NEC

µPD6464, 6465
On-Screen Character Display

Application Note

December 1997

Introduction

This application note describes how to use the µPD6464 or µPD6465 on-screen display with a µPD784026 8/16-bit microcontroller. (In this document, µPD646x means µPD6464 or µPD6465.)

Note: If there are any questions regarding this application note, please contact Technical Support at (800) 366-9782 for help.

NEC builds CMOS LSIs that create character displays to accompany pictures on televisions, VCRs, and camera monitors. An on-screen character display device consists of a clock oscillator circuit that generates clocks in synchronization with TV signals, a character ROM that stores character data, a video RAM, and a circuit that controls the output of display data and backgrounds.

These CMOS LSI devices can overlay video information on top of an incoming video signal (from a VCR, for example), or they can generate the needed horizontal (Hsync) and vertical (Vsync) synchronizing pulses from an internal clock source.

The µPD6464 displays 128 different characters including lower-case and upper-case alphanumeric characters and character patterns such as Kanji, Hiragana, and Katakana. The µPD6465 contains 256 different characters, which vary by mask code option.

Each character consists of 12 horizontal dots by 18 vertical dots. Pictograms are made possible by combining two or more characters. The 12-line by 24-column display area can hold up to 288 characters. Four different character attributes can be selected with three blinking frequencies. If no external signal exists, the user can program the internal mode to generate the needed video signals with a blue, black, green, or white screen background.

Video RAM data is cleared by a video RAM clear command and the power-on clear functions. Both capabilities have been included in the device to mitigate the workload of the host microcontroller. The device also has a separation circuit for composite signal synchronization and a x4 multiplier. This circuit eliminates the need to connect an external separator IC and a crystal resonator, reducing the mounting area and total cost. Although the IC has an internal sync separation circuit, the user must strip Hsync and Vsync and combine them into a composite sync signal (C-sync).

Using the µPD646x

The on-screen display has a three-line serial interface to receive commands and data from a microcontroller. The µPD646x uses 8-bit and 16-bit instructions to perform all of its functions.

Figure 1 is a schematic of the µPD646x configured in the internal mode. The color background is generated internally, overlaid with on-screen character data and then driven out of the µPD646x at the Video Out jack. The NRE pin (pin 20) performs noise reduction.

Figure 2 is a schematic of the µPD646x configured in the external mode. The top left portion of the schematic shows the video input circuit that clamps the composite video input signal to the proper level. The input signal consists of the negative synchronization signals (Hsync/Vsync) and positive video signals. In the external

mode, the user must strip Hsync and Vsync from the external source using the sync separation circuit shown on the top middle portion of the schematic in Figure 2. These two signals must be combined into a composite sync signal (C-sync) that drives pin 17.

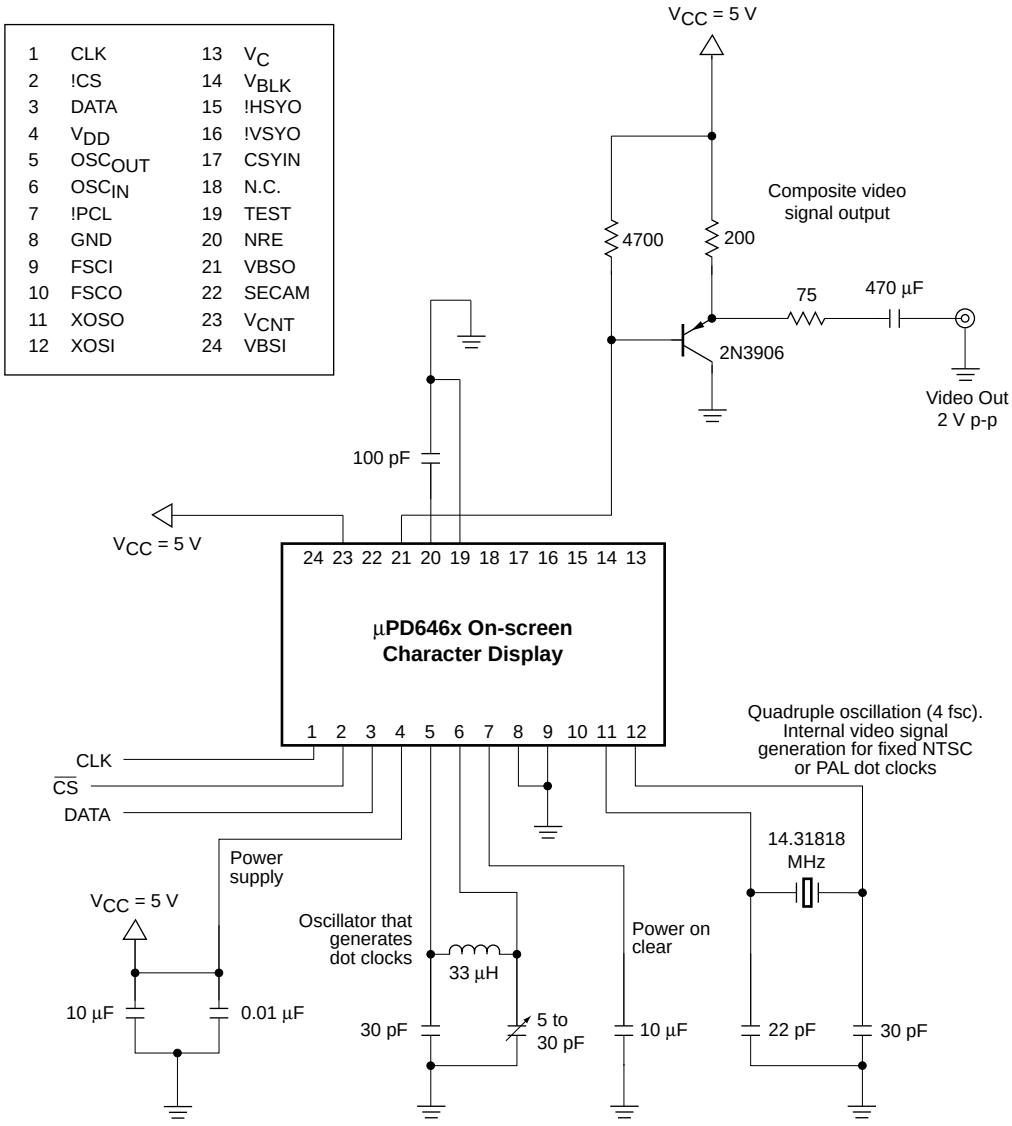
A sync separator chip such as the EL4581C from Elantec can also be used instead of the discrete sync separation circuit. The EL4581C chip extracts timing information from the standard video sync signals found in NTSC, PAL, and SECAM broadcast systems. Figure 3 shows how the EL4581C could be combined with the circuit in Figure 2. Please be advised that this circuit was not tested during the making of this application note.

The LC oscillator shown on the lower middle side of the schematic (pins 5 and 6) generates the character dot clock. When no character is displayed, LC oscillation can be stopped with the display control command to reduce power consumption. Remember that data cannot be written to the video RAM with the oscillation stopped. To write data to the video RAM, be sure to turn ON the LC oscillation.

The lower-right side of the schematic contains the crystal oscillator (pins 11 and 12) for sync pulse generation. The µPD646x uses this oscillator to generate internal NTSC or PAL dot clocks. If the user is operating in external mode and an external sync source generates the needed video input signal, this crystal oscillator is not required. In this application note, the NTSC protocol was fixed by connecting the crystal oscillator on pins 11 and 12. For demonstration purposes, our software initialized the µPD646x for NTSC. If your application requires the ability to switch between NTSC/PAL/SECAM, you must use the external oscillator circuitry shown in Figure 8.

Note that all the PNP transistors shown in Figures 1, 2, and 3 are 2N3906s and all the NPN transistors are 2N3904s.

Figure 1. µPD646x Internal Mode Circuitry



97YL-0275C (10/97)

Figure 2. µPD646x with Discrete Hsync and Vsync Separator Circuits

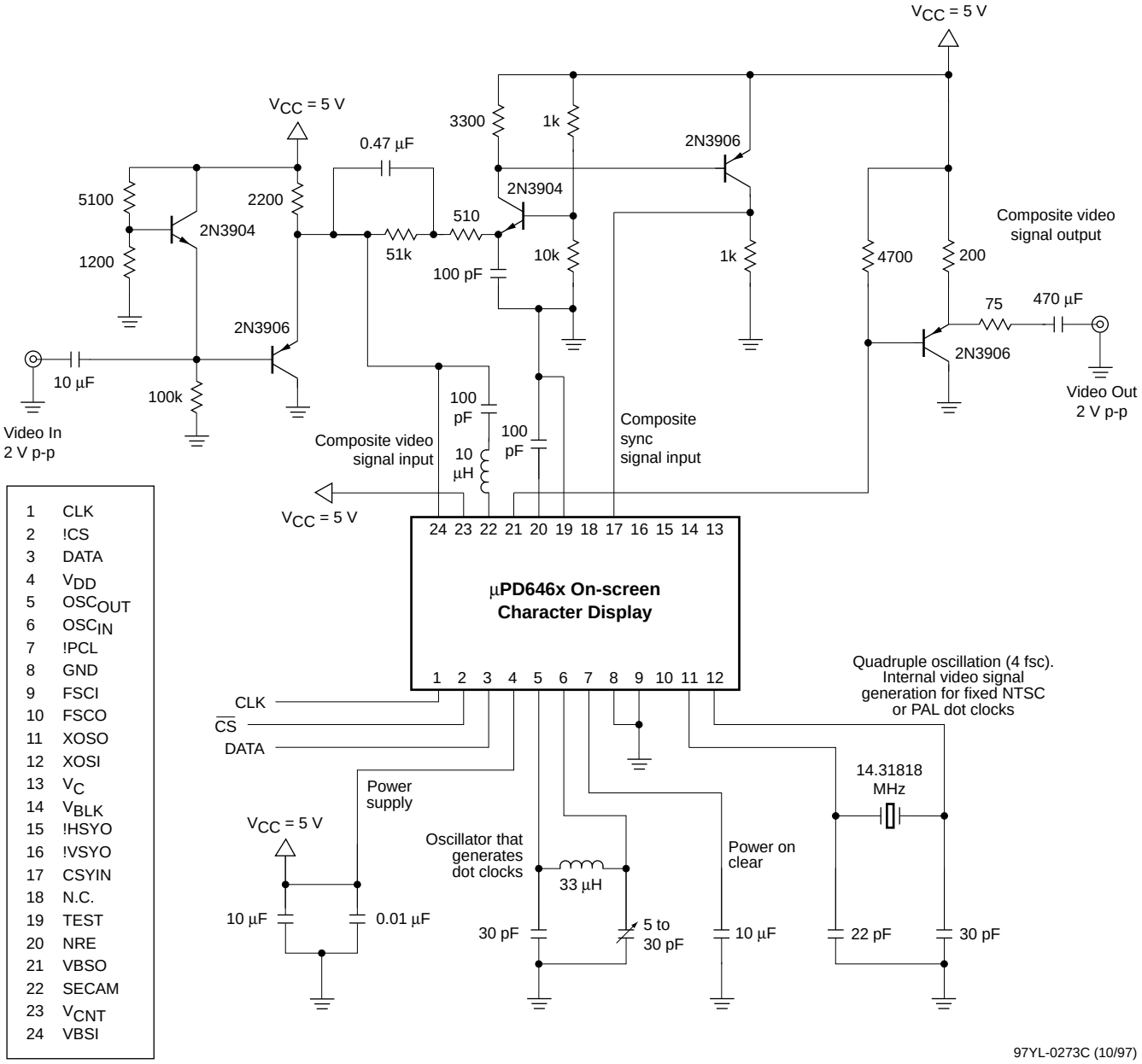
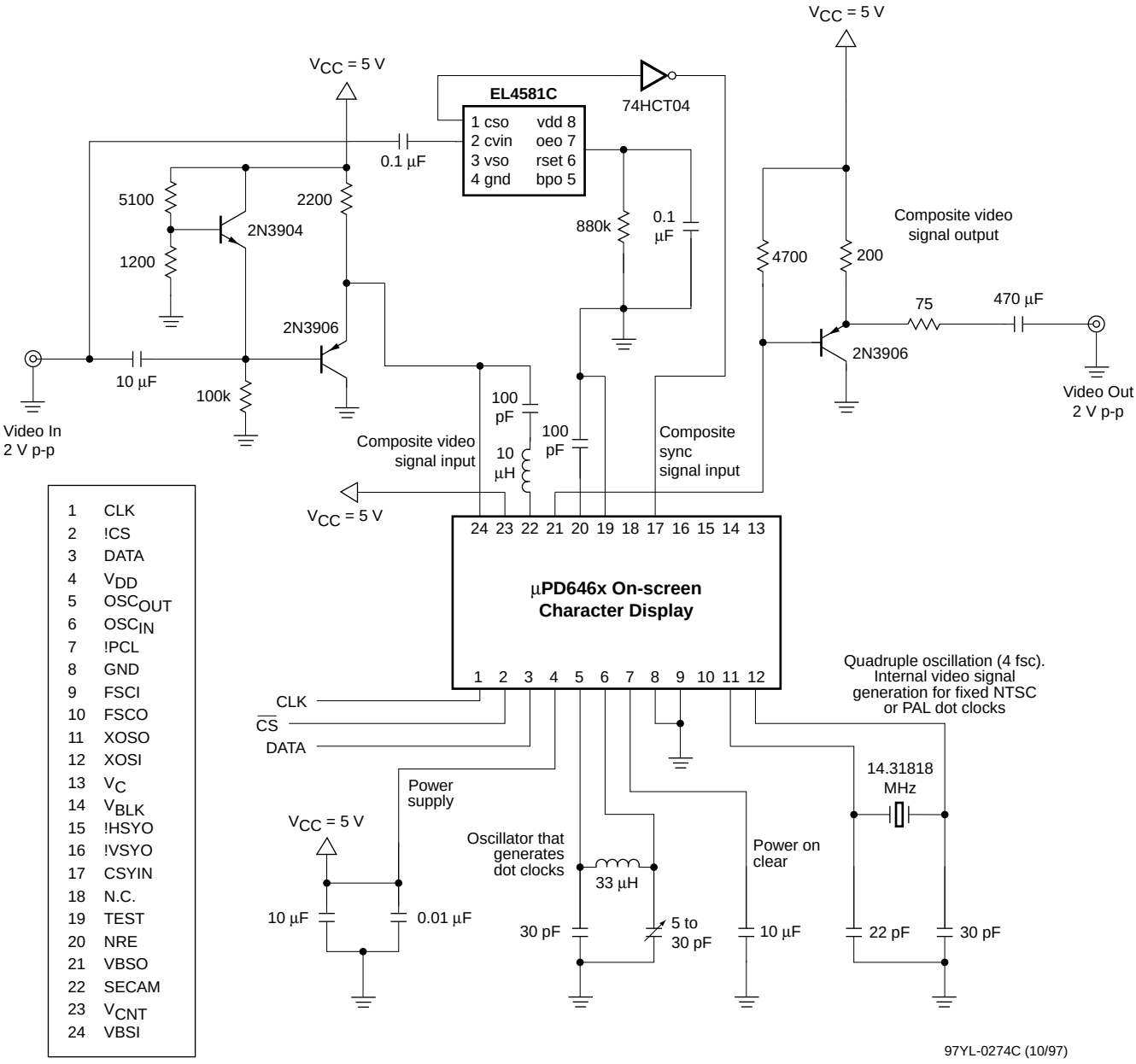


Figure 3. µPD646x with Elantec EL4581C Sync Separator Chip



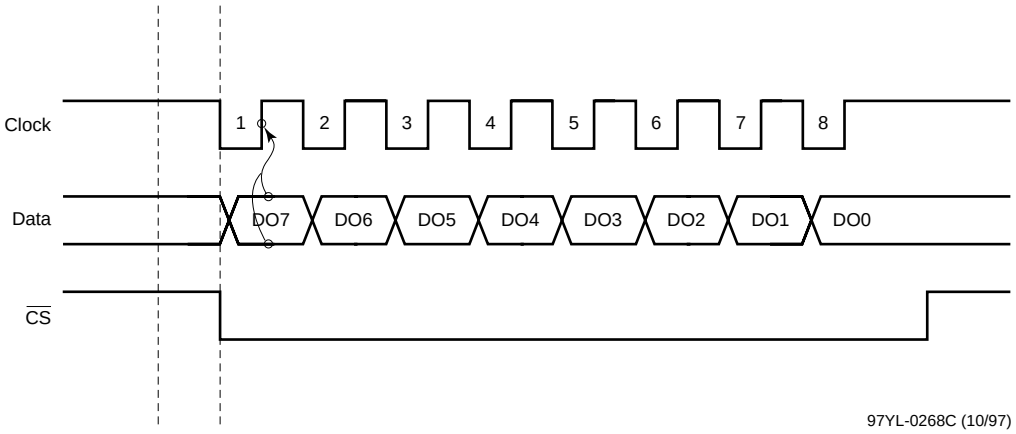
Programming the µPD646x

The µPD646x is programmed with commands sent serially from a PC or microcontroller to the serial data input. Commands must be transmitted with the most significant bit first. The command bit stream is not RS-232-compatible because it requires a companion clock. Serial transfer is begun by making the chip select input low. The data presented to the µPD646x through the serial data input is shifted into its buffer each time the CLK input transitions from a low to a high level (see Figure 4). After eight clocks, the CLK input must go high and remain there for the minimum high-level width of 400 ns. Afterward, the clock pattern can start again to input another command or character.

Transferring a two-byte command is similar to a one-byte command except that when

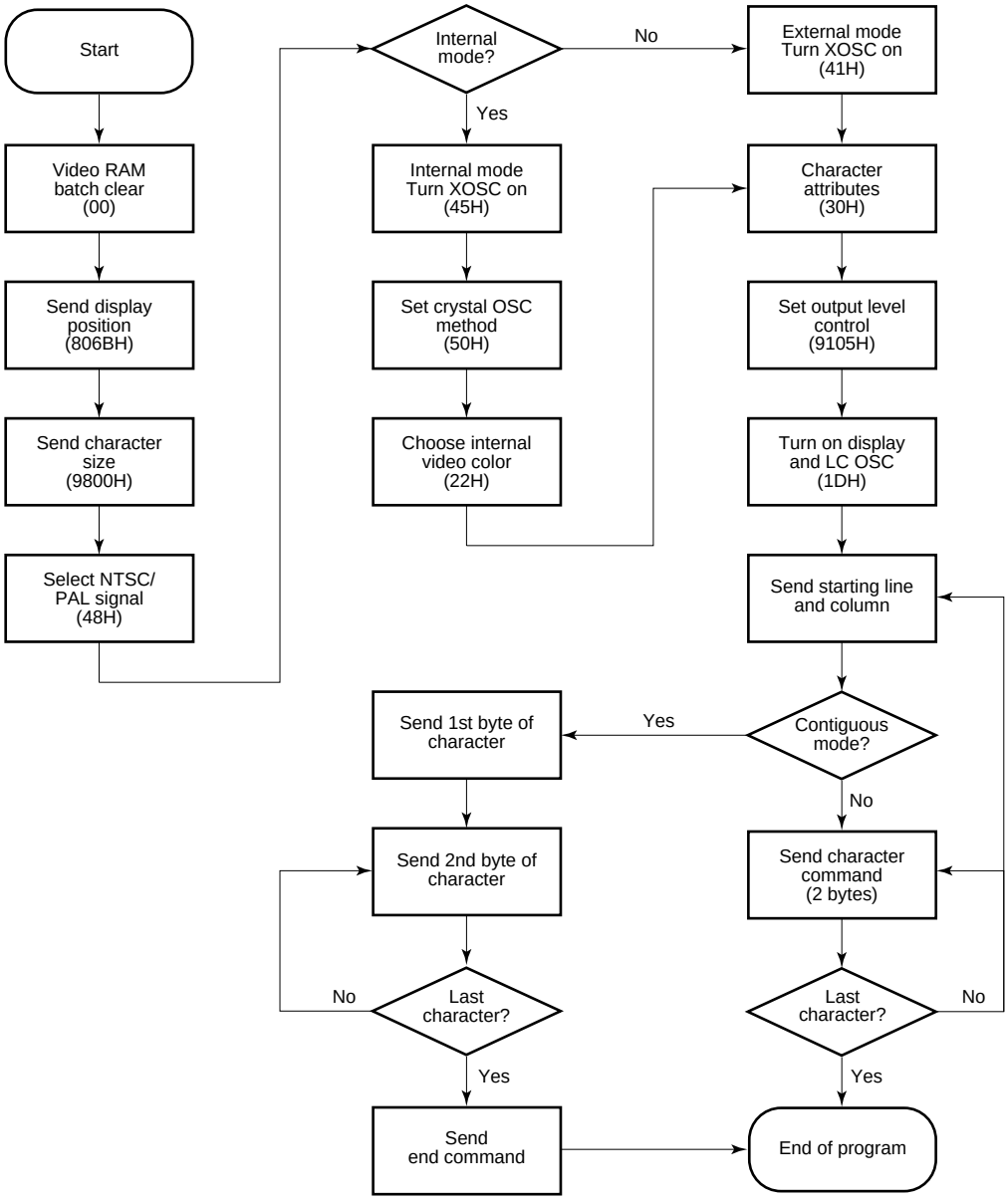
transferring a two-byte command, you must keep the chip select input ($\overline{CS}$) low during the transmission of both bytes.

Figure 4. µPD646x One-Byte Command (MSB-first)



97YL-0268C (10/97)

Figure 5. µPD646x Programming Flow Chart



97YL-0269C (10/97)

Internal Mode of the µPD646x

The µPD646x can be programmed for internal mode as well as external mode. In the internal mode, characters are output in sync with the video signal internally created by the IC. When programming the µPD646x, the first instruction should be a video RAM batch clear command (see Figure 5). This command clears all the character data in the video RAM buffer.

The next command in the flow chart is the display position control command that defines the vertical and horizontal starting positions on the screen. This position is relative to the first pixel on the TV and it defines the point where the block of data in video RAM will start to be displayed on the TV or monitor. For example, start location 0,0 would locate the screen display in the upper-left corner of the TV screen. Row and

column positions greater than 0,0 will move the displayed block of characters toward the lower-right side of the screen.

The character size, a two-byte command, is the next command to be sent serially. Select the video signal, NTSCor PAL, depending on the video protocol of your country. Select internal mode and turn ON the XOSC oscillation. Set the OSC crystal oscillation to quadruple oscillation, and choose the color of the internal video generated by the µPD646x.

Define the character attributes and the output level that controls the luminance of both the character and background. Using the display control command, turn ON both the display and the LC oscillation and, if you want blinking characters, choose the blinking frequency with this command.

Use the write address control command to send the starting line and column addresses that define where the character data will begin display within the 12 line by 24 column display area. This command differs from the previous display position command that defined the starting vertical and horizontal positions on the screen. If a new line and column address is not sent with each character, the µPD646x will automatically write new characters into sequential locations in the video buffer. For a sample internal mode program, please see the Appendix.

External Mode of the µPD646x

In external mode, character data is overlaid on a video signal created from an external source such as a VCR. When programming the µPD646x, it is recommended that the user follow the same four beginning steps as in the internal mode. (See Figure 5.) Afterward, set the video signal to external mode and turn on the crystal oscillation.

Next, define the character attributes and the output level that controls the luminance of both the character and background. Using the display control command, turn ON both the display and the LC oscillation and if you want blinking characters, choose the blinking frequency with this command.

Use the write address control command to send the starting line and column addresses that define where the character data will begin to display within the 12-line by 24-column display area. If a new line and column address is not sent with each character, the µPD646x will automatically write new characters into sequential locations in the video buffer. For a sample external mode program, please see the Appendix.

Contiguous Command of the µPD646x

The µPD646x is programmable to display characters on the screen using the contiguous command instead of a single two-byte transfer command. The contiguous command is a successive command that is used when characters are to be displayed sequentially on the display area of the screen using the same character attributes. If the characters are not in succession or if the character attributes change, then the two-byte display character control command must be used instead of the contiguous command.

The display character control command is a two-byte command that specifies the character data and character attributes to be written to the video RAM. If character data is sequentially written without a change in character attributes, the second character and those that follow can be abbreviated to the lower 8 bits (D7 to D0). Then only the lower byte needs to be transferred to display the character. In this case, the write column address is automatically incremented.

To use the contiguous command, transfer the first byte (the display character control command) once and then transfer data bytes of the subsequent characters to be displayed. After the last byte is transferred, send an end code command or terminate

the two-byte contiguous command by making the chip select ($\overline{CS}$) go high. Although only one of these two is necessary to terminate the contiguous command, both are recommended as a countermeasure against noise. Observe that the µPD6464 and µPD6465 have different end code commands. For a sample program using the contiguous command, please see the Appendix.

FAQs

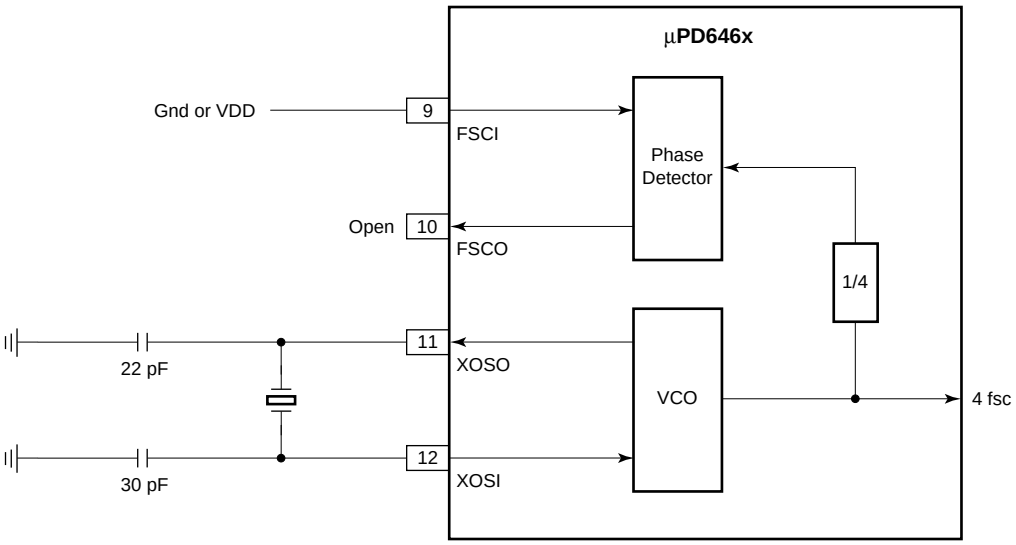
Q Is the C-sync signal a combination of the horizontal and vertical sync pulses?

A. Yes. The µPD646x has a sync separator that gives Hsync and Vsync out of the C-sync input.

Q. What is the fsc signal input?

A. Through an on-chip x4 multiplier, the fsc signal generates the 4fsc signal used in the sync signal separation circuit for external mode and in the sync signal generator for internal video signals. The x4 multiplier eliminates the need for crystal resonator, and in turn reduces the mounting area and the total cost. However, if you don't use the multiplier, you can use the crystal resonator set on quadruple oscillation in the crystal select control command. The crystal resonator application circuit. (Figure 6) can generate a fixed NTSCor PAL dot clock (4fsc) without the fsc input data pin 9.

Figure 6. Crystal Resonator Circuit

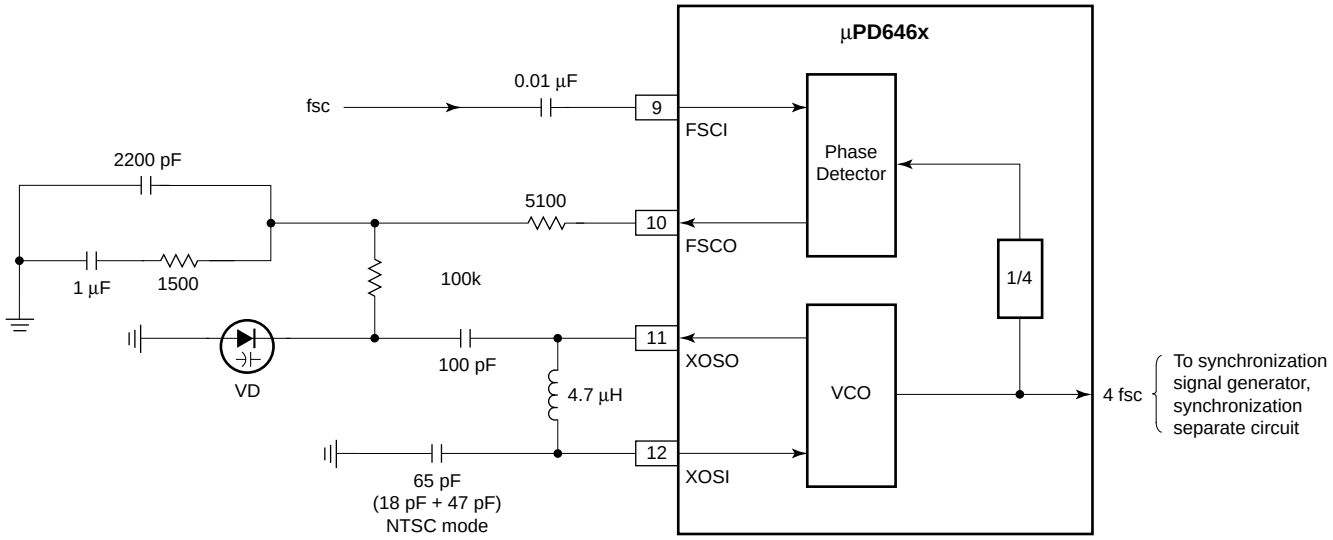


97YL-0270C (10/97)

Q. What configuration should be used if fsc is input from an external source?

A. The fsc is required to sync up the LC oscillator (VCO) phase (See Figure 7). It connects 4fsc from the VCO through a phase detector. Varactor diode VD is used for phase-locking between fsc, the color carrier signal, and 4fsc generated by the VCO. In NTSC mode, fsc is 3.579545 MHz and 300 mVp-p. The 100k-ohm resistor is used to control the capacitance of VD. If the device is used for NTSC application only, you can combine the two capacitors on the XOSI input for a total of65 pF.

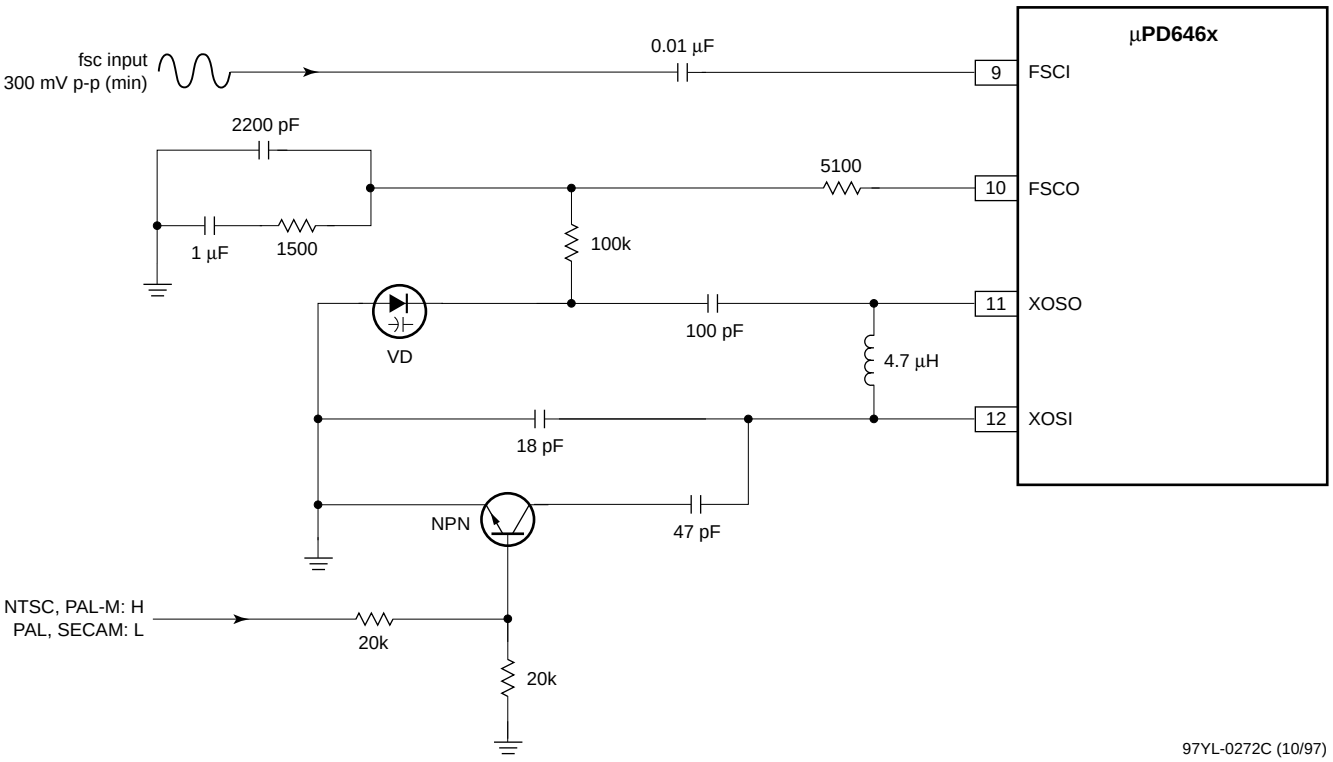
Figure 7. Phase-Locking 4fsc to fsc



97YL-0271C (10/97)

With a transistor to switch ON/OFF the 47-pF capacitor on XOSI, then the circuit shown in Figure 8 can be used to switch between NTSC and PAL with fsc for sync. (This circuit is the same as shown in the application circuit diagram of the μPD646x data sheet, document no. IC-3567.)

Figure 8. Switching Between NTSC and PAL



97YL-0272C (10/97)

Q. Why does NEC claim that µPD646x has a built-in sync separator if the design engineer must strip off the C-sync signal with an external circuit?

A. The internal sync separator gives you Hsync and Vsync out of C-sync. The design engineer still needs to use an external sync separation circuit as shown on the top middle portion of the schematic in Figure 2 of this application note to get C-sync from the composite input.

Q. If you do not use SECAM, how do you connect the SECAM pin?

A. SECAM method can be selected using the video signal method control command. If you are not using SECAM, you can connect the pin to V_{DD}, GND, or leave it open.

APPENDIX

This Appendix contains three program examples written in assembly language to demonstrate the µPD646x software commands. In the setup, the µPD646x is controlled by a 78K4 microcontroller evaluation board (DDB-K4026) connected to a PC via the parallel port.

Figure A-1. This internal mode example fills the screen with the letters NEC and then displays blinking NECs in the corners and the middle of the screen.

Figure A-2. This external mode example executes a scrolling effect at the bottom of the screen while displaying NEC CORP.

Figure A-3. This external mode example shows how to use the contiguous commands of the µPD646x.

Figure A-1. Internal 64.ASM Program

```
$TITLE('***INTERN64.ASM****')
$ PROCESSOR (4026) ;device type (REQUIRED)
;
;********** INTERNAL MODE OF THE µPD6464 ************
;
;The program shown below is an example written in assembly language for  the µPD6464, On-Screen
;Display. In this example, the µPD6464 is controlled by the 78K4 microcontroller evaluation board
;(DDB-K4026). The 78K4 evaluation board was connected to a PC via the parallel port. This program is a
;demo example that shows how to use the software commands of the µPD6464. The internal mode
;example, fills the display screen with the letters NEC and then displays blinking NEC in the corners
;and in the middle of the display screen.

*** DEFINITION OF VARIABLES ***
CHPSEL          EQU     P3.1        ;Chip Select
CLOCK           EQU     P3.2        ;Clock
DATA            EQU     0           ;Input Data OSD
RAMCLR          EQU     P3.3        ;RAM batch clear
POSCTRL         EQU     806BH       ;9Hx3 of Vsync,12x13 of Hsync
CHRSIZE         EQU     9800H       ;Character size
INT_ON          EQU     45H         ;Internal video, Oscillation On
NTSC            EQU     48H         ;NTSC video signal
FSC             EQU     50H         ;4fsc crystal oscillation
SIGNCOLOR       EQU     22H         ;Blue internal video signal color
CHAR_ATTRIB     EQU     30H         ;No Background on the characters
OUTLEVEL        EQU     9105H       ;2 Vpp amplitude,75 IRE level control
DISP_CTRL       EQU     1DH         ;2Hz Blinking freq./LC and Display On
WRITEADD        EQU     8800H       ;Starting line 0 and column 0
LETTR_N         EQU     0C01EH      ;Letter N
LETTR_E         EQU     0C015H      ;Letter E
LETTR_C         EQU     0C013H      ;Letter C
SPACE           EQU     0C010H

;***************************************
;* MAIN ROUTINE  *
;***************************************

;The main routine does the following functions:
;Initializes port 3 as an output port.
;Calls a subroutine to initialize the On-Screen Display.
;Sets line 0 and column 0 where the first NEC is to be displayed.
;Fills up the screen with the letters NEC plus a space and
;Calls a subroutine to display several blinking NECs.
```

Figure A-1. Internal 64.ASM Program (Continued)

```
           ORG      0000H
RSTVCT: DW          START          ;Reset program start address
           ORG      080H           ;Code segment starts at address 80
START:     LOCATION 15
;          ----- Sets port 3 as output port only -----
           MOV      PM3,#0         ;all output pins in port 3
           MOV      PMC3,#0        ;all pins in input/output mode
           CALL     INI_6464       ;Subroutine to initialize OSD
;          ----- Displays the first NEC at position 0,0 -----
           MOVW     AX,#WRITEADD   ;Starting line 0 and column 0
           CALL     SIXTN          ;Executes sixteen bit commands
           CALL     DISPLAY        ;Subroutine to display characters
;          ----- Fills up the rest of the screen with NEC -----
           MOV      B,#49H         ;Number 49 counter
LOOP0:
           CALL     DISPLAY        ;displays the character
           DBNZ     B,$LOOP0       ;Decrement and Branch if not zero
;          ; ----- Calls a subroutine that displays several blinking "NEC" -----
           CALL     BLINKCHAR      ;Subroutine for blinking characters
;          ----- Maintains the processor idle indefinitely -----
LOOP:      BR       $LOOP          ;continue for ever
;****************************************
;* INITIALIZATION OF UPD6464 OSD *
;****************************************
;This subroutine initializes the internal mode of the UPD6464. The internal video signal created by the
;OSD is a blue screen. Some of the "NEC" to be displayed will be blinking, therefore a blinking
;frequency needs to be chosen using the display control command. The commands are both 8-bit and
;16-bit commands.
INI_6464:
           MOV      A,#RAMCLR      ;Video RAM batch clear
           CALL     EIGHT          ;Executes eight bit commands
           MOVW     AX,#POSCTRL    ;Position control command
           CALL     SIXTN          ;Executes sixteen bit commands
           MOVW     AX,#CHRSIZE    ;Character size control command
           CALL     SIXTN          ;Executes sixteen bit commands
           MOV      A,#INT_ON      ;Internal mode control/Oscillation On
           CALL     EIGHT          ;Executes eight bit commands
           MOV      A,#NTSC        ;NTSC video signal method
           CALL     EIGHT          ;Executes eight bit commands
           MOV      A,#FSC         ;4fsc Crystal oscillation method
           CALL     EIGHT          ;Executes eight bit commands
           MOV      A,#SIGNCOLOR   ;Blue internal video signal color
           CALL     EIGHT          ;Executes eight bit commands
           MOV      A,#CHAR_ATTRIB ;No background on the characters
           CALL     EIGHT          ;Executes eight bit commands
           MOVW     AX,#OUTLEVEL   ;2Vpp amplitude, 75 IRE output level
           CALL     SIXTN          ;Executes sixteen bit commands
           MOV      A,#DISP_CTRL   ;Turn on Display and LC
           CALL     EIGHT          ;Executes eight bit commands
RET
;****************************************
;* SUBROUTINE TO DISPLAY 'NEC' *
;****************************************
;This subroutine displays the Word "NEC" plus a space one time on the screen. For each character to be
;displayed, the subroutine calls another subroutine to transmit the 16-bit character to the UPD6464.
;After the space is transmitted, it then returns to its subroutine call.
```

Figure A-1. Internal 64.ASM Program (Continued)

```
DISPLAY
          MOVW     AX,#LETTR_N    ;Loads the letter N into register A
          CALL     SIXTN          ;Executes sixteen bit commands
          MOVW     AX,#LETTR_E    ;Loads the letter E into register A
          CALL     SIXTN          ;Executes sixteen bit commands
          MOVW     AX,#LETTR_C    ;Loads the letter C into register A
          CALL     SIXTN          ;Executes sixteen bit commands
          MOVW     AX,#SPACE      ;Loads the space bar into register A
****************************************
;* SUBROUTINE TO BLINK CHARACTERS *
;****************************************
;This subroutine displays a blinking "NEC" at each corner of the 12 lines by 24 column display area. It
;also displays a blinking "NEC" at the four middle positions of the display area. Each time a new
;position is entered into the registers of the microprocessor, the subroutine calls another subroutine
;that displays the blinking characters. After the last blinking "NEC" is displayed, the program goes
;into an idle routine.
BLINKCHAR:
          MOVW     AX,#8800H      ;Starting line 0 and column 0
          CALL     BLINK          ;Displays upper left corner of screen
          MOVW     AX,#8814H      ;Starting line 0 and column 21
          CALL     BLINK          ;Displays upper right corner of screen
          MOVW     AX,#88A8H      ;Starting line 5 and column 8
          CALL     BLINK          ;Displays upper left of the four middle
          MOVW     AX,#88ACH      ;Starting line 6 and column 8
          CALL     BLINK          ;Displays upper right of the four middle
          MOVW     AX,#88CCH      ;Starting line 5 and column 12
          CALL     BLINK          ;Displays lower left of the four middle
          MOVW     AX,#8960H      ;Starting line 6 and column 12
          CALL     BLINK          ;Displays lower right of the four middle
          MOVW     AX,#8974H      ;Starting line 11 and column 0
          CALL     BLINK          ;Displays lower left corner of screen
          MOVW                    ;Starting line 11 and column 21
          CALL     BLINK          ;Displays lower right corner of screen
RET
;****************************************
;* ROUTINE TO DISPLAY BLINKING CHARACTERS *
;****************************************
;This subroutine displays blinking characters on the screen. For each character to be displayed, it
;calls a subroutine that transmits 16-bit commands to the UPD6464. After it displays the blinking "NEC"
;at the position specified by the calling subroutine, it goes back to its subroutine call.
BLINK
          CALL     SIXTN          ;Executes sixteen bit commands
          MOVW     AX,#0C21EH     ;Loads the blinking letter N
          CALL     SIXTN
          MOVW     AX,#0C215H     ;Loads the blinking letter E
          CALL     SIXTN
          MOVW     AX,#0C213H     ;Loads the blinking letter C
          CALL     SIXTN
RET
;****************************************
;* SUBROUTINE FOR AN 8 BIT COMMAND *
;****************************************
;This subroutine transmits an 8-bit command serially.
;It uses output pins from Port 3 as follow: pin1=clock, pin2=chip select and pin3=data line.
;This subroutine executes 8 times before returning to its subroutine
;call. Each time the loop executes, it sends one bit of the 8-bit command used to initialize the
;µPD6464.
```

Figure A-1. Internal 64.ASM Program (Continued)

```
EIGHT:
          CLR1     CHPSEL         ;activate chip select
          MOV      C,#08          ;number 8 counter
LOOP1:
          CLR1     CLOCK          ;clear the clock
          ROL      A,1            ;rotate left wrap around
          BF       A.0,$LOW_OUT   ;if bit = 0 branch to output LOW signal
          SET1     DATA           ;output HIGH signal
          BR       NEXT
LOW_OUT:
          CLR1     DATA           ;output LOW signal
NEXT:
          SET1     CLOCK          ;latch in data
          DBNZ     C,$LOOP1       ;Repeat eight times
RSET:
          SET1     CHPSEL         ;reset chip select to HI
          SET1     CLOCK          ;reset clock to HISET1DATA
          SET1                    ;reset data to HI
;****************************************
;* SUBROUTINE FOR A 16 BIT COMMAND *
;****************************************
;This subroutine transmits a 16 bit command serially.
;It uses output pins from Port 3 as follow: pin1=clock, pin2=chip select and pin3=data line. This
;subroutine executes 16 times before returning to its subroutine call. Each time the loop executes, it
;sends one bit of the 16 bit command or character used by the µPD6464.
SIXTN:
          CLR1     CHPSEL         ;activate chip select
          MOV      C,#0010H       ;number 16 counter
LOOP2:
          CLR1     CLOCK          ;clear the clock
          SHLW     AX,1           ;rotate left to carry
          BNC      $LOW_OUT2      ;if bit = 0 branch to output LOW signal
          SET1     DATA           ;output HIGH signal
          BR       NEXT2
LOW_OUT2:
          CLR1     DATA           ;output LO signal
NEXT2:
          SET1     CLOCK          ;latch in data
          DBNZ     C,$LOOP2       ;Repeat Sixteen times
          BR       RSET
END
```

Figure A-2. External 64.ASM Program

```
$TITLE('***EXTERN64.ASM****')
$ PROCESSOR (4026) ;device type (REQUIRED)
;
;********** EXTERNAL MODE OF THE UPD6464 ************
;
;The program below is an example written in assembly language for the µPD6464, On-Screen Display.
;In this example, the µPD6464 is controlled by the 78K4 microcontroller evaluation board
;(DDB-K4026). The 78K4 evaluation board was connected to a PC via the parallel port. This external
;mode example shows how to use software commands and the µPD6464 to execute a scrolling effect at
;the bottom of the screen when displaying "NEC CORP."
;-----------------------------------
;*** DEFINITION OF VARIABLES ***
CHPSEL        EQU      P3.1        ;Chip select
CLOCK         EQU      P3.2        ;Clock
DATAINPUT     EQU      P3.3        ;Data input for On-Screen-Display
RAMCLR        EQU      0           ;RAM batch clear
POSCTRL       EQU      806BH       ;9Hx3 of Vsync,12x13 of Hsync
CHRSIZE       EQU      9800H       ;Character size
EXT_ON        EQU      41H         ;External video, Oscillation ON
CHAR_ATTRIB   EQU      32H         ;Characters black framing background
OUTLEVEL      EQU      9105H       ;2 Vpp amplitude,75 IRE level control
DISP_CTRL     EQU      1DH         ;2Hz Blinking freq./LC and Display ON
;-----------------------------------
;
;The ADD_STRING data table stored at memory address FD20, contains the addresses for the columns
;in line 11 of the display area. These values will be read by the processor to determine where in the
;display area the next character is to be displayed.
DATA          DSEG     AT          0FD20H
ADD_STRING:            DB          40H,41H,42H,43H,44H,45H,46H,47H
                       DB          48H,49H,4AH,4BH,4CH,4DH,4EH,4FH,50H
                       DB          51H,52H,53H,54H,55H,56H,57H,58H
;****************************************
;* MAIN ROUTINE  *
;****************************************
;The main routine does the following functions:
;Initializes port 3 as an output port.
;Calls a subroutine to initialize the On-Screen Display.
;Points to the first address value stored in memory address FD20, and Calls a subroutine that executes
;the scrolling effect indefinitely.
          ORG      ‘0000H
RSTVCT:   DW       START          ;Reset program start address
          CSEG
          ORG      080H           ;Code segment starts at address 80
START:    LOCATION 15
;****************************************
;* INITIALIZATION OF µPD6464 OSD *
;****************************************
;This subroutine initializes the external mode of the µPD6464.
;The commands are 8-bit and 16-bit commands.
```

Figure A-2. External 64.ASM Program (Continued)

```
INI_6464
          MOV      A,#RAMCLR      ;Video RAM batch clear
          CALL     EIGHT          ;Executes eight bit commands
          MOVW     AX,#POSCTRL    ;Position control command
          CALL     SIXTN          ;Executes sixteen bit commands
          MOVW     AX,#CHRSIZE    ;Character size control command
          CALL     SIXTN          ;Executes sixteen bit commands
          MOV      A,#EXT_ON      ;Internal mode control/Oscillation On
          CALL     EIGHT          ;Executes eight bit commands
          MOV      A,#CHAR_ATTRIB ;Black Framing character attributes
          CALL     EIGHT          ;Executes eight bit commands
          MOVW     AX,#OUTLEVEL   ;2Vpp amplitude, 75 IRE output level
          CALL     SIXTN          ;Executes sixteen bit commands
          MOV      A,#DISP_CTRL   ;Turn on Display and LC
          CALL     EIGHT          ;Executes eight bit commands
RET
;****************************************
;* SUBROUTINE TO HANDLE SCROLLING EFFECT *
;****************************************
;This routine executes 24 times for the 24 columns of the display area.
;Each "NEC CORP." message, will be printed starting at every consecutive column. However the current
;letter "N" will be erased before repeating the message starting at the next consecutive column. This
;procedure mimics the scrolling effect. When the characters disappear at the end of line 11, the
;following character in the message is displayed at the beginning of line 11 obtaining a wrap around
;effect.
SCROLL_ADD:
          MOV      B,#18H         ;Execute 24-times Counter
AGAIN:
          CMP      B,#08H         ;The first 16 columns displayed?
          BGT      $SCROLL        ;If not, perform normal scrolling
          CALL     DISP_CONT      ;if yes, do special display procedure
          BR       BEGIN          ;skip regular display procedure
;         ----- Normal display procedure of "NEC CORP." -----
SCROLL:
          MOV      A,#89H         ;Get upper byte of display address
          XCH      A,X            ;Save current value of register A
          MOV      A,[HL]         ;Get lower byte of display address
          XCH      A,X            ;Restore the value of register A
          CALL     SIXTN          ;Transmit the display address
          CALL     DISPLAY        ;Regular display procedure
BEGIN:
          CALL     STALL          ;Creates a delay for the scrolling effect
          CALL     STALL
          CALL     STALL
          MOV       A,#89H        ;Get upper byte of display address
          XCH      A,X            ;Save current value of register A
          MOV       A,[HL+]       ;in memory and point to the next value.
          XCH      A,X            ;Restore the value of register A
          CALL     SIXTN          ;Transmit the display address
          MOVW     AX,#0C010H     ;Display a blank character
          CALL     SIXTN
          DBNZ     B,$AGAIN       ;Repeat 24 times
          MOVW     HL,#ADD_STRING  ;Point to the first address in the table
          RET
;****************************************
;* SUBROUTINE TO DISPLAY 'NEC CORP.' *
;****************************************
;This subroutine displays the message "NEC CORP." on the display area. For each character to be
;displayed, the subroutine calls another subroutine to transmit the 16-bit character to the µPD6464.
```

Figure A-2. External 64.ASM Program (Continued)

```
DISPLAY:   MOVW     AX,#0C01EH     ;Loads the letter N
           CALL     SIXTN          ;Executes sixteen bit commands
           MOVW     AX,#0C015H     ;Loads the letter E
           CALL     SIXTN          ;Executes sixteen bit commands
           MOVW     AX,#0C013H     ;Loads the letter C
           CALL     SIXTN          ;Executes sixteen bit commands
           MOVW     AX,#0C010H     ;Loads the space bar
           CALL     SIXTN
           MOVW     AX,#0C013H     ;Loads the letter C
           CALL     SIXTN
           MOVW     AX,#0C000H     ;Loads the letter O
           CALL     SIXTN
           MOVW     AX,#0C022H     ;Loads the letter R
           CALL     SIXTN
           MOVW     AX,#0C020H     ;Loads the letter P
           CALL     SIXTN
           MOVW     AX,#0C00EH     ;Loads the period
           CALL     SIXTN
RET
;****************************************
;* SUBROUTINE TO STALL THE PROCESSOR *
;****************************************
;This subroutine creates a delay for the microprocessor. It uses registers B and C to repeat this
;subroutine 256 times.
STALL:
           PUSH     BC             ;Save the current values
RLOOP2:    MOV      B,#0FFH        ;Number-16 counter
RLOOP3:    MOV      C,#0FFH        ;Number-16 counter
RLOOP4:    NOP                     ;Do nothing
           DBNZ     C,$RLOOP4      ;Repeat 16 times for each of next 16
           DBNZ     B,$RLOOP3      ;For a total of 256 repeats
           POP      BC             ;Restore the values
           RET
;****************************************
;* SPECIAL DISPLAY SUBROUTINE *
;****************************************
;This subroutine performs a special display for the last message of "NEC CORP." in line 11. It requires
;special handling because each of the following characters past column 24 needs to be relocated to the
;beginning of line 11 in order to create a wrap around effect.
DISP_CONT:
           MOVW     AX,#0C01EH     ;Load the letter N
           CALL     SIXTN          ;Executes sixteen bit commands
           CMP      B,#01H         ;Is it column 24?
           BE       $WRAP          ;If so, execute wrap around routine
           MOVW     AX,#0C015H     ;Otherwise Load the letter E
           CALL     SIXTN          ;Executes sixteen bit commands
           CMP      B,#02H         ;Is it column 23?
           BE       $WRAP          ;If so, execute wrap around routine
           MOVW     AX,#0C013H     ;Otherwise load the letter C
           CALL     SIXTN          ;Executes sixteen bit commands
           CMP      B,#03H         ;Is it column 22?
           BE       $WRAP          ;If so, execute wrap around routine
           MOVW     AX,#0C010H     ;Otherwise load the space bar
           CALL     SIXTN
           CMP      B,#04H         ;Is it column 21?
           BE       $WRAP          ;If so, execute wrap around routine
           MOVW     AX,#0C013H     ;Otherwise load the letter C
           CALL     SIXTN
           CMP      B,#05H         ;Is it column 20?
```

Figure A-2. External 64.ASM Program (Continued)

```
            BE      $WRAP           ;If so, execute wrap around routine
            MOVW    AX,#0C000H      ;Otherwise load the letter O
            CALL    SIXTN
            BE      $WRAP           ;If so, execute wrap around routine
            CMP     B,#06H          ;Is it column 19?
            MOVW    AX,#0C022H      ;Otherwise load the letter R
            CALL    SIXTN
            CMP     B,#07H          ;Is it column 18?
            BE      $WRAP           ;If so, execute wrap around routine
            MOVW    AX,#0C020H      ;Otherwise load the letter P
            CALL    SIXTN
            CMP     B,#08H          ;Is it column 18?
            BE      $WRAP           ;If so, execute wrap around routine
            MOVW    AX,#0C00EH      ;Otherwise load the period
            CALL    SIXTN
WRAP:
            MOVW     AX,#8940H      ;Position next command at column 0
            CALL    SIXTN
            CMP     B,#01H          ;Is it column 17?
            BE      $UNO            ;if so display "EC CORP."
            CMP     B,#02H          ;Is it column 18?
            BE      $DOS            ;if so display "C CORP."
            CMP     B,#03H          ;Is it column 19?
            BE      $TRES           ;if so display "CORP."
            CMP     B,#04H          ;Is it column 20?
            BE      $QUATRO         ;if so display "CORP."
            CMP     B,#05H          ;Is it column 21?
            BE      $CINCO          ;if so display "ORP."
            CMP     B,#06H          ;Is it column 22?
            BE      $SEIS           ;if so display "RP."
            CMP     B,#07H          ;Is it column 23?
            BE      $SIETE          ;if so display "P."
            CMP     B,#08H          ;Is it column 24?
            BE      $OCHO           ;if so display "."
                                    ;Otherwise you would not even be here!
UNO:        MOVW    AX,#0C015H      ;Loads the letter E
            CALL    SIXTN           ;Executes sixteen bit commands
DOS:        MOVW    AX,#0C013H      ;Loads the letter C
                    CALL  SIXTN     ;Executes sixteen bit commands
TRES:       MOVW    AX,#0C010H      ;Loads the space bar
            CALL    SIXTN
QUATRO:     MOVW    AX,#0C013H      ;Loads the letter C
            CALL    SIXTN
CINCO:      MOVW    AX,#0C000H      ;Loads the letter O
            CALL    SIXTN
SEIS:       MOVW    AX,#0C022H      ;Loads the letter R
            CALL    SIXTN
SIETE:      MOVW    AX,#0C020H      ;Loads the letter P
            CALL    SIXTN
OCHO:       MOVW    AX,#0C00EH      ;Loads the period
            CALL    SIXTN
            BR      BLANK           ;Displays a blank character over the "N"
            RET
;****************************************
;* SUBROUTINE TO ERASE LETTER "N" *
;****************************************
;This subroutine displays a blank character over the Letter "N" in the message "NEC CORP.". It is as if
;the Letter N was erased. This subroutine is executed before the "NEC CORP." message is displayed
;again.
```

Figure A-2. External 64.ASM Program (Continued)

```
BLANK:     MOV       A,#89H        ;Load the upper byte of display address
           XCH      A,X            ;Save current value of register A
           MOV      A,[HL]         ;Load the lower byte of display address
           XCH      A,X            ;Format the address for transmitting
           CALL     SIXTN          ;Transmit the address to the µPD6464
           MOVW     AX,#0C010H     ;Load a blank character
           CALL     SIXTN          ;Transmit the blank character
           RET
;****************************************
;* SUBROUTINE FOR AN 8 BIT COMMAND *
;****************************************
;This subroutine transmits an 8-bit command serially.
;It uses output pins from Port 3 as follow: Pin1=Clock, Pin2=Chip Select and Pin3=Data Input. This
;subroutine executes 8 times before going back to its last place of origin. Each time the loop executes,
;it sends one bit of the 8-bit command used to initialize the µPD6464.
EIGHT:
           CLR1     CHPSEL         ;activate chip select
           MOV      C,#08          ;number 8 counter
LOOP1:
           CLR1     CLOCK          ;clear the clock
           ROL      A,1            ;rotate left wrap around
           BF       A.0,$LOW_OUT   ;if bit = 0 branch to output a LOW signal
           SET1     DATAINPUT      ;Otherwise output a HIGH signal
           BR       NEXT
LOW_OUT:
           CLR1     DATAINPUT      ;output LOW signal
NEXT:      SET1     CLOCK          ;latch in data
           DBNZ     C,$LOOP1       ;Repeat 8 times
RSET:
           SET1     CHPSEL         ;reset chip select to HI
           SET1     CLOCK          ;reset clock to HI
           SET1     DATAINPUT      ;reset data to HI
           RET
;****************************************
;* SUBROUTINE FOR A 16 BIT COMMAND *
;****************************************
;This subroutine transmits a 16 bit command serially.
;It uses output pins from Port 3 as follow: pin1=Clock, pin2=Chip Select and pin3=Data Input. This
;subroutine executes 16 times before going back to its last place of origin. Each time the loop
;executes,it sends one bit of the 16 bit command or character used by the µPD6464.
SIXTN:
           CLR1     CHPSEL         ;activate chip select
           MOV      C,#0010H       ;number 16 counter
LOOP3:
           CLR1     CLOCK          ;clear the clock
           SHLW     AX,1           ;rotate left to carry
           BNC      $LOW_OUT3      ;if bit = 0 branch to output a LOW signal
           SET1     DATAINPUT      ;output a HIGH signal
           BR       NEXT3
LOW_OUT3:
           CLR1     DATAINPUT      ;output a LOW signal
NEXT3:     SET1     CLOCK          ;latch in data
           DBNZ     C,$LOOP3       ;Repeat 16 times
           BR       RSET
END
```

Figure A-3. Contiguous 64.ASM Program

```
$TITLE('***CONTIG64.ASM****')
$ PROCESSOR (4026) ;device type (REQUIRED)
;
;********** CONTIGUOUS COMMAND IN THE **********
; ********** EXTERNAL MODE OF THE UPD6464 **********
;
;The program shown below is an example written in assembly language for the µPD6464, On-Screen
;Display. In this example, the µPD6464 is controlled by the 78K4 microcontroller evaluation board
;(DDB-K4026). The 78K4 evaluation board was connected to a PC via the parallel port. This program is
;a demo example that shows how to use the contiguous commands of the µPD6464.
;
;---------------------------------
;*** DEFINITION OF VARIABLES ***
CHPSEL        EQU   P3.1          ;Chip Select
CLOCK         EQU   P3.2          ;Clock
DATAINPUT     EQU   P3.3          ;Data for On-Screen-Display
RAMCLR        EQU   0             ;RAM batch clear
POSCTRL       EQU   806BH         ;9Hx3 of Vsync,12x13 of Hsync
CHRSIZE       EQU   9800H         ;Character size
EXT_OFF       EQU   41H           ;External video, Oscillation ON
CHAR_ATTRIB   EQU   32H           ;Black framing character background
OUTLEVEL      EQU   9105H         ;2 Vpp amplitude,75 IRE level control
DISP_CTRL     EQU   1DH           ;2Hz Blinking freq./LC and Display On
WRITEADD      EQU   8940H         ;Starting line 11 and column 0
PERIOD        EQU   0C00EH        ;Period character

;The STRING data table stored at memory FD20, contains the message "This is a contiguous
;Command.". The period at the end of the data table is used as an end-of-data indicator. When the
;microcontroller detects the period, it stops displaying characters from the memory and stays idle

DATA          DSEG  AT 0FD20H
STRING:       DB    24H,58H,59H,63H,10H,59H,63H,10H
              DB    51H,5EH,10H,55H,68H,51H,5DH,60H,5CH
              DB    55H,10H,5FH,56H,10H,51H,10H,53H,5FH
              DB    5EH,64H,59H,57H,65H,5FH,65H,63H,10H
              DB    53H,5FH,5DH,5DH,51H,5EH,54H,0EH

;****************************************
;* MAIN ROUTINE  *
;****************************************
;The main routine does the following functions:
;Initializes port 3 as an output port.  Calls a subroutine to initialize the On-Screen Display.  Sets
;line 11 and column 0 as the first address to display and  Calls a subroutine to execute the contiguous
;commands.
```

Figure A-3. Contiguous 64.ASM Program (Continued)

```
              ORG   0000H
RSTVCT:       DW    START          ;Reset program start address
              CSEG
              ORG   080H            ;Code segment starts at address 80
START:        LOCATION 15
;             ----- Sets port 3 as output port only -----
              MOV   PM3,#0         ;all output pins in port 3
              MOV   PMC3,#0        ;all pins in input/output mode
              MOV   B,#0           ;Set B as a flag for contiguous
              CALL   INI_6464      ;Subroutine to initialize OSD
;             ----- Displays the first NEC at position 0,0 -----
              MOVW  AX,#WRITEADD   ;Starting line 0 and column 0
              CALL  SIXTN          ;Executes sixteen bit commands
              CALL  CONTIGUOUS     ;Execute contiguous commands
;             ----- Maintains the processor idle indefinitely -----
LOOP:         BR    $LOOP          ;stay here indefinitely
```

```
;****************************************
;* INITIALIZATION OF UPD6464 OSD *
;****************************************
;This subroutine initializes the external mode of the µPD6464. The commands are 8-bit and 16-bit
;commands.

INI_6464:
              MOV   A,#RAMCLR      ;Video RAM batch clear
              CALL  EIGHT          ;Executes eight bit commands
              MOVW  AX,#POSCTRL    ;Position control command
              CALL  SIXTN          ;Executes sixteen bit commands
              MOVW  AX,#CHRSIZE    ;Character size control command
              CALL  SIXTN          ;Executes sixteen bit commands
              MOV   A,#EXT_OFF     ;External mode control/Oscillation ON
              MOV   A,#CHAR_ATTRIB  ;Black Framing character attribute
              CALL  EIGHT          ;Executes eight bit commands
              CALL  EIGHT          ;Executes eight bit commands
              MOVW  AX,#OUTLEVEL   ;2Vpp amplitude, 75 IRE output level
              CALL  SIXTN          ;Executes sixteen bit commands
              MOV   A,#DISP_CTRL   ;Turn on Display and LC
              CALL  EIGHT          ;Executes eight bit commands
              RET

;****************************************
;* SUBROUTINE TO RUN A CONTIGUOUS COMMAND *
;****************************************
;This subroutine reads the characters from the memory table and displays them in a contiguous
;command. It also sets the flag to indicate the processor that a contiguous command started here. When
;the period is ;detected by the processor, no more characters are displayed.

CONTIGUOUS:
              MOV   B,#1           ;Set the flag to indicate that ;a contiguous command
                                   ;starts now
              MOVW  HL,#STRING     ;Point to the first value of the string
              MOV   A,#0C0H        ;Load the upper byte of the character
              CALL  EIGHT          ;Transmit eight bit command
AGAIN:
              XCH   A,X            ;Save the upper value of the character
              MOV   A,[HL+]        ;Load the lower byte of the character
              CALL   EIGHT
              XCH   A,X            ;Restore upper value of the character
              CMPW  AX,#PERIOD     ;Is this a period?
              BNZ   $AGAIN         ;If not, continue displaying characters Otherwise..
              MOV   B,#0
              MOV   A,#7FH         ;Load the end-code command
              CALL  EIGHT          ;Transmit the end-code command
              RET
```

Figure A-3. Contiguous 64.ASM Program (Continued)

```
;*****************************************
;* SUBROUTINE FOR AN 8 BIT COMMAND *
;*****************************************
;This subroutine transmits an 8-bit command serially. It uses output pins from Port 3 as follow:
;Pin1=Clock, Pin2=Chip Select and Pin3=Data Input. This subroutine executes 8 times before returning
;to its subroutine call. Each time the loop executes, it sends one bit of the 8-bit command used by the
;µPD6464.

EIGHT:
              CLR1  CHPSEL          ;activate chip select
              MOV   C,#08           ;number 8 counter
MLOOP1:
              CLR1  CLOCK           ;clear the clock
              ROL   A,1             ;rotate left wrap around
              BF    A.0,$LOW_OUT    ;if bit = 0 branch to output a LOW signal
              SET1  DATAINPUT       ;output a HI signal
              BR    NEXT
LOW_OUT:
              CLR1  DATAINPUT       ;output LOW signal
              NEXT: SET1  CLOCK     ;latch in data
              DBNZ  C,$LOOP1        ;repeat eight times
RSET:
              CMP   B,#1H           ;If a contiguous command then....
              BE    $SKIP_CS        ;Keep Chip Select LOW...Otherwise
              SET1  CHPSEL          ;reset chip select to HI
SKIP_CS:
              SET1  CLOCK           ;reset clock to HI
              SET1  DATAINPUT       ;reset data to HI
              RET

 ;*****************************************
;* SUBROUTINE FOR A 16 BIT COMMAND *
;*****************************************
;This subroutine transmits a 16 bit command serially. It uses output pins from Port 3 as follow:
;Pin1=Clock, Pin2=Chip Select and Pin3=data line. This subroutine executes 16 times before returning
;to its subroutine call. Each time the loop executes, it sends one bit of the 16 bit command or
;character used by On-Screen Display.

SIXTN:
              CLR1    CHPSEL        ;activate chip select
              MOV     C,#0010H      ;number 16 counter
LOOP3:
              CLR1    CLOCK         ;clear the clock
              SHLW    AX,1          ;rotate left to carry
              BNC     $LOW_OUT3     ;if bit = 0 branch to output a LOW signal
              SET1    DATAINPUT     ;output a Hi signal
              BR      NEXT3
LOW_OUT3:
              CLR1    DATAINPUT     ;output LOW signal
              NEXT3:  SET1  CLOCK   ;latch in data
              DBNZ    C,$LOOP3      ;repeat sixteen times
              BR      RSET
END
```

NEC

Some of the information contained in this document may vary from country tocountry. Before using any NEC product in your application, please contact a representative from the NEC office in your country to obtain a list of authorized representatives and distributors who can verify the following:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (France) S.A.
Velizy-Villacoublay,France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, the Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document. NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others. While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features. NEC devices are classified into the following three quality grades: "Standard," "Special," and "Specific." The Specific quality grade applies only to devices developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers may check the quality grade of each device before using it in a particular application. Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots. Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment (not specifically designed for life support). Specific: Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc. The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's data sheets or data books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

In North America: No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics Inc. (NECEL). The information in this document is subject to change without notice. All devices sold by NECEL are covered by the provisions appearing in NECEL Terms and Conditions of Sales only. Including the limitation of liability, warranty, and patent provisions. NECEL makes no warranty, express, statutory, implied or by description, regarding information set forth herein or regarding the freedom of the described devices from patent infringement. NECEL assumes no responsibility for any errors that may appear in this document. NECEL makes no commitments to update or to keep current information contained in this document. The devices listed in this document are not suitable for use in applications such as, but not limited to, aircraft control systems, aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. "Standard" quality grade devices are recommended for computers, office equipment, communication equipment, test and measurement equipment, machine tools, industrial robots, audio and visual equipment, and other consumer products. For automotive and transportation equipment, traffic control systems, anti-disaster and anti-crime systems, it is recommended that the customer contact the responsible NECEL salesperson to determine the reliability requirements for any such application and any cost adder. NECEL does not recommend or approve use of any of its products in life support devices or systems or in any application where failure could result in injury or death. If customers wish to use NECEL devices in applications not intended by NECEL, customer must contact the responsible NECEL sales people to determine NECEL's willingness to support a given application.

U13061EU1V0AN00