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

EV0338

Micro-Board for μPD78F0338 Microcontroller

Document no. U17182EU1V0UM00

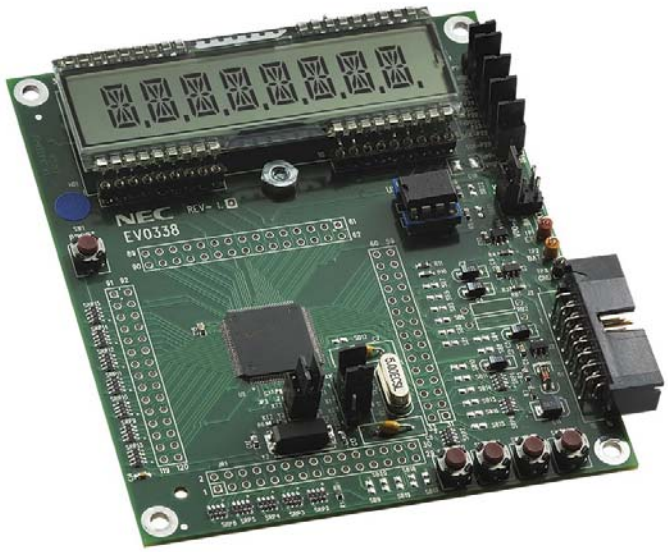
Contents

1. INTRODUCTION

The EV0338 micro-board is designed to enable you to demonstrate and evaluate the CPU and on-chip peripheral functions of the NEC Electronics 8-bit μPD78F0338 microcontroller (MCU). Out of the box, you can operate the EV0338 as a standalone piece to evaluate MCU features such as liquid crystal display (LCD) drivers and power consumption modes. In addition, you can also connect the micro-board to NEC Electronics America's M-Station tool (sold separately) to exercise the MCU's flash programming and source-level debugging features during system evaluation and development.

You can select various operating modes using the four pushbutton switches or by connecting to a host and communicating with a terminal emulation program such as HyperTerminal. These modes allow you to observe and investigate various features of the μPD780338 MCU. In addition, the C language code is provided for all of the utilities to help speed your own system development.

Figure 1. EV0338 Micro-Board



2. μPD78F0338 MICROCONTROLLER

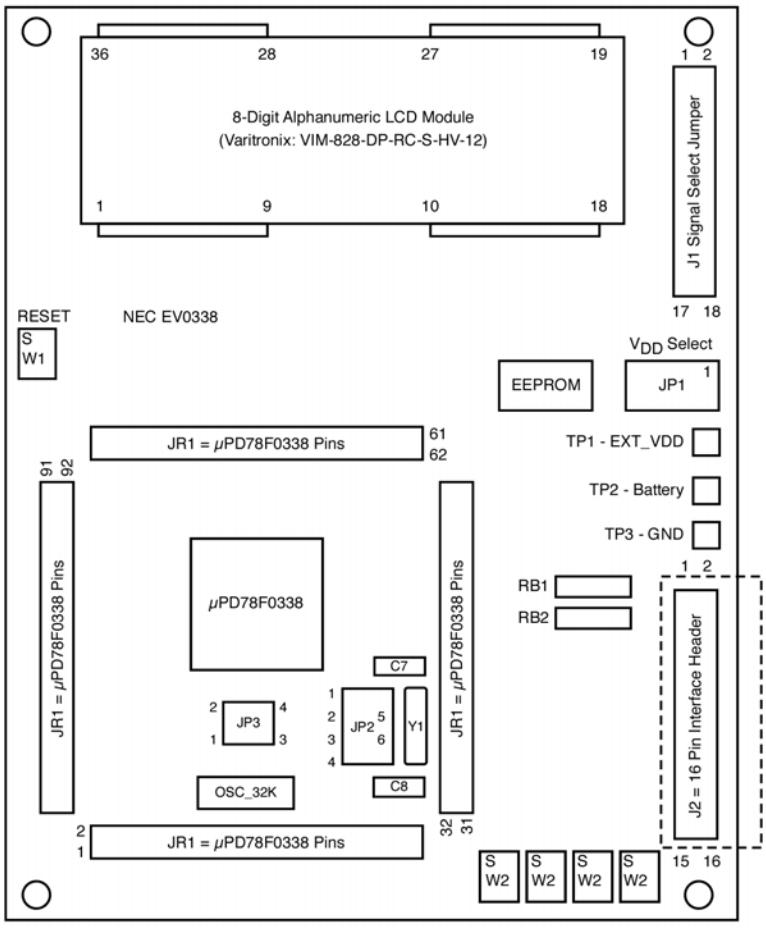
The EV0338's on-board μPD78F0338 microcontroller is fully explained in its user's manual (document no. U14701EJ3V0UD00). Its key features are listed below.

- Memory
  - 48 KB flash program memory
  - 1 KB high-speed data RAM
  - 1536-byte expansion RAM
- 40-segment LCD controller/driver
  - 40-byte LCD display memory
  - Booster-type LCD reference voltage
- 70 I/O ports
- Ten-channel, 10-bit A/D converter
- Serial interfaces
  - One-channel UART/3-wire serial I/O
  - One-channel 3-wire serial I/O
- Timers
  - Two-channel, 16-bit timer
  - Three-channel, 8-bit timer/event counter
  - One-channel watch timer
  - One-channel watchdog timer
- 25 vectored interrupts
- 120-pin TQFP (14 × 14 mm)
- 1.8–5.5V power supply voltage

3. EV0388 MICRO-BOARD

- Clock oscillators
  - 5 MHz main clock
  - 32 kHz subclock
- Port connector that provides all of the microcontroller's I/O pins
- Eight-digit alphanumeric LCD display module (Varitronix VIM-828-DP-RC-S-HV-12)
- Three AAA-size batteries for power supply voltage
- 2 KB data EEPROM
- 16-pin connector for debugging and flash programming interface
- Board size of 3.5 × 4 inches (W × L)

Figure 2. EV0338 Micro-Board Layout



04RC-0011B (05/04)

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

Table 1. Factory Jumper Settings

Jumper	Setting	Function	Description
JP1	3–4	V_{DD} select	Selects battery power as the default
JP2	2–5	Main clock select	Selects clock oscillator mounted on EV0338
	3–6	X1/X2 select	Supplies EV0338 with 5 MHz main clock frequency
JP3	1–2	Subclock select	Selects subclock oscillator mounted on EV0338
	3–4	XT1/ XT2 select	Fixes subclock frequency at 32 kHz
J1	3–4	V_{DD} to V_{DE} connect	Connects μPD78F0338's V_{DD} to V_{DE} for power detection
	5–6	Debugging/flash communication	Connects RxD of host interface to TxD of μPD78F0338 (P21)
	9–10	Debugging/flash communication	Connects TxD of host interface to RxD of μPD78F0338 (P20)
	17–18	V_{PP} select	Connects flash programmer's V_{PP} to μPD78F0338's V_{PP}

3.1 Operation

3.1.1 Power Up

When power is first applied to the EV0338 (or when it is reset after you press SW1), the LCD briefly turns on all segments and shows "EV0338" before starting operation in its default mode (stored at EEPROM location 58). At this time, the EV0338 also transmits the following text via the UART:

```
EV0338 MCU Evaluation Board
μPD780338 Mini-Apps + K0 Mini-Monitor
Copyright ©2004 NEC Electronics America, Inc.
Firmware version 1.00 May 24, 2004
```

3.1.2 Default Configuration

You can use the standalone EV0338 micro-board with its default settings to execute programs loaded into on-chip flash memory that demonstrate the CPU and its on-chip peripheral functions.

3.1.3 Debugging User Programs

1. Connect the EV0338 micro-board to the M-Station baseboard through the 16-pin header.
2. Keep the default jumper settings when using the UART port for the debugging or flash-programming interface.
3. Use the 16-pin header to supply power or else change the setting of pins 5–6 of JP1 to derive power from the M-Station baseboard.
4. Choose one of the following methods for debugging:
    - M-Station flash programming software, FP4MST, to reprogram the μPD78F0338's program memory.
    - M-Station terminal emulation software, MSTTERM, to communicate between the PC and the UART on the μPD78F0338 so that user programs can transmit and receive serial data
    - M-Station integrated debugging software, ID78K0-MST, to communicate with a monitor linked with the user program for real-time program debugging

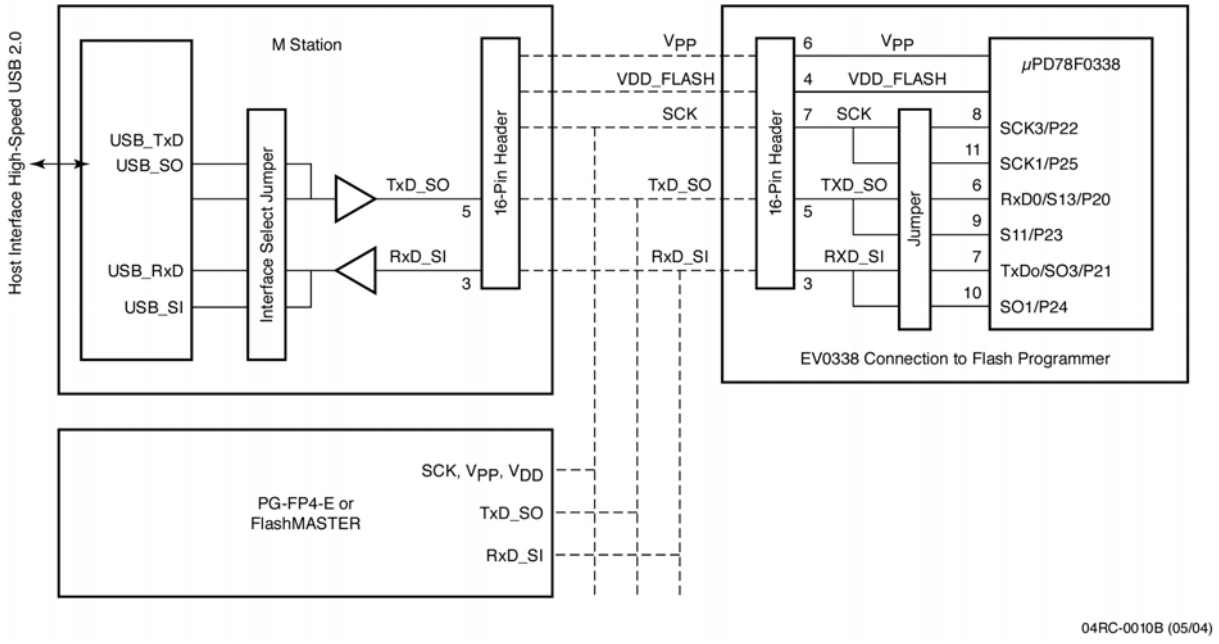
3.1.4 Reprogramming the Microcontroller's Flash Memory

The MCU's on-chip flash memory can be reprogrammed at any time using the built-in flash programming features of the M-Station or the PG-FP4-E flash programmer (available for purchase from your local NEC Electronics America representative). The micro-board's 16-pin debugging and flash programming interface header is pin-configured for use with either.

When reprogramming, select either the UART or serial I/O programming interface and then set the programming interface signal jumpers accordingly.

- Refer to sections 3.2.1 and 3.2.4 of this document before setting the micro-board's programming interface signals.
- 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.

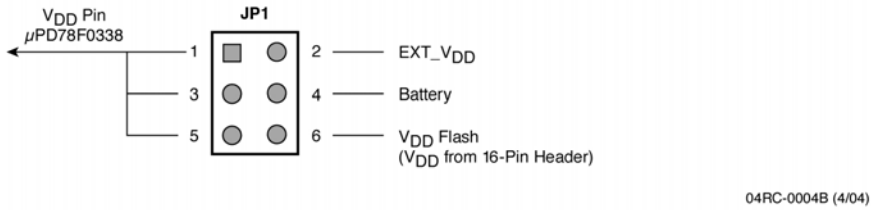
Figure 3. EV0338 Micro-Board Connection to Flash Programmer



3.1.5 Measuring the Microcontroller's Power Consumption

There are three ways to measure the µPD78F0338's power consumption on the EV0338 micro-board.

Figure 4. JP1 Pins



3.1.5.1 Battery-Supplied V_{DD} (Standalone Operation)

In this case, the EV0338 is not connected to the M-Station or other target boards, and JP1 is set on the BAT pins.

1. Make sure batteries are installed on the battery pack attached to the EV0338.
2. Remove the jumper from the BAT pins of JP1.
3. Connect an ampere meter to the BAT pins of JP1 to read the power consumption.

3.1.5.2 M-Station-Supplied V_{DD}

In this case, the EV0338 is connected to the M-Station through a 16-pin ribbon cable, and the jumper on JP1 should be set on the V_{DD} J2 pins.

1. To eliminate additional loads from the M-Station, remove the jumper from pins 3–4 of J1.
2. Remove the jumper from the V_{DD} J2 pins of JP1.
3. Connect an ampere meter to the V_{DD} J2 pins of JP1 to read the power consumption.

3.1.5.3 Externally Supplied V_{DD}

When an external source supplies V_{DD}, you can provide the power desired according to the specification defined in the μPD780338's data sheet.

1. Connect the external power supply.
2. Remove the jumper from the EXT pins of JP1.
3. Remove the jumper from pins 3–4 of J1 if the micro-board is connected to the M-Station.
4. Connect an ampere meter to the EXT pins of JP1 to read the power consumption.

Special Note About Measuring Power for the μPD78F0338

If EV0338 is connected to the M-Station for debugging, and J1 pin 3 to pin 4 is connected (default setting), the μPD78F0338 V_{DD} terminal is connected to V_{DE} by its default setting. The M-Station uses V_{DE} to bias the MCU's interface drivers and some application support circuits. In this case, the μPD78F0338's V_{DD} signal has additional load that will consume more power.

3.2 On-Board Components

3.2.1 16-Pin Debugging and Flash Programming Interface Header

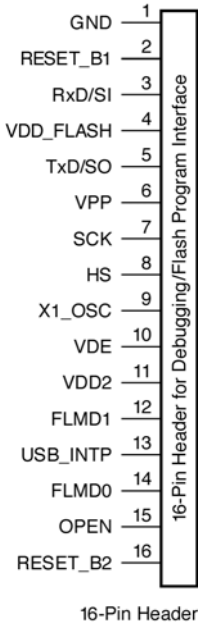
The 16-pin J2 header provides a debugging and flash-programming interface for the micro-board which, when connected to the M-Station using a 16-pin ribbon cable, supports debugging of application programs and programming of flash memory.

The 16-pin interface header is designed for compatibility with most NEC Electronics flash programmers, for example, the PG-FP4-E or the FlashMASTER. For confirmation, please refer to the appropriate user's manual before choosing a standalone programmer. Table 2 shows the signal assignments for connection.

Table 2. J2 Pin Assignments

Pin No.	Signal		Description
	PG-FP4	M-Station	
1	GND	GND	
2	RESET_B	RESET_B1	RESET_B1 reset for K0S real chip
3	RxD/SI	RxD/SI	Serial I/O or UART data receive
4	V_{DD}	V_{DD}_FLASH	
5	TxD/SO	TxD/SO	
6	V_{PP}	V_{PP}	
7	SCK	SCK	
8	HS	HS	
9	CLK	X1_OSC	
10	V_{DE}	V_{DE}	
11	V_{DD2}	V_{DD2}	
12	FLMD1	FLMD1	
13	RFU-1	USB_INTP	
14	FLMD0	FLMD0	
15	Not used	OPEN	
16	Not used	RESET_B2	

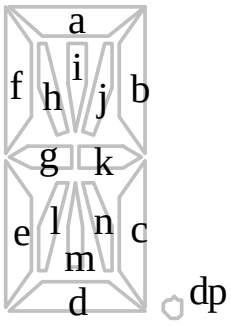
Figure 5. 16-Pin Header



3.2.2 Eight-Digit Alphanumeric LCD Module

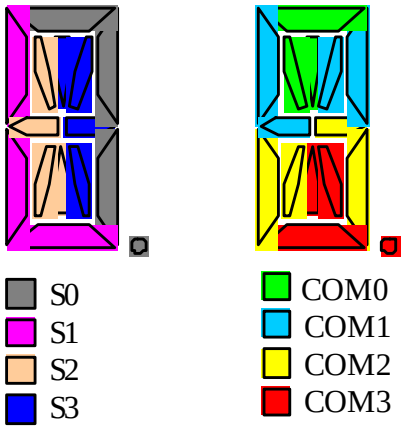
The EV0338 micro-board has an 8-digit alphanumeric LCD module that requires a minimum 4.5V for proper operation. The part used, Varitronix part no. VIM-828-DP-RC-S-HV-12, displays eight digits configured with fourteen segments plus a decimal point:

Figure 6. LCD Module



This arrangement of segments allows high-quality representation of letters and numbers. To drive the LCD module, the μPD78F0338 uses a combination of segment lines (S0–S31) and common lines (COM0–COM3). Figure 7 shows the connection of segment and common lines for the right-most digit.

Figure 7. Connection of Segment and Common Lines



The µPD78F0338 on-board LCD controller driver automatically drives waveforms on the common and segment lines to turn a desired segment in a digit on or off. Writing either a zero or a one into a bit of internal memory causes the appropriate segment to turn on (if the bit is 1) or off (if the bit is 0). Table 3 shows the correspondence of internal LCD memory bits to segments.

Table 3. Correspondence of Internal LCD Memory Bits to Segments

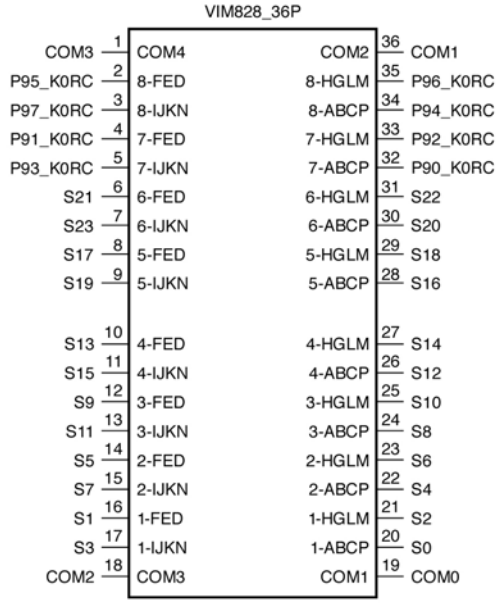
RAM ADDR	Bits							
	7	6	5	4	3	2	1	0
	Unused by Microcontroller				COM3	COM2	COM1	COM0
Seg+0	0	0	0	0	DP	C	B	A
Seg+1	0	0	0	0	D	E	F	—
Seg+2	0	0	0	0	M	L	G	H
Seg+3	0	0	0	0	N	K	J	I

Each of the eight digits in the display takes four RAM locations to specify whether the segments are on or off. Only the lower four bits of each location, the COM0–COM3 areas, are used; the address of the RAM location specifies the segment(s) to be driven. For example, to display the digit “8”, segments a, b, c, d, e, f, g, and k would be turned on, all other segments would be off, and the data to be stored in RAM would be as shown in Table 4.

Table 4. Data Stored in RAM

RAM ADDR	Bits								
	7	6	5	4	3	2	1	0	
	Unused by Microcontroller				COM3	COM2	COM1	COM0	
Seg+0	0	0	0	0	0	1	1	1	= 0x07
Seg+1	0	0	0	0	1	1	1	0	= 0x0E
Seg+2	0	0	0	0	0	0	1	0	= 0x02
Seg+3	0	0	0	0	0	1	0	0	= 0x04

Figure 8. Pinout of LCD Module



04RC-0007B (05/04)

3.2.3 RB1 and RB2 LCD Booster Biasing Resistors

The default for booster biasing is fixed at 5V to drive the micro-board's LCD module. If operation at a lower voltage is desired, you can change the setting as follows.

1. Disconnect SB6 and SB7 by cutting a thin trace between the two pads.
2. Refer to the *μPD78F0338 User's Manual* and then install new resistor values on RB1 and RB2.

It is advisable to replace the existing LCD module with one suitable for biasing at lower voltages. The module recommended, part no. VIM-828-DP-RC-S-LV-12, can be purchased directly from Varitronix.

3.2.4 J1 Interface Select Jumper

Use J1 to configure the debugging and flash-programming interface signals on the 16-pin J2 header with the corresponding pins of the μPD78F0338.

Figure 9. EV0338 Micro-Board Layout

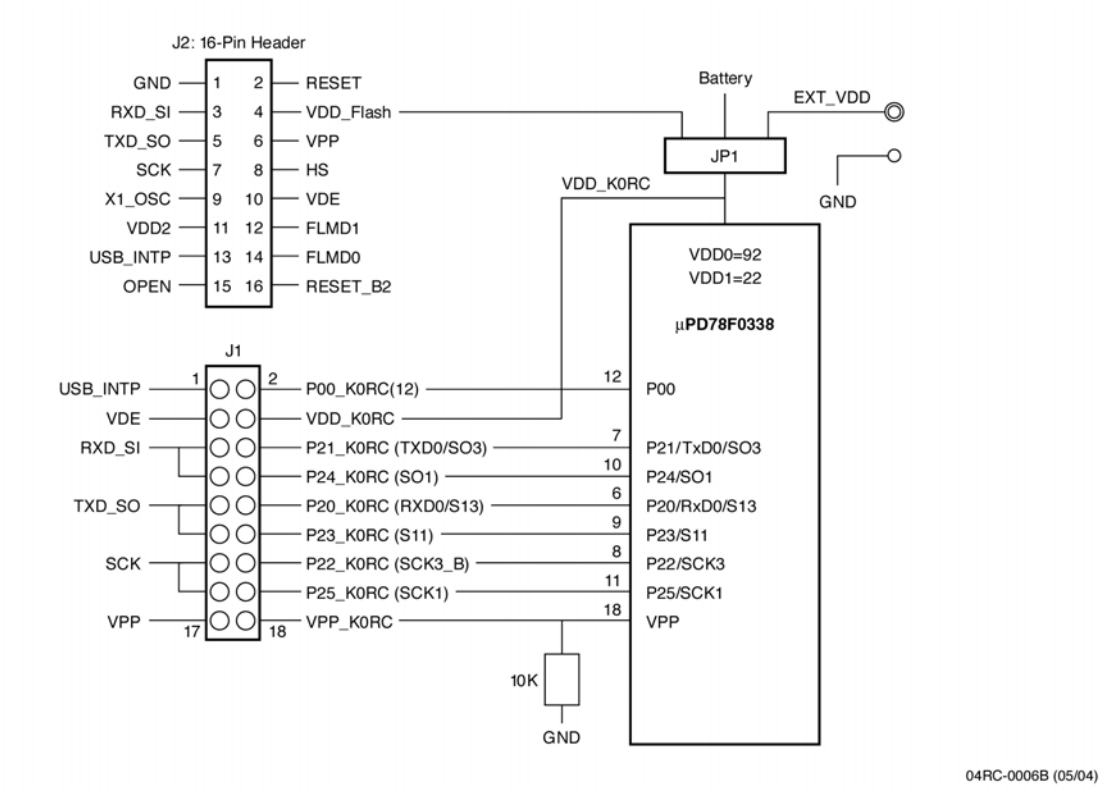


Table 5. J1 Settings

Pins	Interface Signal	Default Setting
1– 2	Reserved for future use	Open
3– 4	Connects EV0338 V_{DD} to V_{DE} terminal on M-Station	Installed
5– 6	Connects RxD/SI from M-Station to TxD/SO3 of μPD78F0338	Installed
7– 8	Connects RxD/SI from M-Station to SO1 (port 2.4) of μPD78F0338	
9– 10	Connects TxD/SO from M-Station to RxD/SO3 of μPD78F0338	Installed
11– 12	Connects TxD/SO from M-Station to SI1 (port 2.3) of μPD78F0338	
13– 14	Connects SCK from M-Station to SCK3_B (port 2.2) of μPD78F0338	
15– 16	Connects SCK from M-Station to SCK1 (port 2.5) of μPD78F0338	
17– 18	Connects V_{PP} from M-Station to V_{PP} (pin 18) of μPD78F0338	Installed

Example settings when the EV0338 is connected to the M-Station are explained in the following subsections.

3.2.4.1 UART as Communication Interface or Flash-Programming Interface

1. Select UART interface on JP8 and JP9 on M-Station (refer to the *M-Station User's Manual*).
2. Use J1 5–6 to connect RxD/SI from the M-Station to TxD/SO3 of the μPD78F0338.
3. Use J1 9–10 to connect TxD/SO from the M-Station to RxD/SI3 of the μPD78F0338.

3.2.4.2 Serial I/O3 as Flash Programming Interface

1. Select serial I/O interface on JP8 and JP9 on the M-Station (refer to the *M-Station User's Manual*).
2. Use J1 5–6 to connect RxD/SI from the M-Station to TxD/SO3 of the μPD78F0338.
3. Use J1 9–10 to connect TxD/SO from the M-Station to RxD/SI3 of the μPD78F0338.
4. Use J1 13–14 to connect SCK from the M-Station to SCK3_B of the μPD78F0338.

3.2.4.3 Serial I/O1 as Flash Programming Interface

1. Select serial I/O interface on JP8 and JP9 on the M-Station (refer to the *M-Station User's Manual*).
2. Use J1 7–8 to connect RxD/SI from the M-Station to SO1 of the μPD78F0338.
3. Use J1 11–12 to connect TxD/SO from the M-Station to SI1 of the μPD78F0338.
4. Use J1 15–16 to connect SCK from the M-Station to SCK1 of the μPD78F0338.

3.2.5 JP1: V_{DD} Select

V_{DD} on the EV0338 micro-board can be supplied in one of three ways:

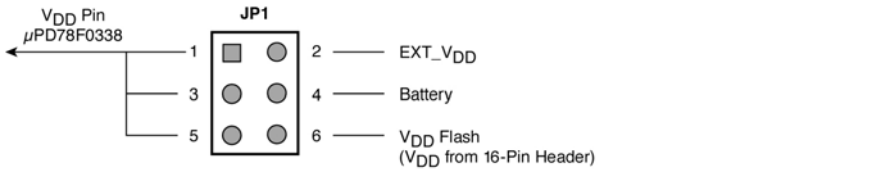
- Three AAA-size batteries mounted on the battery holder
- User-supplied external source (EXT_VDD) between 1.8V to 5.5V
- 16-pin J2 header connected to the M-Station baseboard

When displaying data on the LCD module, the EV0338 micro-board requires V_{DD} to be 5V ±5%.

Table 6. JP1 Settings

Setting	Source	V_{DD}	Note
1 to 2	User-supplied external power	EXT_VDD	
3 to 4	Battery-supplied power	Battery	Default
5 to 6	M-Station baseboard-supplied power	VDD_FLASH	

Figure 10. V_{DD} Select



04RC-0004B (4/04)

3.2.6 EEPROM

The micro-board's 2 KB EEPROM is used to store application data for your target program. If you need more than 2 KB, remove the EEPROM from its 8-pin DIP socket and then insert a replacement. EEPROM of varying sizes can be purchased from DigiKey and changed as needed.

1. Use port 3[1:0] of the μPD78F0338 for the EEPROM's SCL and SDA inputs.
2. Pull the SCL and SDA lines up to V_{DD} with 10 kΩ resistors to enable read and write operations up to 10 kHz.

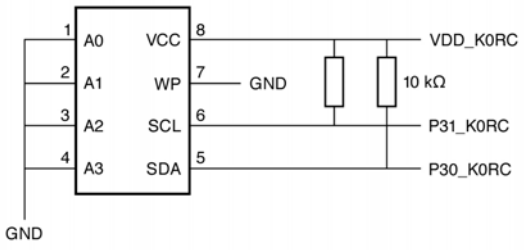
Note that the on-board EEPROM is organized as eight blocks of 256 × 8-bit memory. The sample code provided by NEC Electronics America uses a software-implemented I^2C interface to read and write to the first half (0–127) of the first EEPROM block (BLOCK0). You can extend the EEPROM software routines easily to write to all 256 bytes of all eight blocks if desired. The 128 EEPROM locations used by the EV0038 micro-board are listed in Table 7.

Table 7. EEPROM Locations

Decimal Address	Description
0–48	User message text string
49–57	Name badge 1 text string
58	Default mode number
59	Time format
60	Time display duration
61	Date format
62	Date display duration
63	Display scroll speed
64–127	Unused

Changes to these values, or to other unused EEPROM addresses, can be made in mode 15, EV0338 mini-monitor mode.

Figure 11. Aries A400 Socket (8-Pin DIP)



04RC-0016B (5/04)

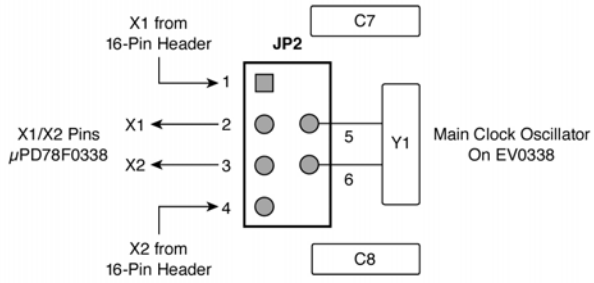
3.2.7 JP2: Main Clock Selection

The 5-MHz crystal oscillator (Y1) mounted on the EV0338 micro-board serves as its main clock. Biasing capacitors (C7 and C8) also are mounted with through-hole parts. To use a different oscillator with another frequency, first remove the existing 5 MHz oscillator and biasing capacitors and then mount the new ones. *Please consult the specification for your chosen oscillator before installing new biasing capacitors*.

Table 8. JP2 Settings

Setting	Clock Source	X1 Clock	Note
2 to 5 3 to 6	From EV0338 micro-board (must be selected in pairs)	X1 = 5 MHz	Default
2 to 1 3 to 4	From M-Station baseboard (must be selected in pairs)	Frequency determined by clock module	

Figure 12. Main Clock Selection



04RC-0005B (4/04)

The default configuration of the EV0338 connects pins 2–5 and 3–6 of JP2 to the on-board 5 MHz clock. To operate the micro-board at a frequency supplied from the oscillator on the M-Station baseboard:

1. Connect JP2 pins 1–2 and 3–4 to use the external clock from the 16-pin header.
2. Connect the X1 and X2 pins of the μPD78F0338 to the I/O pin array.
3. Provide external connection to the appropriate pins of the I/O pin array.
4. Keep all pins of JP2 open.

3.2.8 JP3: Subclock Selection

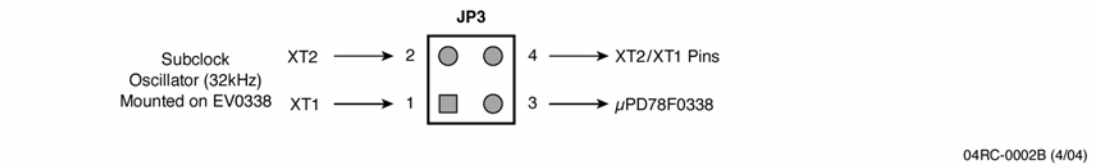
Use JP3 to select the on-board 32 kHz subclock (default) or, when the micro-board is mounted on the target, an externally supplied subclock source. In the first case, connect pins 1–3 and 2–4 of JP3. In the latter case,

1. Connect pins 3 and 4 of JP3 to JR1.
2. Connect the µPD78F0338 pins to the target.
3. Connect the subclock from the target to the µPD78F0338.
4. Keep all pins of JP3 open.

Table 9. JP3 Settings

Setting	Subclock Source	XT1 Clock	Note
1 to 2 3 to 4	From EV0338 (must be selected in pairs)	XT1 = 32 kHz	Default
Removed	User-supplied (connected via JR1)	User-determined frequency	

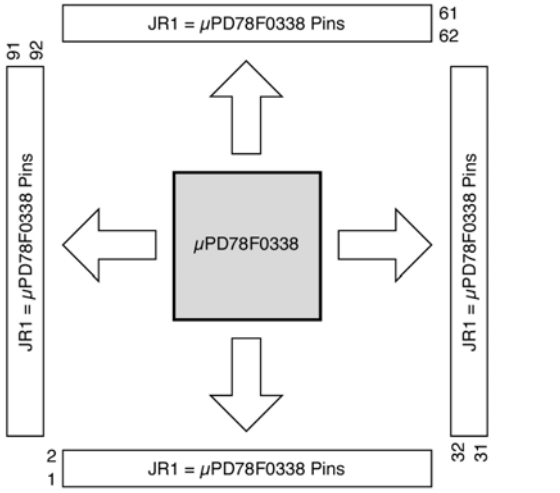
Figure 13. Subclock Selection



3.2.9 JR1 I/O Pin Array

JR1 is a pin array that gives you easy access to the µPD78F0338's I/O pins or enables you to connect the micro-board directly to your target system. JR1 is divided into four 30-pin segments positioned around the µPD78F0338.

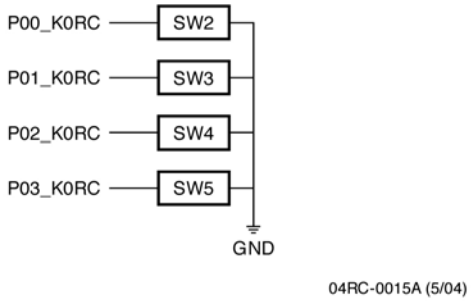
Figure 14. JR1 I/O Pin Array



3.2.10 SW1–SW3 Switches

SW1 is a reset switch. Pushing it connects the RESET/ pin of the μPD78F0338 to GND and resets the CPU. SW2–SW5 are pushbutton switches connected to the lower four bits of the MCU's port 0[3:0]. The other sides of these pushbutton switches are connected to GND. Application programs may read the state of these switches by reading port P0 and then examining the state of the lower four bits. P0 in the μPD78F0338 has optional pull-up resistors that you may connect to the P00–P03 lines by writing ones into the corresponding bits of the port 0 pull-up register, PU0.

Figure 15. SW1–SW3 Switches



4. EV0338 SOFTWARE

Each of the operating modes has unique software functions, a number of which are common to most routines. These are detailed here rather than repeated in the “Operating Modes” section.

4.1 Display Functions

The micro-board’s LCD can display eight characters. The character positions are labeled 8 to 1, going from left to right. To display a character on the LCD, you need to specify which character and at what position using the Display_Character() function.

4.1.1 * Function: Display_Character

```
* Parameters:    Character to be displayed
                 LCD position to display character
* Returns:       Nothing
* Example:       Display_Character('A',4);
```

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

4.1.2 * Function: Display_String

```
* Parameters:    Pointer to string
                 LCD position to start display of string
* Returns:       Nothing
* Example:       Display_String("string",8);
```

For strings longer than eight characters, you must scroll across the display using the Scroll_ROM_String() function(if the string is stored in ROM) or the Scroll_EEPROM_String() function (if the string is stored in EEPROM). To shift the displayed string one character to the left, use the TM4_Interrupt() routine, which calls the Scroll_Step_ROM_String() or Scroll_Step_EEPROM_String() function to shift the displayed message.

4.1.3 * Function: Scroll_ROM_String

```
* Parameters:   Pointer to first character
* Returns:      Nothing
* Example:      Scroll_ROM_String(pointer_to_string)
```

4.1.4 * Function: Scroll_EEPROM_String

```
* Parameters:   Address of starting byte
* Returns:      Nothing
* Example:      Scroll_EEPROM_String(ee_string_address)
```

4.1.5 * Function: TM4_Interrupt

```
* Parameters:   None
* Returns:      Nothing
```

4.1.6 * Function: Scroll_Step_ROM_String

```
* Parameters:   None
* Returns:      Nothing
```

4.1.7 * Function: Scroll_Step_EEPROM_String

```
* Parameters:   None
* Returns:      Nothing
```

4.2 Switch Functions

The EV0338 has four switches labeled SW2, SW3, SW4 and SW5. The effect of these switches is generally dependent on the current operating mode. However, there are three functions that generate a consistent result irrespective of mode.

The Switch_Interrupt() routine initializes Timer50 as a debounce timer and the TM50_Interrupt() routine can be used to debounce the switches and set status flags. You can also press SW3 + SW4 in combination to view or change the operating mode.

4.2.1 * Function: Switch_Interrupt

```
* Parameters:    None
* Returns:       Nothing
```

4.2.2 * Function: TM50_Interrupt

```
* Parameters:    None
* Returns: Nothing
```

4.2.3 SW3 + SW4 Pressed Simultaneously (Except in Mode 15: Mini-Monitor Mode)

Pressing SW3 + SW4 simultaneously causes the EV0338 micro-board to show the Select Mode display, from which you can view the current mode (MODE XX) or step through and select another one.

1. Pressing SW2 advances through the list of modes, one by one with each press.
2. Pressing SW3 steps backward through the list of modes, one by one with each press.
3. Pressing SW5 at any stage selects the displayed mode.

If you don’t press a switch within five seconds, the mode reverts to the one set before you pressed the SW3 + SW4 combination.

4.3 EEPROM Functions

The 2 KB EEPROM on the EV0338 board is used to store various information and parameters. The μPD780338 uses a software-implemented I^2C interface to read and write to the EEPROM locations described in Table 10.

Table 10. EEPROM Locations

Decimal Address	Description
0–48	User message text string
49–57	Name badge 1 text string
58	Default mode number
59	Time format
60	Time display duration
61	Date format
62	Date display duration
63	Display scroll speed

Changes to these values, or to other unused EEPROM addresses, can be made in mode 15 (EV0338 mini-monitor mode). Use the EE_Read_Byte() and EE_Write_Byte() functions to read and write any byte address. Use the EE_WriteString() function to write a string.

4.3.1 * Function: EE_ReadByte

```
* Parameters:    Address of byte to be read
* Returns:       Byte read from eeprom address,
                 0 if unsuccessful
* Example:       ee_data=EE_ReadByte (ee_address);
```

4.3.2 * Function: EE_WriteByte

```
* Parameters:    byte to be written
                 address to write byte to
* Returns:       nothing
* Example:       EE_WriteByte(ee_data,ee_address);
```

4.3.3 * Function: EE_WriteString

```
* Parameters:    Pointer to string
                 Starting address to write string to
* Returns:       Nothing
* Example:       sptr=”eeprom string”;
                 sadd=STORE_STRING_FROM_ADDRESS;
                 EE_WriteString(sptr, sadd);
```

4.4 Timekeeping Functions

The MCU’s watch timer peripheral is initialized to run off the 32.768 kHz subsystem clock and to generate a 0.5-second tick used by the Watch_Timer_Interrupt function to update the real-time clock, calendar date, elapsed time counter, and countdown timer.

```
* Function:      Watch_Timer_Interrupt
* Parameters:    None
* Returns:       Nothing
```

4.5 UART Functions

The MCU’s asynchronous UART runs at 9600 baud, 8 data bits, no parity and 1 stop bit to communicate with a terminal program in mode 15. Use the Rx_Character() and Tx_Character() functions to receive or send a character and the Tx_String() function to send or receive a string.

4.5.1 * Function: Rx_Character

```
* Parameters:    None
* Returns:       Received character
* Example:       my_char=Rx_Character();
```

4.5.2 * Function: Tx_Character

```
* Parameters:    Character to be transmitted
* Returns:       Nothing
* Example:       my_char=’M’
                 Tx_Character(my_char);
```

4.5.3 * Function: Tx_String

```
* Parameters:    Pointer to string
* Returns:       Nothing
* Example:       tx_ptr="transmit string";
                 Tx_String(tx_ptr);
```

5. OPERATING MODES

Table 11 lists the EV0338's operating modes along with a brief description of each.

Table 11. Description of Operating Modes

Mode	Name	Description
1	Current measurement 1	CPU main clock operating at full speed
2	Current measurement 2	CPU main clock operating at quarter speed
3	Current measurement 3	CPU halted but main clock operating
4	Current measurement 4	CPU subclock operating and main clock stopped
5	Current measurement 5	CPU halted but subclock operating
6	Clock/calendar	Alternates display of time and date
7	Clock/calendar with scrolling ROM message	Alternates display of time, date, and scrolling ROM message
8	Clock/calendar with scrolling EEPROM message	Alternates display of time, date, and scrolling EEPROM message
9	Name badge 1	Statically displays 8-character user message
10	Name badge 2	Scrolls through the user message
11	Pseudo-random number generator	Displays a new pseudo-random number on each switch press
12	Four-digit combination lock	Sets the combination and lock/unlock sequence
13	Totalizer	Displays the number of switch presses
14	Oracle	Simulates an "eight-ball"
15	EV0338 mini-monitor	Accepts commands and displays information via serial interface

5.1 Mode 1: Current Measurement 1

This mode sets the CPU to operate at the full, undivided clock speed of Fx.

5.2 Mode 2: Current Measurement 2

This mode sets the CPU to operate at the quarter speed of Fx/4.

5.3 Mode 3: Current Measurement 3

This mode halts the clock to the CPU but keeps the main clock running.

5.4 Mode 4: Current Measurement 4

This mode stops the main clock and forces the CPU to operate off the 32.768 kHz subclock.

5.5 Mode 5: Current Measurement 5

This mode halts the subclock to the CPU but keeps the subclock running. In this mode, you can check the MCU's clock speed using an oscilloscope on P0.0 that is toggling every 14 execution cycles. Note that the MCU's 32 kHz subclock is always running and there is no provision for stopping it. Pressing the **RESET** button is the only way to exit.

5.6 Mode 6: Clock/Calendar

This mode switches the display between the time and date.

- It displays the time as a 12-hour clock (HH-MM AM/PM) or 24-hour clock (HH-MM-SS).
- It displays the date in U.S. format (MM-DD-YY) or non-U.S. format (DD-MM-YY).

The time and date are continuously updated every one-half second in the Watch_Timer_Interrupt() routine. The Display_Date() function displays the date and the Display_Time() function the time. The durations of the time and date displays are set by the values in EEPROM locations 60 and 62, respectively, and counted down using the 0.5-second tick from the Watch_Timer_Interrupt() routine.

5.6.1 * Function: Display_Time

```
* Parameters:    time format
* Returns:       Nothing
* Example:       Display_Time (12-hour clock);
```

5.6.2 * Function: Display_Date

```
* Parameters:    date format
* Returns:       Nothing
* Example:       Display_Date (US date format);
```

5.6.3 Setting the Time

Pressing SW2 while the time is being displayed toggles between the 12-hour clock (HH-MM AM/PM) and the 24-hour clock (HH-MM-SS) formats. The display briefly shows “12-HOUR” or “24-HOUR” to show which format it is changing to. The time format chosen will be used for all modes where the time is displayed.

5.6.4 Setting the Clock

Pressing SW3 while the clock is being displayed enters clock-setting mode, during which either the hours, minutes or seconds digits flashes to indicate that setting is in progress.

- SW2 advances the flashing hours, minutes, or seconds
- SW3 puts the unit is clock-setting mode
- SW4
- SW5 selects the displayed hours, minutes, or seconds
  - If the hours are flashing, pressing SW5 selects the hours and flashes the minutes.
  - If the minutes are flashing, pressing SW5 selects the minutes and flashes seconds.
  - If the seconds are flashing, pressing SW5 selects the seconds and exits the session.

If you don’t press a switch within five seconds, the unit exits the clock-setting session and retains the settings displayed.

The SetTimeUsingSwitches() function sets the time and calls the Flashing_On_For_Character() and Flashing_Off_For_Character() functions to indicate that setting is in progress. Use the Display_Time() function to refresh the display.

5.6.4.1 * Function: SetTimeUsingSwitches

```
* Parameters:    None
* Returns:       Nothing
* Example:       SetTimeUsingSwitches();
```

5.6.4.2 * Function: Flashing_On_For_Character

```
* Parameters:    Position of character on LCD
* Returns:       Nothing
* Example:       Flashing_On_For_Character(7);
```

5.6.4.3 * Function: Flashing_Off_For_Character

```
* Parameters:    Position of character on LCD
* Returns:       Nothing
* Example:       Flashing_Off_For_Character(7);
```

5.6.4.4 * Function: Display_Time

```
* Parameters:    Time format
* Returns:       Nothing
* Example:       Display_Time (12-hour clock);
```

5.6.5 Setting the Date Format

Pressing SW2 while the date is being displayed toggles between the U.S. (MM-DD-YY) and non-U.S. (DD-MM-YY) formats. The display briefly shows “MM-DD-YY” or “DD-MM-YY” to show which format it is changing to. The date format chosen will be used for all modes where the date is displayed.

5.6.6 Setting the Calendar

Pressing SW3 while the calendar is being displayed enters calendar-setting mode, during which either the hours, minutes or seconds values will be flashing to indicate that setting is in progress.

- SW2: advances flashing month/day/year
- SW3: puts unit is calendar setting mode
- SW4:
- SW5: selects displayed month/day/year
  - If the month is flashing, pressing SW5 selects the month and flashes the day.
  - If the day is flashing, pressing SW5 selects the day and flashes the year.
  - If the year is flashing, pressing SW5 selects the year and exits the session.

If you don’t press a switch within five seconds, then the unit exits the calendar-setting session and uses the values displayed as the new date.

The SetDateUsingSwitches() function sets the date and calls the Flashing_On_For_Character() and Flashing_Off_For_Character() functions to indicate that setting is in progress. Use the Display_Date() function to refresh the display.

5.6.6.1 * Function: SetDateUsingSwitches

```
* Parameters:    None
* Returns:       Nothing
* Example:       SetDateUsingSwitches();
```

5.6.6.2 * Function: Flashing_On_For_Character

```
* Parameters:    Position of character on LCD
* Returns:       Nothing
* Example:       Flashing_On_For_Character(7);
```

5.6.6.3 * Function: Flashing_Off_For_Character

```
* Parameters:    Position of character on LCD
* Returns:       Nothing
* Example:       Flashing_Off_For_Character(7);
```

5.6.6.4 * Function: Display_Date

```
* Parameters:    Date format
* Returns:       Nothing
* Example:       Display_Date (US date format);
```

5.7 Mode 7: Clock/Calendar with scrolling ROM message

This mode is similar to mode 6, but it alternates the display of the clock and calendar with a scrolling message stored in ROM. The default message is “NEC ELECTRONICS AMERICA, INC. *** EV0338 EVALUATION BOARD ***”. Note that the display scroll speed is set by the value stored at EEPROM location 63.

Use the Display_Time() and Display_Date() functions to display the time and date, respectively, and the Scroll_ROM_String() function to scroll through the stored message.

5.7.1 * Function: Display_Time

```
* Parameters:    Time format
```

```
* Returns:       Nothing
* Example:       Display_Time (12-hour clock);
```

5.7.2 * Function: Display_Date

```
* Parameters:    Date format
* Returns:       Nothing
* Example:       Display_Date(US date format);
```

5.7.3 * Function: Scroll_ROM_String

```
* Parameters:    Address of starting byte
* Returns:       Nothing
* Example:       Scroll_ROM_String(pointer_to_string)
```

5.8 Mode 8: Clock/Calendar with scrolling EEPROM message

This mode is similar to mode 6, except that it alternates the display of the clock and calendar with the scrolling of a user message stored in EEPROM locations 0–48. The default message is “YOUR MESSAGE CAN BE STORED HERE – UP TO 48 CHARS”. For information about how to store your own message, please see the **UM** command for mode 15.

Use the Display_Time() and Display_Date() functions to display the time and date, respectively, and the Scroll_EEPROM_String() function to scroll through the stored message.

5.8.1 * Function: Display_Time

```
* Parameters:    Time format
* Returns:       Nothing
* Example:       Display_Time (12-hour clock);
```

5.8.2 * Function: Display_Date

```
* Parameters:    Date format
* Returns:       Nothing
* Example:       Display_Date (US date format);
```

5.8.3 * Function: Scroll_EEPROM_String

```
* Parameters:    Address of starting byte
* Returns:       Nothing
* Example:       Scroll_EEPROM_String(ee_string_address)
```

5.9 Mode 9: Name Badge 1

This mode statically displays an eight-character message. The default message stored in EEPROM locations 49–57 is “MY NAME ”. For information about how to store your own message, please see the **NB** command in mode 15.

Use the EE_ReadByte() function to read the eight characters from the EEPROM, starting at address EE_NAME1_ADDRESS, and the Display_Character() function to write the characters to the LCD.

5.9.1 * Function: EE_ReadByte

```
* Parameters:    Address of byte to be read
* Returns:       Byte read from eeprom address,
                 0 if unsuccessful
* Example:       ee_data=EE_ReadByte(ee_address);
```

5.9.2 * Function: Display_Character

```
* Parameters:    Character for display
*               Position of character
* Returns:       Nothing
* Example:       Display_Character('A',4);
```

5.10 Mode 10: Name Badge 2

This mode displays a scrolling message of up to 48 characters stored in EEPROM locations 0–48. The default message is “YOUR MESSAGE CAN BE STORED HERE – UP TO 48 CHARS”. For information about how to store your own message, please see the **UM** command in mode 15.

Use the EE_ReadByte() function to read the message characters from EEPROM, starting at address EE_NAME1_ADDRESS, and the Scroll_EEPROM_String (ee_string_address) function to scroll through the message.

5.10.1 * Function: EE_ReadByte

```
* Parameters:    Address of byte to be read
* Returns:       Byte read from eeprom address,
                 0 if unsuccessful
* Example:       ee_data=EE_ReadByte(ee_address);
```

5.10.2 * Function: Scroll_EEPROM_String

```
* Parameters:    Address of starting byte
* Returns:       Nothing
* Example:       Scroll_EEPROM_String(ee_string_address)
```

5.11 Mode 11: Pseudo-Random Number Generator

This mode displays a four-digit 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 random number and the WordToDec() utility to convert the random number to a decimal character string. To display the string, use the Display_String() function.

5.11.1 * Function: rand()

```
* Parameters:    None
* Returns:       Random integer
* Example:       pseudo_int=rand()
```

5.11.2 * Function: WordToDecString

```
* Parameters:    Integer, include leading zeroes
* Returns:       Array of decimal characters
```

```
* Example:       WordToDec(0x03ff, false);
                 (result stored in global array)
```

5.11.3 * Function: Display_String

```
* Parameters:    Pointer to string
                 LCD position to start display of string
* Returns:       Nothing
* Example:       Display_String("string",8);
```

5.12 Mode 12: Four-Digit Combination Lock

This mode initially displays "SET **** " to indicate that no combination has been set yet. Pressing any switch causes the display to go to "SET 0000", indicating that is it ready for you to set the combination.

5.12.1 Setting the Combination

With "SET 0000" displayed, use SW2–SW5 to set the combination, as shown below.

- Press SW2 to increment the first digit by one.
- Press SW3 to increment the second digit by one.
- Press SW4 to increment the third digit by one.
- Press SW5 to increment the fourth digit by one.

Press SW2 + SW5 together to select the displayed four-digit number as the lock combination. The unit acknowledges the setting by briefly displaying "COMB SET" and then "ULK 0000" to indicate the unlocked state.

5.12.2 Locking/Unlocking

If the correct combination has been selected, then pressing SW2 + SW5 simultaneously will toggle between the locked and unlocked states. P5.0 pulses low for 0.5 second as an example of an action that could be used to drive a locking/unlocking solenoid.

When the unit enters the unlocked state, it displays "ULK 0000". When it enters the locked state, the unit briefly displays "LK ****" and reverts to "LK 0000", indicating that it is ready for you to enter the combination.

Use the Lock() and Unlock() functions lock and unlock the combination lock, and the Display_Character() and Display_String() functions to display a character or string.

5.12.2.1 * Function: Lock()

```
* Parameters:    None
* Returns:       Nothing
* Example:       if(door_closed) Lock();
```

5.12.2.2 * Function: Unlock()

```
* Parameters:    None
* Returns:       Nothing
* Example:       if(correct_combination) Unlock();
```

5.12.2.3 * Function: Display_Character

```
* Parameters:    Character
                 LCD position to display character
* Returns:       Nothing
* Example:       Display_Character('1',4);
```

5.12.2.4 * Function: Display_String

```
* Parameters:    Pointer to string
                 LCD position to start display of string
* Returns:       Nothing
* Example:       Display_String("string",8);
```

5.13 Mode 13: Totalizer

This mode initially displays a count of zero.

- Pressing SW2 increments the displayed count by one.
- Pressing SW3 decrements the count.
- Pressing SW4 sets the count to the maximum (65,535).
- Pressing SW5 resets the count to zero (minimum).

Use the Display_Character() and Display_String() functions to display a character or string.

5.13.1 * Function: Display_Character

```
* Parameters:    Character
                 LCD position to display character
* Returns:       Nothing
* Example:       Display_Character('0',8);
```

5.13.2 * Function: Display_String

```
* Parameters:    Pointer to string
                 LCD position to start display of string
* Returns:       Nothing
* Example:       Display_String("string",8);
```

5.14 Mode 14: Oracle

This mode displays “ASK SW2”. You then ask a question of the oracle and press SW2 to receive its reply. Short replies are displayed statically for a time; longer replies are scrolled through. After replying, the oracle returns to “ASK SW2” mode.

The stdlib.h library rand() function is used to generate a pseudo-random number that is then used to randomly select an answer from among a set of replies.

```
* Function:      rand()
* Parameters:    None
* Returns:       Random integer
* Example:       reply_index=rand()
```

5.15 Mode 15: EV0338 Mini-Monitor (K0M0338)

The K0M0338 is designed to be a useful tool for “bringing up” target hardware the first time. With little or no modification, you can use a K0M0338-programmed MCU to exercise the target hardware immediately, or test a board’s hardware functionality. The ROM used by the monitor code includes a number of potentially useful C language functions—serial receive, numeric entry via serial port, binary-to-ASCII decimal conversion, string output, etc.—that are available for customer use.

The K0M0338 uses standard asynchronous serial communication (9600 baud, no parity, eight data bits) to communicate with a host computer running a terminal emulation program such as HyperTerminal. It does not require a complex GUI running on the host. Please see Appendix A for instructions about how to set up the HyperTerminal software to communicate with the EV0338.

5.15.1 K0M0338 User Interface Basics

The command interface of the K0M0338 allows you to execute commands that are not so convenient for executing under switch control. Examples of such commands are setting a bit in a given memory address, storing a user message for scrolling, and so forth. The command interface is built around two-letter command codes as shown in Table 12 below.

Table 12. Command Descriptions

Command	Parameters	Description
CE		Toggle serial echo
CM	n	Current measurement mode (n=1 to 5)
CR		Return copyright notice
DH	addr1 [-addr2]	Dump memory block as Intel hex
ED		Display first 128 bytes of EEPROM
EE	addr [=byte]	Read/write EEPROM location addr
ET	[=]	Read elapsed time clock (= to reset to zero)
FE	[=]	Verify factory EEPROM values (= to format)
FV		Return firmware version information
HE		Display list of available commands
MB	addr.bit [=0] [=1]	Modify bit (0-7) at location addr
MD	addr1 [-addr2]	Display memory from addr1 to addr2
MF	addr1 [-addr2] =byte	Fill memory with byte from addr1 to addr2
MM	Addr [=byte][byte2,byte3….]	Fill memory with sequential bytes starting at addr
NE		Numeric entry test
RC		Display four-digit hexadecimal ROM checksum
RD	[=mm/dd/yy]	Read/set real-time date
RT	[=hh:mm:ss]	Read/set real-time clock
SF	?	Display list of SFR names
	Sfr [=byte/word]	Read/write SFR location by name
SM	[=n]	Set start-up mode = mode n (n=0–14)
UM	[=string]	Read/set user scrolling message
UN	[=string]	Read/set user name message
XT		Exit monitor mode

5.15.2 CE

This command toggles the serial character echo between on and off. The default is echo on, which displays any character you type in your terminal program because the K0M0338 echoes it back.

5.15.3 CM n

This command causes the unit to enter current measurement mode n (n=1–5), which enables you to measure operating current in the given mode. Note that the UART is turned off for current measurements so you cannot exit these modes without resetting the unit.

- CM1: CPU full-speed (fx)
- CM2: CPU quarter speed (fx/4)
- CM3: CPU halted (main clock still running)
- CM4: CPU running on 32.768 kHz subclock (main clock stopped)
- CM5: CPU halted (subclock still running)

In this mode, you can measure the MCU's clock speed using an oscilloscope on P0.0 that is toggling every 14 execution cycles. (Note that the MCU's 32 kHz subclock oscillator is always running and there is no provision for stopping it unless you press the **RESET** button.)

5.15.4 CR

This command causes the EV0338 to return the copyright notice in the form "Copyright © 2004 NEC ELECTRONICS AMERICA, INC."

5.15.5 DH addr1 [-addr2]

This command causes the EV0338 to dump the memory block between addr1 and addr2 in Intel hexadecimal format. If you don't specify addr2, then the block size defaults to 256 bytes.

Example: Dump memory locations 0080h through 00FFh.

```
>dh 80-ff
:10008000F05FFEE61CF7FB00FC00FD0AF300EF8CBE
:10009000CCE200FFCC38F70AFA1EF7FB00F7F200BB
:1000A000080AD4F2FBF7200FF7300FF72103F731DE
:1000B00003F50202F72205F73205F7230FF7330F96
:1000C000F5050FF72500F7281FF7381FF7293FF729
:1000D000393FF74A03F749800A86E00A86E4F79138
:1000E0000EF790E4279327920A76E00AE6E40AD610
:1000F000E40AE6E00AD6E00A7A1E0A3ACA2289072A
:00000001FF
```

5.15.6 ED

This command causes the EV0338 to display the entire contents of external serial EEPROM.

Example:

```
>ed
00: 55 53 45 52 20 4D 45 53 53 41 47 45 20 55 50 20        USER MESSAGE CAN
10: 54 4F 20 34 38 20 43 48 41 52 53 20 49 53 20 53        BE STORED HERE
20: 54 4F 52 45 44 20 49 4E 20 45 45 50 52 4F 4D 0D        - UP TO 48 CHARS.
30: 2D 4E 41 4D 45 2D 02 05 1E 02 06 8B FF FF FF FF        YOUR NAME<
40: 32 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
60: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

70: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

The hexadecimal values of the memory locations are shown in 16-byte blocks, with the address increasing toward the right. Thus, location 00h contains 55h, location 01h contains 53h, location 02h contains 45h, and so forth. Also shown are the ASCII values corresponding to the memory contents. Non-displayable characters are replaced with periods or blanks.

5.15.7 UN [=string]

This command reads or sets a user name up to eight characters in length that the LCD displays statically in “name badge” mode. Most alphanumeric ASCII characters are allowed, but the character restriction is ultimately based upon the limitations of the 14-segment LCD.

Example: read the current user name:

```
>un
YOURNAME
```

This is the default user name stored in EEPROM. To change the user name, add an equals sign (=) to indicate a write transaction. The name may be less than eight characters in length, but names longer than eight characters will be truncated.

```
>un=harry
>un
HARRY
```

5.15.8 EE addr [=byte]

This command allows you to examine or modify the contents of a single location in the external EEPROM. The specified address must not exceed 7Fh. If the contents of an EEPROM location are to be modified, the data value must be in the range 00h–FFh.

```
Example:  examine the contents of EEPROM location 38h:
>ee 38
1E (30)
```

The contents of EEPROM location 38h are displayed in hexadecimal and (decimal) format.

Example: change the contents of EEPROM location 38h:

```
>ee 38=2c
2C (44)
```

A read-back of the data is compared with the original value written. If for some reason the data read back doesn't match the data written, an error will be indicated.

5.15.9 ET [=]

As soon as the EV0338 comes out of reset, an elapsed time clock based on the watch timer interrupt begins operating. The following example shows a reading of the elapsed time clock.

Example: Displaying the elapsed time clock

```
>et
000:00:27:47
```

The time format is "days (0–255) : hours (0–24) : minutes (0–59) : seconds (0–59)". The elapsed time clock may be reset to zero using the **ET** command with an equals (=) sign:

Example: Resetting the elapsed time clock to zero

```
>et=
```

Reading the elapsed time clock immediately after clearing it shows that it was set to zero.

```
>et
000:00:00:00
```

5.15.10 FE [=]

The EV0338 ROM contains an image of the factory default EEPROM data. To compare the values in EEPROM with the factory default values, use the **FE** command:

```
>fe
Address: 00   Value: 59   Factory: 59
Address: 01   Value: 4F   Factory: 4F
Address: 02   Value: 55   Factory: 55
Address: 03   Value: 52   Factory: 52
Address: 04   Value: 20   Factory: 20
Address: 05   Value: 4D   Factory: 4D
Address: 06   Value: 45   Factory: 45
Address: 07   Value: 53   Factory: 53
Address: 08   Value: 53   Factory: 53
Address: 09   Value: 41   Factory: 41
Address: 0A   Value: 47   Factory: 47
Address: 0B   Value: 45   Factory: 45
Address: 0C   Value: 20   Factory: 20
Address: 0D   Value: 43   Factory: 43
Address: 0E   Value: 41   Factory: 41
Etc.
```

A relatively long list appears. Each line shows the EEPROM address, an EEPROM value, and a factory value. If the EEPROM value does NOT match the factory default, a string of three asterisks is appended to the line to indicate a difference.

To reload the factory default values into EEPROM, enter an **FE=** command:

```
>fe=
Address: 00   Value: 59   Factory: 59
Address: 01   Value: 4F   Factory: 4F
Address: 02   Value: 55   Factory: 55
Address: 03   Value: 52   Factory: 52
Address: 04   Value: 20   Factory: 20
Address: 05   Value: 4D   Factory: 4D
Address: 06   Value: 45   Factory: 45
Address: 07   Value: 53   Factory: 53
Address: 08   Value: 53   Factory: 53
Address: 09   Value: 41   Factory: 41
Address: 0A   Value: 47   Factory: 47
Address: 0B   Value: 45   Factory: 45
Address: 0C   Value: 20   Factory: 20
Address: 0D   Value: 43   Factory: 43
Address: 0E   Value: 41   Factory: 41
Etc.
```

In this case, no asterisk strings are appended and the EEPROM's contents match the factory-set values.

5.15.11 FV

This command returns the firmware version number.

Example:

```
>fv
EV0338 MCU Evaluation Board
μPD780338 Mini-Apps + K0 Mini-Monitor
Copyright (C) 2004 NEC Electronics America Inc.
Firmware version 1.00 May 24, 2004
```

5.15.12 HE

The **Help** (**HE**) command displays the list of commands, parameters, brief descriptions.

5.15.13 MB addr.bit [=0][=1]

This command allows you to examine or change a single bit at a specified address location between 0000h–FFFFh and a bit number from 0–7. Note that you cannot change bits in ROM areas of memory and in some areas of RAM. If you attempt to write to a read-only location, the command returns an error message.

Example: Examine bit 0 of RAM location FD00h:

```
>mb fd00.0
FD00 A3 bit0=1
```

Example: Clear bit 0 of RAM location FD00h:

```
>mb fd00.0=0
FD00 A2 bit0=0
```

Example: Examine bit 3 of ROM location 3000h

```
>mb 3000.3
3000 3B bit3=1
```

Example: Clear bit 3 of ROM location 3000h

```
>mb 3000.3=0
Write error to bit location
>
```

5.15.14 MD addr1 [-addr2]

This command displays the memory block specified by the four-digit hexadecimal address range: addr1–addr2. If you don't specify addr2, the unit displays a default block size of 256 bytes.

Example: Display memory block from 800h to 820h

```
>md 800-820
0800: 10 0A E7 2D 01 22 E8 06 38 0E 0A 87 0A E7 A0 0A   ..ç-"è.8..‡.ç .
0810: 27 C5 C3 C5 C3 04 30 05 14 A0 0A F3 00 A4 AC 20   'ÅÃÅÃ.0.. .ó.¤¬
0820: A6 AA AE F8 00 00 0A F7 07 F5 C9 05 25 C3 15 C2   ¦ª®ø...÷.õÉ.%Ã.Â
```

The hexadecimal values of the memory locations are shown in 16-byte blocks, with the address increasing toward the right. Also shown are the ASCII values corresponding to the memory contents. Non-displayable characters are replaced with periods or blanks.

5.15.15 MF addr1 [-addr2] = byte

This command fills the memory block specified by the four-digit hexadecimal address range, addr1–addr2, with the specified byte value. If addr2 is not entered, a default block size of 256 bytes is filled. Attempting to fill ROM using the **MF** command results in a memory write error.

Example: fill RAM locations FDD0h – FDEFh with 7Fh:

```
>mf fda0-fdbf=7f
FDA0 7F (127)
FDA1 7F (127)
FDA2 7F (127)
FDA3 7F (127)
FDA4 7F (127)
FDA5 7F (127)
FDA6 7F (127)
FDA7 7F (127)
FDA8 7F (127)
FDA9 7F (127)
FDAA 7F (127)
FDAB 7F (127)
FDAC 7F (127)
FDAD 7F (127)
FDAE 7F (127)
FDAF 7F (127)
FDB0 7F (127)
FDB1 7F (127)
FDB2 7F (127)
FDB3 7F (127)
FDB4 7F (127)
FDB5 7F (127)
FDB6 7F (127)
FDB7 7F (127)
FDB8 7F (127)
FDB9 7F (127)
FDBA 7F (127)
FDBB 7F (127)
FDBC 7F (127)
FDBD 7F (127)
FDBE 7F (127)
FDBF 7F (127)
```

This examples uses the **MD** command to examine the memory just written:

```
>md fda0-fdbf
```

```
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, the specified RAM locations now contain 7Fh.

5.15.16 MM addr [=byte1] [byte2,byte3,….]

This command modifies the contents of single or multiple memory locations starting at the four-digit hexadecimal address.

Example: examine the contents of RAM location FA00h

```
>mm fa00
FA00 FF (255)
```

The contents of memory location FA00 are displayed in hexadecimal and (decimal) format.

Example: change the contents of RAM location FA00h:

```
>mm fa00=e6
FA00 E6 (230)
```

Example: change the contents of ROM location 3000h:

```
>mm 3000=e6
3000 Write attempted but could not be verified
```

A write was attempted of data E6 to location 3000. A read-back of the data is compared with the original value written. In this example, location 3000 is in ROM. However you can also get this message after attempting to write to a RAM location (such as a control register) where some of the bits are read-only.

The **MM** command can also be used to enter multiple data 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
FD00 00 (0)
FD01 01 (1)
FD02 02 (2)
FD03 03 (3)
FD04 04 (4)
FD05 05 (5)
FD06 06 (6)
FD07 07 (7)
FD08 08 (8)
FD09 09 (9)
FD0A 0A (10)
FD0B 0B (11)
```

```
FD0C 0C (12)
FD0D 0D (13)
FD0E 0E (14)
FD0F 0F (15)
>
```

The monitor responds with a list of the addresses and values, indicating that the operation was a success.

5.15.17 NE

This command shows how the monitor responds to different number bases. The default entry mode is hexadecimal. It is possible to enter a decimal value (in any command, at any point) by preceding the number with a decimal point (.). It is also possible to enter a binary value by preceding the number with a percent sign (%).

Example: Enter 101h:

```
>ne 101
0101 (257)
```

The first number in the returned string is the hexadecimal value, and the second number (in parentheses) is the decimal value.

Example: enter 101 (decimal):

```
>ne .101
0065 (101)
```

Example: enter 101 (binary):

```
>ne %101
0005 (5)
```

5.15.18 RC

This command is used for testing purposes and returns a simple 16-bit checksum in hexadecimal format for the entire contents of the device's ROM. Depending on the target MCU's ROM size and clock speed, this command could take a noticeable amount of time to execute.

Example: Calculate 16-bit ROM checksum

```
>RC
36DF
```

5.15.19 RD [=mm/dd/yy]

This command reads or sets the month, day, and year of the real-time clock. The entries are assumed to be in decimal format.

Example: Set the date as February 18, 2004, where the month is designated as 2 and only the last two digits of the year are used.

```
>rd=2/18/04
02/18/04
```

The new date value is immediately confirmed.

Example: Read the RTC date

```
>rd
02/18/04
```

5.15.20 RT [=hh:mm:ss]

This command reads or sets the hours, minutes, and seconds of the real-time clock. Enter the time in 24-hour (military) format. For example, enter 12:00AM as 00:00, 6:00 AM as 6:00, and 6:00PM as 18:00. Entries are assumed to be in decimal format.

Example: Set the time as 11:10:45 PM.

```
>rt=23:10:45
23:10:45
```

Example: Read the time.

```
>rt
23:10:49
```

5.15.21 SF sfr [=byte/word]] (SF ? for list)

This command enables you to examine or modify the contents of the μPD780338 SFRs by name.

Example: Examine the value of port 0 mode register PM0:

```
>sf pm0
FF20,CF
```

Example: Examine the 16-bit compare register for timer 4:

```
>sf cr4
CR4 FF64 3FFF
```

The returned values show the SFR's 16-bit address value in hexadecimal format, as well as the contents of the specified SFR. If the SFR is a 16-bit register, then the 16-bit value is shown. If the SFR is an eight-bit register, then the 8-bit value is shown.

Example: modify the contents of port 0 mode register PM0:

```
>sf pm0=0f
PM0 FF20 0F
```

Example: attempt to examine the contents of write-only register TXS0:

```
>sf txs0
TXS0 FF1B
```

```
Cannot do a valid read from this SFR
```

Entering the **SF** command followed by a question mark (?) instead of a valid SFR name will result in a list of the μPD780338's SFR names, along with their addresses and read/write attributes. An attribute string of the form of 1681RW would mean that you could read and write in 16-, 8-, and 1-bit units to this SFR. Underscores are substituted if that attribute is not available. For instance, port 1 is an 8-bit input-only port that can be read in 8- or 1-bit units, so its attribute string is a__81R_.

Example:

```
>sf?
SFR list
Name       Address Attributes
P0         FF00     a__81RW
P1         FF01     a__81R_
P2         FF02     a__81RW
P3         FF03     a__81RW
P4         FF04     a__81RW
P5         FF05     a__81RW
P6         FF06     a__81RW
P7         FF07     a__81RW
P8         FF08     a__8_RW
P9         FF09     a__8_RW
P12        FF0C     a__81RW
ADCR0      FF0E     a16__R_
(Etc.)
```

5.15.22 SM [=n]

This command enables you to read or set the default startup mode. Mode values ranging from 1–15 are valid (Table 11). The startup mode setting determines which application will execute by default if you make no additional mode selection at power-up. The factory-set default is mode 7 (clock/calendar with scrolling factory message), which is stored in EEPROM location 58. Entries are assumed to be in decimal format.

Example: Read the current start-up setting:

```
>sm
12
```

The start-up mode is displayed as mode 12.

Example: Change the start-up mode to 14:

```
>sm=14
14
```

5.15.23 UM [=string]

This command reads or sets a message up to 48 characters long that scrolls through the LCD in several operating modes. Most alphanumeric ASCII characters are allowed, but the character restriction ultimately depends on the limitations of the 14-segment LCD.

Example: read the currently stored user message:

```
>um
USERS MESSAGE UP TO 48 CHARS IS STORED IN EEPROM
```

This is the default 48-character message stored in EEPROM. To change the message, add an equals sign (=) to indicate a write transaction. The message may be fewer than 48 characters, but messages longer than 48 will overflow the monitor's receiving buffer.

Example: Store a new user message:

```
>um=this is a new user message
>um
THIS IS A NEW USER MESSAGE
```

5.15.24 XT

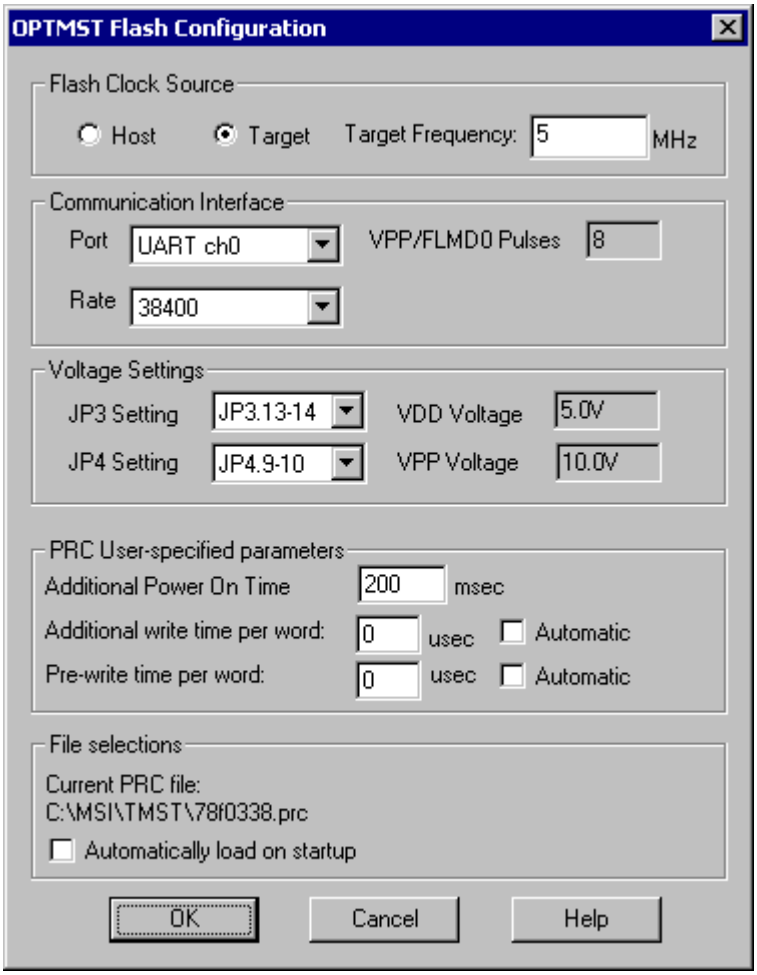
The **XT** command exits the monitor and returns the EV0338 to switch control. Program operation cannot be guaranteed if changes have been made to RAM or any of the SFRs. In this case, it is advisable to press SW1 to reset the EV0338.

Appendix. Using EV0338 with M-Station and M-Station Software

To use the EV0338 with the M-Station baseboard and M-Station software, you must set their jumpers for compatible operation. To use the M-Station to flash program the EV0338 and the MSTTERM program to communicate with the EV0338, you should configure the EV0338 and M-Station for UART flash programming.

1. Set the M-Station jumpers JP8 and JP9 on the M-Station to connect pins 2–3 on both.
    - JP8 2–3 connects the TXDSO line on the flash-programming interface to the UTXD UART output of the USB interface.
    - JP9 2–3 will connect the RXDSI input line on the flash-programming interface to the URXD UART input of the USB Interface.
2. Set JP2 1–2 (REG-VDD) to select power from the on-board regulated V_{DD} voltage.
3. Set JP3 13-14 to select 5.0V for V_{DD}, and JP4 9–10 to select 10.0V for V_{PP}. JP1 does not need to be connected for the EV0338, since the EV0338 can provide the V_{DE} voltage to power the M-Station interface.
4. Set the EV0338 jumpers to the following configuration:
    - JP1: 5–6 V_{DD} from M-Station VDD_FLASH (this is not the default configuration)
    - JP2: 2–5, 3–6 Main clock from on-board 5 MHz crystal
    - JP3: 1–2, 3–4 Subclock from on-board 32.768 kHz crystal
    - J1
        - 3–4 V_{DE} supplied to M-Station
        - 5–6 RXD_SI connected to TxD0 of μPD78F0338 (pin 7, P21/TXD0/SO3)
        - 9–10 TXD_SO connected to RxD0 of μPD78F0338 (pin 6, P20/RXD0/SI3)
        - 17–18 V_{PP} connected to V_{PP} on μPD78F0338
5. Connect the M-Station to the EV0338 using the 16-pin to 16-pin flat cable.
6. Connect the M-Station to a USB port your computer using a standard USB cable. The green D1 USB LED on the M-Station should flash briefly to indicate the start of initialization, and then stay off for 5 to 6 seconds, and then come on steadily to indicate that the M-Station is initialized and ready to communicate.
7. Connect the 12V_{DC} power from the M-Station power supply to the J8 power connector on the M-Station. The red D11 PWR LED should come on. If the default program is in the μPD78F0338 on the EV0338, after a few seconds the LCD display on the EV0338 should have all segments turn on.

8. To operate the M-Station for flash programming of the EV0338, run the OPTMST program to set flash programming options as shown:

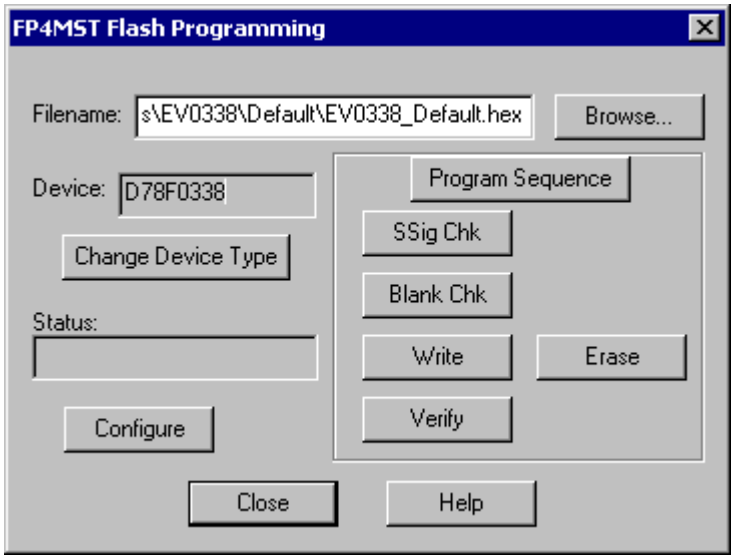


- The source of the flash clock is on the target (EV0338) and the clock frequency is 5 MHz.
- Flash programming communication will take place using the UART ch0 channel (RXD0 and TXD0 on the μPD78F0338) at a baud rate of 38400 baud.
- JP3 has pins 13–14 connected to select 5.0V for V_{DD}, and JP4 has pins 9–10 connected to select 10.0V for V_{PP}.
- An Additional Power On Time of 200 milliseconds is used after V_{DD} is turned on, to allow the power-on reset circuit on the EV0338 to stabilize; other optional parameters are zero and unchecked.
- The PRC file is not be loaded automatically.

9. Close OPTMST to save these settings.

10. Run FP4MST to initialize for flash programming. When the open file dialog shows to select the .PRC file for the device, select the 78f0338.prc file, which can be found in the C:\NECTools32\PRC directory.

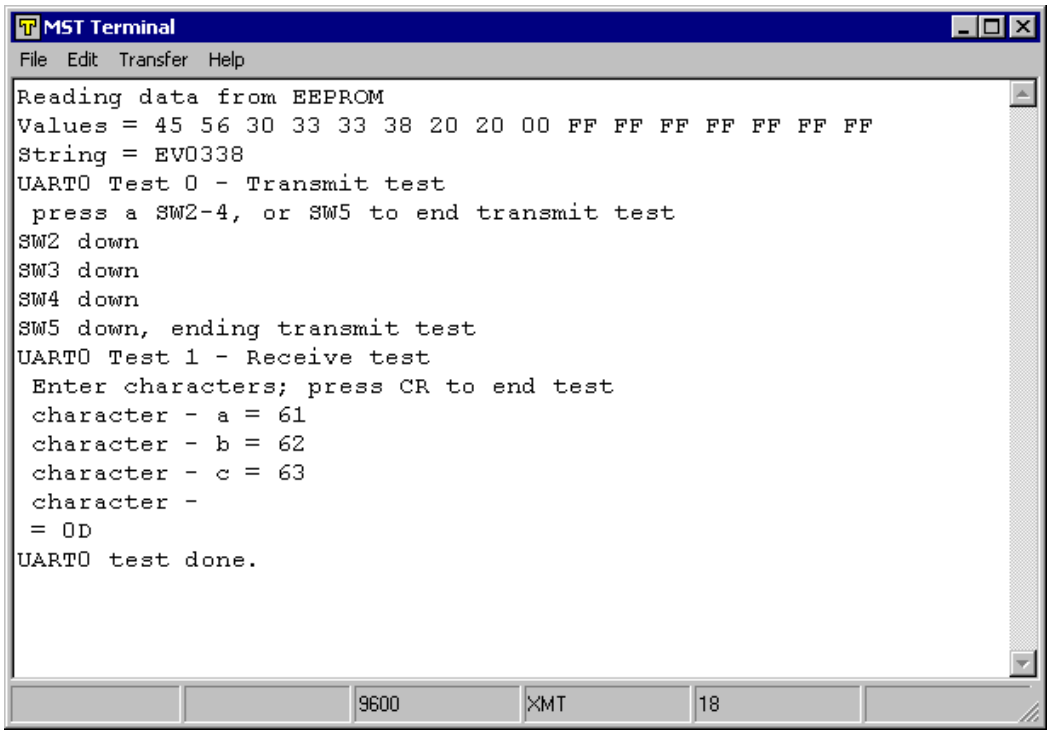
11. You will see the FP4MST Initialization screen briefly and then the FP4MST main window:



12. This dialog box verifies that the M-Station and EV0338 are set up properly for flash programming.

13. Run the MSTTERM program to check for UART communication between the PC and the EV0338 through the M-Station's USB interface. The MSTTERM Terminal window will open. With the default program in the EV0338, after a few seconds all of the LCD segments should come on.

14. If you press SW3, SW5, SW2, SW3, SW4, SW5 in sequence and then press the a, b, c, and <Enter> keys on your keyboard, you should see the following display in the MSTTERM window:



This confirms that the M-Station and the EV0338 are configured properly for UART communication using the MSTTERM program.

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 June 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.

No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

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).