     00 0368  704C                                      moveq   #76,d0
414     00 036A  6000  0088                                jra     .L67
415     00 036E                                        .L69:
416     00 036E  4285                                      clrl    d5
417     00 0370  42AE  FFFA                                clrl    a6@(-6)
418     00 0374  6014                                      jra     .L72
419     00 0376                                        .L20006:
420     00 0376  6100  FE7C                                jbsr    gethexbyte
421     00 037A  2C00                                      movl    d0,d6
422     00 037C  2205                                      movl    d5,d1
423     00 037E  E181                                      asll    #8,d1
424     00 0380  8280                                      orl     d0,d1
425     00 0382  2A01                                      movl    d1,d5
426     00 0384  DD6E  FFF8                                addw    d6,a6@(-8)
427     00 0388  5387                                      subql   #1,d7
428     00 038A                                        .L70:
429     00 038A                                        .L72:
430     00 038A  202E  FFFA                                movl    a6@(-6),d0
431     00 038E  52AE  FFFA                                addql   #1,a6@(-6)
432     00 0392  0C80  0000  0003                          cmpl    #3,d0
433     00 0398  6DDC                                      jlt     .L20006
434     00 039A                                        .L71:
435     00 039A  2D6D  005A  FFF4                          movl    a5@(90),a6@(-1
436     00 03A0  2C07                                      movl    d7,d6
437     00 03A2  600A                                      jra     .L75
438     00 03A4                                        .L20008:
439     00 03A4  6100  FE4E                                jbsr    gethexbyte
440     00 03A8  D16E  FFF8                                addw    d0,a6@(-8)
441     00 03AC                                        .L73:
```

```
442     00 03AC  5386                          subql   #1,d6
443     00 03AE                              .L20007:
444     00 03AE                              .L75:
445     00 03AE  4A86                          tstl    d6
446     00 03B0  6EF2                          jgt     .L20008
447     00 03B2                              .L74:
448     00 03B2  526E  FFF8                    addqw   #1,a6@(-8)
449     00 03B6  302E  FFF8                    movw    a6@(-8),d0
450     00 03BA  48C0                          extl    d0
451     00 03BC  0280  0000  00FF              andl    #255,d0
452     00 03C2  6704                          jeq     .L76
453     00 03C4  704B                          moveq   #75,d0
454     00 03C6  602C                          jra     .L67
455     00 03C8                              .L76:
456     00 03C8  0C2E  0032  FFFF              cmpb    #50,a6@(-1)
457     00 03CE  6618                          jne     .L77
458     00 03D0  2B6E  FFF4  005A              movl    a6@(-12),a5@(90
459     00 03D6                              .L78:
460     00 03D6  5387                          subql   #1,d7
461     00 03D8  6714                          jeq     .L80
462     00 03DA  6100  FE18                    jbsr    gethexbyte
463     00 03DE  2045                          movl    d5,a0
464     00 03E0  1080                          movb    d0,a0@
465     00 03E2  5285                          addql   #1,d5
466     00 03E4  60F0                          jra     .L78
467     00 03E6                              .L79:
468     00 03E6  6006                          jra     .L80
469     00 03E8                              .L77:
470     00 03E8  206E  0008                    movl    a6@(8),a0
471     00 03EC  2085                          movl    d5,a0@
472     00 03EE                              .L80:
473     00 03EE  52AD  0062                    addql   #1,a5@(98)
474     00 03F2  7059                          moveq   #89,d0
475     00 03F4                              .L67:
476     00 03F4  4CEE  20E0  FFE4              moveml  a6@(-_F11),#841
477     00 03FA  4E5E                          unlk    a6
478     00 03FC  4E75                          rts
479                                          _F11 = 28
480                                          _S11 = 8416
481                                          | M11 = 8
482                                              .data
483                                              .globl  _reg_nam
484     00 090A                              _reg_nam:
485     00 090A  4430                            .word   17456
486     00 090C  0044                            .word   68
487     00 090E  3100                            .word   12544
488     00 0910  4432                            .word   17458
489     00 0912  0044                            .word   68
490     00 0914  3300                            .word   13056
491     00 0916  4434                            .word   17460
492     00 0918  0044                            .word   68
493     00 091A  3500                            .word   13568
494     00 091C  4436                            .word   17462
495     00 091E  0044                            .word   68
496     00 0920  3700                            .word   14080
497     00 0922  4130                            .word   16688
498     00 0924  0041                            .word   65
499     00 0926  3100                            .word   12544
500     00 0928  4132                            .word   16690
501     00 092A  0041                            .word   65
502     00 092C  3300                            .word   13056
503     00 092E  4134                            .word   16692
504     00 0930  0041                            .word   65
```

```
505     00 0932  3500                                    .word   13568
506     00 0934  4136                                    .word   16694
507     00 0936  0053                                    .word   83
508     00 0938  5300                                    .word   21248
509     00 093A  5553                                    .word   21843
510     00 093C  0053                                    .word   83
511     00 093E  5200                                    .word   20992
512     00 0940  5043                                    .word   20547
513     00 0942  0000                                    .word   0
514                                                      .text
515                                                      .globl  openreg
516     00 03FE                                  openreg:
517     00 03FE  4E56  FFFC                              link    a6,#-_F12
518     00 0402  48EE  0080  FFFC                        moveml  #_S12,a6@(-_F1
519                                              | A12 = 16
520     00 0408  6100  FE24                              jbsr    getnum
521     00 040C  0280  0000  000F                        andl    #15,d0
522     00 0412  D16E  000A                              addw    d0,a6@(10)
523     00 0416  302E  000A                              movw    a6@(10),d0
524     00 041A  48C0                                    extl    d0
525     00 041C  E580                                    asll    #2,d0
526     00 041E  D1AE  000C                              addl    d0,a6@(12)
527     00 0422                                  .L86:
528     00 0422  0C6E  0012  000A                        cmpw    #18,a6@(10)
529     00 0428  6E7A                                    jgt     .L83
530     00 042A  302E  000A                              movw    a6@(10),d0
531     00 042E  48C0                                    extl    d0
532     00 0430  322E  000A                              movw    a6@(10),d1
533     00 0434  48C1                                    extl    d1
534     00 0436  D081                                    addl    d1,d0
535     00 0438  322E  000A                              movw    a6@(10),d1
536     00 043C  48C1                                    extl    d1
537     00 043E  D081                                    addl    d1,d0
538     00 0440  0680  0000  090A                        addl    #_reg_nam,d0
539     00 0446  2F00                                    movl    d0,sp@-
540     00 0448  6100  FC8C                              jbsr    message
541     00 044C  588F                                    addql   #4,sp
542                                                      .text
543     00 044E  2F3C  0000  0008                        movl    #8,sp@-
544     00 0454  206E  000C                              movl    a6@(12),a0
545     00 0458  2F10                                    movl    a0@,sp@-
546     00 045A  2F3C  0000  09D1                        movl    #.L87,sp@-
547     00 0460  6100  FCF6                              jbsr    queryval
548     00 0464  DFFC  0000  000C                        addl    #12,sp
549     00 046A  6100  FC54                              jbsr    peekchar
550     00 046E  1E00                                    movb    d0,d7
551     00 0470  1007                                    movb    d7,d0
552     00 0472  4880                                    extw    d0
553     00 0474  48C0                                    extl    d0
554     00 0476  2F00                                    movl    d0,sp@-
555     00 0478  6100  FD12                              jbsr    ishex
556     00 047C  588F                                    addql   #4,sp
557     00 047E  4A80                                    tstl    d0
558     00 0480  6D1C                                    jlt     .L88
559     00 0482  0C6E  000F  000A                        cmpw    #15,a6@(10)
560     00 0488  670A                                    jeq     .L84
561     00 048A  6100  FDA2                              jbsr    getnum
562     00 048E  206E  000C                              movl    a6@(12),a0
563     00 0492  2080                                    movl    d0,a0@
564     00 0494                                  .L89:
565     00 0494                                  .L91:
566     00 0494                                  .L90:
567     00 0494                                  .L84:
```

```
568     00 0494  526E  000A                          addqw   #1,a6@(10)
569     00 0498  58AE  000C                          addql   #4,a6@(12)
570     00 049C  6084                                jra     .L86
571     00 049E                                   .L88:
572     00 049E  0C07  000D                          cmpb    #13,d7
573     00 04A2  67F0                                jeq     .L84
574     00 04A4                                   .L86:
575     00 04A4                                   .L83:
576     00 04A4  4CEE  0080  FFFC                    moveml  a6@(-_F12),#1
577     00 04AA  4E5E                                unlk    a6
578     00 04AC  4E75                                rts
579                                               _F12 = 4
580                                               _S12 = 128
581                                               | M12 = 12
582                                                  .globl  doload
583     00 04AE                                   doload:
584     00 04AE  4E56  FFFC                          link    a6,#-_F13
585     00 04B2  48EE  2000  FFFC                    moveml  #_S13,a6@(-_F
586                                               | A13 = 8
587     00 04B8  2A7C  0000  0200                    movl    #512,a5
588     00 04BE  2F3C  0000  0042                    movl    #66,sp@-
589     00 04C4  613A                                jbsr    setmode
590     00 04C6  588F                                addql   #4,sp
591     00 04C8  42AD  0086                          clrl    a5@(134)
592     00 04CC  1B7C  000D  0008                    movb    #13,a5@(8)
593     00 04D2  486D  0008                          pea     a5@(8)
594     00 04D6  6100  FBFE                          jbsr    message
595     00 04DA  588F                                addql   #4,sp
596     00 04DC  42AD  0062                          clrl    a5@(98)
597     00 04E0                                   .L95:
598     00 04E0  4EB9  0000  0000                7   jbsr    getchar
599     00 04E6  0C80  0000  005C                    cmpl    #92,d0
600     00 04EC  66F2                                jne     .L95
601     00 04EE                                   .L96:
602     00 04EE  2B7C  0000  0001  0086              movl    #1,a5@(134)
603     00 04F6                                   .L93:
604     00 04F6  4CEE  2000  FFFC                    moveml  a6@(-_F13),#8
605     00 04FC  4E5E                                unlk    a6
606     00 04FE  4E75                                rts
607                                               _F13 = 4
608                                               _S13 = 8192
609                                               | M13 = 4
610                                                  .text
611                                                  .globl  setmode
612     00 0500                                   setmode:
613     00 0500  4E56  FFF4                          link    a6,#-_F14
614     00 0504  48EE  20C0  FFF4                    moveml  #_S14,a6@(-_F1
615     00 050A  1E2E  000B                          movb    a6@(11),d7
616                                               | A14 = 12
617     00 050E  2A7C  0000  0200                    movl    #512,a5
618     00 0514  1C2D  0085                          movb    a5@(133),d6
619     00 0518  2B7C  0000  0001  0086              movl    #1,a5@(134)
620     00 0520  0C07  0041                          cmpb    #65,d7
621     00 0524  660C                                jne     .L98
622     00 0526  42AD  0082                          clrl    a5@(130)
623     00 052A  4A06                                tstb    d6
624     00 052C  6620                                jne     .L20011
625     00 052E  602C                                jra     .L97
626     00 0530                                   .L99:
627     00 0530  602A                                jra     .L97
628     00 0532                                   .L98:
629     00 0532  0C07  0042                          cmpb    #66,d7
630     00 0536  660A                                jne     .L101
```

```
631     00 0538  2B7C  0000  0004  0082                movl    #4,a5@(130)
632     00 0540  601A                                  jra     .L97
633     00 0542                                      .L101:
634     00 0542  0C07  0054                            cmpb    #84,d7
635     00 0546  6614                                  jne     .L97
636     00 0548  4EB9  0000  0000                7     jbsr    transparent
637     00 054E                                      .L20011:
638     00 054E  2F3C  0000  000A                      movl    #10,sp@-
639     00 0554                                      .L20010:
640     00 0554  4EB9  0000  0000                7     jbsr    putchar
641     00 055A                                      .L20009:
642     00 055A  588F                                  addql   #4,sp
643     00 055C                                      .L103:
644     00 055C                                      .L102:
645     00 055C                                      .L100;
646     00 055C                                      .L97:
647     00 055C  4CEE  20C0  FFF4                      moveml  a6@(-_F14),#8.
648     00 0562  4E5E                                  unlk    a6
649     00 0564  4E75                                  rts
650                                                  _F14 = 12
651                                                  _S14 = 8384
652                                                  | M14 = 4
653                                                      .text
654                                                      .globl  givehelp
655     00 0566                                      givehelp:
656     00 0566  4E56  0000                            link    a6,#-_F15
657     00 056A  48EE  0000  0000                      moveml  #_S15,a6@(-_F1
658                                                  | A15 = 8
659                                                      .text
660     00 0570  2F3C  0000  09D4                      movl    #.L107,sp@-
661     00 0576  6100  FB5E                            jbsr    message
662     00 057A  588F                                  addql   #4,sp
663     00 057C  2F3C  0000  0A03                      movl    #.L108,sp@-
664     00 0582  6100  FB52                            jbsr    message
665     00 0586  588F                                  addql   #4,sp
666     00 0588  2F3C  0000  0A30                      movl    #.L109,sp@-
667     00 058E  6100  FB46                            jbsr    message
668     00 0592  588F                                  addql   #4,sp
669     00 0594  2F3C  0000  0A58                      movl    #.L110,sp@-
670     00 059A  6100  FB3A                            jbsr    message
671     00 059E  588F                                  addql   #4,sp
672     00 05A0  2F3C  0000  0A83                      movl    #.L111,sp@-
673     00 05A6  6100  FB2E                            jbsr    message
674     00 05AA  588F                                  addql   #4,sp
675     00 05AC                                      .L106:
676     00 05AC  4E5E                                  unlk    a6
677     00 05AE  4E75                                  rts
678                                                  _F15 = 0
679                                                  _S15 = 0
680                                                  | M15 = 4
681                                                      .globl  monitor
682     00 05B0                                      monitor:
683     00 05B0  4E56  FFEC                            link    a6,#-_F16
684     00 05B4  48EE  20C0  FFEC                      moveml  #_S16,a6@(-_F1
685                                                  | A16 = 84
686     00 05BA  2A7C  0000  0200                      movl    #512,a5
687     00 05C0  202E  004C                            movl    a6@(76),d0
688     00 05C4  7210                                  moveq   #16,d1
689     00 05C6  E2A0                                  asrl    d1,d0
690     00 05C8  2D40  FFFC                            movl    d0,a6@(-4)
691     00 05CC  0C80  0000  0005                      cmpl    #5,d0
692     00 05D2  670C                                  jeq     .L114
693     00 05D4  2F3C  0000  0041                      movl    #65,sp@-
```

```
694     00 05DA  6100  FF24                        jbsr    setmode
695     00 05DE  588F                              addql   #4,sp
696     00 05E0                                 .L114:
697     00 05E0  202E  FFFC                        movl    a6@(-4),d0
698     00 05E4                                 .L116:
699     00 05E4  0C80  0000  0007                  cmpl    #7,d0
700     00 05EA  6200  00DA                        jhi     .L136
701     00 05EE  D040                              addw    d0,d0
702     00 05F0  D040                              addw    d0,d0
703     00 05F2  207C  0000  0944                  movl    #.L138,a0
704     00 05F8  2070  0000                        movl    a0@(0,d0:w),a0
705     00 05FC  4ED0                              jmp     a0@
706     00 05FE                                 .L117:
707                                                .text
708     00 05FE  2F3C  0000  0AB1                  movl    #.L118,sp@-
709     00 0604  6100  FAD0                        jbsr    message
710     00 0608  588F                              addql   #4,sp
711     00 060A  2F3C  0000  0005                  movl    #5,sp@-
712     00 0610  2F2D  0078                        movl    a5@(120),sp@-
713     00 0614  6100  FAEE                        jbsr    printhex
714     00 0618  508F                              addql   #8,sp
715     00 061A  2F3C  0000  0ACF                  movl    #.L119,sp@-
716     00 0620  6100  FAB4                        jbsr    message
717     00 0624  588F                              addql   #4,sp
718     00 0626  2D6D  0078  0048                  movl    a5@(120),a6@(72
719     00 062C  1B7C  001E  0080                  movb    #30,a5@(128)
720     00 0632  6000  0088                        jra     .L115
721     00 0636                                 .L120:
722     00 0636  2F3C  0000  0AE1                  movl    #.L121,sp@-
723     00 063C                                 .L20017:
724     00 063C  6100  FA98                        jbsr    message
725     00 0640                                 .L20016:
726     00 0640  588F                              addql   #4,sp
727     00 0642  6058                              jra     .L122
728     00 0644                                 .L123:
729     00 0644  2F3C  0000  0AE8                  movl    #.L124,sp@-
730     00 064A  6100  FA8A                        jbsr    message
731     00 064E                                 .L20018:
732     00 064E  588F                              addql   #4,sp
733     00 0650  6000  009A                        jra     .L115
734     00 0654                                 .L125:
735     00 0654  6000  0096                        jra     .L115
736     00 0658                                 .L126:
737     00 0658  2F3C  0000  0AEF                  movl    #.L127,sp@-
738     00 065E  60DC                              jra     .L20017
739     00 0660                                 .L128:
740     00 0660  2D7C  FFFF  FFFF  0008            movl    #-1,a6@(8)
741     00 0668  6000  0284                        jra     .L113
742     00 066C                                 .L129:
743     00 066C  2F3C  0000  0AFB                  movl    #.L130,sp@-
744     00 0672  6006                              jra     .L20013
745     00 0674                                 .L132:
746     00 0674  2F3C  0000  0AFF                  movl    #.L133,sp@-
747     00 067A                                 .L20013:
748     00 067A  6100  FA5A                        jbsr    message
749     00 067E                                 .L20012:
750     00 067E  588F                              addql   #4,sp
751     00 0680                                 .L131:
752     00 0680  2F3C  0000  0B07                  movl    #.L134,sp@-
753     00 0686  6100  FA4E                        jbsr    message
754     00 068A  588F                              addql   #4,sp
755     00 068C  2F3C  0000  0008                  movl    #8,sp@-
756     00 0692  2F2D  0002                        movl    a5@(2),sp@-
```

```
820     00 0726  0280  0000  005F                    andl    #95,d0
821     00 072C                                  .L150:
822     00 072C  0480  0000  0041                    subl    #65,d0
823     00 0732  0C80  0000  0017                    cmpl    #23,d0
824     00 0738  6200  01A2                          jhi     .L181
825     00 073C  D040                                addw    d0,d0
826     00 073E  D040                                addw    d0,d0
827     00 0740  207C  0000  0964                    movl    #.L183,a0
828     00 0746  2070  0000                          movl    a0@(0,d0:w),a0
829     00 074A  4ED0                                jmp     a0@
830     00 074C                                  .L20014:
831     00 074C  6100  F940                          jbsr    getone
832     00 0750  60C0                                jra     .L146
833     00 0752                                  .L151:
834     00 0752  486E  0008                          pea     a6@(8)
835     00 0756  2F3C  0000  0008                    movl    #8,sp@-
836     00 075C                                  .L20022:
837     00 075C  6100  FCA0                          jbsr    openreg
838     00 0760                                  .L20021:
839     00 0760  508F                                addql   #8,sp
840     00 0762  6090                                jra     .L141
841     00 0764                                  .L152:
842     00 0764  6100  FAC8                          jbsr    getnum
843     00 0768  2C00                                movl    d0,d6
844     00 076A  6700  0182                          jeq     .L113
845     00 076E  2D46  0050                          movl    d6,a6@(80)
846     00 0772                                  .L153:
847     00 0772  6000  017A                          jra     .L113
848     00 0776                                  .L154:
849     00 0776  486E  0008                          pea     a6@(8)
850     00 077A  2F3C  0000  0000                    movl    #0,sp@-
851     00 0780  60DA                                jra     .L20022
852     00 0782                                  .L155:
853     00 0782  6100  FAAA                          jbsr    getnum
854     00 0786  0280  00FF  FFFE                    andl    #16777214,d0
855     00 078C  2C00                                movl    d0,d6
856     00 078E  6002                                jra     .L158
857     00 0790                                  .L159:
858     00 0790                                  .L156:
859     00 0790  5486                                addql   #2,d6
860     00 0792                                  .L20015:
861     00 0792                                  .L158:
862     00 0792  2F3C  0000  0002                    movl    #2,sp@-
863     00 0798  2F3C  0000  0006                    movl    #6,sp@-
864     00 079E  2F06                                movl    d6,sp@-
865     00 07A0  2F06                                movl    d6,sp@-
866     00 07A2  6100  FAC4                          jbsr    openitem
867     00 07A6  DFFC  0000  0010                    addl    #16,sp
868     00 07AC  4A80                                tstl    d0
869     00 07AE  66E0                                jne     .L156
870     00 07B0                                  .L157:
871     00 07B0  6000  FF42                          jra     .L141
872     00 07B4                                  .L160:
873     00 07B4  6100  FA78                          jbsr    getnum
874     00 07B8  2D40  FFF8                          movl    d0,a6@(-8)
875     00 07BC  6606                                jne     .L161
876     00 07BE  2D6E  0050  FFF8                    movl    a6@(80),a6@(-8
877     00 07C4                                  .L161:
878     00 07C4  206E  FFF8                          movl    a6@(-8),a0
879     00 07C8  4E90                                jbsr    a0@
880     00 07CA  6000  FF28                          jra     .L141
881     00 07CE                                  .L162:
882     00 07CE  6100  F8BE                          jbsr    getone
```

```
757     00 0696  6100  FA6C                              jbsr    printhex
758     00 069A  508F                                     addql   #8,sp
759     00 069C                                         .L122:
760     00 069C  2F3C  0000  0B16                         movl    #.L135,sp@-
761     00 06A2  6100  FA32                               jbsr    message
762     00 06A6  588F                                     addql   #4,sp
763     00 06A8  2F3C  0000  0006                         movl    #6,sp@-
764     00 06AE  2F2E  0050                               movl    a6@(80),sp@-
765     00 06B2  6100  FA50                               jbsr    printhex
766     00 06B6  508F                                     addql   #8,sp
767     00 06B8                                         .L20020:
768     00 06B8  2F3C  0000  000A                         movl    #10,sp@-
769     00 06BE                                         .L20019:
770     00 06BE  4EB9  0000  0000                   7     jbsr    putchar
771     00 06C4  6088                                     jra     .L20018
772     00 06C6                                         .L136:
773     00 06C6  202E  004C                               movl    a6@(76),d0
774     00 06CA  0280  FFFF  0000                         andl    #-65536,d0
775     00 06D0  2D40  FFFC                               movl    d0,a6@(-4)
776     00 06D4  2F3C  0000  0B1B                         movl    #.L137,sp@-
777     00 06DA  6100  F9FA                               jbsr    message
778     00 06DE  588F                                     addql   #4,sp
779     00 06E0  486E  FFFC                               pea     a6@(-4)
780     00 06E4  6100  F9F0                               jbsr    message
781     00 06E8  588F                                     addql   #4,sp
782     00 06EA  60CC                                     jra     .L20020
783                                                       .data
784     00 0944                                         .L138:
785     00 0944  0000  05FE                               .long   .L117
786     00 0948  0000  0636                               .long   .L120
787     00 094C  0000  0644                               .long   .L123
788     00 0950  0000  06EC                               .long   .L115
789     00 0954  0000  0658                               .long   .L126
790     00 0958  0000  0660                               .long   .L128
791     00 095C  0000  066C                               .long   .L129
792     00 0960  0000  0674                               .long   .L132
793                                                       .text
794     00 06EC                                         .L115:
795     00 06EC  02AE  0000  FFFF  004C                   andl    #65535,a6@(76)
796     00 06F4                                         .L141:
797     00 06F4  2F3C  0000  0B28                         movl    #.L142,sp@-
798     00 06FA  6100  F9DA                               jbsr    message
799     00 06FE  588F                                     addql   #4,sp
800     00 0700  6100  F8FE                               jbsr    getline
801     00 0704                                         .L145:
802     00 0704                                         .L143:
803     00 0704  6100  F988                               jbsr    getone
804     00 0708  1E00                                     movb    d0,d7
805     00 070A  6706                                     jeq     .L146
806     00 070C  0C07  0030                               cmpb    #48,d7
807     00 0710  6DF2                                     jlt     .L143
808     00 0712                                         .L144:
809     00 0712                                         .L146:
810     00 0712  6100  F9AC                               jbsr    peekchar
811     00 0716  0C00  0020                               cmpb    #32,d0
812     00 071A  6730                                     jeq     .L20014
813     00 071C                                         .L147:
814     00 071C  4A07                                     tstb    d7
815     00 071E  67D4                                     jeq     .L141
816     00 0720                                         .L148:
817     00 0720  1007                                     movb    d7,d0
818     00 0722  4880                                     extw    d0
819     00 0724  48C0                                     extl    d0
```

```
883     00 07D2  4880                                    extw    d0
884     00 07D4  48C0                                    extl    d0
885     00 07D6  0280  0000  005F                        andl    #95,d0
886     00 07DC  2F00                                    movl    d0,sp@-
887     00 07DE  6100  FD20                              jbsr    setmode
888     00 07E2                                        .L20023:
889     00 07E2  588F                                    addql   #4,sp
890     00 07E4  6000  FF0E                              jra     .L141
891     00 07E8                                        .L163:
892     00 07E8  2E7C  0000  1000                        movl    #4096,sp
893                                                     .text;
894     00 07EE  4EF9  0000  0000                  7     jmp     SoftReset
895     00 07F4                                        .L165:
896     00 07F4  6100  FCB8                              jbsr    doload
897     00 07F8  6000  FEFA                              jra     .L141
898     00 07FC                                        .L166:
899     00 07FC  6100  FA30                              jbsr    getnum
900     00 0800  0280  0000  003F                        andl    #63,d0
901     00 0806  2C00                                    movl    d0,d6
902     00 0808                                        .L169:
903     00 0808  0C86  0000  0040                        cmpl    #64,d6
904     00 080E  6C00  FEE4                              jge     .L141
905     00 0812  2F3C  0000  0B2A                        movl    #.L170,sp@-
906     00 0818  6100  F88C                              jbsr    message
907     00 081C  588F                                    addql   #4,sp.
908     00 081E  2F3C  0000  0002                        movl    #2,sp@-
909     00 0824  2F3C  0000  0004                        movl    #4,sp@-
910     00 082A  2006                                    movl    d6,d0
911     00 082C  720F                                    moveq   #15,d1
912     00 082E  E3A0                                    asll    d1,d0
913     00 0830  0680  00C0  0000                        addl    #12582912,d0
914     00 0836  2F00                                    movl    d0,sp@-
915     00 0838  2F06                                    movl    d6,sp@-
916     00 083A  6100  FA2C                              jbsr    openitem
917     00 083E  DFFC  0000  0010                        addl    #16,sp
918     00 0844  4A80                                    tstl    d0
919     00 0846  6700  FEAC                              jeq     .L141
920     00 084A                                        .L171:
921     00 084A                                        .L167:
922     00 084A  5286                                    addql   #1,d6
923     00 084C  60BA                                    jra     .L169
924     00 084E                                        .L168:
925     00 084E  6000  FEA4                              jra     .L141
926     00 0852                                        .L172:
927     00 0852  6100  F9DA                              jbsr    getnum
928     00 0856  0280  0000  03FF                        andl    #1023,d0
929     00 085C  2C00                                    movl    d0,d6
930     00 085E                                        .L175:
931     00 085E  0C86  0000  0400                        cmpl    #1024,d6
932     00 0864  6C00  FE8E                              jge     .L141
933     00 0868  2F3C  0000  0B32                        movl    #.L176,sp@-
934     00 086E  6100  F866                              jbsr    message
935     00 0872  588F                                    addql   #4,sp
936     00 0874  2F3C  0000  0002                        movl    #2,sp@-
937     00 087A  2F3C  0000  0004                        movl    #4,sp@-
938     00 0880  2006                                    movl    d6,d0
939     00 0882  720B                                    moveq   #11,d1
940     00 0884  E3A0                                    asll    d1,d0
941     00 0886  0680  00A0  0000                        addl    #10485760,d0
942     00 088C  2F00                                    movl    d0,sp@-
943     00 088E  2F06                                    movl    d6,sp@-
944     00 0890  6100  F9D6                              jbsr    openitem
945     00 0894  DFFC  0000  0010                        addl    #16,sp
```

```
 946    00 089A  4A80                          tstl    d0
 947    00 089C  6700  FE56                    jeq     .L141
 948    00 08A0                              .L177:
 949    00 08A0                              .L173:
 950    00 08A0  5286                          addql   #1,d6
 951    00 08A2  60BA                          jra     .L175
 952    00 08A4                              .L174:
 953    00 08A4  6000  FE4E                    jra     .L141
 954    00 08A8                              .L178:
 955    00 08A8  486E  0008                    pea     a6@(8)
 956    00 08AC  2F3C  0000  000F              movl    #15,sp@-
 957    00 08B2  6000  FEA8                    jra     .L20022
 958    00 08B6                              .L179:
 959    00 08B6  486E  0050                    pea     a6@(80)
 960    00 08BA  6100  FA4E                    jbsr    getrecord
 961    00 08BE  588F                          addql   #4,sp
 962    00 08C0  4880                          extw    d0
 963    00 08C2  48C0                          extl    d0
 964    00 08C4  2F00                          movl    d0,sp@-
 965    00 08C6  4EB9  0000  0000            7 jbsr    putchar
 966    00 08CC  6000  FF14                    jra     .L20023
 967    00 08D0                              .L180:
 968    00 08D0  6100  F7BC                    jbsr    getone
 969    00 08D4  1B40  0080                    movb    d0,a5@(128)
 970    00 08D8  6000  FE1A                    jra     .L141
 971    00 08DC                              .L181:
 972    00 08DC  2F3C  0000  0B3B              movl    #.L182,sp@-
 973    00 08E2  6100  F7F2                    jbsr    message
 974    00 08E6  6000  FEFA                    jra     .L20023
 975                                           .data
 976    00 0964                              .L183:
 977    00 0964  0000  0752                    .long   .L151
 978    00 0968  0000  08DC                    .long   .L181
 979    00 096C  0000  0764                    .long   .L152
 980    00 0970  0000  0776                    .long   .L154
 981    00 0974  0000  0782                    .long   .L155
 982    00 0978  0000  08DC                    .long   .L181
 983    00 097C  0000  07B4                    .long   .L160
 984    00 0980  0000  08DC                    .long   .L181
 985    00 0984  0000  07CE                    .long   .L162
 986    00 0988  0000  08DC                    .long   .L181
 987    00 098C  0000  07E8                    .long   .L163
 988    00 0990  0000  07F4                    .long   .L165
 989    00 0994  0000  07FC                    .long   .L166
 990    00 0998  0000  08DC                    .long   .L181
 991    00 099C  0000  08DC                    .long   .L181
 992    00 09A0  0000  0852                    .long   .L172
 993    00 09A4  0000  08DC                    .long   .L181
 994    00 09A8  0000  08A8                    .long   .L178
 995    00 09AC  0000  08B6                    .long   .L179
 996    00 09B0  0000  08DC                    .long   .L181
 997    00 09B4  0000  08DC                    .long   .L181
 998    00 09B8  0000  08DC                    .long   .L181
 999    00 09BC  0000  08DC                    .long   .L181
1000    00 09C0  0000  08D0                    .long   .L180
1001                                           .text
1002    00 08EA                              .L149:
1003    00 08EA                              .L139:
1004    00 08EA  6000  FE08                    jra     .L141
1005    00 08EE                              .L140:
1006    00 08EE                              .L113:
1007    00 08EE  4CEE  20C0  FFEC              moveml  a6@(-_F16),#83
1008    00 08F4  4E5E                          unlk    a6
```

```
1009    00 08F6  4E75                                  rts
1010                                                _F16 = 20
1011                                                _S16 = 8384
1012                                                | M16 = 16
1013                                                   .data
1014    00 09C4                                     .L27:
1015    00 09C4  08 20 08 00                           .byte   8,32,8,0
1016    00 09C8                                     .L44:
1017    00 09C8  3F 20 00                              .byte   63,32,0
1018    00 09CB                                     .L59:
1019    00 09CB  3A 20 00                              .byte   58,32,0
1020    00 09CE                                     .L61:
1021    00 09CE  3A 20 00                              .byte   58,32,0
1022    00 09D1                                     .L87:
1023    00 09D1  3A 20 00                              .byte   58,32,0
1024    00 09D4                                     .L107:
1025    00 09D4  43 6F 6D 6D 61 6E 64 73               .byte   67,111,109,10
1026    00 09DC  3A 0A 41 6E 2C 44 6E 2C               .byte   58,10,65,110,
1027    00 09E4  4D 6E 20 2D 20 6F 70 65               .byte   77,110,32,45,
1028    00 09EC  6E 20 41 2C 20 44 2C 20               .byte   110,32,65,44,
1029    00 09F4  6F 72 20 4D 61 70 20 72               .byte   111,114,32,77
1030    00 09FC  65 67 20 23 6E 0A 00                  .byte   101,103,32,35
1031    00 0A03                                     .L108:
1032    00 0A03  45 20 61 20 2D 20 65 78               .byte   69,32,97,32,4
1033    00 0A0B  61 6D 69 6E 65 20 77 6F               .byte   97,109,105,11
1034    00 0A13  72 64 20 40 61 0A 4F 20               .byte   114,100,32,64,
1035    00 0A1B  61 20 2D 20 65 78 61 6D               .byte   97,32,45,32,1
1036    00 0A23  69 6E 65 20 62 79 74 65               .byte   105,110,101,3
1037    00 0A2B  20 40 61 0A 00                        .byte   32,64,97,10,0
1038    00 0A30                                     .L109:
1039    00 0A30  47 20 5B 61 5D 20 2D 20               .byte   71,32,91,97,93
1040    00 0A38  67 6F 0A 43 20 2D 20 63               .byte   103,111,10,67,
1041    00 0A40  6F 6E 74 69 6E 75 65 0A               .byte   111,110,116,10
1042    00 0A48  42 20 2D 20 62 72 65 61               .byte   66,32,45,32,98
1043    00 0A50  6B 70 6F 69 6E 74 0A 00               .byte   107,112,111,10
1044    00 0A58                                     .L110:
1045    00 0A58  52 20 2D 20 6F 70 65 6E               .byte   82,32,45,32,11
1046    00 0A60  20 70 72 6F 63 20 72 65               .byte   32,112,114,111
1047    00 0A68  67 73 0A 49 20 54 20 2D               .byte   103,115,10,73,
1048    00 0A70  20 74 72 61 6E 73 70 61               .byte   32,116,114,97,
1049    00 0A78  72 65 6E 74 20 6D 6F 64               .byte   114,101,110,11
1050    00 0A80  65 0A 00                              .byte   101,10,0
1051    00 0A83                                     .L111:
1052    00 0A83  4C 20 3C 63 6D 64 3E 20               .byte   76,32,60,99,10
1053    00 0A8B  2D 20 64 6F 77 6E 6C 6F               .byte   45,32,100,111,
1054    00 0A93  61 64 20 77 69 74 68 20               .byte   97,100,32,119,
1055    00 0A9B  3C 63 6D 64 3E 0A 58 20               .byte   60,99,109,100,
1056    00 0AA3  2D 20 73 65 74 20 65 73               .byte   45,32,115,101,
1057    00 0AAB  63 61 70 65 0A 00                     .byte   99,97,112,101,
1058    00 0AB1                                     .L118:
1059    00 0AB1  53 75 6E 20 4D 6F 6E 69               .byte   83,117,110,32,
1060    00 0AB9  74 6F 72 2C 20 56 65 72               .byte   116,111,114,44
1061    00 0AC1  73 69 6F 6E 20 30 2E 38               .byte   115,105,111,11
1062    00 0AC9  20 2D 20 30 78 00                     .byte   32,45,32,48,12
1063    00 0ACF                                     .L119:
1064    00 0ACF  20 62 79 74 65 73 20 6F               .byte   32,98,121,116,
1065    00 0AD7  66 20 6D 65 6D 6F 72 79               .byte   102,32,109,101
1066    00 0ADF  0A 00                                 .byte   10,0
1067    00 0AE1                                     .L121:
1068    00 0AE1  0A 41 62 6F 72 74 00                  .byte   10,65,98,111,1
1069    00 0AE8                                     .L124:
1070    00 0AE8  42 52 45 41 4B 0A 00                  .byte   66,82,69,65,75
1071    00 0AEF                                     .L127:
```

```
 1                                                          .data
 2                                                  _disp_us=64.
 3                                                          .text
 4                                                          .globl  busyouta
 5      00 0000                                     busyouta:
 6      00 0000  4E56  FFFC                                 link    a6,#-_F1
 7      00 0004  48EE  0080  FFFC                           moveml  #_S1,a6@(-_F1)
 8      00 000A  1E2E  000B                                 movb    a6@(11),d7
 9                                                  | A1 = 12
10      00 000E                                     .L13:
11      00 000E  1039  0060  0002                           movb    6291458,d0
12      00 0014  4880                                       extw    d0
13      00 0016  48C0                                       extl    d0
14      00 0018  0280  0000  0004                           andl    #4,d0
15      00 001E  4A00                                       tstb    d0
16      00 0020  67EC                                       jeq     .L13
17      00 0022                                     .L14:
18      00 0022  13C7  0060  0000                           movb    d7,6291456
19      00 0028                                     .L12:
20      00 0028  4CEE  0080  FFFC                           moveml  a6@(-_F1),#128
21      00 002E  4E5E                                       unlk    a6
22      00 0030  4E75                                       rts
23                                                  _F1 = 4
24                                                  _S1 = 128
25                                                  | M1 = 0
26                                                          .text
27                                                          .globl  busyoutb
28      00 0032                                     busyoutb:
29      00 0032  4E56  FFFC                                 link    a6,#-_F2
30      00 0036  48EE  0080  FFFC                           moveml  #_S2,a6@(-_F2)
31      00 003C  1E2E  000B                                 movb    a6@(11),d7
32                                                  | A2 = 12
33      00 0040                                     .L17:
34      00 0040  1039  0060  0006                           movb    6291462,d0
35      00 0046  4880                                       extw    d0
36      00 0048  48C0                                       extl    d0
37      00 004A  0280  0000  0004                           andl    #4,d0
38      00 0050  4A00                                       tstb    d0
39      00 0052  67EC                                       jeq     .L17
40      00 0054                                     .L18:
41      00 0054  13C7  0060  0004                           movb    d7,6291460
42      00 005A                                     .L16:
43      00 005A  4CEE  0080  FFFC                           moveml  a6@(-_F2),#128
44      00 0060  4E5E                                       unlk    a6
45      00 0062  4E75                                       rts
46                                                  _F2 = 4
47                                                  _S2 = 128
48                                                  | M2 = 0
49                                                          .text
50                                                          .globl  putchar
51      00 0064                                     putchar:
52      00 0064  4E56  FFF8                                 link    a6,#-_F3
53      00 0068  48EE  2080  FFF8                           moveml  #_S3,a6@(-_F3)
54      00 006E  1E2E  000B                                 movb    a6@(11),d7
55                                                  | A3 = 12
56      00 0072  4AB9  0000  0282                           tstl    642
57      00 0078  6608                                       jne     .L21
58      00 007A  2A7C  0000  0000                           movl    #busyouta,a5
59      00 0080  6006                                       jra     .L22
60      00 0082                                     .L21:
61      00 0082  2A7C  0000  0032                           movl    #busyoutb,a5
62      00 0088                                     .L22:
63      00 0088  1007                                       movb    d7,d0
```

```
 64    00 008A  4880                          extw    d0
 65    00 008C  48C0                          extl    d0
 66    00 008E  2F00                          movl    d0,sp@-
 67    00 0090  4E95                          jbsr    a5@
 68    00 0092  588F                          addql   #4,sp
 69    00 0094  0207  007F                    andb    #127,d7
 70    00 0098  0C07  000D                    cmpb    #13,d7
 71    00 009C  660A                          jne     .L23
 72    00 009E  2F3C  0000  000A              movl    #10,sp@-
 73    00 00A4  4E95                          jbsr    a5@
 74    00 00A6  588F                          addql   #4,sp
 75    00 00A8                          .L23:
 76    00 00A8  0C07  000A                    cmpb    #10,d7
 77    00 00AC  660A                          jne     .L20
 78    00 00AE  2F3C  0000  000D              movl    #13,sp@-
 79    00 00B4  4E95                          jbsr    a5@
 80    00 00B6  588F                          addql   #4,sp
 81    00 00B8                          .L24:
 82    00 00B8                          .L20:
 83    00 00B8  4CEE  2080  FFF8              moveml  a6@(-_F3),#83
 84    00 00BE  4E5E                          unlk    a6
 85    00 00C0  4E75                          rts
 86                                     _F3 = 8
 87                                     _S3 = 8320
 88                                     | M3 = 4
 89                                         .text
 90                                         .globl  getchar
 91    00 00C2                          getchar:
 92    00 00C2  4E56  FFF8                    link    a6,#-_F4
 93    00 00C6  48EE  00C0  FFF8              moveml  #_S4,a6@(-_F4
 94                                     | A4 = 8
 95    00 00CC  2039  0000  0282              movl    642,d0
 96    00 00D2  0680  0060  0002              addl    #6291458,d0
 97    00 00D8  2C00                          movl    d0,d6
 98    00 00DA                          .L27:
 99    00 00DA  2046                          movl    d6,a0
100    00 00DC  1010                          movb    a0@,d0
101    00 00DE  4880                          extw    d0
102    00 00E0  48C0                          extl    d0
103    00 00E2  0280  0000  0001              andl    #1,d0
104    00 00E8  67F0                          jeq     .L27
105    00 00EA                          .L28:
106    00 00EA  5586                          subql   #2,d6
107    00 00EC  2046                          movl    d6,a0
108    00 00EE  1010                          movb    a0@,d0
109    00 00F0  4880                          extw    d0
110    00 00F2  48C0                          extl    d0
111    00 00F4  0280  0000  007F              andl    #127,d0
112    00 00FA  1E00                          movb    d0,d7
113    00 00FC  4AB9  0000  0286              tstl    646
114    00 0102  670E                          jeq     .L29
115    00 0104  1007                          movb    d7,d0
116    00 0106  4880                          extw    d0
117    00 0108  48C0                          extl    d0
118    00 010A  2F00                          movl    d0,sp@-
119    00 010C  6100  FF56                    jbsr    putchar
120    00 0110  588F                          addql   #4,sp
121    00 0112                          .L29:
122    00 0112  1007                          movb    d7,d0
123    00 0114                          .L26:
124    00 0114  4CEE  00C0  FFF8              moveml  a6@(-_F4),#192
125    00 011A  4E5E                          unlk    a6
126    00 011C  4E75                          rts
```

```
127                                                     _F4 = 8
128                                                     _S4 = 192
129                                                     | M4 = 4
130                                                         .text
131                                                         .globl  transparent
132     00 011E                                         transparent:
133     00 011E  4E56  FFF8                                 link    a6,#-_F5
134     00 0122  48EE  00C0  FFF8                           moveml  #_S5,a6@(-_F5
135                                                     | A5 = 8
136     00 0128  4286                                       clrl    d6
137     00 012A                                         .L34:
138     00 012A  1039  0060  0006                           movb    6291462,d0
139     00 0130  4880                                       extw    d0
140     00 0132  48C0                                       extl    d0
141     00 0134  0280  0000  0001                           andl    #1,d0
142     00 013A  6712                                       jeq     .L35
143     00 013C  1039  0060  0004                           movb    6291460,d0
144     00 0142  4880                                       extw    d0
145     00 0144  48C0                                       extl    d0
146     00 0146  2F00                                       movl    d0,sp@-
147     00 0148  6100  FEB6                                 jbsr    busyouta
148     00 014C  588F                                       addql   #4,sp
149     00 014E                                         .L35:
150     00 014E  1039  0060  0002                           movb    6291458,d0
151     00 0154  4880                                       extw    d0
152     00 0156  48C0                                       extl    d0
153     00 0158  0280  0000  0001                           andl    #1,d0
154     00 015E  67CA                                       jeq     .L34
155     00 0160  1E39  0060  0000                           movb    6291456,d7
156     00 0166  4A86                                       tstl    d6
157     00 0168  6726                                       jeq     .L37
158     00 016A  1007                                       movb    d7,d0
159     00 016C  4880                                       extw    d0
160     00 016E  48C0                                       extl    d0
161     00 0170  0280  0000  005F                           andl    #95,d0
162     00 0176  0C80  0000  0043                           cmpl    #67,d0
163     00 017C  6732                                       jeq     .L33
164     00 017E                                         .L38:
165     00 017E  4286                                       clrl    d6
166     00 0180                                         .L40:
167     00 0180                                         .L39:
168     00 0180  1007                                       movb    d7,d0
169     00 0182  4880                                       extw    d0
170     00 0184  48C0                                       extl    d0
171     00 0186  2F00                                       movl    d0,sp@-
172     00 0188  6100  FEA8                                 jbsr    busyoutb
173     00 018C  588F                                       addql   #4,sp
174     00 018E                                         .L36:
175     00 018E                                         .L32:
176     00 018E  609A                                       jra     .L34
177     00 0190                                         .L37:
178     00 0190  1007                                       movb    d7,d0
179     00 0192  4880                                       extw    d0
180     00 0194  48C0                                       extl    d0
181     00 0196  0280  0000  007F                           andl    #127,d0
182     00 019C  1239  0000  0280                           movb    640,d1
183     00 01A2  4881                                       extw    d1
184     00 01A4  48C1                                       extl    d1
185     00 01A6  B081                                       cmpl    d1,d0
186     00 01A8  66D6                                       jne     .L39
187     00 01AA  5286                                       addql   #1,d6
188     00 01AC  6000  FF7C                                 jra     .L34
189     00 01B0                                         .L33:
```

```
190     00 01B0                                         .L31:
191     00 01B0  4CEE  00C0  FFF8                         moveml  a6@(-_F5),#192
192     00 01B6  4E5E                                     unlk    a6
193     00 01B8  4E75                                     rts
194                                                     _F5 = 8
195                                                     _S5 = 192
196                                                     | M5 = 4
197                                                       .data
198     00 01BC                                         .end
```

```
 1                                                          .text
 2                                                          .globl  Emulate
 3     00 0000                                          Emulate:
 4     00 0000  4E56  FFE0                                 link    a6,#-_F1
 5     00 0004  48EE  38F8  FFE0                           moveml  #_S1,a6@(-_F1)
 6     00 000A  2A6E  0008                                 movl    a6@(8),a5
 7     00 000E  3E2E  000E                                 movw    a6@(14),d7
 8     00 0012  286E  0010                                 movl    a6@(16),a4
 9                                                        | A1 = 20
10     00 0016  267C  0000  0200                           movl    #512,a3
11     00 001C  3007                                       movw    d7,d0
12     00 001E  48C0                                       extl    d0
13     00 0020  0280  0000  2000                           andl    #8192,d0
14     00 0026  6706                                       jeq     .L13
15     00 0028  41ED  0004                                 lea     a5@(4),a0
16     00 002C  2848                                       movl    a0,a4
17     00 002E                                          .L13:
18     00 002E  2C14                                       movl    a4@,d6
19     00 0030  2A2C  0004                                 movl    a4@(4),d5
20     00 0034  282C  0008                                 movl    a4@(8),d4
21     00 0038  262C  000C                                 movl    a4@(12),d3
22     00 003C  4A86                                       tstl    d6
23     00 003E  6C12                                       jge     .L14
24     00 0040  3007                                       movw    d7,d0
25     00 0042  48C0                                       extl    d0
26     00 0044  0280  0000  2000                           andl    #8192,d0
27     00 004A  6606                                       jne     .L14
28     00 004C                                          .L20000:
29     00 004C  70FF                                       moveq   #-1,d0
30     00 004E  6000  00CA                                 jra     .L12
31     00 0052                                          .L14:
32     00 0052  2006                                       movl    d6,d0
33     00 0054                                          .L16:
34     00 0054  0480  FFFF  FFFC                           subl    #-4,d0
35     00 005A  0C80  0000  0009                           cmpl    #9,d0
36     00 0060  62EA                                       jhi     .L20000
37     00 0062  D040                                       addw    d0,d0
38     00 0064  D040                                       addw    d0,d0
39     00 0066  207C  0000  0124                           movl    #.L30,a0
40     00 006C  2070  0000                                 movl    a0@(0,d0:w),a0
41     00 0070  4ED0                                       jmp     a0@
42     00 0072                                          .L17:
43     00 0072  2F05                                       movl    d5,sp@-
44     00 0074  4EB9  0000  0000                        7  jbsr   putchar
45     00 007A  588F                                       addql   #4,sp
46     00 007C  6000  009C                                 jra     .L12
47     00 0080                                          .L19:
48     00 0080  7008                                       moveq   #8,d0
49     00 0082  6000  0096                                 jra     .L12
50     00 0086                                          .L20:
51     00 0086  4EB9  0000  0000                        7  jbsr   getchar
52     00 008C  6000  008C                                 jra     .L12
53     00 0090                                          .L22:
54     00 0090  2745  0086                                 movl    d5,a3@(134)
55     00 0094                                          .L20001:
56     00 0094  4280                                       clrl    d0
57     00 0096  6000  0082                                 jra     .L12
58     00 009A                                          .L23:
59     00 009A  2C2B  007C                                 movl    a3@(124),d6
60     00 009E  2005                                       movl    d5,d0
61     00 00A0  720C                                       moveq   #12,d1
62     00 00A2  E3A0                                       asll    d1,d0
63     00 00A4  33C0  00E0  0000                           movw    d0,14680064
```

```
 64     00 00AA  2004                          movl    d4,d0
 65     00 00AC  720F                          moveq   #15,d1
 66     00 00AE  E3A0                          asll    d1,d0
 67     00 00B0  0680  00C0  0000              addl    #12582912,d0
 68     00 00B6  2040                          movl    d0,a0
 69     00 00B8  3083                          movw    d3,a0@
 70     00 00BA  2006                          movl    d6,d0
 71     00 00BC                        .L20004:
 72     00 00BC  720C                          moveq   #12,d1
 73     00 00BE                        .L20003:
 74     00 00BE  E3A0                          asll    d1,d0
 75     00 00C0                        .L20002:
 76     00 00C0  33C0  00E0  0000              movw    d0,14680064
 77     00 00C6  60CC                          jra     .L20001
 78     00 00C8                        .L24:
 79     00 00C8  2C2B  007C                    movl    a3@(124),d6
 80     00 00CC  2005                          movl    d5,d0
 81     00 00CE  720C                          moveq   #12,d1
 82     00 00D0  E3A0                          asll    d1,d0
 83     00 00D2  33C0  00E0  0000              movw    d0,14680064
 84     00 00D8  2004                          movl    d4,d0
 85     00 00DA  720F                          moveq   #15,d1
 86     00 00DC  E3A0                          asll    d1,d0
 87     00 00DE  0680  00C0  0000              addl    #12582912,d0
 88     00 00E4  2040                          movl    d0,a0
 89     00 00E6  3010                          movw    a0@,d0
 90     00 00E8  48C0                          extl    d0
 91     00 00EA  2600                          movl    d0,d3
 92     00 00EC  2006                          movl    d6,d0
 93     00 00EE  720C                          moveq   #12,d1
 94     00 00F0  E3A0                          asll    d1,d0
 95     00 00F2  33C0  00E0  0000              movw    d0,14680064
 96     00 00F8  2003                          movl    d3,d0
 97     00 00FA  601E                          jra     .L12
 98     00 00FC                        .L25:
 99     00 00FC  202B  007C                    movl    a3@(124),d0
100     00 0100  6018                          jra     .L12
101     00 0102                        .L26:
102     00 0102  2745  007C                    movl    d5,a3@(124)
103     00 0106  2005                          movl    d5,d0
104     00 0108  60B2                          jra     .L20004
105     00 010A                        .L27:
106     00 010A  202B  0078                    movl    a3@(120),d0
107     00 010E  600A                          jra     .L12
108     00 0110                        .L28:
109     00 0110  202B  0074                    movl    a3@(116),d0
110     00 0114  6004                          jra     .L12
111     00 0116                        .L29:
112     00 0116  6000  FF34                    jra     .L20000
113                                            .data
114     00 0124                        .L30:
115     00 0124  0000  0102                    .long   .L26
116     00 0128  0000  00FC                    .long   .L25
117     00 012C  0000  00C8                    .long   .L24
118     00 0130  0000  009A                    .long   .L23
119     00 0134  0000  0110                    .long   .L28
120     00 0138  0000  0072                    .long   .L17
121     00 013C  0000  0080                    .long   .L19
122     00 0140  0000  0086                    .long   .L20
123     00 0144  0000  010A                    .long   .L27
124     00 0148  0000  0090                    .long   .L22
125                                            .text
126     00 011A                        .L15:
```

```
127     00 011A                                         .L12:
128     00 011A  4CEE  38F8  FFE0                           moveml  a60(-_F1),#14
129     00 0120  4E5E                                       unlk    a6
130     00 0122  4E75                                       rts
131                                                     _F1 = 32
132                                                     _S1 = 14584
133                                                     | M1 = 4
134                                                         .data
135     00 014C                                         .end


        2 Error(s) and 0 Warning(s)
         #      Description of Error(s)

        7       Undefined symbol
```