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

Application Note

Measuring Analog Signals

Using NEC Electronics Microcontrollers

Document no. U18271EU1V0AN00

The information in this document is current as of July 2006. 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).

M8E 02.10

Revision History

Date	Revision	Section	Description
July 2006	—	—	First release

Contents:

1. Introduction

This application note gives examples of the use of peripherals in NEC Electronics microcontrollers. The purpose is to help you better understand the use of these peripherals and provide basic routines you can use in more complex applications.

The information provided includes:

- Description of peripheral features
- Example program descriptions and specifications
- Software flow charts
- Applilet reference drivers
- Description of the demonstration platforms used
- Hardware block diagram
- Software modules

Each of the techniques described in this application note use the Applilet, an NEC Electronics software tool that generates driver code for the peripherals. This tool provides a quick and convenient way to generate your code.

For details on using the Applilet and NEC Electronics microcontrollers, please consult the appropriate User's Manuals and related documents.

1.1 Overview of A/D Converters

The A/D (analog-to-digital) converter converts an analog input signal into a digital value. Most NEC Electronics' microcontrollers implement successive-approximation A/D conversion that feature 8 to 16 A/D conversion channels with 8- to 10-bit resolution.

The microcontroller contains all necessary control hardware and control registers. You need only connect analog inputs to the device and read the A/D conversion results for the corresponding analog inputs. The A/D converters are thus simple to use, yet provide high resolution.

The A/D converter generally serves as a real-world interface. For example, you can use the internal A/D converters for measurement and control of temperature or light and in many other measurement and control applications.

2. A/D Conversion

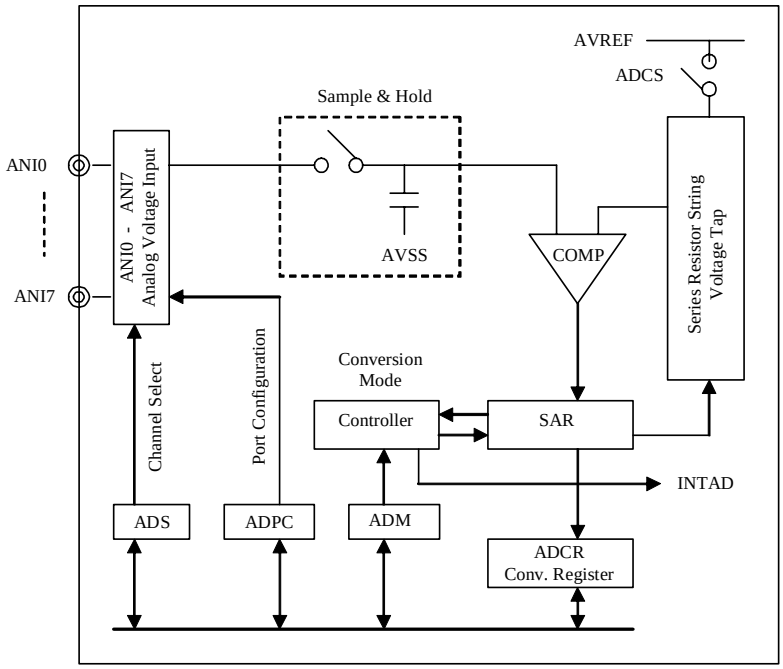
2.1 A/D Converter Features

A typical analog-to-digital converter implemented in an NEC Electronics' microcontroller provides the following features:

- Eight selectable channels of analog input
- 10-bit A/D resolution on selected input, with ½-bit quantization error
- Accuracy of up to +/- 0.4% of full-scale range (at 5V operation)
- A sample-and-hold circuit to hold the selected input while converting
- Selectable conversion times from 5 to 62 microseconds, depending on voltage and clock frequency
- Result available as a 10- or 8-bit value
- Last result remains readable at any time while next conversion is in progress
- Optional interrupt at end of each conversion

A block diagram of a typical A/D converter implemented for NEC Electronics' microcontrollers is shown below.

Figure 1. A/D Converter Block Diagram



Connecting analog signals to the inputs ranging from ANI0 to ANI7 affords eight channels, each of which converts to a 10-bit digital value. Without an external trigger, the A/D conversion repeats continuously for the selected channel. After each conversion, the controller can be configured to generate an interrupt request.

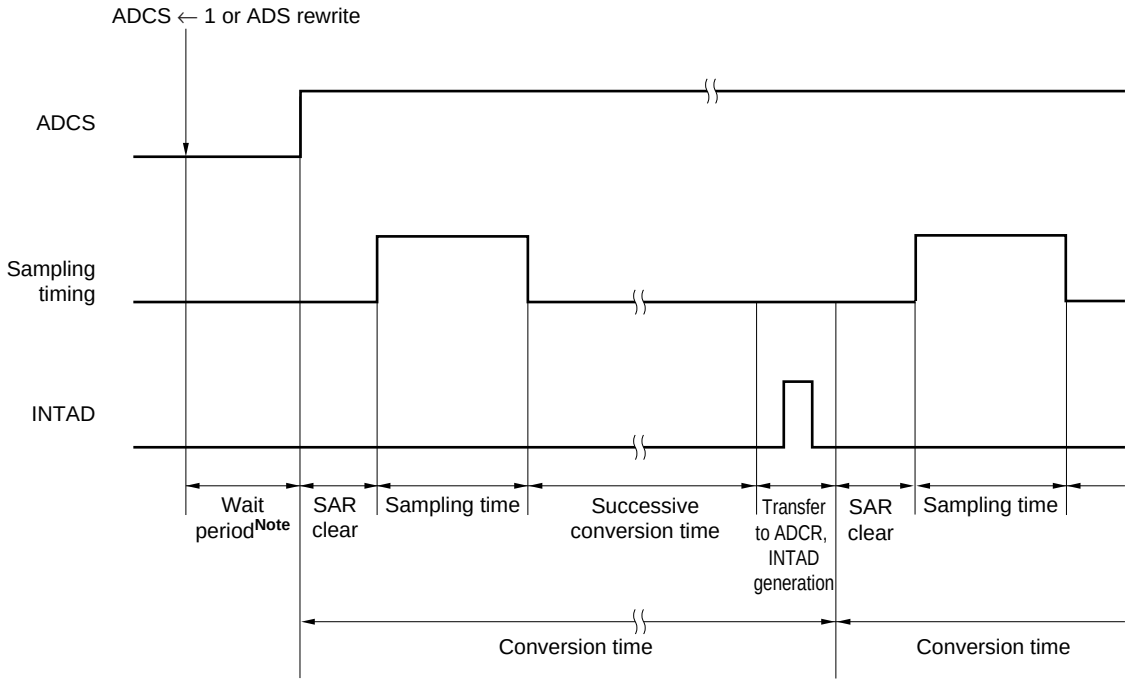
The table below describes the hardware resources controlling the A/D conversion

Table 1. Hardware Resources That Control Conversion

Hardware	Symbol	Brief Description of Functions
Analog Input	ANI0 - ANI7	8-channel A/D converter analog signal input
Sample & Hold	Samples analog signal input selected by selector Holds the sampled analog input voltage value during A/D conversion	
Series Resistor String	Connects between AVREF and AVSS and generates voltage to be compared with the analog input signal	
Voltage Comparator	Compares sampled analog input voltage with output voltage of series resistor string	
Successive Approximation Register	SAR	Compares sampled analog voltage with voltage of series resistor string
		Converts results starting from the most-significant bit (MSB)
		When the voltage value is converted down to LSB, the contents of SAR
		is transferred to A/D Conversion Result Register (ADCR)
Controller	When A/D conversion has been completed, the controller generates INTAD Interrupt	
AVREF Pin	AVREF	Inputs analog reference voltage to A/D Converter
		Converts the analog input into a digital value based on AVREF and AVSS
AVSS Pin	AVSS	Ground potential for A/D converter
Registers Controlling A/D Conversion Operation	ADM	A/D Converter Mode Register
		Sets the conversion -time, starts and stops the conversion
	ADS	Analog Input Channel Specification Register
		Specifies the analog-input port for conversion
	ADPC	A/D Port Configuration Register
		Switches ANI0-ANI7 to analog- or general-purpose port
	PM	Port Mode Register - Switches ANI0 - ANI7 as input or output
	ADCR	10-Bit A/D Conversion Result Register
		The successive approximation register loads this register with the conversion result
	ADCRH	8-Bit A/D Conversion Result Register
		The successive approximation register loads this register with the conversion result

The figure below shows the A/D converter's sampling and conversion timing relationship. For further details of A/D conversion operation, refer to the target device's User's Manual.

Figure 2. A/D Converter Sampling and Conversion Timing Relationship



To configure the A/D converter for operation, follow these steps:

- Use the ADS register to select the desired input channel.
- Use the ADPC and PM2 registers to set the pin as an analog input.
- Use the ADM register to select the conversion time.
- Set the ADCE bit to 1 to enable power to the comparator.
- Wait 1 microsecond or longer.
- Set the ADCS bit to 1 to enable A/D conversion.

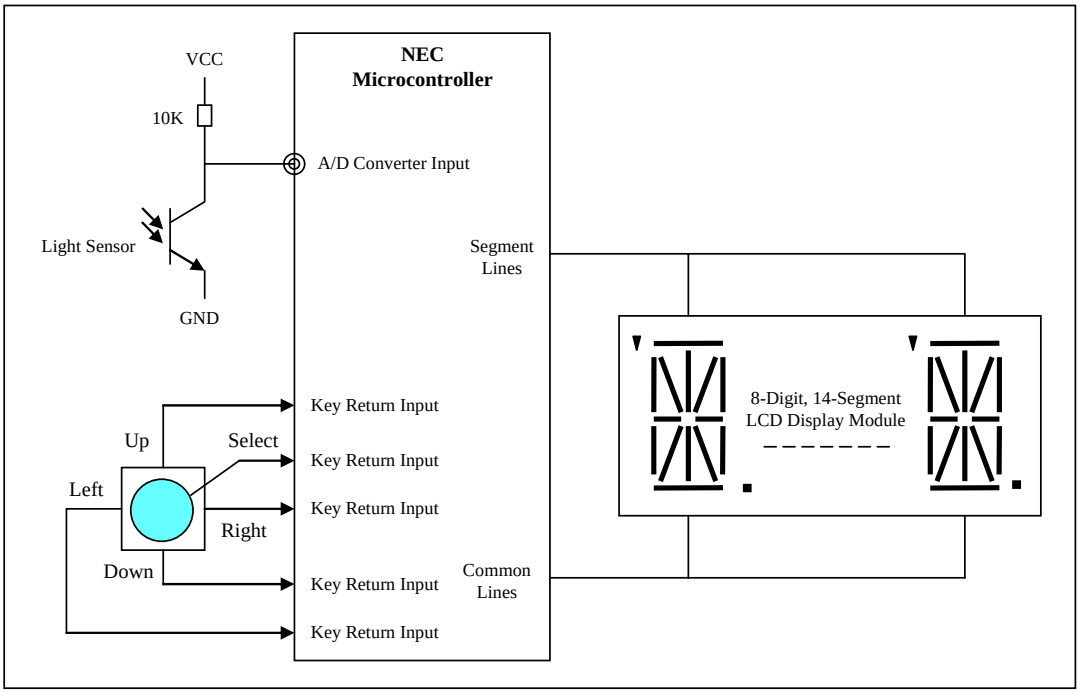
At this point, the A/D converter is operating and will update the result registers after each conversion time.

2.2 Program Description and Specification

The demonstration program uses a light-sensing phototransistor connected to an analog input. Varying the intensity of light produces a varying voltage level at the analog input. Brighter light causes the phototransistor to conduct better, bringing the analog input closer to GND. Dimmer light causes the phototransistor to conduct less well, and the analog input moves towards VCC through a pull-up resistor.

The A/D converter outputs a 10-bit digital value that is displayed on an LCD connected to the NEC Electronics' microcontroller. You can select the display format with an input switch, cycling between hex, decimal, or a percentage of full range.

Figure 3. Hardware for Demonstration Program



Specifications:

- Use ANI0 (Analog Channel 0) as the A/D input.
- Set the A/D conversion time to the longest possible for the given clock rate (33 microseconds for an 8-MHz peripheral clock), to give the most accurate results
- Use the switch input to select among display modes.

2.3 Software Flow Charts

The demonstration program consists of the following major sections related to A/D conversion:

- Program initialization code, (including A/D initialization) called before the main() program starts
- The main program loop, which reads the A/D result and displays it on the LCD
- Subroutines generated by the Applilet for starting, stopping, and reading conversion

The program also includes sections not related directly to A/D conversion:

- IIC0- and LCD-controller initialization
- LCD subroutines to display data by writing to the LCD segment memory
- Subroutines generated by the Applilet for IIC0 communication with the LCD controller
- Subroutines generated by the Applilet to handle timer starting, stopping, and interval setting
- Subroutines with user code for handling key-return interrupts (Applilet-generated stub interrupt service routines, with user code added)

The flowcharts describe only initialization, the main program, and the A/D-related routines, since the purpose of this application note is to show how to use the A/D converter.

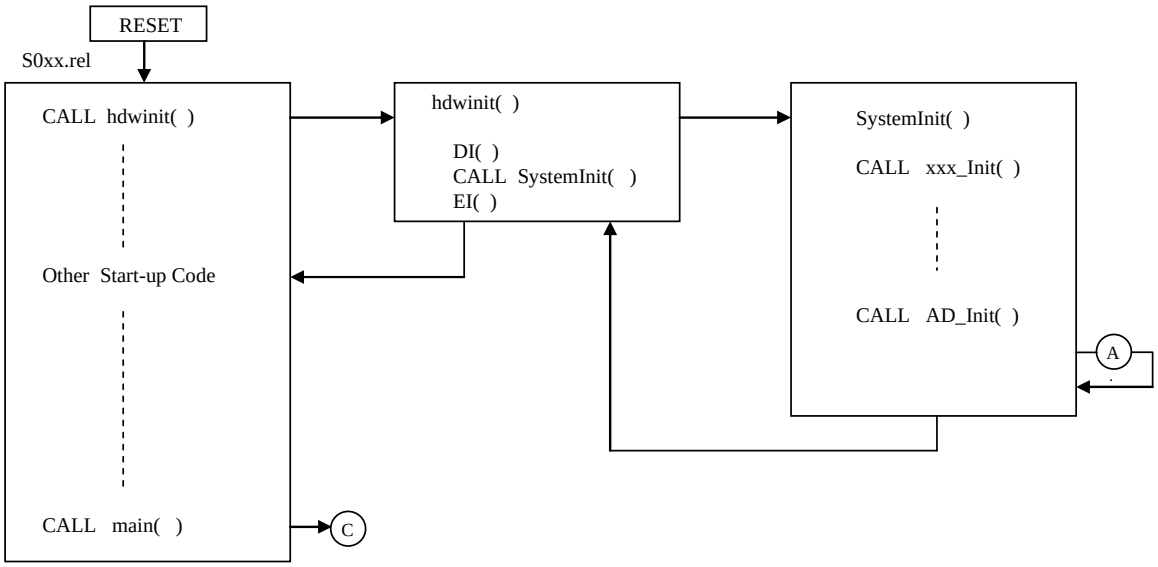
Flowcharts are not included for IIC0 and LCD initialization, IIC0 communication, LCD data transfer, timer-related subroutines and interrupt-service routines, and key-return interrupts. The software listings do include this code, however.

2.3.1 Program Startup and Initialization

For 78K0 programs written in the C language, an object code file such as s01.rel, which is linked into the user program, provides the startup code for the C program. This startup code calls a function named hdwinit( ); you can place any custom hardware-initialization code here.

When the Applilet generates a C program for the 78K0, the tool automatically adds the hdwinit( ) function to the user program and calls the function SystemInit( ). The SystemInit( ) function, in turn, calls the initialization routines for each peripheral.

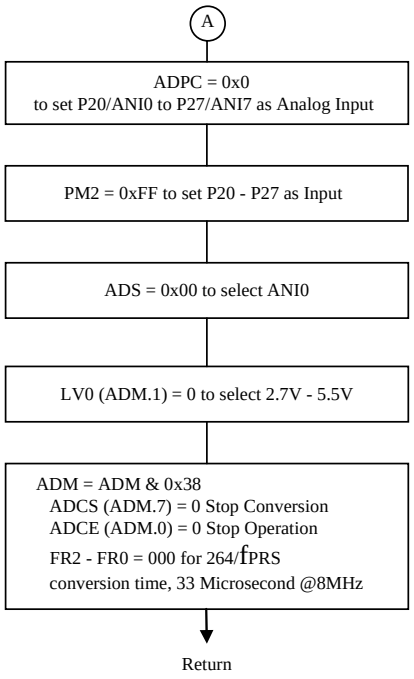
Figure 4. Flowchart for Program Startup



When the hdwinit( ) function finishes, the startup code calls the user program's main( ) function. Thus, at the start of the main( ) function, peripheral initialization has been done for key-return interrupt and timers. The main( ) function does not need to call individual peripheral-initialization routines.

2.3.2 AD_Init( ) – Analog-to-Digital Converter Initialization

Figure 5. Flowchart for Converter Initialization



SystemInit() calls the AD_Init() routine, which sets up the A/D converter function.

The initialization routine first writes 0x00 to the A/D port-configuration register (ADPC), which sets all available pins (P20/ANI0 through P27/ANI7) as analog inputs. Since the demonstration uses P20/ANI0 as the analog input from the light sensor, you must set all of the port pins to analog input. (To use other pins of port P2 as digital input, you could assign the analog input to P27/ANI7, which would leave some of P2's other pins available for digital input.)

The initialization routine sets the port mode-register (PM2) to 0xFF, which configures all P2 pins as inputs to match the selection made with ADPC.

The routine also sets analog input-channel specification register (ADS) to 0x00, which selects ANI0, from pin P20/ANI0, as the current A/D channel to convert.

The initialization sets the LV0 bit (ADM.1) in the A/D converter mode register to zero. This setting tells the A/D converter that the operating voltage will range between 2.7 and 5.5V.

ANDing the value of the A/D converter mode register with 0x38 sets bits 7, 6, 2, 1, and 0 to zero. Setting the ADCS bit (ADM.7) to zero stops the conversion operation, LV0 (ADM.1) to zero (again), selects the

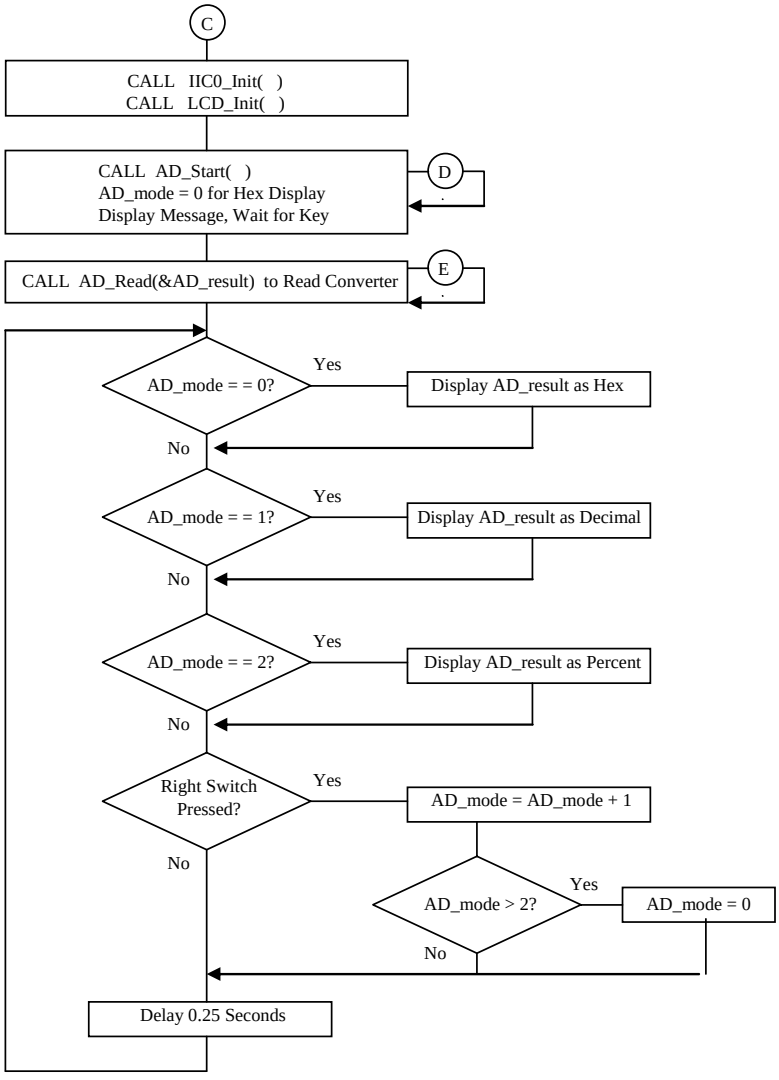
2.7 to 5.5V range, and the ADCE (ADM.0) to zero to stop comparator operation. Bits ADM.6 and ADM.2 (LV1) should always be zero.

This arrangement leaves FR2, FR1, and FR0 bits (ADM.5to ADM.3) at the default of 000 to select a conversion time of 264/fPRS. With an 8-MHz peripheral clock, this selection results in a conversion time of 264/8,000,000 = 33 microseconds.

At this point, the routine has specified the converter operation, but the converter is not yet operating. The main program calls the AD_Start( ) function to start the operation.

2.3.3 Main( ) – Main Program – A/D Conversion and Display

Figure 6. Main Program Flow



After initialization, the startup code calls the main() function. Main() calls IIC0_Init() to initialize the IIC controller, which provides communication with the LCD controller. Then main() calls LCD_Init() to initialize the LCD controller.

The main( ) function calls AD_Start( ) to start the A/D converter operation and sets the A/D converter display mode to 0 for hex display. Main( ) then displays a startup message on the LCD and waits until you press the navigation switch. Pressing this switch causes the program to enter the main program loop.

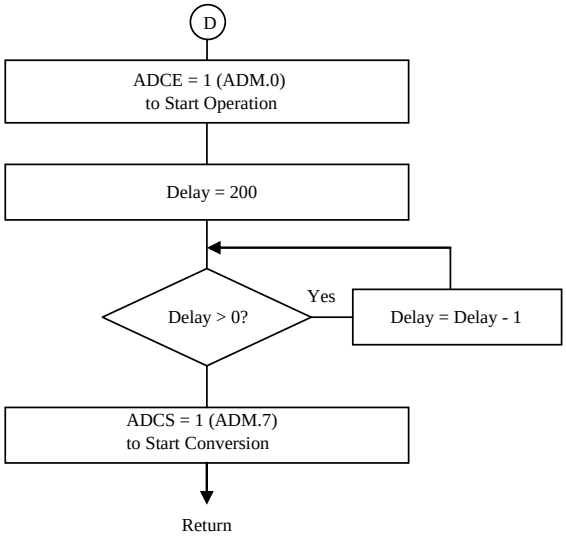
Once in the loop, the program reads the A/D converter into the AD_result variable. This action results in a value between 0 and 0x3FF (1023) when the converter is in its 10-bit range.

Depending on the selected display mode, the result displays on the LCD in hex (000 to 3FF), in decimal (0 to 1023), or as a percentage of full range (0 to 100). A higher value reflects a higher voltage on the ANI0 input, resulting from a darker light input; a lower value reflects a lower voltage, resulting from a brighter light.

The main program loop then checks to see if the navigation switch has been set in the RIGHT position. This switch setting causes the display mode to increment from 0 (hex) to 1 (decimal) to 2 (percent), and then back to 0 if mode 2 was selected. The program then waits for about 0.25 seconds so that the display will not flicker rapidly, and loops to read and display again.

2.3.4 AD_Start( ) – Start Analog-to-Digital Converter Operation

Figure 7. Flowchart for Startup



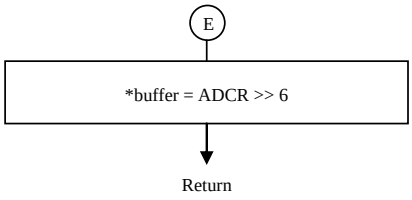
The AD_Start( ) routine starts the A/D converter operation. The routine first sets the ADCE bit (ADM.0) of the A/D converter mode register ADM to 1 to enable comparator operation and start supplying power to the A/D converter.

After ADCE is set to 1, the program waits at least 1 microsecond before setting the ADCS bit to 1 to start conversion operation. The program waits by counting down a delay variable.

The program sets the ADCS bit (ADM.7) to 1 to start conversion. At this point, the A/D converter is operating and updates the ADCR result register every 33 microseconds, as specified in AD_Init( ).

2.3.5 AD_Read(USHORT* buffer) – Read Analog-to-Digital Conversion Result

Figure 8. Flowchart for Reading Conversion Result



The AD_Read(USHORT* buffer) routine reads the result of the A/D converter, which it stores in a location pointed to by the parameter **buffer**. USHORT is defined as “unsigned short” — a 16-bit location representing the values 0 to 65535.

The AD_Read routine gets the current value of the ADCR register, which holds the latest A/D conversion result. Register bits 15 down to 6 hold the 10-bit result. The routine justifies the result by shifting it six bits to the right, to bits 9 down to 0. This shift results in a value from 0 to 0x3FF (1023), which the routine stores in the location pointed to by the **buffer** parameter.

The A/D converter operates continuously, updating the ADCR with the latest result after each conversion. The result stays in the ADCR register until the next conversion is complete, so you can read ADCR at any time to get the latest conversion.

2.4 Applilet's Reference Driver

NEC Electronics’ Applilet program generator automatically generates C or assembly language source code to manage peripherals for NEC Electronics’ microcontrollers. Please see the Appendix for the version of the Applilet used.

The Applilet produces the code for basic initialization, the program’s main function, and for A/D converter control. The tool also produces driver code for the key-return interrupt, for 8-bit timers TM50 and TM51, and for the IIC0 peripheral used to communicate with the LCD controller. After the Applilet produces the basic code, you can add additional code to customize program functions.

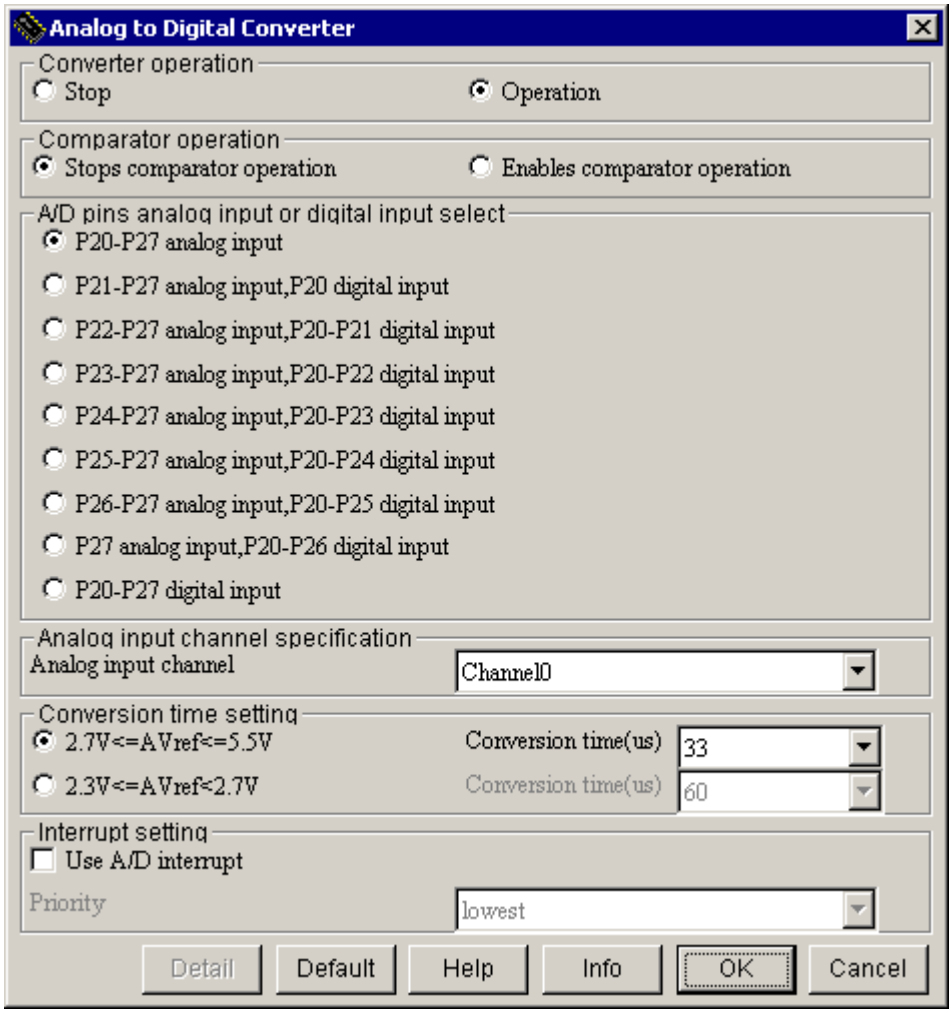
This section describes how to set up the Applilet to produce code for the A/D converter.

When you start the Applilet and select the target device, save your settings to a new project (.prx) file. The Applilet displays a dialog box that lets you select various peripheral blocks for setup.

2.4.1 Configuring Applilet for A/D Converter

Select **A/D Converter** to see a dialog box for setup.

Figure 9. A/D Converter Setup Dialog



Select **Operation** to specify that the A/D converter should be used.

Select **Stops comparator operation** to generate code (AD_Init( )) that initializes the A/D converter with comparator operation stopped, and code for starting (AD_Start( )), which includes start-up of the comparator. (If you select **Enables comparator operation**, the initialization code turns the comparator on, rather than doing it in the AD_Start( ) routine.)

Because the light sensor connects to pin P20/ANI0 on the 78K0/LG2, under **A/D pins** select **P20-P27 analog input**. This is the only selection that allows P20 to serve as an analog input. (If you connect the sensor to a different pin, such as P27/ANI7, you could use other pins on port P2 as digital inputs.)

Under **Analog input channel specification** select **Channel0** from the drop-down menu to specify P20/ANI0 as the conversion channel.

Under **Conversion time setting,** select the AVREF voltage between 2.7V and 5.5V, then select the longest available conversion time (**33 microseconds**) from the drop-down menu.

This demonstration does not require an A/D interrupt at the end of conversions, so leave **Use A/D Interrupt** unchecked. (If you check this box, the Applilet inserts code to enable the interrupt and provides a stub function, MD_INTAD( ), to handle the interrupt.)

2.4.2 Configuring the Applilet for Other Peripherals

For the demonstration program, you need to configure the Applilet to produce code for Timer 50 and Timer 51. These timers are used for delays, serial channel IIC0 for communication with the LCD controller, and for the key-return interrupt.

2.4.3 Applilet-Generated Files and Functions for A/D Converter

The Applilet stores code generated for A/D converter support in the files ad.h, ad.c and ad_user.c.

2.4.3.1 Ad.h

The header file ad.h contains declarations for the functions controlling the A/D converter and definitions of values for A/D converter initialization. The header file macrodriver.h, used for all of the Applilet-generated code, also defines some data types and values, such as the MD_STATUS values returned by some functions.

2.4.3.2 Ad.c

The source file ad.c contains the following function generated by the Applilet for the A/D converter:

void AD_Init(void);

The AD_Init( ) routine initializes the A/D converter.

MD_STATUS AD_Start(void);

The AD_Start( ) routine starts the A/D converter operation.

MD_STATUS AD_Read(USHORT* buffer);

The AD_Read(USHORT* buffer) routine reads the A/D converter result and places it in the location pointed to by the **buffer** parameter.

MD_STATUS AD_Stop(void);

The AD_Stop( ) routine stops the A/D converter operation. The demonstration program does not use this routine.

2.4.3.3 AD_user.c

The source file AD_user.c contains stub functions for user code. These functions are empty on code generation, so that you can add application-specific code.

Because the demonstration does not use an A/D interrupt on end of conversion, this file contains no code. If you used the interrupt, the Applilet would generate the function **__interrupt void MD_INTADD(void)**.

2.4.4 Other Applilet-Generated Files

For the demonstration program, the Applilet generates several other source files. The files and their functions are shown below.

Table 2. Applilet-Generated Files

File	Function
Macrodriver.h	General header file for Applilet-generated programs
Systeminit.c	SystemInit() and hdwinit() functions for initialization
Main.c	The main program function
System.h	Clock-related definitions
System.c	Clock_Init() function
Int.h	Interrupt-related definitions
Int.c	Key-return Interrupt-related functions
Int_user.h	User code for key-return interrupt
Serial.h	IIC0-related definitions
Serial.c	IIC0-related functions
Serial_user.c	User code for IIC0 callback routines
Timer.h	Timer-related definitions
Timer.c	Timer functions
TIMER_user.c	User code for timer interrupt handling
Option.asm	Defines the option byte and security bytes
Option.inc	Defines settings for the option byte and security settings

2.4.5 Demonstration Program Files Not Generated by Applilet

The demonstration program also includes the following files, not generated by the Applilet.

Table 3. Files Not Generated by Applilet

File	Function
define.h	Definitions of navigation switch values
lcd.h	Header file for high-level LCD functions
lcd.c	Code to write characters and strings to the LCD display
LcdDrvApp.h	Header file for LCD driver functions
LcdDrvApp.c	Code to initialize, write and read the LCD display controller; These functions call Applilet-generated IIC0 routines

2.5 Demonstration Platform

The demonstration platform for A/D conversion is a development board from NEC Electronics. You may be able to duplicate the same hardware using off-the-shelf components along with the NEC Electronics microcontroller of interest.

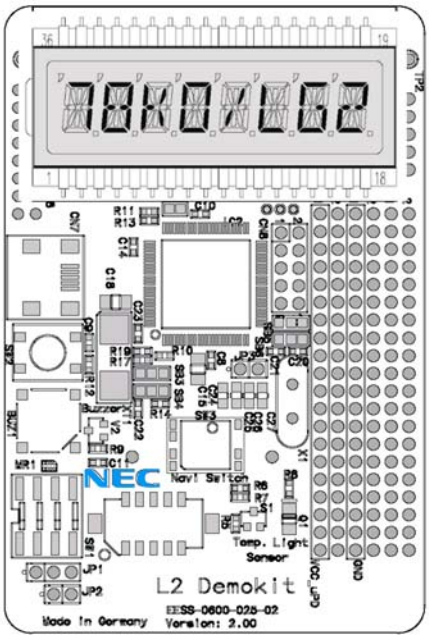
2.5.1 Resources

To program uses the following resources:

- DemoKit-LG2 demonstration board, with uPD78F0397, an 8-bit microcontroller
- DemoKit-LG2 resources:
  - Light-sensing phototransistor Q1 connected to P20/ANI0
  - 5-position navigation switch SW3
  - Eight-character LCD display

For details on the hardware listed above, please refer to the appropriate User's Manual, available from NEC Electronics on request.

Figure 10. Demonstration Hardware



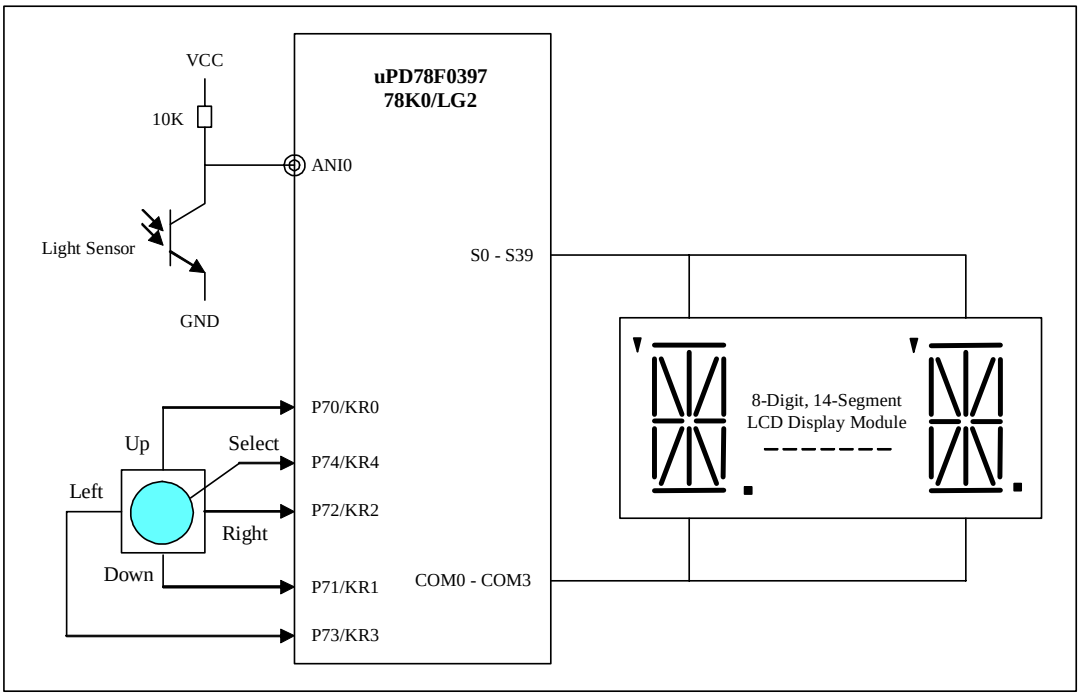
2.5.2 Program Demonstration

With the hardware configured and the uPD78F0397 microcontroller programmed with the demonstration code, the demonstration is as follows:

- Observe the LCD for the digital value of the light-sensor input.
- Cover Q1; observe the response to dimmer light.
- Press SW3 RIGHT to change the display mode.

2.6 Hardware Block Diagram

Figure 11. Hardware Block Diagram



At the ANI0 input a light-sensing phototransistor generates a voltage that varies with the intensity of the incoming light. An A/D converter built into the uPD78F0397 converts that analog level to a digital value.

2.7 Software Modules

The following files make up the software modules for the demonstration program. The table below shows which files were generated by the Applilet, and which of those needed modification to create the demonstration program.

The listings for these files are located in the Appendix.

Table 4. Software Modules in Demonstration Program

File	Purpose	Generated By Applilet	Modified By User
Main.c	Main program	Yes	Yes
Macrodriver.h	General definitions used by the Applilet	Yes	No
System.h	Clock-related definitions	Yes	No
Systeminit.c	SystemInit() and hdwinit() functions	Yes	No
System.c	Clock_Init() function	Yes	No
Int.h	Interrupt-related definitions	Yes	Yes[Note 1]
Int.c	Key-return interrupt-related functions	Yes	No
Int_user.h	User code for key-return interrupt	Yes	Yes[Note 1]
Serial.h	IIC0-related definitions	Yes	Yes[Note 2]
Serial.c	IIC0-related functions	Yes	Yes[Note 2]
Serial_user.c	User code for IIC0 callback routines	Yes	Yes[Note 2]
Ad.h	A/D converter-related definitions	Yes	No
Ad.c	A/D converter-related functions	Yes	No
Ad_user.c	User code for A/D converter	Yes	No
Timer.h	Timer-related definitions	Yes	No
Timer.c	Timer functions	Yes	No
TIMER_user.c	User code for timer interrupt handling	Yes	No
Option.inc	Option-byte, POC, and security definitions	Yes	No
Option.asm	Option-byte, POC, and security data	Yes	No
define.h	Definitions of navigation switch values	--	Yes
Lcd.h	LCD high level function definition	--	Yes
Lcd.c	LCD high level functions	--	Yes
LcdDrvApp.h	LCD controller driver definitions	--	Yes
LcdDrvApp.c	LCD controller driver functions	--	Yes

Note 1: Int.h requires modification to add the declaration of sw3_in, the variable that stores the navigation switch's debounced input. This variable must also be added to Int_user.c along with the code for handling the key-return interrupt.

Note 2: Serial.h must be modified to add the declarations of IIC0-related global variables, defined in Serial_user.c, and also to declare the functions IIC0_Init( ) (not declared by default) and IIC0_MasterStartAndSend( ). Serial.c must be modified slightly to remove settings for P6 not supported for 78K0/LG2 and to replace "asm" versions of DI and EI instructions with NEC Electronics C compiler equivalents. Serial_user.c must be modified to have IIC0 callback functions set the IIC0-related global variables and to add the IIC0_MasterStartAndSend( ) function.

3. Appendix A - Development Tools

This demonstration described in this application note uses the following software and hardware tools.

3.1 Software Tools

Table 5. Software Tools Used for Demonstration

Tool	Version	Comments
Applilet for 78K0KX2	V1.51	Source-code generation tool for 78K0/KE2 devices
PM Plus	V5.20	Project manager for program compilation and linking
CC78K0	V3.60	C Compiler for NEC Electronics 78K0 devices
RA78K0	V3.70	Assembler for NEC Electronics 78K0 devices
DF0397.78K	V1.01	Device file for uPD78F0397 device

Note: The version of the Applilet used produces code for the 78K0/Kx2 family of microcontrollers, with the 78K0/KE2 (uPD78F0537_64) selected as the output device. This device is very similar to the 78K0/LG2 device (uPD78F0397) used in the DemoKit-LG2.

3.2 Hardware Tools

Table 6. Hardware Tools Used for Demonstration

Tool	Version	Comments
DemoKit-LG2	V2.00	Demonstration kit for 78K0397 (78K0/LG2)

4. Appendix B – Software Listings

4.1 Main.c

```
/*  main.c for NEC A/D Converter Application Note */
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename : main.c
**  Abstract : This file implements main function
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler:  NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "int.h"
#include "timer.h"
#include "serial.h"
/*
************************************************************************
**  MacroDefine
************************************************************************
*/

/* include files for DemoKit-LG2 LCD */
#include "lcd.h"
#include "defines.h"
#include "LcdDrvApp.h"

/* add include for sprintf function */
#include "stdio.h"

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     main function
**
**  Parameters:
```

```
**     None
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void main( void )
{
USHORT AD_result;          /* result of A/D conversion read */
unsigned char AD_mode;     /* mode to display A/D conversion */
unsigned char buf[9];      /* string for display */

       IMS = MEMORY_IMS_SET;
       IXS = MEMORY_IXS_SET;
       /* TODO. add user code */

       /* initialize IIC */
       IIC0_Init();

       /* initialize LCD */
       LCD_Init();

       /* Start A/D Converter */
       AD_Start();

       /* set A/D mode to hex display */
       AD_mode = 0;

       while(!sw3_in) {
              LCD_string_shift((unsigned char *)"NEC DEMOKIT-LG2");
              LCD_string_shift((unsigned char *)"ANALOG APP NOTE");
              LCD_string_shift((unsigned char *)"PRESS SW3 TO START");
       }
       sw3_in = 0;

       LCD_string((unsigned char *)" -SW3-  ", 0);

       while(1){
              /* get A/D input, display it in selected mode */
              AD_Read(&AD_result);

              /* format buffer to display result */
              if (AD_mode == 0) {
                     /* hex display */
                     sprintf((char *)buf, "HEX  %03X", AD_result);
              }
              if (AD_mode == 1) {
                     /* decimal display */
                     sprintf((char *)buf, "DEC %4d", AD_result);
              }
              if (AD_mode == 2) {
                     /* percent mode */
                     /* full range is 0-1023, so percent would be n/1023 * 100 */
                     /* we will do an integer calc of n/1024 * 100, or n/256 * 25 */
                     /* note: if we multiply by 100, top value is out of range for int */
                     AD_result = AD_result * 25;      /* do multiply first to keep precision
*/
                     AD_result = AD_result >> 8;      /* quick divide by 256 */
                     sprintf((char *)buf, "PCT  %3d", AD_result);
              }
              /* display result in LCD */
              LCD_string(buf, 0);

              /* check if RIGHT switch input to change mode */
```

```
            if (sw3_in != 0) {
                  /* got a key input, see if it is RIGHT key */
                  if (sw3_in == RIGHT) {
                        AD_mode = AD_mode + 1;
                        if (AD_mode > 2)
                              AD_mode = 0;
                  }
                  sw3_in = 0;  /* clear switch input */
            } /* end switch check */

            /* wait inbetween displays, to not flicker too quickly */
            Wait(5);     /* about 0.25 sec */

      } /* end while (1) loop */
} /* end main() */
```

4.2 Ad.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      ad.h
**  Abstract :      This file implements device driver for AD module.
**  APIlib : NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device : uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
#ifndef      _MDAD_
#define      _MDAD_
/*
************************************************************************
**  MacroDefine
************************************************************************
*/
#define      AD_ADPC      0x0
#define      AD_ADS 0x0
#define      AD_PM2 0xff
#define      AD_ADM 0x38

enum INTMode { INTMode0, INTMode1, INTMode2 };

void AD_Init( void );
MD_STATUS AD_Start( void );
MD_STATUS AD_Read( USHORT* buffer );
MD_STATUS AD_Stop( void );
MD_STATUS AD_Stop( void );
```

```
#endif
```

4.3 Ad.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      ad.c
**  Abstract :      This file implements device driver for AD module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "ad.h"

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     This function initializes A/D module.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------
*/
void AD_Init( void )
{
       ADPC = AD_ADPC;
       PM2 = PM2 | AD_PM2;
       ADS = AD_ADS;
       ClrIORBit(ADM, 0x02);                    /* 2.7V<=AVref<=5.5V */
       ADM = ADM & AD_ADM;
}

/*
**-----------------------------------------------------------------------
**
```

```
**  Abstract:
**     This function starts the A/D converter.
**
**  Parameters:
**     None
**
**  Returns:
**     MD_OK
**
**-------------------------------------------------------------------------
*/
MD_STATUS AD_Start( void )
{
       int delay = 200;
       SetIORBit(ADM, 0x01);                    /* comparator operation */
       while(delay--);

       SetIORBit(ADM, 0x80);
       return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     This function can be called after an A/D conversion is completed,
**     and returns the conversion result(s) in the buffer.
**
**  Parameters:
**     buffer The address where to write the conversion result.
**
**  Returns:
**     MD_OK
**
**-------------------------------------------------------------------------
*/
MD_STATUS AD_Read( USHORT* buffer )
{
       *buffer = (USHORT)( ADCR >> 6 );
       return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     This function stops the A/D converter.
**
**  Parameters:
**     None
**
**  Returns:
**     MD_OK
**
**-------------------------------------------------------------------------
*/
MD_STATUS AD_Stop( void )
{

       ClrIORBit(ADM, 0x81);
       return MD_OK;
}
```

4.4 Ad_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      ad_user.c
**  Abstract :      This file implements device driver for AD module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/

/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "ad.h"
```

4.5 Macrodriver.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename : macrodriver.h
**  Abstract : This is the general header file
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler:  NEC/CC78K0
**
************************************************************************
*/
#ifndef      _MDSTATUS_
#define _MDSTATUS_
```

```
#pragma sfr
#pragma di
#pragma ei
#pragma NOP
#pragma HALT
#pragma STOP

/* data type defintion */
typedef      unsigned long ULONG;
typedef      unsigned int  UINT;
typedef      unsigned short      USHORT;
typedef      unsigned char UCHAR;
typedef      unsigned char BOOL;

#define      ON     1
#define      OFF    0

#define      TRUE   1
#define      FALSE  0

#define IDLE  0                  /* idle status */
#define READ  1                  /* read mode */
#define WRITE 2                  /* write mode */

#define SET   1
#define CLEAR 0

#define MD_STATUS          unsigned short
#define MD_STATUSBASE            0x0

/* status list definition */
#define MD_OK                    MD_STATUSBASE+0x0   /* register setting OK */
#define MD_RESET                 MD_STATUSBASE+0x1   /* reset input */
#define MD_SENDCOMPLETE                MD_STATUSBASE+0x2   /* send data complete */
#define MD_OVF                         MD_STATUSBASE+0x3   /* timer count overflow */

/* error list definition */
#define MD_ERRORBASE             0x80
#define MD_ERROR           MD_ERRORBASE+0x0    /* error */
#define MD_RESOURCEERROR   MD_ERRORBASE+0x1    /* no resource available */
#define MD_PARITYERROR           MD_ERRORBASE+0x2    /* UARTn parity error */
#define MD_OVERRUNERROR          MD_ERRORBASE+0x3    /* UARTn overrun error */
#define MD_FRAMEERROR            MD_ERRORBASE+0x4    /* UARTn frame error */
#define MD_ARGERROR        MD_ERRORBASE+0x5    /* Error agrument input error */
#define MD_TIMINGERROR           MD_ERRORBASE+0x6    /* Error timing operation error */
#define MD_SETPROHIBITED   MD_ERRORBASE+0x7    /* setting prohibited */
#define MD_DATAEXISTS            MD_ERRORBASE+0x8    /* Data to be transferred next exists
in TXBn register */
#define MD_SPT                   MD_STATUSBASE+0x8   /*IIC stop*/
#define MD_NACK                  MD_STATUSBASE+0x9   /*IIC no ACK*/
#define MD_SLAVE_SEND_END  MD_STATUSBASE+0x10  /*IIC slave send end*/
#define MD_SLAVE_RCV_END   MD_STATUSBASE+0x11   /*IIC slave receive end*/
#define MD_MASTER_SEND_END MD_STATUSBASE+0x12   /*IIC master send end*/
#define MD_MASTER_RCV_END  MD_STATUSBASE+0x13  /*IIC master receive end*/

/* main clock and subclock as clock source */
enum ClockMode { HiRingClock, SysClock };

/* the value for IMS and IXS */
#define  MEMORY_IMS_SET          0xCC
#define  MEMORY_IXS_SET          0x00
/* clear IO register bit and set IO register bit */
#define ClrIORBit(Reg, ClrBitMap) Reg &= ~ClrBitMap
#define SetIORBit(Reg, SetBitMap) Reg |= SetBitMap
```

```
enum INTLevel { Highest, Lowest };


#define      SYSTEMCLOCK  8000000
#define      SUBCLOCK     32768
#define      MAINCLOCK    8000000
#define      FRCLOCK      8000000
#define      FRCLOCKLOW   240000


#endif
```

4.6 System.h

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      system.h
**  Abstract :      This file implements device driver for SYSTEM module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
*********************************************************************************
*/
#ifndef      _MDSYSTEM_
#define      _MDSYSTEM_
/*
*********************************************************************************
**  MacroDefine
*********************************************************************************
*/
#define      CG_X1STAB_SEL 0x5
#define      CG_X1STAB_STA 0x1f
#define      CG_CPU_CLOCKSEL     0x0

enum CPUClock { SystemClock, Sys_Half, Sys_Quarter, Sys_OneEighth, Sys_OneSixteen,
Sys_SubClock };
enum PSLevel { PS_STOP, PS_HALT };
enum StabTime { ST_Level0, ST_Level1, ST_Level2, ST_Level3, ST_Level4 };

void  Clock_Init( void );

#endif
```

4.7 Systeminit.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      systeminit.c
**  Abstract :      This file implements macro initialization.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "int.h"
#include "timer.h"
#include "serial.h"
/*
************************************************************************
**  MacroDefine
************************************************************************
*/

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     Init every Macro
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------
*/
void SystemInit( void )
{
      /* Clock generator initiate */
      Clock_Init();
      /* INT initiate */
      INT_Init();
      /* AD initiate */
      AD_Init();
```

```
      /* TM50 initiate */
      TM50_Init();
      /* TM51 initiate */
      TM51_Init();
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Init hardware setting
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void hdwinit( void )
{
      DI( );
      SystemInit( );
      EI( );
}
```

4.8 System.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      system.c
**  Abstract :      This file implements device driver for System module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
/*
************************************************************************
```

```
**  MacroDefine
*******************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**     Init the Clock Generator and Oscillation stabilization time.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------------
*/
void Clock_Init( void )
{
       ClrIORBit(MCM, 0x05);                    /* High-Internal-OSC operate for CPU */

       SetIORBit(MCM, 0x01);                    /* peripheral hardware clock:frh */
       SetIORBit(PM12, 0x18);                   /* P123/124 input mode */
       ClrIORBit(OSCCTL, 0x20);          /* XT1 input mode */
       SetIORBit(OSCCTL, 0x10);
       SetIORBit(MOC, 0x80);                    /* stop X1 clock */
       PCC = CG_CPU_CLOCKSEL;
}
```

4.9 Int.h

```
/*
*******************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      int.h
**  Abstract :      This file implements device driver for INT module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
*******************************************************************************
*/

#ifndef      _MDINT_
#define      _MDINT_
/*
*******************************************************************************
**  MacroDefine
```

```
*******************************************************************************
*/
#define      EGP_INT      0x0
#define      EGN_INT      0x0
#define      PU7_KR 0x1f
#define      PM7_KR 0x1f
#define      KRM_KR 0x1f

enum ExternalINT {
      EX_INTP0, EX_INTP1, EX_INTP2, EX_INTP3,
      EX_INTP4, EX_INTP5, EX_INTP6, EX_INTP7
      };

enum INTInputEdge {
      None, RisingEdge, FallingEdge, BothEdge
      };

enum MaskableSource {
      INT_LVI, INT_INTP0, INT_INTP1, INT_INTP2,
      INT_INTP3, INT_INTP4, INT_INTP5, INT_SRE6,
      INT_SR6, INT_ST6, INT_CSI10_ST0, INT_TMH1,
      INT_TMH0, INT_TM50, INT_TM000, INT_TM010,
      INT_AD, INT_SR0, INT_WTI, INT_TM51,
      INT_KR, INT_WT, INT_INTP6, INT_INTP7,
      INT_IIC0_DMU, INT_CSI11, INT_TM001, INT_TM011
      };
void INT_Init( void );
__interrupt void MD_INTKR( void );

/* global value used for debounced switch input */
extern __sreg volatile unsigned char sw3_in;

#endif
```

4.10 Int.c

```
/*
*******************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      int.c
**  Abstract :      This file implements device driver for INT module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
*******************************************************************************
*/

/*
```

```
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
/*
************************************************************************
**  MacroDefine
************************************************************************
*/

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     This function initializes the external interrupt, key return function.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------
*/
void INT_Init( void )
{
       EGP = EGP_INT;
       EGN = EGN_INT;

       KRMK = 1;                        /* disable INTKR */
       PU7 |= PU7_KR;
       PM7 |= PM7_KR;
       KRM = KRM_KR;                    /* set KR input mode */
       KRPR = 1;
       KRIF = 0;
       KRMK = 0;                        /* enable INTKR */
}
```

4.11 Int_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      int_user.c
**  Abstract :      This file implements device driver for INT module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
```

```
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/

#pragma      interrupt    INTKR  MD_INTKR

/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
/*
************************************************************************
**  MacroDefine
************************************************************************
*/

#include "timer.h"  // for TM51 routines and values
/* global value used for debounced switch input */
__sreg volatile unsigned char sw3_in;

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     INTKR Interrupt service routine.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------
*/
__interrupt void MD_INTKR( void )
{
unsigned char sw3_first,sw3_second;
    sw3_first= (~P7) & 0x1f;     // read SW3 first time
    TM51_ChangeTimerCondition(TM_TM51_INTERVALVALUE);
    TM51_Start();
    while(!TMIF51);             // wait for Timer51 Interrupt
    TM51_Stop();
    TMIF51=0;
    sw3_second= (~P7) & 0x1f;     // read SW3 second time
    if(sw3_first==sw3_second)
        sw3_in=sw3_first; // debounce SW3
    else
        sw3_in=0;
}
```

4.12 Serial.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
```

```
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      serial.h
**  Abstract :      This file implements device driver for SERIAL module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
*******************************************************************************
*/
#ifndef _MDSERIAL_
#define _MDSERIAL_
/*
*******************************************************************************
**  Global variables
*******************************************************************************
*/
#define      IIC0_SLAVEADDRESS   0x0


enum TransferMode { Send, Receive };
MD_STATUS IIC0_MasterStart( enum TransferMode , UCHAR , UCHAR );
MD_STATUS IIC0_MasterSendData( UCHAR* , UINT );
MD_STATUS IIC0_MasterReceiveData( UCHAR* , UINT );
void IIC0_Stop( void );
__interrupt void MD_INTIIC0( void ) ;
void IIC0_User_Init( void );
MD_STATUS IIC0_SlaveHandler( void );
MD_STATUS IIC0_MasterHandler( void );
void CALL_IIC0_SlaveAddressMatch( void );
void CALL_IIC0_MasterFindSlave( void );
void CALL_IIC0_MasterSendEnd( void  );
void CALL_IIC0_MasterReceiveEnd( void  );
void CALL_IIC0_MasterError( MD_STATUS flag );

/* added flags set by callback routines for use by upper level routine */
extern MD_STATUS UI_MasterError;
extern MD_STATUS UI_MasterSendEnd;
extern MD_STATUS UI_MasterReceiveEnd;
extern MD_STATUS UI_MasterFindSlave;

/* added definition for initialize routine */
void IIC0_Init( void );

/* added combined function */
MD_STATUS IIC0_MasterStartAndSend(UCHAR sadr, UCHAR* txbuf, UINT txnum);

#endif
```

4.13 Serial.c

```
/*
*******************************************************************************
```

```
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      serial.c
**  Abstract :      This file implements device driver for SERIAL module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
*******************************************************************************
*/


#pragma interrupt   INTIIC0             MD_INTIIC0

/*
*******************************************************************************
**  Include files
*******************************************************************************
*/
#include "macrodriver.h"
#include "serial.h"
UCHAR  iic0_m_sta_flag;          /* start flag for send address check by master mode */
UCHAR  *iic0_m_send_pbuf;        /* send data pointer by master mode */
UINT   iic0_m_send_size;         /* send data size by master mode */
UCHAR  *iic0_m_rev_pbuf;         /* receive data pointer by master mode */
UINT   iic0_m_rev_size;          /* receive data size by master mode */
UCHAR  iic0_s_sta_flag;          /* start flag for send address check by slave mode */
UCHAR  *iic0_s_send_pbuf;        /* send data pointer by slave mode */
UINT   iic0_s_send_size;         /* send data size by slave mode */
UCHAR  *iic0_s_rev_pbuf;         /* receive data pointer by slave mode */
UINT   iic0_s_rev_size;          /* receive data size by slave mode */

/*
**-----------------------------------------------------------------------------
**
** Abstract:
**     This function initializes IIC0 module.
**
** Parameters:
**     None
**
** Returns:
**     None
**
**-----------------------------------------------------------------------------
*/
void IIC0_Init( void )
{
      ClrIORBit(IICC0, 0x80);               /* stop IIC0 */

      SetIORBit(MK1H, 0x1);                 /* disable interrupt */

      ClrIORBit(PM6, 0x03);                 /* port setting */
// remove setting of P6 for 78F0397 (78K0/LG2)
```

```
//     ClrIORBit(P6, 0x03);

       SetIORBit(IICF0, 0x02);                /* start-condition doesn't need stop-
condition */
       SetIORBit(IICF0, 0x01);                /* comunication reserve - disable */
       SetIORBit(IICC0, 0x10);                /* stop-condition interrupt - enable */
       ClrIORBit(IICC0, 0x08);                /* interrupt control - 8 clock falling edge
*/

       /* transfer clock */
       /* fprs/88 */
       ClrIORBit(IICCL0, 0x08);         /* normal mode */
       ClrIORBit(IICCL0, 0x3);                /* CL00 = 0 CL01 = 0 */
       ClrIORBit(IICX0, 0x1);                 /* disable extension */

       /* selection interrupt priority */
       SetIORBit(PR1H, 0x1);                  /* interrupt priority low */

       ClrIORBit(MK1H, 0x1);                  /* enable interrupt */

       IIC0_User_Init( );

       return;
}

/*
**-------------------------------------------------------------------------------
**
** Abstract:
**     This function is responsible for start IIC0 by master mode.
**
** Parameters:
**     enum TransferMode mode :   select transfer mode
**            Send :       send data
**            Receive :    receive data
**     UCHAR adr :  set address for select slave
**     UCHAR wait :  set wait for need waiting when get start condition
**
** Returns:
**     MD_OK
**     MD_ERROR
**     MD_ARGERROR
**
**-------------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterStart( enum TransferMode mode, UCHAR adr, UCHAR wait )
{
       /* bus check */
//     __asm("di"); // replace in-line assembly with psuedo-function
       DI();
       if( IICF0 & 0x40 ){
//            __asm("ei");        /* bus busy */
              EI();  // replace in-line assembly with psuedo-function
              return MD_ERROR;
       }

       /* start  IIC0 */
       SetIORBit(IICC0, 0x18);          /* SPIE0 = WTIM0 = 1 */
       SetIORBit(IICC0, 0x80);          /* IICE0 = 1 */

       SetIORBit(IICC0, 0x02);          /* generate start condition */
//     __asm("ei");
       EI();  // replace in-line assembly with psuedo-function

       /* wait */
```

```
	while( wait-- );

	if( !(IICS0 & 0x2) ){	/* check start condition */
		return MD_ERROR;
	}

	/* set transfer mode to address */
	/* slave would be selected trans or receive from bit0 at address */
	if( mode == Send ){
		ClrIORBit(adr, 0x01);	/* if master is send mode, clear bit0 */
	}
	else if( mode == Receive ){
		SetIORBit(adr, 0x01);	/* if master is receive mode, set bit0 */
	}
	else{
		return MD_ARGERROR;
	}

	iic0_m_sta_flag = 0;
	IIC0 = adr;		/* send address */

	return MD_OK;
}

/*
**-----------------------------------------------------------------------------
**
** Abstract:
**	This function is responsible for stop IIC0.
**
** Parameters:
**	None
**
** Returns:
**	None
**
**-----------------------------------------------------------------------------
*/
void IIC0_Stop( void )
{
	ClrIORBit(IICC0, 0x80);		/* stop transfer */
	return;
}

/*
**-----------------------------------------------------------------------------
**
** Abstract:
**	This function is responsible for IIC0 data transfering by master mode.
**
** Parameters:
**	UCHAR* txbuf :	transfer buffer pointer
**	UINT txnum :	buffer size
**
** Returns:
**	MD_OK
**	MD_ERROR :	cannot send address
**
**-----------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterSendData(UCHAR* txbuf, UINT txnum)
{
	if( iic0_m_sta_flag == 0 ){
		return MD_ERROR;	/* cannnot send address */
	}
```

```
	/* set parameter */
	iic0_m_send_size = txnum;
	iic0_m_send_pbuf = txbuf;

	IIC0 = *iic0_m_send_pbuf ++ ;		/* start transfer */
	iic0_m_send_size--;

	return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
** Abstract:
**    This function is responsible for IIC0 data receiving by master mode.
**
** Parameters:
**    UCHAR* rxbuf :      receive buffer pointer
**    USHORT rxnum :      buffer size
**
** Returns:
**    MD_OK
**    MD_ERROR :   cannot send address
**
**-------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterReceiveData(UCHAR* rxbuf, UINT rxnum)
{
	if( iic0_m_sta_flag == 0 ){
		return MD_ERROR;		/* cannnot send address */
	}

	/* set parameter */
	iic0_m_rev_size = rxnum;
	iic0_m_rev_pbuf = rxbuf;

	ClrIORBit(IICC0, 0x08);			/* clear WTIM0 */
	SetIORBit(IICC0, 0x04);			/* set ACKE0 */

	SetIORBit(IICC0, 0x20);			/* start receive */

	return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
** Abstract:
**    IIC0 interrupt service routine
**
** Parameters:
**    None
**
** Returns:
**    None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTIIC0( void )
{
	MD_STATUS sta;
	if( IICS0 & 0x80 ) {
		sta = IIC0_MasterHandler();
	}
```

```
	else {
		sta = IIC0_SlaveHandler();
	}
}

/*
**-------------------------------------------------------------------------
**
** Abstract:
**    The function call at IIC0 interrupt request
**
** Parameters:
**    None.
**
** Returns:
**    MD_OK
**    MD_ERROR :   cannot get address
**                 not slave mode
**    MD_SLAVE_RCV_END : all data received
**    MD_SLAVE_SEND_END : all data sended
**    MD_SPT : get stop condition
**
**-------------------------------------------------------------------------
*/
MD_STATUS IIC0_SlaveHandler( void )
{
	/* control for stop condition */
	if( IICS0 & 0x01 ){                          /* get stop condition */
		/* slave send end and get stop condition */
		if( iic0_s_sta_flag &&( iic0_s_send_size == 0 ) ){
			return MD_SLAVE_SEND_END;
		} else {
			return MD_SPT;
		}
	}

	/* control for get address */
	if( iic0_s_sta_flag == 0 ){
		if( !(IICS0 & 0x20) ){               /* check EXC0 -> external code */
			if( IICS0 & 0x10 ){          /* check COI0 -> address */
				iic0_s_sta_flag = 1;
				CALL_IIC0_SlaveAddressMatch();   /* slave address match */
			}
			else{
				return MD_ERROR;
			}
		}
		else{
			return MD_ERROR;
		}
	}

	/* slave send control */
	else if( IICS0 & 0x08 ){
		if(!( IICS0 & 0x04 )){               /* check ACKD0 -> acknowledge */
			return MD_NACK;
		}
		IIC0 = *iic0_s_send_pbuf ++ ;
		iic0_s_send_size--;
	}
	/* slave receive control */
	else{
		*iic0_s_rev_pbuf ++ = IIC0;
		iic0_s_rev_size--;
		SetIORBit(IICC0, 0x20);                      /* WREL1 = 1 start receive */
```

```
            if( iic0_s_rev_size == 0 ){                    /* check all data received */
                  ClrIORBit(IICC0, 0x04);                  /* clear ACKE0 */
                  return MD_SLAVE_RCV_END;
            }
      }
      return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
** Abstract:
**    The function call at IIC0 interrupt request.
**
** Parameters:
**    None.
**
** Returns:
**    MD_OK
**    MD_ERROR :   cannot get ack after sended address
**                       not master mode
**                       slave did not send ack
**    MD_MASTER_RCV_END : all data received
**    MD_MASTER_SEND_END : all data sended
**
**-------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterHandler( void )
{
      /* control for stop condition */
      if( !( IICF0 & 0x40 ) ){                        /* get stop condition */
            CALL_IIC0_MasterError(MD_SPT);
            return MD_SPT;
      }

      /* control for sended condition */
      if( !iic0_m_sta_flag ){
            if( IICS0 & 0x4 ){                        /* check ACK */
                  iic0_m_sta_flag = 1;                      /* address complete */
                  CALL_IIC0_MasterFindSlave();
            }
            else{
                  CALL_IIC0_MasterError(MD_NACK);
                  return MD_NACK;
            }
      }
      /* master send control */
      else if( IICS0 & 0x8 ){
            if( !(IICS0 & 0x4) ){                     /* check ACK */
                  SetIORBit(IICC0, 0x01);                   /* generate stop condition */
                  CALL_IIC0_MasterError(MD_NACK);
                  return MD_NACK;
            }

            if( !iic0_m_send_size ){                  /* sended finish */
                  SetIORBit(IICC0, 0x01);                   /* generate stop condition */
                  CALL_IIC0_MasterSendEnd( );
                  return MD_MASTER_SEND_END;
            }
            IIC0 = *iic0_m_send_pbuf ++ ;                   /* send data */
            iic0_m_send_size--;
      }
      /* master receive control */
      else {
            *iic0_m_rev_pbuf ++ = IIC0;                     /* receive data */
```

```
            iic0_m_rev_size--;
            if( iic0_m_rev_size == 0 ){                    /* receive finish */
                  ClrIORBit(IICC0, 0x04);                  /* ACK STOP */
                  SetIORBit(IICC0, 0x01);                  /* generate stop condition */
                  CALL_IIC0_MasterReceiveEnd( );
                  return MD_MASTER_RCV_END;
            }
            SetIORBit(IICC0, 0x20);                        /* start receive */
      }
      return MD_OK;
}
```

4.14 Serial_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      serial_user.c
**  Abstract :      This file implements device driver for SERIAL module.
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "serial.h"

/* added flags set by callback routines for use by upper level routine */
MD_STATUS UI_MasterError;
MD_STATUS UI_MasterSendEnd;
MD_STATUS UI_MasterReceiveEnd;
MD_STATUS UI_MasterFindSlave;

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**     This function is an empty function for user code when IIC0 initializing
**
**  Parameters:
**     None
**
**  Returns:
```

```
**     None
**
**-------------------------------------------------------------------------
*/

void IIC0_User_Init( void )
{

}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Master Error,
**     callback function open for users operation
**
**  Parameters:
**     MD_STATUS flag
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void CALL_IIC0_MasterError( MD_STATUS flag )
{
       /* user operation */
       UI_MasterError = flag;
       return;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Master recevice finish,
**     callback function open for users operation
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void CALL_IIC0_MasterReceiveEnd( void )
{
       /* user operation */
       UI_MasterReceiveEnd = MD_OK;
       return;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Master send finish,
**     callback function open for users operation
**
**  Parameters:
**     None
**
**  Returns:
```

```
**     None
**
**-------------------------------------------------------------------------
*/
void CALL_IIC0_MasterSendEnd( void )
{
       /* user operation */
       UI_MasterSendEnd = MD_OK;
       return;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     IIC0 slave address match
**     callback function for users operation
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void CALL_IIC0_SlaveAddressMatch( void )
{
       /* user operation */
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Master find the slave address
**     callback function open for users operation
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-------------------------------------------------------------------------
*/
void CALL_IIC0_MasterFindSlave( void )
{
       /* user operation */
       UI_MasterFindSlave = MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**     Combines IIC0_MasterStart(Send, (sadr), 10)
**       and IIC0_MasterSendData(UCHAR* txbuf, UINT txnum)
**
**  Parameters:
**     UCHAR sadr :  set address for select slave
**     UCHAR* txbuf :       transfer buffer pointer
**     UINT txnum :  buffer size
**
**  Returns:
```

```
**     MD_OK
**     MD_ERROR
**     MD_ARGERROR
**   MD_NACK - timeout on slave address
**
**-------------------------------------------------------------------------
*/

MD_STATUS IIC0_MasterStartAndSend(UCHAR sadr, UCHAR* txbuf, UINT txnum)
{
MD_STATUS status;
int i,j;

       // Init IIC in case it needs it
//     IIC0_Init();

       // set up for first operation
       UI_MasterError = MD_OK;
       UI_MasterSendEnd = MD_ERROR;
       UI_MasterFindSlave = MD_ERROR;

       status = IIC0_MasterStart(Send, sadr, 10);
       if (status != MD_OK) {
              return status;
       }

       i = 10000;
       do {
              i--;
       } while ( (UI_MasterFindSlave == MD_ERROR) && (UI_MasterError == MD_OK) && ( i > 0 )
);

       if (i == 0) {
              return MD_NACK;
       }

       if (UI_MasterError != MD_OK) {
              return UI_MasterError;
       }

       // got slave address ok here
       status = IIC0_MasterSendData(txbuf, txnum);    // send data bytes
       if (status != MD_OK) {
              return status;
       }
       i = 10000;
       do {
              i--;
       } while ( (UI_MasterError == MD_OK) && (UI_MasterSendEnd == MD_ERROR) && (i > 0) );

       if (i == 0) {
              return MD_NACK;
       }
       if (UI_MasterError != MD_OK) {
              return UI_MasterError;
       }
       if (UI_MasterSendEnd != MD_OK) {
              return UI_MasterSendEnd;
       }

       // fix to interact with other LCD code for now
//     SetIORBit(MK1H, 0x1);                   /* disable interrupt */
//     ClrIORBit(IICC0, 0x10);                 /* clear SPIE0 bit */
//     ClrIORBit(IF1H, 0x01);                  /* clear IICIF0 bit */
       return MD_OK;/* no error */
```

```
}
```

4.15 Timer.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename : timer.h
**  Abstract : This file implements a device driver for the timer module
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler:  NEC/CC78K0
**
************************************************************************
*/

#ifndef _MDTIMER_
#define _MDTIMER_
/*
************************************************************************
**  MacroDefine
************************************************************************
*/
#define      REGVALUE_MAX  0xff
#define      TM_TM00_CLOCK 0x0
#define      TM_TM00_INTERVALVALUE      0x00
#define      TM_TM00_SQUAREWIDTH 0x00
#define      TM_TM00_PPGCYCLE    0x00
#define      TM_TM00_PPGWIDTH    0x00
#define      TM_TM00_ONESHOTCYCLE       0x00
#define      TM_TM00_ONEPULSEDELAY      0x00
#define      TM_TM01_CLOCK 0x0
#define      TM_TM01_INTERVALVALUE      0x00
#define      TM_TM01_SQUAREWIDTH 0x00
#define      TM_TM01_PPGCYCLE    0x00
#define      TM_TM01_PPGWIDTH    0x00
#define      TM_TM01_ONESHOTCYCLE       0x00
#define      TM_TM01_ONEPULSEDELAY      0x00
#define      TM_TM50_CLOCK 0x7
#define      TM_TM50_INTERVALVALUE      0x2f
#define      TM_TM50_SQUAREWIDTH 0x2f
#define      TM_TM50_PWMACTIVEVALUE     0x2f
#define      TM_TM51_CLOCK 0x7
#define      TM_TM51_INTERVALVALUE      0x8
#define      TM_TM51_SQUAREWIDTH 0x8
#define      TM_TM51_PWMACTIVEVALUE     0x8
#define      TM_TMH0_CLOCK 0x0
#define      TM_TMH0_INTERVALVALUE      0x00
#define      TM_TMH0_SQUAREWIDTH 0x00
```

```
#define      TM_TMH0_PWMCYCLE    0x00
#define      TM_TMH0_PWMDELAY    0x00
#define      TM_TMH1_CLOCK 0x0
#define      TM_TMH1_INTERVALVALUE      0x00
#define      TM_TMH1_SQUAREWIDTH 0x00
#define      TM_TMH1_PWMCYCLE    0x00
#define      TM_TMH1_PWMDELAY    0x00
#define      TM_TMH1_CARRIERDELAY       0x00
#define      TM_TMH1_CARRIERWIDTH       0x00


/* timer00 to 01,50,51,H0,H1 configurator initiation */
void TM50_Init(void);
void TM51_Init(void);

/*timer start*/
void TM50_Start(void);
void TM51_Start(void);

/*timer stop*/
void TM50_Stop(void);
void TM51_Stop(void);
MD_STATUS TM50_ChangeTimerCondition(UCHAR value);
MD_STATUS TM51_ChangeTimerCondition(UCHAR value);

#endif       /* _MDTIMER_*/
```

4.16 Timer.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename : timer.c
**  Abstract : This file implements a device driver for the timer module
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  Device :  uPD78F0537
**
**  Compiler:  NEC/CC78K0
**
************************************************************************
*/

/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"
```

```
/*
************************************************************************************
**  MacroDefine
************************************************************************************
*/
/*TM00 pulse width measure*/

/*TM01 pulse width measure*/

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**     This function can initialize TM50_module.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------------
*/
void TM50_Init( void )
{
       ClrIORBit(TMC50, 0x80);
       TCL50 = TM_TM50_CLOCK;                  /* countclock=fx/8192 */
       /* TM50 interval */
       CR50 = TM_TM50_INTERVALVALUE;
}

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**     This function can start the TM50 counter.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------------
*/
void TM50_Start( void )
{
       /* TM50 interval */
       SetIORBit(TMC50, 0x80);
}

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**     This function can stop the TM50 counter and clear the count register.
**
**  Parameters:
**     None
**
**  Returns:
**     None
**
**-----------------------------------------------------------------------------
```

```
*/
void TM50_Stop( void )
{
      ClrIORBit(TMC50, 0x80);
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function can change TM50 condition.
**
**  Parameters:
**    UCHAR :        value
**  Returns:
**    MD_OK
**    MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM50_ChangeTimerCondition(UCHAR value)
{
      CR50 =value;
      return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function Initializes TM51_module.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void TM51_Init( void )
{
      ClrIORBit(TMC51, 0x80);
      TCL51 = TM_TM51_CLOCK;                  /* countclock=fx/4096 */
      /* TM51 interval */
      CR51 = TM_TM51_INTERVALVALUE;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function starts the TM51 counter.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void TM51_Start( void )
{
      /* TM51 interval */
```

```
      SetIORBit(TMC51, 0x80);
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function stops the TM51 counter and clear the count register.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void TM51_Stop( void )
{
      ClrIORBit(TMC51, 0x80);
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function can change TM51 condition.
**
**  Parameters:
**    UCHAR :      value
**  Returns:
**    MD_OK
**    MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM51_ChangeTimerCondition(UCHAR value)
{
      CR51 =value;
      return MD_OK;
}
```

4.17 TIMER_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2005
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename : timer_user.c
**  Abstract : This file implements a device driver for the timer module
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
```

```
**  Device :  uPD78F0537
**
**  Compiler:  NEC/CC78K0
**
************************************************************************
*/

#pragma sfr
/*
************************************************************************
**  Include files
************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"


/*
************************************************************************
**  MacroDefine
************************************************************************
*/

/* Timer00, Timer01 pulse width measure */
```

4.18 Option.inc

```
;************************************************************************
;**
;**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
;**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
;**
;**  Copyright(C) NEC Electronics Corporation 2002 - 2005
;**  All rights reserved by NEC Electronics Corporation.
;**
;**  This program should be used on your own responsibility.
;**  NEC Electronics Corporation assumes no responsibility for any losses
;**  incurred by customers or third parties arising from the use of this file.
;**
;**  Filename : option.asm
;**  Abstract : This file implements OPTION-BYTES/SECURITY-ID setting.
;**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
;**
;**  Device : uPD78F0537
;**
;**  Compiler :     NEC/CC78K0
;**
;************************************************************************

;
;************************************************************************
;**  MacroDefine
;************************************************************************
;
OPTION_BYTE   EQU    00H
POC81  EQU    00H
POC82  EQU    00H
POC83  EQU    00H
CG_ONCHIP     EQU    02H
CG_SECURITY0  EQU    0ffH
CG_SECURITY1  EQU    0ffH
```

```
CG_SECURITY2  EQU    0ffH
CG_SECURITY3  EQU    0ffH
CG_SECURITY4  EQU    0ffH
CG_SECURITY5  EQU    0ffH
CG_SECURITY6  EQU    0ffH
CG_SECURITY7  EQU    0ffH
CG_SECURITY8  EQU    0ffH
CG_SECURITY9  EQU    0ffH
```

4.19 Option.asm

```
;*****************************************************************************
;**
;**  This device driver was created by Applilet for the 78K0/KB2, 78K0/KC2,
;**  78K0/KD2, 78K0/KE2 and 78K0/KF2 8-Bit Single-Chip Microcontrollers.
;**
;**  Copyright(C) NEC Electronics Corporation 2002 - 2005
;**  All rights reserved by NEC Electronics Corporation.
;**
;**  This program should be used on your own responsibility.
;**  NEC Electronics Corporation assumes no responsibility for any losses
;**  incurred by customers or third parties arising from the use of this file.
;**
;**  Filename : option.asm
;**  Abstract : This file implements OPTION-BYTES/SECURITY-ID setting.
;**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
;**
;**  Device : uPD78F0537
;**
;**  Compiler :     NEC/CC78K0
;**
;*****************************************************************************

;
;*****************************************************************************
;**  Include files
;*****************************************************************************
$ INCLUDE (option.inc)
       OPT_SET CSEG AT 80H
OPTION:       DB     OPTION_BYTE
       DB     POC81
       DB     POC82
       DB     POC83
       ONC_SET CSEG AT 84H
ONCHIP:       DB     CG_ONCHIP

       CSEG SECUR_ID
SECURITY0:    DB     CG_SECURITY0
SECURITY1:    DB     CG_SECURITY1
SECURITY2:    DB     CG_SECURITY2
SECURITY3:    DB     CG_SECURITY3
SECURITY4:    DB     CG_SECURITY4
SECURITY5:    DB     CG_SECURITY5
SECURITY6:    DB     CG_SECURITY6
SECURITY7:    DB     CG_SECURITY7
SECURITY8:    DB     CG_SECURITY8
SECURITY9:    DB     CG_SECURITY9
END
```

4.20 define.h

```
#define UP     0x01
#define DOWN   0x02
#define RIGHT  0x04
#define LEFT   0x08
#define SELECT 0x10

#define BAUDRATE 115200
```

4.21 Lcd.h

```
/*
************************************************************************
**
**  This file was created for the NEC Application Notes
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      lcd.h
**  Abstract :      This file implements header for LCD functions on DemoKit-LG2
**
**  Device :  uPD78F0397
**
**  Compiler :      NEC/CC78K0
**
************************************************************************
*/
#ifndef      _LCD_H_
#define      _LCD_H_

void Wait(unsigned char Number);
void LCD_Init(void);

__callt extern void LCD_putc(unsigned  char digit, unsigned  char data);
__callt extern void LCD_string(unsigned char const *point, unsigned char dpos);
__callt extern void LCD_string_shift(unsigned char const *point);

#endif /* _LCD_H_ */
```

4.22 Lcd.c

```
/*
************************************************************************
**
**  This file was created for the NEC Application Notes
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
```

```
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :      lcd.c
**  Abstract :      This file implements LCD functions on DemoKit-LG2
**              Modified from DemoKit-LG2 example code to use LcdDrvApp.h
**
**  Device :  uPD78F0397
**
**  Compiler :      NEC/CC78K0
**
*****************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"  /* for Timer50 routines */
#include "int.h"    /* for sw3_in definition */
#include "defines.h"
#include "lcd.h"
#include "LcdDrvApp.h"

//---------------------------------------------------------------------------
// Global constants: minimum ASCII table for 14 segment LCD panel
//---------------------------------------------------------------------------
unsigned const short characters[43]   = {

                                          0x0e70, // '0'
                                0x2060, // '1'
                                0x4c32, // '2'
                                0x4872, // '3'
                                0x4262, // '4'
                                0x4a52, // '5'
                                0x4e52, // '6'
                                0x0070, // '7'
                                0x4e72, // '8'
                                0x4a72, // '9'
                                0x500a, // '+'
                                0x4002, // '-'
                                0x4232, // '°'
                                0x0800, // '_'
                                0x2004, // '/'
                                0x4c42, // 'o'
                                0x0000, // ' '  => space
                                0x4672, // 'A'
                                0x4e42, // 'B'
                                0x0e10, // 'C'
                                0x4c62, // 'D'
                                0x0e12, // 'E'
                                0x0612, // 'F'
                                0x4e50, // 'G'
                                0x4662, // 'H'
                                0x1008, // 'I'
                                0x0c70, // 'J'
                                0xa602, // 'K'
                                0x0e00, // 'L'
                                0x2661, // 'M'
                                0x8661, // 'N'
                                0x0e70, // 'O'
                                0x4632, // 'P'
                                0x8e70, // 'Q'
                                0xc632, // 'R'
                                0x4a52, // 'S'
                                0x1018, // 'T'
                                0x0e60, // 'U'
                                0x8061, // 'V'
                                0x9664, // 'W'
                                0xa005, // 'X'
```

```
                                0x2009, // 'Y'
                                0x2814  // 'Z'
                               };

// string of spaces for clearing display
const unsigned char *s_clear   ="        ";

//-----------------------------------------------------------------------------
// Global variables
//-----------------------------------------------------------------------------
__sreg unsigned char transmit_buffer[2];
__sreg char message_byte_count;

//-----------------------------------------------------------------------------
// Module name: Wait
// Description: This module delays the program for (number * 50ms).
//-----------------------------------------------------------------------------
void Wait(unsigned char number)
{
       TM50_Start();// start 50 ms timer
       while (number > 0) {
              while (TMIF50 == 0)
                     ;            // wait for flag at end of time
              TMIF50 = 0;  // clear flag
              number--;    // count down
       }
       TM50_Stop(); // stop timer
       TMIF50 = 0;         // insure flag is zero
}

//-----------------------------------------------------------------------------
// Module name: LCD_Init
// Description: Calls LcdDrv functions to initialize LCD
//-----------------------------------------------------------------------------
void LCD_Init(void)
{
       LcdDrvInit();
       LcdDrvOnWait();

       Wait(80);                  // voltage boost wait time = 4s
       LcdDrvOn();
}

//-----------------------------------------------------------------------------
// Module:   LCD_putc
// Description: Sent character to LCD controller
//-----------------------------------------------------------------------------
__callt void LCD_putc (unsigned char digit, unsigned char data)
{
    // convert special characters
    switch(data)
    {
      case 0x2f:  data = 0x0d;  // '_'
                  break;
      case 0xf0:  data = 0x10;  // ' ' => space
                  break;
      case 0x80:  data = 0x0c;  // '°'
                  break;
      case 0xfb:  data = 0x0a;  // '+'
                  break;
      case 0xfd:  data = 0x0b;  // '-'
                  break;
      case 0xff:  data = 0x0e;  // '/'
                  break;
      case 0x3f:  data = 0x0f;  // 'o'
```

```
                    break;
      default:      break;
    }

    // load transmit buffer
    transmit_buffer [0] = ((characters[data])& 0xff00)>>8;
    transmit_buffer [1] = ((characters[data])& 0x00ff);
    message_byte_count = 2;

    digit<<=1;

    LcdDrvSegWrite(&transmit_buffer[0],digit,message_byte_count);
}

//-------------------------------------------------------------------------------
// Module:   LCD_string
// Description: Send character string to LCD module
//-------------------------------------------------------------------------------
__callt void LCD_string(unsigned char const *point, unsigned char dpos)
{
  while(dpos<=7)
  {
      if(point[0])
      {
          LCD_putc(dpos,*point-0x30);
          *point++;
      }
      else
      {
          LCD_putc(dpos,0xf0);
      }
      dpos++;
  }
}

//-------------------------------------------------------------------------------
// Module:   LCD_string_shift
// Description: Send character string to LCD module
//-------------------------------------------------------------------------------
__callt void LCD_string_shift(unsigned char const *point)
{
  unsigned char dpos=7;
  while(dpos!=0xff)
  {
      if(sw3_in)
      {
        LCD_string(&s_clear[0],0);
        return;
      }
      LCD_string(point,dpos);
      dpos--;
      Wait(4);
  }
  *point++;
  dpos=0;
  while(point[0])
  {
      if(sw3_in)
      {
         LCD_string(&s_clear[0],0);
         return;
      }
      *point++;
      LCD_string(point,dpos);
      Wait(4);
```

```
  }
}
```

4.23 LcdDrvApp.h

```
/************************************************************************
;
;    NNNNNN          NN  EEEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;    NNNNNNNN        NN  EEEEEE               CCCCCC
;    NNNNNNNNNN      NN  EEEEEE              CCCCCC
;    NN  NNNNNNNN    NN  EEEEEEEEEEEEEEEEE   CCCCCC
;    NN    NNNNNNNN  NN  EEEEEE              CCCCCC
;    NN      NNNNNNNNNN  EEEEEE               CCCCCC
;    NN          NNNNNN    EEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;
;    NEC Electronics    78K0/Lx2
;
;************************************************************************
;   78K0/Lx2    LCD Control Sample Program
;************************************************************************
;   LCD Controller/Driver Control   -- Function and constant definition file
;************************************************************************
;[History]
;   2005.06.--  Newly created
;   2006.01.18 - mod to use Applilet-generated IIC routines instead of iic.c
;              - mod for DemoKit-LG2, LCDC = 0x02 instead of 0x0C
;************************************************************************/
/*======================================================================
;   External reference
;======================================================================*/
#ifndef _LCDDRVAPP_H_
#define _LCDDRVAPP_H_

/*== Various interface functions ==*/
/* LCD driver initialization processing  */
extern  unsigned char LcdDrvInit( void );
/* LCD driver display ON wait start pre-processing
   (used only when internal step-up mode is selected) */
extern  unsigned char LcdDrvOnWait( void );
/* LCD driver display ON processing */
extern  unsigned char LcdDrvOn( void );
/* LCD driver display OFF processing */
extern  unsigned char LcdDrvOff( void );
/* LCD driver control register write processing */
extern  unsigned char LcdDrvCtrWrite( unsigned char *src,
                                      unsigned char addr,
                                      unsigned char size );
/* LCD driver segment data write processing */
extern  unsigned char LcdDrvSegWrite( unsigned char *src,
                                      unsigned char addr,
                                      unsigned char size );
/* LCD driver segment data clear processing */
extern  unsigned char LcdDrvSegClr( void );
/* Single byte write processing in LCD driver control register */
extern  unsigned char LcdDrvCtrWrite1Byte( unsigned char addr,
                                           unsigned char data );
/* Single byte write processing of LCD driver segment data */
extern  unsigned char LcdDrvSegWrite1Byte( unsigned char addr,
                                           unsigned char data );
/*== Control register address value ==*/
#define CLDR_ADDR_LCDMD 0x00    /* 0x00: LCD mode select register     */
```

```
#define CLDR_ADDR_LCDM  0x01    /* 0x01: LCD display mode register    */
#define CLDR_ADDR_LCDC  0x02    /* 0x02: LCD clock control register   */
#define CLDR_ADDR_VLCG0 0x03    /* 0x03: LCD step-up control register */

/*== Error type value ==*/
enum
{
    CLDR_ERR_NONE       /* 0: No errors                         */
,   CLDR_ERR_NACK       /* 1: NACK received                     */
,   CLDR_ERR_BUSY       /* 2: Busy (communication disabled)     */
,   CLDR_ERR_PARA       /* 3: Parameter error                   */
};

/*============================================================================
;   Definition of constants
;
;   Settings in registers that control the LCD controller/driver
;   and main unit's control register
;============================================================================*/
/*====================================================================
;   Settings in control register for LCD chip's LCD control unit
;====================================================================*/
/*
;[Communication format]
;
;   Address                 Bit
;             7       6       5       4       3       2       1       0
;             +-------+-------+-------+-------+-------+-------+-------+-------+
; 00H LCDMD   |SEGSET2|SEGSET1|SEGSET0|   0   |   0   |   0   |MDSET1 |MDSET0 |
;             +-------+-------+-------+-------+-------+-------+-------+-------+
;          MDSET1/0   : Selects LCD reference voltage generator
;          SEGSET2-0  : Sets number of segments (fixed as 40)
;
;             +-------+-------+-------+-------+-------+-------+-------+-------+
; 01H LCDM    | LCDON | SCOC  | VLCON |   0   |   0   | LCDM2 | LCDM1 | LCDM0 |
;             +-------+-------+-------+-------+-------+-------+-------+-------+
;           LCDM2-0   : Selects LCD controller/driver display mode
;           VLCON     : Enables/disables operation of step-up circuit
;           SCOC      : Controls segment pins/common pins output
;           LCDON     : Enables/disables LCD display
;
;             +-------+-------+-------+-------+-------+-------+-------+-------+
; 02H LCDC    |   0   |   0   |   0   |   0   | LCDC3 | LCDC2 | LCDC1 | LCDC0 |
;             +-------+-------+-------+-------+-------+-------+-------+-------+
;          LCDC1/0    : Sets LCD clock
;          LCDC3/2    : Sets LCD source clock
;
;             +-------+-------+-------+-------+-------+-------+-------+-------+
; 03H VLCG0   |CTSEL1 |CTSEL0 |   0   |   0   |   0   |   0   |   0   | GAIN  |
;             +-------+-------+-------+-------+-------+-------+-------+-------+
;          GAIN       : Sets  reference voltage level
;          CTSEL1/0   : Selects contrast adjustment
;
;
; **Selects each register's settings from the following patterns.
;
;====================================================================*/
/*--------------------------------------------------------------------
;   LCD mode select register (register address: 00H)
;--------------------------------------------------------------------*/
/*--- Selects LCD reference voltage generator ---*/

//  #define CLDR_LCDMD  0b00000000  /* <1> External resistor division mode */
//  #define CLDR_LCDMD  0b00000001  /* <2> Internal resistor division mode */
    #define CLDR_LCDMD  0b00000010  /* <3> Internal step-up mode           */
```

```
/*            76543210                                    */
/*            -----000 ....... 0: Bit must be set         */
/*            ||||||||                                    */
/*            ||||||||  **Settings are from patterns <1> to <3> above        */
/*            ||||||||                                                       */
/*            ||||||++--    MDSET1/0    Sets LCD reference voltage generator */
/*            |||+++----    <000 fixed>                                      */
/*            +++-------    <000 fixed> Selects number of segments (SEGSET2/1/0) */

/*-------------------------------------------------------------------
;   LCD display mode register (register address:  01H)
;-------------------------------------------------------------------*/
/*-- Selects LCD controller/driver display mode --*/
/*                                                 */
/*                                      |   Resistor    |    Step-up    | */
/*                                      | division mode |      mode     | */
/*                                      |  Time  | Bias |  Time  | Bias | */
/*                                      |division|method|division|method| */
    #define CLDR_LCDM   0b11100000  /* <1> |   4    | 1/3  |   4    | 1/3  | */
//  #define CLDR_LCDM   0b11100001  /* <2> |   3    | 1/3  |   3    | 1/3  | */
//  #define CLDR_LCDM   0b11100010  /* <3> |   2    | 1/2  |   4    | 1/3  | */
//  #define CLDR_LCDM   0b11100011  /* <4> |   3    | 1/2  |   3    | 1/3  | */
//  #define CLDR_LCDM   0b11100100  /* <5> |     Static    |Set prohibited | */
/*            76543210                                    */
/*            XXX--000...... 0: Bit must be set, */
/*            ||||||||       X: Bit setting not required, control by software  */
/*            ||||||||                                                        */
/*            ||||||||  **Settings are from patterns <1> to <5> above         */
/*            ||||||||  **Bits 7 to 5 are controlled by software              */
/*            ||||||||                                                        */
/*            |||||+++--    LCDM2/1/0   Selects LCD controller/driver disp mode */
/*            |||++-----    <00 fixed>                                        */
/*            ||+-------    VLCON (1 fixed) Enables/disables step-up circuit  */
/*            |+--------    SCOC  (1 fixed) Controls segment/common pins output */
/*            +---------    LCDON (1 fixed) Enables/disables LCD display      */

/*-------------------------------------------------------------------
;   LCD clock  control register (register address:  02H)
;-------------------------------------------------------------------*/
/*-- Selects LCD source clock (fLCD) & LCD clock  --*/
/*                                                  */
/*                                      | LCD source | LCD clock  | */
/*                                      |clock (fLCD)| LCD clock  | */
//  #define CLDR_LCDC   0b00000000  /* <1> | fPCL       | fLCD/2^6   | */
//  #define CLDR_LCDC   0b00000001  /* <2> | fPCL       | fLCD/2^7   | */
    #define CLDR_LCDC   0b00000010  /* <3> | fPCL       | fLCD/2^8   | */
//  #define CLDR_LCDC   0b00000011  /* <4> | fPCL       | fLCD/2^9   | */
//  #define CLDR_LCDC   0b00001000  /* <5> | fPCL/2     | fLCD/2^6   | */
//  #define CLDR_LCDC   0b00001001  /* <6> | fPCL/2     | fLCD/2^7   | */
//  #define CLDR_LCDC   0b00001010  /* <7> | fPCL/2     | fLCD/2^8   | */
//  #define CLDR_LCDC   0b00001011  /* <8> | fPCL/2     | fLCD/2^9   | */
//  #define CLDR_LCDC   0b00001100  /* <9> | fPCL/2^2   | fLCD/2^6   | */
//  #define CLDR_LCDC   0b00001101  /* <10>| fPCL/2^2   | fLCD/2^7   | */
//  #define CLDR_LCDC   0b00001110  /* <11>| fPCL/2^2   | fLCD/2^8   | */
//  #define CLDR_LCDC   0b00001111  /* <12>| fPCL/2^2   | fLCD/2^9   | */
/*            76543210                                    */
/*            ----0000 ...... 0: Bit must be set          */
/*            ||||||||                                    */
/*            ||||||||  **Settings are from patterns <1> to <12> above */
/*            ||||||||                                    */
/*            ||||||++--    LCDC1/0     Sets LCD clock    */
/*            ||||++----    LCDC3/2     Sets LCD source clock */
/*            ++++------    <0000 fixed>                  */
/*-------------------------------------------------------------------
;   LCD step-up control register (register address:03H)
```

```
;-------------------------------------------------------------------*/
/*-- Selects reference voltage (VLC2) level & contrast adjustment (TYP. value) -*/
/*                                                  */
/*                                        |Reference|        Contrast        | */
/*                                        | voltage |      adjustment        | */
/*                                        | (VLC2)  |     (TYP. value)       | */
/*                                        |level *1 |  VLC0 |  VLC1 |  VLC2  | */
    #define CLDR_VLCG0  0b10000000  /* <1> |1.5V     | 4.89V | 3.27V | 1.633V | */
//  #define CLDR_VLCG0  0b11000000  /* <2> |1.5V     | 4.71V | 3.13V | 1.567V | */
//  #define CLDR_VLCG0  0b00000000  /* <3> |1.5V     | 4.50V | 3.00V | 1.500V | */
//  #define CLDR_VLCG0  0b01000000  /* <4> |1.5V     | 4.29V | 2.87V | 1.433V | */
//  #define CLDR_VLCG0  0b10000001  /* <5> |1.0V     | 3.29V | 2.27V | 1.133V | */
//  #define CLDR_VLCG0  0b11000001  /* <6> |1.0V     | 3.21V | 2.13V | 1.067V | */
//  #define CLDR_VLCG0  0b00000001  /* <7> |1.0V     | 3.00V | 2.00V | 1.000V | */
//  #define CLDR_VLCG0  0b01000001  /* <8> |1.0V     | 2.79V | 1.87V | 0.933V | */
/*            76543210                                                       */
/*            00-----0 ...... 0: Bit must be set                             */
/*            ||||||||                                                       */
/*            ||||||||  **Settings are from patterns <1> to <8> above        */
/*            ||||||||                                                       */
/*            |||||||+--    GAIN         Sets reference voltage level       */
/*            ||+++++---    <00000 fixed>                                   */
/*            ++--------    CTSEL1/0    Selects contrast adjustment         */
/*                                                                           */
/*  *1: The reference voltage level is selected 1.5 V when the target LCD panel */
/*      is rated at 4.5V, and is selected as 1.0 V when the target LCD panel is */
/*      rated at 3.0 V.                                                      */
/*=====================================================================
;   Definition of main control register settings
;=====================================================================*/
/*---------------------------------------------------------------------
;    Selects clock output
;-------------------------------------------------------------------*/
/*-- Selects PCL's output clock --*/
/*   fSUB=32.768kHz               */
/*   fPRS=peripheral clock        */
/*                                                  | fPRS=10MHz  | fPRS=20MHz | */
//  #define CLDR_CKS    0b00000110  /* <1> | fPRS/2^6 | 156.25kHz   | 312.5 kHz  | */
//  #define CLDR_CKS    0b00000111  /* <2> | fPRS/2^7 |  78.125kHz  | 156.25kHz  | */
    #define CLDR_CKS    0b00001000  /* <3> | fSUB     |  32.768kHz  |  32.768KHz | */
/*            76543210                                                       */
/*            ---X0000 ...... 0: Bit must be set                             */
/*            ||||||||        X: Bit setting not required, control by software */
/*            ||||||||                                                       */
/*            ||||||||  **Settings are from patterns <1> to <3> above        */
/*            ||||||||  **Bit 4 is controlled by software                    */
/*            ||||||||                                                       */
/*            ||||++++--    CCS3/2/1/0  Selects PCL's output clock           */
/*            |||+------    CLOE (0 fixed)  Enables/disables clock output to LCD */
/*            +++-------    <000 fixed>                                     */
/*-----------------------------------------< END OF FILE >----------*/
#endif /* _LCDDRVAPP_H */
```

4.24 LcdDrvApp.c

```
/************************************************************************
;
;    NNNNNN          NN  EEEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;    NNNNNNNN        NN  EEEEEE                 CCCCCC
;    NNNNNNNNNN      NN  EEEEEE                CCCCCC
;    NN  NNNNNNNN    NN  EEEEEEEEEEEEEEEEE    CCCCCC
```

```
;    NN    NNNNNNNN  NN  EEEEEE              CCCCCC
;    NN      NNNNNNNNNN  EEEEEE               CCCCCC
;    NN          NNNNNN    EEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;
;    NEC Electronics    78K0/Lx2
;
;*************************************************************************
;   78K0/Lx2    LCD control sample program
;*************************************************************************
;   LCD controller/driver control   -- Processing file--
;*************************************************************************
;[History]
;   2005.06.--  Newly created
;   2005.07.01  [050701] Provisionally added voltage supply to VLC0
;   2006.01.11  Modified to use Applilet-generated IIC routines - rdh
;   2006.01.27  Correction in LcdDrvOff in turning off VLCON - rdh
;   2006.03.30  Use routines for writing only for Application Notes
*************************************************************************/
#pragma sfr

/*========================================================================
;   INCLUDE
;========================================================================*/
#include    "macrodriver.h"
#include    "serial.h"
#include    "LcdDrvApp.h"

static void LcdDrvClkOut( void );
static void LcdDrvClkStop( void );

/*========================================================================
;   Definition of control area for LCD driver and various settings
;========================================================================*/
/*===========
;   Slave ID
;==========*/
#define     CSLV_ID_LCDCTL  0b01110000      /* Control register (LCDCTL) */
#define     CSLV_ID_LCDSEG  0b01110010      /* Segment data (LCDSEG)  */

/*===========
;   Definition of control register settings on main side
;==========*/
/*-- Clock output select register --*/
#define     LDR_CKS     CKS
#define    LDR_CKS_CLOE    LDR_CKS.4       /* Clock output enable/disable */

/*-- Port/port mode register (directly connected in microcontroller) --*/
#define     PO_LDR_RST  P13.0           /* Reset to LCD chip            */
#define     PM_LDR_OUT  PM14.0          /* Clock output to LCD chip     */

/*-- Port/port mode register --*/                          /*[050701]>>*/
#define     PO_VLC0_HL  P7.7            /* Voltage supply to VLC0 */
#define     PM_VLC0_HL  PM7.7           /* Voltage supply to LLC0 [050701] */

/*===========
   Control register setting
==========*/
/* Selects LCD reference voltage generator: internal step-up mode */
#define     CLDR_LCDMD_VOL  0b00000010

/*************************************************************
;   LCD driver initialization
;-------------------------------------------------------------
;   [I N]   -
;   [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy
```

```
;*****************************************************************/
unsigned char LcdDrvInit( void )
{
    register unsigned char result = CLDR_ERR_NONE;

    PO_LDR_RST = 1;                 /* Cancels LCD chip's reset status  */
    LDR_CKS = ( CLDR_CKS & 0b00001111);  /* Output clock setting (CCS3-0) */

    /*-- Enables clock output to LCD chip  --*/
    LcdDrvClkOut();

    /*--  Selects reference voltage generator --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDMD, CLDR_LCDMD );

    /*-- Clears segment data  --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvSegClr();
    }

    /*-- Selects display mode --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00000111 ));
    }

    /*-- LCD clock setting --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDC, CLDR_LCDC );
    }
    return  ( result );
}

/*****************************************************************
; LCD driver display ON wait start pre-processing (when step-up mode is selected)
;-----------------------------------------------------------------
;   << Note >>
;
; Only N is called when internal step-up mode is selected for the reference
; voltage generator.  After this processing is called, a wait period
; of at least 500 ms should occur, then the "LcdDrvOn" should be called.
;
;-----------------------------------------------------------------
;   [I N]   -
;   [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy
;*****************************************************************/
unsigned char LcdDrvOnWait( void )
{
#if (CLDR_LCDMD==CLDR_LCDMD_VOL)
    /*  Reference voltage generator:  internal step-up mode */
    register unsigned char result = CLDR_ERR_NONE;

    /*-- Enables clock output to LCD chip  --*/
    LcdDrvClkOut();

    /*-- Sets LCD step-up level and contrast--*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_VLCG0, CLDR_VLCG0 );

    /*-- Enables LCD step-up  --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
    }
    /* -- After 500 ms, the "LcdDrvOnWait" function must be called -- */
    return  ( result );
#else/*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    /*  Reference voltage generator:  resistor division mode */
    return  ( 0 );
```

```
#endif/*(CLDR_LCDMD)*/
}

/*****************************************************************
;   LCD driver ON processing
;-----------------------------------------------------------------
;   << Note >>
;
; When internal step-up mode has been selected for the reference voltage
; generator, after the "LcdDrvOnWait" has been called, a wait period of
; at least 500 ms must occur before calling the next function.
;
;-----------------------------------------------------------------
;   [I N]   -
;   [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy
;*****************************************************************/
unsigned char LcdDrvOn( void )
{
    register unsigned char result = CLDR_ERR_NONE;

#if (CLDR_LCDMD==CLDR_LCDMD_VOL)
    /*  Reference voltage generator: internal step-up mode */
    /*-- Setting of deselect potential output --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01100111 ));

    /*-- Display ON setting --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b11100111 ));
    }

#else/*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    /*  Reference voltage generator: resistor division mode */
    /*-- Enables clock output to LCD chip  --*/
    LcdDrvClkOut();

    /*-- Setting of deselect potential output --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01000111 ));

    /*-- Display ON setting --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b11000111 ));
    }

#endif/*(CLDR_LCDMD)*/
    return  ( result );
}

/*****************************************************************
;   LCD driver display OFF processing
;-----------------------------------------------------------------
;   [I N]   -
;   [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy
;
;   *Clears AX register
;*****************************************************************/
unsigned char LcdDrvOff( void )
{
    register unsigned char result = CLDR_ERR_NONE;

    /*-- Clears segment data  --*/
    result = LcdDrvSegClr();

    /*-- Display OFF setting --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01100111 ));
```

```
    }

    /*-- Segment/common buffer output disable setting --*/
    if( result == CLDR_ERR_NONE ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
    }

#if (CLDR_LCDMD==CLDR_LCDMD_VOL)

    /*-- LCD step-up disable setting --*/
    if( result == CLDR_ERR_NONE ){
#if 0   /* correction 060127 - bit 5 is 1, does not turn off VLCON */
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
#else   /* correction turns off VLCON by having bit 5 as zero */
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00000111 ));
#endif
    }

#endif/*(CLDR_LCDMD)*/

    if( result == CLDR_ERR_NONE ){
        /*-- Disables clock output to LCD chip --*/
        LcdDrvClkStop();
    }
    return  ( result );
}

/*********************************************************************
;   Writes LCD driver control data
;---------------------------------------------------------------------
; [I N]   src : Control register data's storage address
;     addr    : Control register address value
;     size    : Number of bytes to be transmitted (4 bytes maximum)
; [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy, 3 = Parameter error
;*********************************************************************/
unsigned char SLDRCTLW( unsigned char *src,
                        unsigned char addr, unsigned char size )
{
    register unsigned char result = CLDR_ERR_NONE;
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[5];
    unsigned char uc;

    /*-- Enables clock output to LCD chip --*/
    LcdDrvClkOut();

    /*-- Checks parameters   --*/
    if(( size == 0 )||( addr > 0x03 )||(( 0x03+1 - addr ) < size )){
        result = CLDR_ERR_PARA;
    }
    buf[0] = addr;
    for (uc = 0; uc < size; uc++) {
        buf[uc+1] = src[uc];
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDCTL, buf, size + 1 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   Writes LCD driver segment data
;---------------------------------------------------------------------
```

```
; [I N]   src : Address of segment data to be written
;     addr    : Data address for start of write operation
;     size    : Number of bytes to be transmitted (maximum:  20 bytes)
; [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy, 3 = Parameter error
;-------------------------------------------------------------------
;   ** Smooths data before transmitting segment data.
;
;   <Received data>
;                    bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
;                   +----+
;     1st byte (00H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                   |<-  segment:S1   ->|<-  segment:S0   ->|
;
;                   +----+----+----+----+----+----+----+----+
;     2nd byte (01H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                     :
;        :            :
;                   +----+----+----+----+----+----+----+----+
;    19th byte (12H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                   +----+----+----+----+----+----+----+----+
;    20th byte (13H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                   |<-  segment:S39  ->|<-  segment:S38  ->|
;
;   <Sent data>
;                    bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
;                   +----+----+----+----+----+----+----+----+
;      1st byte(00H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                                       |<-  segment:S0   ->|
;                                            Address:00H    |
;
;                   +----+----+----+----+----+----+----+----+
;     2nd byte (01H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                     :
;                     :
;                   +----+----+----+----+----+----+----+----+
;    39th byte (26H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                   +----+----+----+----+----+----+----+----+
;    40th byte (27H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                   +----+----+----+----+----+----+----+----+
;                                       |<-  segment:S39  ->|
;                                       |    Address:27H    |
;
;****************************************************************/
unsigned char LcdDrvSegWrite( unsigned char *src,
                              unsigned char addr, unsigned char size )
{
    register unsigned char result = CLDR_ERR_NONE;
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[41];
    unsigned char uci,ucd;

    /*-- Enables clock output to LCD chip --*/
    LcdDrvClkOut();

    /*-- Checks parameters   --*/
    if(( size == 0 )||( addr > 0x13 )||(( 0x13+1 - addr ) < size )){
        result = CLDR_ERR_PARA;
```

```
    }

    buf[0] = addr*2;
    for (uci = 0, ucd = 1; uci < size; uci++, ucd = ucd + 2) {
        buf[ucd]   = src[uci] & 0x0f ;
        buf[ucd+1] = (src[uci] >> 4) & 0x0f;
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, (size * 2) + 1 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*****************************************************************
;   Clears LCD driver segment data
;-----------------------------------------------------------------
; [I N]   -
; [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy
;*****************************************************************/
unsigned char LcdDrvSegClr( void )
{
    register unsigned char result = CLDR_ERR_NONE;
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[41];
    unsigned char uc;

    /*-- Enables clock output to LCD chip --*/
    LcdDrvClkOut();

    buf[0] = 0; // start at location zero
    for (uc = 1; uc < 41; uc++) {
        buf[uc] = 0;
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, 41 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*****************************************************************
;   Writes to LCD driver control register (1 byte setting)
;-----------------------------------------------------------------
; [I N]   addr    : control register  address value
;         data    : control register  data
; [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy, 3 = Parameter error
;*****************************************************************/
unsigned char LcdDrvCtrWrite1Byte( unsigned char addr, unsigned char data )
{
    register unsigned char result = CLDR_ERR_NONE;
    MD_STATUS status;
    unsigned char buf[2];
    unsigned char uc;

    /*-- Enables clock output to LCD chip --*/
    LcdDrvClkOut();

    buf[0] = addr;
    buf[1] = data;

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDCTL, buf, 2 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
```

```
    return  ( result );
}

/*********************************************************************
;   Writes LCD driver segment data (1 byte setting)
;---------------------------------------------------------------------
; [I N]   addr    : Control register address value
;         data    : control register  data
; [OUT]   0= Setting OK, 1 = NACK received, 2 = Busy, 3 = Parameter error
;*********************************************************************/
unsigned char LcdDrvSegWrite1Byte( unsigned char addr, unsigned char data )
{
    register unsigned char result = CLDR_ERR_NONE;
    register unsigned char work;
    MD_STATUS status;
    unsigned char buf[5];
    unsigned char uc;

    /*-- Enables clock output to LCD chip --*/
    LcdDrvClkOut();

    buf[0] = addr;
    buf[1] = data & 0x0f;
    buf[2] = ( data >>4 ) & 0x0f;

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, 3 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   Enables clock and power output to LCD chip
;---------------------------------------------------------------------
; [I N]   -
; [OUT]   -
;*********************************************************************/
static void LcdDrvClkOut( void )
{
    PM_LDR_OUT   = 0;
    LDR_CKS_CLOE = 1;
#if (CLDR_LCDMD==CLDR_LCDMD_VOL)                        /*[050701]>>*/
    PO_VLC0_HL   = 0;   /* Low output (power is supplied to VLC0)*/
#else /*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    PO_VLC0_HL   = 1;   /* High output (power is not supplied to VLC0)*/
#endif/*(CLDR_LCDMD)*/                                  /*[050701]<<*/
}

/*********************************************************************
;   Disables clock and power output to LCD chip
;---------------------------------------------------------------------
; [I N]   -
; [OUT]   -
;*********************************************************************/
static void LcdDrvClkStop( void )
{
    LDR_CKS_CLOE = 0;
    PM_LDR_OUT   = 1;
    PO_VLC0_HL   = 0;                                   /*[050701]*/
}

/*-----------------------------------------------< END OF FILE >---*/
```