To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “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.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

User's Manual

EV9835

Micro-Board for μPD78F9835 Microcontroller

Document no. U17184EU1V0UM00

Contents

1. INTRODUCTION

The EV9835 micro-board is designed to enable you to demonstrate and evaluate the CPU and on-chip peripheral functions of the NEC Electronics 8-bit μPD78F9835 microcontroller (MCU). Out of the box, you can operate the EV9835 as a standalone unit to evaluate key MCU features, such as the high-pixel-capacity LCD driver/controller, complex sound generator, and various power consumption modes. In addition, you can also connect the micro-board to NEC Electronics America's M-Station USB-based baseboard (sold separately), which communicates with the EV9835's ROM monitor so you ca exercise the MCU's peripherals, employ its debugging features, or flash program its memory. Evaluation and development have never been easier.

2. μPD78F9835 MICROCONTROLLER

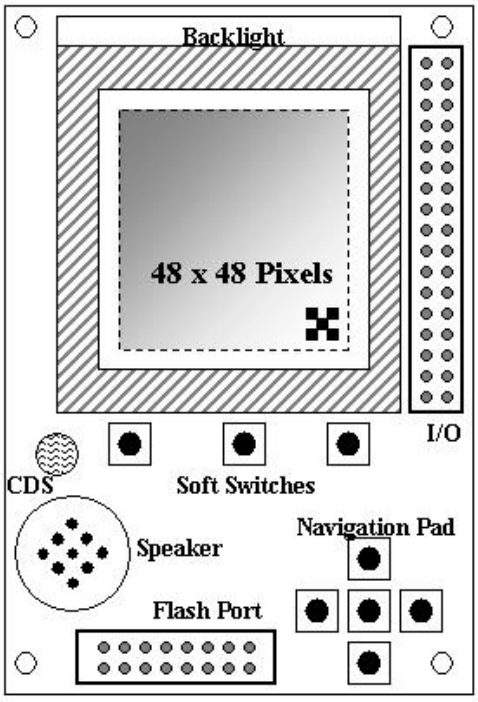
The EV9835's on-board μPD78F9835 MCU is fully explained in its user's manual (document no. U15559EJ1V1UD00). Key features are listed below.

- Memory
  - 60 KB flash program memory
  - 1024-byte high-speed data RAM
  - 2240-byte expansion RAM
- Complex LCD controller/driver with four modes
  - 80 × 8, 80 × 16, 64 × 32, 48 × 48
  - 288 × 2-byte LCD display memory
  - Booster-type LCD reference voltage
- 37 I/O ports
- Three-channel, 8-bit A/D converter
- Serial interface: one-channel UART/3-wire serial I/O
- Timers
  - Six-channel, 8-bit timer/event counter
  - One-channel watch timer
  - One-channel watchdog timer
- Sound generator
  - Volume: 16 levels
  - Scale: three octaves
- 21 vectored interrupts
- 144-pin LQFP (20 × 20 mm)
- Power supply voltage
  - Masked: 1.8–3.6V
  - Flash: 3.0–3.6V
- Ceramic resonator oscillator

3. EV9835 MICRO-BOARD HARDWARE

- Two clock oscillators
  - 5 MHz main clock
  - 32 kHz subclock
- 48 × 48 LCD monochrome backlit display (Densitron part no. 80-0197-000)
- Amplifier and speaker
- Five-key switch navigation matrix
- Three-key soft switch function
- Photocell for ambient light detection
- Multi-chemistries powered from two AA-size battery cells
- 32-pin port connector with all available MCU I/O pins
- 16-pin connector for the debugging and flash programming interface
- Board size of 2.79 × 4.04 inches (70.8 cm × 102.6 cm)

Figure 1. EV9835 Micro-Board Layout



The battery holder, BT1, is located on the underside of the board, below the LCD display module.

Table 1. Factory Jumper Settings

Jumper	Setting	Function	Description
JP1A	1–2	Battery power	Battery power selection
	2–3	External DC	User-supplied external power selection
	No connection	Powered by programmer	Jumper removed for flash programming
J1	Closed	Audio amplifier shutdown	Cut trace to permanently disable audio amplifier
J2	Closed	+ Speaker output	Cut trace for alternate port use
J3	Closed	– Speaker output	Cut trace for alternate port use
J4	Open	+ DC external	User-supplied external positive DC voltage
J5	Open	– DC external	User-supplied external ground
J6	Open	+ 3.3 V_{DC}	Regulator output for expansion circuits (100 mA max.)
J7	Open	Ground return	Regulator ground (system ground)
J8	Closed	Power indicator	Cut trace for alternate port use (low battery/power good)
J9	Closed	3.3V regulator enable	Cut trace to put 3.3V regulator in snooze mode
J10	Closed	CDS photocell input	Cut trace for alternate port use (no optical sensor)
J12	Closed	LED backlight shutdown	Cut trace to permanently disable the LED backlight

3.1 Operation

3.1.1 Default Configuration

The default settings in the EV9835 micro-board enable all of the following features:

- Power supply snooze function
- Sound generator amplifier
- Backlight brightness control
- Ambient light detector
- Power good or low battery detection

3.1.2 Debugging User Programs

1. Connect the EV9835 micro-board to the M-Station baseboard through the 16-pin header.
2. Use the serial port for both debugging and flash programming.
3. The EV9835 is preprogrammed with a ROM monitor and application code for you to explore. See section 4 for additional information.
4. Since the M-Station will supply power to the EV9835, remove the JP1A jumper when connecting the 16-pin cable to the M-Station. After debugging and flash programming, re-insert the JP1A jumper for battery or external DC power.
5. Choose one of the following methods for debugging:
    - The M-Station flash programming software, FP4MST, to reprogram the μPD78F9835's program memory
    - The M-Station terminal emulation software, MSTTERM, to communicate between the PC and the UART on the μPD78F9835 so that your programs can transmit and receive serial data
    - The M-Station integrated debugging software, ID78K0-MST, to communicate with the embedded ROM monitor program for real-time program debugging

3.1.3 Reprogramming the Microcontroller's Flash Memory

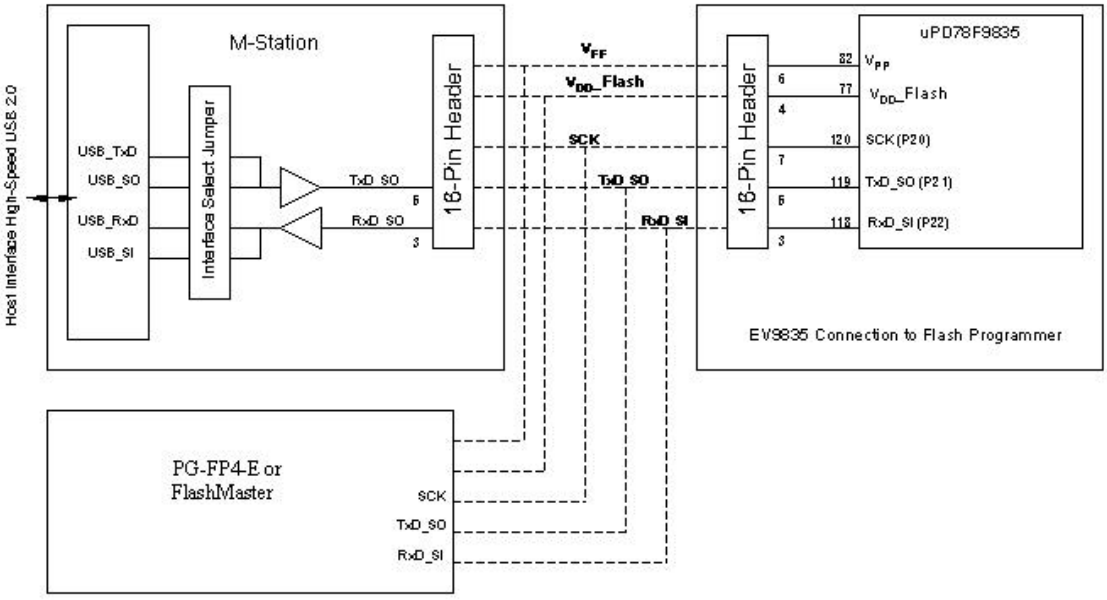
The MCU's on-chip flash memory can be reprogrammed at any time using the built-in flash programming features of the M-Station baseboard or the PG-FP4-E flash programmer (available for purchase from your local NEC Electronics America representative/distributor). The EV9835 micro-board's 16-pin debugging and flash programming interface is a common NEC Electronics standard for all of its programming tools.

The pre-programmed ROM monitor and application code are available on the CD-ROM shipped with the designer kit, should you want to use all or part of it.

When reprogramming, be sure to set the programmer to the correct communication mode.

- Refer to the *M-Station User's Guide* before setting the M-Station programming interface signals.
- Refer to the *PG-FP4-E User's Manual* before using the PG-FP4-E programmer.

Figure 2. EV9835 Micro-Board Connection to Flash Programmer



3.1.4 Measuring the Microcontroller's Power Consumption

You can measure power consumption of the EV9835 micro-board at the JP1A power selection jumper.

1. Insert a milliamp (mA) meter or digital voltmeter across the JP1A pins (pins 1 and 2 when using a battery or pins 2 and 3 when using an external supply).
2. Make sure you observe the correct polarity of the meter in relation to the jumper pins. The positive meter terminal should be attached to the supply side (pins 1 or 3) and the negative meter terminal to the load (pin 2).
3. Measure the current drawn or the power savings.

Be advised that the peripheral circuitry also draws current, but each circuit is under the control of the MCU and either can be turned off or placed in a low-current-consuming mode using ROM monitor commands.

Figure 3. JP1A Pins

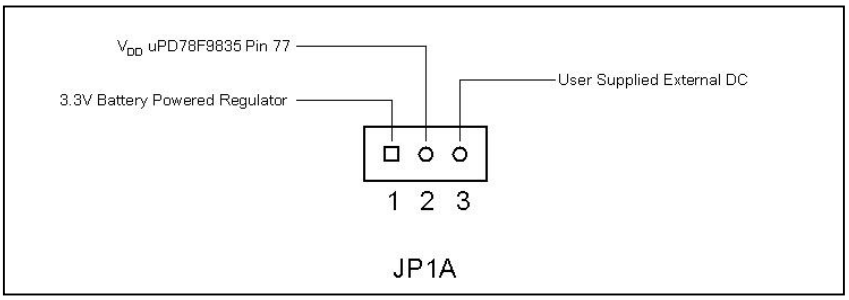
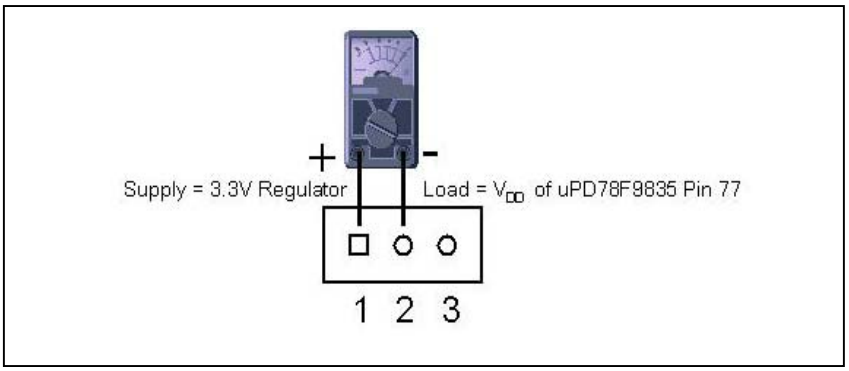


Figure 4. Measuring Power Consumption with External Batteries



3.2 On-Board Components

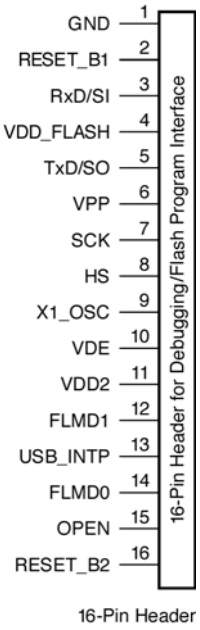
3.2.1 µPD78F9835 Microcontroller

The µPD78F9835 is a member of the NEC Electronics 78K0S family of MCUs, each of which includes a very powerful LCD driver/controller, two crystal clock sources, a main crystal of 4.9152 MHz, and a subclock oscillator of 32.768 kHz.

Using external capacitors on pins 87–91, you can set five levels of bias voltage to drive the LCD. To boost the internal V_{DD} to 5 volts and drive the LCD to high contrast, use capacitors on pins 83–86. The U3A IC resets the MCU upon first application of voltage. Use the S1A switch to perform a manual reset.

3.2.2 16-Pin Debugging and Flash Programming Interface Header

Figure 5. 16-Pin Header



3.2.3 48 × 48 Pixel Graphics LCD

The EV9835 micro-board supports only one of the four graphic modes of the MCU's LCD controller: 48 × 48 pixels. The capacitive boost circuitry that provides the 5 volts to drive the display is integrated into the MCU. Four white LEDs under the control of the MCU's P11port pin (pin 103) provide backlight for the monochrome LCD. By pulse-width modulating this port pin, you can adjust the brightness of the display while it is backlit. The 48 × 48 matrix of pixels makes it possible to generate alphanumeric characters of various fonts and sizes or any graphical shape, limited only by your imagination.

The LCD controller's memory is split into two "pages" that allow easy implementation of animated features such as sprites, as one screen can be drawn while the other is displayed. The CDS photocell also is part of the display system. So that a digital threshold can be set for detecting ambient light, the CDS photocell's output signal is applied to the input of the internal A/D converter and used to determine when to use the backlight, thus saving battery power.

The μPD78F9835's on-board LCD controller/driver automatically drives waveforms on the common and segment lines to turn the desired pixel on or off. There are 48 common signals and 48 segment signals, for a duty cycle of 1/48 and a bias of 1/5. Writing either a zero or a one into a bit of internal memory causes the appropriate pixel to turn on (if the bit is 1) or off (if the bit is 0). Table 14-3 in the *μPD78F9835 User's Manual* lists the actual pin names and number assignments.

3.2.4 Switch Functions

Switch 1 is the reset switch located on the rear of the board (opposite the display). Switches 2, 3 and 4 are “soft” switches located directly under the display. The functionality of these switches is under the control of your application software, and can be used to display the switch function at the bottom of the LCD, similar to using the soft switches in a cellular telephone.

Switches 5, 6, 7, 8 and 9 are positioned in a navigational matrix, and also are under the control of your application software. You can use this switch matrix to scroll through items in the display in a menu fashion, using the up and down switches to move through the selection, the left and right keys to open more options, and the center switch as the ENTER key.

3.2.5 CDS Photocell

R12A is a cadmium-sulfide, light-dependent resistor biased to half the supply voltage for a nominal light input; this signal is coupled directly into analog input AN0 (pin 94). By comparing the digital conversion result to a stored table, you can determine the relative amount of ambient light by experimentation. Some approximate values are shown in Table 2.

Table 2. Approximate Values for Ambient Light

Ambient Light Condition	CDS Relative Resistance	A/D Digital Code
Dark	500 KΩ	3FH
Dim	10 KΩ	E4H
Bright	100 Ω	FFH

3.2.6 Voltage Regulator

The EV9835 micro-board derives power from two AA-size batteries and supports multiple chemistries, such as carbon, alkaline, nickel cadmium (Ni-Cad), or nickel metal hydride (NiMH). NEC Electronics America recommends NiMH batteries because they offer the greatest energy density. However, there is no provision for charging secondary batteries, as it is assumed you will accomplish this externally in the appropriate charger. The on-board voltage regulator accepts inputs as low as 1.8V and will boost and regulate them to ensure a constant 3.3V supply for the MCU and support circuitry.

3.2.7 Audio Amplifier

The audio amplifier is AC-coupled to the differential output of the sound generator in the μPD78F9835. The gain of this external amplifier is fixed, since the sound generator inside the MCU has 16 adjustable volume levels. If you need to use the SGEN (sound generator enable) port pin for other functions, cut the J12A cut-trace to free port pin P27 (pin 113) and disable the amplifier. To re-enable the amplifier, simply wire a jumper from the amplifier side (pin 1) to V_{DD}.

3.2.8 LED Backlight

Four white LEDs driven by a boost-switching LED driver provide backlight for the LCD. Operation of the driver is controlled by a shutdown pin, giving you on and off control via software. Furthermore, by pulse-width modulating this pin, you can adjust the brightness level of the LED backlight.

3.2.9 Additional User I/O

The 32-position J11A header can be used to connect to external circuitry, as shown in Table 3.

Table 3. User I/O Pins

User I/O Port	Function	User I/O Port	Function
1	GND	17	P23
2	GND	18	P24
3	V_{DD}	19	P25
4	P00	20	P26
5	P01	21	P27
6	P02	22	P30
7	P03	23	P31
8	P04	24	P32
9	P05	25	P33
10	P06	26	P34
11	P07	27	P35
12	P10	28	P36
13	P11	29	P37
14	P20	30	P60
15	P21	31	P61
16	P22	32	P62

4. SOFTWARE

When the EV9835 is powered up with the sample application code provided by NEC Electronics America, it displays a welcome message. Pressing any switch causes the unit to go into Select Mode, at which point you can select a desired mode of operation. If you don't press a switch within five seconds after power up, the unit automatically reverts to its default operating mode.

4.1 Commonly Used Software Functions

The EV9835 micro-board's LCD can display graphics or text. When used to display text, it is treated as a 6-line × 8-character display, with lines numbered 0–5 top to bottom and character positions numbered 0–7 left to right. The next two sections provide more detail about the available character and graphic functions.

4.1.1 Character Display Functions

4.1.1.1 * Function: DrawCharacter

To display a character on the LCD, you must specify that character, its line number, and position on the line using the ***DrawCharacter( )*** function.

```
*Parameters:     character to be displayed
                 line number
                 position on line
* Returns:       nothing
* Example:       DrawCharacter('A',4,1);
```

4.1.1.2 * Function: DrawString

To display a string of characters, use the DrawString() function.

```
*Parameters:     pointer to string
                 line number
                 position on line
* Returns:       nothing
* Example:       DrawString("string",5,0);
```

4.1.1.3 * Function: Scroll_ROM_String

For strings longer than eight characters, use the Scroll_ROM_String( ) function to enable the string for scrolling across the display.

```
* Parameters:    pointer to first character
* Returns:       nothing
* Example:       Scroll_ROM_String(pointer_to_string)
```

4.1.1.4 * Function: TM4_Interrupt

Timer4 is used as the time base to provide the shift clock that moves the displayed string one character to the left on each interrupt.

```
* Parameters:    none
* Returns:       nothing
```

4.1.1.5 * Function: Scroll_Step_ROM_String

The TM4_Interrupt() routine calls Scroll_Step_ROM_String to shift the displayed message one character to the left.

```
* Parameters:    none
* Returns:       nothing
```

4.1.2 Graphics Display Functions

The graphics functions use an x,y coordinate system to draw on the 48 × 48 pixels of the LCD. The x coordinate runs from 0–47 left to right across the display; the y coordinate runs from 0–47 top to bottom. The graphics functions, in addition to enabling you to specify the x,y coordinates, also enable you to specify the pixel color using a black dot to draw a visible pixel and a clear dot to erase a visible pixel.

4.1.2.1 * Function: DrawDot

```
* Parameters:    (x,y,color)
* Returns:       nothing
* Example:       DrawDot(14,4,BLACK_DOT);
```

4.1.2.2 * Function: DrawHorizontalLine

```
* Parameters:    (x1,x2,y,color)
* Returns:       nothing
* Example:       DrawHorizontalLine(10,20,40,BLACK_DOT);
```

4.1.2.3 * Function: DrawVerticalLine

```
* Parameters:    (x,y1,y2,color)
* Returns:       nothing
* Example:       DrawVerticalLine(10,20,40,CLEAR_DOT);
```

4.1.2.4 * Function: DrawDiagonalLine

```
* Parameters:    (x1,y1,x2,y2,color)
* Returns:       nothing
* Example:       DrawDiagonalLine(10,20,20,30,CLEAR_DOT)
```

4.1.2.5 * Function: DrawRectangle

```
* Parameters:    (x1,y1,x2,y2,color)
* Returns:       nothing
* Example:       DrawRectangle(5,25,20,40,BLACK_DOT);
```

4.1.2.6 * Function: DrawCircle

```
* Parameters:    (x,y,radius,color)
* Returns:       nothing
* Example:       DrawCircle(23,23,20,BLACK_DOT);
```

4.1.3 Switch Functions

The EV9835 switches under your control are labeled SW2–SW9 (SW1 is the reset switch). Pushing any one of these causes a Switch_Interrupt( ) that starts Timer80 for use as a debounce timer. To debounce the switches and set flags to indicate the switch status, use the TM80_Interrupt( ) routine.

4.1.3.1 * Function: Switch_Interrupt

```
* Parameters:    none
* Returns:       nothing
```

4.1.3.2 * Function: TM80_Interrupt

```
* Parameters:    none
* Returns:       nothing
```

4.1.3.3 Switch Control

The effect of the eight switches (SW2–SW9) is generally dependent on the current operating mode of the application software. However, SW8, the center switch of the navigational matrix, is consistent in function despite the mode.

- Pressing SW8 in any mode causes the EV9835 micro-board to show the Select Mode display (equivalent to the home page), from which you can view the current operating mode (MODE X) or step through and select another one.
- Pressing SW7 increments the mode; pressing SW9 decrements it.
- Pressing SW6 at any stage selects the displayed mode. If you don't press a switch within five seconds, then the mode reverts to the one that was set before you pressed SW8.
- Pressing SW8 while you are already in Select Mode will toggle the LCD backlight between the on and auto-sensing modes. Auto-sensing mode automatically senses the light from the CDS photocell and turns the backlight on or off depending on the detected light level. The thresholds for turning on and off the light are set as defined in the *common.h* file.

As SW8 is commonly used to enter Select Mode, a separate Check_Switch_Select( ) routine is used.

```
* Function:      Check_Switch_Select
* Parameters:    none
* Returns:       1 if SW8 press detected
                 0 otherwise
* Example:       if(Check_Switch_Select( )) new_mode=TRUE;
```

4.1.4 Timekeeping Functions

The MCU's watch timer peripheral is initialized to run off the 32.768 kHz subsystem clock and generate a 0.5-second tick for use in timekeeping functions. The Watch_Timer_Interrupt uses this tick to update the real-time clock, calendar date, elapsed time counter, and countdown timer.

```
* Function:      Watch_Timer_Interrupt
* Parameters:    none
* Returns:       nothing
```

4.2 Operating Modes

Table 4 shows the list of operating modes along with a brief description of each. For a more detailed description of these modes, please see the sections following.

Table 4. Description of Modes

Mode	Name	Description
1	User message	Scrolling user message with time display
2	Soft key	Soft key graphics displayed to display switch menu
3	Home security	Arm/disarm/sound alarm plus application graphics
4	Sprinkler one	Set start time and duration of sprinkler zone plus application graphics
5	Pseudo-random number generator	Generation and display of pseudo-random numbers
6	Combination Lock	Multi-digit combination lock with safe graphic
7	Radar	Scan-line radar display
8	Emoticon	Animated emoticon graphic

4.2.1 Mode 1: Message Line Scrolling

The scrolling message is a 30-character string stored in ROM. The message currently stored in the application code is "*EV9835* NEC ELECTRONICS AMERICA, INC. " This mode scrolls the message across the display while simultaneously displaying the time and date.

4.2.2 Mode 2: Setting the Time and Date Using Soft Keys

In this mode, you can view the time and date or use the soft keys at the bottom of the display—above switches SW2, SW3 and SW4—to navigate the menu when changing the time and date.

1. When you enter this mode, you will see the soft key [S] displayed above SW3.
2. Press SW3 to view the Settings Display display.
3. Press soft key [A] to view the Set Time display.
4. Press soft key [B] to view the Set Date display.

4.2.2.1 Setting the Time

The LCD initially displays the time in the format previously set and also provides these options.

- Press soft key [A] to display the time in 12-hour format (hh:mm am/pm).
- Press soft key [B] to display the time in 24-hour format (hh:mm:ss).
- Press soft key [=] to accept the display format and launch the Set Time display. The time is always set in 24-hour clock format; you will see the display hh:mm:ss with the hh digits initially flashing.
- Press soft key [+] to increment the flashing digits; press soft key [=] to accept the displayed value and move to the next set of digits.
- Press soft key [=] to accept the displayed seconds value, finish setting the time, and return to the display showing the newly set time and date.

Use the SetTimeUsingSwitches( ) function to set the time and the DrawTime() function to refresh the displayed time.

4.2.2.1.1 * Function: SetTimeUsingSwitches

```
* Parameters:    none
* Returns:       nothing
* Example:       SetTimeUsingSwitches();
```

4.2.2.1.2 * Function: DrawTime

```
* Parameters:    line number
                 time format
* Returns:       nothing
* Example:       DrawTime(3,12_HOUR);
```

4.2.2.2 Setting the Date

The LCD displays the date in the format currently set and provides these options.

- Press soft key [A] to display the time in U.S. format (mm-dd-yy).
- Press soft key [B] to display the time in non-U.S. format (dd-mm-yy).
- Press soft key [=] to accept the displayed format and launch the Set Date display. The date is always set using the U.S. date format; you will see the display mm-dd-yy with the mm digits initially flashing.
- Press soft key [+] to increment the flashing digits; press soft key [=] to accept the displayed value and move to the next set of digits.
- Press soft key [=] to accept the value displayed, finish setting the date, and return to the display showing the newly set time and date.

4.2.2.2.1 *Function: SetDateUsingSwitches

```
* Parameters:    none
* Returns:       nothing
* Example:       SetDateUsingSwitches();
```

4.2.2.2.2 * Function: DrawDate

```
* Parameters:    line number
                 date format
* Returns:       nothing
* Example:       DrawDate(3,us_format);
```

4.2.3 Mode 3: Home Security Application

This mode simulates a home security system by displaying a graphical outline of a typical home with doors and windows. The doors and windows can be opened or closed by you. The security system can be armed or disarmed. The system can only be armed when all doors and windows are closed. Opening a door or window while the system is armed causes the alarm to sound.

- SW2 opens/closes the front door.
- SW3 opens/closes the left front window.
- SW4 opens/closes the left back window.
- SW5 opens/closes the back window.
- SW6 opens/closes the right back window.
- SW7 opens/closes the back door.
- SW9 arms/disarms the security alarm.

In this application, the small back window represents a security risk because it has not been wired into the security system.

4.2.3.1.1 * Function: DrawSecurityGraphic()

```
* Parameters:    none
* Returns:       nothing
* Example:       DrawSecurityGraphic();
```

4.2.3.1.2 * Function: ToggleEntryState()

```
* Parameters:    entry type
* Returns:       nothing
* Example:       ToggleEntryState(front_door);
```

4.2.3.1.3 * Function: ToggleArmedState()

```
* Parameters:    none
* Returns:       nothing
* Example:       ToggleArmedState();
```

4.2.3.1.4 * Function: SoundSecurityAlarm()

```
* Parameters:    none
* Returns:       nothing
* Example:       SoundSecurityAlarm();
```

4.2.3.1.5 * Function: SilenceSecurityAlarm()

```
* Parameters:    none
* Returns:       nothing
* Example:       SilenceSecurityAlarm();
```

4.2.3.1.6 * Function: PlaySound()

```
* Parameters:    sound frequency
* Returns:       nothing
* Example:       PlaySound(freq_2khz);
```

4.2.3.1.7 * Function: StopSound()

```
* Parameters:    none
* Returns:       nothing
* Example:       StopSound();
```

4.2.4 Mode 4: Sprinkler Zone Application

This mode displays a diagram of a sprinkler system with three zones. A starting time plus sprinkler duration time can be set for all three zones individually. The Z1, Z2 and Z3 zone identifiers change to

the word “On” when the sprinkler is in operation and back again after the specified duration has elapsed.

- SW2 cycles the edit box between the zone starting time and zone duration time.
- SW3 increments the parameter in the zone edit box.
- SW4 confirms the displayed settings for zone and returns to the graphical zone display.
- SW5 displays zone2 (Z2) information.
- SW7 displays zone3 (Z3) information.
- SW9 displays zone1 (Z1) information.

4.2.4.1.1 * Function: DrawSprinklerGraphic()

```
* Parameters:    none
* Returns:       nothing
* Example:       DrawSprinklerGraphic();
```

4.2.4.1.2 * Function: DrawZoneInformation()

```
* Parameters:    zone number
* Returns:       nothing
* Example:       DrawZoneInformation(zone_number);
```

4.2.4.1.3 * Function: DrawZoneEditBox()

```
* Parameters:    edit box number
* Returns:       nothing
* Example:       DrawZoneEditBox(zone_start_time);
```

4.2.4.1.4 * Function: EditBoxIncrement()

```
* Parameters:    edit box number
* Returns:       nothing
* Example:       EditBoxIncrement(zone_start_time);
```

4.2.4.1.5 * Function: DrawStartTime()

```
* Parameters:    zone number
* Returns:       nothing
* Example:       DrawStartTime(zone_number);
```

4.2.4.1.6 * Function: SetSprinklerStates()

```
* Parameters:    none
* Returns:       nothing
* Example:       SetSprinklerStates();
```

4.2.5 Mode 5: Pseudo-Random Number Generator Application

This mode displays a pseudo-random number ranging from 0 to 32,767. Pressing SW3 causes a new pseudo-random number to be generated and displayed. Use the stdlib.h library rand( ) function to generate a pseudo-random number, the WordToDecString( ) utility to convert the random number to a decimal character string, and the DrawString() routine to display the string.

4.2.5.1 * Function: rand()

```
* Parameters:    none
* Returns:       random integer
* Example:       pseudo_int=rand()
```

4.2.5.2 * Function: WordToDecString

```
* Parameters:    integer
                 leading zeroes (Boolean)
* Returns:       array of decimal characters
* Example:       WordToDecString(0x0234,FALSE);
                 (result stored in global array)
```

4.2.6 Mode 6: Combination Lock Application

This mode displays the diagram of a safe with a door shown opened or closed to indicate the unlocked or locked state, respectively. The safe is guarded by a five-digit combination lock. Upon first entering this mode, the safe is unlocked and the combination is set at 00000.

- SW2 increments the first digit of the dial by one.
- SW3: increments the second digit of the dial by one.
- SW4 increments the third digit of dial by one.
- SW5 increments the fourth digit of the dial by one.
- SW7 increments the fifth digit of the dial by one.
- SW6 locks / unlocks the safe.

4.2.6.1 Setting the Combination

When the safe is unlocked, changing the combination from all zeroes causes the word “Set” to appear in the display, indicating that a new combination is in the process of being set. Pressing SW2–SW6 as defined above increments the corresponding digits until the combination desired is set. Pressing SW6 locks the safe with the currently displayed combination. Once the safe is locked, the combination resets to all zeroes.

4.2.6.2 Unlocking the Safe

The combination dial digits are changed using SW2–SW6. If the dial is at the correct combination then pressing SW6 causes the safe to unlock and the graphic to change to show the safe with an open door. The dial display shows *****, masking the safe combination.

4.2.6.3 Locking the Safe

When the safe has been unlocked and is displaying ***** to mask the combination, then pressing SW6 locks the safe and resets the combination dial digits to all zeroes. Pressing any of the combination dial switches SW2—SW6 leaves the safe open but resets the dial digits to all zeroes, indicating that the program is ready for you to set a new combination.

- Use the IncrementDialDigit( ) function to increment the combination dial digits and the LockAndUpdateCombination( ) function to lock the safe and update the combination.
- Use the UnlockSafe() function to unlock the safe.
- Use the DrawSafeGraphic( ) and DrawSafeDoor( ) functions together to draw the safe graphic with a closed or open door, and these supporting functions to display the dial digits.
  - DrawDot( )
  - DrawHorizontalLine( )
  - DrawVerticalLine( )
  - DrawDiagonalLine( )
  - DrawRectangle( )
  - DrawCharacter( )
  - DrawString( )

4.2.6.3.1 * Function: IncrementDialDigit()

```
* Parameters:    dial digit
* Returns:       nothing
* Example:       IncrementDialDigit(3);
```

4.2.6.3.2 * Function: LockAndUpdateCombination()

```
* Parameters:    none
* Returns:       nothing
* Example:       LockAndUpdateCombination();
```

4.2.6.3.3 * Function: UnlockSafe()

```
* Parameters:    none
* Returns:       nothing
* Example:       UnlockSafe();
```

4.2.6.3.4 * Function: DrawSafeGraphic()

```
* Parameters:    none
* Returns:       nothing
* Example:       DrawSafeGraphic();
```

4.2.6.3.5 * Function: DrawSafeDoor()

```
* Parameters:    door_state
* Returns:       nothing
* Example:       DrawSafeDoor(CLOSED);
```

4.2.7 Mode 7: Radar Application

This mode is a simulation of radar display function and is divided in half. The top part of the screen displays a scanning radar line that blips as the vertical scan line encounters obstacles. The bottom half of the display draws the radar echo of the obstacle encountered.

The main radar application functions used are ScanRadarLine( ),ClearRadarLine( ), DrawRadarEcho(), and ClearRadarEcho( ). Supporting graphics functions include DrawString( ),DrawDot( ), DrawVerticalLine( ), DrawHorizontalLine( ).

4.2.7.1 * Function: ScanRadarLine()

```
* Parameters:    number of scan line
* Returns:       y coordinate of any obstacle
* Example:       echo_y=ScanRadarLine(9);
```

4.2.7.2 * Function: ClearRadarLine()

```
* Parameters:    number of scan line
* Returns:       nothing
* Example:       ClearRadarLine(9);
```

4.2.7.3 * Function: DrawRadarEcho( )

```
* Parameters:    y coordinate of any obstacle
* Returns:       nothing
* Example:       DrawRadarEcho(echo_y);
```

4.2.7.4 * Function: ClearRadarEcho( )

```
* Parameters:    none
* Returns:       nothing
* Example:       ClearRadarEcho( );
```

4.2.8 Mode 8: Emoticon Application

This mode displays an animated emoticon. The animation effect is achieved by drawing sixteen different emoticon frames in rapid succession. While one frame is being displayed, the next frame is being drawn in the alternate LCD RAM pattern. The functions used are AnimateEmoticon( ) and DrawEmoticonFrame( ).Supporting graphics functions include DrawDot( ), DrawVerticalLine( ), DrawHorizontalLine( ), DrawDiagonalLine( ), DrawCircle( ).

4.2.8.1 * Function: AnimateEmoticon()

```
* Parameters:    none
* Returns:       nothing
* Example:       AnimateEmoticon;
```

4.2.8.2 * Function: DrawEmoticonFrame()

```
* Parameters:    frame number
* Returns:       nothing
* Example:       DrawEmoticonFrame(9);
```

5. K0SM9835 ROM MINI-MONITOR

The K0SMM runs self-contained within a single-chip 78K0S device and allows you to examine all memory locations and change the contents of most locations that are changeable, such as RAM or the SFRs. Additional functions allow you to read the A/D conversion values, read a simple real-time clock, and put the MCU in HALT or STOP mode to facilitate the measurement of current consumption. The K0SMM already has proven useful as a means for "bringing up" target hardware the first time. With little or no modification, you can use a K0SMM-programmed MCU to exercise your target hardware immediately, and test your board's hardware functionality.

This ROM monitor easily can be included in whole or part into your application software, and can provide useful diagnostic functions when embedded in the end application. The code is free to use and modify, but no claims to its robustness or applicability to the application are made.

The K0SM9835 ROM monitor is made to communicate from the M-Station to the EV9835 micro-board. The command interface is built around two-letter command codes. Some commands require additional operands, such as address or data values.

5.1 General Usage Guidelines

Alpha input is not case-sensitive. The monitor prompt (>) indicates that the program is ready to process a command. Spaces are not required between command codes and operand data, but they do make for easier readability.

Numeric entries default to hexadecimal values, with valid choices being 0–9 and A–F. Leading zeros (0x)and a trailing "h" are not required. Decimal numbers may be entered by preceding the number with a period (.); valid digit choices are limited to 0–9. Binary numbers may be entered by preceding the number with a percent symbol (%); valid digit choices are now limited to 0–1.

A special mode has been included that allows the customer to access SFRs by name. This requires a fair amount of ROM, but is a handy way to examine or modify SFRs without having to look up their addresses. This feature can be removed to minimize ROM space usage if so desired.

5.2 Device-Specific Extensions for μPD78F9835

The K0SM9835, a ROM monitor program written in assembly language, uses the following on-chip resources:

- Serial interface configured for UART mode
- ROM space of about 5 KB, including SFR access mode table
- Six-byte SADDR RAM (FE20h–FE25h)
- 160 bytes of non-SADDR RAM
  - 64-byte stack
  - 48-byte serial receive buffer
  - 48-byte command process buffer
- Reset and serial receive interrupt vectors

The serial receive routine is interrupt-driven, and places received characters into a circular receive buffer. The ROM used by the monitor code includes a number of potentially useful subroutines, i.e. serial receive, numeric entry via serial port, binary-to-ASCII decimal conversion, etc., that are available for customer use. Please refer to the source code listing for more information.

5.3 Monitor Commands

A brief list of the currently supported commands may be obtained by entering the **HELP** command. Either the entire word or the two-character abbreviation can be entered. The following list was obtained from the K0SM9835 running on a μPD78F9835 MCU:

```
AD n Read A/D converter channel (n=0-6) value

CE Toggle serial char echo

CM n [=MK0,MK1] Current Measurement mode (n=1-5) (see
instructions)

CR Return copyright notice

DH addr1 [-addr2] Dump memory block as Intel hex

FV Return firmware version information

HE Display this list!

MB addr.bit [=0][=1] Modify Bit (0-7) at location (addr)

MD addr1 [-addr2] MCU Memory Display (read only)

MF addr1 [-addr2] =byte Memory Fill from addr1 - addr2

MM addr1 [=byte] [,byte2,byte3,...] MCU Memory direct R/W access

NE n Numeric entry test
```

```
RC Calculate 16-bit ROM checksum
SF sfr [=byte/word]] SFR name-based access (SF? for list)
XT Restart monitor, do complete re-initialization
```

5.4 Commands Specific to K0SM9835

5.4.1 AD n: Read A/D converter channel (n=0–6) value

This command turns on the A/D converter, performs an A/D conversion on the selected channel, returns the 8-bit result, and shuts down the A/D converter after the command is executed.

For example, this command performs an A/D conversion on analog channel 0 and displays the results in hexadecimal and (decimal) formats:

```
>ad 0
0069 (00105)
```

The A/D conversion clock must be set so that the conversion time is not less than 14μs. The K0SM9835 selects a conversion clock of $f_x/72$, and since f_x is assumed to be 4.9152 MHz, this equation gives a conversion time of 14.6μs.

The A/D conversion value depends on the voltage at the AV_{REF} pin. On the K0RE9835 board, AV_{REF} is connected to AV_{DD}/V_{DD} through a trace jumper. See the *μPD78F9835 User's Manual* for more information about the A/D converter.

5.4.2 CM n [=MK0,MK1]: Current Measurement mode (n=1–5)

This command puts the MCU into defined states to facilitate the measurement of operating current. There is no return from this command, and you must press the RESET button to exit. If you don't specify values for interrupt mask registers MK0 and MK1, the CPU remains in the specified mode forever, until you press the RESET button. The UART turns off prior to entering any of the current measurement modes. You may enable or disable all other peripherals by modifying the SFRs.

Table 5. Combinations of CPU Clock Source and Halt Modes

Monitor Mode	State	CPU Mode	Main Clock	Subclock	Note
CM1	RUN	Main	Run	Run*	
CM2	HALT	Main	Run	Run*	Wake on interrupt
CM3	STOP	Main	Stop	Run*	Wake on interrupt
CM4	RUN	Sub	Stop	Run	
CM5	HALT	Sub	Stop	Run	Wake on interrupt

The following is an example of the **CM** command being used to put the μPD78F9835 into mode 1, where the CPU is running at full speed on the main clock:

```
>cm1

CM1 - CPU in RUN mode on main clock

Subclock is running

Interrupt mask registers MK0,MK1 = FF,FF

Press RESET switch to exit mode
```

The MCU executes a tight loop when operating in any of the continuous run modes:

```
Loop:      XOR   P0,#01            ; Toggle port 0 bit 0 for
monitoring with oscilloscope

BR   $Loop       ; This loop typically requires 12 clock cycles
```

If you want to observe P00 toggling, you must manually set its corresponding port mode register bit (PM00) to a 0 to configure P00 as an output. This can be done with the SF command to access PM0. In the following example, the command configures bit 0 of the μPD78F9835's eight-bit I/O port 0 (P00) as an output; the remaining bits (P01–P07) are left as inputs:

```
>sf pm0=e
```

See the SF command instruction for more information on how to access the MCU's special-function registers by name. To determine the MCU's actual clock rate, measure the high or low period of the waveform (not BOTH!) and divide by number of clock cycles required to execute the loop. Typically, 12 instruction clock cycles are used while executing the loop (check the MCU user's manual to be sure); invert the result to give the MCU's clock frequency.

Table 6. Expected Values at V_{DD} = 3.3V (Main Clock = 4.9152 MHz; Subclock = 32.768 kHz)

Mode	P00 High/Low	Clock Period	Clock Frequency
1	2.441 μs	203.5 ns	4.9152 MHz
1	9.766 μs	813.8 ns	1.2288 MHz
4	183.1 μs	61.04 μs	16.384 kHz

Current measurement operating modes 3 and 5 put the CPU into the HALT or STOP modes. In these cases, the execution loop looks more like the following:

```
Loop:      XOR   P0,#01            ; Toggle port 0 bit 0 for
monitoring with oscilloscope

HALT/STOP        ; halt or stop the CPU

NOP              ; any unmasked, active interrupt source wakes
MCU

(clear active interrupt flags)

BR   $Loop       ; This loop typically requires 12 clock cycles
```

When you don't specify values for the MK0 and MK1 interrupt mask registers, the CPU remains in HALT or STOP mode until you press the RESET button. You may manually enable various interrupt sources (timers and so forth) that will wake the CPU, but if you directly unmask an active interrupt source by clearing its mask bit in the MK*n* register, you will receive an 'Unsupported Interrupt' error message and the monitor will re-initialize.

The **CM** command allows you to indirectly set the mask values so you can observe the selected active interrupt source waking the CPU from the HALT or STOP mode. Please note that if the MK*n* values are not specified, they will default to FFh (no interrupts active). To specify MK*n* values other than the default settings, you must specify values for ALL interrupt mask registers available for that MCU. Setting the MK*n* values in operating modes where the CPU is running continuously will have no effect.

The following is a complete example about how to set the μPD78F9835 to operate in HALT mode, toggling P00 as the watch timer creates a 500 ms interrupt.

```
>sf wtm=83 Set watch timer mode register to 83h to enable watch
timer using subclock.

>cm2=7f,ff Select CPU HALT mode on main clock and unmask watch
timer interrupt

CM2 - CPU in HALT mode on main clock

Subclock is running

Interrupt mask registers MK0,MK1 = 7F,FF

Press RESET switch to exit mode
```

As you can see, P00 toggles every time the CPU awakens. (Please refer to the *μPD789835 User's Manual* for more information about standby mode.)

The following are examples of the **Memory Display** (**MD)** command in operation. In the first, no address field is specified and the monitor displays memory at the default starting address of 0000. The hexadecimal values of the memory locations are shown, along with the ASCII values corresponding to the memory contents. Undisplayable characters are replaced with periods.

```
>md
0000: 80 00 FF FF 5F 0B 5F 0B 5F 0B 5F 0B 5F 0B D9 0A
0010: 5F 0B A2 0A D3 0A 46 0A 5F 0B 5F 0B 5F 0B 5F 0B
0020: 5F 0B 5F 0B FF FF FF FF FF FF FF FF FF FF FF FF
0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080: F0 1F FD E6 1C F7 FB 00 F7 F0 00 F7 F2 00 F7 20
0090: FF 0A 06 F7 F7 F3 FD F5 02 02 F7 22 FD F7 24 FF
00A0: 0A 46 F7 F5 05 FF F7 25 00 F7 28 FF 0A 06 F4 0A
00B0: 16 F4 0A 26 F4 0A 36 F4 F7 29 FF 0A 46 F4 0A 56
00C0: F4 F0 00 FD EC 0A F3 00 EF 8C CC E2 E0 FE CC 38
00D0: F7 F7 72 00 F7 73 40 F7 70 CC 27 71 25 10 0A 66
00E0: E0 0A D6 E4 F7 4A 83 0A F6 E4 F7 53 02 F7 50 4F
00F0: F7 53 82 0A 96 E5 0A 7A 1E 0A F3 11 22 E9 07 22
```

This example specifies a starting address (in hexadecimal format) for the memory display:

```
>md 7f0
07F0: E6 E0 E5 10 A0 20 A2 2F 13 04 3C 08 22 E9 07 DC
0800: 80 EC 30 F3 A0 20 AE 25 23 15 22 3A 10 A2 25 00
0810: C0 0A F3 00 D2 20 FD EC A0 2F 04 30 01 14 AC 20
0820: A6 AA AE F8 00 00 0A F7 07 25 23 15 22 3A 2F 25
0830: 23 C0 0A F3 00 D2 20 FD EC 0A 27 13 07 3E 21 2F
0840: 13 20 3E 03 B2 C9 08 13 25 3E 05 0A F7 02 30 00
0850: 13 2E 3E 05 0A F7 0A 30 70 0A F7 10 30 02 30 72
0860: 2F 13 0D 3C 69 13 20 3C 65 13 3D 3C 61 13 2C 00
0870: 5D 13 2E 3C 59 13 2D 3C 55 13 47 3A 55 13 40 38
0880: 04 93 07 30 08 13 30 38 49 13 3A 3A 45 93 30 0A
0890: E5 0A 97 3A 3D 0A 27 0A E1 0A 29 22 F5 08 EC 0A
08A0: 27 0A E1 0A 2B 22 F5 08 13 00 3E 26 0A 21 0A F1
08B0: 00 E8 DC C0 0A 89 0A E9 C0 0A AB 0A EB 38 13 D8
08C0: C0 0A 85 C0 A3 00 E8 38 09 C5 23 B2 29 08 D8 04
08D0: 30 07 F0 00 00 14 F5 29 0C AC A8 A4 20 AE 25 23
08E0: 15 22 3A 0E 25 23 C0 0A F3 00 D2 20 FD EC 2F 04
```

In both examples, the default display block size is 256 bytes. You can also specify a beginning and ending address of the memory to be displayed:

```
>md 7f0-824
07F0: E6 E0 E5 10 A0 20 A2 2F 13 04 3C 08 22 E9 07 DC
0800: 80 EC 30 F3 A0 20 AE 25 23 15 22 3A 10 A2 25 23
0810: C0 0A F3 00 D2 20 FD EC A0 2F 04 30 01 14 AC 20
0820: A6 AA AE F8 00 00 0A F7 07 25 23 15 22 3A 2F 25
```

Each line displays a minimum of 16 address locations.

The **Modify Memory** (**MM**) command can be used to examine or modify single or multiple memory locations, as follows:

```
>mm fa00=e
>
```

In this example, the K0SMM responds with a prompt, indicating that the operation was a success. The next example examines location FA00 and displays the new value in hexadecimal and (decimal) formats:

```
>mm fa00
$0E (014)
```

This example shows how to write different values to a sequence of memory locations:

```
>mm fdd0=0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
>
```

The K0SMM responds with a prompt indicating that the operation was a success.

The next example, which starts at memory address FA00, modifies the memory locations with sequential data.

```
>md fdd0-fddf
FDD0: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
................
```

It is also possible to fill a block of memory with a single value using the **Memory Fill** (**MF**) command:

```
>mf fdd0-fdef=7f
```

To examine the memory just written:

```
>md fdd0-fdef
FDD0: 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
FDE0: 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F
```

As expected, all RAM locations now contain 7F.

To access the special-function registers by name, use the **Special Function** (**SF**) command:

```
>sf pm0
FF20,FF
```

PM0 is the port mode (data direction control) register for port 0. The returned values show the SFR's 16-bit address value as well as the contents of the specified SFR in hexadecimal format. The contents of most SFRs can be changed as follows:

```
>sf pm0=0f
FF20,
```

The SFR address is returned, but no data value is shown. As with all K0SMM write commands, the data is read back and verified. *NOTE: Some SFRs are write-only; data will be written to the SFR correctly, but a write error may be displayed since the contents could no be read back for verification.*

These commodities, technology or software, must be exported from the U.S. in accordance with the export administration regulations. Diversion contrary to U.S. law prohibited.

The information in this document is current as of September 2004. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.

Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC Electronics no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

NEC Electronics products are classified into the following three quality grades: “Standard”, “Special” and “Specific”.

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated “quality assurance program” for a specific application. The recommended applications of NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product 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 and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is “Standard” unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact NEC Electronics sales representative in advance to determine NEC Electronics 's willingness to support a given application.

(Notes)

(1) " NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

(2) " NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).