This is the only information available regarding physical page 0 on the Geneve 9640

```
        DEF  HEAD#T,TAIL#T,TASK#C
        DEF  CONTEX
*
INTREG  EQU  >F080
XOPREG  EQU  >F0A0
SYS1    EQU  >F0A0
SYS2    EQU  >F0C0
SYSREG  EQU  >F0E0
SCRATC  EQU  >F0E0
MAPPER  EQU  >F110
*
        AORG 0
*
RSETVC  DATA >F000,OSINT0
        DATA INTREG,OSINT1      9901 interrupt
        DATA INTREG,OSINT2      MID trap
        DATA INTREG,OSINT3      internal timer interrupt
        DATA INTREG,OSINT4      external bus interrupt
*
        AORG >20
LIBTAB  DATA XOPTAB,PWRTAB                      LIBRARY USAGE
MAXXOP  DATA XOPCNT,XOPUSE

HEAD#T  DATA TASK#0
TAIL#T  DATA TASK#N
TASK#C  DATA TASK#0
*
MOVLOC  DATA MOVTAB
MOVLST  DATA MOVEND
*
        AORG >40                XOP table
SYSXOP  DATA SYS2,XOPCAL
        DATA >FFD8,>FFF8        USER #1         RSBUG
        DATA 0,0                USER #2
        DATA 0,0                USER #3
        DATA 0,0                USER #4
        DATA 0,0                USER #5
        DATA 0,0                USER #6
        DATA 0,0                USER #7
        DATA 0,0                USER #8
        DATA 0,0                USER #9
        DATA 0,0                USER #10
        DATA 0,0                USER #11
        DATA 0
*
*
```

This is the only information available and documented regarding the Geneve 9640 and your Task 0 Page

```
        DEF  FREPTR,CURDRV,MAXDRV,ALIASA
        DEF  PMTSTR,PATH#P,GPLPAR,ALIASP
        DEF  TSKMAP,BREAK,CTLP,CTLS
        DEF  CMDSTR
*
        DEF  TURX,TURXMN,TURXMX
        DEF  TURY,TURYMN,TURYMX
        DEF  PAGE,PAGEPX
        DEF  COTASK
        DEF  LINFLG,LINPTR
        DEF  ESCSEQ,ESCCNT,ESCROU,ESCDAT,STDCLR
        DEF  DELON
*
INTREG  EQU  >F080
XOPREG  EQU  >F0A0
SYS1    EQU  >F0A0
SYS2    EQU  >F0C0
SYSREG  EQU  >F0E0
SCRATC  EQU  >F0E0
MAPPER  EQU  >F110
*
        AORG 0
*
RSETVC  DATA INTREG,KILLIT
        DATA INTREG,INT1        9901 interrupt
        DATA INTREG,INT2        MID trap
        DATA INTREG,INT3        internal timer interrupt
        DATA INTREG,INT4        external bus interrupt
*
        DATA 0,0
*
INT1    LIMI 0
        LI   R11,1
        JMP  INTCAL
*
INT2    LIMI 0
        LI   R11,2
        JMP  INTCAL
*
INT3    LIMI 0
        LI   R11,3
        JMP  INTCAL
*
INT4    LIMI 0
        LI   R11,4
        JMP  INTCAL
*
        AORG >40                XOP table
SYSXOP  DATA XOPREG,XOPCAL
        DATA >FFD8,>FFF8        USER #1         RSBUG
        DATA XOPREG,ABORTX      USER #2
```

```
        DATA XOPREG,ABORTX      USER #3
        DATA XOPREG,ABORTX      USER #4
        DATA XOPREG,ABORTX      USER #5
        DATA XOPREG,ABORTX      USER #6
        DATA XOPREG,ABORTX      USER #7
        DATA XOPREG,ABORTX      USER #8
        DATA XOPREG,ABORTX      USER #9
        DATA XOPREG,ABORTX      USER #10
        DATA XOPREG,ABORTX      USER #11
        DATA XOPREG,ABORTX      USER #12
        DATA XOPREG,ABORTX      USER #13
        DATA XOPREG,ABORTX      USER #14
        DATA XOPREG,ABORTX      USER #15
*
        AORG >80                start after XOPS
*
XOPCAL  LIMI 0
        MOV  *R11,R11
INTCAL  MOV  @MAPPER,@TSKMAP
        MOV  @MAPPER,@SYSREG
        MOV  @MAPPER+2,@TSKMAP+2
        MOV  @MAPPER+4,@TSKMAP+4
        MOV  @MAPPER+6,@TSKMAP+6
        CLR  @MAPPER
        NOP                                     we page OS here...
ABORTX  RTWP
*
KILLIT  LIMI 0
        CLR  R11
        JMP  INTCAL
*
        DATA 0,0,0,0,0,0
        DATA 0,0,0,0,0,0,0,0
*
        DATA 0,0,0,0
DELON   DATA 0
ESCSEQ  DATA 0
ESCCNT  DATA 0
ESCROU  DATA 0
ESCDAT  DATA 0
STDCLR  DATA 0
LINFLG  DATA 0
LINPTR  DATA 0
COTASK  DATA 0
PAGE    DATA 0
PAGEPX  DATA 0
TURX    DATA 0
TURXMN  DATA 0
TURXMX  DATA 0
TURY    DATA 0
TURYMN  DATA 0
TURYMX  DATA 0
CTLP    DATA 0
CTLS    DATA 0
```

```
BREAK  DATA 0
*
       AORG >0100
TSKTBL BYTE 00          TASK ID #
MAXDRV BYTE 'G'         MAXIMUM ALIAS LETTER USER CAN SPECIFY
STATE  BYTE >FF         PROCESS STATE
SLICE  BYTE 6           NUMBER OF SLICES LEFT UNTIL SWAPPED OUT
PNAME  TEXT '           '   NAME OF THIS TASK
UWP    DATA >F000       USER WORKSPACE POINTER
UPC    DATA >0400       USER PROGRAM COUNTER
UST    DATA >0002       USER STATUS REGISTER
MEMLST DATA 0           POINTER TO MEMORY LIST
TSKMAP DATA 0,0,0,0     SAVED MEMORY MAP USED DURING XOPS AND INTERRUPTS
CURDRV DATA ALIASA      POINTER TO CURRENT DRIVE ENTRY
PATH#P DATA 0           POINTER TO TEXT FOR PATH COMMAND
BLKDEV DATA 0           POINTER TO NAMES OF BLOCK DEVICES
FREPTR DATA FRENOD      POINTER TO FREE NODES
PMTSTR DATA STRPMT      POINTER TO PROMPT STRING'S NODES
GPLPAR DATA 0           STRING TO CONTROL SPEED OF GPL INTERPRETER
CMDSTR DATA 0           STRING CONTAINING COMMAND LINE OPTIONS
*
ALIASA DATA DSK1,0      POINTERS TO THE ALIAS STRINGS
ALIASB DATA DSK2,0
ALIASC DATA DSK3,0
ALIASD DATA DSK4,0
ALIASE DATA HDS1,0
ALIASF DATA HDS2,0
ALIASG DATA HDS3,0
ALIASH DATA 0,0
ALIASI DATA 0,0
ALIASJ DATA 0,0
ALIASK DATA 0,0
ALIASL DATA 0,0
ALIASM DATA 0,0
ALIASN DATA 0,0
ALIASO DATA 0,0
ALIASP DATA 0,0
*
*
* for all handles, 0 input is keyboard, 0 output is screen
*
HANDL0 DATA 0               stdin
HANDL1 DATA 0               stdout
HANDL2 DATA 0               stderr
HANDL3 DATA 0
HANDL4 DATA 0
HANDL5 DATA 0
HANDL6 DATA 0
HANDL7 DATA 0
HANDL8 DATA 0
HANDL9 DATA 0
HANDLA DATA 0
*
       AORG >0180
```

```
SAVPAD BSS  128
*
       AORG >0200
STRPMT DATA >0000,>04*256+'$','n$','g'*256+>00    default prompt
*
DSK1   DATA 0,>04*256+'D','SK','1'*256+>00
DSK2   DATA 0,>04*256+'D','SK','2'*256+>00
DSK3   DATA 0,>04*256+'D','SK','3'*256+>00
DSK4   DATA 0,>04*256+'D','SK','4'*256+>00
HDS1   DATA 0,>04*256+'H','DS','1'*256+>00
HDS2   DATA 0,>04*256+'H','DS','2'*256+>00
HDS3   DATA 0,>04*256+'H','DS','3'*256+>00
FRENOD
FRES00 DATA FRES01,0,0,0
FRES01 DATA FRES02,0,0,0
FRES02 DATA FRES03,0,0,0
FRES03 DATA FRES04,0,0,0
FRES04 DATA FRES05,0,0,0
FRES05 DATA FRE1,0,0,0
FRE1   DATA FRE2,0,0,0
FRE2   DATA FRE3,0,0,0
FRE3   DATA FRE4,0,0,0
FRE4   DATA FRE5,0,0,0
FRE5   DATA FRE6,0,0,0
FRE6   DATA FRE7,0,0,0
FRE7   DATA FRE8,0,0,0
FRE8   DATA FRE9,0,0,0
FRE9   DATA FREA,0,0,0
FREA   DATA FREB,0,0,0
FREB   DATA FREC,0,0,0
FREC   DATA FRED,0,0,0
FRED   DATA FREE,0,0,0
FREE   DATA FREF,0,0,0
FREF   DATA FREG,0,0,0
FREG   DATA FREH,0,0,0
FREH   DATA FREI,0,0,0
FREI   DATA FREJ,0,0,0
FREJ   DATA FREK,0,0,0
FREK   DATA FREL,0,0,0
FREL   DATA FREM,0,0,0
FREM   DATA FREN,0,0,0
FREN   DATA FREO,0,0,0
FREO   DATA FREP,0,0,0
FREP   DATA FREQ,0,0,0
FREQ   DATA FRER,0,0,0
FRER   DATA FRES,0,0,0
FRES   DATA FRET,0,0,0
FRET   DATA FREU,0,0,0
FREU   DATA FREV,0,0,0
FREV   DATA FREW,0,0,0
FREW   DATA FREX,0,0,0
FREX   DATA FREY,0,0,0
FREY   DATA FREZ,0,0,0
FREZ   DATA FRFA,0,0,0
```

```
FRFA    DATA FRFB,0,0,0
FRFB    DATA FRFC,0,0,0
FRFC    DATA FRFD,0,0,0
FRFD    DATA FRFE,0,0,0
FRFE    DATA FRFF,0,0,0
FRFF    DATA FRFG,0,0,0
FRFG    DATA FRFH,0,0,0
FRFH    DATA FRFI,0,0,0
FRFI    DATA FRFJ,0,0,0
FRFJ    DATA FRFK,0,0,0
FRFK    DATA FRFL,0,0,0
FRFL    DATA FRFM,0,0,0
FRFM    DATA FRFN,0,0,0
FRFN    DATA FRFO,0,0,0
FRFO    DATA 0,0,0,0
*
ENDNOD EQU  $
*
        END
*
```

```
MYREG   EQU   >F000          Program Registers
MYREG1  EQU   >F040          Subroutine Registers
MYREG2  EQU   >F060          Subroutine Registers

MYREG3  BSS   >20            Proportion Registers
MYREG4  BSS   >20            Redraw Registers

TEMPWS  BSS   >20            Temporary workspace

*
* I/O Port Addresses
*

VDP0    EQU   >F108          VDP I/O Port #0
VDP1    EQU   >F10A          VDP I/O Port #1
VDP2    EQU   >F10C          VDP I/O Port #2
VDP3    EQU   >F10E          VDP I/O Port #3

*
* Library Call XOP Vectors
*
SAVER0  DATA 0               TEMP R0
SAVR10  DATA 0               TEMP R10
KSCAN   DATA >0005           Keyboard scan library
VIDEO   DATA >0006           VDP access library
MEMMAN  DATA >0007           Memory management library
IO      DATA >0008           I/O access library
UTIL    DATA >0009           Utility library
MATH    DATA >000A           Math library
WIND    DATA >000C           Windows library

MRESTR  BYTE >00,>01,>00,>00
        BYTE >00,>00,>00,>00
        BYTE >D4,>00,>00,>00
        BYTE >E0
        EVEN
*
* Subprogram BLWP Vectors
*

VSBR    DATA MYREG1,VSBR0        VDP Single Byte Read
VMBR    DATA MYREG1,VMBR0        VDP Multiple Byte Read
VSBW    DATA MYREG1,VSBW0        VDP Single Byte Write
VMBW    DATA MYREG1,VMBW0        VDP Multiple Byte Write
VWTR    DATA MYREG1,VWTR0        VDP Write to Register
VMWR    DATA MYREG1,VMWR0        VDP Multiple Write to Registers

*
* Program Subroutines
*

*
* SETVWA   Set VDP Write Address
```

```
*
SETVWA MOV   *R13,R0       Get VDP address
       ORI   R0,>4000      Set write bit
       JMP   SETVA1        Continue with routine

*
* SETVRA    Set VDP Read Address
*
SETVRA MOV   *R13,R0       Get VDP address

SETVA1 MOV   @2(R13),R1    Get buffer pointer
       MOV   @4(R13),R2    Get byte count
       LIMI  0             No interrupts during VDP access

SETVA2 MOV   R11,R10       Save return address
       BL    @CMDEND       Wait for VDP to be ready

SETVA3 SWPB  R0            LSB goes first
       MOVB  R0,@VDP1      Write it
       SWPB  R0            MSB goes next
       MOVB  R0,@VDP1      Write it
       B     *R10          Return to caller

*
* VMBW      VDP Multiple Byte Write
*
* Pass:     R0  VDP Address
*           R1  CPU Data Address
*           R2  Number of Bytes to Write
*
VMBW0  BL    @SETVWA       Set VDP write address
       MOV   R2,R2         Zero byte count?
       JEQ   VMBW2         Yes, Return to caller

VMBW1  MOVB  *R1+,@VDP0    No, Write a byte
       DEC   R2            Any bytes left?
       JNE   VMBW1         Yes, Write another

VMBW2  RTWP                No, Return to caller

*
* VMBR      VDP Multiple Byte Read
*
* Pass:     R0  VDP Address
*           R1  CPU Buffer Address
*           R2  Number of Bytes to Read
*
```

```
VMBR0   BL    @SETVRA       Set VDP read address
        MOV   R2,R2         Zero byte count?
        JEQ   VMBR2         Yes, Return to caller

VMBR1   MOVB  @VDP0,*R1+    No, Read a byte
        DEC   R2            Any bytes left?
        JNE   VMBR1         Yes, Read another

VMBR2   RTWP                No, Return to caller


*
* FILL      VDP Fill with a Byte
*
* Pass:     R0  VDP Address
*           R1  MSB  Byte to Fill with
*           R2  Number of Bytes to Fill
*

FILL0   BL    @SETVWA       Set VDP write address
        MOV   R2,R2         Zero byte count?
        JEQ   FILL2         Yes, Return to caller

FILL1   MOVB  R1,@VDP0      No, Write a byte
        DEC   R2            Any bytes left?
        JNE   FILL1         Yes, Write another

FILL2   RTWP                No, Return to caller


*
* VSBW      VDP Single Byte Write
*
* Pass:     R0  VDP Address
*           R1  MSB  Byte to Write
*

VSBW0   BL    @SETVWA       Set VDP write address
        MOVB  R1,@VDP0      Write the byte
        RTWP                Return to caller


*
* VSBR      VDP Single Byte Read
*
* Pass:     R0  VDP Address
*
* Return:   R1  MSB  Byte Read
*

VSBR0   BL    @SETVRA       Set VDP read address
        MOVB  @VDP0,@2(R13)     Read the byte
        RTWP                Return to caller
```

```
*
* VWTR      VDP Write to Register
*
* Pass:     R0  MSB  Register Number
*               LSB  Byte to Write
*

VWTR0  MOV  *R13,R0       Get register and data
       ORI  R0,>8000      Set register bit
       LIMI 0             No interrupts during VDP access
       BL   @SETVA2       Write data to VDP register
       RTWP               Return to caller


*
* VMWR      VDP Multiple Write to Register
*
* Pass:     R0  Starting Register Number
*           R1  CPU Data Address
*           R2  Number of Registers to Write to
*

VMWR0  MOV  *R13,R0       Get starting register
       ORI  R0,>9100      Add on register 17
       MOV  R10,@SAVR10   Save R10 (for CHAR)
       BL   @SETVA1       Set up starting register
       MOV  @SAVR10,R10   Restore R10
       MOV  R2,R2         Zero byte count?
       JEQ  VMWR2         Yes, Return

VMWR1  MOVB *R1+,@VDP3    No, Send a byte
       DEC  R2            Any more bytes left?
       JNE  VMWR1         Yes, Send another

VMWR2  RTWP               No, Return to caller


*
* CMDEND    Waits for a VDP Command to Finish
*

CMDEND MOV  R0,@SAVER0    Save contents of R0

CMDEN1 LI   R0,>028F      Set up to read status reg 2
       MOVB R0,@VDP1      Write LSB first
       SWPB R0            Get at the MSB
       MOVB R0,@VDP1      Write MSB to register 15
       SWPB R0            Waste some time
       MOVB @VDP1,R0      Get status register 2 data
       SRL  R0,9          Is CE bit set?
       JOC  CMDEN1        Yes, Command not finished yet

       MOV  @SAVER0,R0    No, Restore R0 data
       RT                 Return to caller
```