NEC

Application Note

Demonstration of Power-Down Modes

Document no. U19132EU1V0AN00
©February 2008. NEC Electronics America, Inc.
All rights reserved.

The information in this document is current as of February 2008. 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
February 2008	—	—	First release

Contents

1. Introduction 7

1.1 Overview of Power-Down Features 7

2. Clock Control and Standby Functions 7

2.1 Features of Clock Control and Standby Functions 8
2.1.1 Clock Control Features 8
2.2 Standby Features 10
2.2.1 HALT Mode 11
2.2.2 IDLE Mode 12
2.2.3 STOP Mode 12

3. Program Description and Specification 13

3.1 Software Flowcharts 13
3.2 Applilet's Reference Drivers 13
3.3 Demonstration Platform 14
3.3.1 Resources 14
3.3.2 Demonstration Program 15
3.3.3 Hardware Block Diagram 17
3.4 Software Modules 17

4. Appendix A — Software Flowcharts 19

4.1 main.c 19
4.2 clock_PLL_mode 23
4.3 clock_XTAL_mode 24
4.4 clock_SUB_mode 25
4.5 clock_HALT_mode 26
4.6 clock_IDLE_mode 27
4.7 clock_STOP_mode 28
4.8 systeminit.c 29
4.9 port.c 31
4.10 port_user.c 32
4.11 watchtimer.c 35
4.12 watchtimer_user.c 37
4.13 watchdogtimer_user.c 39
4.14 watchdogtimer.c 40
4.15 write_special.s 41
4.16 crtE.s 45

5. Appendix B – Applilet Tool 47

5.1 System Memory 47
5.2 System Peripherals 48
5.3 System 49
5.3.1 System Foundation settings 49
5.3.2 System Startup Settings 49
5.3.3 System Watchdog Settings 49
5.4 Port Subsystem 50
5.4.1 Port 9-1 Selection 50
5.4.2 Port DH Selection 51
5.4.3 Port DL-2 Selection 51

5.5 Timer Subsystem 52
5.5.1 Watch Timer Selections 52
5.6 Interrupt Selections 53
5.7 Generation of Source files 53
5.7.1 Interrupt and Port Functions 53
5.7.2 Timer Functions 54

6. Appendix C — Software Code Listings 55

6.1 crtE.s 55
6.2 systeminit.c 59
6.3 main.c 61
6.4 int.c 68
6.5 int_user.c 70
6.6 port.c 72
6.7 port_user.c 75
6.8 timer.c 82
6.9 timer_user.c 85
6.10 watchdogtimer.c 87
6.11 watchdogtimer_user.c 89
6.12 watchtimer.c 91
6.13 watchtimer_user.c 94
6.14 write_special.s 98
6.15 int.h 102
6.16 macrodriver.h 103
6.17 port.h 105
6.18 timer.h 109
6.19 watchdogtimer.h 112
6.20 watchtimer.h 113

1. Introduction

This application note illustrates the use of the peripherals in NEC Electronics microcontrollers (MCUs). It will help you better understand the peripherals and provide you with 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
- Descriptions of the demonstration platforms
- 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 code.

For details on using the Applilet and NEC Electronics MCUs, please consult the appropriate user manuals and related documents.

1.1 Overview of Power-Down Features

The power dissipation of CMOS devices is:

P = C x (V**2) x (frequency of operation)

Power dissipation is a direct function of the operating frequency.

NEC Electronics MCUs implement many advanced features to select and control CPU and peripheral clocks. Using these features, you can reduce the CPU clock frequency when you do not need full processing power and turn off on-chip peripherals when they are not needed. Standby features, implemented for most NEC Electronics microcontrollers, reduce power consumption to a minimum. These features make NEC Electronics microcontrollers an ideal choice for battery-operated portable systems. This document demonstrates various methods of controlling the clocks and using the standby features.

2. Clock Control and Standby Functions

Whether using an on-board oscillator or external clock, the clock generator is a major power consumer. NEC Electronics microcontrollers have two major functions that reduce this power consumption:

- Comprehensive methods of controlling clocks
- Standby features

2.1 Features of Clock Control and Standby Functions

This section provides a brief description of the features for clock control and standby.

2.1.1 Clock Control Features

The clocking options for NEC Electronics microcontrollers include:

- Main clock operation from external crystal oscillator for high speed
- Operation from a driven external clock
- Operation from an internal high-speed oscillator (available in some microcontrollers)
- Ability to operate the CPU at the main clock frequency or at a fraction of it (for power saving)
- Ability to operate CPU and peripherals from different clocks
- Support for a low-speed external subclock (32.768 kHz) for timekeeping
- Ability to operate the CPU from the subclock for reduced power consumption
- Ability to operate some peripherals on the subclock
- Low-speed internal oscillator for some peripherals (available in some microcontrollers)
- Clock pins not used for clocks available for use as I/O ports

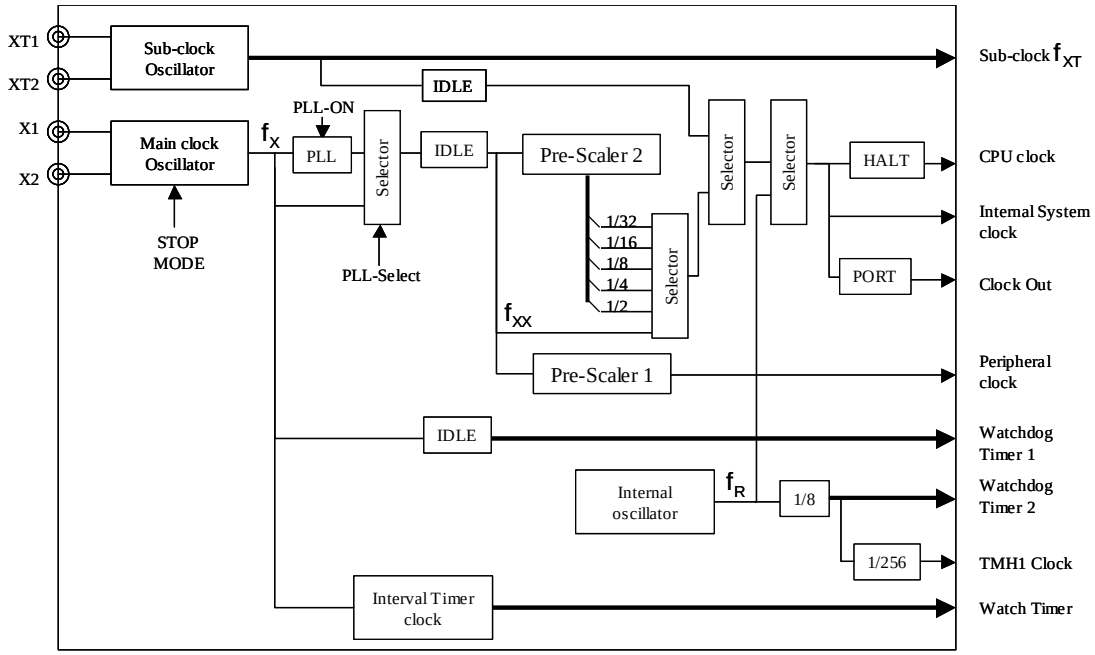
The clock generator generates clocks for the CPU and peripheral hardware. The main clock (X1) can be selected by the main-clock mode register (MCM) and clock-operation mode-select register (OSCCTL). Various sources of the system clock for NEC Electronics microcontrollers are shown in the table below. The subsystem clock is a 32.768 kHz crystal oscillator.

Table 1. System Clock Sources

Clock Category	Clock Select	Description
Main Clock (X1)	Main oscillator	High-speed main oscillator Main oscillator can be stopped by STOP instruction. Main oscillator can also be stopped by setting OSC control register.
	External clock	External clock can be disabled by executing STOP instruction.
		External clock can also be disabled by RCM register.
Subsystem Clock (XT1)	Subsystem clock oscillator	The subsystem clock oscillator frequency is 32.768 kHz.
	External subsystem clock	External subsystem clock can be disabled by processor clock control or oscillator control register.
Low-Speed Ring Oscillator	Internal oscillator oscillates at typical frequency of 240 kHz. After reset, the low-speed ring oscillator always starts operating.	
	Oscillation can be stopped by Ring Oscillator Mode Register (depending on mask option or option-byte setting).	
	The low-speed ring oscillator cannot be used for CPU clock.	
	Low-speed ring oscillator operates watchdog timer and Timer H1.	

The implementation of clock sources differs from one MCU to another. For example, some MCUs may have a low-speed internal oscillator implemented, while others do not. Please refer to product specifications for clock source implementations.

Figure 1. Typical Clock System in NEC Electronics Microcontrollers



As the main system clock, you can select either the X1 externally supplied high-speed clock (f_X) or the XT1 Subclock (f_{XT}) oscillator. The peripheral hardware clock derives from the main system clock.

Newer members of the NEC Electronics microcontroller family feature the clock 256 divider to Timer H1. After the release of reset, the main oscillator waits for the oscillation stabilization time set by the OSTS register. Once the oscillator stabilizes, this clock source is set to Fxx/8 of the CPU clock.

Newer members of NEC Electronics microcontroller family also incorporate an internal low-speed oscillator. This oscillator serves not only for the watchdog timer and 8-bit timer H1, but also for a CPU clock monitor that generates a reset signal (CLMRES) when oscillation of the main clock is stopped. The status of the CPU clock can be read via the CCLS register. After release of reset, the low-speed internal oscillator starts—typically running at 240 kHz.

The microcontroller's prescaler generates various clocks by dividing the main-system clock you select as the CPU clock source.

You can use the MCU's phase-locked loop (PLL) function to output the operating clock for the CPU and peripherals at a frequency 4 times higher than the oscillation frequency.

Table 2. Frequency Ranges With and Without PLL

PLL function	Input Clock (MHz)	Operating Clock (MHz)
Using PLL	2 - 5	8 - 20
Not using PLL	2 - 10	2 - 10

When the PLL function is used, the PLL operates immediately after the Reset has been released, provided that the PLLON bit is set in the PLL Control Register. Note that the default is Clock-through Mode, where the main clock oscillator is selected. Therefore, to use the PLL function, you must use the PLL Control Register.

To enable PLL operation, first set the PLLON bit in the PLL Control Register and wait 200 µs until the PLL function stabilizes. Next, select the PLL function by setting the SELPLL bit.

To disable the PLL function, select Clock-through Mode by setting the SELPLL bit and wait 8 clock cycles or more to stabilize the Clock-through Mode. Then reset the PLLON bit (PLLON = 0) to stop the PLL function.

The PLL function operates even if Clock-through Mode is selected. Therefore, you need to reset the PLLON bit when you are not using the PLL function. In addition, the PLL operates in the IDLE Mode. To minimize power consumption, stop the PLL in the IDLE Mode or enter into Stop Mode.

Table 3. Registers Controlling Clock Generator

Register	Symbol	Description of Functions
Processor Clock Control Register	PCC	Selects CPU-clock and division ratio
		Sets operation mode for subsystem clock
Ring oscillator Mode Register	RCM	Sets operation mode of Ring oscillator
CPU Operation Clock Status Register	CCLS	CPU operation in Main clock or Subclock or Ring-OSC Clock
PLL Control Register	PLLCTL	Controls PLL Operation and PLL Mode

To set up the system clocks for operation, the typical sequence is:

- Configure the I/O pins for external high-speed clock (X1 and X2).
- Wait for external oscillator stabilization with the OSTS register.
- Set the OSTS register for restart stabilization time from STOP mode.
- Stop the Ring oscillator, if desired, with the RCM register.
- Select a CPU clock divider or subclock with the PCC register.

2.2 Standby Features

The Standby Mode is provided to reduce system operating current. A typical V850ES Series microcontroller supports HALT, STOP and IDLE Modes as part of the Standby Modes, as shown in the following table.

Table 4. Standby Modes

Mode	Description	Note
HALT Mode	Stop operating clock of CPU - Modest power savings	
IDLE Mode	Stop all operations of the internal circuit, except Oscillators	1
STOP Mode	Stop all operations of internal circuit, except Subclock Osc.	2
Subclock Operation Mode	Use Subclock as internal system clock	
Internal Clock Operation Mode	Use internal oscillator as internal system clock, using Clock Monitor Function	
Internal HALT Mode	Stop operating clock of CPU in Internal Clock Operation Mode	

Note 1: PLL operation does not stop in IDLE Mode. For low power consumption, stop PLL operation before entering this mode.

Note 2: Before entering STOP Mode:

- Stop PLL operation by setting PLLON = 0 in the PLL Control Register
- Change to Clock-through Mode by setting SELPLL = 0
- Then enter STOP Mode

Features of these standby modes are:

- The MCU retains RAM data and I/O states in either standby mode.
- CPU execution stop and quick restart in HALT mode achieves moderate power savings.
- Peripherals can operate in HALT mode.
- For large power savings, the main CPU clock stops in STOP mode.
- Peripherals can operate on the subclock or low-speed ring oscillator in STOP mode.
- The MCU can exit from standby by unmasked interrupt or RESET.
- The MCU provides an automatic stabilization wait for oscillators in exiting STOP mode.

Table 5. Registers Controlling Standby Operation

Register	Description of Functions	Note
Power Save Control Register (PCS)	Specifies Normal or Standby Modes	1
	Specifies Release from Standby Mode (by Interrupts)	
Power Save Mode Register (PSMR)	Specifies Standby Operation Modes, IDLE Mode or STOP Mode	
	Specifies use of Subclock (Subclock used or not used)	
Oscillation Stabilization Time Selection Register	Specifies Oscillation Stabilization Time after releasing STOP Mode	

Note 1: The PCS register is treated as a "special register," which requires a special sequence to modify. Refer to the Architecture Section of the MCU’s User's Manual for more details.

2.2.1 HALT Mode

Set HALT Mode by executing a HALT instruction. In HALT mode, the main CPU clock does not stop oscillating. The clock is not supplied to the CPU, however, so the CPU portion of the microcontroller consumes less current. The CPU clock selected before entering HALT Mode continues to operate when HALT Mode is released. Although HALT Mode does not reduce operating current as much as STOP Mode, HALT Mode is effective for restarting operations immediately after an interrupt that releases HALT Mode.

Peripherals can operate from the same clock as the CPU, in which case their clock is available during HALT mode. In this case, you can have peripherals operating during HALT mode or stop the peripherals and their external clock before entering HALT mode.

You release HALT Mode with a non-maskable interrupt (NMI), an interrupt from a watchdog timer, maskable interrupt signals that are not masked, or a reset signal.

2.2.2 IDLE Mode

Set the MCU's IDLE Mode by using the Power Save Control Register (PSC) to select Standby Mode and the Power Save Mode Register (PSMR) to select IDLE Mode.

In IDLE Mode, the clock oscillator continues to operate. The clock to the CPU and on-chip peripherals is stopped. As a result, program execution stops. The MCU retains the contents of memory before the IDLE state, while the CPU and on-chip peripherals stop operating. Any on-chip peripherals that are operating with the subclock, Ring oscillator, or external clock can continue to operate.

IDLE Mode reduces power consumption more than HALT Mode, because some of the on-chip peripheral operations are stopped. Because the main clock oscillator does not stop during IDLE Mode, you need no waiting time for oscillation stabilization when the normal operating mode is restored.

You release IDLE Mode with a non-maskable interrupt, maskable interrupt signals that are not masked, or a reset signal. After the IDLE Mode is released, normal operation resumes.

2.2.3 STOP Mode

You put the MCU in STOP Mode by using the Power Save Control Register (PSC) to select Standby Mode, then using the Power Save Mode Register (PSMR) to select STOP Mode. In STOP Mode, the subclock continues to operate, while the main clock stops. No clock is supplied to the CPU and some on-chip peripherals. As a result, program execution stops, and the MCU retains the contents of internal memory before entering STOP Mode. The on-chip peripherals that operate with the subclock, ring oscillator or external clock continue to operate.

Because STOP Mode stops operation of the main-clock oscillator, this mode reduces power consumption to a level lower than the IDLE Mode. If you do not use the subclock, ring oscillator or external clock, you minimize power consumption in STOP Mode.

You release STOP Mode by using a non-maskable interrupt (NMI), interrupt from watchdog timer, maskable interrupt signals that are not masked, or a reset signal.

3. Program Description and Specification

The demonstration program sets different clocks as the system clock and allows you to select various clock reduction and standby modes. You can see how these clock selections affect power consumption by using an ammeter to measure CPU current.

In the demonstration, one push-button selects a system clock, while a second push-button triggers execution of the demonstration program. The system clocks include:

- Full-speed clock using PLL
- System clock without using PLL (clock-through mode)
- Subclock as system clock
- Stop, idle and halt (standby modes)

3.1 Software Flowcharts

Appendix A provides the flowcharts for the demonstration programs.

Table 6. Demonstration Flowcharts

Demonstration Program	Indicator	Description of Flowchart
System Initialization		
Main.C		
Systeminit.c		System Initialization
Port.c		Initialization of Ports
Port_user.c		Initialization of ports used for demonstration, LED & SW
Watchtimer.c		Initialization, Set-up, Enable/Disable Watchtimer Used
Watchtimer_user.c		Initialization, Set-up, Enable/Disable Watchtimer Used
Watchdogtimer.c		Initialization, Set-up, Enable/Disable Watchdog Timer Used
Watchdogtimer_user.c		Initialization, Set-up, Enable/Disable Watchdog Timer Used
Special Function Register		
Write_specials.s		Initialization and write Special Function Registers used
ertE.s		
Demonstration Programs		Display Operation Indicator on the 2-digit, 7-Segment LED
clock_PLL_mode	Display "PL"	Set PLL Control Register
clock_XTAL_mode	Display "HL"	Set PCC Register to select operating frequencies
clock_SUB_mode	Display "SU"	Select Subclock as CPU clock
clock_HALT_mode	Display "hi"	Enter into HALT Mode
clock_IDLE_mode	Display "id"	Enter into IDLE Mode
clock_STOP_mode	Display "So"	Enter into STOP Mode

3.2 Applilet's Reference Drivers

You can use the Applilet to produce the basic initialization code and main function for the program, clock initialization code, watch timer, watchdog timer, and various driver codes. After the Applilet produces the basic code, you need to add additional code to customize the functioning of the program.

The system memory allocation and initialization, and set-up for the on-chip peripherals use the Applilet's reference drivers. Appendix B shows screen captures of the applicable Applilet selections and descriptions of the selections made for the demonstration. The Applilet's reference drivers used for the demonstration programs are:

- System Startup Settings
- System Watchdog Settings
- Port Subsystem Settings
  - PORT9-1 Selections
  - PORT DH Selections
  - PORT DL-1 Selections
- Timer Subsystem Selections
- Watch Timer Selections
- Interrupt Selections
  - TM000: Timer-0 Interrupt
  - WDTM1: Watchdog Timer-1
  - WT: Watch Timer
  - WTI: Watch Timer

Reference Appendix B and the Applilet's Reference Driver manual for further details.

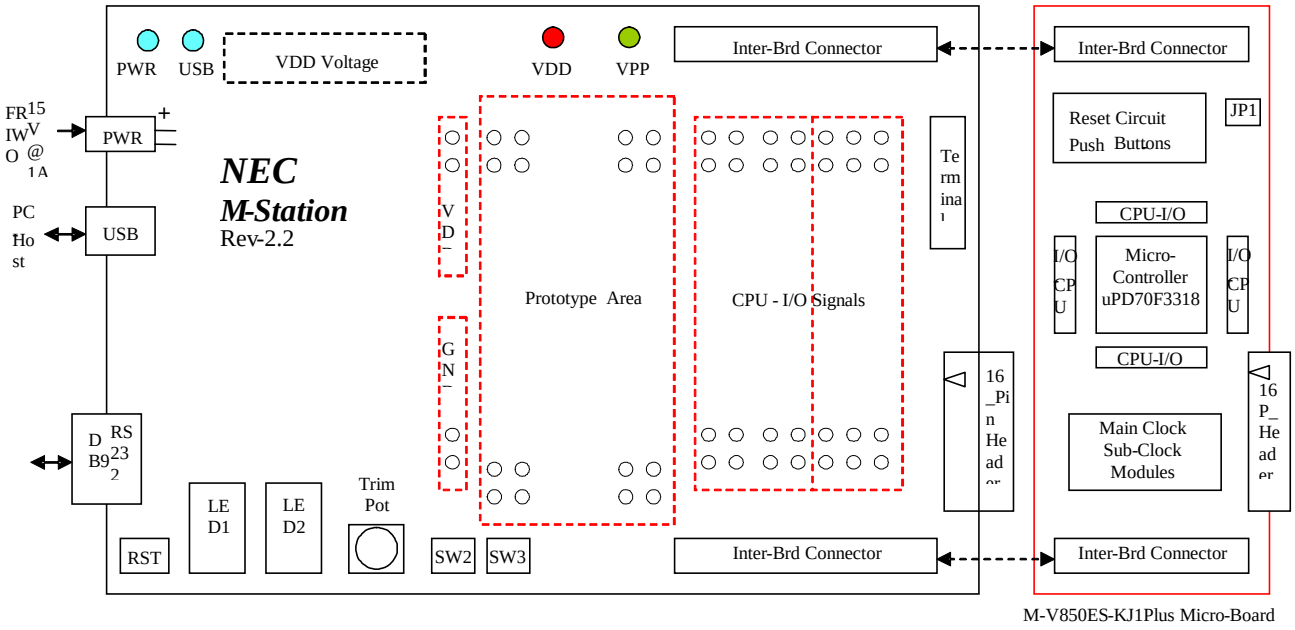
3.3 Demonstration Platform

The demonstration uses 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.

3.3.1 Resources

This demonstration uses the NEC Electronics M-Station-II with V850ES/KJ1+ Micro-Board.

Figure 2. Demonstration Hardware Setup



When the V850ES/KJ1+ Micro-Board is attached to the M-Station-II, Switch 2 (SW2) and Switch 3 (SW3) are connected to general-purpose I/O ports of the V850ES/KJ1+ microcontroller. LED1 and LED2 are also connected to general-purpose I/O ports of the microcontroller. (Refer to M-V850ES-KJ1 Plus Micro-Board schematics for connections.)

For current measurements, connect an ammeter to the Micro-board JP1, a jumper for the VDD terminal. As you select various system clocks and execute the demonstration program, you can observe the resulting current readings.

SW2 selects the system clock. When you press SW2, the selection mode number appears on the LEDs on the M-Station-II. Pressing SW3 executes the demonstration program.

3.3.2 Demonstration Program

The demonstration program shows the following modes of operation.

Table 7. Demonstration Modes

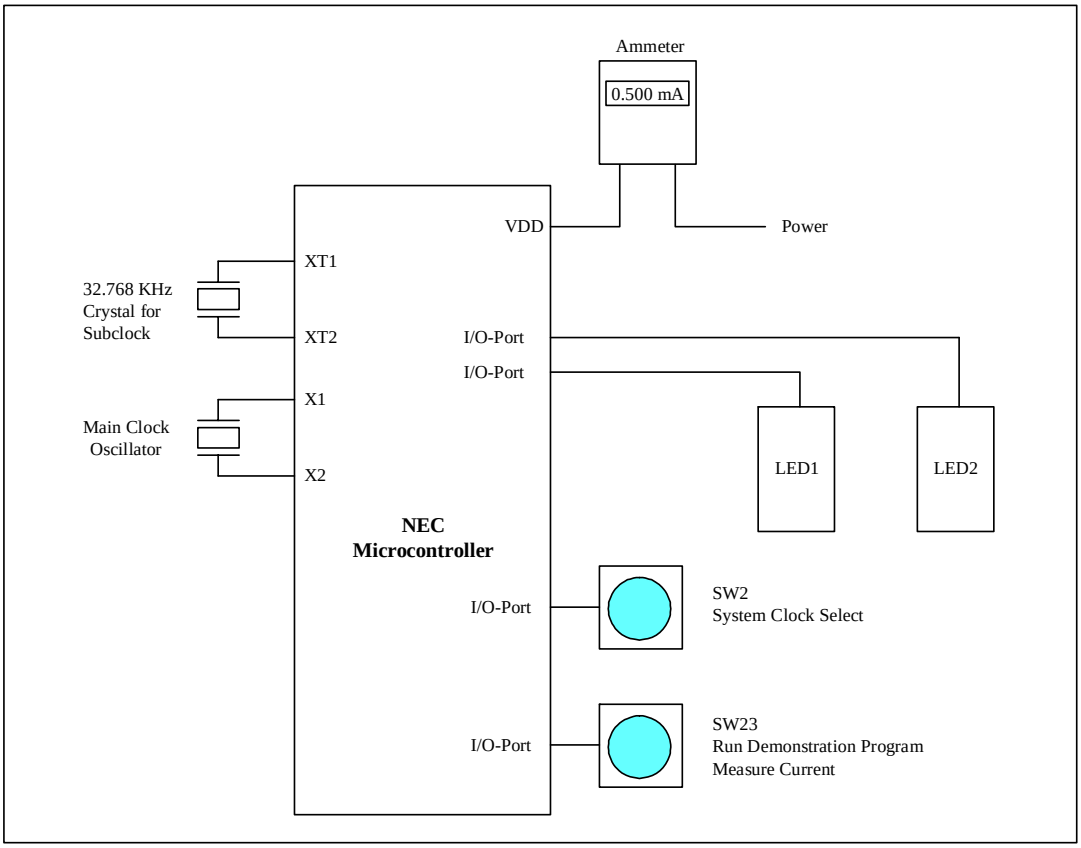
Test Number	Demonstration	Display
1	Clock running in PLL multiplier mode	PL
2	Clock running in straight-through crystal mode	HL
3	Clock running from subclock oscillator	SU
4	Clock in HALT mode	hl
5	Clock in IDLE mode	id
6	Clock in STOP mode	So

To run the demonstration, select the test number you want by pressing the left switch button (SW2). The table above lists the test numbers. After selecting the test, press the right switch button (SW3) to execute the test. The program displays the test symbol (also listed in the table) for a few seconds. Then the display clears, and the test runs. The tests operate as follows:

- Test 1—PLL multiplier. This test operates the clock at full speed. The display shows “PL” for about 5 seconds and then turns off. The processor runs at full speed (20 MHz). After measuring the operating current, press SW2 to exit and go back to the test select mode.
- Test 2—Crystal speed. In this test, the phase lock loop (PLL) multiplier is disabled, and the output of the crystal (5 MHz) feeds straight to the processor’s clock input. Since the processor is running $1/4^{th}$ as fast as in Test 1, power consumption decreases.
- Test 3—Subclock oscillator. This test runs very slow—610 times slower than high-speed mode. Press SW3 to exit this mode, and be patient because the system is not very responsive at this speed.
- Test 4—Clock halt mode. The timer runs for about 10 seconds, then takes the CPU out of HALT mode. The display shows “88” when the test completes.
- Test 5—Clock idle mode. This mode can be described as “The clock is running but no one is home.” On-chip peripherals are stopped. But since the clock is running, you can resume operation quickly, thus saving more power than in HALT mode. The display shows “88” when the test completes.
- Test 6—Clock stop mode. Press SW2 until “=6” shows on the display, then press SW3 to put the processor in STOP mode. The MCU is now running off the subclock (hibernating) with most of the peripherals stopped. Only a couple of timers are left functioning to furnish a wake-up signal. The display shows “88” when the test completes.

3.3.3 Hardware Block Diagram

Figure 3. Hardware Block Diagram



JP1 is the power terminal for the uPD70F3318 CPU on the V850ES/KJ1+ Micro-Board. To measure CPU current (and thus power consumption), install an ammeter between JP1.1 and JP1.2.

3.4 Software Modules

Reference Appendix A for program flowcharts and Appendix C for the complete software listings.

The software order is set up by the Applilet, which builds a make file when you generate the base code. The make file has the name of your project and the suffix ".mak". In the make file, the start function is given by the command line "STARTUP = crtE.o". This module is linked to load at address zero, which is where the reset vector is located. In crtE.s in section "RESET" is an instruction to jump to __start, which is where the program begins when the processor starts running.

In crtE, when __start is called, the various registers that control program operation must be initialized. These registers include a text pointer, global variable pointer, and stack pointer. The routine clears sbss, bss and IRAM areas of memory. The routine also loads variables __argc into r6, and pointer __argv into r7.

Then main() is called, which is the normal start of a C program. Although registers r6 and r7 have been loaded, it is unusual for an embedded program to take input parameters.

When main() is entered, it immediately calls SystemInit() to set up the predefined variables for the C program and initialize all of the peripheral subsystems. On return, main() starts the watch timer and interval timer 1. The routine displays the current test number, then enters an infinite loop, checking for input from push-buttons SW2 and SW3.

The watch timer blinks the left LED decimal point, changing state every 50 interrupts. The interval timer T01 blinks the right LED decimal point, changing state on every interrupt.

When you press SW2, main() increments the test number. The test number continues to increment until the maximum number of tests is reached, then the number restarts at 1.

Pressing SW3 calls the function corresponding to the current test number (as set by SW2). The flowcharts show the operation of each test function.

Table 8. Files for Power-Down Demonstration

File	Generated by Applilet	Modified by User
cretE.s	Applilet	
systeminit.c	Applilet	
System_user.c	Applilet	
main.c	Applilet	modified
int.c	Applilet	
int_user.c	Applilet	
port.c	Applilet	
port_user.c	Applilet	modified
timer.c	Applilet	
timer_user.c	Applilet	modified
watchdogtimer.c	Applilet	
watchdogtimer_user.c	Applilet	modified
watchtimer.c	Applilet	
watchtimer_user.c	Applilet	
write_special.s		modified
macrodriver.h	Applilet	
int.h	Applilet	
port.h	Applilet	
watchtimer.h	Applilet	
timer.h	Applilet	
watchdogtimer.h	Applilet	

4. Appendix A — Software Flowcharts

4.1 main.c

Figure 4. main.c Part 1

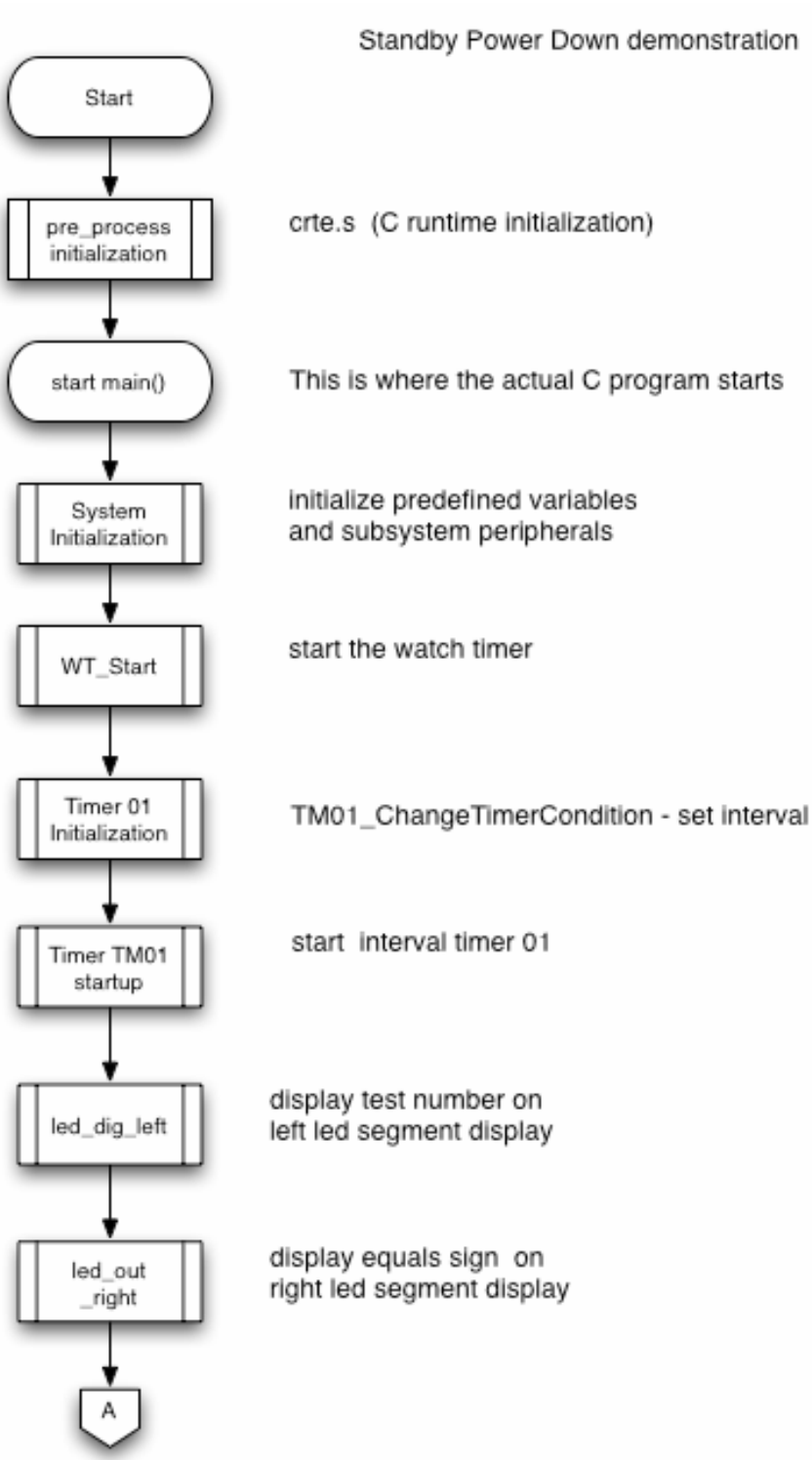


Figure 5. main.c Part 2

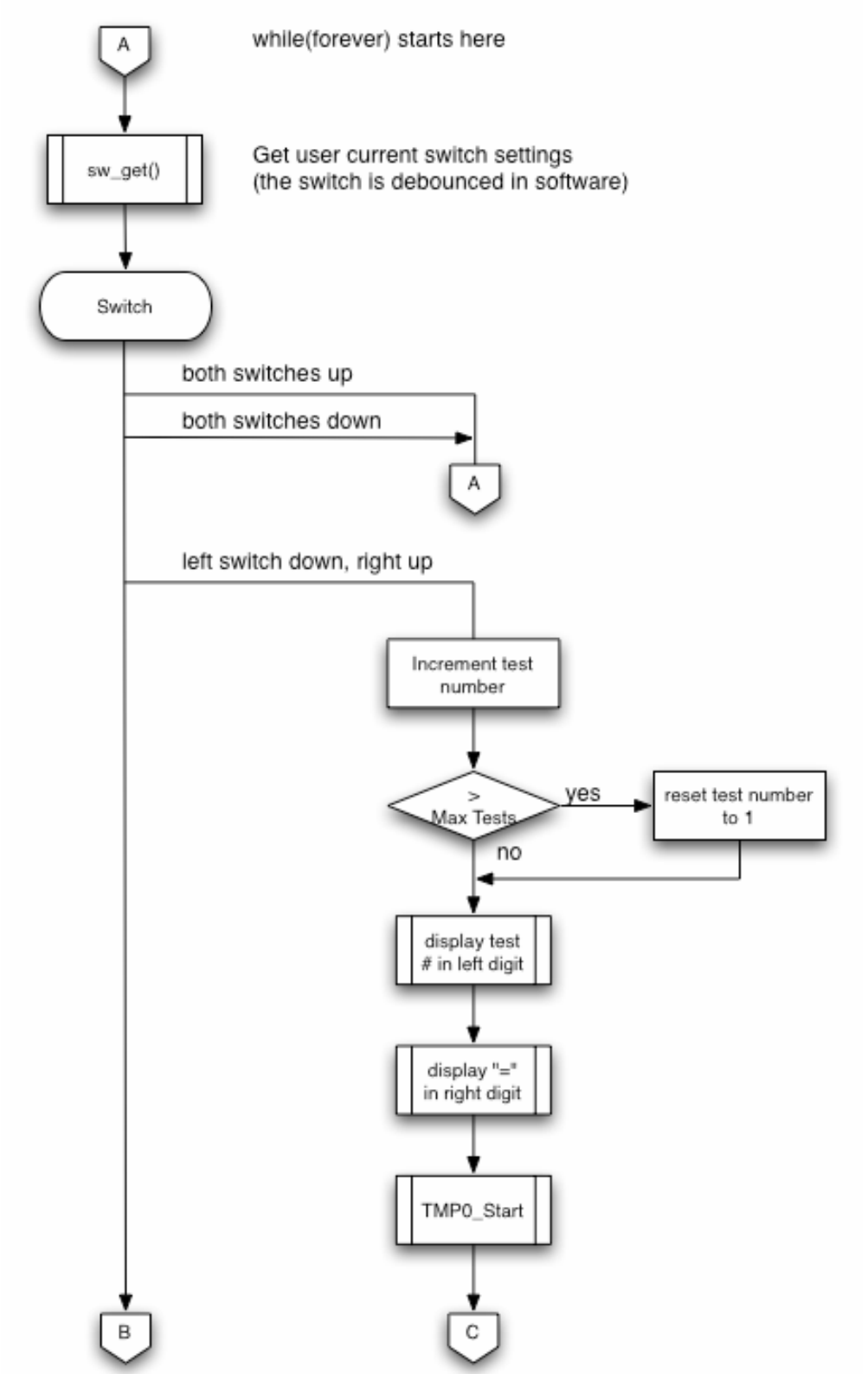


Figure 6. main.c Part 3

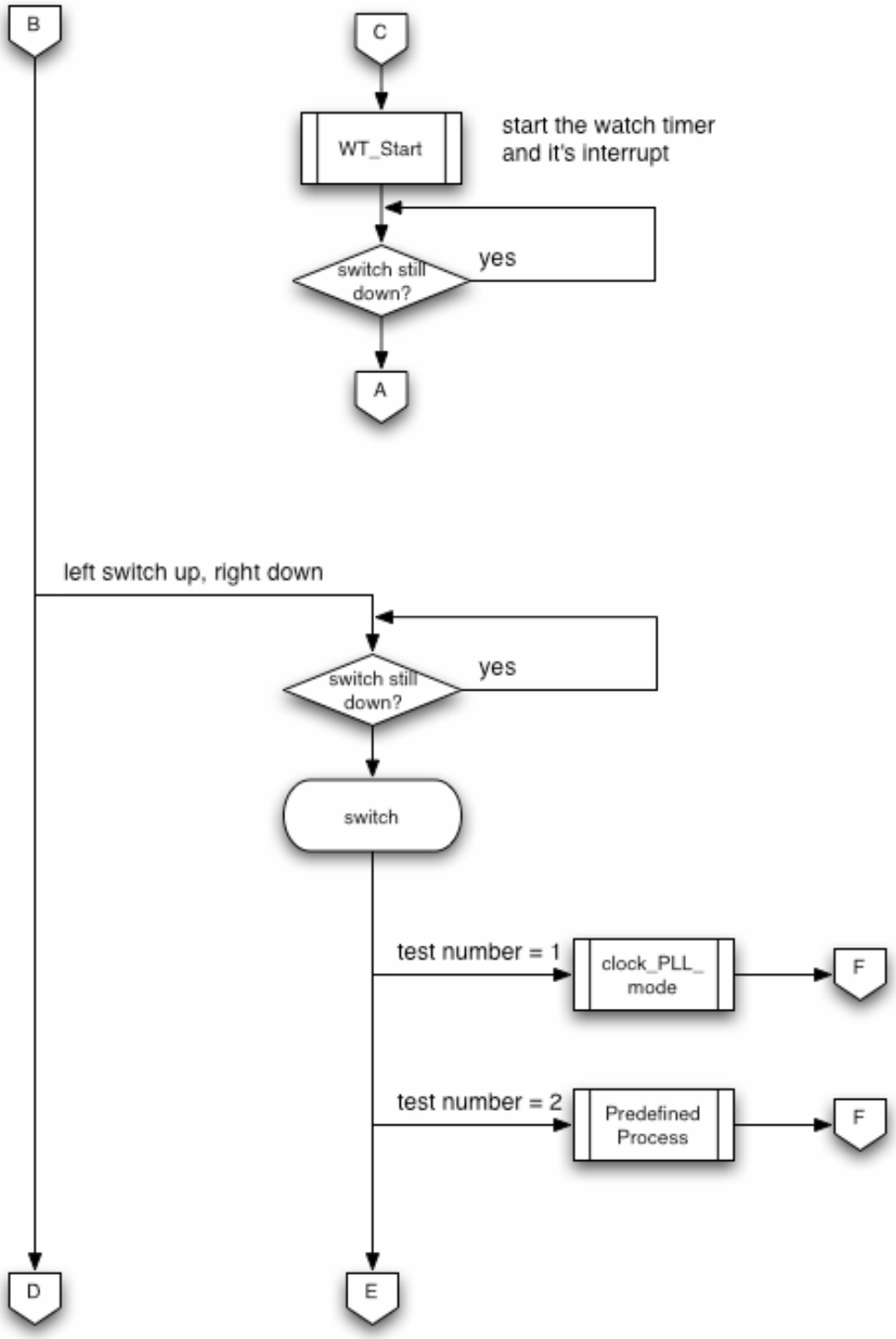
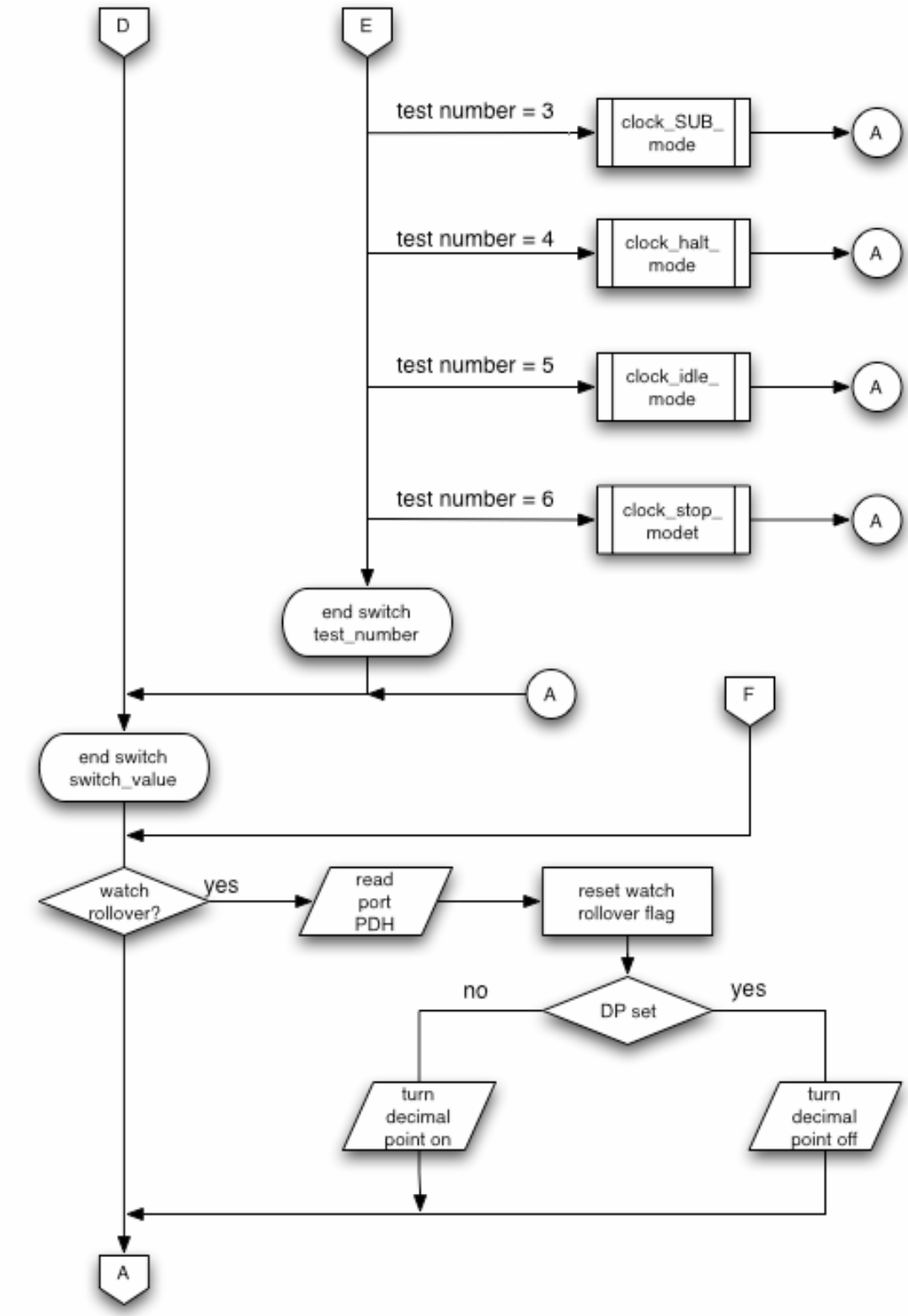
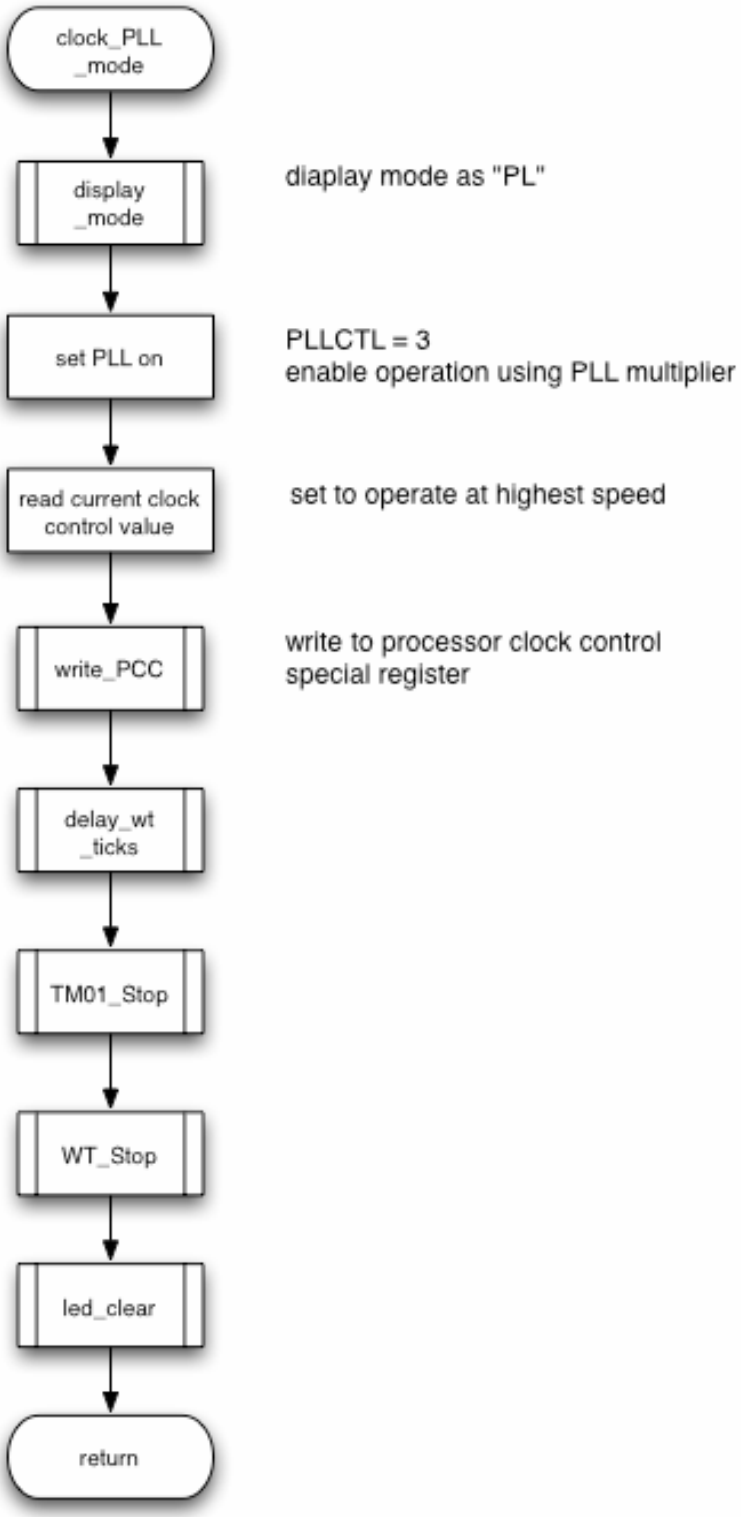


Figure 7. main.c Part 4



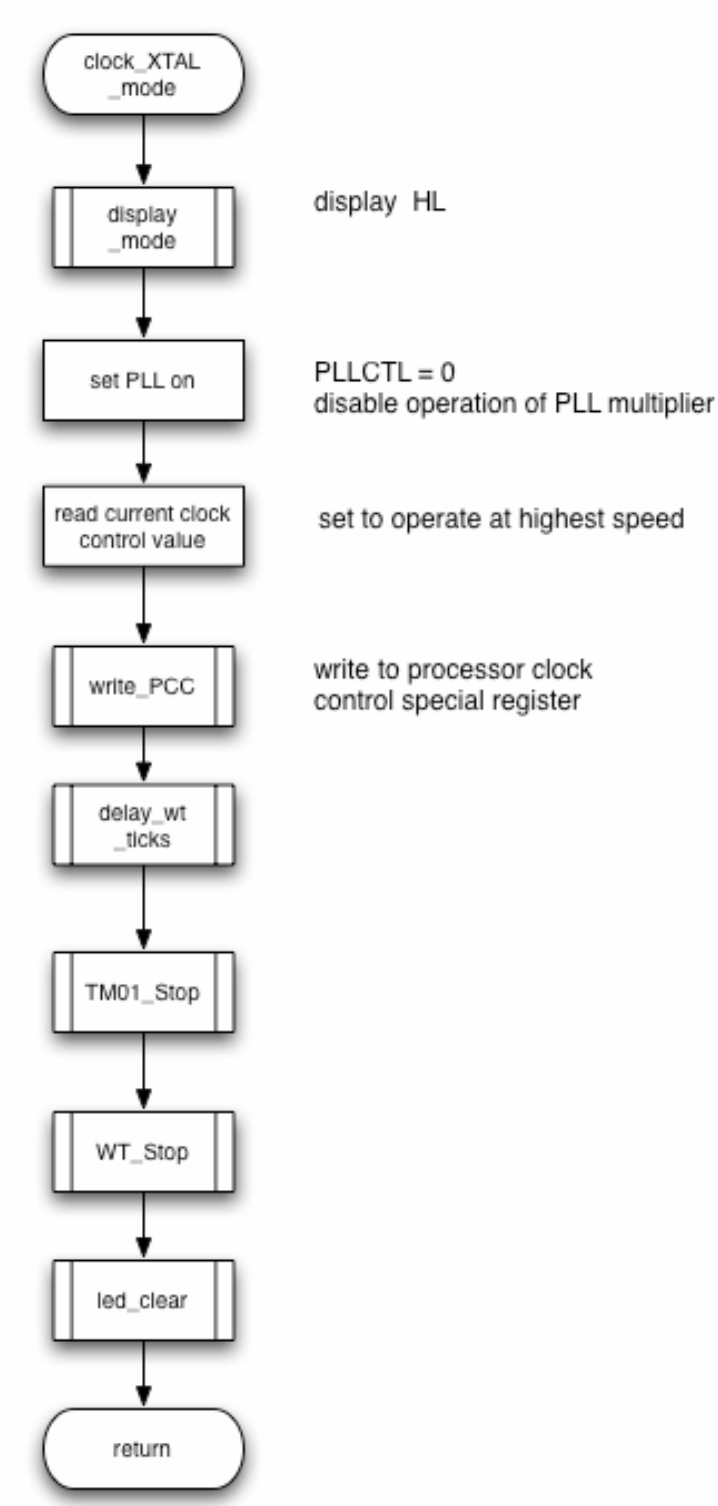
4.2 clock_PLL_mode

Figure 8. clock_PLL_mode



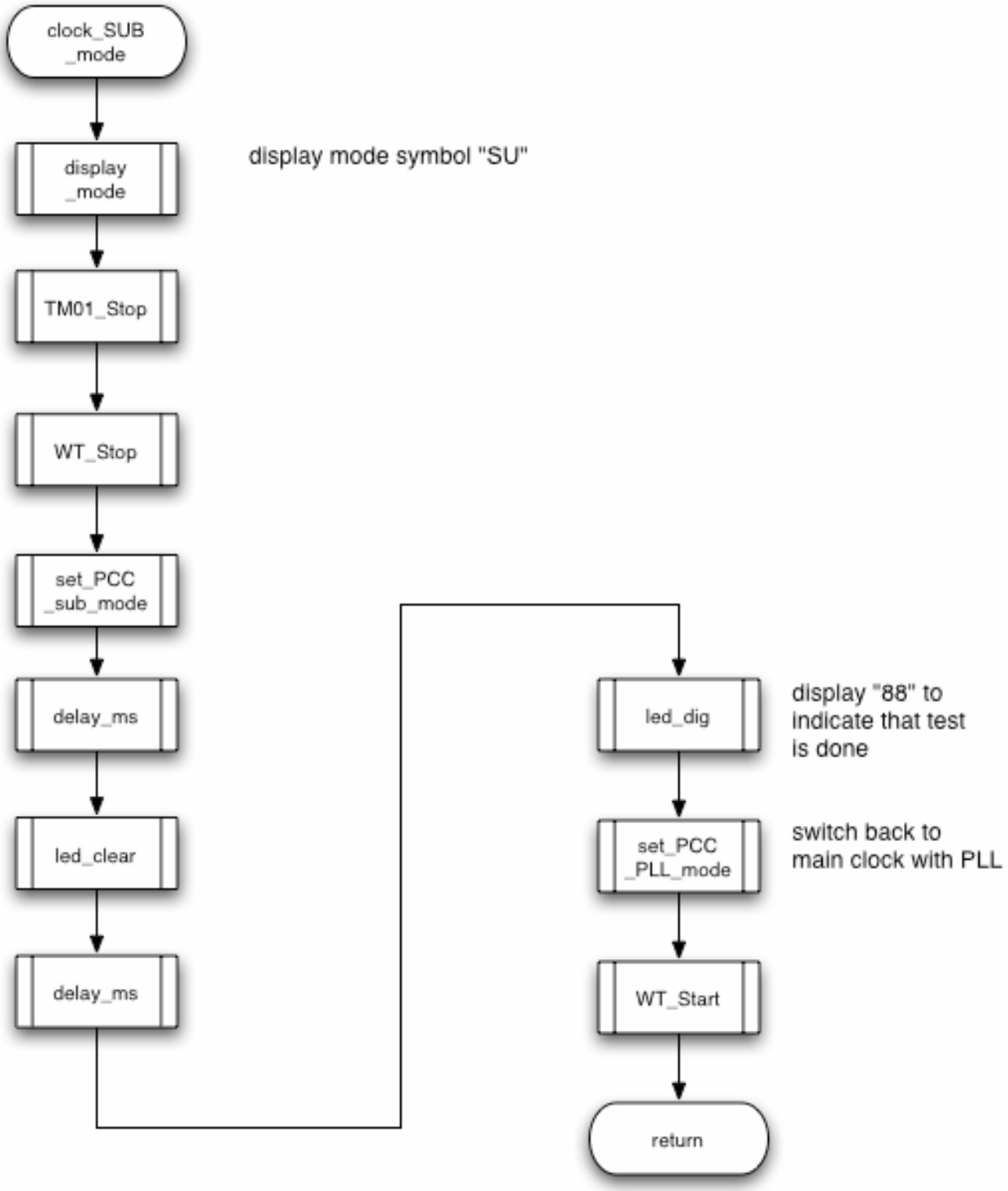
4.3 clock_XTAL_mode

Figure 9. clock_XTAL_mode



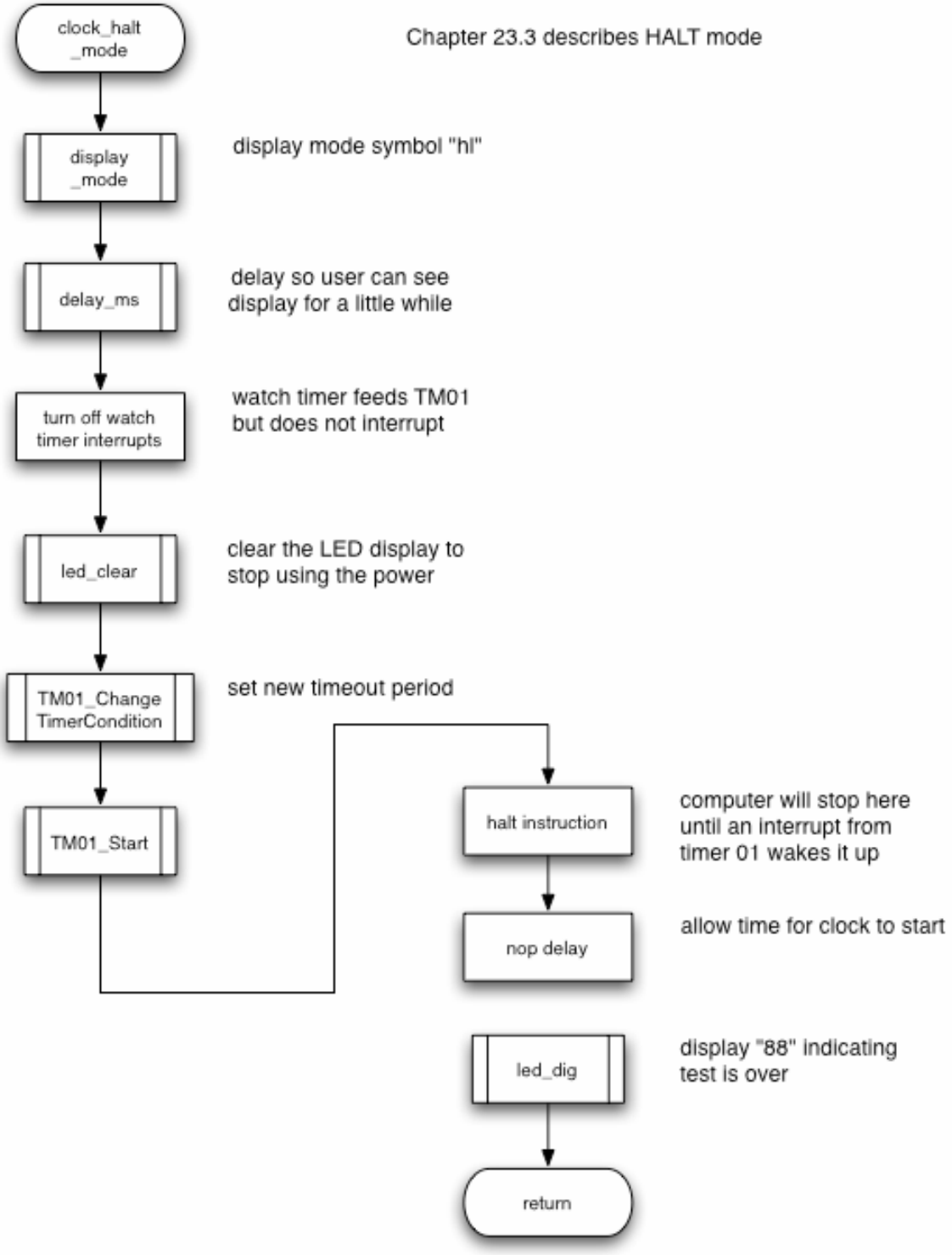
4.4 clock_SUB_mode

Figure 10. clock_SUB_mode



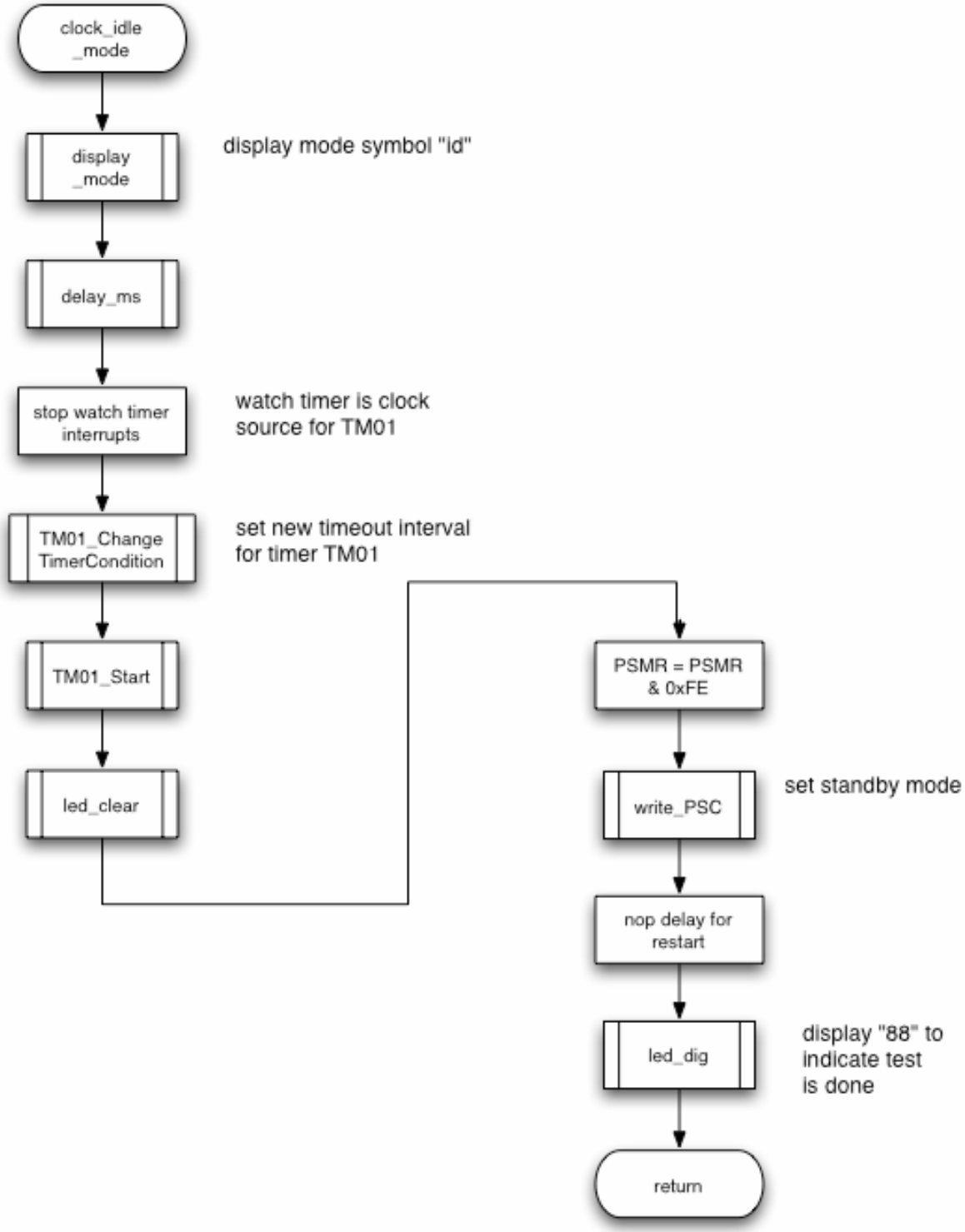
4.5 clock_HALT_mode

Figure 11. clock_HALT_mode



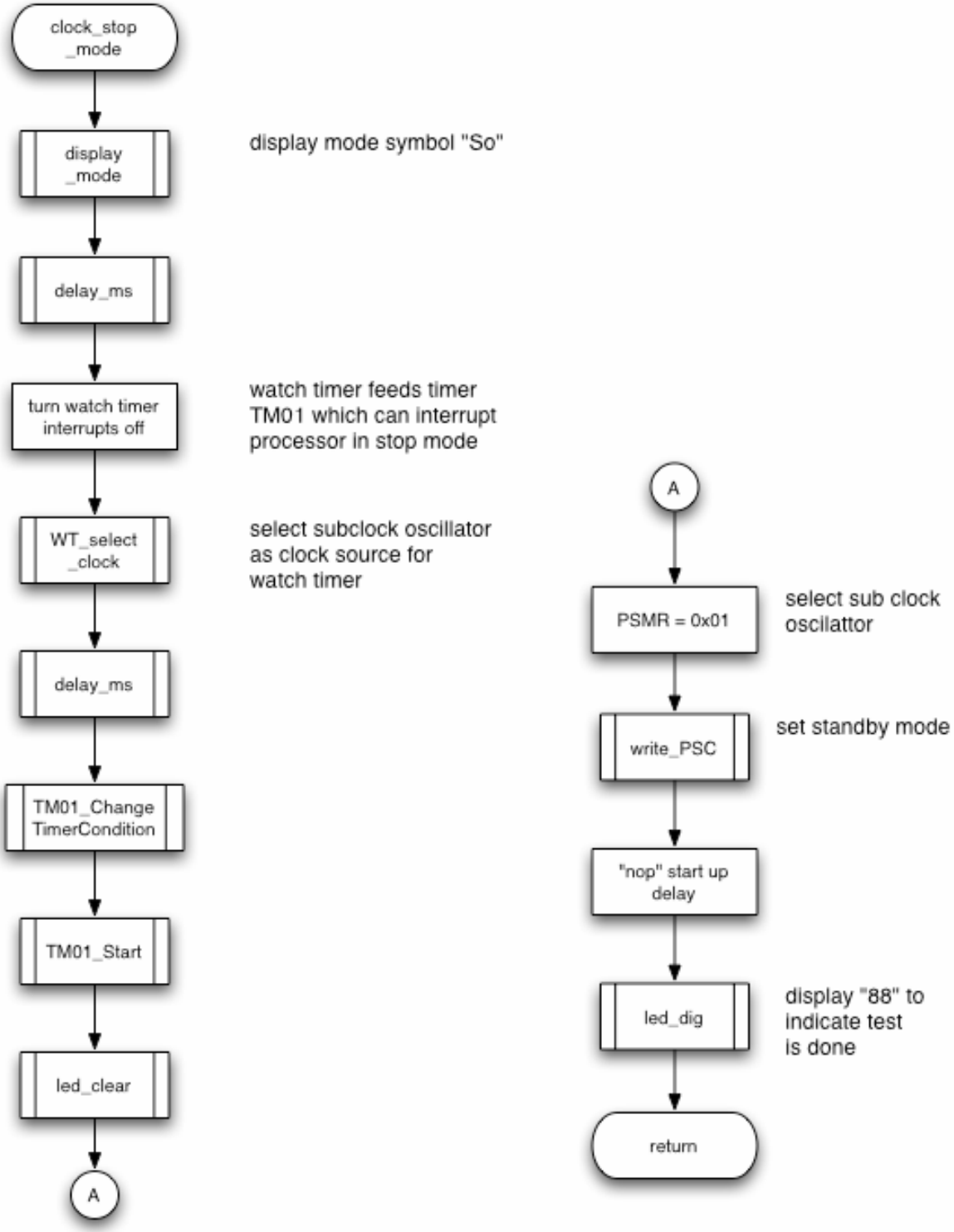
4.6 clock_IDLE_mode

Figure 12. clock_IDLE_mode



4.7 clock_STOP_mode

Figure 13. clock_STOP_mode



4.8 systeminit.c

Figure 14. systeminit.c Part 1

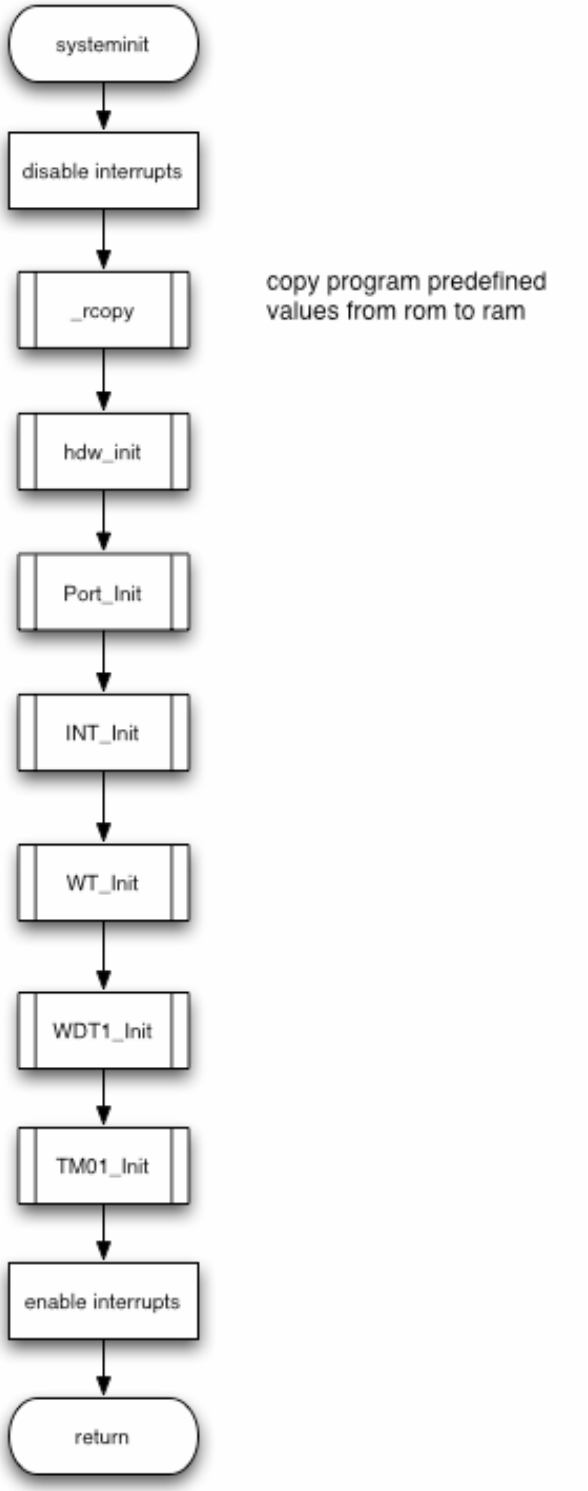
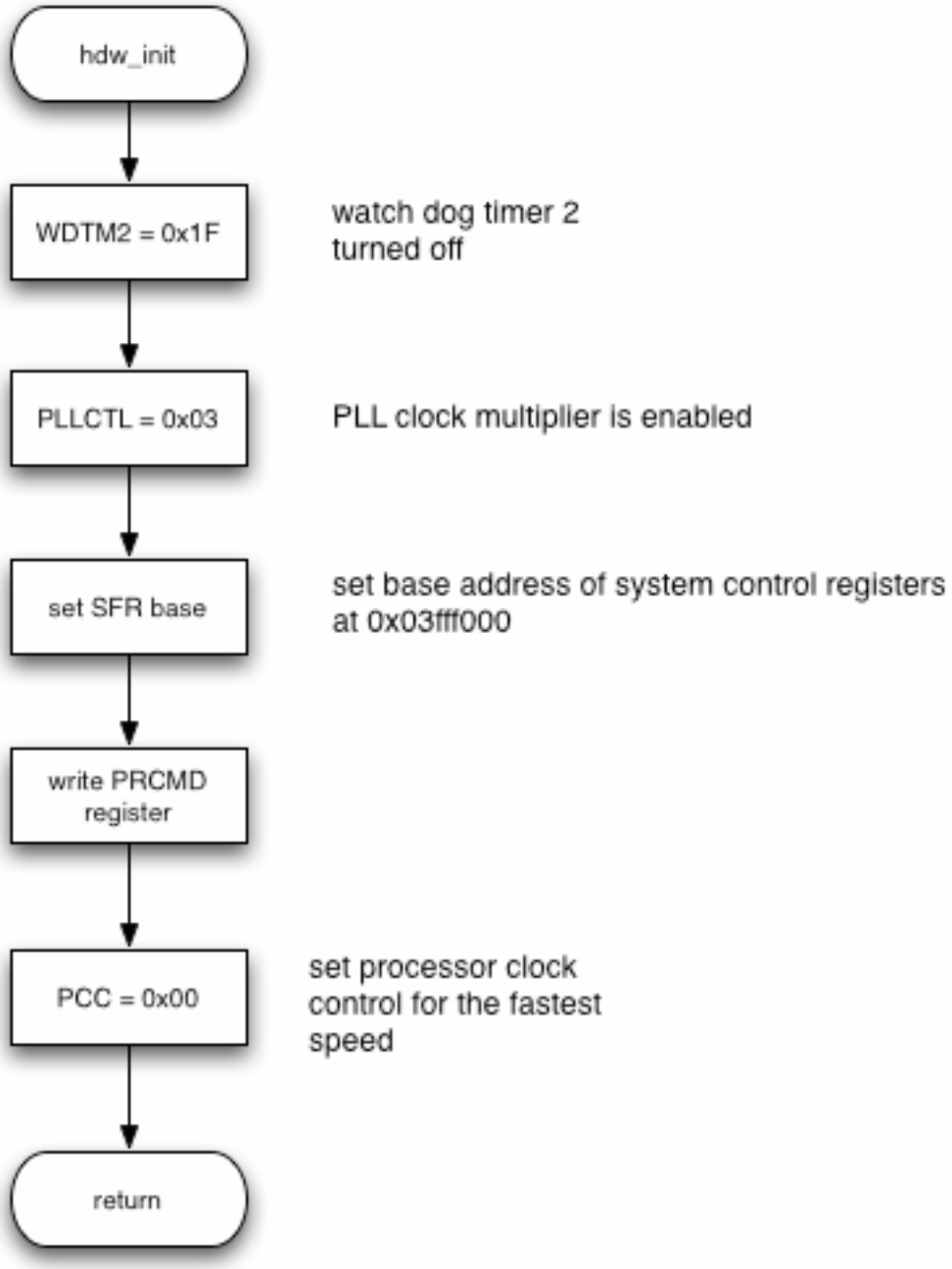
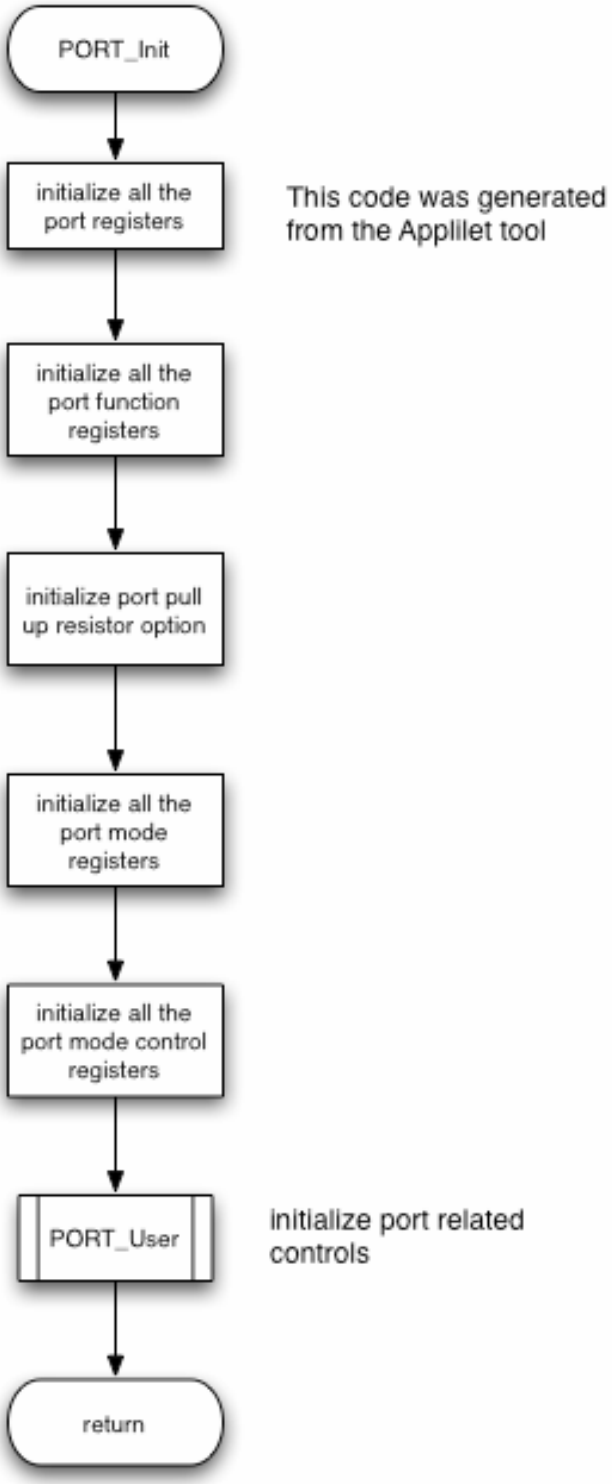


Figure 15. systeminit.c Part 2



4.9 port.c

Figure 16. port.c



4.10 port_user.c

Figure 17. port_user.c Part 1

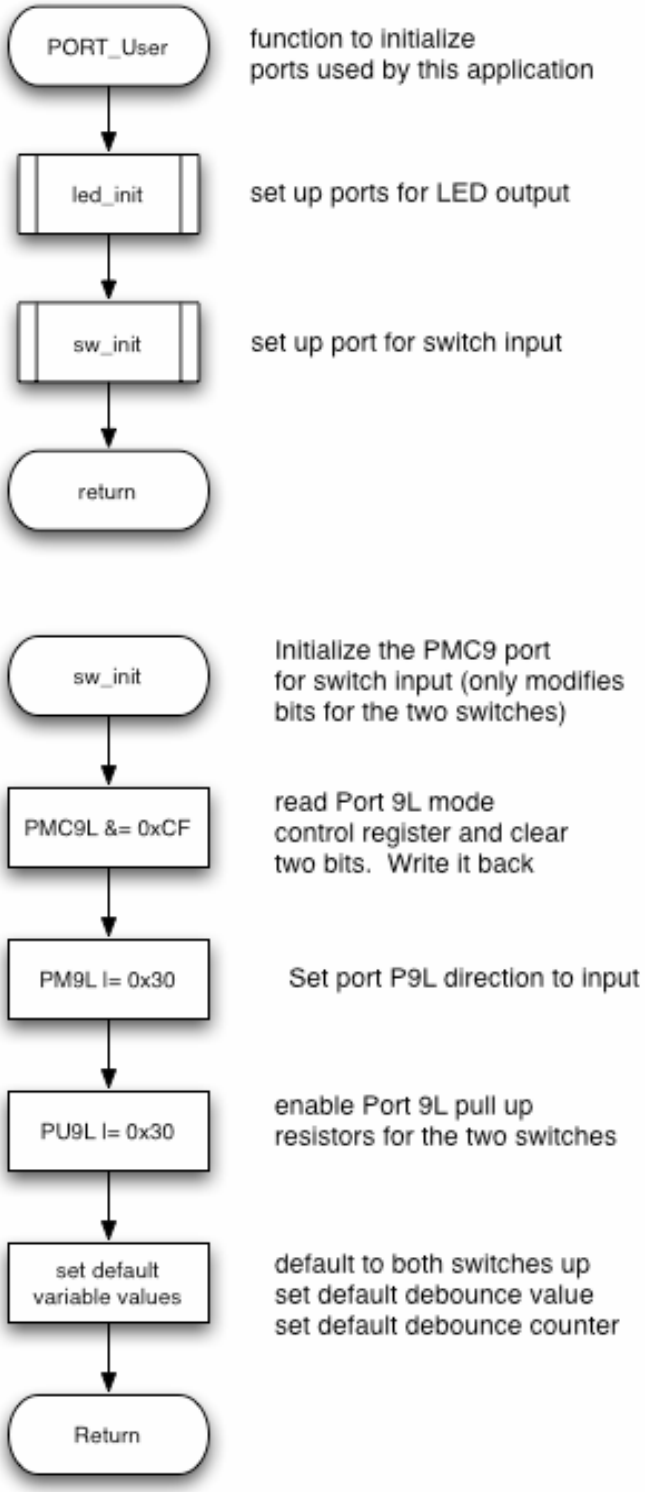
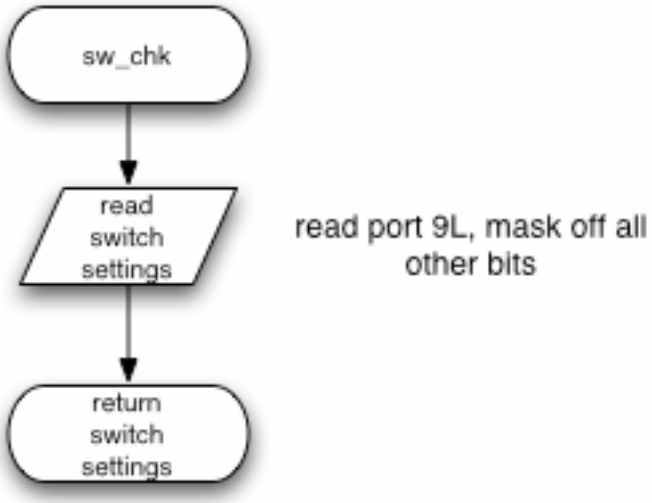


Figure 18. port_user.c Part 2



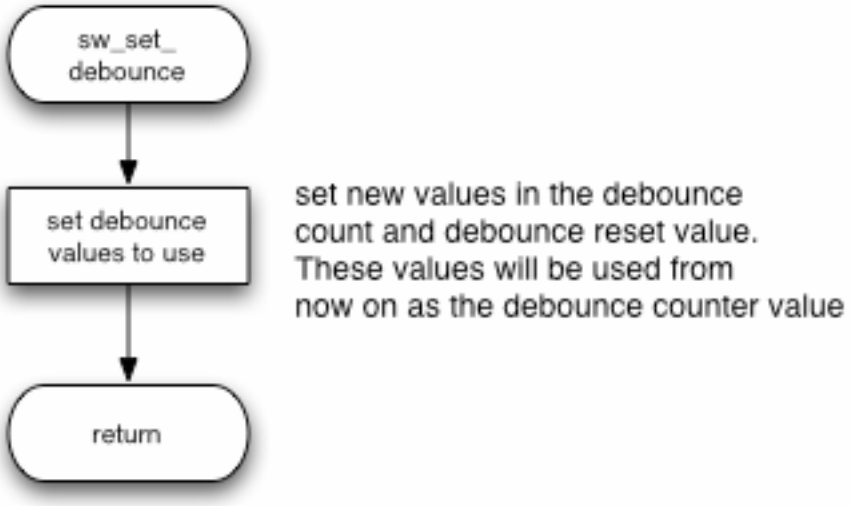
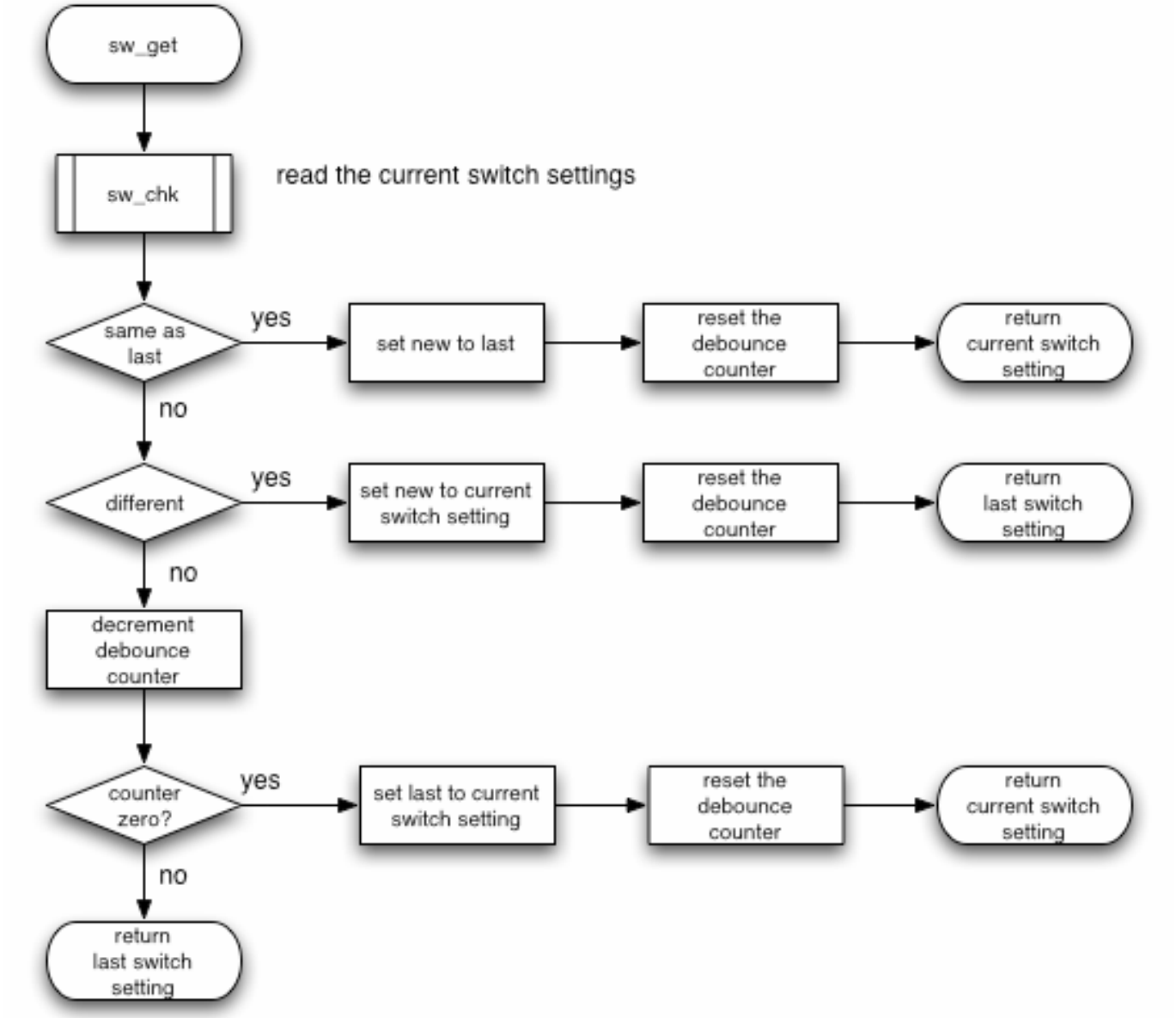


Figure 19. port_user.c Part 3



4.11 watchtimer.c

Figure 20. watchtimer.c Part 1

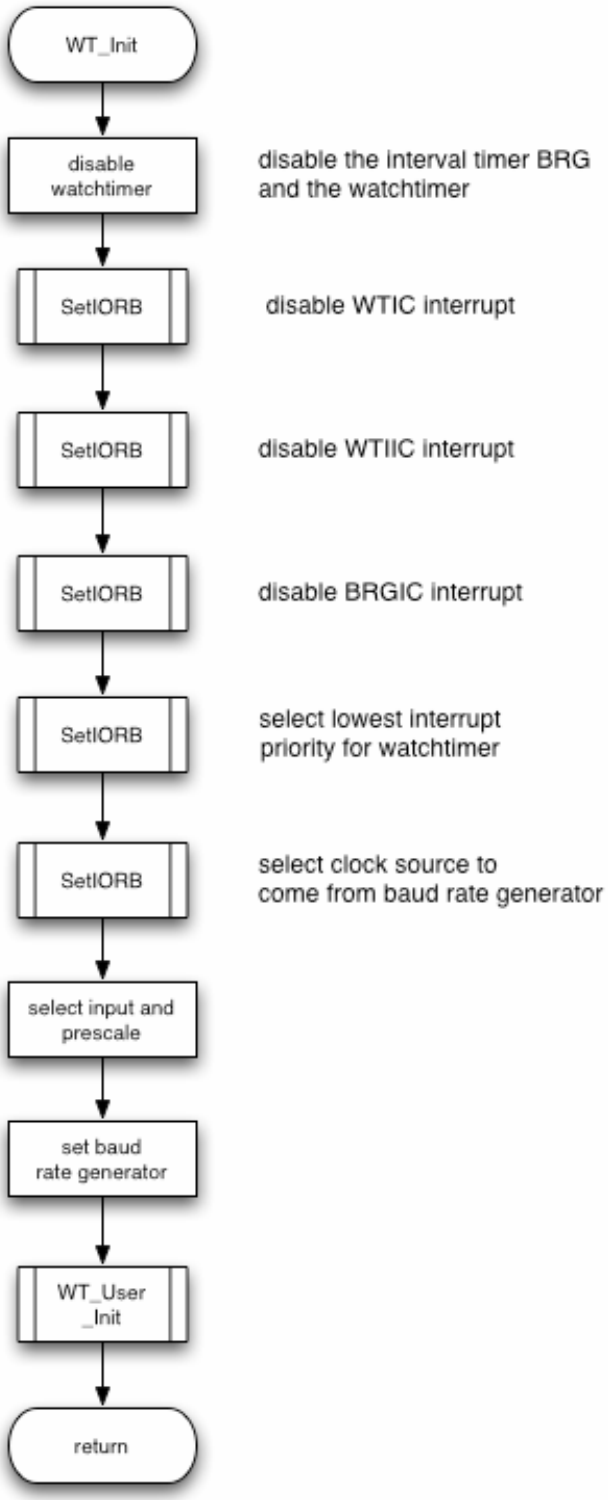
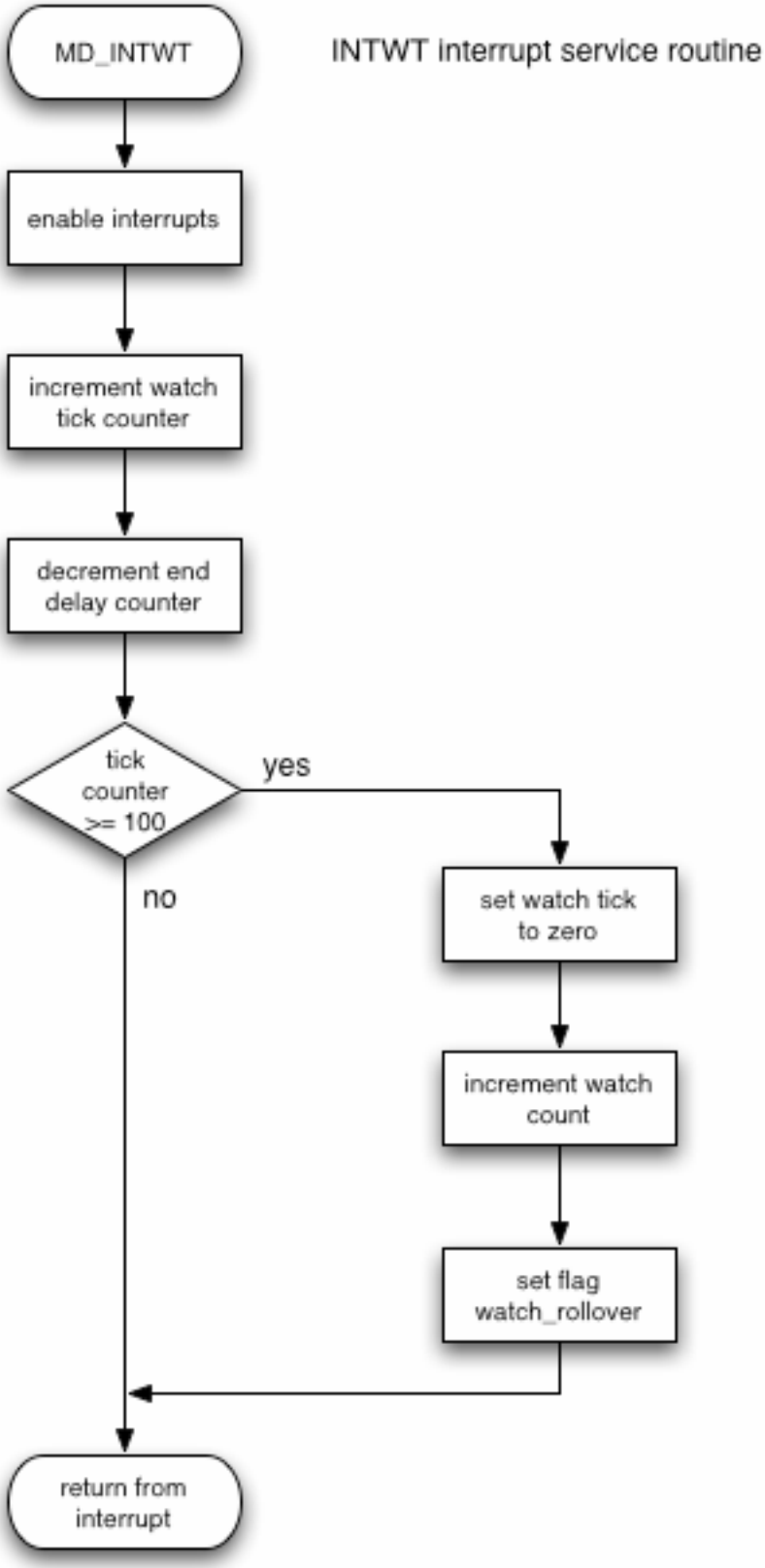


Figure 21. watchtimer.c Part 2



4.12 watchtimer_user.c

Figure 22. watchtimer_user.c Part 1

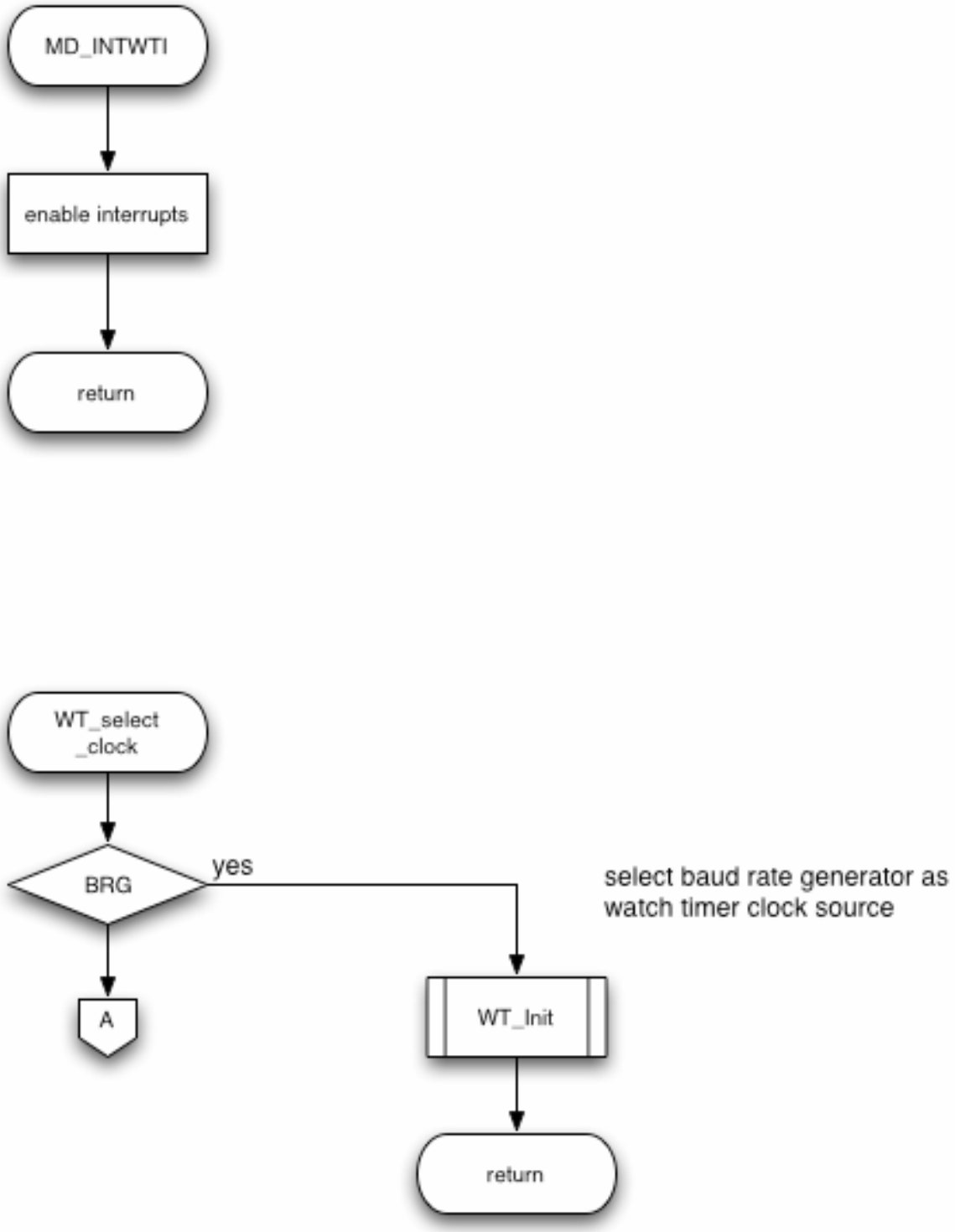
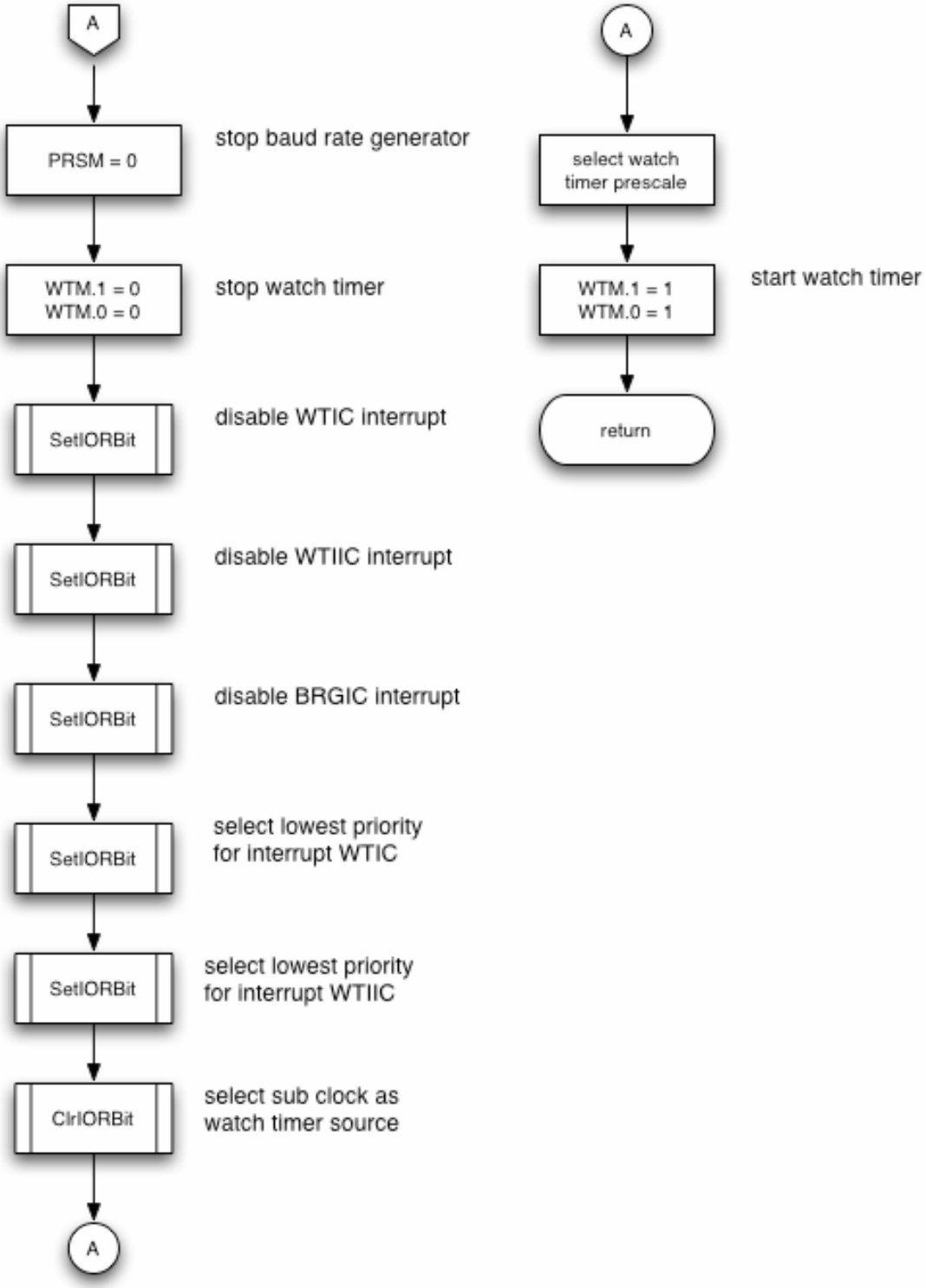
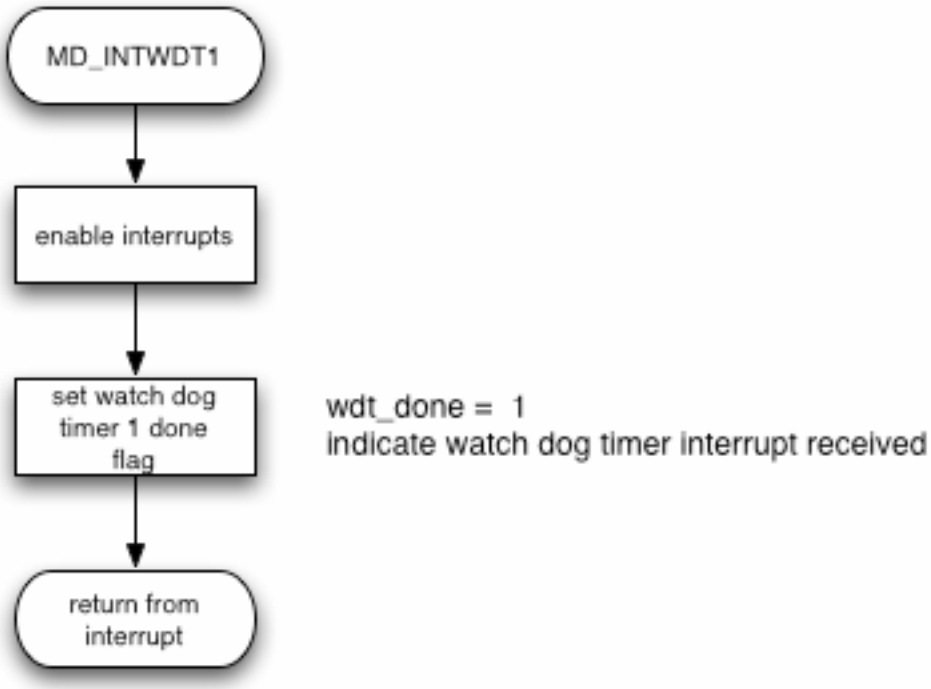


Figure 23. watchtimer_user.c Part 2



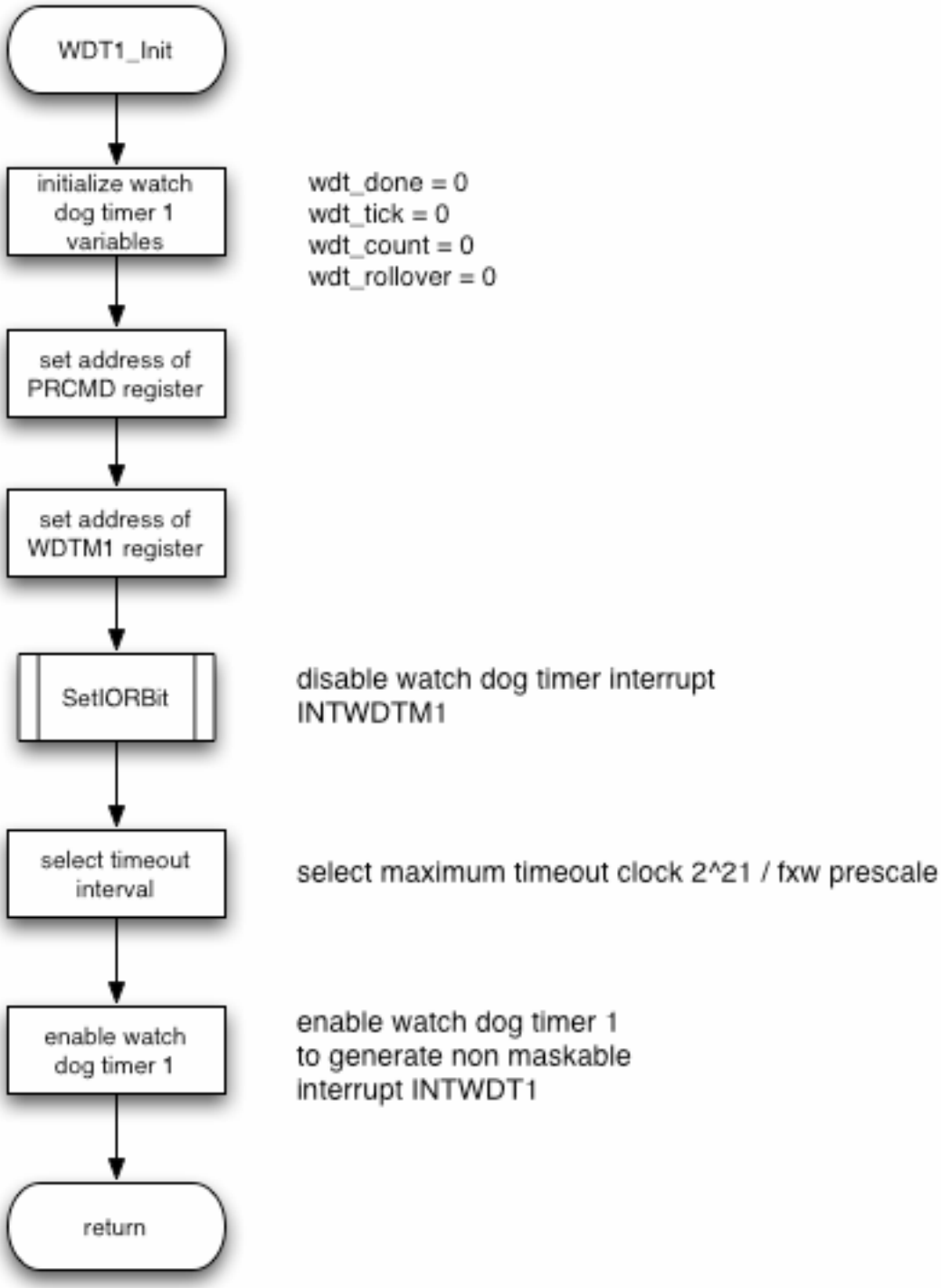
4.13 watchdogtimer_user.c

Figure 24. watchdogtimer_user.c



4.14 watchdogtimer.c

Figure 25. watchdogtimer.c



4.15 write_special.s

Figure 26. write_special.s Part 1

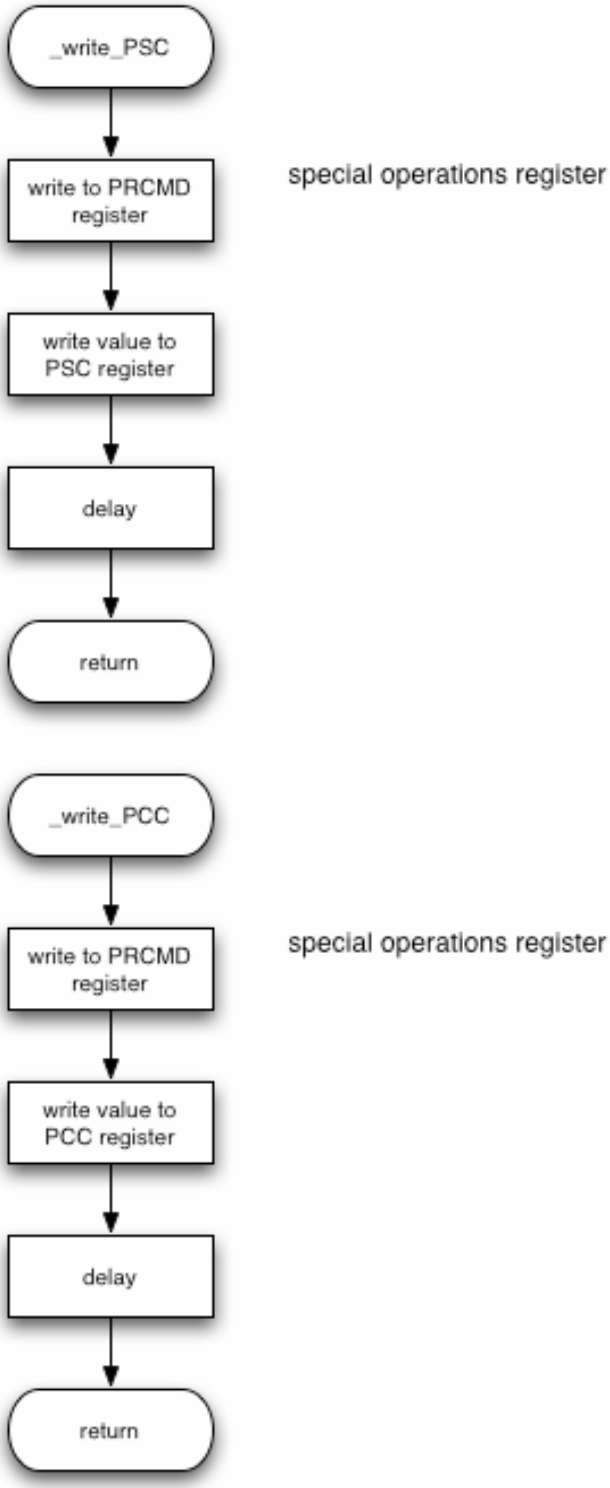


Figure 27. write_special.s Part 2

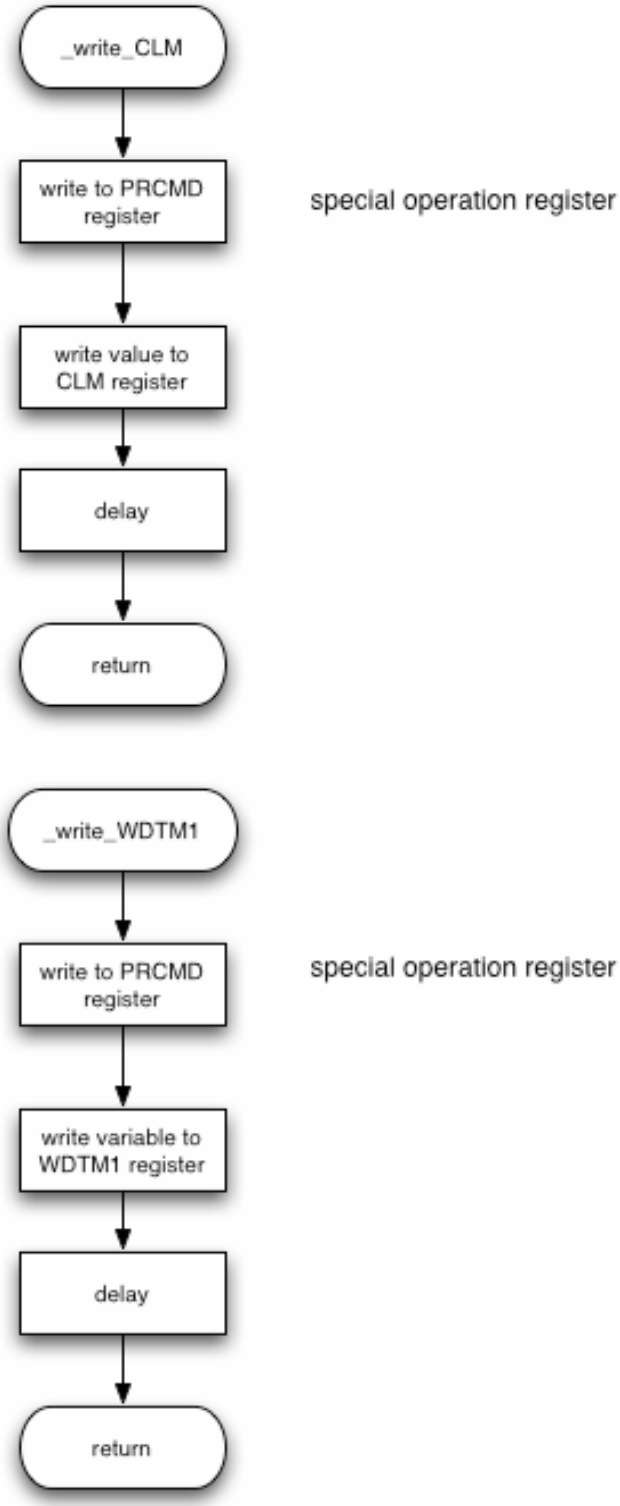


Figure 28. write_special.s Part 3

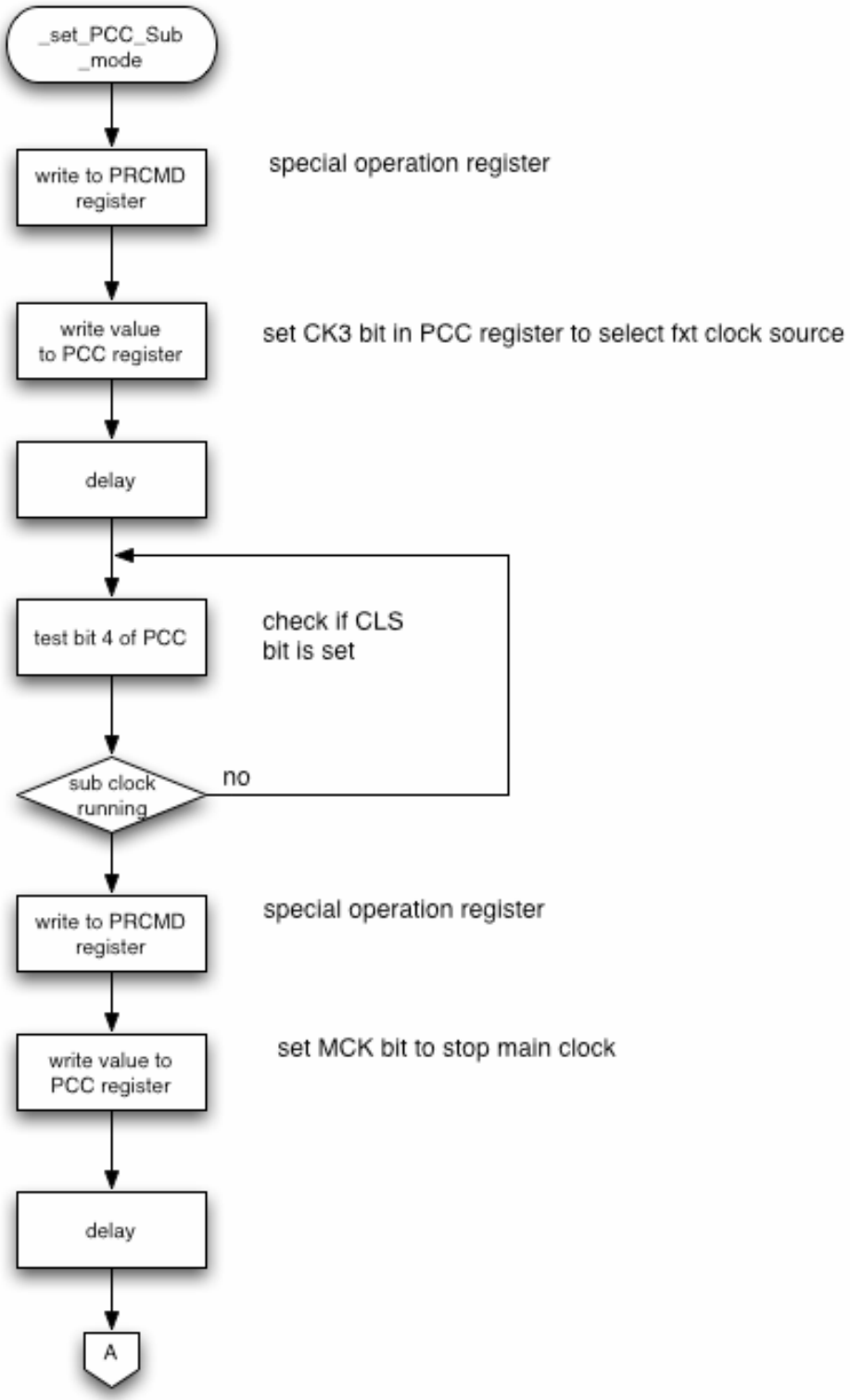
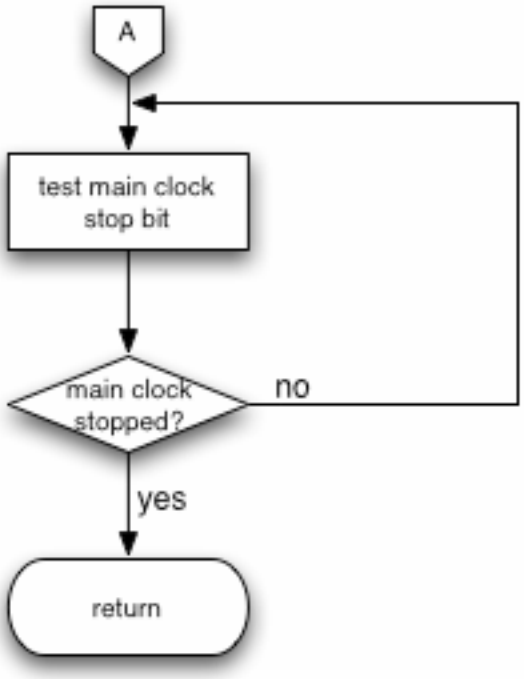


Figure 29. write_special.s Part 4



4.16 crtE.s

Figure 30. crtE.s Part 1

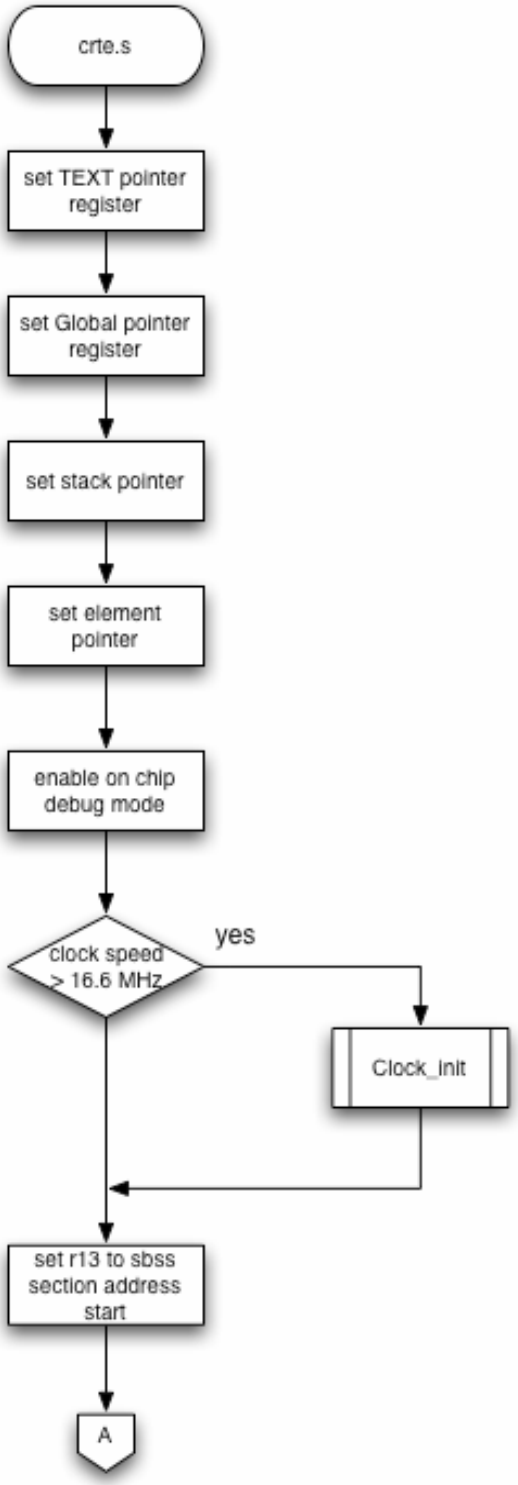
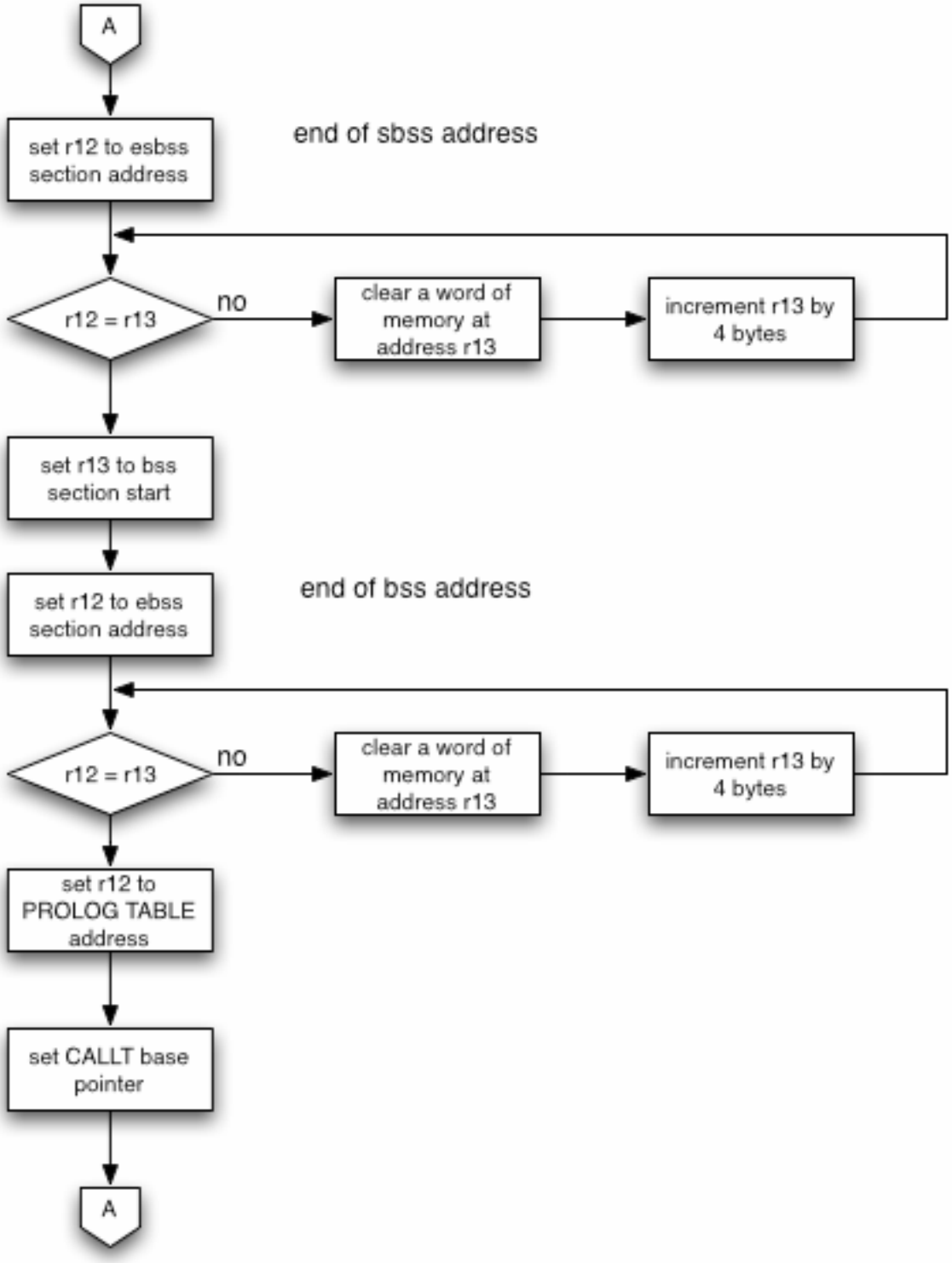


Figure 31. crtE.s Part 2



5. Appendix B – Applilet Tool

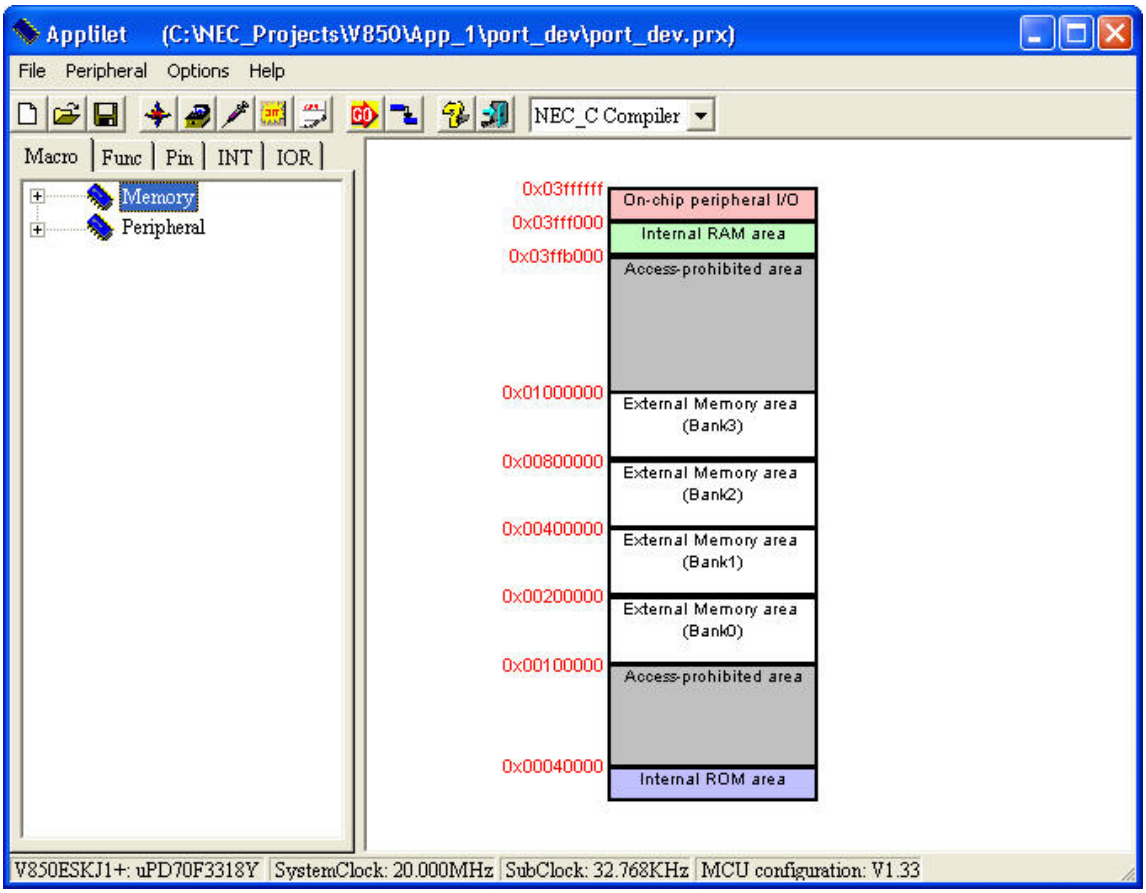
The Applilet tool provides a GUI-based environment for generating a base framework of code. As shown in the following series of screen captures, you can select the features needed for your application and get a quick start in creating your code. The screens shown below are only for the features applicable to this demonstration.

5.1 System Memory

To enable the Applilet to include the options available for your MCU, you need to import the MCU information. As shown at the bottom of the screen below, this demonstration uses the V850ESKJ1+ uPD70F3318Y MCU. This screen is just informational and shows how the internal memory is laid out in the processor. Other Option menu items are:

- Options | Compilers | NEC Compiler
- Options | Source Code Language | C Language
- Options | Import MCU

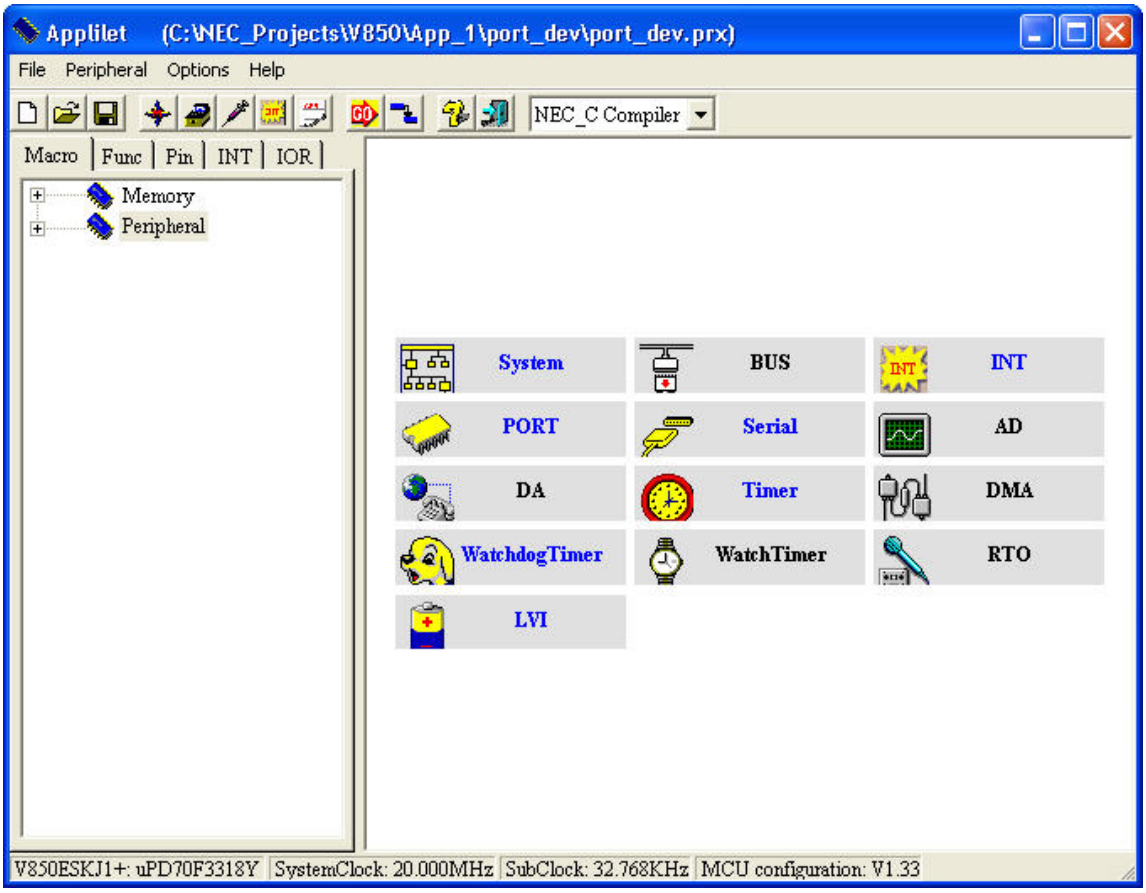
Figure 32. Applilet Screen Showing Memory Layout



5.2 System Peripherals

The Applilet screen below is your starting point. From this screen you select each of the peripheral subsystems. It is best to start with the system first since the clock selections made here are used by the other peripherals.

Figure 33. Selecting Peripherals



5.3 System

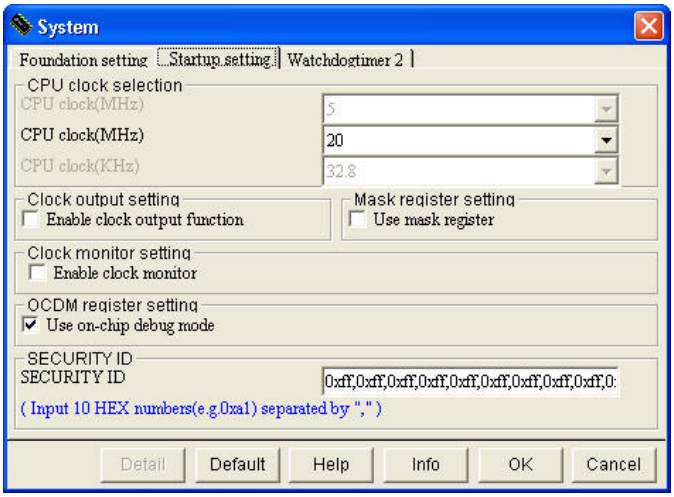
5.3.1 System Foundation settings

After you click **System**, the Applilet's **Foundation setting** tab allows you to select various clock operations. When you start the demonstration, you want to be in high-speed mode, so select the main clock option and the PLL multiplier. You also want to be flexible, so allow the ring oscillator to be stopped by software. The main oscillator selection should be based on the crystal used to drive the main clock of the microcontroller. The same is true for the sub-clock oscillator crystal or whatever method you use to generate the low frequency for that clock source.

5.3.2 System Startup Settings

When you click on the **Startup setting** tab, the Applilet shows **CPU clock(MHz)** at "20" because you previously selected a 5MHz main clock with PLL on. Under **OCDM register setting**, allow on-chip debug mode since you are working with development code. To be safe, leave the Security ID as all ONEs; you do not want to pick a strange value and then forget what it was.

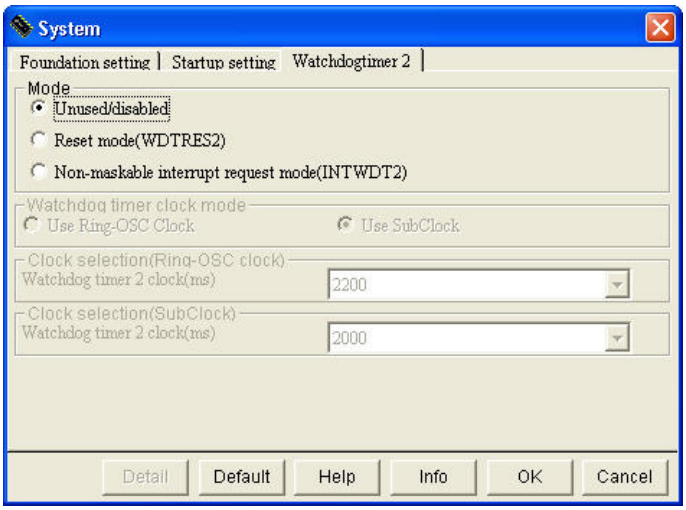
Figure 34. System Startup Settings



5.3.3 System Watchdog Settings

After clicking the **Watchdogtimer 2** tab, select the non-maskable interrupt. Select the ring oscillator as the clock source for Watchdog Timer 2. Select the longest delay possible from the clock selection pull down menu.

Figure 35. System Watchdog Settings



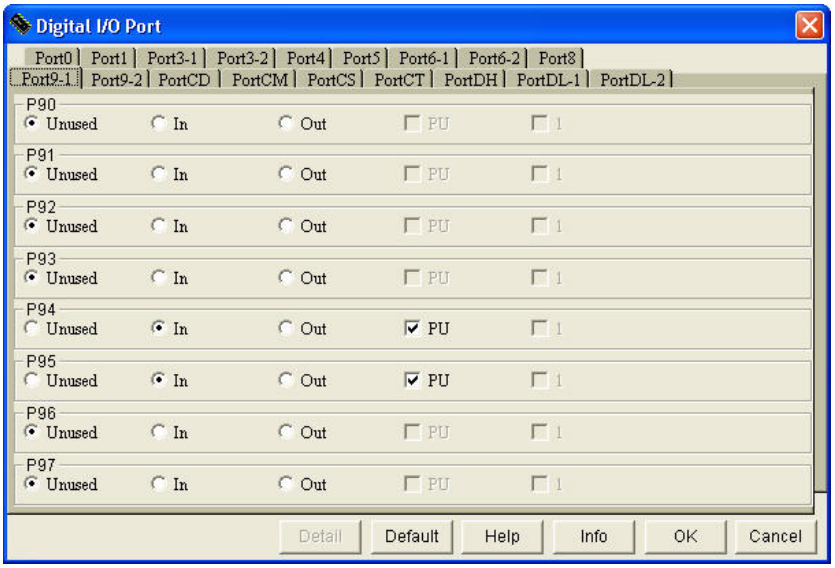
5.4 Port Subsystem

Clicking the **Port** button on the main Applilet screen gives you access to port settings.

5.4.1 Port 9-1 Selection

Port 9-1 connects to the push-buttons (SW2 and SW3). Note in the figure below that the pull-ups are enabled. With this setting, you do not need extra pull-up resistors on the switches. Pressing the switch pulls the input to ground. Pin P94 connects to SW2 on the M-Station (the left switch) and P95 connects to SW3 (the right switch).

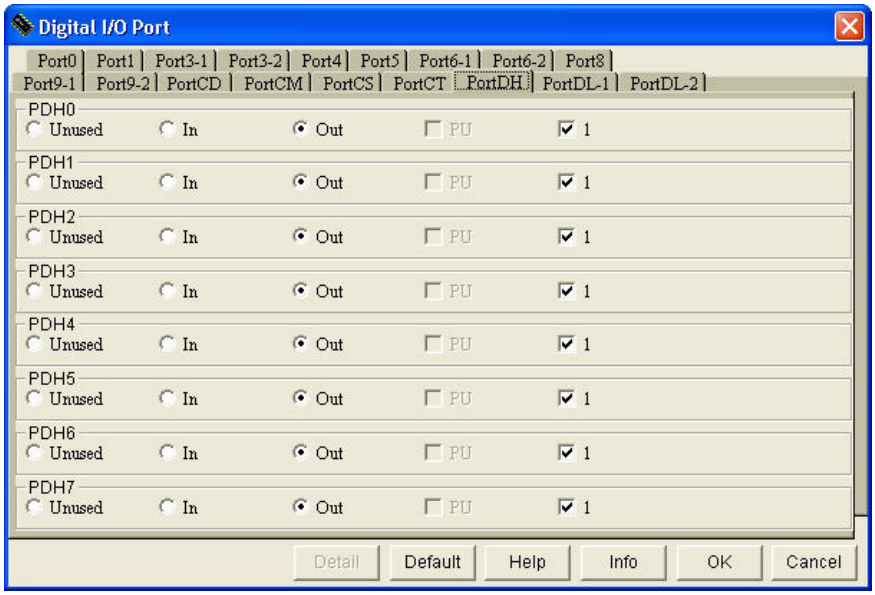
Figure 36. Port 9-1 Selection



5.4.2 Port DH Selection

Go to the tab for Port DH to set up this port as an output for the right LED. When the port pin outputs a 1, the LED segment turns off. Outputting a 0 causes the port pin to sink current, and the LED turns on. On each pin, select a 1 for the initial output so that the LEDs do not come on until they are supposed to.

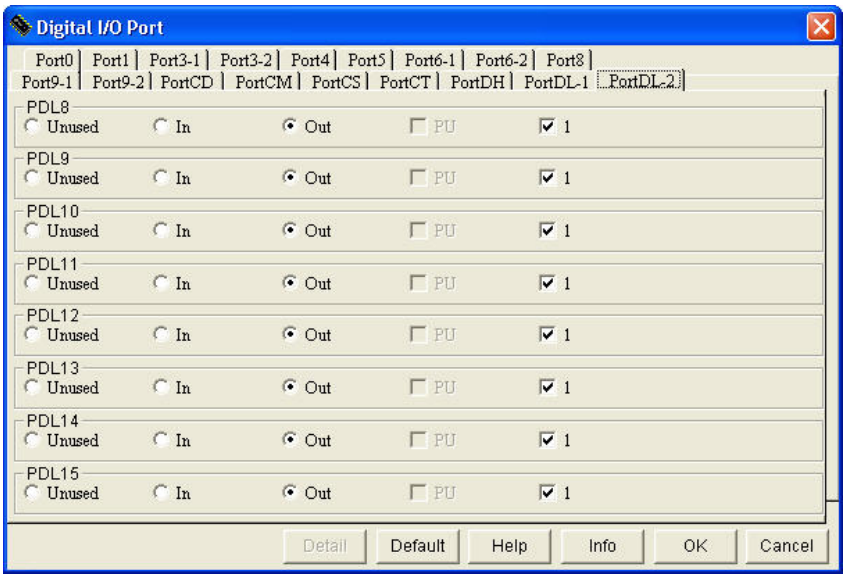
Figure 37. Port DH Selection



5.4.3 Port DL-2 Selection

Using the Port DL-2 tab, set up this port as an output for the left LED with the same settings as for Port DH above.

Figure 38. Port DL-2 Selection



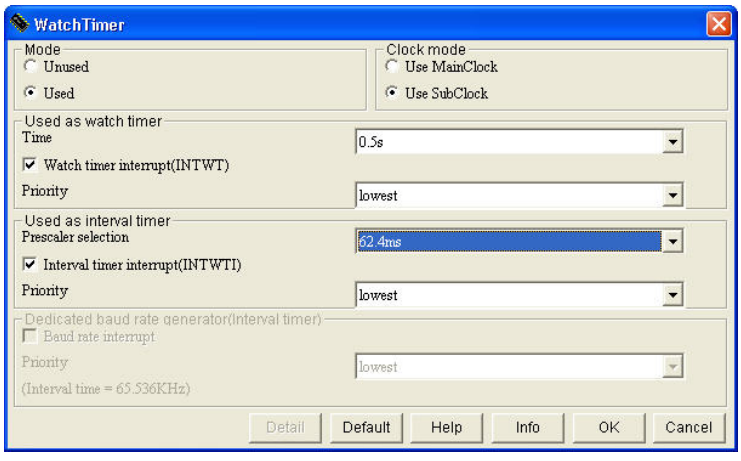
5.5 Timer Subsystem

This demonstration uses only timer TM01 as an interval timer from the timer subsystem.

5.5.1 Watch Timer Selections

The initial selection for the watch timer is to run from the subsystem clock. The demonstration requires a slow watch timer for long delays, in contrast to the faster interval timer, which suits blinking and shorter delays.

Figure 39. Watch Timer Selections

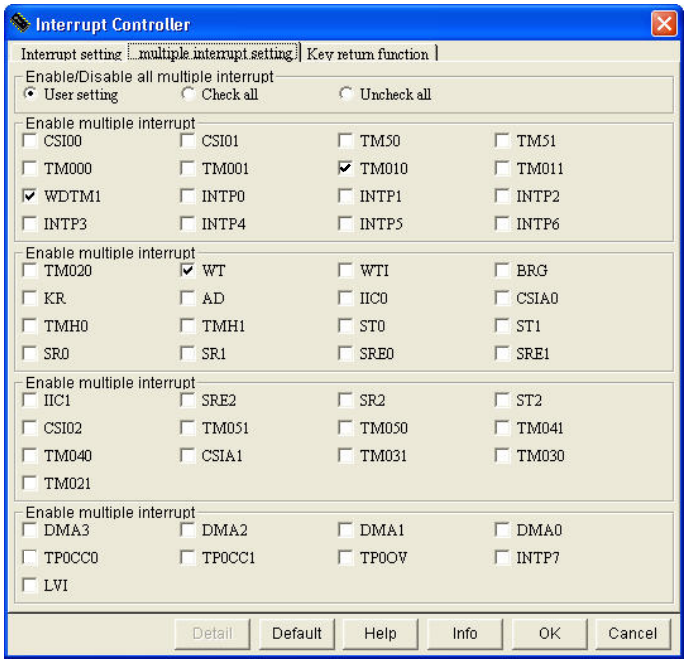


5.6 Interrupt Selections

Enable interrupts from the following sources:

- TM010: Timer 0
- WDTM1: Watch Dog Timer 1
- WT: Watch Timer
- WTI: Watch Timer

Figure 40. Interrupt Selections



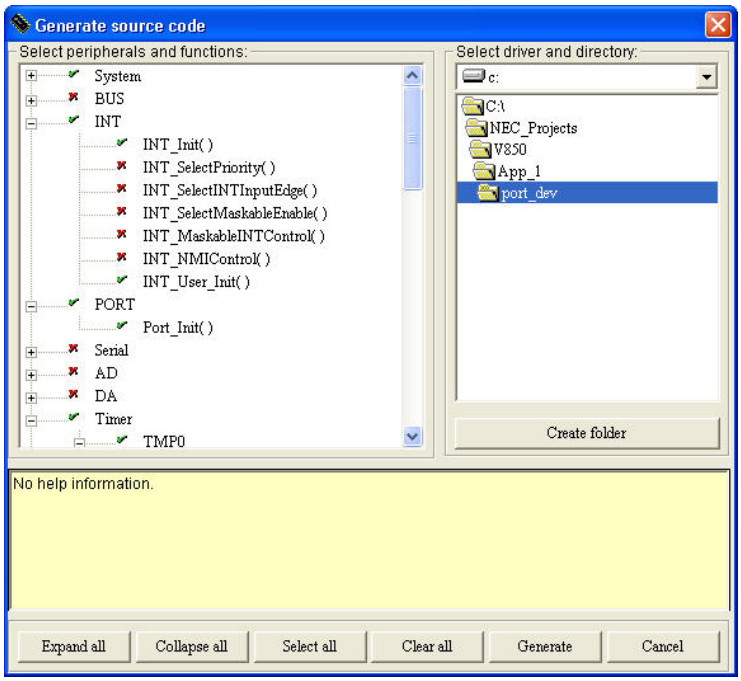
5.7 Generation of Source files

Once you have chosen all the subsystem selections, it is time to specify the code you want the Applilet to generate. If you do not make all the choices correctly, you can always come back, click a different selection, and regenerate the base code. Therefore, it is a good idea to save the files that are generated in a separate directory and make a copy of them for use in development. Then you can easily generate changes and just do a “difference.”

5.7.1 Interrupt and Port Functions

Make the selections shown below.

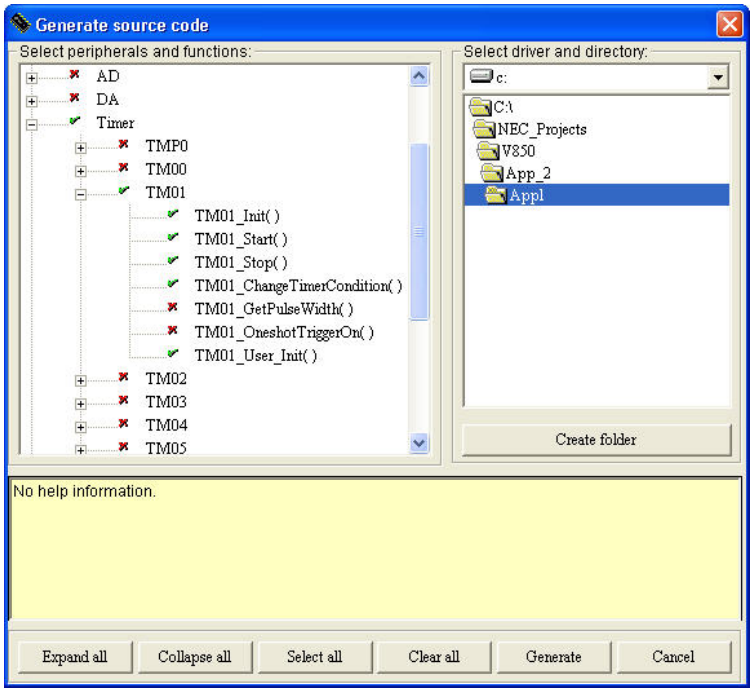
Figure 41. Interrupt and Port Functions



5.7.2 Timer Functions

Make the selections shown below.

Figure 42. Timer Functions



6. Appendix C — Software Code Listings

6.1 crtE.s

```
#   Copyright (C) NEC Electronics Corporation 1998,2002
#   NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
#   All rights reserved by NEC Electronics Corporation.
#   This program must be used solely for the purpose for which
#   it was furnished by NEC Electronics Corporation.  No part of this
#   program may be reproduced or disclosed to others, in any
#   form, without the prior written permission of NEC Electronics
#   Corporation.  Use of copyright notice does not evidence
#   publication of the program.

#   @(#)crtE.s  1.8 02/12/12 15:19:37

#=============================================================================
# NAME
#    crtE.s -  start up module for ca850(V850E)
#
# DESCRIPTIONS:
#      This assembly program is a sample of start-up module for ca850(V850E).
#    If you modified this program, you must assemble this file, and
#    locate a given directory.
#
#    Unless -G is specified, sections are located as the following.
#
#                  |        :        |
#                  |        :        |
#       tp -> -+-----------------+ __start     __tp_TEXT
#                  |  start up       |
#                  |-----------------|
#   text section   |                 |
#                  | user program    |
#                  |                 |
#                  |-----------------|
#                  | library         |
#              -+-----------------+
#                  |        :        |
#                  |        :        |
#              -+-----------------+ __argc
#                  |      0          |
#                  |-----------------| __argv
#   data section   |    #.L16        |
#                  |-----------------| .L16
#                  | 0x0,0x0,0x0,0x0 |
#                 -+-----------------+
#                  |                 |
#   sdata section  |                 |
#                  |                 |
#            gp-> -+-----------------+             __ssbss
#                  |                 |
#   sbss section   |                 |
#                  |                 |
```

```
#                 +-----------------+ __stack     __esbss     __sbss
#                 | stack area      |
#   bss section   |                 |
#                 |  0x200 bytes    |
#           sp-> -+-----------------+ __stack + STACKSIZE     __ebss
#
#===============================================================================


#-------------------------------------------------------------------------------
#   special symbols

#-------------------------------------------------------------------------------
    .extern __tp_TEXT, 4
    .extern __gp_DATA, 4
    .extern __ep_DATA, 4
    .extern __ssbss, 4
    .extern __esbss, 4
    .extern __sbss, 4
    .extern __ebss, 4


#-------------------------------------------------------------------------------
#    C program main function
#-------------------------------------------------------------------------------
    .extern _main


#-------------------------------------------------------------------------------
#   for argv
#-------------------------------------------------------------------------------
    .data
    .size   __argc, 4
    .align  4
__argc:
    .word   0
    .size   __argv, 4
__argv:
    .word   #.L16
.L16:
    .byte   0
    .byte   0
    .byte   0
    .byte   0


#-------------------------------------------------------------------------------
#   dummy data declaration for creating sbss section
#-------------------------------------------------------------------------------
    .sbss
    .lcomm  __sbss_dummy, 0, 0


#-------------------------------------------------------------------------------
#   system stack
#-------------------------------------------------------------------------------
```

```
    .set    STACKSIZE, 0x200
    .bss
    .lcomm  __stack, STACKSIZE, 4


#-------------------------------------------------------------------------------
#   RESET handler
#-------------------------------------------------------------------------------

    .section        "RESET", text
    jr      __start

#-------------------------------------------------------------------------------
#   start up
#       pointers:  tp - text pointer
#                  gp - global pointer
#                  sp - stack pointer
#                  ep - element pointer
#       mask reg:  r20 - 0xff
#                  r21 - 0xffff
#   exit status is set to r10
#-------------------------------------------------------------------------------
    .text
    .align  4
    .globl  __start
    .globl  __exit
    .globl  __startend
    .extern  ___PROLOG_TABLE
__start:
    mov #__tp_TEXT, tp      -- set tp register
    mov #__gp_DATA, gp      -- set gp register offset
    add tp, gp          -- set gp register
    mov #__stack+STACKSIZE, sp  -- set sp register
    mov #__ep_DATA, ep      -- set ep register
#
    .option nowarning
    mov 0xff,   r20     -- set mask register
    mov 0xffff, r21     -- set mask register
    .option warning
#
    mov #__ssbss, r13       -- clear sbss section
    mov #__esbss, r12
    cmp r12, r13
    jnl .L11
.L12:
    st.w    r0, [r13]
    add 4, r13
    cmp r12, r13
    jl  .L12
.L11:
#
    mov #__sbss, r13        -- clear bss section
    mov #__ebss, r12
    cmp r12, r13
    jnl .L14
.L15:
```

```
    st.w    r0, [r13]
    add 4, r13
    cmp r12, r13
    jl  .L15
.L14:
#
    mov     #___PROLOG_TABLE, r12   -- for prologue/epilogue runtime
    ldsr    r12, 20         -- set CTBP (CALLT base pointer)
#
#                   -- for Programmable peripheral I/O
#   mov 0x9234, r13     -- 0x1234(addr) | 0x8000(use pI/Or)
#   st.h    r13, BPC        -- set BPC
#
    ld.w    $__argc, r6     -- set argc
    movea   $__argv, gp, r7     -- set argv
    jarl    _main, lp       -- call main function
__exit:
    halt                -- end of program
__startend:
#                               #
#-------------------- end of start up module -------------------#
#                               #
```

6.2 systeminit.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+, V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 initiate
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ************************************************************************
** Include files
** ************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
#include "port.h"
#include "timer.h"
#include "watchtimer.h"
#include "watchdogtimer.h"
/*
** ************************************************************************
** MacroDefine
** ************************************************************************
*/
extern unsigned long _S_romp;
// RAM

/************************************************************************/
/* Function:    hdwinit()                                               */
/* Description: set up initial hardware                                 */
/*      note - this is called by startup code before main() is called*/
/* Input:       none                                                    */
/* Return:      none                                                    */
/************************************************************************/
void hdw_init( void )
{
unsigned char uc;
```

```
    /* not necessary to stop Watchdog timer 1, not running by default*/
    // stop Watchdog timer 2, which is running by default
    WDTM2 = 0x1F;   // stop operation and deselect clock (recommended value to stop)

    PLLCTL = 0x03;                      // Enable PLL
    __asm("mov 0x3fff000,r10");         // Set SFR base in R10
    __asm("st.b r0,0x1fc[r10]");        // Write to PRCMD
    __asm("st.b r0,0x828[r10]");        // Set PCC for Fxx
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  Init every Macro
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
void  SystemInit( void )
{
    __asm("di");                /* disable interrupt */
    _rcopy(&_S_romp, -1);       /* copy predefined values from rom to ram */
    hdw_init();                 /* some hardware initialization */
    PORT_Init( );               /* Port initiate    */
    INT_Init( );                /* INT initiate */
    WT_Init( );                 /* WT initiate      */
    WDT1_Init( );               /* WDT1 initiate    */
    TMO1_Init( );               /* TMO1 initiate */
    __asm("ei");                /* enable interrupt */
}
```

6.3 main.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+, V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**  The Standby functions are described in chapter 23 of the hardware Users Manual
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ************************************************************************
** Include files
** ************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
#include "port.h"
#include "timer.h"
#include "watchtimer.h"
#include "watchdogtimer.h"

/*
** ************************************************************************
** MacroDefine
** ************************************************************************
*/
#define TRUE          1
#define FALSE         0
#define MAX_TEST      6        /* maximum test number defined */

/* global variables */
extern volatile int watch_tick;
extern volatile int watch_rollover;
extern unsigned char watch_count;
extern volatile int TM01_count;
extern void  SystemInit( void );
```

```
/* prototypes */
void display_mode(int mode);
void clock_PLL_mode(void);
void clock_XTAL_mode(void);
void clock_SUB_mode(void);
void clock_stop_mode(void);
void clock_idle_mode(void);
void clock_halt_mode(void);
void write_CLM(unsigned char);
void write_PSC(unsigned char);
void set_PCC_SUB_mode(void);
void set_PCC_PLL_mode(void);

// RAM

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      main function
**              Power Down modes demonstration program
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void  main( void )
{
    int test_number = 0;
    int forever = 1;
    unsigned char swval;
    unsigned char port_data;
    unsigned short interval[2];

    /* initialize watch timer, the watch time will count */
    /* initialize 16 bit timer TM, used by PLL, XTAL and IDLE modes */
    /*    it will be an interrupt timer for dp1 */
    /* initialize interrupts   */
    /* initialize ??? to run from subclock and generate interrupt  */
    /*    it will be an interrupt timer for dp2 */
    /* set up port for switch input */
    /* set up port for seven segment led output */
    SystemInit( );


    WT_Start();            /* start the watch timer */
    interval[0] = 0x0100;  /* this is only about a ? second delay */
    interval[1] = 0xffff;
    TM01_ChangeTimerCondition(interval,1); /* set interval compare value */
    TM01_Start();

    led_dig_left(test_number);
    led_out_right(LED_PAT_EQUAL);
```

```
/* loop forever waiting for operator input */
while(forever)
{
    swval = sw_get();   /* get a debounced switch value */
    switch (swval) {
    case SW_LU_RU:
        break;      /* do nothing if both switches are up */

    case SW_LD_RU:  /* left switch (SW2) is down */
        test_number++;
        if(test_number > MAX_TEST)
            test_number = 1;
        led_out_right(LED_PAT_EQUAL);
        led_dig_left(test_number);
        TM01_Start();
        WT_Start();
        while(sw_get() == SW_LD_RU) {};  /* wait for switch to be released */
        break;

    case SW_LU_RD:  /* right switch (SW3) is down */
        while (sw_get() != SW_LU_RU)    /* wait for switches to both be up */
            ;
        switch(test_number)
        {
            case 1:
                clock_PLL_mode();
                break;
            case 2:
                clock_XTAL_mode();
                break;
            case 3:
                clock_SUB_mode();
                break;
            case 4:
                clock_halt_mode();
                break;
            case 5:
                clock_idle_mode();
                break;
            case 6:
                clock_stop_mode();
                break;
            default:
                break;
        }  /* end switch */

        break;

    case SW_LD_RD:  /* both switches down   */
        break;      /* ignore at this level */
    } /* end switch (swval) */


    if(watch_rollover)
    {
```

```
            port_data = PDH;    /* read back the port data */
            watch_rollover = 0;
            /* toggle the upper led decimal point */
            if(port_data & 0x80)
                PDH = port_data & 0x7F;  /* clear the decimal point */
            else
                PDH = port_data | 0x80;  /* set the decimal point  */
        }
        /* service watchdog 2 */
        //WDTE= 0xAC;  hdw_init turned it off

    } /* end while forever */

    return;  /* will never get here */
}

/*********************************************************************/
/* Function:    clock_PLL_mode()                                     */
/* Description: set up and operate clock at full speed using         */
/*              pll multiplier. IDLE control off, Halt control off    */
/*              stop timers TM01 and Watch                           */
/* Input:       none                                                 */
/* Return:      none                                                 */
/*********************************************************************/
void clock_PLL_mode(void)
{
    char clock;

    display_mode(MODE_PLL);
    PLLCTL = 0x03;      /* PLLON = operating, SELPLL = PLL operation */
    clock = PCC;
    clock &= 0xa0;      /* keep FRC, MFRC settings, clear the rest */
    write_PCC(clock);   /* operate at full multiplier speed        */
    delay_wt_ticks(600);
    TM01_Stop();
    WT_Stop();
    led_clear();        /* to get an honest power reading */
}

/*********************************************************************/
/* Function:    clock_XTAL_mode                                      */
/* Description: set up and operate clock at crystal speed (PLL off)   */
/*              stop timers TM01 and Watch                           */
/* Input:       none                                                 */
/* Return:      none                                                 */
/*********************************************************************/

void clock_XTAL_mode(void)
{
    char clock;

    display_mode(MODE_XTAL);
    PLLCTL = 0x00;      /* PLLON = clock through operation, SELPLL = PLL stopped */
    clock = PCC;
    clock &= 0xa0;      /* keep FRC, MFRC settings, clear the rest */
    write_PCC(clock);   /* operate at crystal speed fx */
```

```
    delay_wt_ticks(600);
    TM01_Stop();
    WT_Stop();
    led_clear();        /* to get an honest power reading */
}

/*************************************************************************/
/* Function:    clock_SUB_mode                                           */
/* Description: set up and operate clock using subsystem reference       */
/*              the subclock is 32.768kHz                                */
/*              stop timers TM01 and Watch                               */
/* Input:       none                                                     */
/* Return:      none                                                     */
/*************************************************************************/
void clock_SUB_mode(void)
{
    char clock;

    display_mode(MODE_SUB);
    TM01_Stop();
    WT_Stop();

    set_PCC_SUB_mode();
    delay_ms(6);   /* the delay time gets real long when running from subclock */
    led_clear();
    delay_ms(25);
    led_dig(0x88);
    /* may want to turn on main clock here */
    set_PCC_PLL_mode();
    WT_Start();
}

/*************************************************************************/
/* Function:    clock_halt_mode                                          */
/* Description: set up and operate clock in halt mode   23.3             */
/*              use interrupt from TM10 to release from halt, set        */
/*              watchtimer to use fxt input and generate interrupt       */
/*              which TM01 counts for a timeout of 10 seconds            */
/* Input:       none                                                     */
/* Return:      none                                                     */
/*************************************************************************/
void clock_halt_mode(void)
{
    unsigned short interval[2];

    display_mode(MODE_HALT);
    delay_ms(400);

    /* stop watch timer interrupts */
    WTIC = 0x47;
    WTIIC= 0x47;
    watch_rollover = 0;
    led_clear();

    /* set up timer TM01 for delay, use INTWT as clock source */
    interval[0] = 0x0980;
```

```
    TM01_ChangeTimerCondition(interval,1); /* set interval compare value */
    TM01_Start();  // just to be sure it is running

    __asm("halt");     /* HALT */
    __asm("nop");     /* delay for clock to start up again */
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
    led_dig(0x88);
}

/*********************************************************************/
/* Function:    clock_idle_mode                                      */
/* Description: set up and operate clock in idle mode  23.4          */
/*              PSC 23.2 p705  PSMKR 23.2 (2) p706                   */
/*              stop timers TM01 and Watch                           */
/* Input:       none                                                 */
/* Return:      none                                                 */
/*********************************************************************/
void clock_idle_mode(void)
{
    unsigned short interval[2];

    display_mode(MODE_IDLE);

    delay_ms(400);
    /* stop watch timer interrupts */
    WTIC = 0x47;
    WTIIC= 0x47;
    watch_rollover = 0;

    /* set up timer TM01 for delay, use INTWT as clock source */
    interval[0] = 0x0980;
    TM01_ChangeTimerCondition(interval,1); /* set interval compare value */
    TM01_Start();  // just to be sure it is running
    led_clear();

    PSMR = PSMR & 0xFE; /* clear PSM bit */
    write_PSC(0x02);  /* set standby mode, allow all interrupts for wakeup*/
    __asm("nop");     /* delay for clock to start up again */
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
    led_dig(0x88);
}

/*********************************************************************/
/* Function:    clock_stop_mode                                      */
/* Description: set up to stop clock 23.5                            */
/*              stop timers TM01 and Watch                           */
/* Input:       none                                                 */
/* Return:      none                                                 */
/*********************************************************************/
void clock_stop_mode(void)
```

```
{
    unsigned short interval[2];
    unsigned char temp;

    display_mode(MODE_STOP);

    delay_ms(400);
    /* stop watch timer interrupts, just using interrupt as input to timer 01 */
    WTIC = 0x47;
    WTIIC= 0x47;
    watch_rollover = 0;

    /* switch watch timer to sub clock input */
    WT_select_clock(WT_SUB_CLOCK);
    delay_ms(10);

    /* set up timer TM01 for delay, use INTWT as clock source */
    interval[0] = 0x2800;
    TM01_ChangeTimerCondition(interval,1); /* set interval compare value */
    TM01_Start();  // just to be sure it is running and interrupts are on

    led_clear();
    PSMR = 0x01;     /* set PSM bit, subclock oscillator used */

    write_PSC(0x02); /* set standby mode, allow all interrupts for wakeup*/
    __asm("nop");    /* delay for clock to start up again */
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
    led_dig(0x88);
}
```

6.4 int.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KF1,
**  V850ES/KG1, V850ES/KJ1 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation  2002-2004
**  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 a device driver for the INT module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ********************************************************************
** Include files
** ********************************************************************
*/
#include "macrodriver.h"
#include "int.h"


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

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**  Init the external INT, including enable or disable,
**  priority setting
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-----------------------------------------------------------------------
*/
```

```
void  INT_Init( void )
{
    ClrIORBit(INTR0, 0x04);         /* falling edge */
    SetIORBit(INTF0, 0x04);
    ClrIORBit(P0, 0x04);
    SetIORBit(PM0, 0x04);
    SetIORBit(PMC0, 0x04);
    INT_User_Init ( );
    return;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  All Maskable interrupt contorol
**
**  Parameters:
**  BOOL EnableFlag : set enable/disable
**      MD_TRUE -> enable
**      MD_FALSE -> disable
**
**  Returns:
**  MD_OK
**  MD_ARGERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS INT_MaskableINTControl( BOOL enableflag )
{
    if( enableflag == MD_TRUE ){
        IMR0 = 0;
        IMR1 = 0;

        IMR2 = 0;
        IMR3 = 0;
    }
    else if( enableflag == MD_FALSE ){
        IMR0 = 0xffff;
        IMR1 = 0xffff;
    }
    else{
        return MD_ARGERROR;
    }

    return MD_OK;
}
```

6.5 int_user.c

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KF1,
**  V850ES/KG1, V850ES/KJ1 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation  2002-2004
**  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 a device driver for the INT  interrupt service routine
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/

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

#pragma interrupt NMI MD_INTNMI
/*
** ***************************************************************************
** MacroDefine
** ***************************************************************************
*/


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

```
*/
void INT_User_Init( void )
{
    /* TODO. Add user code here */
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  NMI Interrupt service routine.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTNMI( void )
{
    /* TODO. Add user defined interrupt service routine */
}
```

6.6 port.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : port.c
**  Abstract : This file implements a device driver for the PORT module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/

/*
**=======================================================================
** Include files
**=======================================================================
*/
#include "macrodriver.h"
#include "port.h"

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**      Initialises the I/O module
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-----------------------------------------------------------------------
*/
void PORT_Init( void )
{
    /* initialize the port registers */
    P0 = PORT_P0;
    P1 = PORT_P1;
    P3 = PORT_P3;
```

```
P4 = PORT_P4;
P5 = PORT_P5;
P6 = PORT_P6;
P8 = PORT_P8;
P9 = PORT_P9;
PCD = PORT_PCD;
PCM = PORT_PCM;
PCS = PORT_PCS;
PCT = PORT_PCT;
PDH = PORT_PDH;
PDL = PORT_PDL;

/* initialize the function registers */
PF3H = PORT_PF3;
PF4 = PORT_PF4;
PF5 = PORT_PF5;
PF6 = PORT_PF6;
PF8 = PORT_PF8;
PF9H = PORT_PF9;

 /* initialize the Pull-up resistor option registers */
PU0 = PORT_PU0;
PU1 = PORT_PU1;
PU3 = PORT_PU3;
PU4 = PORT_PU4;
PU5 = PORT_PU5;
PU6 = PORT_PU6;
PU8 = PORT_PU8;
PU9 = PORT_PU9;
PUCD = PORT_PUCD;
PUCM = PORT_PUCM;
PUCS = PORT_PUCS;
PUCT = PORT_PUCT;
PUDH = PORT_PUDH;
PUDL = PORT_PUDL;

 /* initialize the mode registers */
PM0 = PORT_PM0;
PM1 = PORT_PM1;
PM3 = PORT_PM3;
PM4 = PORT_PM4;
PM5 = PORT_PM5;
PM6 = PORT_PM6;
PM8 = PORT_PM8;
PM9 = PORT_PM9;
PMCD = PORT_PMCD;
PMCM = PORT_PMCM;
PMCS = PORT_PMCS;
PMCT = PORT_PMCT;
PMDH = PORT_PMDH;
PMDL = PORT_PMDL;

 /*--- initialize the mode control registers ---*/
PMC0 &= ~PORT_PMC0;
PMC3 &= ~PORT_PMC3;
PMC4 &= ~PORT_PMC4;
```

```
    PMC5 &= ~PORT_PMC5;
    PMC6 &= ~PORT_PMC6;
    PMC8 &= ~PORT_PMC8;
    PMC9 &= ~PORT_PMC9;
    PMCCM &= ~PORT_PMCCM;
    PMCCS &= ~PORT_PMCCS;
    PMCCT &= ~PORT_PMCCT;
    PMCDH &= ~PORT_PMCDH;
    PMCDL &= ~PORT_PMCDL;

    PORT_User();
}
```

6.7 port_user.c

```
/* functions making use of the I/O ports                */
/*                                                      */
/*  routines for LED display                            */
/*  for M-V850ES-KJ1 CPU board on M-Station base board  */
/*      PDL8-PDL15 = output to right digit (LED2)       */
/*      PDH0-PDH7  = output to left digit  (LED1)       */
/*  For M-Station V2.1, assumes default SBs inserted    */
/*      SB27-SB33 inserted to connect                   */
/*       J1.27-J1.33 (PDL8-PDL15) to LED2 segments A-G  */
/*      SB35-SB42 inserted to connect                   */
/*       J1.35-J1.42 (PDH0-PDH7) to LED1 segments A-DP  */
/*  For M-Station V1.1, insert jumpers connecting:      */
/*       ROW1.25-32 (PDL8-PDL15) to ROW2.25-32 (LED2)   */
/*       ROW1.17-24 (PDH0-PDH7) to ROW2.17-20 (LED1)    */

/* need pragma declaration to access SFR's in C */
#pragma ioreg

#include "macrodriver.h"
#include "int.h"
#include "port.h"

/* table of bit patterns for seven-segment digits */
static unsigned char dig_tab[] = {
   LED_PAT_0,    /* 0 */
   LED_PAT_1,    /* 1 */
   LED_PAT_2,    /* 2 */
   LED_PAT_3,    /* 3 */
   LED_PAT_4,    /* 4 */
   LED_PAT_5,    /* 5 */
   LED_PAT_6,    /* 6 */
   LED_PAT_7,    /* 7 */
   LED_PAT_8,    /* 8 */
   LED_PAT_9,    /* 9 */
   LED_PAT_A,    /* A */
   LED_PAT_B,    /* B */
   LED_PAT_C,    /* C */
   LED_PAT_D,    /* D */
   LED_PAT_E,    /* E */
   LED_PAT_F     /* F */
};

/************************************************************************/
/* Function:    PORT_User()                                             */
/* Description: user initialization done after system initialization */
/*              of the ports.                                           */
/* Input:       none                                                    */
/* Return:      none                                                    */
/************************************************************************/
void PORT_User (void)
{
```

```
    sw_init();              /* set up switches */
    sw_set_debounce(64);    /* initial debounce value */
    led_init();             /* initialize LED control ports */
}

/*********************************************************************/
/* Function:    led_init()                                           */
/* Description: set up ports for display of LED digits               */
/* Input:       none                                                 */
/* Return:      none                                                 */
/*********************************************************************/
void led_init(void)
{
    PMCDH = 0x00;       /* set port DH to port mode */
    PMDH = 0x00;        /* set port DH to output    */
    PDH  = 0xFF;        /* all LED's off            */

    PMCDLH = 0x00;      /* set port DL high 8-bits to port mode*/
    PMDLH = 0x00;       /* set port DL high 8-bits to output   */
    PDLH  = 0xFF;       /* all LED's off            */
}

/*********************************************************************/
/* Function:    led_out_right()                                      */
/* Description: output raw data to right LED                         */
/* Input:       unsigned char val - value to output to right segment */
/*              led port                                             */
/* Return:      none                                                 */
/*********************************************************************/
void led_out_right(unsigned char val)
{
    PDLH = val;
}

/*********************************************************************/
/* Function:    led_out_left()                                       */
/* Description: output raw data to left LED                          */
/* Input:       unsigned char val - value to output to left segment  */
/*              led port                                             */
/* Return:      none                                                 */
/*********************************************************************/
void led_out_left(unsigned char val)
{
    PDH = val;
}

/*********************************************************************/
/* Function:    led_dig()                                            */
/* Description: display input number as two hex digits               */
/* Input:   num - number to display                                  */
/*          bits 0-3 in right digit, in P7                           */
/*          bits 4-7 in left digit, in P0                            */
/* Return:  none                                                     */
/*********************************************************************/
void led_dig(unsigned char num)
{
```

```
    led_out_right(dig_tab[num & 0x0F]);         /* lower nibble*/
    led_out_left(dig_tab[(num >> 4) & 0x0F]);   /* upper nibble*/
}

/*********************************************************************/
/* Function: led_dig_bcd()                                           */
/* Description: display two digits of BCD coded bcdnum               */
/* Input:   bcdnum - number to display in BCD                        */
/*              0 - 9    displayed as right decimal digit, left blank*/
/*             10 - 99   displayed as two decimal digits            */
/*            100 - 255 displayed as blank                           */
/* Return:  none                                                     */
/*********************************************************************/
void led_dig_bcd(unsigned char bcdnum)
{
unsigned char tens_dig;
    if (bcdnum > 99)
    {
        led_clear();    /* display both digits blank */
        return;
    }

    if (bcdnum < 10)
    {
        led_out_right(dig_tab[bcdnum]);     /* just display right LED*/
        led_out_left(LED_PAT_BLANK);        /* blank left LED        */
        return;
    }

    /* 10 <= bcdnum <= 99 */
    tens_dig = 0;
    do {                  /* calculate ten's place and remainder */
        bcdnum -= 10;     /* by multiple subtractions of 10      */
        tens_dig += 0x10; /* while counting up the tens digit    */
    } while (bcdnum >= 10);
    /* now tens_dig has ten's place */
    /* and bcdnum has remainder     */
    led_dig(tens_dig+bcdnum);
}

/*********************************************************************/
/* Function:    led_dig_right()                                      */
/* Description: display number in right LED seven segment display    */
/* Input:       num  - number to display(0..0xf) in right digit      */
/* Return:      none                                                 */
/*********************************************************************/
void led_dig_right(unsigned char num)
{
    if (num > 0x0F)
        led_out_right(LED_PAT_BLANK);
    else
        led_out_right(dig_tab[num]);
}

/*********************************************************************/
/* Function:    led_dig_left()                                       */
```

```
/* Description: display number in left LED segment display          */
/* Input:       unsigned char num (0..0x0f) number to display       */
/* Return:      none                                                */
/**********************************************************************/
void led_dig_left(unsigned char num)
{
    if (num > 0x0F)
        led_out_left(LED_PAT_BLANK);
    else
        led_out_left(dig_tab[num]);
}

/**********************************************************************/
/* Function:    led_clear()                                           */
/* Description: blank the LED seven segment display                   */
/* Input:       none                                                  */
/* Return:      none                                                  */
/**********************************************************************/
void led_clear(void)
{
    led_out_right(LED_PAT_BLANK);   /* lower nibble*/
    led_out_left(LED_PAT_BLANK);    /* upper nibble*/
}

/**********************************************************************/
/* Description: display the cause of the reset based on RESF register */
/*   "oo" - manual reset or power-on-clear                            */
/*   "Lr" - low voltage reset                                         */
/*   "L1" - low voltage interrupt                                     */
/*   "dr" - watchdog reset                                            */
/*   "d1" - watchdog interrupt                                        */
/*   "cL' - clock monitor                                             */

/* MH_027  Seg_A2         aaaaa        MH_035  Seg_A1        */
/* MH_028  Seg B2        f     b       MH_036  Seg_B1        */
/* MH_029  Seg C2        f     b       MH_037  Seg_C1        */
/* MH_030  Seg D2         ggggg        MH_038  Seg_D1        */
/* MH_031  Seg E2        e     c       MH_039  Seg_E1        */
/* MH_032  Seg F2        e     c       MH_040  Seg_F1        */
/* MH_033  Seg G2         dddd         MH_041  Seg_G1        */
/* MH_034  Seg_dp2             o       MH_042  Seg_dp1       */
/* J1_B                                                      */
/* these go to P1_27 .. P1_42                                */
/* P1_27 - PDL08                       P1_35 - PDH0          */
/* P1_28 - PDL09                       P1_36 - PDH1          */
/* P1_29 - PDL10                       P1_37 - PDH2          */
/* P1_30 - PDL11                       P1_38 - PDH3          */
/* P1_31 - PDL12                       P1_39 - PDH4          */
/* P1_32 - PDL13                       P1_40 - PDH5          */
/* P1_33 - PDL14                       P1_41 - PDH6          */
/* P1_34 - PDL15                       P1_42 - PDH7          */
/**********************************************************************/

/**********************************************************************/
/* Function:    display_mode()                                        */
/* Description: display a symbol on the seven segment display which   */
```

```
/*              represents the clock mode we are operating in         */
/* Input:       int mode - index into the table of mode symbols       */
/*************************************************************************/
void display_mode(int mode)
{
    /* upper and lower segment to display for each reason */
    unsigned short led_seg[] = {0x8cc7, /* PLL      PL*/
                                0x89c7, /* crystal  HL*/
                                0xafef, /* ring     ri */
                                0x92c1, /* subsytem SU*/
                                0x92a3, /* stop     So*/
                                0xfba1, /* idle     id*/
                                0x8bcf, /* halt     hl*/
                                0x5838, /* clock monitor */
                                0xbfbf};/* not allowed --*/
    /* write the data to the LED ports */
    led_out_left( led_seg[mode]>>8);
    led_out_right( led_seg[mode]);
}

/*  routines for switch input that use input port       */
/*  for M-V850ES-KJ1 CPU board on M-Station base board  */
/*      P94 = input for left switch  (SW2)              */
/*      P95 = input for right switch (SW3)              */
/*  For M-Station 2.1, assumes default SBs inserted     */
/*      SB7 connecting J1.7 (P94) to SW2                */
/*      SB8 connecting J1.7 (P95) to SW3                */
/*  For M-Station 1.1, insert jumpers connecting:       */
/*      ROW1.5 (P94) to ROW2.5 (SW2)                    */
/*      ROW1.6 (P95) to ROW2.6 (SW3)                    */


/* local variables for switch handling */
static unsigned char sw_last;       /* last debounced switch value */
static unsigned char sw_new;        /* new value being debounced   */
static unsigned char sw_deb_value;  /* value of debounce counter   */
static unsigned char sw_deb_count;  /* debounce counter            */

/*************************************************************************/
/* Function:    sw_init()                                                */
/* Description: set up ports for user switch input                       */
/* Input:       none                                                     */
/* Return:      none                                                     */
/*************************************************************************/
void sw_init(void)
{
    /* set P94 and P95 to port mode */
    PMC9L &= 0xCF;
    /* set P94 and P95 to inputs    */
    PM9L |= 0x30;
    /* set pullups on P94 and P95   */
    PU9L |= 0x30;
    /* set static variables         */
    sw_last = SW_LU_RU; /* default is right up, left up (no switch pressed)*/
    sw_deb_value = SW_DEF_DEB_COUNT;    /* set default debounce counter value*/
    sw_deb_count = SW_DEF_DEB_COUNT;    /* set counter to max*/
```

```
}

/*********************************************************************/
/* Function:    sw_chk()                                             */
/* Description: return input from switches, undebounced              */
/* Input:       none                                                 */
/* Return:      unsigned char value of switches, not shifted         */
/*********************************************************************/
unsigned char sw_chk(void)
{
    return P9L & 0x30;
}

/*********************************************************************/
/* Function:    sw_set_debounce()                                    */
/* Description: set the debounce counter value                       */
/* Input:       unsigned char count - number of times to check the   */
/*              switch has not changed, before returning it          */
/* Return:      none                                                 */
/*********************************************************************/
void sw_set_debounce(unsigned char count)
{
    sw_deb_value = count;   /* set new debounce counter value*/
    sw_deb_count = count;   /* set counter to max*/
}

/*********************************************************************/
/* Function:    sw_get()                                             */
/* Description: return debounced switch input                        */
/*        This routine should be called periodically; on the nth call*/
/*        with a new switch value, will return the debounced value.  */
/*        Should be called often enough to rapidly poll switches;    */
/*        debounce count can be adjusted to filter out bounces.      */
/*        This method of debouncing requires no timers, returns quickly*/
/* Input:       none                                                 */
/* Return:      unsigned char - the current debounced switch values  */
/*********************************************************************/
unsigned char sw_get(void)
{
unsigned char val;

    val = sw_chk();     /* get current value*/
    /* if we have seen this before, just return it*/
    if (val == sw_last) {
        sw_new = sw_last;
        sw_deb_count = sw_deb_value; /* reset debounce counter to max*/
        return val;
    }

    /* val != sw_last, there is a new input            */
    /* if it's not the same as the previous new one,   */
    /* set the new new one, reset the debounce counter */
    if (val != sw_new) {
        sw_new = val;
        sw_deb_count = sw_deb_value;
        return sw_last;
```

```
    }

    /* val != sw_last, val == sw_new   */
    /* count down the debounce counter */
    sw_deb_count--;

    /* if we have counted down to zero, we have seen the same sw_new  */
    /* for debounce count times, it is now the debounced switch value */
    if (sw_deb_count == 0) {
        sw_last = val;
        sw_deb_count = sw_deb_value;
        return val;
    }

    /* if still debouncing, return the last value */
    return sw_last;
}
```

6.8 timer.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+ and V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/

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

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  Initiate TM01, select founction and input parameter
**  count clock selection, INT init
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
```

```
void TM01_Init( void )
{
    TMC01 = 0x0;                        /* stop TM01 */
    ClrIORBit(PRM01, 0x3);
    ClrIORBit(SELCNT1, 0x2);

    SELCNT1 |= ( TM01_Clock&0x4)>>1;    /* internal count clock */
    PRM01 |=( TM01_Clock&0x3);

    /* INTTM010 setting */
    TM0IC10 = Lowest;
    SetIORBit(TM0IC10, 0x40);
    /* TM01 interval */
    ClrIORBit(CRC01, 0x01);
    CR010 = TM01_INTERVALVALUE;
    CR011 = 0xffff;
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  start the TM01 counter
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**
**-------------------------------------------------------------------------
*/
void TM01_Start( void )
{
    TMC01 = 0x0c;               /* interval timer start */
    ClrIORBit(TM0IC10, 0x40);       /* enable INTTM010 */
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  stop the TM01 counter and clear the count register
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
void TM01_Stop( void )
{
    TMC01 = 0x0;                    /* stop TM01 */
    SetIORBit(TM0IC10, 0x40);       /* disable INTTM010 */
```

```
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  Change TM01 condition.
**
**  Parameters:
**  USHORT*:     array_reg
**  USHORT:   array_num
**  Returns:
**  MD_OK
**  MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM01_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
   switch (array_num){
         case 2:
                  CR011=*(array_reg + 1);
         case 1:
                  CR010=*(array_reg + 0);
                  break;
         default:
                  return MD_ERROR;
         }
      return MD_OK;
}
```

6.9 timer_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+ and V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 interrupt
**             service routine
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
************************************************************************
**Include files
************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"
#pragma interrupt INTTM010 MD_INTTM010

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

volatile int TM01_count;

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**  This function is an empty function for user code when TM01 initializing
**
**  Parameters:
**  None
**
**  Returns:
**  None
```

```
**
**-------------------------------------------------------------------------
*/
void TM01_User_Init( void )
{
    /* Add user code here */
    TM01_count = 0;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  TM01 INTTM010 Interrupt service routine
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTTM010( void )
{
    __asm("ei");
    TM01_count++;
    /* used to show activity of the clock              */
    /* toggle the decimal point of the lower digit led */
    if(PDLH & 0x80)
        PDLH &= 0x7f;  // clear the right decimal point
    else
        PDLH |= 0x80;  // set the right decimal point
}
```

6.10 watchdogtimer.c

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchdogtimer.c
**  Abstract : This file implements a device driver for the WATCHDOGTIMER module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/
/*
** ***************************************************************************
** Include files
** ***************************************************************************
*/
#include "macrodriver.h"
#include "watchdogtimer.h"

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  Watchdog timer 1 initiate in this function,
**  including mode selection clock selection, oscilliation stablization
**  selection.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
void WDT1_Init( void )
{
    wdt_done = 0;
    wdt_tick = 0;
    wdt_count = 0;
```

```
    wdt_rollover = 0;

     __asm(".set  _PRCMD, -0xe04");
     __asm(".set  _WDTM1, -0x93e");  /* 0xfffff6c2 */

    SetIORBit(WDT1IC, 0x40);    /* disable INTWDTM1 */
    WDCS = 0x07;            /* set watchdog timer 1  clock:2^21/fxw */

     /* watchdog timer mode 1:                                        */
     /* Upon overflow, non-maskable interrupt INTWDT1 is generated  */
     __asm("movea 0x10, r0, r10");
     __asm("st.b  r10, _PRCMD[r0]");
     __asm("st.b  r10, _WDTM1[r0]");
     __asm("nop");
     __asm("nop");
     __asm("nop");
     __asm("nop");
     __asm("nop");
}
```

6.11 watchdogtimer_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchdogtimer_user.c
**  Abstract : This file implements a device driver for the WATCHDOGTIMER
**             interrupt service routine
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ************************************************************************
** Include files
** ************************************************************************
*/
#include "macrodriver.h"
#include "watchdogtimer.h"

extern volatile int wdt_done;

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**  INTWDT1 interrupt service routine.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTWDT1( void )
{
    __asm("ei");   /* enable interrupts */
    wdt_done = 1;  /* indicate watch dog timer interrupt received */
```

```
}
```

6.12 watchtimer.c

```
/*
************************************************************************
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchtimer.c
**  Abstract : This file implements a device driver for the WATCHTIMER module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ************************************************************************
** Include files
** ************************************************************************
*/
#include "macrodriver.h"
#include "watchtimer.h"
/*
** ************************************************************************
** MacroDefine
** ************************************************************************
*/
extern volatile int watch_rollover;
/*
**-------------------------------------------------------------------------
**  Abstract:
**  Initiate the watch timer, select its working mode, count clock,
**  enable/disable interrupt etc through Configurator.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-------------------------------------------------------------------------
*/
void WT_Init( void )
{
    PRSM = 0x00;  /* stop interval timer baud rate generator */
```

```
                /* which is one source of watchtimer clock */
    /* disable watchtimer before changing the settings */
    WTM.1 = 0;
    WTM.0 = 0;
    PRSM.4 = 0;
    /* disable all the interrupts of watchtimer */
    SetIORBit(WTIC,  0x40); /* set interrupt mask to disable */
    SetIORBit(WTIIC, 0x40); /* set interrupt mask to disable */
    SetIORBit(BRGIC, 0x40); /* set interrupt mask to disable */
    SetIORBit(WTIC,  0x07); /* select lowest interrupt priority */
    SetIORBit(WTIIC, 0x07); /* select lowest interrupt priority */
    //ClrIORBit(WTM,   0x80); /* select watchtimer clock:fw=fxt */
SetIORBit(WTM,  0x80); /*fw=fbrg*/
    WTM &= 0xf3;  /* watchtimer flag setting:2^14/fw(0.5s) clear WTM2 & WTM3 */
WTM |= 0x08;        /* set prescale  WTM3 WTM2 */
    WTM &= 0x8f;  /* interval timer prescaler setting:2^11/fw */
    WTM |= 0x70;  /* cleared it then setting it */
/* set the dedicated baud rate generator */
PRSM |= WT_PRSM_BGCS;
PRSCM = WT_PRSCM;   /* baud rate generator compare reg */
PRSM.4 = 1;

    WT_User_Init( );
}

/*
**-----------------------------------------------------------------------
**  Abstract:
**  Restart the watch timer after stopping. Enable the interrupt.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**-----------------------------------------------------------------------
*/
void  WT_Start( void )
{
    WTM.1 = 1;
    WTM.0 = 1;
    ClrIORBit(WTIC, 0x40);      /* enable INTWT */
    ClrIORBit(WTIIC, 0x40);     /* enable INTWTI */
    return;
}

/*
**-----------------------------------------------------------------------
**  Abstract:
**  Stop the watch timer.
**
**  Parameters:
**  None
**
**  Returns:
```

```
**  None
**
**-------------------------------------------------------------------
*/
void WT_Stop( void )
{
    WTM.1 = 0;
    WTM.0 = 0;
    /* disable all the interrupts of watchtimer */
    SetIORBit(WTIC,  0x40);
    SetIORBit(WTIIC, 0x40);
    SetIORBit(BRGIC, 0x40);
    watch_rollover = 0;
    return;
}
```

6.13 watchtimer_user.c

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/Kx1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchtimer_user.c
**  Abstract : This file implements a device driver for the watchtimer
**             interrupt service routine
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ********************************************************************
** Include files
** ********************************************************************
*/
#include "macrodriver.h"
#include "watchtimer.h"

#pragma interrupt INTWT MD_INTWT
#pragma interrupt INTWTI MD_INTWTI

#define TICKS_PER_MS    1 /* tbd */

volatile int watch_tick;
volatile int watch_rollover;
unsigned char watch_count;
/*
** ********************************************************************
** MacroDefine
** ********************************************************************
*/

#define TICKS_PER_MS    1 /* tbd */

volatile int end;
/*
**-----------------------------------------------------------------------
**
```

```
**  Abstract:
**  This function is an empty function for user code when WT initializing
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**--------------------------------------------------------------------
*/
void WT_User_Init( void )
{
    watch_tick = 0;
    watch_count = 0;
    watch_rollover = 0;
}

/*********************************************************************/
/* Function:    delay_wt_ticks()                                     */
/* Description: delay for specified number of watch timer interrupts */
/* Input:       int count - number of ticks to delay                 */
/* Return:      none                                                 */
/*********************************************************************/
void delay_wt_ticks(int count)
{
    int i;
    __asm("di");
    end = count;
    __asm("ei");

    while(end > 0){};  // wait for watch timer to count down to end
}

/*********************************************************************/
/* Function:    delay_ms()                                           */
/* Description: delay for specified number of counts, software only  */
/* Input:       int count - number of 1K iterations to delay         */
/* Return:      none                                                 */
/*********************************************************************/

void delay_ms(int count)
{
    int i;

    for(i=0; i<count*1000; i++)
    {
        __asm("nop");
    }
}

/*
**--------------------------------------------------------------------
**
**  Abstract:
**  INTWT interrupt service routine.
```

```
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**--------------------------------------------------------------------
*/
__multi_interrupt void MD_INTWT( void )
{
    __asm("ei");
    /* user defined interrupt service routine */
    watch_tick++;
    end--;

    if(watch_tick >= 50)
    {
      watch_tick = 0;
      watch_count++;
      watch_rollover = 1;
    }
}

/*
**--------------------------------------------------------------------
**
**  Abstract:
**  INTWTI interrupt service routine.
**
**  Parameters:
**  None
**
**  Returns:
**  None
**
**--------------------------------------------------------------------
*/
__interrupt void MD_INTWTI( void )
{
    /* TODO. Add user defined interrupt service routine */
    __asm("ei");
}

void WT_select_clock(int select)
{
     switch (select)
     {
         case WT_SUB_CLOCK:
             PRSM = 0x00;  /* stop interval timer baud rate generator */
                           /* which is one source of watchtimer clock */
                           /* disable watchtimer before changing the settings */
             WTM.1 = 0;
             WTM.0 = 0;
             /* disable all the interrupts of watchtimer */
             SetIORBit(WTIC,  0x40); /* set interrupt mask to disable */
```

```
            SetIORBit(WTIIC, 0x40); /* set interrupt mask to disable */
            SetIORBit(BRGIC, 0x40); /* set interrupt mask to disable */
            SetIORBit(WTIC,  0x07); /* select lowest interrupt priority */
            SetIORBit(WTIIC, 0x07); /* select lowest interrupt priority */
            ClrIORBit(WTM,   0x80); /* select watchtimer clock:fw=fxt */
            WTM &= 0xf3;  /* watchtimer flag setting:2^14/fw(0.5s) clear WTM2 & WTM3 */
            WTM |= 0x08;  /* set prescale  WTM3 WTM2 */
            WTM &= 0x8f;  /* interval timer prescaler setting:2^11/fw */
            WTM |= 0x70;  /* cleared it then setting it */
            WTM.1 = 1;  /* start it back up */
            WTM.0 = 1;
            break;

        case WT_BAUD_RATE_GEN:
            WT_Init();  // default is baud rate generator input
            break;
    }
}
```

6.14 write_special.s

```
--/*
--************************************************************************
--**
--**  Filename : write_pcc.s85
--**  Abstract : This file implements a service routine to write the PCC
--**             (Processor Clock Control) register which is protected
--**
--**  Device:  uPD70F3318Y
--**
--**  Compiler:  NEC/C Compiler
--**
--************************************************************************
--*/

    .text       --RSEG CODE
    .globl   _write_PSC
    .globl   _write_PCC
    .globl   _write_CLM
    .globl   _write_WDTM1
    .globl   _set_PCC_SUB_mode
    .globl   _set_PCC_PLL_mode
    .align   4

#define        DCHC0           0xFFFFF0E0
#define        DCHC1           0xFFFFF0E2
#define        DCHC2           0xFFFFF0E4
#define        DCHC3           0xFFFFF0E6
#define        PRCMD           0xFFFFF1FC
#define        PSC             0xFFFFF1FE

#define        PSMR            0xFFFFF820
#define        PCC             0xFFFFF828
#define        CLM             0xFFFFF870


--/*
--**-------------------------------------------------------------------
--**
--** Abstract:
--**        This function sets the Power Save Control register
--**
--**  Parameters:
--**        R6 = value to set Power Save Control to
--**
--**  Returns:
--**         None
--**
--**-------------------------------------------------------------------
--*/
_write_PSC:
        st.b    r6, PRCMD[r0]  -- PRCMD register write
```

```
        st.b    r6, PSC[r0]    -- PSC register setting
        nop
        nop
        nop
        nop
        nop
        jmp [lp]

--/*
--**-------------------------------------------------------------------
--**
--**  Abstract:
--**        This function sets the Processor Clock Control register
--**
--**  Parameters:
--**        R6 = value to set clock control to
--**
--**  Returns:
--**         None
--**
--**-------------------------------------------------------------------
--*/
_write_PCC:
        st.b    r6, PRCMD[r0]  -- PRCMD register write
        st.b    r6, PCC[r0]    -- PCC register setting
        nop
        nop
        nop
        nop
        nop
        jmp [lp]

--/*
--**-------------------------------------------------------------------
--**
--**  Abstract:
--**        This function sets the Clock Monitor Control register
--**
--**  Parameters:
--**        R6 = value to set clock monitor control to
--**
--**  Returns:
--**         None
--**
--**-------------------------------------------------------------------
--*/
_write_CLM:
        nop
        st.b    r6, PRCMD[r0]  -- PRCMD register write
        st.b    r6, CLM[r0]    -- CLM register setting
        nop
        nop
        nop
        nop
        nop
        jmp [lp]
```

```
--/*
--**-------------------------------------------------------------------
--**
--**  Abstract:
--**        This function sets the Watchdog timer mode register 1
--**
--**  Parameters:
--**        R6 = value to set register to
--**
--**  Returns:
--**         None
--**
--**-------------------------------------------------------------------
--*/
_write_WDTM1:
        nop
        st.b    r6, PRCMD[r0]  -- PRCMD register write
        st.b    r6, WDTM1[r0]  -- WDTM1 register setting
        nop
        nop
        nop
        nop
        nop
        jmp [lp]

--/*
--**-------------------------------------------------------------------
--**
--**  Abstract:
--**        This function sets the subclock as the processor clock source
--**
--**  Parameters:
--**        none
--**
--**  Returns:
--**        none
--**
--**-------------------------------------------------------------------
--*/
_set_PCC_SUB_mode:
        st.b    r0, PRCMD[r0]
        set1    3, PCC[r0]    -- set CK3 bit to select fxt
        nop
        nop
        nop
_CHECK_CLS:
        tst1    4, PCC[r0]    -- wait for subclock to start operating
        nop
        bz      _CHECK_CLS    -- loop if not set

-- subclock in operation, now stop main clock
_stop_main_clock:
        st.b    r0, PRCMD[r0]
        set1    6, PCC[r0]    -- set MCK bit to stop main clock_halt_mode
        nop
```

```
        nop
        nop
-- check if main clock has stopped
_CHECK_MCK:
        tst1    6, PCC[r0]   -- wait for main clock to stop
        nop
        bz      _CHECK_MCK  -- loop if not set
        jmp [lp]

--/*
--**-------------------------------------------------------------------
--**
--**  Abstract:
--**        This function sets the PCC PLL mode
--**
--**  Parameters:
--**        none
--**
--**  Returns:
--**        none
--**
--**-------------------------------------------------------------------
--*/
_set_PCC_PLL_mode:
        add     -4, sp        -- push r11 onto stack
        st.w    r11, 0[sp]

        st.b    r0, PRCMD[r0]
        clr1    6, PCC[r0]    -- clear MCK bit to allow clock to oscillate
        movea   0x55,r0,r11   -- time out value
_WAIT_OST:
        nop                   -- delay
        nop
        nop
        addi    -1, r11, r11
        --mp      r0, r11
        bne     _WAIT_OST
        st.b    r0, PRCMD[r0]
        clr1    3, PCC[r0]    -- clear CK3 bit
_CHECK_CLS2:
        tst1    4, PCC[r0]    -- check if main clock running
        nop
        nop
        bnz     _CHECK_CLS2   -- no, loop

        ld.w    0[sp], r11    -- pop r11 from the stack
        add     4, sp
        jmp     [lp]          -- return
--  END
```

6.15 int.h

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KF1,
**  V850ES/KG1, V850ES/KJ1 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation  2002-2004
**  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 a device driver for the INT module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/
#ifndef _MDINT_
#define _MDINT_
/*
** ***************************************************************************
** MacroDefine
** ***************************************************************************
*/
#define IC_BASE     0xfffff110   /* interrupt control register base address */
enum ExternalINT { EX_NMI, EX_INTP0, EX_INTP1, EX_INTP2, EX_INTP3, EX_INTP4, EX_INTP5, EX_INTP6, EX_INTP7 };
enum MaskableSource{
        INT_WDT1, INT_INTP0, INT_INTP1, INT_INTP2, INT_INTP3, INT_INTP4, INT_INTP5, INT_INTP6,
        INT_TM000, INT_TM001, INT_TM010, INT_TM011, INT_TM50, INT_TM51, INT_CSI00, INT_CSI01,
        INT_SRE0, INT_SR0, INT_ST0, INT_SRE1, INT_SR1, INT_ST1, INT_TMH0, INT_TMH1,
        INT_CSIA0, INT_IIC0, INT_AD, INT_KR, INT_WTI, INT_WT, INT_BRG, INT_TM020,
        INT_TM021, INT_TM030, INT_TM031, INT_CSIA1, INT_TM040, INT_TM041, INT_TM050, INT_TM051,
        INT_CSI02, INT_SRE2, INT_SR2, INT_ST2, INT_IIC1,
        INT_LVI=48, INT_INTP7, INT_TP0OV,INT_TP0CC2, INT_TP0CC1, INT_DMA0, INT_DMA1, INT_DMA2,
        INT_DMA3
        };
void INT_Init( void );
__interrupt  void MD_NMI( void );
MD_STATUS INT_MaskableINTControl( BOOL enableflag );
void INT_User_Init ( void );

#endif
```

6.16 macrodriver.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+ and V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/

#ifndef _MDSTATUS_
#define _MDSTATUS_
#pragma ioreg       /*enable use the register directly in ca850 compiler*/

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

#define MD_ON       1
#define MD_OFF      0

#define MD_TRUE         1
#define MD_FALSE        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_ADDRESSMATCH     MD_STATUSBASE+0x3   /*IIC slave address match*/
#define MD_OVF              MD_STATUSBASE+0x4   /*timer count overflow*/
#define MD_DMA_END          MD_STATUSBASE+0x5   /*DMA transfer end*/
#define MD_DMA_CONTINUE     MD_STATUSBASE+0x6   /*DMA transfer continue*/
#define MD_SPT              MD_STATUSBASE+0x7   /*IIC stop*/
#define MD_NACK             MD_STATUSBASE+0x8   /*IIC no ACK*/
#define MD_SLAVE_SEND_END   MD_STATUSBASE+0x9   /*IIC slave send end*/
```

```
#define MD_SLAVE_RCV_END    MD_STATUSBASE+0x0   /*IIC slave receive end*/
#define MD_MASTER_SEND_END  MD_STATUSBASE+0x11  /*IIC master send end*/
#define MD_MASTER_RCV_END   MD_STATUSBASE+0x12  /*IIC master receive end*/


/*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 n=0,1,2*/
#define MD_OVERRUNERROR     MD_ERRORBASE+0x3    /*UARTn overrun error n=0,1,2*/
#define MD_FRAMEERROR       MD_ERRORBASE+0x4    /*UARTn frame error n=0,1,2*/
#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_ODDBUF           MD_ERRORBASE+0x8    /*in 16bit transfer mode,buffer size should be even*/
#define MD_DATAEXISTS       MD_ERRORBASE+0x9    /*Data to be transferred next exists in TXBn register*/

/* macro fucntion definiton */

/*main clock and subclock as clock source*/
enum ClockMode { MainClock, SubClock };
void  Clock_Init( void );
/*clear IO register bit and set IO register bit */
#define ClrIORBit(Reg, ClrBitMap)   Reg &= ~ClrBitMap
#define SetIORBit(Reg, SetBitMap)   Reg |= SetBitMap

enum INTLevel{Highest,Level1,Level2,Level3,Level4,Level5,Level6,Lowest};
enum TrigEdge { None, RisingEdge,FallingEdge, BothEdge };


#define SYSTEMCLOCK 20000000
#define SUBCLOCK    32768
#define MAINCLOCK   5000000

#endif
```

6.17 port.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : port.h
**  Abstract : This file implements a device driver for the PORT module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/

#ifndef _MDPORT_
#define _MDPORT_
/*
** ************************************************************************
** MacroDefine
** ************************************************************************
*/
#define PORT_PMC0   0x0
#define PORT_PM0    0xff
#define PORT_PU0    0x0
#define PORT_P0     0x0
#define PORT_PU1    0x0
#define PORT_PM1    0xff
#define PORT_P1     0x0
#define PORT_PMC3   0x0
#define PORT_PM3    0xffff
#define PORT_PU3    0x0
#define PORT_P3     0x0
#define PORT_PF3    0x0
#define PORT_PMC4   0x0
#define PORT_PM4    0xff
#define PORT_PU4    0x0
#define PORT_P4 0x0
#define PORT_PF4    0x0
#define PORT_PMC5   0x0
#define PORT_PM5    0xff
#define PORT_PU5    0x0
#define PORT_P5 0x0
```

```
#define PORT_PF5    0x0
#define PORT_PMC6   0x0
#define PORT_PM6    0xffff
#define PORT_PU6    0x0
#define PORT_P6 0x0
#define PORT_PF6    0x0
#define PORT_PMC8   0x0
#define PORT_PM8    0xff
#define PORT_PU8    0x0
#define PORT_P8 0x0
#define PORT_PF8    0x0
#define PORT_PMC9   0x0
#define PORT_PM9    0xffff
#define PORT_PU9    0x0
#define PORT_P9 0x0
#define PORT_PF9    0x0
#define PORT_PMCD   0xff
#define PORT_PCD    0x0
#define PORT_PMCM   0xff
#define PORT_PCM    0x0
#define PORT_PMCCM  0x0
#define PORT_PMCS   0xff
#define PORT_PCS    0x0
#define PORT_PMCCS  0x0
#define PORT_PMCT   0xff
#define PORT_PCT    0x0
#define PORT_PMCCT  0x0
#define PORT_PMDH   0x0
#define PORT_PDH    0x0
#define PORT_PMCDH  0xff
#define PORT_PMDL   0xff
#define PORT_PDL    0x0
#define PORT_PMCDL  0xff00
#define PORT_PUCD   0x0
#define PORT_PUCM   0x0
#define PORT_PUCS   0x0
#define PORT_PUCT   0x0
#define PORT_PUDH   0x0
#define PORT_PUDL   0x0

void PORT_Init( void );
void PORT_User( void );

/*  header for M-V850ES-KJ1 CPU board for LED digit display */
/*****************************************************************************/
/* Define definitions                                                        */
/*****************************************************************************/
/* LED Patterns for decimal and hex digits, characters */
/*  for individual bits,     ---A---    */
/*   0=on 1=off             |       |   */
/* bit 0 = segment A        F       B   */
/* bit 1 = segment B        |       |   */
/* bit 2 = segment C         ---G---    */
/* bit 3 = segment D        |       |   */
/* bit 4 = segment E        E       C   */
/* bit 5 = segment F        |       |   */
```

```
/* bit 6 = segment G         ---D--- DP */
/* bit 7 = decimal point                */

#define LED_PAT_0   0xC0
#define LED_PAT_1   0xF9
#define LED_PAT_2   0xA4
#define LED_PAT_3   0xB0
#define LED_PAT_4   0x99
#define LED_PAT_5   0x92
#define LED_PAT_6   0x82
#define LED_PAT_7   0xF8
#define LED_PAT_8   0x80
#define LED_PAT_9   0x98
#define LED_PAT_A   0x88
#define LED_PAT_B   0x83
#define LED_PAT_C   0xC6
#define LED_PAT_D   0xA1
#define LED_PAT_E   0x86
#define LED_PAT_F   0x8E
#define LED_PAT_BLANK   0xFF
#define LED_PAT_DP      0x7F
#define LED_PAT_DASH    0xBF
#define LED_PAT_ULINE   0xF7
#define LED_PAT_OLINE   0xFE
#define LED_PAT_EQUAL   0xB7

/* definitions for modes of clock operation */
#define MODE_PLL   0
#define MODE_XTAL  1
#define MODE_RING  2
#define MODE_SUB   3
#define MODE_STOP  4
#define MODE_IDLE  5
#define MODE_HALT  6
#define MODE_MON   7
#define MODE_XXX   8
/*********************************************************************/
/* Export functions                                                  */
/*********************************************************************/
extern void led_init(void);                    /* init ports for LED output    */
extern void led_dig(unsigned char num);        /* display number as hex        */
extern void led_dig_bcd(unsigned char bcdnum); /* display number as BCD        */
extern void led_dig_right(unsigned char num);/* display number in right LED */
extern void led_dig_left(unsigned char num);   /* display number in left LED   */
extern void led_out_right(unsigned char val);  /* output value to right LED    */
extern void led_out_left(unsigned char val);/* output value to left LED     */

/*  header for M-V850ES-KJ1 CPU board for base board switch reading */
/*********************************************************************/
/* Define definitions                                                */
/*********************************************************************/
/* symbolic definitions for switch inputs */
/* SW2 = left switch = P94 */
/* SW3 = right switch = P95 */
/*                                              P95     P94  */
#define SW_LU_RU    0x30    /* left up, right up     1       1   */
```

```
#define SW_LD_RU    0x20    /* left down, right up   1       0   */
#define SW_LU_RD    0x10    /* left up, right down   0       1   */
#define SW_LD_RD    0x00    /* left down, right down 0       0   */
#define SW_DEF_DEB_COUNT    16  /* default debounce counter */

/*********************************************************************/
/* Export functions                                                  */

/*********************************************************************/
extern void PORT_User (void);
extern void led_init(void);
extern void led_out_right(unsigned char val);
extern void led_out_left(unsigned char val);
extern void led_dig(unsigned char num);
extern void led_dig_bcd(unsigned char bcdnum);
extern void led_dig_right(unsigned char num);
extern void led_dig_left(unsigned char num);
extern void led_clear(void);
extern void display_mode(int mode);
extern void sw_init(void);          /* init ports for switch input   */
extern unsigned char sw_chk(void);  /* get undebounced switch input  */
extern unsigned char sw_get(void);  /* get debounced switch input    */
extern void sw_set_debounce(unsigned char count); /* set debounce cound*/
#endif
```

6.18 timer.h

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KJ1+, V850ES/KG1+,
**  V850ES/KF1+ and V850ES/KE1+ 32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/

#ifndef _MDTIMER_
#define _MDTIMER_

/*
** ***************************************************************************
**MacroDefine
** ***************************************************************************
*/
#define TM_TMP0_CLOCK             0x07
#define TM_TMP0_INTERVALVALUE   0x3d08
#define TM_TMP0_INTERVALVALUE2  0x1e83
#define TM_TMP0_ONESHOTOUTPUTCYCLE  0x3d08
#define TM_TMP0_ONESHOTOUTPUTDELAY  0x1e83
#define TM_TMP0_EXTTRIGGERCYCLE 0x3d08
#define TM_TMP0_EXTTRIGGERDELAY 0x1e83
#define TM_TMP0_PWMCYCLE    0x3d08
#define TM_TMP0_PWMWIDTH    0x1e83
#define TM_TMP0_CCROCOMPARE 0x3d08
#define TM_TMP0_CCR1COMPARE 0x1e83
#define TM00_Clock  0x1
#define TM00_INTERVALVALUE  0xc34f
#define TM00_SQUAREWIDTH    0xc34f
#define TM00_PPGCYCLE   0xc34f
#define TM00_PPGWIDTH   0x00
#define TM00_ONESHOTCYCLE   0xc34f
#define TM00_ONEPULSEDELAY  0x00
#define TM01_Clock  0x2
#define TM01_INTERVALVALUE  0x3e8
```

```
#define TMO1_SQUAREWIDTH    0x3e8
#define TMO1_PPGCYCLE   0x3e8
#define TMO1_PPGWIDTH   0x0
#define TMO1_ONESHOTCYCLE   0x3e8
#define TMO1_ONEPULSEDELAY  0x00
#define TMO2_Clock  0x0
#define TMO2_INTERVALVALUE  0x00
#define TMO2_SQUAREWIDTH    0x00
#define TMO2_PPGCYCLE   0x00
#define TMO2_PPGWIDTH   0x00
#define TMO2_ONESHOTCYCLE   0x00
#define TMO2_ONEPULSEDELAY  0x00
#define TMO3_Clock  0x0
#define TMO3_INTERVALVALUE  0x00
#define TMO3_SQUAREWIDTH    0x00
#define TMO3_PPGCYCLE   0x00
#define TMO3_PPGWIDTH   0x00
#define TMO3_ONESHOTCYCLE   0x00
#define TMO3_ONEPULSEDELAY  0x00
#define TMO4_Clock  0x0
#define TMO4_INTERVALVALUE  0x00
#define TMO4_SQUAREWIDTH    0x00
#define TMO4_PPGCYCLE   0x00
#define TMO4_PPGWIDTH   0x00
#define TMO4_ONESHOTCYCLE   0x00
#define TMO4_ONEPULSEDELAY  0x00
#define TMO5_Clock  0x0
#define TMO5_INTERVALVALUE  0x00
#define TMO5_SQUAREWIDTH    0x00
#define TMO5_PPGCYCLE   0x00
#define TMO5_PPGWIDTH   0x00
#define TMO5_ONESHOTCYCLE   0x00
#define TMO5_ONEPULSEDELAY  0x00
#define TM5O_Clock  0x5
#define TM5O_INTERVALVALUE  0x1e
#define TM5O_SQUAREWIDTH    0x1e
#define TM5O_PWMACTIVEVALUE 0x1e
#define TM51_Clock  0x5
#define TM51_INTERVALVALUE  0x1e
#define TM51_SQUAREWIDTH    0x1e
#define TM51_PWMACTIVEVALUE 0x1e
#define TMHO_Clock  0x3
#define TMHO_INTERVALVALUE  0x7c
#define TMHO_SQUAREWIDTH    0x7c
#define TMHO_PWMCYCLE   0x7c
#define TMHO_PWMDELAY   0x3d
#define TMHO_CARRIERDELAY   0x7c
#define TMHO_CARRIERWIDTH   0x3d
#define TMH1_Clock  0x3
#define TMH1_INTERVALVALUE  0x7c
#define TMH1_SQUAREWIDTH    0x7c
#define TMH1_PWMCYCLE   0x7c
#define TMH1_PWMDELAY   0x3d
#define TMH1_CARRIERDELAY   0x7c
#define TMH1_CARRIERWIDTH   0x3d
```

```
/*timer00 to 05,50,51,H0,H1 configurator initiation*/
void TM01_Init( void );

/*timer00 to 05 free running start,50,51,H0,H1 timer start*/
void TM01_Start( void );

/*timer00 to 05,50,51,H0,H1 timer stop*/
void TM01_Stop( void );
MD_STATUS TM01_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
void TM01_User_Init( void );
#endif
```

6.19 watchdogtimer.h

```
/*
*********************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchdogtimer.h
**  Abstract : This file implements a device driver for the WATCHDOGTIMER module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/
#ifndef _MDWTACHDOGTIMER_
#define _MDWTACHDOGTIMER_
/*
** ***************************************************************************
**MacroDefine
** ***************************************************************************
*/
volatile int wdt_done;                /* flag for watch dog timer interrupt */
unsigned char wdt_count;              /* counter for display                */
volatile int wdt_tick, wdt_rollover;  /* interrupt counter, flag indicating */
                                      /* count has reached some limit       */

void  WDT1_Init( void );
__interrupt void MD_INTWDT1( void );
#endif
```

6.20 watchtimer.h

```
/*
************************************************************************
**
**  This device driver was created by Applilet for the V850ES/KX1+
**  32-Bit Single-Chip Microcontrollers
**
**  Copyright(C) NEC Electronics Corporation 2002-2004
**  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 : watchtimer.h
**  Abstract : This file implements a device driver for the WATCHTIMER module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
#ifndef _MDWATCHTIMER_
#define _MDWATCHTIMER_
/*
** ************************************************************************
**MacroDefine
** ************************************************************************
*/
#define WT_PRSM_BGCS    0x01
#define WT_PRSCM        0x98

#define WT_SUB_CLOCK     01
#define WT_BAUD_RATE_GEN 02

extern void delay_wt_ticks(int count);

void WT_Init( void );
void WT_Start( void );
void WT_Stop( void );
void WT_select_clock(int select);
void delay_ms(int count);
__multi_interrupt void MD_INTWT( void );
#endif
```