```
SERIES-III PL/M-86 V1.0 COMPILATION OF MODULE HARDWARE
NO OBJECT MODULE REQUESTED
COMPILER INVOKED BY:  P.86 TEMP.SRC OPTIMIZE(3) PAGELENGTH(42) PAGEWIDTH(109) PRINT(:F4:HW.LS) NOOBJECT


          $TITLE ('SIRIUS Victor Business Products (c) 1982  V9000 Hardware')
          $SUBTITLE ('Example software drivers for V9000 Hardware')
           /************************************************************************
            *                                                                      *
            *  Proprietary Rights Notice:   All rights  reserved.  This material  *
            *  contains  the   valuable   properties   and   trade  secrets   of  *
            *  Victor Business Productse, Inc. (VICTOR)                         *
            *  of  Chicago, Illinois, USA.                                        *
            *  embodying   substantial   creative   efforts   and   confidential  *
            *  information, ideas,  and  expressions;  no part of  which  may be  *
            *  reproduced or transmited in any form or by any means; electronic,  *
            *  mechanical, or otherwise;  including  photo copying and recording  *
            *  or  in  connection  with  any  information  storage or  retrieval  *
            *  system  without the permission  in writing  from SIRIUS.           *
            *                                                                      *
            *  Copyright notice:  Copyright (c) 1982                              *
            *                                                                      *
            *  An unpublished work by:                                            *
            *                                                                      *
            *                  Victor Businss Products, Inc.                      *
            *                  3900 N. Rockwell Street                            *
            *                  Chicago, IL  60618                                 *
            *                                                                      *
            ************************************************************************/
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
                $eject
                $SMALL ROM

  1          •  Hardware: do;

  2    1          Declare dcl       literally 'declare';
  3    1          Dcl     lit       literally 'literally';
  4    1          Dcl     addr      lit       'address',
                          ext       lit       'external',
                          init      lit       'initial',
                          intg      lit       'integer',
                          proc      lit       'procedure',
                          ptr       lit       'pointer',
                          pub       lit       'public',
                          rent      lit       'reentrant',
                          ret       lit       'return',
                          struc     lit       'structure',
                          boolean   lit       'byte',
                          true      lit       '0FFH',
                          false     lit       '0000H';
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
            $subtitle('KB: Hardware bit defs')


   5   1    dcl SR$intbit           lit '4';   /* KB shift register interrupt mask in6522 IER/IFR */
   6   1    dcl SR$enable           lit '0ch';        /* KB shift register enable in 6522 ACR */
   7   1    dcl CB1$intbit          lit '10h';    /* KB RDY edge-sense interrupt mask 6522 PCR */
   8   1    dcl CB1$pos_edge        lit '10h';       /* KB RDY edge-sense control in 6522 PCR */

   9   1    dcl kb$databit          lit '40h';        /* KB DATA level                         */
  10   1    dcl kb$ackctl           lit '2';          /* KB ACK control for 6522 output        */
  11   1    dcl kb$TIMEOUT          lit '300';        /* error timeout in milliseconds         */
  12   1    dcl timer1_ena          lit '0c0h';      /* timer 1 interrupt mask in 6522 IER/IFR */
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
              $eject

              /*  KYBRD PORT (e8040..e804f)  */

 13   1         dcl via(16)  struc(                                /* 6522 port organization              */
                                  RB      byte,
                                  RA      byte,
                                  DDRB    byte,
                                  DDRA    byte,
                                  TIMER1  word,
                                  TIMER1L word,
                                  TIMER2  word,
                                  SR      byte,
                                  ACR     byte,
                                  PCR     byte,
                                  IFR     byte,
                                  IER     byte,
                                  RAX     byte) at(0e8000h);

 14   1       dcl kb$state               byte;                     /* current state of keyboard stateware  */
 15   1       dcl kb$data                byte;                     /* constructed data from keyboard       */


                                                              /* nybble convert table for inverted shift reg */
 16   1       dcl Ctable(*) byte data (0,8,4,0ch, 2,0ah,6,0eh, 1,9,5,0dh, 3,0bh,7,0fh);

 17   1       dcl tick                lit '50';                    /* console clock rate in milliseconds   */
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$subtitle('KB: external routines')

  /*
   *    signal user about keyboard error state -- ring bell
   */
  dcl signal$KB$error lit 'Ringbell';
     /* Ringbell found in SOUND module */

  /*
   *    Process key board event -- in external module
   */
Process$Event: proc(event) byte ext;
  dcl event byte;
end;


  /*
   *   Software clock resource -- set timeout for interrupt to KB$reset
   */

set$KB$clock: proc(Period) ext;
  dcl Period intg;                                /* timeout delay in milliseconds       */
end set$KB$clock;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$subtitle('KB: Keyboard Stateware')
  /*
   *    KB interrupt entry  (level 6)
   */
kb$irq: proc pub rent;
    do case kb$state;
  /*
   *    state 0 to state 1:  shift register (full) interrupt
   */
  kbst0: do;
        via(4).ACR= via(4).ACR and not SR$enable; /* disable shift register            */
                                /* prepare for interrupt on negative edge of KB RDY */
        via(4).PCR= via(4).PCR and not CB1$pos_edge;
        via(4).IER= 80h or CB1$intbit;
       disable;                                   /* time critical section             */
        kb$data = via(4).SR;                      /* get KB data from SR (clears SR IRQ)  */
        via(4).IER= SR$intbit;                    /* disable SR interrupt              */
                                                  /* assert KB ACK control on interrupt   */
        via(4).RB = via(4).RB or kb$ackctl;       /* (CB1 IRQ is reset)                */
       enable;                                    /* end of critical section           */
        kb$state = 1;                             /* set to state 1                    */
      end;
  /*
   *    state 1 to state 2:  interrupt from negative edge on KB$RDY
   */
  kbst1: do;
       disable;                                   /* time critical section             */
        if (via(4).RA and kb$databit) <> 0 then   /* if data bit is not low then       */
            call kb$error;                        /* stop bit error has occurred       */
         else do;               /* prepare for interrupt on positive edge of KB RDY */
            via(4).PCR= via(4).PCR or CB1$pos_edge;
                                                  /* release KB ACK control on interrupt  */
            via(4).RB = via(4).RB and not kb$ackctl; /* (CB1 IRQ is reset)             */
            kb$state = 2;                         /* set to state 2                    */
          end;
       enable;                                    /* end of critical section
      end;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
        $eject

          /*
           * state 2 to state 0:  interrupt from positive edge on KB$RDY
           */
          kbst2: do;
                if (via(4).RA and kb$databit) = 0 then    /* if data bit is low then                */
                    call kb$error;                          /* stop bit error has occurred            */
                 else do;
                    call kb$reset;                         /*reset hardware/software for next event */
                                  /* call event processing routine with order of bits reversed to */
                                  /*   reflect physical key number and event type (open or close) */
                    if not Process$Event( shl(Ctable(kb$data and 0fh),4)
                        or Ctable(shr(kb$data,4)) ) then
                            call signal$KB$error;          /* signal error in event process        */
                  end;
              end;

          end;

        end kb$irq;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
         $subtitle('KB: Keyboard support routines')

   1     kb$reset:  proc rent;                           /* puts KB hardware/software into state 0 */
   2       dcl dummy byte;

   2         via(4).IER = CB1$intbit;                     /* clear CB1 interrupts                */
   2         via(4).RB  = via(4).RB and not kb$ackctl;    /* release kb$ack                      */
   2         via(4).ACR = via(4).ACR or SR$enable;        /* enable shft reg                     */
   2         dummy      = via(4).SR;                      /* clr any pending irq                 */
   2         via(4).IER = 80h or SR$intbit;               /* enable sr interrupts                */
   2         kb$state   = 0;                              /* init keybrd state                   */
   2         call set$KB$clock(0);                        /* clear timeout counter               */
   2     end kb$reset;

   1     kb$error:  proc rent;
   2         via(4).RB  = via(4).RB or kb$ackctl;         /* force kb$ack high                   */
   2         via(4).IER = 7fh;                            /* allow no interrupts                 */
   2         call set$KB$clock(kb$TIMEOUT);               /* time out keyboard                   */
   2     end kb$error;

   1     kb$init:  proc  pub rent;

   2         via(4).RB   = via(4).RB and (0FFh-3);
   2         via(4).DDRA = via(4).DDRA and not kb$databit;
   2         via(4).DDRB = via(4).DDRB or kb$ackctl;
   2         via(4).IER  = 7fh;
   2         via(4).PCR  = 0;
   2         via(4).ACR  = 0;

   2         via(2).ACR= (via(2).ACR and 0c0h) or 40h;
   2         via(2).timer1L= tick*1000;
   2         via(2).IER  = timer1_ena and 7fh;
   2         call kb$reset;
   2       end kb$init;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
               $SUBTITLE ('CRTreg: controller chip registers')

  87   1       DCL   CRT$0      byte   AT (0E8000H);              /* CRT-chip address register          */
  88   1       DCL   CRT$1      BYTE   AT (0E8001H);              /* CRT-chip internal register port    */


               /*
                *     Set CRT register
                */
  89   1       set$CRT$reg: proc (reg,value) rent;
  90   2         dcl reg byte;
  91   2         dcl value byte;
  92   2           CRT$0= reg;                                    /* select register                    */
  93   2           CRT$1= value;                                  /* set data                           */
  94   2         end set$CRT$reg;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$SUBTITLE ('CRTreg: cursor-display mode control')

dcl   rast$start  lit    '10';             /* CRT reg: cursor-start & cursor-display mode */


 DCL   Cursor$PAR   BYTE; /* VAR: contents for CRT cursor-start raster & cursor display mode */
 dcl   blink$on      boolean;                        /* FLAG:  =0  Blinking cursor on (fast) */
 dcl   curs$off      boolean;                        /* FLAG: <>0  Cursor off                */


/*
 *      Set cursor to current Cursor parameter byte.
 */
set$cursor: proc rent;
    call set$CRT$reg(rast$start,Cursor$PAR);      /* set raster start reg                   */
  end set$cursor;

/*
 *      Set block cursor.
 */
BLOCK$CRS:PROC  RENT;
    Cursor$PAR = Cursor$PAR AND 0E0h;             /* set block cursor                       */
    call set$cursor;                              /* set cursor mode reg                    */
  END BLOCK$CRS;

/*
 *      Set underscore cursor.
 */
UNDERSCORE$CRS:PROC RENT;
    Cursor$PAR = 00Fh OR (Cursor$PAR AND 0E0h);   /* set underscore cursor                  */
    call set$cursor;                              /* set cursor mode reg                    */
  END UNDERSCORE$CRS;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
              $eject
              /*
               *    Return cursor to previous modes: block or underline, steady or flashing
               */
 110   1      CURSOR$ON:PROC  RENT;
 111   2          curs$off= false;                              /* reset cursor off flag                 */
 112   2          if blink$on then Cursor$par= Cursor$par or 060h; /* set to flashing mode              */
 114   2           else Cursor$par= Cursor$par and 01Fh;        /* set to steady mode                    */
 115   2          call set$cursor;                              /* set cursor mode reg                   */
 116   2        END CURSOR$ON;
              /*
               *      Turn cursor off.
               */
 117   1      CURSOR$OFF:PROC  RENT;
 118   2          curs$off= true;                               /* set cursor off flag                   */
 119   2          Cursor$PAR = 020h OR (Cursor$PAR AND 01Fh);   /* set to off mode                       */
 120   2          call set$cursor;                              /* set cursor mode reg                   */
 121   2        END CURSOR$OFF;

              /*
               *      Set cursor blinking.
               */
 122   1      CRS$BLINK$ON:PROC  RENT;
 123   2          blink$on= true;                               /* set blinking on flag                  */
 124   2          if not curs$off then Cursor$PAR= 060h OR Cursor$PAR; /* set flashing,if not off       */
 126   2          call set$cursor;                              /* set cursor mode reg                   */
 127   2       END CRS$BLINK$ON;
              /*
               *      Set cursor steady.
               */
 128   1      CRS$BLINK$OFF:PROC  RENT;
 129   2          blink$on= false;                              /* reset blinking on flag                */
 130   2          if not curs$off then Cursor$PAR= 01Fh and Cursor$PAR; /* set steady,if not off       */
 132   2          call set$cursor;                              /* set cursor mode reg                   */
 133   2        END CRS$BLINK$OFF;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
        $SUBTITLE ('CRTreg: Cursor positioning')


        dcl    cursaddrH   lit    '14'; /* CRT reg: MSByte of cursor location word, bits: xx54$3210 */
        dcl    cursaddrL   lit    '15';                  /* CRT reg: LSByte of cursor location word */


          /*
           *   Position Cursor to Absolute Font Cell number
           *     and display bank
           */

        POS$Cursor: proc (Cell$number) pub rent;
          dcl Cell$Number word;                          /* Absolute Font Cell Number & diplay bank */
            call set$CRT$reg (cursaddrL, low(Cell$number));
            call set$CRT$reg (cursaddrH, high(Cell$number));
          end POS$Cursor;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
             $SUBTITLE ('CRT: video contrast & brightness')

      1      DCL   CBctrl    BYTE   AT (0E8040H);             /* Contrast & Brightness control register */
                                                               /* bits: CCCB$BB--                        */

             /*
              *      Raise video contrast one level.
              */
      1      contrast$up: proc rent;
      2        dcl a byte;
      2        if (a:= (CBctrl + 20h) and 0E0h) <> 0 then      /* add & check upper limit              */
      2          CBctrl= (CBctrl and 01FH) or a;               /* set contrast, bits: 765              */
      2      end contrast$up;
             /*
              *      Lower video contrast one level.
              */
      1      contrast$down: proc rent;
      2        dcl a byte;
      2        if (a:= (CBctrl - 20h) and 0E0h) <> 0E0h then   /* sub & check lower limit              */
      2          CBctrl= (CBctrl and 01FH) or a;               /* set contrast, bits: 765              */
      2      end contrast$down;

             /*
              *      Raise video brightness one level.
              */
      1      bright$up: proc rent;
      2        dcl a byte;
      2        if (a:= (CBctrl + 4) and 01CH) <> 0 then        /* add & check upper limit              */
      2          CBctrl= (CBctrl and 0E3H) or a;               /* set brightness, bits: 432            */
      2      end bright$up;
             /*
              *      Lower video brightness one level.
              */
      1      bright$down: proc rent;
      2        dcl a byte;
      2        if (a:= (CBctrl - 4) and 01Ch) <> 01Ch then     /* sub & check lower limit              */
      2          CBctrl= (CBctrl and 0E3H) or a;               /* set brightness, bits: 432            */
      2      end bright$down;
```

```
                $SUBTITLE ('CRT: display RAM/Font Cells')

      1        dcl   screen$ram word at (0F0000h);                  /* memory address of display RAM          */
      1        dcl   screen$addr  ptr;                           /* display ram pointer, base of word ARRAY */
      1        DCL   SCREEN based screen$addr (2000) word;          /* ARRAY of Font Cell Pointers            */

                  /*
                   *    Screen Buffer Word variables
                   */

      1        dcl   char$mode      word       pub;                 /* CRT attribute bits: 7654$3---          */

      1        dcl   char$base      word       pub;                 /* CRT Font Cell Pointer base for         */
                                                                    /*     ASCII symbol index                 */

      1         DCL   REVBIT       LIT    '8000H';
      1         DCL   BGBIT        LIT    '4000H';
      1         DCL   UNDBIT       LIT    '2000H';
      1         dcl   INVBIT       lit    '1000h';
      1         dcl   extraBIT     lit    '0800h';


                   /*
                    *  Display symbol from character set (typically ASCII)
                    *     at absolute Font Cell number
                    *      (typically: <line> * <display width> + <column> )
                    *     with current Cursor & Display modes.
                    */

      1        Display$symbol: proc (Symbol$code,Cell$number) pub rent;
      2          dcl Symbol$code byte;                              /* Symbol print code                      */
      2          dcl Cell$Number word;                              /* Absolute Font Cell Number              */
      2            screen(Cell$Number)= (Symbol$code + char$base) OR char$mode;
      2          end Display$symbol;
```

NOTE: ample Programs only - Not BIOS Reference Listi.

```
             $SUBTITLE ('CRT hardware initialization')
                                                    /* COMMENT THIS !!!!                  */
      1      DCL CRT$config (*) BYTE DATA (92,80, 81,0CFh, 25,6, 25,25, 3,14, 0,15, 0,0, 0,0);


      1      CRT$Init:  PROC;
      2        DCL  I  BYTE;

screen$ram;

      2        char$mode= BGBIT;
      2        char$base= 20;

      2        curs$off= false;
      2        blink$on= false;
      2        Cursor$PAR= 0;

      2        DO I=0 TO 0FH;
      3          CALL SET$CRT$REG (I,(CRTconfig(I)));
      3          END;

      2        END CRT$Init;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
              $SUBTITLE ('SOUND variables & hardware defs')


       1      dcl bell$freq LIT    '76';                         /* period of bell tone: frequency= 14.9KHz */

       1      dcl     codec$clk       word at (0E8084h);         /* TIMER1: codec clock frequency        */
       1      dcl     codec$ctl       byte at (0E808Bh);         /* ACR: codec clock control register    */
       1      dcl     codec$sda       word at (0E8060h);         /*                                      */
       1      dcl     volume          byte at (0E802Ah);         /* SR: volume shift-register            */
       1      dcl     vol$ctl         byte at (0E802Bh);         /* ACR: SR control register             */
       1      dcl     vol$clk         word at (0E8028h);         /* TIMER2: volume SR clock              */

       1      dcl bell$on   byte;                                /* FLAG: bell sound presently active    */
       1      dcl vol$level byte;    /*current volume level (nine levels: 0 --> 8)
                                                                 /* volume shift pattern lookup table    */
       1      dcl vol$table (*) byte data (0FFh,7FH,3FH,1FH,0FH,7,3,1,0);
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$SUBTITLE ('SOUND: Bell control')

  /*
   *   Software clock resource -- set timeout for interrupt to Bell$clock
   */

set$BELL$clock: proc (Period) ext;
  dcl Period intg;                                /* timeout delay in milliseconds       */
  end set$BELL$clock;

  /*
   *    CODEC Hardware reset
   */

Bell$init: proc pub rent;
   vol$level= length(vol$table)-2;                /* set initial volume level near max    */
   call Bell$clock;                              /* set hardware to a known & quiet state */
  end Bell$init;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
                $eject

      1         Bell$clock:     proc  pub rent;

      2            codec$ctl = codec$ctl and not 0C0h;              /* disable codec clock                        */

      2            codec$sda = 5E00h;                               /* initialize codec SDA to input mode...      */
      2            codec$sda = 0D40h;                               /* ... to reduce extraneous noise             */
      2            codec$sda = 0AA80h;
      2            codec$sda = 00C0h;

      2            vol$ctl = (vol$ctl and not 3Ch) or 10h;          /* set SR & T2 volume register modes          */
      2            vol$clk = 1;                                  /* volume clock frequency set beyond perception */
      2            volume = vol$table(vol$level);                   /* set volume to current level                */
      2            bell$on = false;                                 /* set bell state to off                      */

      2            end bell$clock;

      1         Ring$bell: proc pub rent;
      2             if not bell$on then do;                         /* start bell if sound is off                 */
      3             call bell$clock;                                   /* init codec hardware on every bell     */
      3             codec$sda = 0f80h;                               /* set output waveform to 4 up & 4 down, */
                                                                    /*    a low amplitude triangle wave.          */
      3             codec$ctl = codec$ctl or 0c0h;                     /* set codec clock to free run
      3             codec$clk = bell$freq;                             /* set audio pitch frequency             */
      3             bell$on = true;                                    /* set bell state on                     */
      3               end;
      2             call set$bell$clock(100);                       /* turn off bell in 100 milliseconds          */
      2          end;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$SUBTITLE ('SOUND: volume control')
/*
 *      Raise CODEC volume one level.
 */
volume$up: proc rent;
    if vol$level >= length(vol$table)-1 then      /* check upper limit             */
        vol$level= length(vol$table)-1;           /* set to max volume             */
     else vol$level= vol$level+1;                 /* bump level up by one          */
        volume= vol$table(vol$level);             /* set volume register           */
 end volume$up;

/*
 *      Lower CODEC volume one level.
 */
volume$down: proc rent;
    if vol$level >= length(vol$table)-1 then      /* check upper limit             */
        vol$level= length(vol$table)-2;           /* set to max volume-1           */
     else
        if vol$level<>0 then vol$level= vol$level-1; /* drop level by one          */
    volume= vol$table(vol$level);                 /* set volume register           */
end volume$down;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
                $subtitle('SIO: Serial I/O dvrs for TTY: and UL1:')

                /*ctr device dcls*/
 242   1        dcl sioctr struc
                    (adata byte,
                     bdata byte,
                     xxx   byte,
                     ctrctl byte) at (0E0020h);
                /*sio device dcls*/
 243   1        dcl siodev struc
                    (adata byte,
                     bdata byte,
                     actl  byte,
                     bctl  byte) at(0E0040h);
 244   1        dcl rx$avail literally '1',
                    tx$empty literally '4';

 245   1        dcl serial_params struc
                    (actrlsb byte,                             /*LSByte of chan a.'s baud rate        */
                     actrmsb byte,                             /*MSByte ...                           */
                     bctrlsb byte,                             /*LSByte of chan b.'s baud rate        */
                     bctrmsb byte,                             /*MSByte ...                           */
                      /* if <baud> then lsb = ??h  msb = ??h 1.25Mhz/(<baud>*16)
                               50 ===>       1Ah       06h      50.00  -0-    (min.tol.dist.43.75%)
                               75 ===>       11h       04h      75.00  -0-    (     "        43.75%)
                              110 ===>       C6h       02h     110.00  -0-    (     "        43.75%)
                              134.5 ===>     44h       02h     134.00  -0.37% (     "        40.23%)
                              150 ===>       08h       02h     150.00  -0-    (     "        43.75%)
                              200 ===>       86h       01h     200.00  -0-    (     "        43.75%)

                              300 ===>       04h       01h     300.00  -0-    (     "        43.75%)

                              600 ===>       82h       00h     600.00  -0-    (     "        43.75%)
                             1.2k ===>       41h       00h    1201.00  +0.08% (     "        42.99%)
```

```
$eject

------------------------------------------------
                          2Ch       00h   1775.00  -1.39% (      "        30.54%)
         1.8k ===>        2Bh       00h   1816.00  +0.09% (      "        42.88%)
------------------------------------------------
                          28h       00h   1953.00  -2.36% (      "        21.33)
         2.0k ===>        27h       00h   2003.00  +0.15% (      "        42.32)
------------------------------------------------
         2.4k ===>        21h       00h   2367.00  -1.38% (      "        30.64%)
                          20h       00h   2441.00  +1.71% (      "        27.51%)
------------------------------------------------
         3.6k ===>        16h       00h   3551.00  -1.36% (       "       30.83%)
                          15h       00h   3720.00  +3.33% (       "       12.4%)
------------------------------------------------
                          11h       00h   4595.00  -4.27% (       "       3.185%)
         4.8k ===>        10h       00h   4882.00  +1.02% (       "       34.06%)
------------------------------------------------
                          09h       00h   8680.55  -9.58% (DISTORTED)
         9.6k ===>        08h       00h   9765.56  +1.73% (min.tol.dist.27.32%)
-----------------------------------------------
                          06h       00h  13020.83  -9.58% (DISTORTED)
                          05h       00h  15625.00  +8.51% (DISTORTED)
-----------------------------------------------
                          05h       00h  15625.00 -18.62% of 19.2k (DISTORTED)
        19.2k ===>        04h       00h  19531.25  +1.02% (min.tol.dist.34.06%)

min.tol.dist. figure assumes no channel noise effects.
  NOTE: possible noise DOES NOT includes bias distorition
        caused by various cable capacitance effects*/
        $eject

            cr2a    byte,       /*bus interface option: 10h if baud a <= baud b
                                                        ~~14h~~ if baud a > baud b*/

            cr4a    byte,
            cr4b    byte,
              /*cr4x (16x)$54$(stops)$(even)$(parenb) = 4?h
                      01    00 ss      e      p
                                ss = 01 1 stop
                                   = 10 1.5 stop
                                   = 11 2 stop
                                        e = 1 even
                                        e = 0 odd, byte transparent
                                              p = 1 even or odd
                                              p = 0 byte transparent*/

            cr3a    byte,
            cr3b    byte,
              /*cr3x (rbits)$(autoenb)$4$3$2$1$(renb) = ?1h
                      bb      1        0 0 0 0  1
                      bb = 11 byte transparent cr3x = E1h
                         = 01 even,odd          cr3x = 61h*/

            cr5a    byte,
            cr5b    byte) EXT;
              /*cr5x (dtr)$(tbits)$(br)$(tenb)$2$(rts)$0 = ?Ah
                      1     bb      0    1      0  1    0
                            bb = 11 space,mark cr5x = EAh
                            bb = 01 even,odd,no cr5x = AAh*/
```

```
$subtitle('SIO: Serial I/O dvrs for port A -- TTY$INSTAT & TTY$STAT')

     1      TTY$in$stat:proc boolean PUB;

     2      if ( (siodev.actl AND rx$avail) <> 0)
            then return(true);
     2      return(false);

     2      end TTY$in$stat;


     1      TTY$stat:proc boolean PUB;

     2      if ( (siodev.actl AND tx$empty) = 0)
            then return(true);
     2      return(false);

     2      end TTY$stat;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
              $subtitle('SIO: Serial I/O dvrs for port A -- TTY$GET & TTY$PUT')


 256   1      TTY$get:proc byte PUB;

              /*user must not activate this procedure if siodev chan. a reg. ptr
                is not set to 0 (only <> 0 if user has been mucking with hardware*/
 257   2      do while( (siodev.actl AND rx$avail) = 0);         /*wait forever till empty              */
 258   3      end;
 259   2      return(siodev.adata);                              /*input form 7201                      */

 260   2      end TTY$get;


 261   1      TTY$put:proc(char) PUB;
 262   2      dcl char byte;

              /*user must not activate this procedure if siodev chan. a reg. ptr
                is not set to 0 (only <> 0 if user has been mucking with hardware*/
 263   2      do while( (siodev.actl AND tx$empty) = 0);         /*wait forever till empty              */
 264   3      end;
 265   2      siodev.adata = char;                               /*output a char                        */
 266   2      return;

 267   2      end TTY$put;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$subtitle('SIO: Serial I/O dvrs for port B -- UL1$STAT & UL1PUT')


UL1$stat:proc boolean PUB;

if ( (siodev.bctl AND tx$empty) = 0)
then return(true);
return(false);

end UL1$stat;



UL1$put:proc(char) PUB;
dcl char byte;

/*user must not activate this procedure if siodev chan. b reg. ptr
  is not set to 0 (only <> 0 if user has been mucking with hardware*/
do while( (siodev.bctl AND tx$empty) = 0);        /*wait forever till empty            */
end;
siodev.bdata = char;                              /*output a char                      */
return;

end UL1$put;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$subtitle('SIO: Serial I/O dvrs for ports A & B -- SIO$INIT')
SIO$init:proc PUB;
siodev.actl = 00$011$000b;                        /*chan. a reset                     */
siodev.bctl = 00$011$000b;                        /*chan. b reset                     */

/*load timer now; cant touch 7201 chip for 4 2.5Mhz clocks*/
sioctr.ctrctl = 36h;                              /*7$(ctra)$(rl)$(mode)$(bin)        */
sioctr.adata = serial_params.actrlsb;
sioctr.adata = serial_params.actrmsb;
sioctr.ctrctl = 76h;                              /*7$(ctrb)$(rl)$(mode)$(bin)        */
sioctr.bdata = serial_params.bctrlsb;
sioctr.bdata = serial_params.bctrmsb;

/*cr2a bus interface option*/
siodev.actl = 2;                                  /*-->cr4a                           */
siodev.actl = serial_params.cr2a;

/*cr4x*/
siodev.actl = 4;                                  /*-->cr4a                           */
siodev.actl = serial_params.cr4a;
siodev.bctl = 4;                                  /*-->cr4b                           */
siodev.bctl = serial_params.cr4b;

/*cr3x*/
siodev.actl = 3;                                  /*-->cr3a                           */
siodev.actl = serial_params.cr3a;
siodev.bctl = 3;                                  /*-->cr3b                           */
siodev.bctl = serial_params.cr3b;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$eject

/*cr5x*/
siodev.actl = 5;                          /*-->cr5a                                  */
siodev.actl = serial_params.cr5a;
siodev.bctl = 5;                          /*-->cr5b                                  */
siodev.bctl = serial_params.cr5b;

/*cr0x reset ext/st intrs to enable modem control sense--> autoenb chans.
  also --> cr1x, set intr params*/
siodev.actl = 00$010$001b;
siodev.actl = 0;                          /*no intrs                                 */
siodev.bctl = 00$010$001b;
siodev.bctl = 0;                          /*no intrs                                 */

end sio$init;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$subtitle ('PPORT -- centronics interface routines ')

/*
 * This module implements the initialization, LISTST, and LIST functions
 * for a Centronics-compatible parallel printer interface, using the
 * 6522 VIA chip.
 *
 * Our entry points are named pp$init, LPT$stat, and LPT$put respectively,
 * it's up to our caller to decode the I/O byte and call the approp-
 * riate routines.
 */
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
              $eject
 308   1      declare pp$base pointer;                         /* baseaddr for a 6522                    */
 309   1      declare pp based pp$base structure (              /* 6522 template                          */
                      rb byte,                                 /* out-in reg ´b´                         */
                      ra byte,                                 /* out-in reg ´a´                         */
                      ddrb byte,                               /* data-direction, reg ´b´                */
                      ddra byte,                               /* data-direction, reg ´a´                */
                      t1cl byte,                               /* t1 ctr(r)/lat(w) lo                    */
                      t1ch byte,                               /* t1 ctr hi                              */
                      t1ll byte,                               /* t1 latch lo                            */
                      t1lh byte,                               /* t1 latch hi                            */
                      t2cl byte,                               /* t2 ctr(r)/lat(w) lo                    */
                      t2ch byte,                               /* t2 ctr hi                              */
                      sr byte,                                 /* shift register                         */
                      acr byte,                                /* auxiliary ctrl reg                     */
                      pcr byte,                                /* peripheral ctrl reg                    */
                      ifr byte,                                /* interrupt flg register                 */
                      ier byte,                                /* interrupt enbl register                */
                      rax byte                                 /* out-in reg ´a´ NO HANDSHAKE            */
              );
              /*
               * Bit definitions for Centronics-style parallel interface, ´via1´.
               */
 310   1      declare via1$base literally ´0e8020h´;           /* baseaddr for this chip                 */
 311   1      declare ds$l literally ´01h´;                    /* data strobe (pb0)                      */
 312   1      declare pi$h literally ´02h´;                    /* this datum for vfu (pb1)               */
 313   1      declare bz$h literally ´20h´;                    /* printer busy (pb5)                     */
 314   1      declare ak$l literally ´40h´;                    /* printer ack (pb6)                      */
 315   1      declare sl$h literally ´80h´;                    /* on-line and no error (pb7)             */
              /*
               * Bit definitions for multi-use pio, ´via2´.
               */
 316   1      declare via2$base literally ´0e8040h´;           /* baseaddr for this chip                 */
 317   1      declare te$h literally ´01h´;                    /* talk-enable line                       */
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

```
$eject
/*
 * initial setup for parallel printer port
 * Note we use via2 during this setup to get talk-enable turned on, and
 * thus someone MUST ALREADY HAVE VIA2 INITIALIZED.
 */

pp$init: procedure public;
    pp$base = via2$base;                    /* point to secondary chip for te      */
    pp.rb = pp.rb or te$h;                  /* set 'talk enbl'                     */
    pp$base = via1$base;                    /* point struc at primary chip         */
    pp.ra = 0;                              /* ra is dataport, init with 0's       */
    pp.ddra = 0ffh;                         /* set all ra bits as outgoing         */
    pp.rb = ds$l;                           /* rb is ctrlport, init no ds/pi       */
    pp.ddrb = ds$l or pi$h;                 /* these 2 only are outgoing           */
                                            /* ca1/ca2 cb1/cb2 not used            */
                                            /* timers/shiftreg not used            */
end pp$init;
```

NOTE: Sample Programs only - Not BIOS REFERENCE LISTINGS!

```
$eject

/*
 * Test status of printer, return true if on-line and not busy, else
 * false.  For some reason, the Altos code explicitly deasserted data
 * strobe before testing; we´ll assume that this represents an Altos
 * fubar and is not required here.
 */
LPT$stat: procedure byte public;
    if (pp.rb and (sl$h or bz$h)) = sl$h then return 0ffh;
    return 0;
end LPT$stat;


/*
 * Put one character to the printer interface.
 */
LPT$put: procedure(ch) public;
   declare ch byte;
    do while LPT$stat = 0; end;              /* wait for printer ready         */
    pp.ra = ch;                              /* put outgoing char on the port  */
  disable;
    pp.rb = pp.rb and not ds$1;              /* assert data strobe             */
    pp.rb = pp.rb or ds$1;                   /* deassert data strobe           */
  enable;
    return;
end LPT$put;
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

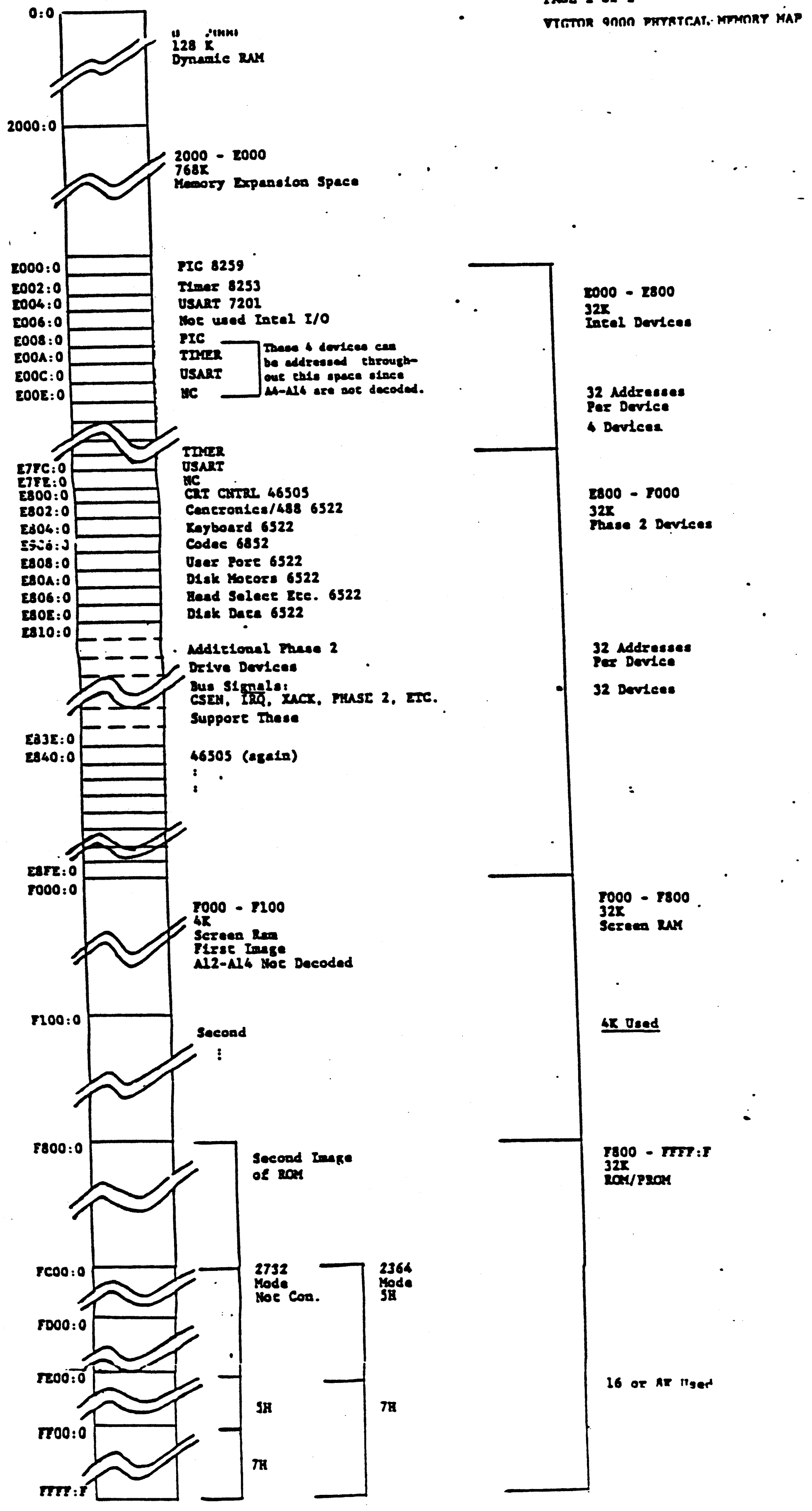
```
              $SUBTITLE ('Example software drivers for V9000 Hardware')

 343   1      end Hardware;




MODULE INFORMATION:

     CODE AREA SIZE     = 073EH   1854D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0014H     20D
     MAXIMUM STACK SIZE = 000EH     14D
     807 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
c
```

NOTE: Sample Programs only - Not BIOS Reference Listings!

Memory Map

PAGE 1 of 1

VICTOR 9000 PHYSICAL MEMORY MAP



0:0
128 K
Dynamic RAM
2000:0
2000 - E000
768K
Memory Expansion Space
E000:0
E002:0
E004:0
E006:0
E008:0
E00A:0
E00C:0
E00E:0
PIC 8259
Timer 8253
USART 7201
Not used Intel I/O
PIC
TIMER
USART
NC
These 4 devices can be addressed through-out this space since A4-A14 are not decoded.
E000 - E800
32K
Intel Devices
32 Addresses
Per Device
4 Devices
E7FC:0
E7FE:0
E800:0
E802:0
E804:0
E806:0
E808:0
E80A:0
E80C:0
E80E:0
E810:0
TIMER
USART
NC
CRT CNTRL 46505
Centronics/488 6522
Keyboard 6522
Codec 6852
User Port 6522
Disk Motors 6522
Head Select Etc. 6522
Disk Data 6522
Additional Phase 2
Drive Devices
Bus Signals:
CSEN, IRQ, XACK, PHASE 2, ETC.
Support These
E83E:0
E840:0
46505 (again)
E800 - F000
32K
Phase 2 Devices
32 Addresses
Per Device
32 Devices
E8FE:0
F000:0
F000 - F100
4K
Screen Ram
First Image
A12-A14 Not Decoded
F000 - F800
32K
Screen RAM
F100:0
Second
4K Used
F800:0
Second Image
of ROM
F800 - FFFF:F
32K
ROM/PROM
FC00:0
2732
Mode
Not Con.
2364
Mode
5H
FD00:0
FE00:0
5H
7H
FF00:0
7H
FFFF:F
16 or 8K Used