```
; O R B I T   Microcode
;
; Location 720b points to a control table (even address) with
; the following entries:
;0      Directive -- tells which function to execute
;1      Argument 1 -- first argument to the function
;2      Argument 2 -- second argument
;
; ...entries below here used only for directive 12 and 13 ...
;3      nBandsM1 -- number of bands (-1) to generate
;              A band is 16. scan-lines
;4      FA -- first read address for output (left half-word)
;5      LoTable -- pointer to left-over table
;              Must be even; must be initialized with @LoTable=0
;6      FontTable -- pointer to table of font characters:
;              (ICC is a mnemonic for 'internal character code')
;              FontTable!(ICC+100000b) => Fdes (must be even)
;                     Fdes!0=-Height (in bits)
;                     Fdes!1=Width-1
;                     Fdes!2,3,... bit map for character
;7      NewRp -- pointer to list of "new characters" :
;              NewRp!1 is first word and must be even
;              Each entry on NewRp is one of:
;              (a) A character (2 words).  First word is ICC+100000b
;                     Second word is xInBand (bit 4), Y (bit 12.)
;                     XY address is of lower left corner of box.
;              (b) A "rule" (4 words).  First word is 1.
;                     Second word is xInBand (bit 4), Y (bit 12.) -- same
;                     interpretation as for a character.
;                     Third word is -Height (in bits). Fourth word is
;                     Width-1.
;              (c) A band terminator (2 words, both =0) used to mean
;                     that no more information is required for this band.
;
; ...these entries are filled in by the microcode with values ...
;8.     Result -- if a function returns a result, it is placed here
;9.     Status -- records latest Orbit status
;              Some "firmware status" bits are added to the normal Orbit status
;              word:
;                     100000b -- Orbit is "in a character segment" (IACS)
;                     40000b -- Timeout in Slot band switch wait (Directive=14b)
;                     20000B -- ROS status WORD unstable (Directive=16b)
;
; Idling.  When a function is finished execution, the microcode sets
; location 720b to zero, and will then idle in one of two ways:
;      (a) Normal.  The Orbit task is blocked entirely; there is no activity.
;              A StartIO(4) is required to resume processing.
;      (b) Refreshing.  The microcode goes "to sleep," but will awake to
;              refresh the Orbit memory (the buffer being written into)
;              periodically.  If, upon awakening, a non-zero value is detected
;              in location 720b, it goes off to execute the specified function.
;              Awakening may be hastened by executing StartIO(4)
; (a) is default; turn on 100000b bit in Directive to get (b)
;
;
; F U N C T I O N S
;
; Directive=0: OrbitControl _ argument 1
; Directive=1: result _ OrbitData
; Directive=2: OrbitHeight _ argument 1,
;              result _ (if RefreshingNeeded then -1 else 0)
; Directive=3: OrbitXY _ argument 1
; Directive=4: OrbitDBCWidth _ argument 1
; Directive=5: OrbitFont _ argument 1
; Directive=6: result _ OrbitDeltaWC
; Directive=7: OrbitInk _ argument 1
; Directive=10b: result _ OrbitDBCWidth
; Directive=11b: OrbitROSCommand _ argument 1
; Directive=12b: Read data words (OrbitData)
```

```
;               argument 1 = count of words to read
;               argument 2 = address to put first word
; Directive=13b: Shovel a character to Orbit
;               OrbitHeight _ argument 1
;               OrbitXY _ argument 2
;               OrbitDBCWidth _ nBandsM1
;               for i=0 to abs(LoTable)-1 do OrbitFont_FA!i
;                       (use "tasking" loop if LoTable>0, else non-tasking)
;               result _ OrbitDBCWidth
;               NewRp _ OrbitDeltaWC
; Directive=14b: Process an entire page and send to Slot
;               argument 1 = pointer (even) to ROS command table.
;                       ROS command table entries are 2 words each:
;                               command bit 4   -- which command to do
;                               BandM1 bit 12. -- do it when nBandsM1 matches this field
;                               rosArgument bit 16. -- this is the argument
;                       Commands:
;                               0 -- Send rosArgument as a ROS command
;                               1 -- spare
;                               2 -- spare
;                               3 -- Wait (used if 2 ROS commands needed in same band)
;               argument 2 = Timeout count, in number of refresh cycles
;               result _ updated NewRp (remember it's 1 below next char)
; Directive=15b: Process a single band and return
;               result _ updated NewRp (remember it's 1 below next char)
; Directive=16b: Read a single word of ROS status (argument 1
;               contains, in the left half, the first address of ROS memory
;               to look in).
;               result _ status word
```

```
;Orbit microcode
#AltoConsts23.Mu;
;
;Orbit -- task 1 (just below the emulator)
; PackMu OrbitMc.Mb OrbitMc.Br 177774
;
; Comment conventions:
;       **      Put on any instruction after which a TASK switch might happen
;       !a,b    Branching between locations a and b may occur
;       //              Put on instructions inserted to make sure Alto clock does not
;                       stop during execution of an Orbit function.
;
;
;Orbit definitions
;
$ORBITDWC       $L0, 070014, 100;       F1=14 Alto_Delta Word Count
$ORBITDBCWID    $L26010, 70015, 120100; F1=15 Alto_Delta BC, Width (-1)
;                                       F2=10, Delta Bc, Width_Alto, Cinit
$ORBITDBCWIDx   $L16015, 0, 160000;     F1=15 (version that does not define BUS)
$ORBITXY        $L26011, 0, 120000;     F2=11, X,Y _ Alto
$ORBITHEIGHT    $L26012, 0, 120000;     F2=12, Height _ Alto !!!BRANCHES!!!
$ORBITFONT      $L26013, 0, 120000;     F2=13, Font data _ Alto
$ORBITINK       $L26014, 0, 120000;     F2=14, Ink data _ Alto
$ORBITCONTROL   $L26015, 0, 120000;     F2=15, Control, FA _ Alto
$ORBITROSCMD    $L26016, 0, 120000;     F2=16, Ros command _ Alto
$ORBITDATA      $L0, 70016, 100;        F1=16, Alto _ Output data
$ORBITSTATUS    $L0, 70017, 100;        F1=17, Alto _ Status !!!BRANCHES!!!
$ORBITSTATUSx   $L16017, 0, 160000;     F1=17 (version that does not define BUS)

$BUSUNDEF       $L0, 14002, 100;                Leaves BS=2 so bus is =-1
;

; Orbit Control bits:
$GOAWAY         $10;
$WHICH          $4;
$CLRFRESH       $2;
$CLRBEHIND      $20;
$SLOTTAKE       $300;           Includes "enable" bit
$STABLEROS      $40;
$600            $600;
$400            $400;
$6000           $6000;
$170000         $170000;

!17,20,NOVEM,ORBIT,,,,,,,,,,,,;
!17,20,START,,,,,,,,,,,,,,;

;Ye olde silent boot:
NOVEM:  SWMODE;
        :START;
;JumpRam(20b) will set reset mode register
$RMR            $L20013, 00000, 124000; set Reset Mode Register DF1=13
$AC0    $R3;
START:  RMR_AC0, :NOVEM;

; Global dispatch definitions:
!1,2,LOLOOP,NCHAR;              Return from TFRCHR
!1,2,BANDRET0,BANDRET1;         Return from DOBAND
!3,4,REFRET0,REFRET1,REFRET2,REFRET3;   Return from REFRESH
!17,20,DIR0,DIR1,DIR2,DIR3,DIR4,DIR5,DIR6,DIR7,DIR10,DIR11,DIR12,DIR13,DIR14,DIR15,DIR16,DIR17;

$DIRECTIVE      $R76;
$ARG1           $R75;
$ARG2           $R74;
$NBANDSM1       $R73;
$FA             $R72;
$LOTABLE        $R71;
$FONTTABLE      $R70;
```

```
$NEWRP          $R67;
$STATUS         $R66;
$RET0           $R64;
$RET1           $R63;
$LORP           $R62;
$LOWP           $R61;
$ORBBL          $R60;
$XY             $R57;
$BEGFADR        $R56;
 $ICC           $R56;
$C720           $R55;
$Shit           $R54;

$LASTL          $R40;           Last value of L in RAM

$TEMP           $R14;
$FONTADR        $R15;
$HEIGHT         $R16;
```

```
; ORBIT -- here to load state from the control block, and to
; dispatch on the directive.

!1,2,ORBITX,ORREFA;

ORBIT:  T_600;
        L_100+T, TASK;
        C720_L;                 ** Wait here on a block...
        T_20;
        MAR_L_C720+T;
        C720_L;
        L_MD, TASK;
        ORBBL_L;                ** ORBBL points to our control block

; Pick up the directive, argument 1
ORBITX: MAR_ORBBL;
        L_ORBBL+1;
        ORBBL_L;
        L_MD;
        T_MD;
        DIRECTIVE_L, L_T, TASK;
        ARG1_L;         **

; Pick up argument 2, nBandsM1
        MAR_L_ORBBL+1;
        L_LASTL+1;
        ORBBL_L;
        L_MD;
        T_MD;
        ARG2_L, L_T, TASK;
        NBANDSM1_L;     **

; Pick up FA, LoTable
        MAR_L_ORBBL+1;
        L_LASTL+1;
        ORBBL_L;
        L_MD;
        T_MD;
        FA_L, L_T, TASK;
        LOTABLE_L;      **

; Pick up FontTable, NewRp
        MAR_L_ORBBL+1;
        L_LASTL+1;
        ORBBL_L;
        L_MD;
        T_MD;
        FONTTABLE_L, L_T, TASK;
        NEWRP_L;        **

; Dispatch to appropriate function
        T_17;
        L_DIRECTIVE AND T;
        SINK_LASTL, BUS, L_T, TASK;
        :DIR0;  **

; ORBITDONE/STORE -- here when done with a function. Zero location
; 720b, and idle.  Idle is normally BLOCK; but will sit in a loop
; refreshing if high order bit of directive is on.
;       ORBITSTORE -- returns what is in L as result
;       ORBITDONEX -- XOR's what is in L with status

!1,2,ORQUIET,ORREF;
!4,5,IACS,,,,NOIACS;

ORBITSTORE: NEWRP_L;
ORBITDONE:      L_0, TASK;
ORBITDONEX: ICC_L;      **
```

```
        L_ORBITSTATUS;
        T_100000, :IACS;
IACS:   L_LASTL XOR T;
NOIACS: T_ICC;
        MAR_ORBBL+1;
        L_LASTL XOR T;
        MD_NEWRP, TASK;
        MD_LASTL;       **

;Shit
        MAR_C720+1;
        TASK;
        MD_Shit;        **

        MAR_C720;
        L_DIRECTIVE;
        SH<0, TASK;
        MD_0, :ORQUIET; **

; ORQUIET -- let Orbit be quiet between invocations.  Turn off
; the RUN flop with BLOCK, and wait for StartIO to get it going
; again.  Note that the block actually "takes" at the next TASK,
; just after ORBIT.

ORQUIET: BLOCK, :ORBIT;

; ORREF -- Orbit's idle loop will refresh if needed.  Whenever it wakes
; up and discovers a non-zero command block, it will resume execution.
; Issuing a StartIO at any time will simply hasten the wakeup.

ORREF:  ORBITCONTROL_GOAWAY;
        NOP, TASK;
REFRET3: NOP;   **      Wait for a good long time (2ms)
        MAR_C720;
        NOP;
        L_MD, BUS=0, TASK;
        ORBBL_L, :ORBITX;       ** !ORBITX,ORREFA

ORREFA: L_3, TASK, :REFRESH;    Return to REFRET3 on new wakeup
```

```
; Directive 0: Control _ argument 1
DIR0:   ORBITCONTROL_ARG1;
        NOP,    TASK;
        NOP, :ORBITDONE; **     -- Let us go away if GOAWAY was set in last instr

; Directive 1: argument _ Orbit data
DIR1:   L_ORBITDATA, :ORBITSTORE;

; Directive ?: Set Height, return -1 if refresh needed
!4,5,NONEEDREF,,,,NEEDREF;

DIR2:   ORBITHEIGHT_ARG1;
        L_0, :NONEEDREF;
NONEEDREF: :ORBITSTORE;
NEEDREF: L_ALLONES, :ORBITSTORE;

; Directive 3: Set XY _ argument 1
DIR3:   ORBITXY_ARG1, :ORBITDONE;

; Directive 4: Set delta BC, Width _ argument 1
DIR4:   ORBITDBCWID_ARG1, :ORBITDONE;

; Directive 5: Ship font data _ argument 1
DIR5:   ORBITFONT_ARG1, TASK;
        NOP, :ORBITDONE; **  In case FIFO full....

; Directive 6: Read delta WC
DIR6:   L_ORBITDWC, :ORBITSTORE;

; Directive 7: Set ink data _ argument 1
DIR7:   ORBITINK_ARG1, :ORBITDONE;

; Directive 10b: Read delta BC, Width
DIR10:  L_ORBITDBCWID, :ORBITSTORE;

; Directive 11b: ROS command _ argument 1
DIR11:  ORBITROSCMD_ARG1, :ORBITDONE;

; Directive 12b: Read orbitdata words.  First argument is count
; Second argument is address

!1,2,F12CONT,F12DON;

DIR12:  L_ARG2-1, TASK;
        TEMP_L; ** Temp=beginning core address-1
        T_TEMP;
        L_ARG1+T, TASK;
        ARG1_L; **              ARG=last address

; 6 instructions in the loop to allow Orbit memory adequate time.
; This could be made faster -- Orbit really needs only 4 instructions
; per ORBITDATA function.

F12CONT: MAR_L_T_TEMP+1;
        TEMP_L;
        L_ARG1-T;
        MD_ORBITDATA;
        SH=0, TASK;
        :F12CONT;       **

F12DON: NOP, :ORBITDONE;

; Directive 13b: Load Height, XY, then DBCWID, shovel
; a number of words at the interface, then read delta WC
;       Argument 1: Height
;       Argument 2: XY
;       NBANDSM1: Width
;       FA: pointer to data vector
```

```
;       LOTABLE: count (may be positive or negative -- see below)

!4,1,KILLHEIGHT13;
!1,2,CONTLOP,DONLOP;
!1,2,CONTLOPT,DONLOPT;
!1,2,LOPTA,LOPNT;

DIR13:  ORBITHEIGHT_ARG1;
        ORBITXY_ARG2, :KILLHEIGHT13;
KILLHEIGHT13:   ORBITDBCWID_NBANDSM1;
        L_T_LOTABLE;
        L_FA-1, SH<0;
        FA_L, L_T, :LOPTA;              !LOPTA,LOPNT

; Two versions of this loop. First one does not task, so logic
; analyzer can be used on small loops.
; Second one tasks so can be used on large characters.
; Count<0 selects non-tasking loop.

CONTLOP: MAR_L_FA+1;
        FA_L;
        L_LOTABLE+1;
        NOP;            //////////////////////
        ORBITFONT_MD;
LOPNT:  LOTABLE_L, SH=0;
        NOP, :CONTLOP;          **DONT DO A TASK**

CONTLOPT: MAR_L_FA+1;
        FA_L;
        L_LOTABLE-1;
        NOP;            //////////////////////
        ORBITFONT_MD;
LOPTA:  LOTABLE_L, SH=0, TASK;
        NOP, :CONTLOPT;         **DO A TASK**

!1,2,WAITLOP,WAITDONE;
DONLOPT: L_100, :WAITLOP0;
DONLOP: L_100;
WAITLOP0: TEMP_L, :WAITLOP;     **
WAITLOP: L_TEMP-1, BUS=0, TASK, :WAITLOP0;

; The following is a non-standard way of storing a result
; (it comes back in the word of the control table occupied by
; NewRp)
WAITDONE: MAR_ORBBL;
        NOP, TASK;
        MD_ORBITDWC;    **
        L_ORBITDBCWID, :ORBITSTORE;

; Directive 15b: Do one band

DIR15:  L_0+1, TASK, :DOBAND;
BANDRET1: :ORBITDONE;

; Directive 16b: Read one word of ROS Status

!1,2,WDSTLP,WDSTDN;
!1,2,MSH1,MSH2;
!1,2,MSHGOOD,MSHBAD;
!4,1,MSHKILL;
!4,1,MSHKILL1;
!4,1,MSHKILL2;

DIR16:  L_ORBITSTATUS;
        FA_L, :MSHKILL1;                Remember StableROS bit
MSHKILL1: L_3, TASK, :MSH3;

WDSTLP: L_3, :MSH0;
MSH1:   L_TEMP, TASK;
```

```
        TEMP _ L LSH 1; **
        L_ICC-1, BUS=0;
MSH0:   ICC_L, :MSH1;
MSH2:   T_400;
        L_ORBITCONTROL_ARG1+T;  L_ARG1+T, ORBITCONTROL_ARG1
        ARG1_L;
        T_17;
        T_ORBITSTATUS.T;                Branches
        L_TEMP + T, TASK, :MSHKILL;
MSHKILL: TEMP_L;        **
        L_XY-1, BUS=0, TASK;
MSH3:   XY_L, :WDSTLP;  **

WDSTDN: T_FA;
        T_ORBITSTATUS.T;
        L_STABLEROS AND T, :MSHKILL2;
MSHKILL2:       L_TEMP, SH=0;
        NEWRP_L, :MSHGOOD;
MSHGOOD:        :ORBITDONE;
MSHBAD: T_20000, :ORBITDONEX;
;
; Directive = 17b: Spare
;
DIR17:  :ORBITDONE;
```

```
; Slot Printing loop (directive=14b)
; Register values on entering:
;       FA -- first read address in band
;       NBANDSM1 -- NumberOfBands-1
;       ARG1 -- pointer to ROS command table of pairs (b, cmd)
;               where b=band number (first one to be executed is NBANDSM1,
;               and so on decreasing to 0).
;       ARG2 -- timeout count down
;       + see subrs
; Uses: RET0
; Calls: REFRESH, DOBAND

!4,1,NXROS;             Make sure NXROS will kill ORBITSTATUS branch
!1,2,SVBND,DIR14DN;
!1,2,NOROS,GIVEROS;
!1,2,NOTIMEOUT,TIMEOUT;
!4,5,NXBAND,,,,REFRSH2;

DIR14:  ORBITCONTROL_SLOTTAKE;
        T_WHICH;                Set FA properly.
        L_FA+T, TASK;
        ORBITCONTROL_LASTL;     **

; Ship out ROS commands in the command table.

NXROS:  MAR_ARG1;               Look for ROS command
        T_NBANDSM1;
        L_MD-T;
        T_7777;
        L_LASTL AND T;
        SH=0, TASK;
        :NOROS; ** !NOROS,GIVEROS

NOROS:  L_0, TASK, :DOBAND;     Go do most of the work.

; Should really check BEHIND here somewhere (but after setting GOAWAY).
BANDRET0: ORBITCONTROL_GOAWAY;
        L_ARG2, TASK;
        ICC_L;  **              Wait a long time here (2 ms)

REFRET2: ORBITHEIGHT_0; Branches!! check to see if refresh needed
        :NXBAND;                !NXBAND,REFRSH2

REFRSH2: L_ICC-1, BUS=0;
        ICC_L, :NOTIMEOUT;      !NOTIMEOUT,TIMEOUT
NOTIMEOUT: L_2, TASK, :REFRESH; Return to REFRET2 on new wakeup

; Time to begin generating a new band.

NXBAND: L_NBANDSM1-1, BUS=0, TASK;
        NBANDSM1_L, :SVBND;     **
SVBND:  T_4;
        MAR_ORBBL-T;            Update the command block to show progress
        TASK;
        MD_NBANDSM1, :NXROS;    **

TIMEOUT: L_40000, :ORBITDONEX;
DIR14DN: :ORBITDONE;
```

```
; Dispatch on ROS command table entry (high 4 bits)

%360,360,0,ROS0,ROS1,ROS2,ROS3;

GIVEROS: MAR_ARG1;
        T_2;
        L_ARG1+T;
        ARG1_L;
        T_30000;
        L_MD AND T;
        TEMP_ L LCY 8;
        L_MD, TASK;
        RET0_L; **
        SINK_TEMP, BUS, TASK;
        :ROS0;

; Op code 0: send a 16-bit ROS command
ROS0:   ORBITROSCMD_RET0, :NXROS;

; Op code 1: read status
ROS1:   ORBITCONTROL_RET0;
        MAR_ARG1-1;
        NOP;
        MD_ORBITSTATUS; Warning--may or 4 into NEXT
        :NXROS;

; Op code 2: spare
ROS2:   :NXROS;

; Op code 3: wait in a loop -- argument tells how long.
ROS3:   NOP;
!1,2,ROS3A,ROS3B;
ROS3A:  L_RET0-1, BUS=0, TASK;
        RET0_L, :ROS3A; **
ROS3B:  :NXROS;
```

```
;DOBAND -- enter here to process a band.
;       L = index of return (BANDRET0,BANDRET1)
;       NEWRP => next new character -1
;       LOTABLE => left-over table (@LOTABLE=0 at the very first)
;       FONTTABLE => ICC table (-#100000)
; Uses RET1,LORP,LOWP,FONTADR,HEIGHT,XY,ICC
; Calls REFRESH, TFRCHR

DOBAND: RET1_L; **
        L_LOTABLE-1;
        LORP_L, TASK;
        LOWP_L; **

; Process left-overs
!4,5,NOREFRSH0,,,,REFRSH0;
!1,2,LOLOOPA,NOLOV;

LOLOOP: MAR_L_LORP+1;
        L_LASTL+1;
        LORP_L;
        NOP;            ////////////////////////
        L_ORBITHEIGHT_MD;       ORBITHEIGHT branches!!
        HEIGHT_L, SH=0, :NOREFRSH0;
NOREFRSH0: L_ORBITXY_MD, TASK, :LOLOOPA;

LOLOOPA: XY_L;  **
        MAR_L_LORP+1;
        L_LASTL+1;
        LORP_L;
        NOP;            ////////////////////////
        ORBITDBCWID_MD;
        L_MD;
        FONTADR_L, L_0, TASK, :TFRCHR;  L=0 => return to LOLOOP

; Come to REFRSH0 to refresh when in the left-over
; loop.  Backs up the LORP pointer and returns to LOLOOP.
!1,1,REFRSH0A;  Kill SH=0 coming from above
REFRSH0: T_2, :REFRSH0A;
REFRSH0A: L_LORP-T, TASK;               Back up the read pointer
        LORP_L; **
        L_0, TASK, :REFRESH;
REFRET0: :LOLOOP;

; Come to NCHAR when finished with left-over table (terminating
; 0 encountered).

; Process "new characters" from the main list.

!4,5,NOREFRSH1,,,,REFRSH1;
!1,2,REGRULE,ENDBAND;
!1,2,NOTCHAR,REGCHAR;

NOLOV:  NOP;    **
NCHAR:  MAR_L_NEWRP+1;
        L_LASTL+1;
        NEWRP_L;
        L_MD;
        ORBITXY_T_MD;
        ICC_L, L_T, SH<0, TASK;
        XY_L, :NOTCHAR; **

REGCHAR: L_T_ICC;
        MAR_FONTTABLE+T;
        NOP;
        L_MD+1, TASK;
        FONTADR_L;      **
        MAR_FONTADR-1;
        L_FONTADR+1;
```

```
        NOP;            //////////////////////
GETHW:  FONTADR_L;
        ORBITHEIGHT_L_MD;       Branches!!
        HEIGHT_L, L_0+1, :NOREFRSH1;

NOREFRSH1: ORBITDBCWID_MD, TASK, :TFRCHR;       L=1 => return to NCHAR

!4,1,KILLHEIGHT0;

REFRSH1: L_MD, TASK;
        ICC_L;          ** Save DBCWID
        L_0+1, :REFRESH;
REFRET1: ORBITHEIGHT_HEIGHT;    Branches!!
        ORBITXY_XY, :KILLHEIGHT0;
KILLHEIGHT0: ORBITDBCWID_ICC;
        L_0+1, TASK, :TFRCHR;

; If not a character, check for a rule.

NOTCHAR: SINK_ICC, BUS=0, TASK;
        :REGRULE;       **      !REGRULE,ENDBAND

; Do a rule.  Note that XY is already processed, and given to Orbit
REGRULE: MAR_L_NEWRP+1;
        L_LASTL+1;
        NEWRP_L;
        L_0, :GETHW;

; We have an "end band" signal in the new character list.

ENDBAND: MAR_LOWP+1;            Terminate the left over list
        SINK_RET1, BUS, TASK;
        MD_0, :BANDRET0;
```

```
; TFRCHR -- Transfer a character part to the Orbit.
;
; Register values on entering (*** means already passed to Orbit):
;       L -- "return" address (0=>LOLOOP; 1=>NCHAR)
;       XY -- xy position in Orbit format (ORBITXY)  ***
;       HEIGHT -- height in Orbit format (ORBITHEIGHT) ***
;       FONTADR -- pointer to first data word of the font character
;               If FONTADR=0, it will be a "rule" instead
;       (bc,width) -- has already been given to Orbit (ORBITDBCWID_) ***
;       LOWP -- write pointer into left overs
; Uses RET0, BEGFADR
;
!7,10,FONTODD,FONTEVEN,,,FONTDONE,,,;
!1,2,FONTLA,FONTRULE;
!1,2,FONTLO,FONTDN;
!4,5,FONTRULEC,,,,FONTRULED;

TFRCHR: RET0_L; **      Save return address
        MAR_T_FONTADR, BUS=0;
        L_ONE AND T, :FONTLA;   Check to see if first address odd or even
FONTLA: L_FONTADR+1, SH=0;
        BEGFADR_L, :FONTODD;

FONTLP: MAR_L_FONTADR+1;
        NOP;            //////////////////////
        L_LASTL+1;
FONTEVEN: ORBITSTATUSx, FONTADR_L;      Branches!! check to see if done
        ORBITFONT_MD, TASK, :FONTODD;
FONTODD: ORBITFONT_MD, :FONTLP; **

; Come here to put out a "rule" -- just give Orbit all -1's
; as font bits.
FONTRULE: ORBITSTATUSx; Check to see if finished
        ORBITFONT_BUSUNDEF, TASK, :FONTRULEC;
FONTRULEC: ORBITFONT_BUSUNDEF, :FONTRULE;       **

; Now undo the effect on bumping the Font Address

FONTRULED: NOP;**
        T_ORBITDWC;
        L_ONE-T, TASK;
        BEGFADR_L, :FONTDNX;    **

FONTDONE: NOP;  **
FONTDNX: T_7777, ORBITDBCWIDx;  Read remaining width
        L_7777-T;
        T_ORBITDWC-1;
        L_BEGFADR+T, SH=0, TASK;
        FONTADR_L, :FONTLO;     **

; Left over to record. Format:
;               word 0: height
;               word 1: y (x=0)
;               word 2: dbc,width
;               word 3: font address

FONTLO: T_7777;
        L_XY AND T, TASK;
        BEGFADR_L;      **

        MAR_L_LOWP+1;
        L_LASTL+1;
        MD_HEIGHT;
        MD_BEGFADR, TASK;
        LOWP_L; **

        MAR_L_LOWP+1;
        L_LASTL+1;
```

```
        MD_ORBITDBCWID;
        MD_FONTADR, TASK;
        LOWP_L; **

FONTDN: SINK_RET0, BUS, TASK;
        :LOLOOP;        **
```

```
; REFRESH -- the place that all refreshing is done (for now)
; Requires about 35*3+7 = 112 microcycles
; Register values on entering:
;       L -- return index
; Uses RET0, TEMP(R) -- but doesn't change it

; Note that a good deal of Orbit state is destroyed.
; Calculation of number of times through loop:
;       let n=number to store in L at beginning. (n+1)*2=64.+6.
;       6 is to leave enough in the FIFO.

!4,5,REFLPA,,,,REFLPD;
!4,1,REFLP;             Kill height branch

REFRESH: RET0_L;        **
        ORBITXY_0;
        ORBITHEIGHT_6000;       Branches!! 1024. bits = 64. words
        ORBITDBCWID_0, :REFLP;
REFLP:  L_TEMP, ORBITSTATUSx;   Branches!! check to see if done.
        ORBITFONT_0, TEMP_L, TASK, :REFLPA; TEMP_L just to hold BUS=0
REFLPA: ORBITFONT_0, :REFLP;    **

; Note about the exit code.  If GOAWAY is set, the REFRESH subroutine
; does not return until a new wakeup is generated (either by
; another refresh request or by someone clearing GOAWAY: StartIO
; or buffer-switch).

REFLPD: ORBITCONTROL_CLRFRESH;  **
        SINK_RET0, BUS, TASK;
        :REFRET0;       **              Usually hangs up here if no wakeup
```