To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

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

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

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

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

RENESAS

Application Note

Flash Self-Programming of V850ES Microcontrollers

Document no. U18517EU1V0UM00

Contents

1. INTRODUCTION

In most microcontroller applications, programs and data are stored in a non-volatile memory, so that such data can be retained even if power to the microcontroller is turned off completely. Flash memory, such as the program and data storage implemented in many NEC Electronics microcontrollers, provides extremely fast reading time and small cell size. This allows large amounts of program and data memory (for example, 256 KB) to be implemented on-chip for a low cost.

Although program and data memory is non-volatile, it is useful to be able to change the program and/or data once programmed. Flash memory can be written, erased, and rewritten many times. Flash memory can be written from the erased state (generally all *one* bits) fairly quickly, but can only be erased (zero bits converted to ones) on a block by block or chip basis.

Writing of flash memory can be done by an external flash programmer, or with NEC Electronics microcontrollers, internal flash memory can be “self-programmed”, in other words written or erased by the microcontroller itself. NEC Electronics supplies a flash self-programming library of code, providing an easy user interface for flash self-programming.

This application note will show the use of flash self-programming library functions to allow an on-chip program to modify itself. The program will show both boot loading (loading and executing an external program) and boot rewriting (loading of a modified boot program).

1.1 Flash Memory Self-Programming Technique

Different types of NEC Electronics microcontrollers may have differing physical flash memory structures or process technology, which can affect the details of how the flash memory programming is done. To make the process of self-flash programming simple for the user, NEC Electronics provides prewritten programs for self-programming.

Each NEC Electronics microcontroller which supports self-programming has an internal fixed memory area, call boot ROM or BROM, containing firmware which will manage the process of erasing, writing, and verifying the flash memory. Normally, this memory area is not available for the processor, and the on-chip circuitry controlling flash memory writing is disabled. This prevents accidental modification of flash memory through a runaway program.

In order to activate the flash programming circuitry and BROM firmware, an external pin (usually FLMD0) must be set to a high logic level. This pin is pulled down for normal

operation. Connecting this pin to a pull-down resistor and another I/O pin will provide the microcontroller with the ability to control the FLMD0 pin to enable flash programming mode.

Since the process of flash memory writing modifies the internal program space of the microcontroller, it is not possible to fetch instructions from flash memory while it is being written or erased. The program which makes calls on the internal BROM firmware must be located in external memory, or in internal RAM, while the flash memory self-programming functions are active.

The process of setting up for flash self-programming, with execution in internal RAM, is as follows:

- (Flash) Copy program code for handling self-programming to internal RAM
- (Flash) Activate FLMD0 to enable entry to flash programming mode
- (Flash) Call function in internal RAM
  - (RAM) Enable flash programming (BROM active, flash not available)
  - (RAM) Call function in BROM for flash programming function
    - (BROM) Perform function, return status
  - (RAM) Disable flash programming (BROM not available, flash available)
- (RAM) Return status to calling function
- (Flash) Continue program, act on status returned

In this way, the microcontroller is not executing instructions from flash memory while it is being programmed.

1.2 Flash Self-Programming Libraries

In order to make flash self-programming easier, NEC Electronics has developed a set of program libraries for individual types of devices, which manage the above process of setting up for flash programming and performing the functions. The flash self-programming libraries are provided by NEC Electronics for inclusion in user programs, and present a simple interface of routines for doing initialization, writing, erasing, and verifying of flash memory.

The user program needs to set aside an area of internal RAM to hold the flash self-programming code that will be loaded into internal RAM, and call the initialization routine to copy code from flash ROM to this internal RAM space.

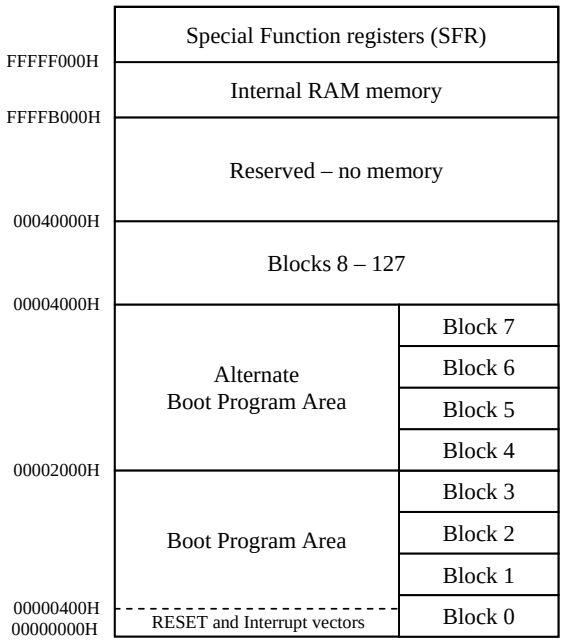
Once this code is copied, the process of self-programming is simplified to:

- Activate FLMD0
- Call flash self-programming library routine
- Deactivate FLMD0

This application note uses the flash self-programming libraries to perform flash memory writing, erasing, and verifying in order to load user programs or to revise the boot program. Please refer to section 3 for details of the documentation and library version used of the flash self-programming libraries.

1.3 Flash Memory Block Structure

In NEC Electronics V850ES microcontrollers with flash memory for programs and non-volatile data, the structure of memory looks like the example shown below.



In this example, an NEC Electronics microcontroller with 256 KB of flash memory is shown. This memory area extends from 00000000H through 0003FFFFH. This flash memory is divided into 128 blocks of 2 KB each. These blocks can be erased selectively, so that portions of the flash memory may be changed while other parts remain the same.

At the lowest addresses of the memory area are vectors for RESET and interrupts. When the microcontroller receives a RESET signal, it will begin operation by executing the instruction at 00000000H; this is known as the RESET vector. Following the RESET vector are a series

of vectors for interrupts; when a particular interrupt occurs and is enabled, the processor will save the program status and transfer execution to the appropriate interrupt vector.

The rest of the flash memory above the interrupt vectors is available for program and data storage, but the lowest several blocks may be configured as boot block areas, which may be handled specially.

1.4 Flash Memory Boot Swap Feature

Through the use of the flash self-programming functions, a program executing in memory can erase and reprogram other sections of memory. This process can include checksums and verification, to ensure that the loaded program is correct before execution. Such a program which can load other portions of the memory is often known as a “boot loader” program.

However, it may be desirable to reprogram the boot loader portion itself. This could be done by having a program in external memory, or performing reprogramming using an external flash programmer, but this will add to system cost or require additional equipment. Ideally, the microcontroller should be able to reprogram even the boot loader portion by erasing and rewriting the boot loader area.

This could be done by loading a “boot-rewriter” program into internal RAM, to erase and rewrite the boot loader area. But if the processor power were to fail during this process, it would be possible that the boot area could be left in a partially programmed state, and the device would no longer be functional.

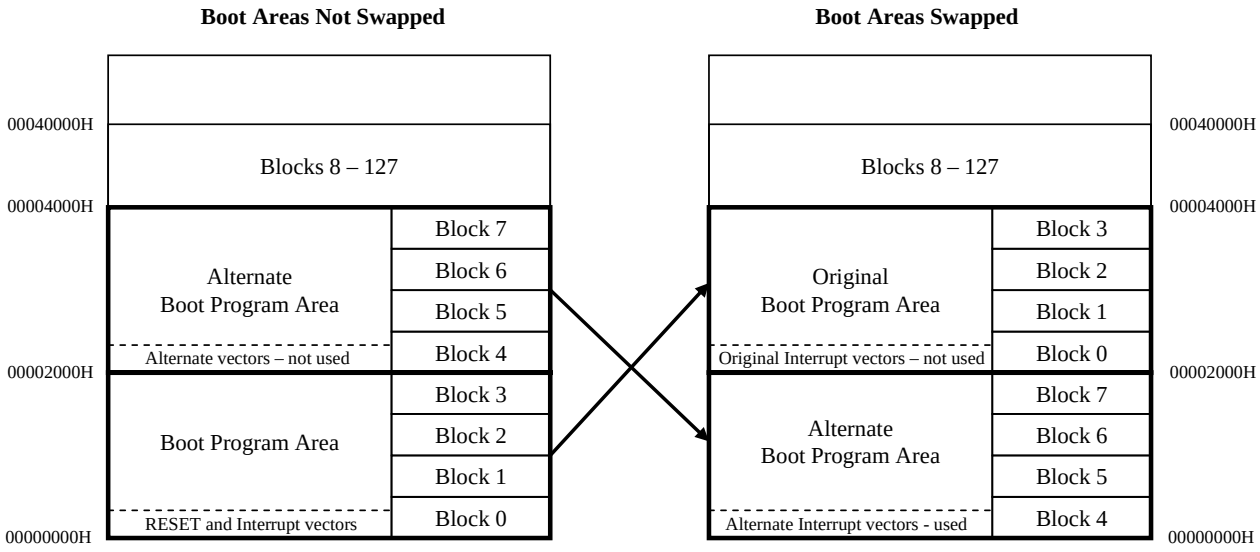
NEC Electronics microcontrollers with the “boot swap” feature treat some of the lowest blocks of memory as two separate boot areas, which can be swapped.

In this example, the area consisting of flash blocks 0 through 3 (00000000H through 00001FFFH), is the initial boot program area, and the area consisting of flash blocks 4 through 7 (00002000H through 00003FFFH) is the alternate boot program area.

A boot loader program in the initial boot program area can erase, rewrite, and verify an alternate boot program by programming it into the alternate boot program area. Once the loaded program has been verified as correct, a boot swap operation can be done to change the physical addresses associated with the two areas.

At this point, the alternate boot program area in blocks 4 through 7 is now located at addresses 00000000 to 00001FFF, and will be the active boot program on RESET. The original boot program area in blocks 0 through 3 is now located at addresses 00002000H to 00003FFFH, and may be rewritten under control of the alternate boot loader program.

Figure 1. Boot Swap Feature



In this way, a valid boot loader program can always be available in memory. Before the boot swap process, the original boot program will regain control after a RESET or unexpected power fail. The alternate program is verified before the boot swap, and after the boot swap will regain control on a RESET or power fail.

1.5 Interrupt Service During Flash Self-Programming

NEC Electronics microcontrollers have a variety of interrupt sources, including internal peripherals and external pins. Interrupts may be maskable (capable of being ignored) or non-maskable (always serviced). For maskable interrupts, each interrupt may be individually masked or enabled; in addition, a global interrupt enable flag may be set or cleared to enable or disable all maskable interrupts which are currently enabled by their individual mask flags.

When a non-maskable interrupt or an enabled maskable interrupt occurs, program execution is interrupted, the current program counter and program status word are saved in special system registers, and an interrupt service routine is executed. Each separate interrupt has an **interrupt vector**, located in low memory, which contains the address or a jump instruction to the interrupt service routine. The set of interrupt vectors for a program is known as the **interrupt vector table**, and for flash memory devices, is located in the lowest flash memory areas.

When the flash self-programming library code is being executed in RAM or in the BROM area, the flash memory containing the interrupt vector table will not be available. In this case, interrupts may still be serviced, but the handling of interrupts is different. All unmasked interrupts will save the program counter and program status word, and transfer

execution to the start of the internal RAM area. If is necessary to have interrupts serviced during flash self-programming functions, a specialize interrupt handling process and the code for interrupt service must be copied into internal RAM.

The interrupt handler must save all general purpose registers which may be modified. Then, since all interrupts will cause execution to begin at the same address, the processor must figure out which interrupt source caused the interrupt. In the V850ES series, on an interrupt the exception cause register (ECR**)** will be set to a value corresponding to the interrupt vector. By checking the ECR, the interrupt handler for flash self-programming mode can execute the proper interrupt service for the interrupt source.

During some parts of the flash programming operations, interrupts may be disabled for a period of time, and there will be a defined maximum latency time before the interrupt may be serviced. This latency time depends on the particular device, and is available from NEC along with the flash self-programming library.

The details of handling interrupts during flash self-programming may vary from device to device. The flash self-programming libraries include information and examples of interrupt handling during flash self-programming. This application note will show handling of multiple interrupts during flash self-programming.

1.6 Program Description and Specification

The boot program communicates with the user through a UART serial port, accepting commands and data and showing messages. The UART serial lines are connected to a computer running a terminal emulation program, which has the ability to transmit text files. Two seven-segment LEDs will also be used to show data and indicators.

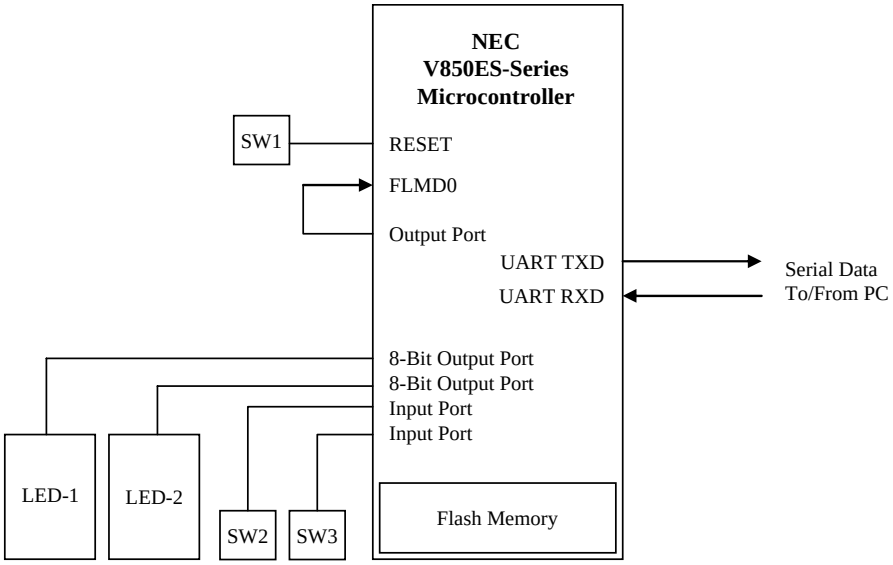
The Load command in the boot program will erase an area of memory used to store user programs, and load a user program into this area by accepting hex-format data transferred from the computer. The **Run** command will then execute the loaded user program.

The Bootload command will erase the second boot swap area, and load an alternate boot program into this area by accepting hex-format data. The Swap command will demonstrate swapping the two boot areas, to show how the boot loader itself may be changed.

The **Demo ISR** command will demonstrate interrupt service during flash programming and non-flash programming modes, by changing LED patterns on a periodic timer interrupt. Also, interrupt service for the UART reception of data will be active in flash programming mode.

In order to enable flash self-programming, an output port is used to control the FLMD0 input pin. This pin is usually pulled down; by setting a high level on the output port, the processor can activate FLMD0 to enable flash self-programming.

Figure 2. Demonstration Hardware



Specifications

- The processor will use a 5 MHz crystal to generate a 20 MHz system clock
- Two flash memory boot block areas will be used to hold the boot and alternate boot programs
- A flash memory user program area will hold the loaded user program
- UART1 will be used accepting commands and file data, and displaying results
- UART1 will operate at 38400 baud, 8 data bits, no parity, one stop bit
- An XOFF/XON protocol will be implemented to manage file download
- Timer TM00 will be used to generate a periodic 1 millisecond interrupt for ISR demonstration

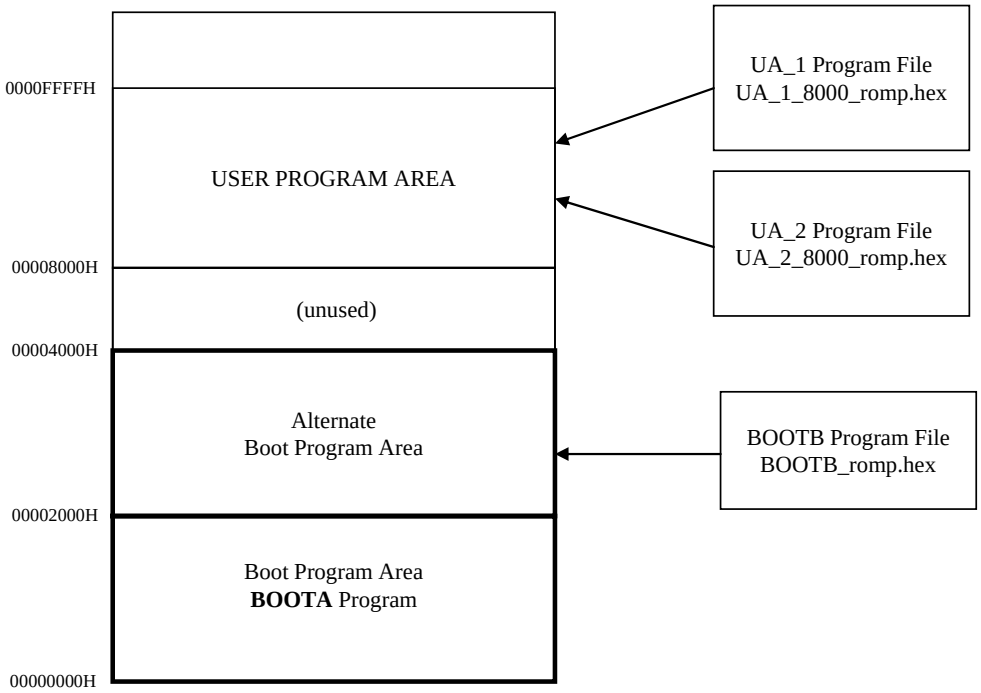
For the user application programs loaded, the LEDs will be used for data display; two input switches will also be used for user input.

1.7 Boot and User Program Structure

On initial loading, the boot program, version A (BOOTA) will be programmed into the first boot program area, from 00000000H to 00001FFFH. This program will accept the

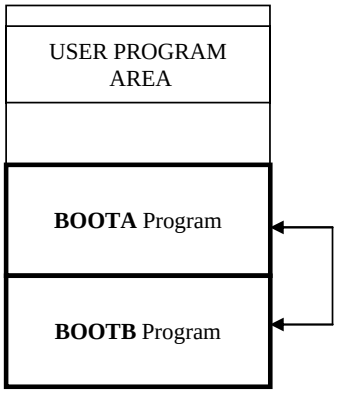
commands for loading and running of user programs, the loading of the alternate boot area, swapping of boot areas, and demonstration of interrupt service during flash programming.

Figure 3. Boot and User Program Areas



Two separate user application programs, UA_1 and UA_2, will be loaded from external files into the user program area, from 00008000H to 0000FFFFH. These programs will run independently of the boot program.

Figure 4. Old and New Boot Programs



A separate version of the boot program, version B (BOOTB) will be loaded from an external file, containing instructions and data in Intel hexadecimal format. BOOTB is essentially

identical to BOOTA, with the same commands; the sign-on message and LED will distinguish the two boot programs.

BOOTA will load the BOOTB program into the alternate boot program area, from 00002000H to 00003FFFH. When the boot swap command is executed, the two boot areas will be swapped on the next RESET of the processor.

1.8 Software Flow Charts

The demonstration boot program BOOTA (and BOOTB) consists of the following major sections:

- Initialization code for the program, called before the main() program starts, including clock and peripheral initialization
- The main program loop, which responds to commands received from the UART
- Subroutines for flash self-programming demonstration, which call the flash self-programming library
- The NEC Electronics flash self-programming library routines, which will perform flash memory operations
- UART subroutines for displaying strings and values, and reading input characters or lines
- LED subroutines
- Interrupt service routines for UART character reception and TM00 timer interrupt

The flowcharts here will describe initialization, the main program, flash self-program demonstration functions, and UART and TM00 timer interrupt service routines in both flash programming and non-flash programming modes.

Flowcharts are not included for UART communication functions, TM00 timer functions or LED. The software listings in the Appendix include this code.

This application note will include a summary of the routines used from the flash self-programming library. Please see the application note for the flash self-programming library for flowcharts and detailed descriptions of these routines. The flash self-programming library routines used are included in the software listings in the Appendix.

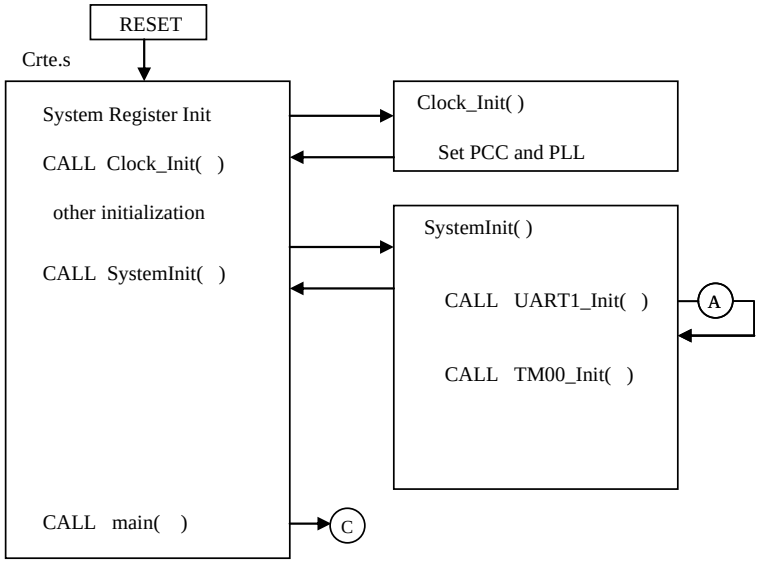
The user application programs UA_1 and UA_2 are simple I/O-based programs. Flowcharts are shown for the startup code and main() routines of these programs.

1.8.1 Boot Program Startup and Initialization

For V850ES programs written in the C language, the startup code for the C program is supplied by an assembly language startup file, generally named crte.s. This startup code specifies the Reset vector which determines where the program will begin on a hardware Reset, and provides initial setup of system registers before calling the main() function.

When Applilet is used to generate a C program for the V850ES microcontroller, the crte.s startup assembly language file is automatically generated, and includes a call to the Clock_Init() function to set the system clock, and a call to the SystemInit() function, which in turn calls initialization routines for peripherals.

Figure 5. Boot Program Startup



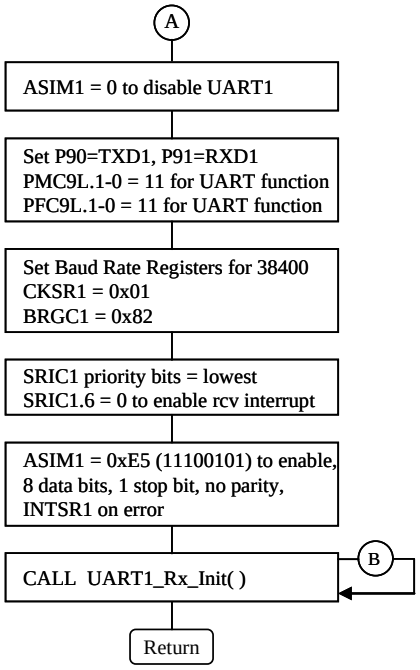
After the SystemInit() function finishes, the startup code calls the main() function of the user program. So at the start of the main() function, peripheral initialization has been done for several peripherals. The main() function does not need to call these individual peripheral initialization routines.

1.8.2 UART1_Init( ): Initialize UART1 Peripheral

The UART1_Init() routine, called from SystemInit(), sets up the UART1 peripheral for communication. First the ASIM1 control register is set to zero to disable the UART1 peripheral while registers are being set.

The port mode control register PMC9L and the port function control register PFC9L are set so that the P90 and P91 port pins will function as the TXD1 (transmit data) and RXD1 (receive data) functions respectively, instead of as general-purpose I/O ports.

Figure 6. Boot Program Initialization



The CKSR1 clock source and BRGC1 baud rate generator divider register are set to divide the main 20MHz clock to provide 38400 baud for UART1.

The SRIC1 interrupt control register for serial receive interrupt INTSR1 is set to enable the interrupt. We will be using interrupt-driven reception on UART1. Note: As generated by Applilet, the initialization code also enabled INTST1, the serial transmit end interrupt. This program does not use interrupt-driven transmission, so in the program code this interrupt is not enabled.

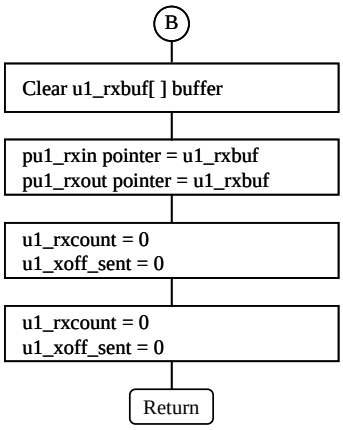
The appropriate bits in the ASIM0 control register are set to enable transmit and receive operation, with eight data bits, 1 stop bit, no parity. If a serial reception error is detected, a INTSR1 interrupt will be generated instead of a separate INTSRE1 interrupt.

Finally, the routine calls UART1_Rx_Init(), to set up variables for buffering received characters.

1.8.3 UART1_Rx_Init(): Initialize Variables For UART1 Receive Buffering

The UART1_Rx_Init() routine, called from UART1_Init(), sets up a buffer, pointers, and counter to store received characters on UART1. These variables are used in the interrupt service for the INTSR1 serial receive interrupt, and in routines which read buffered characters. An explanation of the variables and the buffering method is done here.

Figure 7. UART Initialization



The buffer **u1_rxbuf[ ]** is an array of bytes used to store received characters. The length of this buffer is arbitrary, and needs to be set based on the interaction between the microcontroller program and the remote COMPUTER. For this program, the length of the buffer is 32.

The **pu1_rxin** variable is a pointer to where in the buffer to read received characters, and is initially set to the start of the buffer. When an INTSR1 interrupt occurs, the received character is stored at the location pointed to by **pu1_rxin**, and the pointer is incremented. When it passes the end of the buffer, it is set to the start again.

The **pu1_rxout** variable is a pointer to where in the buffer to read received characters, and is also initially set to the start of the buffer. When the main program wants to read a character, and a character is available (**u1_rxcount** is not zero), the character is read from the location pointed to by **pu1_rxout**, and the pointer is incremented. When it passes the end of the buffer, it is set to the start again.

The **u1_rxcount** variable is the count of received characters in the buffer, initially zero. When an INTSR1 interrupt causes a character to be written to the buffer, the count is incremented; when the main program reads a character from the buffer, the count is decremented.

The **u1_xoff_sent** flag is used to manage XOFF/XON flow control for reception. When the **u1_rxcount** goes up, the receive buffer is filling up. To prevent overflow, when the count reaches a point somewhere below the maximum number of characters, an XOFF character is sent to the COMPUTER, telling it to stop sending characters, and the **u1_xoff_sent** flag is set. There is still room in the buffer for several more characters to be received, until the XOFF is acted on by the remote system.

When the program reads characters from the buffer, the **u1_rxcount** count will go down. When it approaches zero, and an XOFF has been sent as indicated by the **u1_xoff_sent** flag, then an XON character is sent to the COMPUTER, telling it to resume transmission.

1.8.4 Main( ): The Main Program—Flash Self-Programming

The main() program is called from the startup code after peripheral initialization. The program calls routines to initialize LED output ports, and to display the program letter (A or B) and version.

The main() program then calls the UART1_Rx_Clr() routine to clear any pending UART1 input, and the TM00_Start() routine to start the periodic millisecond timer.

The main() program then calls FSP_Init(); this initializes the flash self-programming library.

Main() then prints the boot program version string on UART1, and then prints the prompt string “L,R,B,S,I, or D” on UART1 to list commands available. It then waits for a character to be entered by the user. If no character is available, the program loops, waiting for input.

If a character is input, the program checks the letter for the set of commands supported, and takes action depending on the input.

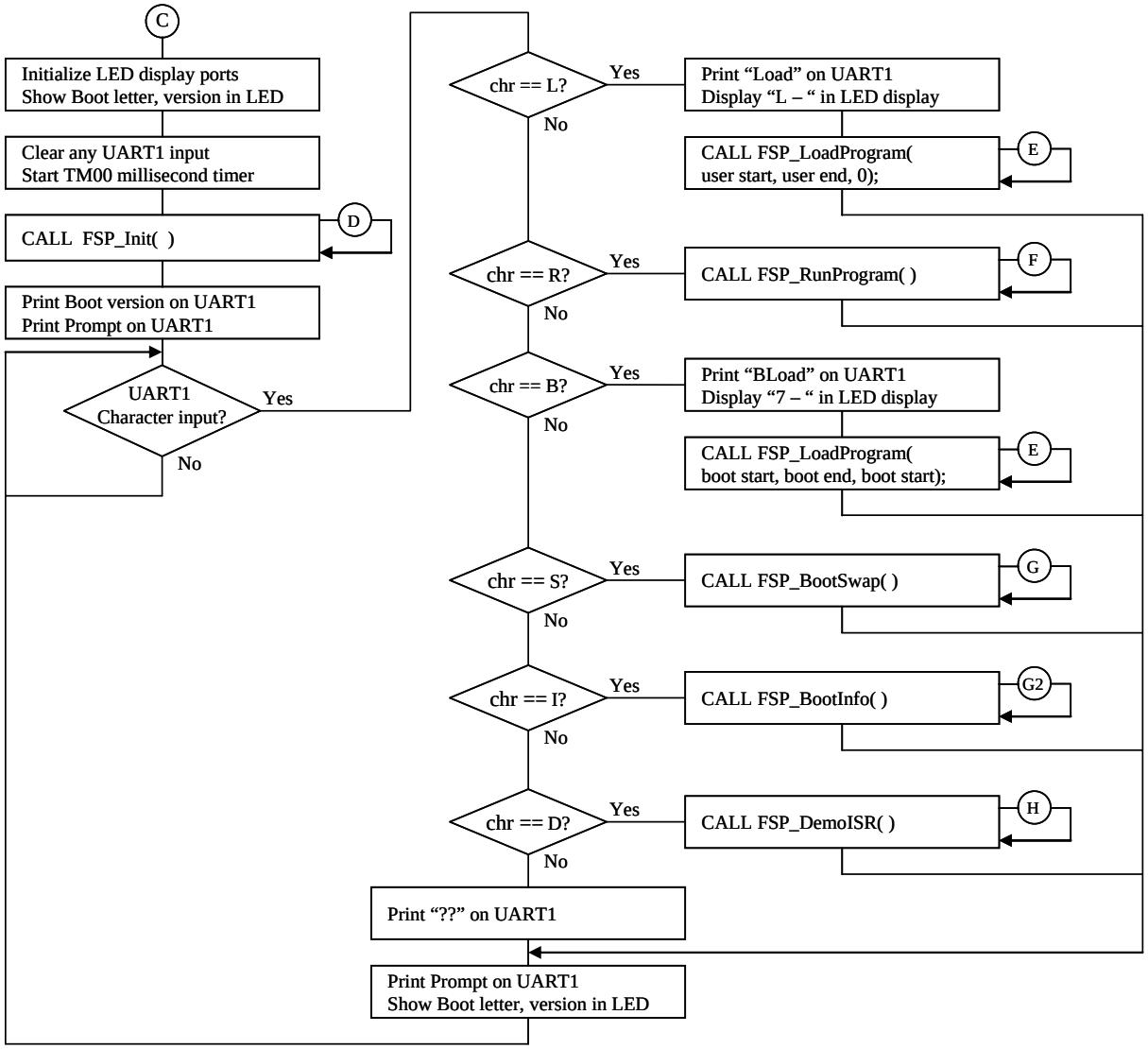
If an “L” is entered, the program will expect to load a program to the user area. The string “Load” is printed, and the LED shows “L –“. The L is to indicate “Load,” and the dash will be used to show UART1 input while loading. The main program calls FSP_LoadProgram(), with input parameters defining the start and end of the user program area, and an offset of zero to be added to the input data. When the FSP_LoadProgram() routine returns, the prompt is printed and the boot version is displayed again, and main() waits for the next character.

If an “R” is entered, the program will run the user program which has been loaded into the user program area, by calling FSP_RunProgram(). We do not expect the FSP_RunProgram() routine to return, but if it does, main() will print the prompt and wait for the next command.

If a “B” is entered, the program will expect to load a boot program to the alternate boot program area. The string “BLoad” is printed, and the LED shows “7 –“. The 7 is an upside-down L is to indicated BootLoad, and the dash will be used to show UART1 input while

loading. The main program calls FSP_LoadProgram(), with input parameters defining the start and end of the alternate boot program area, and an offset of the start of the boot program area to be added to the input data. When the FSP_LoadProgram() routine returns, the prompt is printed and the boot version is displayed again, and main() waits for the next character.

Figure 8. Main Program Flow



If an "S" is entered, the program will expect to swap in the alternate boot program by calling FSP_BootSwap(). Rather than returning, the FSP_BootSwap() routine will restart the current boot program by jumping to the RESET vector at address 00000000H; but if it were to return, main() would go to the common return point.

If an “I” is entered, the program calls the FSP_BootInfo() routine to report on the current state of the boot swap bit and the security flags as read.

If a “D” is entered, the program will expect to demonstrate interrupts while in the Flash self-programming mode, by calling the FSP_DemoISR() routine. When this routine returns, main() will display the prompt and wait for the next command.

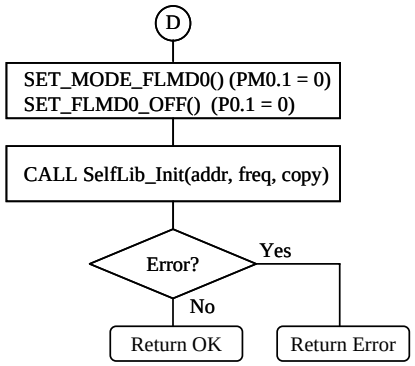
If the character input does not match any of the above choices, main() prints “??” on UART1, displays the prompt again, and waits for the next command.

1.8.5 FSP_Init(): Initialize Flash Self-Programming

The FSP_Init() routine initializes the flash self-programming software and hardware.

FSP_Init() first uses the macros (defined in target.h) to set the mode and initial state of the output pin used to control FLMD0. In this application, the I/O pin P01 is used to control the FLMD0 input. First the mode of this pin is set to output by setting bit one of the port mode register PM0 to zero. Then the output itself is set to zero to set a low level on the FLMD0 input.

Figure 9. Flash Self-Programming Initialization



The FSP_Init() routine then calls the flash self-programming library routine SelfLib_Init(SELFLIB_RAMADR, FREQUENCY, SELFLIB_SECTIONS). This sets up the flash self-programming library, by setting up a system data area, including the processor’s frequency (needed by the flash programming subroutines for timing), and copying necessary code sections from Flash memory to internal RAM.

The parameters passed to the SelfLib_Init() function depend on the target device, and are defined in the header file target.h. The SELFLIB_RAMADR parameter is the start address of internal RAM; for the µPD70F3318Y processor, this is set as FFFFB000H. The FREQUENCY parameter is 20000000 (20 MHz), which is the system clock for the demonstration program. The SELFLIB_SECTIONS parameter is set to SELFLIB_COPY,

which copies the necessary sections from flash memory to RAM (if the code for flash programming were located in external memory, it would not be necessary to copy them).

The result is that several sections of code, defined in the source files by "#pragma text (section name)" statements, are copied to internal RAM:

- SelfLib_ToRamUsrInt interrupt service handler, for start of internal RAM
- SelfLib_ToRamUsr user interrupt service routines, to be executed in RAM during programming
- SelfLib_ToRam library code to be executed when flash not available and possibly:
- SelfLib_RomOrRam library code which can execute in flash memory or RAM

For any interrupts which must be enabled and serviced during flash programming operations, the normal interrupt vectors at the bottom of flash memory may not be available. Instead, the processor will branch to the start of internal RAM for interrupt service. Code must be located here to handle the interrupts, and so is placed in the special section "SelfLib_ToRamUsrInt". The code for this section is in the assembly language source file userFuncInt.s.

The code for interrupt handling is in the special section "SelfLib_ToRamUsr", and is in the C source file userFunc.c. Interrupt handling is described in detail in a later section of this document

The code in the flash self-programming library which must execute in internal RAM is located in the special section "SelfLib_ToRam". This consists of routines which enable Flash programming (disabling Flash memory access), call the hidden BROM to perform functions, and then disable Flash programming (enabling Flash memory access again) before returning.

The code in the flash self-programming library which does not need to execute in RAM is located in the special section "SelfLib_RomOrRam". In the demonstration program, and the version of the flash self-programming library used, this code is located in flash memory, and is not copied to internal RAM. This saves internal RAM space.

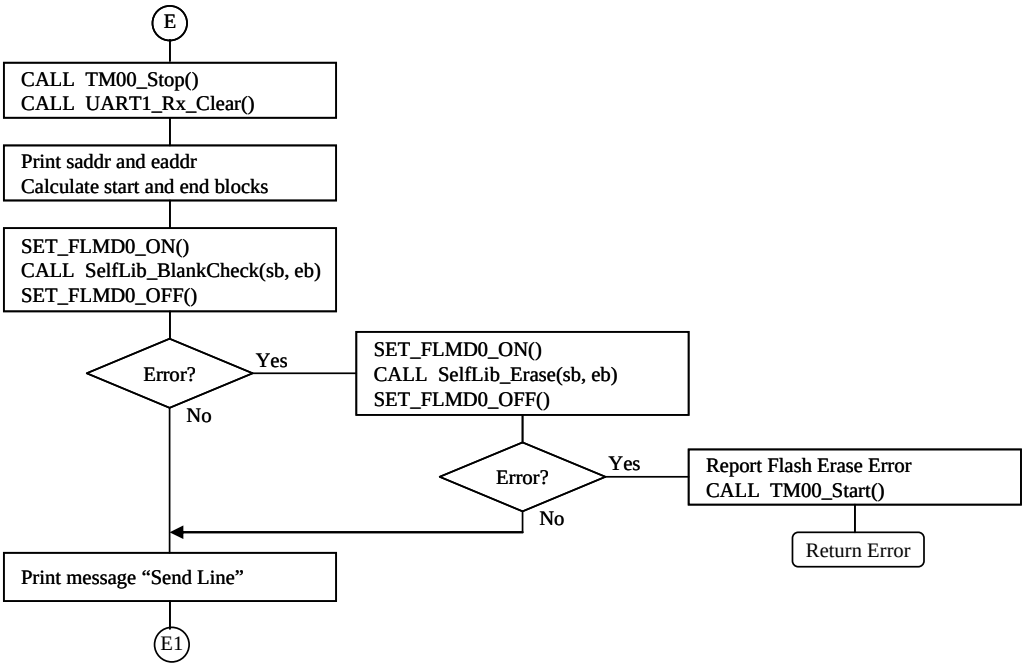
If there is any error returned by the SelfLib_Init() function, FSP_Init() will return an error; otherwise it will return a value indicating that initialization was successful.

1.8.6 FSP_LoadProgram(saddr, eaddr, offset): Load Program to Flash Memory

The FSP_LoadProgram() routine is used by both the “L” (Load) and “B” (Bload) commands to load the user program area or alternate boot program area, depending on the parameters passed.

The FSP_LoadProgram() routine reads input data from UART1 in Intel HEX format, and writes the input data to Flash memory in the range selected by the **saddr** (start address) and **eaddr** (end address) parameters. The **saddr** parameter defines the start of the area to be loaded; the **eaddr** parameter is one more than the last address to be loaded. For example, in the demonstration program, the user program area is from 00008000H to 0000FFFFH; to load the user program area, **saddr** would be 00008000H, and **eaddr** would be 00010000H. The offset specified by the **offset** parameter is added to the address in the input data, allowing a program starting at address 00000000H (such as the alternate boot program) to be loaded to a different memory area.

Figure 10. Flash Load Program



First the FSP_LoadProgram() routine stops the TM00 timer, to avoid the periodic 1 millisecond timer interrupts while doing flash programming. This is not absolutely necessary, as these interrupts would be handled during flash programming, but it will improve program efficiency. Any pending UART1 input is cleared by calling the UART1_Rx_Clear() routine.

FSP_LoadProgram() calculates the flash memory blocks which include the **saddr** and (**eaddr** – 1) locations, as variables **sb** (starting block) and **eb** (ending block). In the demonstration program, the block size is 2 KB per block. The starting block corresponding to 00008000H is 16; the ending block which contains 0000FFFFH is 31.

The routine turns FLMD0 on, calls Flash Self-Programming Library routine SelfLib_BlankCheck(**sb**, **eb**), and turns FLMD0 off. The return value from the blank check routine will indicate whether the area is blank. If the return value is zero (no error), the area is blank, and file loading can proceed.

If the SelfLib_BlankCheck() routine returns an error, the area is not blank. FSP_LoadProgram() then calls the flash self-programming library routine SelfLib_Erase(**sb**, **eb**) to erase the area. If an error is returned, the area cannot be erased. An error message is displayed, the TM00 timer is restarted, and an error status is returned.

If the erase is successful, or if the area was blank to start, a message is printed on UART1, telling the user to send the file. This is done by doing a “Send Text File” operation in the terminal program on the computer.

The FSP_LoadProgram() routine then calls the routine ihex_fget_hline(&hexline, 0); this routine will read lines of input from UART1. The lines will be scanned for Intel-format hex data, and the result returned by filling a structure of data, **hexline**. This structure will hold the line number of the line being processed, the type of data record, the address of the data, the extended segment address currently in use, the number of bytes in the line, and an array of bytes holding the data. The second parameter to this routine is a timeout value, defining how long the routine will wait for input data. For the first line, the timeout parameter is zero, indicating that the routine will not timeout.

The ihex_fget_hline() routine will return an error if the data is not in the proper format, a checksum error occurs, etc. If an error is returned, an error message is displayed, and the load is aborted. Any further UART1 input is cleared, the timer is restarted, and an error status is returned.

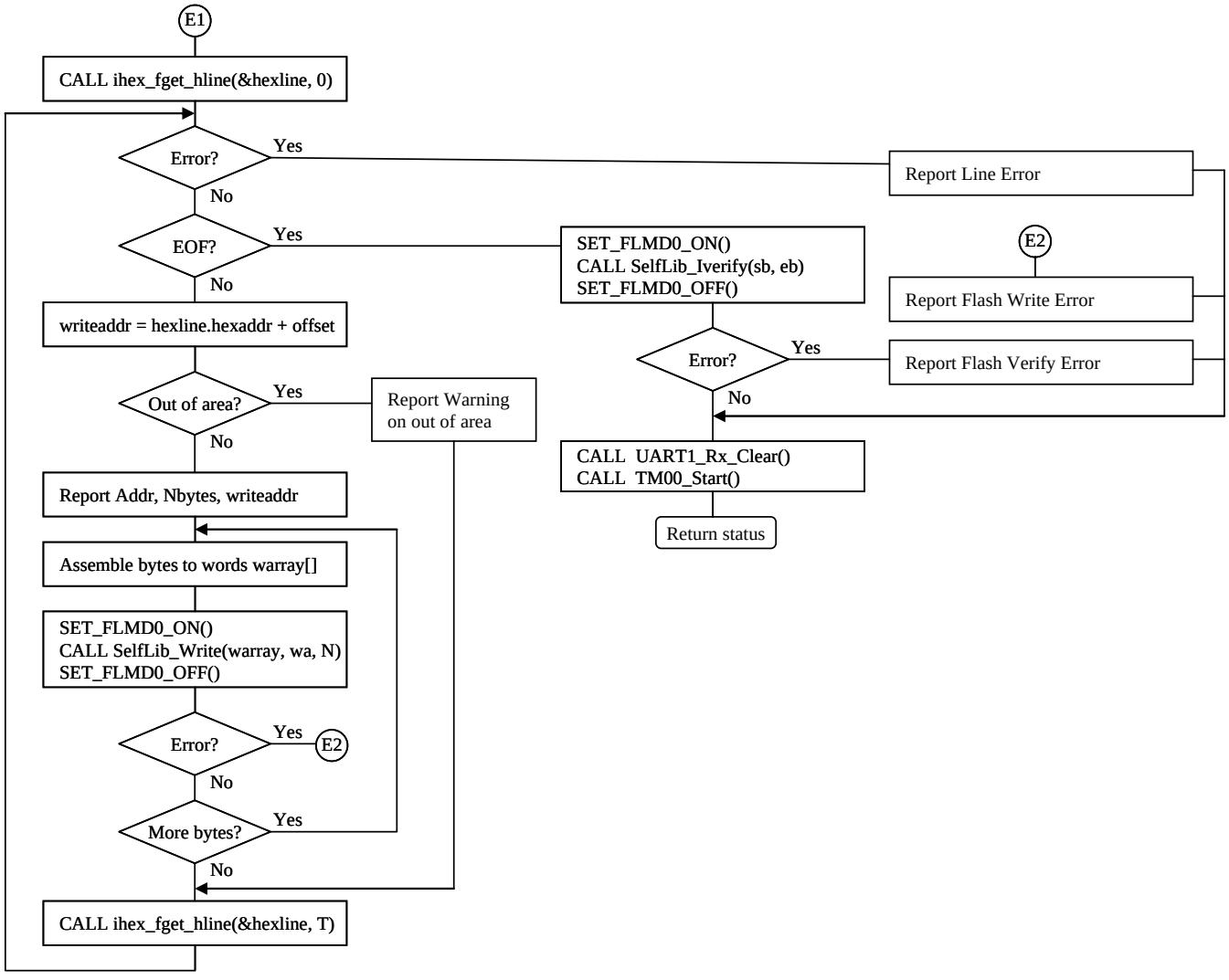
If the type of input hex record indicates EOF (end of file), the load is complete. The flash self-programming library routine SelfLib_Iverify() is called to verify any data written to the Flash memory in the selected area. If there is an error on verify, the error is reported. . Any further UART1 input is cleared, the timer is restarted, and either no error (correct verify) or error (incorrect verify is returned).

If the line type is not EOF, it contains data to be written to memory. A **writeaddr** variable is calculated, from the address specified in the hex record, plus the **offset** parameter passed to the FSP_LoadProgram() routine. The **writeaddr** variable is checked to make sure **writeaddr** is not below the start address **saddr**, or that **writeaddr** plus the number of bytes in the hex

record is not past the **eaddr** end address. If the data in the hex record is outside the range specified by **saddr** and **eaddr**, a warning message is printed, and the data is ignored.

If the data is in the proper area, the original hex address, the number of bytes, and the writeaddr values are printed. The bytes of data are assembled into 32-bit words in a word array **warray[]**. The size of the word array depends on the word granularity of the device being written; in the demonstration program for the µPD70F3318Y, the granularity is one word, so **warray[]** will only contain one word.

Figure 11. Hex File Download Processing



Once the word array has been filled with byte data, the word array is written to Flash memory by calling the SelfLIb_Write() routine in the flash self-programming library. The first parameter to the routine is the address of the array containing words; the second parameter is the address where the data is to be written; the third parameter is the number of

words, and is set to the word granularity of the device. The **wa** word address is calculated from the **writeaddr** variable, and incremented by the proper amount each time a set of bytes is written.

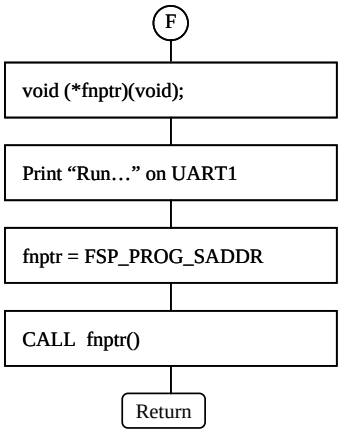
If the write results in an error, the error is reported and the routine exits with an error status, clearing any further UART1 input and restarting the TM00 timer. If the write is successful, the next set of bytes from the input line is assembled into the **warray[]** array, and written.

When all of the bytes in the line have been exhausted, the ihex_fget_hline() function is called to get the next line of data. This time, a timeout value **T** is specified, to timeout if the file does not end with an EOF record, or if the file or sending is truncated.

Once FSP_LoadProgram() is complete, the Intel hex format data sent will have been programmed and verified in the specified memory area.

1.8.7 FSP_RunProgram( ): Run Loaded User Program

Figure 12. Run New Program



The FSP_RunProgram() routine executes the user program, which has been loaded into the user progRAM by the “L” (Load) command.

The routine uses a variable, **fnptr**, which is defined as a pointer to a function, taking no parameters, and returning no parameters. The routine prints “Run…” on UART1, sets the function pointer **fnptr** to the address of the start of the user program area, and then calls the function pointed to.

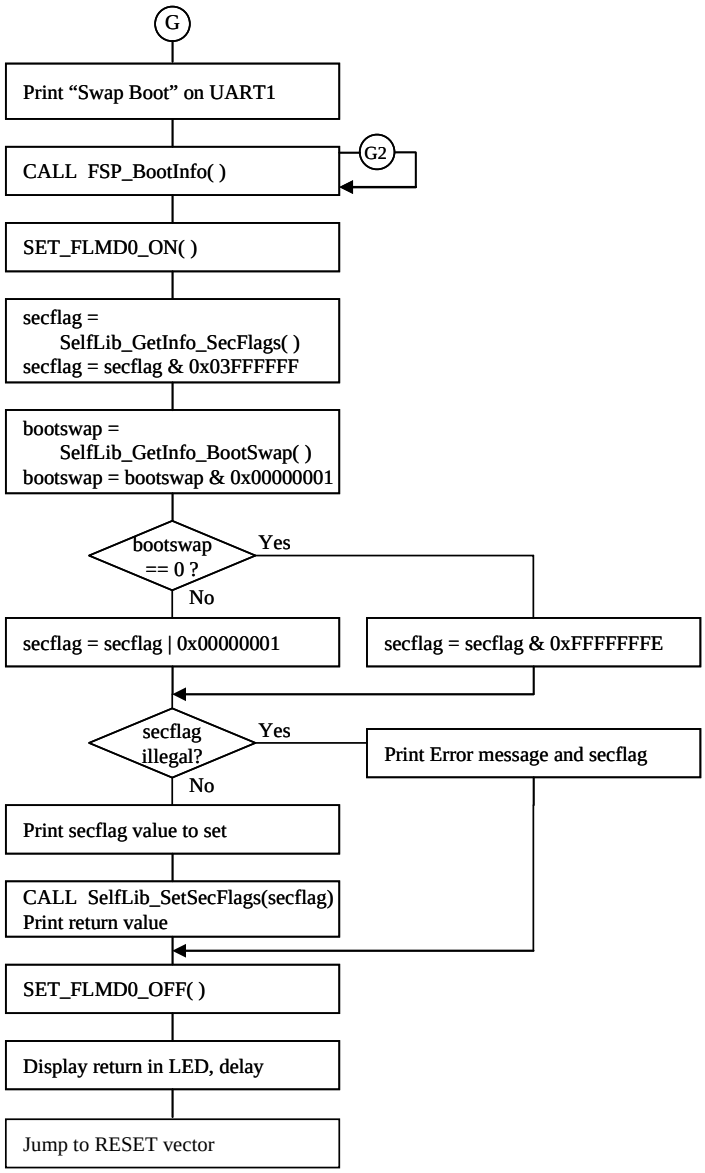
This results in a call to the instruction located at the start of the user program area, which is 00008000H in the demonstration program. It is assumed that this location contains a jump to the start of the user program. Please see later sections on the construction of the user program, to see how this is assured.

In this demonstration program, there is no checking of the loaded area for a valid program. In an actual application, a boot loader would check for a proper program loaded to memory.

In the demonstration program, the user program will not return to the boot program, so there will be no return from FSP_RunProgram(). The return after the call to **fnptr()** will not be executed.

1.8.8 FSP_BootSwap( ): Swap In Loaded Alternate Boot Program

Figure 13. Swap Loaded Boot Program



The FSP_BootSwap() routine does a boot swap operation, to allow the alternate boot program to execute on the next Reset. The alternate boot program has been loaded by the “B” (Bload) command.

First the routine calls the FSP_BootInfo() routine to report the current state of the boot swap bit and the security flags. It then turns FLMD0 on in preparation for Flash programming operations.

The FSP_BootSwap() routine calls the flash self-programming library routine SelfLib_GetInfo_SecFlags(), and assigns the return value to the **secflag** variable. It then clears the upper six bits of the **secflag** variable; this will result in the upper eight bits being 0x03, if the device has been erased; this is the portion of the security flag value where the size of the boot cluster of the device must be specified. Since we are using blocks 0-3 and 4-7 as boot blocks, we set 0x03 here, to specify four boot blocks.

Note: the documentation on the flash self-programming library states that bit zero of the return value from SelfLib_GetInfo_SecFlags() represents the boot flag bit. This is misleading; the return value of bit zero is one, regardless of the state of the boot swap bit.

The routine then calls the flash self-programming library routine SelfLib_GetInfo_BootSwap() to get the current state of the boot swap bit into the **bootswap** variable. The return value is 0 if the boot swap bit is not set (current boot area is blocks 0-3), and 1 if the boot swap bit is set (current boot area is alternate boot area in blocks 4-7).

To do a boot swap operation, bit zero of the security flags must be set to the same state as the current boot swap bit. If **bootswap** is zero (not swapped), the routine clears bit zero of the **secflag** variable; if **bootswap** is one, the routine sets bit zero of **secflag** to one.

At this point, the **secflag** variable should hold the proper value to write into the security flags in order to flip the boot swap bit on the next reset. Before doing the write of the security flags, a check is done to make sure that bit 5 (boot cluster protect bit) and bit 1 (chip erase disable) are **not** cleared to zero. **WARNING: If bit 5 is zero when the security flags are written, the boot block will no longer be able to be changed, even with an external Flash programmer. If bit 1 is zero when the security flags are written, the chip will no longer be able to be erased, even with an external Flash programmer. Either of these operations will make the device impossible to reprogram!**

If the security flag value to be written has either bit 5 or bit 1 cleared to zero, an error message and the value of the **secflag** variable are printed, and no writing of the security flags is done.

If the security flag value has both bits 5 and 1 set to one, the new value to write to the security flags is printed, and the flash self-programming library routine

SelfLib_SetSecFlags(**secflag**) is called. This will change the state of the boot swap bit, effective on the next device RESET. The return value from the routine is printed; the value should be 00000000 to indicate correct writing of the security flags.

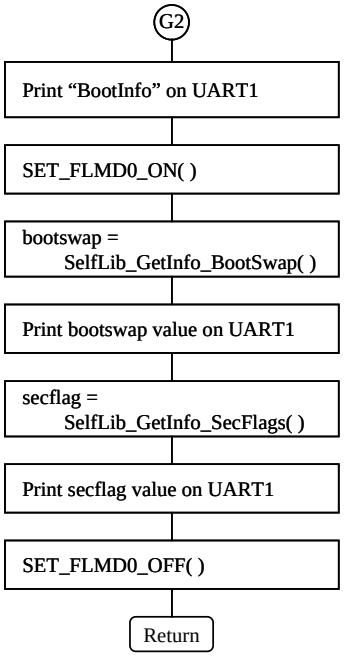
The FSP_BootSwap() routine sets FLMD0 off, and calls FSP_BootInfo() to report the state of the boot swap bit and security flags. **Note:** at this point, the current state of the boot swap bit has not been changed, and will show the same value as before the security flags were written. The boot swap bit will not change until a RESET is done.

The routine shows the return value from the SelfLib_SetSecFlags() routine in the LED, delays for a short time to make this visible, and then jumps to location 00000000H, which is the RESET vector.

This will cause the program to restart, but is not the same as an actual RESET. The sign-on message and prompt will be displayed, but the message will still indicate the current boot swap area.

1.8.9 FSP_BootInfo( ): Report State of Boot Swap Bit and Security Flags

Figure 14. Boot Swap Information Processing



The FSP_BootInfo() routine is called from main() when the “I” command is given, and from FSP_BootSwap() before and after doing a boot swap operation. It reports the current state of the boot swap bit and the security flags.

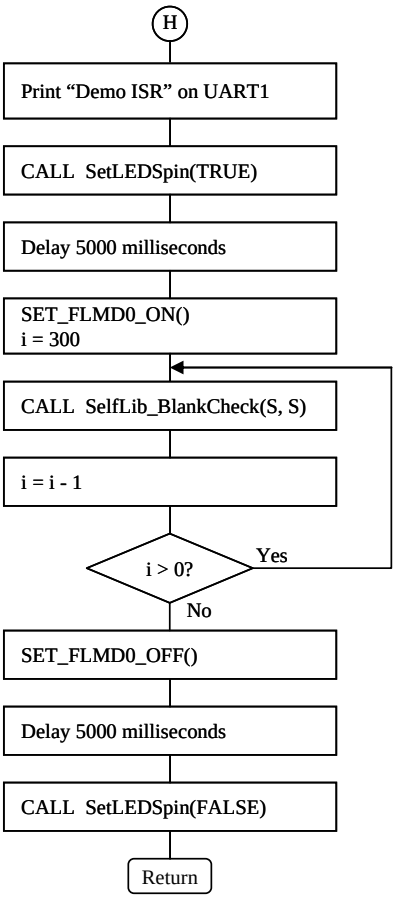
The routine first prints the message “BootInfo” on UART1, and then turns FLMD0 on to enable Flash self-programming. It calls the flash self-programming library routine SelfLib_GetInfo_BootSwap(), and prints the return value. The return value is 0 if the boot swap bit is not set (current boot area is blocks 0–3), and 1 if the boot swap bit is set (current boot area is alternate boot area in blocks 4-7).

The routine then calls the flash self-programming library routine SelfLib_GetInfo_SecFlags(), and prints the return value. The return value is the current state of the security flags. The routine then turns FLMD0 off, and returns to the calling routine.

1.8.10 FSP_DemoISR(): Demonstrate Interrupts During flash self-programming Mode

The FSP_DemoISR() routine demonstrates the different handling of interrupts during flash self-programming mode and non-flash self-programming mode, by servicing the periodic millisecond timer differently depending on mode, and affecting the LED to provide a visual display of the difference.

Figure 15. Interrupt Processing During Self-Programming



The routine first prints “Demo ISR” on UART1 in response to the command, and calls the routine SetLEDSpin(TRUE). This sets a global variable, **ledSpin**, to the TRUE state; this variable is checked in the timer interrupt service routine. If it is true, the timer interrupt service routine will rotate the data in the I/O port controlling either the left or right LED, with one segment lit. This will look like the lit segment is rotating around the LED. Which LED is rotated will depend on whether the interrupt was serviced during flash self-programming mode or not.

The FSP_DemoISR() routine then delays for 5000 milliseconds, or 5 seconds. This is done by setting a millisecond counter, which is counted down by the timer interrupt, and checking for the timer to run down to zero. During this time, the processor is not in flash self-programming mode; timer interrupts will be serviced by using the vector in low Flash memory and interrupt service code in flash memory. This code will cause the right LED to spin at the rate of 3 changes per second.

The FSP_DemoISR() routine then sets FLMD0 on, and calls the flash self-programming library routine SelfLib_BlankCheck(S, S), where S is the block number of the start of the user program area. This will check whether the first block of the user program area is blank or not. We are not interested in the return value; the purpose of the call is to have the processor enter flash self-programming mode. It will return briefly to non-flash self-programming mode, and then reenter it to do the next blank check. The blank check routine will take some time for each call; in the µPD70F3318Y (V850ES/KJ1+) device running at 20MHz used in the demonstration program, each check will take 12 milliseconds (ms). The loop of 300 calls will take 3600 ms; with the overhead of the calls, this will take about 4 seconds.

During this time, the processor will be spending most of its time in flash self-programming mode. Timer interrupts will be occurring every 1 ms. These interrupts will be serviced, with a latency of 40 microseconds or less, but they will be handled by the interrupt service code copied to the start of RAM, not by the interrupt vector and code in flash memory. This code will cause the left LED to spin rather than the right, and at a rate of one change every 100 milliseconds.

While the processor is doing the 300 blank checks, the spinning will switch from the right LED to the left, and seem to speed up.

Once the blank checks are completed, the processor turns FLMD0 off, and does another 5000 millisecond delay. During this time, the processor will not be in flash self-programming mode, and the right LED will seem to spin again, at the slower rate.

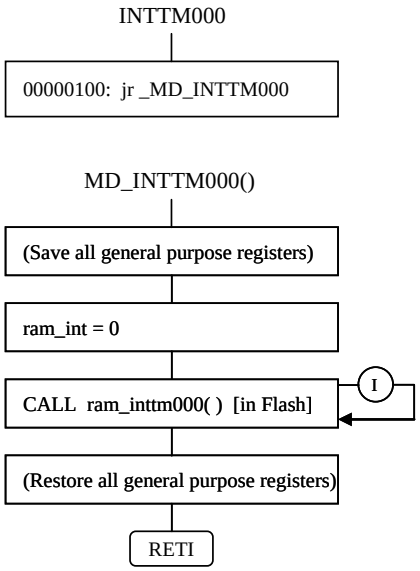
When the FSP_DemoISR() routine is finished with this delay, it sets the **ledSpin** variable FALSE, which will stop the timer interrupt service routine from spinning the LED, and returns to the main() program.

1.8.11 MD_INTTM000( ): Interrupt Service for INTTM000 (Non-FSP Mode)

The TM00 timer peripheral has been initialized to give an interrupt every 1 millisecond, and the interrupt enabled. When the interrupt occurs, named INTTM000, the action of the processor depends on whether it is in flash self-programming mode or not.

In non-flash self-programming mode, the INTTM000 interrupt will set the interrupt flag TM0IF00 in the TM0IC00 interrupt control register. If interrupts are enabled, and no higher priority interrupt is pending, the processor will save the PC and PSW registers in special system registers, change the PSW to disable interrupts, clear the TM0IF00 flag, and set the PC to the address of the interrupt vector for INTTM000, which is 0x00000100.

Figure 16. Example of Timer Interrupt Processing



The demonstration program sets the interrupt vector located at 0x00000100 with an instruction which jumps to the interrupt service routine MD_INTTM000().

The MD_INTTM000() routine has been declared with the special reserved word "__interrupt"; this tells the compiler to insert instructions at the start of the routine to save all of the general purpose registers which may be used by C code in this routine or others called by this routine. These registers are saved on the stack.

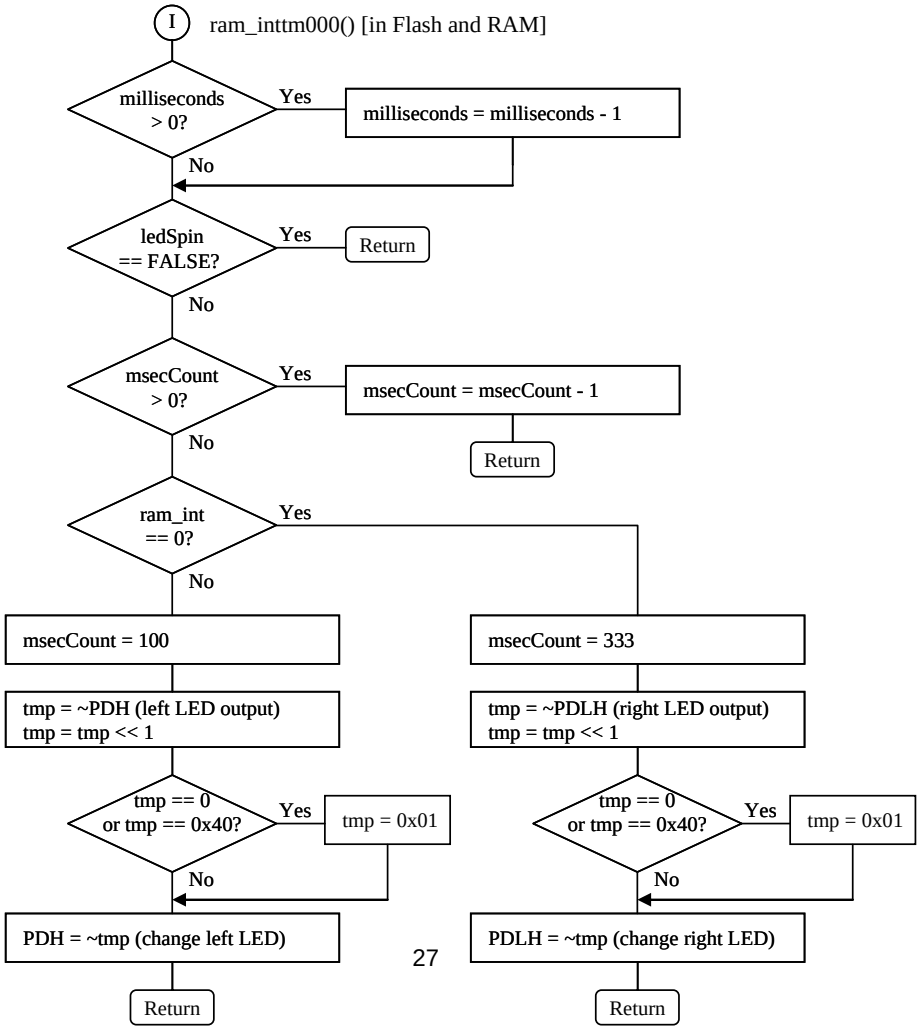
The MD_INTTM000() routine sets the variable **ram_int** to zero; this will tell the ram_inttm000() routine that the interrupt was serviced in non-flash self-programming mode by the code in flash memory.

MD_INTTM000() then calls the ram_inttm000() routine to handle the timer interrupt. **Note:** the ram_inttm000() code is in a special section of the program, and has been copied to RAM by the FSP_Init() function, so there are actually two identical copies of the code for ram_inttm000() in memory, one in the original Flash memory location, and one in RAM. MD_INTTM000() calls the location in flash memory.

1.8.12 ram_inttm000( ): Interrupt Service for INTTM000 (FSP Mode and Non-FSP Mode)

When ram_inttm000() returns, MD_INTTM000() restores all saved general purpose registers from the stack, and then executes a RETI instruction (return from interrupt). This restores the PC and PSW registers from the special system registers where they were saved, and the program continues from the point where the INTTM000 interrupt occurred.

Figure 17. Interrupt Service Processing in Internal RAM



The ram_inttm000() routine exists in two places in the demonstration program. The code is compiled and placed in flash memory, where it is called from MD_INTTM000() when not in flash self-programming mode. A copy of the code also exists in internal RAM, where it has been copied by the FSP_Init() routine; the RAM version is called from the userFunc() routine, also located in RAM, when the INTTM000 interrupt occurs in flash self-programming mode.

The routine is executed once every millisecond, caused by the TM00 timer interrupt INTTM000. The routine counts down a variable, **milliseconds**, which is used to time delays. If **milliseconds** has been counted down to zero, it is left as zero. The routines SetMsecTimer() and CheckMsecTimer() are used to set and check this variable.

The routine then checks the **ledSpin** variable; if this variable is FALSE (LED should not spin), the routine returns without affecting the LED.

If **ledSpin** is TRUE, a separate millisecond counter, **msecCount**, is counted down; if it has not reached zero, the program returns without affecting the LED.

If **msecCount** has reached zero, the action taken depends on the **ram_int** variable, which has been set to zero if the interrupt occurred in non-flash self-programming mode, or to one if in flash self-programming mode.

If not in flash self-programming mode (**ram_int** is zero), the **msecCount** variable is set to 333. The contents of the PDLH register are read and inverted into a temporary variable, shifted one bit to the left, and reinverted and written back to the PDLH register. The PDLH register is the output port connected to the right LED; bits which are zero cause lit segments, and bits which are one cause off segments. The shifting of the inverted data will cause a single lit segment to move around the LED from the “A” segment to the “B” segment, or from “B” to “C”, etc. Once the value of the temporary variable reaches 0x40 (corresponding to the “G” middle horizontal segment), or reaches zero by shifting all bits to the left, the value is set back to 0x01, which will cause the “A” segment to be lit.

In non-flash self-programming mode, this will cause the right LED to look like a segment is rotating clockwise around the LED, with the segment changing every time **msecCount** reaches zero, every 333 milliseconds, or three times per second.

If in flash self-programming mode (**ram_int** is one), the **msecCount** variable is set to 100. The contents of the PDH register are read and rotated; the PDH register is the output port connected to the left LED.

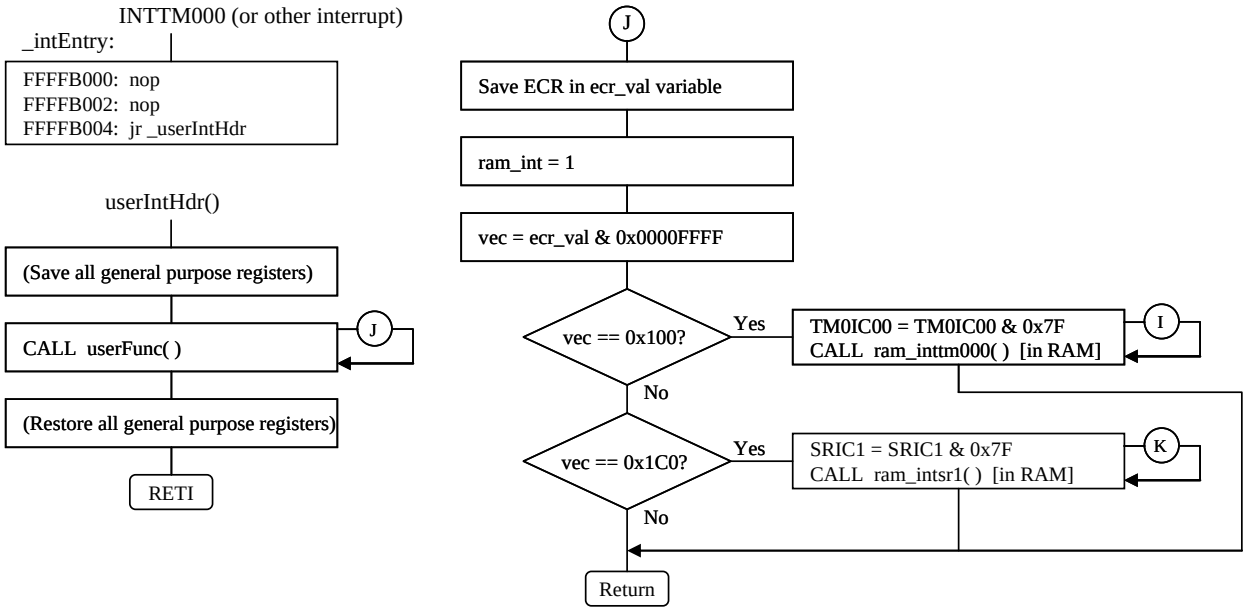
In flash self-programming mode, this will cause the left LED to look like a segment is rotating clockwise around the LED, with the segment changing every 100 milliseconds, or ten times per second.

The visible result, seen when the FSP_DemoISR() routine is executed, is a slowly spinning right LED when not in flash self-programming mode, and a more rapidly spinning left LED in flash self-programming mode.

1.8.13 _int_Entry, _userIntHdr, userFunc( ): Interrupt Service (FSP Mode)

For any interrupts that must be enabled and serviced during flash programming operations, the normal interrupt vectors at the bottom of flash memory may not be available. Instead, the processor will save the PC and PSW registers in special system registers, and branch to the start of internal RAM for interrupt service. Code must be located here to handle the interrupts, and so is placed in the special section “SelfLib_ToRamUsrInt”. The code for this section is in the assembly language source file userFuncInt.s, and has the entry label _intEntry. The code has been copied to the start of internal RAM by the FSP_Init() routine.

Figure 18. ISR Code Copied to Internal Ram



Note: the details of interrupt handling in flash self-programming mode depend on the particular processor. For the uPD70F3318Y (V850ES/KJ1+) processor used for the demonstration program, interrupts vector to the start of RAM. Some processor have programmable vectors which can be set, to allow interrupts to vector directly to service

routines. Please see the details of interrupt handling described in the NEC documentation for the flash self-programming library.

The code at **_intEntry** is located at FFFFB000H, which is the start of internal RAM for the µPD70F3318Y. The code consists of two **nop** instructions, followed by a jump to the start of the userIntHdr() routine. The reason for the two nop instructions is that non-maskable interrupts will branch to FFFFB000, and non-maskable interrupts will branch to FFFFB004. This code branches to the routine usrIntHdr() regardless of whether the cause was a non-maskable or maskable interrupt.

The code for interrupt handling is in the special section “SelfLib_ToRamUsr”, and is in the C source file userFunc.c. This code has been copied to RAM by the FSP_Init() routine. Because the V850ES code for relative branches is relative to the PC register, the branch to usrIntHdr(), which originally branched from Flash memory to Flash memory, now branches from RAM to a location in RAM located the same relative distance as the original branch.

The user interrupt handler usrIntHdr() is declared as “__interrupt”, so the compiler will insert instructions to save all of the general purpose processor registers on the stack. The routine then calls the routine userFunc(). When this routine returns, the general purpose registers are restored, and a RETI instruction is executed to restore the PC and PSW registers. The program will resume at the point where the interrupt occurred in flash self-programming mode.

The userFunc() routine must determine the type of interrupt which occurred, and service it properly. It saves the value of the special system register ECR (Exception Cause Register) which will contain the vector number of the interrupt, in the variable **ecr_val**. It masks the value to the lower 16-bits, which contains the cause of a maskable interrupt. The upper 16-bits would contain the cause of a non-maskable interrupt. If the interrupt were caused by the TM00 timer interrupt INTTM000, the value will be 0x0100; if the interrupt were caused by the UART1 serial reception interrupt INTSR1, the value will be 0x01C0. These values correspond to the address of the vector in low Flash memory which would be branched to if not in flash self-programming mode.

If the interrupt has been caused by the INTTM000 interrupt, the userFunc() routine clears the interrupt flag TM0IF00, which is bit 7 in the interrupt control register TM0IC00. If the interrupt were serviced in non-flash self-programming mode, the processor would automatically clear this bit when the interrupt was serviced. InfFlash self-programming mode, interrupt request flags are **not** automatically cleared, and must be cleared by the program. The routine then calls the ram_inttm000() routine; this will be done with a relative branch instruction, so the code executed will be the copy of ram_inttm000() which has been copied from Flash memory to internal RAM.

If the interrupt has been caused by the INTSR1 interrupt, the interrupt request flag SRIF1 in the SRIC1 interrupt control register is reset, and the ram_intsr1() routine is called. This will be the copy of ram_intsr1() which is in RAM, not the original code located in flash memory.

If any other interrupt has occurred, it is not handled, and userFunc() returns. This has the effect of ignoring any other interrupts while in flash self-programming mode.

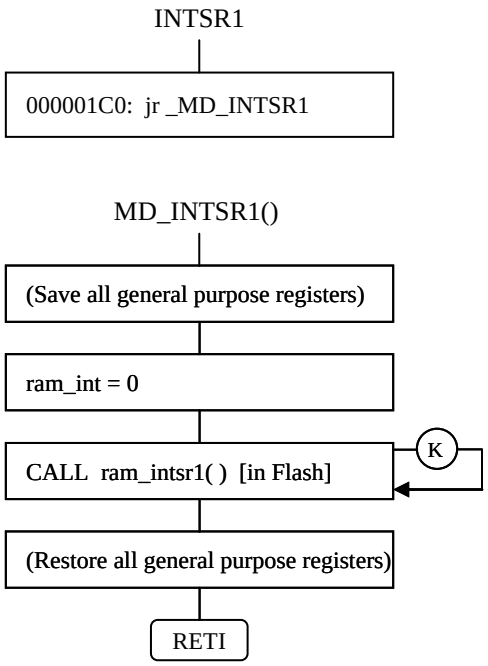
All of the code for dealing with interrupts in flash self-programming mode is a copy located in RAM, and does not require the use of the original code in flash memory. The structure of this handling of interrupts is provided in the routines userIntHdr() and userFunc() supplied in the flash self-programming library.

Note: it would be possible to eliminate the userFunc() routine, and locate all of its code inside the userIntHdr() routine, saving a function call and return, and a stack location. The current structure follows the model of the flash self-programming library as supplied by NEC Electronics America.

1.8.14 MD_INTSR1( ): Interrupt Service for UART1 Receive (Non-FSP Mode)

The UART1 peripheral has been initialized to give an interrupt when a serial character is received. When the interrupt occurs, named INTSR1, the action of the processor depends on whether it is in flash self-programming mode or not.

Figure 19. UART Interrupt Processing in Normal Mode



In non-flash self-programming mode, the INTSR1 interrupt will set the interrupt flag SRIF1 in the SRIC1 interrupt control register. If interrupts are enabled, and no higher priority interrupt is pending, the processor will save the PC and PSW registers in special system registers, change the PSW to disable interrupts, clear the SRIF1 flag, and set the PC to the address of the interrupt vector for INTSR1, which is 0x000001C0.

The demonstration program sets the interrupt vector located at 0x000001C0 with an instruction which jumps to the interrupt service routine MD_INTSR1().

The MD_INTSR1() routine has been declared with the special reserved word "__interrupt"; this tells the compiler to insert instructions at the start of the routine to save all of the general purpose registers which may be used by C code in this routine or others called by this routine. These registers are saved on the stack.

The MD_INTSR1() routine sets the variable **ram_int** to zero; this will tell the ram_intsr1() routine that the interrupt was serviced in non-flash self-programming mode by the code in flash memory.

MD_INTSR1() then calls the ram_intsr1() routine to handle the received character interrupt. **Note:** the ram_intsr1() code is in a special section of the program, and has been copied to RAM by the FSP_Init() function, so there are actually two identical copies of the code for ram_intsr1() in memory, one in the original flash memory location, and one in RAM. MD_INTSR1() calls the location in flash memory.

When ram_intsr1() returns, MD_INTSR1() restores all saved general purpose registers from the stack, and then executes a RETI instruction (return from interrupt). This restores the PC and PSW registers from the special system registers where they were saved, and the program continues from the point where the INTSR1 interrupt occurred.

1.8.15 ram_intsr1( ): Interrupt Service for UART1 Receive (FSP Mode and Non-FSP Mode)

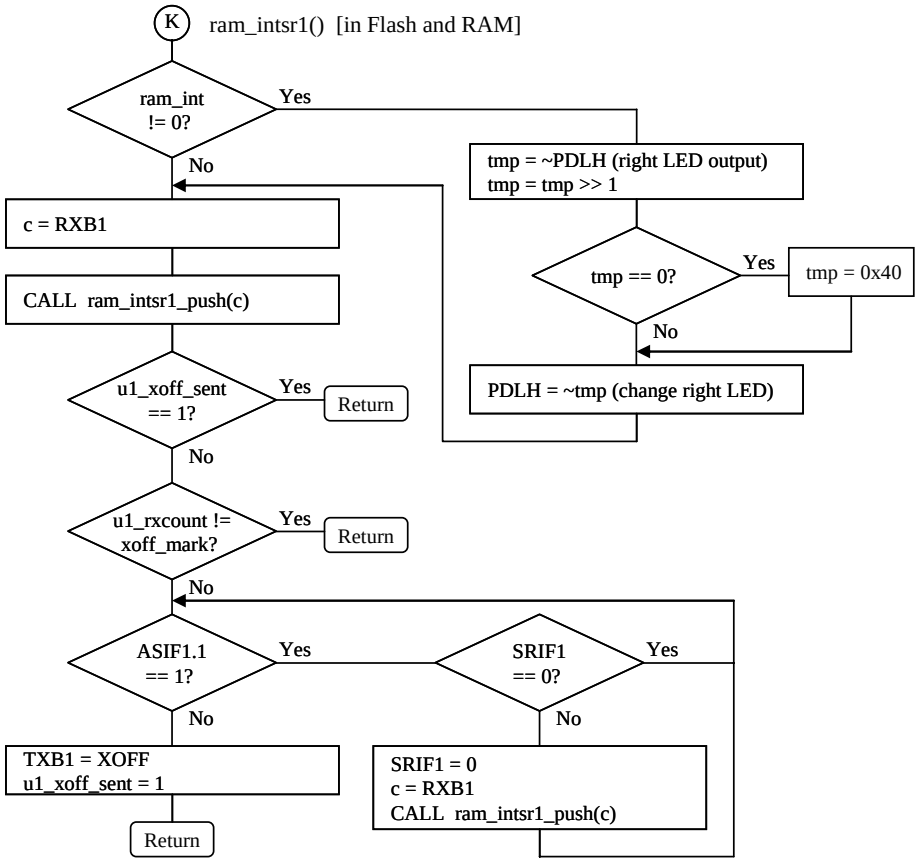
The ram_intsr1() routine exists in two places in the demonstration program. The code is compiled and placed in flash memory, where it is called from MD_INTSR1() when not in flash self-programming mode. A copy of the code also exists in internal RAM, where it has been copied by the FSP_Init() routine; the RAM version is called from the userFunc() routine, also located in RAM, when the INTSR1 interrupt occurs while in flash self-programming mode.

The routine is executed when UART1 receives a character, and asserts the serial receive interrupt INTSR1. The initial action taken depends on the **ram_int** variable, which has been

set to zero if the interrupt occurred in non-flash self-programming mode, or to one if in flash self-programming mode.

If not in flash self-programming mode (**ram_int** is zero), no change is made to the LED.

Figure 20. UART Interrupt Processing during Flash-Self Programming



If in flash self-programming mode (**ram_int** is one), the contents of the PDLH register are read and rotated; the PDLH register is the output port connected to the right LED. Unlike the action taken on a timer interrupt, the display data is rotated right, which will result in the left LED rotating counterclockwise every time a character is received in flash self-programming mode. When loading data and writing it to Flash memory in the FSP_LoadProgram() routine, this will show characters received and service of the INTSR1 interrupt while in flash self-programming mode.

The received character is read from the RXB1 register, and saved to the receive buffer by calling the ram_intsr1_push() routine. This will save the character and update pointers to the buffer and the count of received characters.

The variable **u1_xoff_sent** is checked, and if already one (an XOFF has been sent) the routine returns. If XOFF has not been sent, the count of characters received is compared to the set point where an XOFF should be sent to pause the sender. If the count has not reached this point, the routine returns.

If an XOFF has not been sent, and the count has reached the point where it should be sent, the routine checks the ASIF1 register bit 1, which indicates that a character may be written to the transmit buffer. If the bit is set to 1, indicating the buffer is busy, the routine waits for it to be available.

So that no received characters are missed while waiting for the transmit buffer to be available, the SRIF1 interrupt flag is checked to see if another serial characters has been received. If it has, the flag is cleared, the character is read and saved to the receive buffer.

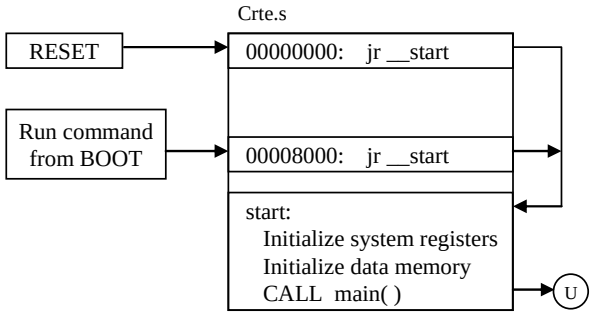
When the transmit buffer is available, an XOFF character is sent, and the **u1_xoff_sent** flag is set. When enough characters have been taken out of the receive buffer by the program, an XON will be sent and this flag will be cleared.

1.8.16 User Application Program (UA_1 and UA_2) Startup and Initialization

For V850ES programs written in the C language, the startup code for the C program is supplied by an assembly language startup file, generally named crte.s. This startup code specifies the Reset vector which determines where the program will begin on a hardware Reset, and provides initial setup of system registers before calling the main() function.

The user application programs, UA_1 and UA_2, were not generated by Applilet. They instead use a version of crte.s provided as a template with the NEC Electronics C compiler.

Figure 21. User Program Startup and Initialization



The startup code in the standard crte.s file provides a Reset vector, to be located at 00000000H, which branches to the start of the startup code, normally located in low Flash memory above the interrupt vectors. The startup code initializes system registers, clears data

memory areas used for uninitialized variables, sets up **argc** and **argv** variables for possible use by main(), and then calls the main() routine.

Note that the standard crte.s does not do any clock or peripheral initialization prior to calling main(); the main() routine must provide any necessary hardware initialization.

For use in this application note, the standard startup code has been modified slightly, to provide an additional branch instruction to the start of the startup code. The modified file is renamed to crte_8000.s. The standard link directive file has also been modified, to locate this second entry branch at 00008000H, and to locate the program code following this location.

This allows the user application program to be loaded at 00008000H by the boot program, and executed by branching to the entry address at 00008000H. The Reset vector at 00000000H, and the option and security bytes for the user program, located at 00000070H – 0000007FH, are not loaded; these are not necessary for execution of the user program.

The main() routine for user application program UA_1 calls hdwinit() to initialize hardware registers, such as clock and peripherals, and calls routines to initialize the LED and switch inputs. It sets the initial value of the **count** variable to zero, and displays it in the two-digit LED as a decimal number.

The main() routine calls the sw_get() routine to get a debounced switch input. It checks the return value for one of the two switches, SW2 or SW3, to be down.

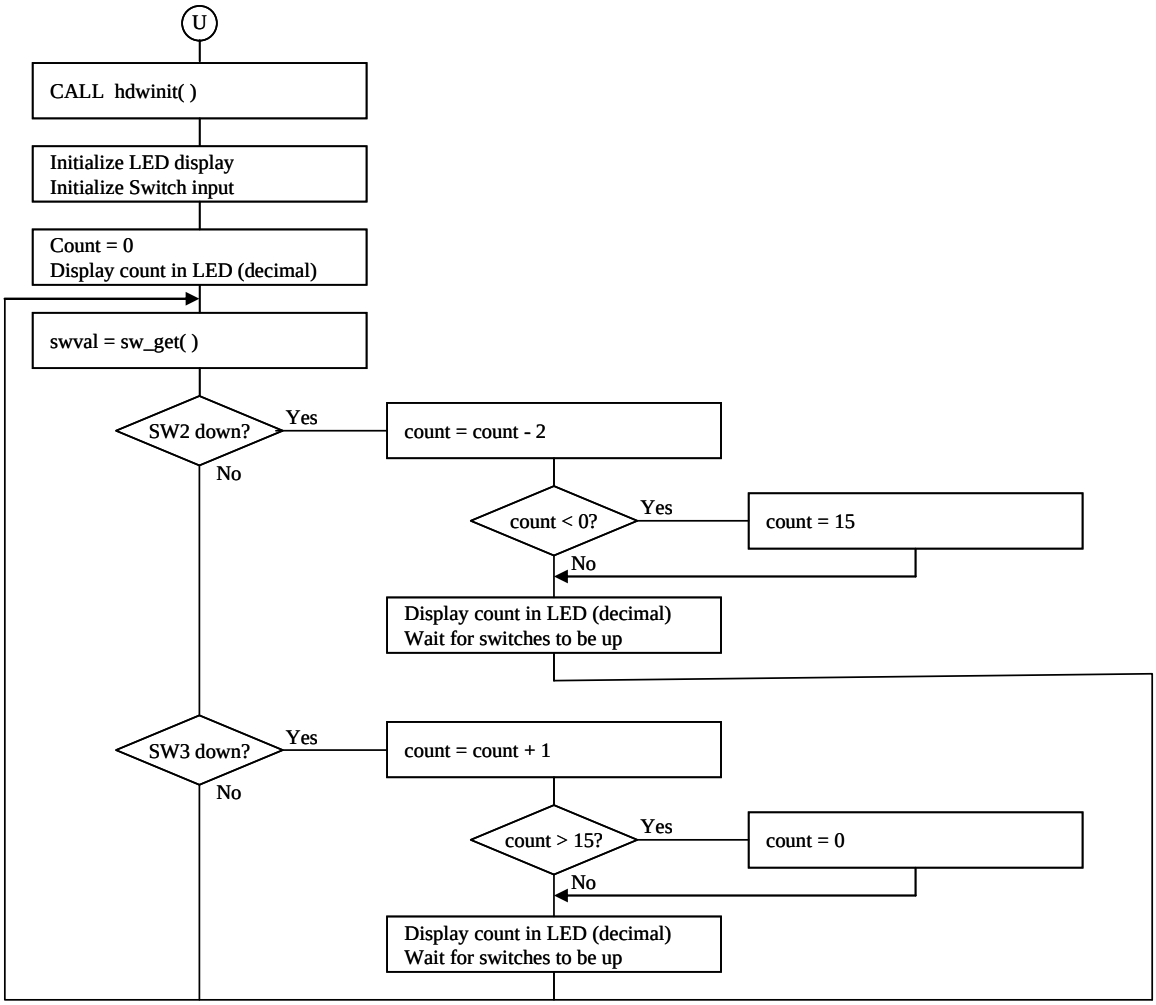
If SW2 is down, the value of **count** is decreased by two; if this results in a value below zero, the maximum value of 15 is set. The new value of **count** is displayed in the LED, and the program waits for the switches to be up.

1.8.17 User Application Program UA_1 Main() Routine

If SW3 is down, the value of **count** is incremented by one; if this results in a value above the maximum of 15, zero is set. The new value of **count** is displayed in the LED, and the program waits for the switches to be up.

The main() routine then continues checking switches. The result is a program which will count up the display in decimal by one when SW3 is pressed, or down by two when SW2 is pressed, and rolling over at the value 15.

Figure 22. User Program UA_1 Main Routine



1.8.18 User Application Program UA_2 Main() Routine

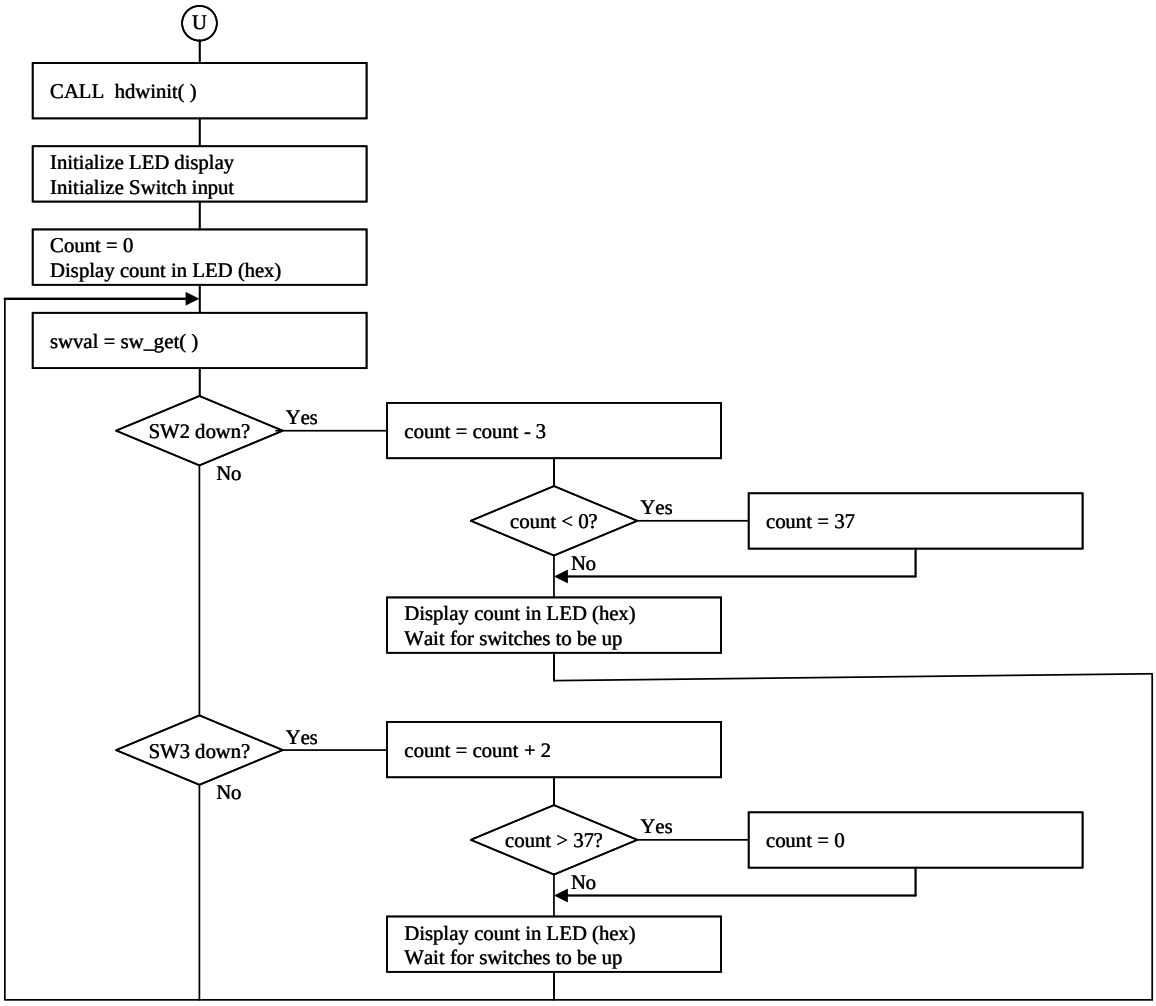
The main() routine for user application program UA_2 is basically the same structure as UA_1, with changes in the way data is displayed, count up and count down amounts, and rollover value.

After initialization, the program initially displays the count of zero as two hexadecimal digits, rather than a single decimal zero.

If SW2 is down, the value of **count** is decreased by three; if this results in a value below zero, the maximum value of 37 (25H) is set. The new value of **count** is displayed in the LED, and the program waits for the switches to be up.

If SW3 is down, the value of **count** is incremented by two; if this results in a value above the maximum of 37 (25H), zero is set. The new value of **count** is displayed in the LED, and the program waits for the switches to be up.

Figure 23. User Program UA_2 Main Routine



The main() routine then continues checking switches. The result is a program which will count up the display in hexadecimal by two when SW3 is pressed, or down by three when SW2 is pressed, and rolling over at the value 25H.

1.9 Applilet's Reference Driver

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

Applilet was used to produce the starting point for the boot program. Because of specialized space requirements for the boot program, modifications to interrupt handling, and the addition of the NEC Electronics flash self-programming library code, several of the source files produced by Applilet have been modified. Details of the modifications is noted in sections below. The example user programs loaded by the boot program were not produced by Applilet.

For the boot program, Applilet is used to produce the basic initialization code and main function for the program, clock initialization code, and initialization and driver code for the UART1 and timer TM00 peripherals. After Applilet produces the basic code, additional code is added by the user to customize the functioning of the program.

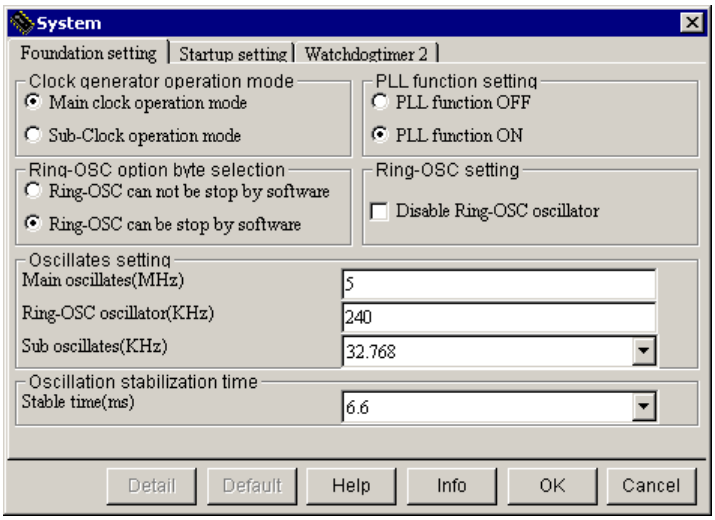
This section describes how Applilet is set up to produce code for these peripherals, and lists the files and routines produced. Additional files not generated by Applilet, such as those written for flash self-programming functions, and the flash self-programming library, are also listed.

Applilet is started, a new project file is created and saved as a .prx file. Applilet shows a screen allowing different peripheral blocks to be selected for setup.

1.9.1 Configuring Applilet for Clock Initialization

On the **System** properties sheet, and the system settings for clocks are displayed. The “Foundation Setting” tab sets which clocks will operate at startup.

Figure 24. Clock Setup



This dialog controls how the clocks are initialized in the Clock_Init() routine. The “Main clock operation mode” is selected to operate on the external crystal; “PLL function On” is

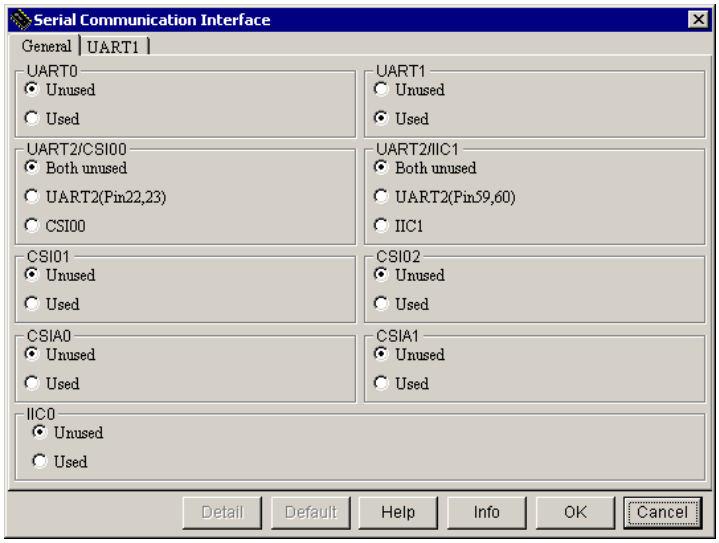
selected to set the clock to use the PLL multiplier. The external clock frequency is set at 5 MHz; with the PLL on, this results in a system clock of 20 MHz.

The option “ring oscillator can be stop by software” is selected; if this option is not selected, the watchdog timer 2 (WDTM2) cannot be stopped. The watchdog timer must be stopped in order not to reset the processor during flash self-programming.

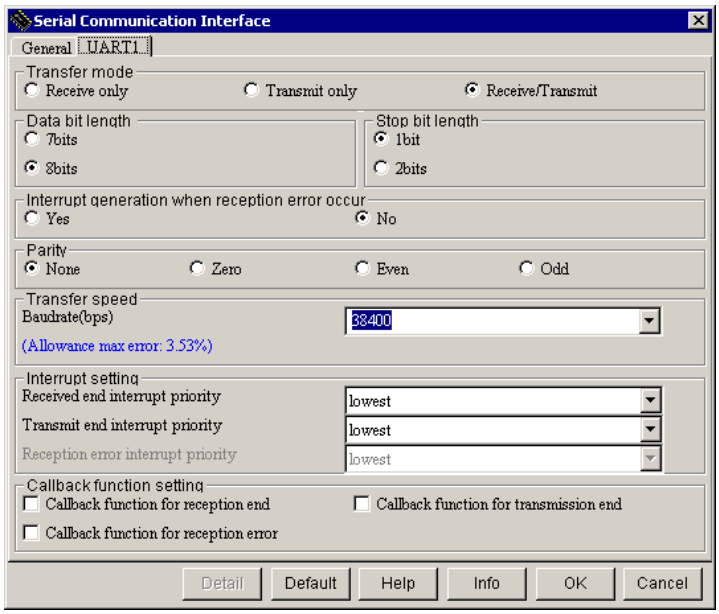
1.9.2 Configuring Applilet for UART1 Serial Communication

1. On the **General** tab of the **Serial Communication Interface** property sheet, click **Used** in the UART1 box to display the **UART1** tab.

Figure 25. UART Setup



2. On the UART1 tab, in the **Transfer mode** box, click **Receive/Transmit**.
3. In the **Data bit length** box, click **8bits**.
4. In the **Interrupt generation when reception error occur** box, click **No**.
5. In the **Parity** box, click **None**.
6. In the **Transfer speed** box, select **38400**.



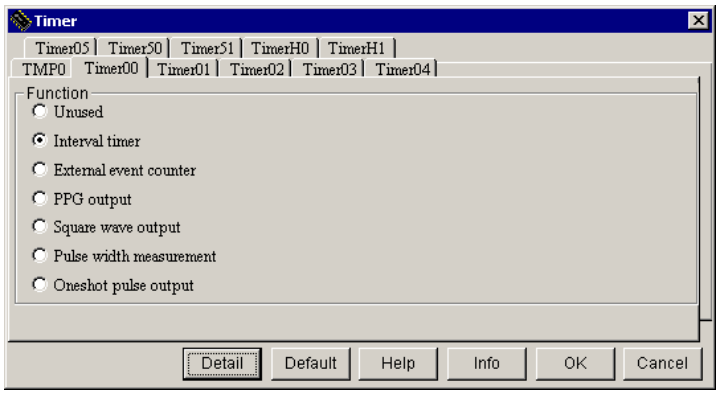
7. In the **Interrupt setting** box, select lowest for R**eceived end interrupt priority** and **Transmit end interrupt** priority.

8. Clear the options in the **Callback function setting** box.

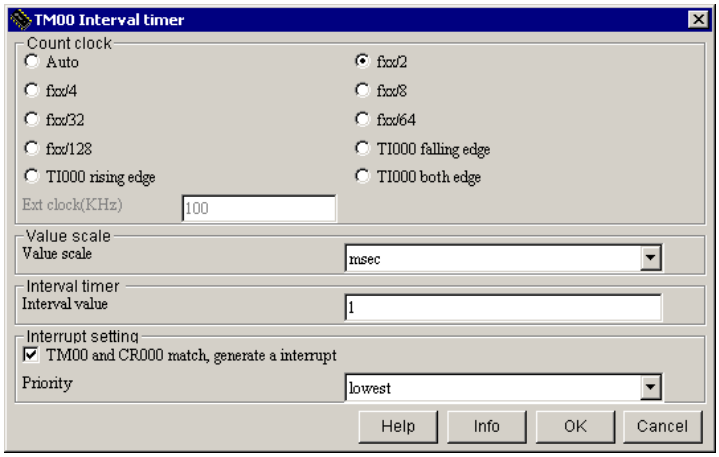
Note: The Applilet-generated code for UART1 initialization is used as a basis for initialization in the boot program, but other Applilet-generated routines for UART1 communication are not used. In the boot program, the receive end interrupt is used, but the transmit end interrupt is not used.

1.9.3 Configuring Applilet for Timer 00 (TM00)

1. On the **Timer00** tab in the **Timer** properties sheet, select **Interval timer** to provide a periodic interrupt.



2. Click **Detail** to set the Timer 00 interval timer.



3. Click **fxx/2** to use a 10 MHz clock for the timer.

4. Select **msec** in the **Value scale** box.

5. Select **TM00 and CR000 match** in the **Interrupt setting** box so that Timer 00 will generate an interrupt every millisecond. This interrupt is used for timing millisecond intervals for delays, as well as for demonstrating interrupt functions during flash self-programming.

1.9.4 Generating Code With Applilet

Once the various dialog boxes are set up, select the “Generate code” option. Applilet will show the peripherals and functions to be generated, and allows you to select a directory to store the source code.

When the “Generate” button is pressed, Applilet creates the code in several C-language source files (extension .c), C header files (extension .h), and assembly language source and header files (extensions .s and .inc), and shows the list of files created in a dialog box.

To support the initial startup code, Applilet generates assembly source file crte.s. For clock initialization, Applilet generates system.inc and system.s. The SystemInit() function is generated in systeminit.c. For specification of unused interrupt vectors, the assembly source file inttab.s is generated.

To support the UART1 peripheral, Applilet generates serial.h, serial.c, and serial_user.c. To support the Timer 00 peripheral, Applilet generates timer.h, timer.c and timer_user.c.

Several other files are generated, including a main.c file with a blank main function. Applilet also generates a link directive file, 850.dir, to control the linking process.

1.9.5 Applilet-Generated Files

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

File	Function
Macrodriver.h	General header file for Applilet-generated programs
Crte.s	Reset vector, program startup code
System.inc	Assembly-language header for system.s
System.s	Assembly source for Clock_init() routine
System_user.s	Empty file (would contain code for System interrupt if used)
Systeminit.c	SystemInit() routine for peripheral initialization
Main.c	The main program routine
Inttab.s	Interrupt vectors with RETI for unused interrupts
Serial.h	Header file for serial.c
Serial.c	UART1 and CSI00 functions generated by Applilet
Serial_user.c	Callback functions for UART1 and CSI00 for user code
Timer.h	Header file for timer.c
Timer.c	Timer 00 functions
Timer_user.c	User code for INTTM000 interrupt
850.dir	Link directive file

1.9.6 Modifications to Applilet-Generated Files For Demonstration Program

To accommodate the specialized needs of the boot program, several modifications have been made to the source files generated by Applilet. The modifications are listed below by file.

1.9.6.1 Crte.s

The file crte.s contains the reset vector, and the code executed on a reset to initialize the program. System registers are set, data areas are cleared, calls to Clock_Init() and SystemInit() are made, and then the main() routine is called.

The standard crte.s code as produced by Applilet includes initialization of variables **argc** and **argv[0]**, which are then passed to main() as parameters, as if the call were main(argc, argv). Since these variables are not used, the code to initialize them has been commented out in crte.s.

The handling of interrupts during flash self-programming mode requires more space on the stack than the standard 512 bytes reserved in crte.s as generated by Applilet; this has been increased to 1024 bytes.

A specialized data section, S_ROM_END, with a specific value, has been added to this file. The link directive file will located this value at the end of the boot program space. A program which loads the boot program can check for this value being written to flash

memory, to insure the boot program has been completely loaded; also, placing this value at the end of the boot program ROM space will insure an error message if the boot program code exceeds the size of the boot program area.

1.9.6.2 System.s

A slight error in the Clock_Init() function was corrected, to avoid excessive delay in waiting for the internal PLL to stabilize.

1.9.6.3 Main.c

The blank main() function as generated by Applilet has been replaced with the demonstration program code; additional “#include” and “#define” directives have been added for the demonstration program.

Message strings, which are initialized constant data, are placed in the special data section “S_SCONST0” rather than in the standard section “.sconst”, to allow them to be located in a separate memory area.

1.9.6.4 Inttab.s

In the standard inttab.s file as produced by Applilet, there are short sections of code for every interrupt vector supported by the processor, which is **not** used by the program. These sections provide a **reti** (return from interrupt) instruction for each of these vectors, in case such interrupts are accidentally enabled and occur. An example is the code:

```
section "INTP0", text                --INTP0 pin
reti
```

This places the **reti** instruction in a code section named “INTP0”. The NEC Electronics compiler will automatically locate this segment at the proper address for the INTP0 interrupt vector, which is 00000090H. If the INTP0 interrupt is enabled and occurs, the processor will execute the code at 00000090H, which will return from the interrupt, effectively ignoring it.

For interrupt vectors which are used in the program, Applilet produces source lines like the following in serial.c:

```
#pragma interrupt INTSR1 MD_INTSR1
```

This causes the NEC compiler to automatically create code at the interrupt vector for INTSR1 to branch to the MD_INTSR1() routine; if written in assembly language, this would look like:

```
section “INTSR1”, text
jr _MD_INTSR1
```

All of the interrupt vectors specified in inttab.s and by “#pragma interrupt” statements in C source files are collected into one memory segment, INT, by the linker. This results in the memory area from 00000000H through the highest interrupt vector location specified (INTIIC1 at 00000340H) to be taken up by interrupt vectors. A special section for security codes and option bytes is also placed in this area, from 00000070H to 0000007F. This area is not available for other code or data uses, even though most interrupts are not used.

In the demonstration program, this area is freed up by changing inttab.s to specify only the interrupt vectors from RESET (00000000H) through ILGOP (00000060H) and the security/option bytes ending at 0000007F. The other interrupt vectors are commented out. This results in an INT segment only from 0000000H through 0000007F, with the area from 00000080 through 000000340H available.

To provide interrupt vectors for the interrupts which are used, INTTM000 at 00000100H and INTSR1 at 000001C0H, code is added to inttab.s to specify the interrupt vectors explicitly, and the “#pragma interrupt” statements in the C source files are commented out.

The remaining “holes” in the memory area from 00000080H to 00000340H are available for program code and/or data. These areas are used to hold special sections of initialized data. See the section below on the modifications to the standard link directive file 850.dir.

Note: the demonstration program also reserves an interrupt vector for the UART0 serial receive interrupt, INTSR0. This interrupt is used during debugging by and internal monitor program, and the memory area for this interrupt vector (00000190H – 00000193H) needs to remain unused. Otherwise, this location could also be used by the demonstration program.

1.9.6.5 Serial.h

The serial.h header file contains definitions and declarations used in serial.c and serial_user.c. The declaration of functions generated by Applilet but unused in the demonstration program are commented out. Definitions of values and declaration of new routines for UART1 serial communication are added.

1.9.6.6 Serial.c

The “#pragma interrupt” statements for the interrupts INTSR1 and INTST1 are commented out (see comments on inttab.s above). All Applilet-generated routines except UART1_Init() and MD_INTSR1 are commented out.

The UART1_Init() routine has been changed from the code generated by Applilet in the following ways:

— Code to properly set stop bits and data length has been added, to correct an error in the code

— The transmit interrupt INTST1 is not enabled

— The routine UART1_Rx_Init() is called after hardware initialization has been done

The MD_INTSR1() routine, which is the target of the vector at 000001C0H, and is executed when INTSR1 occurs in non-flash self-programming mode, has been modified. All Applilet-generated code has been removed and replaced with a call to UART1_Rx_Isr() which in turn calls ram_intsr1().

1.9.6.7 Serial_user.c

The serial_user.c file generated by Applilet is normally empty, and is used for user-written code for serial I/O. The following routines have been added:

void UART1_Rx_Init (void)

This routine, called by UART1_Init(), initializes several variables used in handling characters received when a serial receive interrupt INTSR1 occurs.

void UART1_Rx_Isr (void)

This routine is called by MD_INTSR1() when a serial receive interrupt INTSR1 is received in non-flash self-programming mode. It sets the variable **ram_int** to zero, and calls the ram_intsr1() routine, at the original location in flash memory.

BOOL UART1_Rx_Chk (void)

This routine checks the receive buffer to see if a character is available, returning TRUE if one is available, or FALSE if the buffer is empty.

char UART1_Rx (void)

This routine is used to read a character from the receive buffer. A zero is returned if the buffer is empty.

void UART1_Rx_Clear (void)

This routine empties the serial receive buffer, and waits until no characters are being received.

char * UART1_RxFgets(char *s, int n, int timeout)

This routine gets a string of characters to the buffer pointed to by **s**, terminating when a LF character is received, ending a line, or when a timeout has occurred with no input. A timeout value of zero will cause the routine to wait with no timeout.

void UART1_Tx (char c)

This routine transmits one character from UART1.

void UART1_TxStr (char *str)

This routine transmits a null-terminated string of characters, pointed to by **str**, from UART1.

void UART1_TxNibble (unsigned char nib)

This routine prints a character to UART1 representing the hexadecimal value (0–9, A–F) of the low four bits in the nibble variable **nib**.

void UART1_TxByte (unsigned char byte)

This routine prints two characters to UART1 representing the hexadecimal value of the 8-bit parameter **byte**.

void UART1_TxHalfword (unsigned short hword)

This routine prints four characters to UART1 representing the hexadecimal value of the 16-bit parameter **hword**.

void UART1_TxWord (unsigned long word)

This routine prints eight characters to UART1 representing the hexadecimal value of the 32-bit parameter **word**.

void UART1_TxCRLF (void)

This routine prints two characters to UART1, a CR followed by a LF.

1.9.6.8 Timer.h

The file timer.h contains definitions and declarations related to timer functions. For the demonstration program, definitions have been added for the additional functions in timer_user.c.

1.9.6.9 Timer_user.c

The timer_user.c file as generated by Applilet contains the MD_INTTM000() interrupt service routine. This is empty on generation, for the insertion of user code.

For the demonstration program, the “#pragma interrupt” statement has been commented out, to avoid the automatic generation of an interrupt vector for INTTM000 (see the notes on inttab.s above).

The MD_INTTM000() function has had code added to set the variable **ram_int** to zero, and to call the ram_inttm000() function, at the original fash memory location. This code is executed when the INTTM000 interrupt occurs in non-flash programming mode.

The following functions have been added in timer_user.c.

void SetMsecTimer (int time)

This routine sets the millisecond variable to the specified value; millisecond will be counted down once every millisecond by the INTTM000 interrupt service routine.

BOOL CheckMsecTimer (void)

This routine checks whether the millisecond variable has reached zero, and returns TRUE if it has, and FALSE if it has not.

Void SetLEDSpin (BOOL onoff)

This routine sets the ledSpin variable to the value of the onoff parameter, either TRUE or FALSE. The ledSpin variable is checked in the ram_inttm000() routine, to decide whether to alter the LED or not.

1.9.6.10 850.dir and 850_splib.dir

To specify the location of segments in memory, Applilet produces the standard link directive file 850.dir. This file has been extensively modified, and renamed to 850_splib.dir. The changes in the file are as follows.

The segment CONST, originally located at 0x400 (00000400H), now follows the SCONST section.

Special segments have been added for the interrupt vectors used in the program, specified in inttab.s.

Special SCONST segments are specified in the “holes” in the interrupt vector area, and used to locate initialized data sections. For example, segment S_SCONST0 is defined as starting at 0x80 (00000080H), following the security/option bytes. This segment is used to locate data assigned to segment S_SCONST0.sconst, which is initialized string data in main.c, led_vkj1.c, and fsp.c, preceded by the statement ‘#pragma section sconst “S_SCONST0” begin’.

The SCONST segment for other initialized strings has been located at 0x1C4 (000001C4H), immediately following the INTSR1 interrupt vector. This is followed by the CONST and TEXT segments, containing program data and code.

The special TEXTSL segment is used to locate code for the flash self-programming library routines. The segment S_ROM_END is located at 0x1FFC (00001FFCH–00001FFFH), to mark the end of the boot program.

The bulk of these changes have been made in order to have the code for the boot loader demonstration program fit within the space 00000000H–00001FFFH, which is the size of the boot program area and alternate boot program area in the µPD70F3318Y device used.

Note: The original 850.dir file produced by Applilet located the DATA segment starting at 0x3FFE000 (03FFE000H, equivalent to FFFFE000H), which allows a 4 KB area of RAM below the SFR area starting at 03FFF000H (or equivalent FFFFF000H). The actual device used, the µPD70F3318Y, has 16KB of internal RAM starting at 03FFB000H (or FFFFB000H).

However, since the start of internal RAM must be used for the code placed in RAM by the flash self-programming library, it is not possible to start the DATA segment at 03FFB000H/FFFFB000H. For the demonstration program, the DATA segment has been located at 0xFFFFE000, the same effective location as specified in the original 850.dir. This leaves 12 KB of RAM, from FFFFB000H to FFFFDFFFH, available for the code copied by the flash self-programming library.

The actual amount of RAM taken by the flash self-programming library is much smaller than 12KB; the actual code is slightly less than 1100 bytes. The size of this area will depend on the code for handling interrupts in RAM; boot loader programs could adjust the start of the DATA segment downward to accommodate RAM for flash self-programming less than 12 KB.

1.9.7 Files for Flash Self-Programming Routines

The following files were written for flash self-programming functions.

1.9.7.1 Fsp.h

The header file fsp.h contains definitions of program areas and other constants used in flash self-programming, and declarations for the flash self-programming functions.

The following symbolic variables have been defined for user program area and boot program area.

```
#define FSP_PROG_SADDR  0x00008000  /* entry address for program */
#define FSP_PROG_SIZE   0x8000           /* 32K size for loaded user program
*/

#define FSP_BOOT_SADDR  0x00002000  /* start address for boot swap area */
#define FSP_BOOT_SIZE   0x2000           /* size of boot */
```

The boot program area is from 00000000H to 00001FFFH; the alternate boot program area is from 00002000H to 00003FFFH. The two boot areas of 8 KB each, consisting of four 2 KB blocks of flash memory, are specific to the µPD70F3318Y device used.

In the demonstration program, the user program area is set to start at 00008000H, and take up 32 KB of memory. For the device used, this could be increased to start at 00004000H, after the alternate boot program area, and ending at 0003FFFFH, at the top of 256 KB of flash memory.

1.9.7.2 Fsp.c

The source file fsp.c contains the following functions for flash self-programming:

MD_STATUS FSP_Init (void)

This routine sets up hardware and software for flash self-programming, including calling the SelfLib_Init() routine from the flash self-programming library.

MD_STATUS FSP_LoadProgram (unsigned long saddr, unsigned long eaddr, unsigned long offset)

This routine loads an Intel hex format object file into memory in the area starting at **saddr** and ending at one below **eaddr**. The parameter **offset** is added to the address specified in the hex data, to allow a program normally addressed at zero to be loaded at a different address, used in loading a boot program to the alternate boot area.

MD_STATUS FSP_RunProgram(void)

This routine runs the user program, by calling the location at the start of the user program area (00008000H).

void FSP_DemoIsr(void)

This routine sets a variable so that the timer interrupt will alter the LED, and then spends time in non-flash self-programming mode, and in flash self-programming mode. The timer interrupt will modify one or the other LED digit, to provide visible evidence of interrupts in one mode or the other.

void FSP_BootInfo(void)

This routine, called by FSP_BootSwap(), reports the current state of the boot swap bit, and Flash programming security flags.

MD_STATUS FSP_Bootswap(void)

This routine changes the value of the boot swap flag, so that on the next RESET, the boot program loaded into the alternate boot program area will run as the boot program.

1.9.8 Files from NEC Electronics Flash Self-Programming Library

The following files were included in the demonstration program from the NEC Electronics flash self-programming library.

File	Function
df3381.h	Definitions of μPD70F3318 I/O register locations
Nec_types.h	Definitions of data types
SelfLib.h	Definitions of self-programming library functions and values
SelfLibSpecific.h	Device-specific definitions
Selfprog.h	Definitions of default self-programming function
Target.h	Target hardware specific definitions
SelfLibAsm.s	Basic self-programming functions in assembly language
SelfLibBrom.c	Low-level interface to assembly language functions
SelfLibCommands.c	User interface functions of the self-programming library
SelfLibDebug.c	Debugging functions (not compiled)
SelfLibEnvironment.h	Definitions for device-dependant flash programming items
SelfLibGlobal.h	Definitions for functions and global variables used by library
SelfLibInit.c	Initialization functions for self-programming library
userFuncInt.s	Code to be placed at start of internal RAM for interrupt handling
userFunc.c	User interrupt handling in flash self-programming mode

Please see the documentation for the NEC Electronics flash self-programming library for detailed information on files and functions.

For the demonstration program, the following files from the flash self-programming library were modified.

1.9.8.1 Target.h

The header file target.h defines values and macros used in accessing target hardware, and is specific to the processor, clock frequency, and method of activating FLMD0. As provided in the flash self-programming library, this is set for an NEC Electronics test environment.

Changes were made to this file to define the operating environment used, which was an M-V850ES/KJ1 board, with a 20 MHz main clock, and P01 used to activate FLMD0.

1.9.8.2 userFunc.c

This file, as supplied with the flash self-programming library, provides a simple handler for an interrupt in the NEC Electronics test environment.

This file has been modified to handle the INTTM000 and INTSR1 interrupts in flash self-programming mode in the demonstration program. The following functions have been added or altered.

__interrupt void userIntHdr (void)

This routine is the destination of the branch placed at the start of internal RAM in the *userFuncInt.s* file. This routine saves and restores general purpose registers (by definition as an __interrupt routine), and calls the *userFunc()* routine to process interrupts.

Void userFunc (void)

This routine checks the value of the ECR register to determine the source of the interrupt. For INTTM000 interrupts, this routine calls ram_inttm000(); for INTSR1 interrupts, the routine calls ram_intsr1(); all other interrupts are ignored. For the handled interrupts, the triggering interrupt flag is reset. The **ram_int** variable is set to one, to provide a means for the interrupt service routines to know that the interrupt happened in flash self-programming mode.

void ram_inttm00(void)

This routine is originally located in flash memory, and is copied to internal RAM by FSP_Init(). The copy in internal RAM is called by userFunc() when an INTTM000 interrupt happens during flash self-programming mode. The original code in flash memory is called by MD_INTTM000() when an INTTM000 interrupt happens in non-flash self-programming mode.

The routine counts down the **millisecond** variable used in timing routines. If the **ledSpin** variable is set, it will modified one or the other of the LEDs, depending on the state of the **ram_int** variable.

void ram_intsr1(void)

This routine is originally located in flash memory, and is copied to internal RAM by FSP_Init(). The copy in internal RAM is called by userFunc() when an INTSR1 interrupt happens during flash self-programming mode. The original code in flash memory is called by MD_INTSR1() when an INTSR1 interrupt happens in non-flash self-programming mode.

The routine reads the received character, and calls ram_intsr1_push() to store it in the receive buffer. If the buffer is approaching the full state, an XOFF character is transmitted to pause the sending program.

void ram_intsr1_push(char c)

This routine stores the received character in the **u1_rxbuf[]** buffer, and updates the count of characters received and the pointer to the location to store the next character, wrapping around to the start of the buffer if necessary. If the buffer is full, the character is dropped.

1.9.9 Other Demonstration Program Files Not Generated by Applilet

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

Table 1. Other Demonstration Programs

File	Function
Ihexin.h	Intel hexadecimal input definitions
Ihexin.c	Intel hexadecimal input functions
Led_vkj1.h	Header file for seven-segment LED patterns and functions
Led_vkj1.c	Code to display data in seven-segment LEDs

1.10 Demonstration Platform

A demonstration platform was chosen from the NEC development tools available at the time when this document was prepared. In some cases users may be able to duplicate the same hardware by using standard off-the-shelf components along with the NEC Electronics microcontroller of interest.

1.10.1 Resources

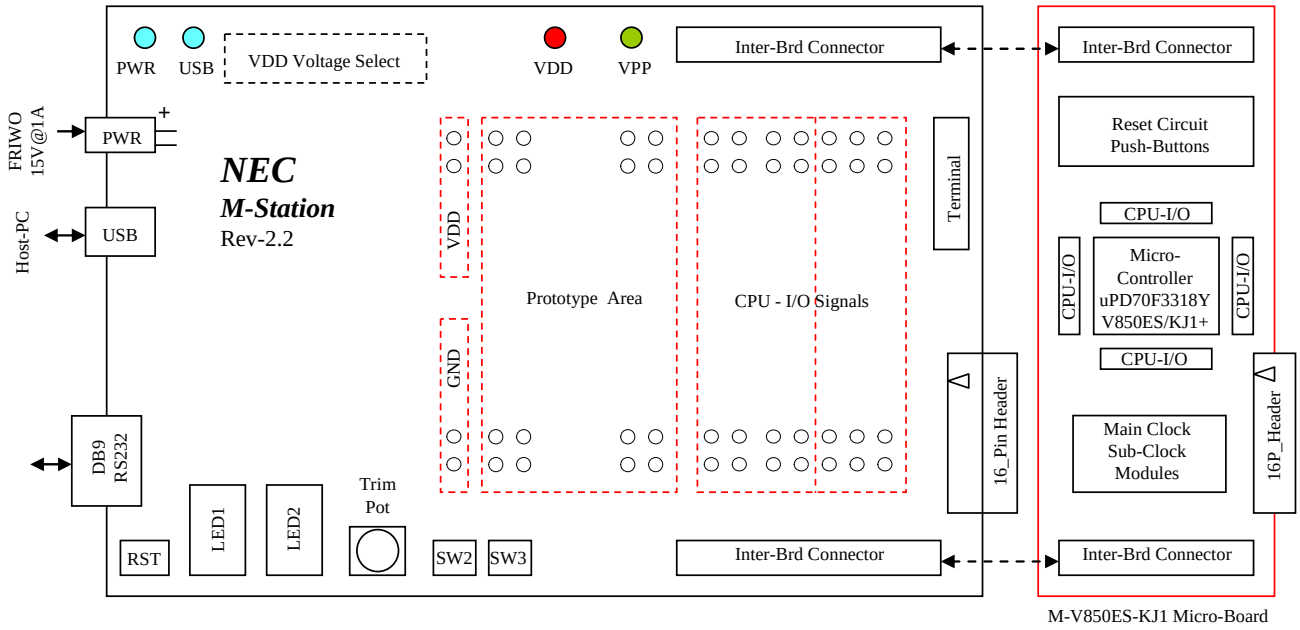
To demonstrate the program, the following resources have been used:

- M-V850ES-KJ1 micro-board, with µPD70F3318Y (V850ES/KJ1+) 32-bit microcontroller mounted
- µPD70F3318Y resources used:
- 256 KB flash memory
- 16 KB internal RAM
- Port P01 for controlling the FLMD0 input
- UART1 peripheral, using P90/TXD1 and P91/RXD1
- Ports PDH and PDLH for controlling seven-segment LEDs
- M-Station II Evaluation System, using M-Station II resources:
  - Seven-segment LEDs LED1 and LED2
  - RS-232 transceiver connected to UART1 pins
  - SW1 RESET switch to provide hardware Reset

— SW2 and SW3 input switches (used by user application programs)

For details on the hardware listed above, please refer to the appropriate user's manual, available online at **www.am.necel.com.**

Figure 26. Demonstration Hardware Blocks



1.10.2 Demonstration of Program

Make sure the hardware has been configured correctly, the µPD70F3318Y microcontroller has been programmed with the demonstration program boot program code (BOOTA_romp.hex version), and the M-Station II RS232 transceiver is connected to a computer running a terminal emulation program.

The terminal emulation program should be set to operate at 38400 baud, and for XOFF/XON flow control. For the terminal emulation program used in developing this document, HyperTerminal PE V6.3, it was also necessary to set a delay of 1 millisecond after each character transmitted, to allow proper response to XOFF characters received. If this delay was not set, the HyperTerminal program would transmit errors or drop a character when XOFF was received.

Upon reset of the CPU, the display will show "A1" (for boot program A, version 1). The sign-on message and prompt will be sent to UART1 and displayed on the computer in the terminal emulation window:

```
BootVer A1.0
L,R,B,S,I or D --
```

1. Enter “D” to run a demonstration of interrupts in flash self-programming mode. The following message will be shown on the computer:

   ```
   L,R,B,S,I or D – Demo ISR
   ```

   The two LEDs will show “- -“ initially; then the right LED will have one lit segment rotate clockwise around the display for five seconds, showing timer interrupts occurring in non-flash self-programming mode.

   The program will then enter flash self-programming mode for about four seconds. During this time, the left LED will have one lit segment rotate clockwise around the display, moving faster than in the non-flash self-programming mode. **Note:** during this time, the program briefly exits and reenters flash self-programming mode. It is possible that the right LED may rotate one segment during this time, if the timer interrupt occurs during the brief time period between flash self-programming operations.

   After this, the program will return to non-flash self-programming mode for five seconds; the right LED will rotate again at the slower rate. When the demonstration is finished, the display will show the boot program letter and version, and the prompt for command will be displayed in the terminal emulation window.

2. Enter “L” to load a user program to the user program area. The LED will show “L –“, and the terminal emulation window will display:

   ```
   L,R,B,S,I or D -- Load
   Start = 00008000, End = 0000FFFF
   Blank Check...Blank
   Send file...
   ```

   This shows the user program area boundaries, and that the user program area is blank, with no erase needed.

   Using the terminal emulation program, select “Send Text File”, and send the file UA_1_8000_romp.hex to program the first user program, UA_1. The boot program will echo the lines as received, and report the areas programmed:

   ```
   Send file...:04000003081F0000D2
   :020000020000FC
   :040000008007F08104
   loading....
   Wn: addr 00000000 < start - ignored
   :020000020007F5
   :0A000000FFFFFFFFFFFFFFFF00
   Wn: addr 00000070 < start – ignored
      :
   ```

   The above display shows the initial lines specifying data outside of the user program area, and a warning that this data is ignored. This is ok; for the user program developed, these records specify the Reset vector and option bytes, which would be used if the user program were loaded into memory directly, but are not used in the loaded user program.

```
   :
:020000020800F4
:040000008007F00184
Addr=00008000, Nbytes=0004, Waddr=00008000
:020000020801F3
:10000000400148015001580160011801200128 01F8
Addr=00008010, Nbytes=0010, Waddr=00008010
   :
```

The above display shows the start of the user program loaded starting at address 00008000H; first four bytes containing a branch to the entry point of the user program, and then additional program code.

During loading and programming of data, characters will continue to be received by the boot program through INTSR1 interrupts. If the interrupt is received while in flash self-programming mode, the right LED will rotate one lit segment counterclockwise for each character received.

The echoing of lines, listing of addresses, and spinning of the right LED should continue until the loading of the program is completed.

```
   :
:10003C0000000000C0F9A4B0999282F880988883DF
Addr=000086DC, Nbytes=0010, Waddr=000086DC
:04004C00C6A1868E35
Addr=000086EC, Nbytes=0004, Waddr=000086EC
:00000001FF
Load OK
Verify OK

L,R,B,S,I or D --
```

This shows data at the end of the file; with a few records loading data, and then the end-of-file record. The “Load OK” message confirms correct programming of the data. The “Verify OK” message confirms the proper result of an internal verify operation on the user program area.

At this point, the user program has been loaded and is ready to run. The boot program returns to the main() routine and displays the prompt.

3. Enter “R” to run the user program which has been loaded. The terminal emulation window will show:

```
L,R,B,S,I or D -- Run...
```

If the UA_1 program has been loaded, the LED will show “ 0”. Pressing the SW2 and SW3 switches on the M-Station will cause the display to count up by one to 15 (SW3), or down by two from 15 (SW2), with numbers wrapping around at zero or 15.

This shows the operating of the loaded user program. At this point, the boot program code is still in memory, but is not being used.

4. Press the SW1 RESET switch on the M-Station, to reset the processor. The LED will show “A1” again and the sign-on message and prompt will be shown in the terminal emulation window. Enter “L” at the prompt, select “Send Text File” on the terminal emulation program, and select the UA_2_8000_romp.hex file to load the second user program UA_2.

   This time, the user program contains data, and the boot program erases the user program area before beginning to load the file.

```
L,R,B,S,I or D -- Load
Start = 00008000, End = 0000FFFF
Blank Check...Erasing...Erased
Send file...:04000003081F0000D2
:020000020000FC
:040000008007F08104
loading....
   :
   :
:00000001FF
Load OK
Verify OK
```

   This shows the second user program loaded correctly. The program will return to main() and show the prompt.

5. Enter “R” to run the second user program. The terminal emulation window will show

```
L,R,B,S or D -- Run...
```

   If the UA_2 program has been loaded, the LED will show “00”. Pressing the SW2 and SW3 switches on the M-Station will cause the display to count up by twos in hexadecimal from zero to 36 (display shows “24”) (SW3), or down by threes from 36 to zero (SW2), with numbers wrapping around at zero or 36.

6. Press the SW1 RESET switch on the M-Station, to reset the processor. The LED will show “A1” again and the sign-on message and prompt will be shown in the terminal emulation window. Enter “B” at the prompt, to give the BootLoad command. The LED will show “7 –“ (with the seven representing an upside-down “L”), and the terminal emulation window will show:

```
L,R,B,S or D -- BLoad
Start = 00002000, End = 00003FFF
Blank Check...Blank
Send file...
```

   This shows the alternate boot program area from 00002000H to 00003FFFH as the target of the load, and that the alternate boot program area is blank. Using “Send Text File” on the terminal emulation program, send the alternate boot program, BOOTB_romp.hex.

```
Send file...:0400000300420000B7
:020000020000FC
:040000008007200451
loading....
Addr=00000000, Nbytes=0004, Waddr=00002000
```

```
:020000020001FB
:04000000E0074001D4
Addr=00000010, Nbytes=0004, Waddr=00002010
   :
   :
:0200000201FFFC
:04000C0078563412DC
Addr=00001FFC, Nbytes=0004, Waddr=00003FFC
:00000001FF
Load OK
Verify OK
```

The program is loaded into the alternate boot program area. At the end of loading, the alternate boot program area is verified, and the program returns to main() and displays the prompt.

7. For a demonstration of the boot-swapping operation, use the following steps once the BOOTB program has been loaded to the alternate boot program area.

    a. Enter the “I” command. The FSP_BootInfo() routine reads and shows the current state of the boot swap bit and the security flags. Assuming the device has been chip erased prior to programming, the boot swap bit will be zero, and the security flags will be all ones.

    ```
    L,R,B,S,I or D -- BootInfo
    BootSwap=00000000
    SecFlags=FFFFFFFF
    ```

    b. Enter the “S” command to swap the current boot area with the alternate boot area.

    ```
    L,R,B,S,I or D -- Swap Boot
    BootSwap=00000000
    SecFlags=FFFFFFFF
    Set secflag=03FFFFFE
    00000000
    BootSwap=00000000
    SecFlags=03FFFFFF

    BootVer A1.0
    L,R,B,S,I or D --
    ```

    The FSP_BootSwap() routine first calls FSP_BootInfo() to show the current state of the boot swap bit and security flags. It then modifies the security flag value, setting the upper eight bits to 0x03, and setting bit zero to zero, the same value as the boot swap bit, in order to swap on the next RESET. The return value from setting the security flags, 00000000, indicates proper operation.

    The FSP_BootInfo() routine is called again; note that the current state of the boot swap bit has not changed, and that the upper bits of the security flag have changed, but bit zero always shows a one. The program jumps to the Reset vector, and restarts the program without doing a full chip RESET. The sign-on

message indicates that BOOTA is still the current boot program (boot is not swapped).

c. Enter the “I” command to check the boot swap information.

```
L,R,B,S,I or D -- BootInfo
BootSwap=00000000
SecFlags=03FFFFFF
```

The information shows the boot is not yet swapped.

d. Press the SW1 RESET switch on the M-Station, and release it. This performs a hardware RESET of the device. At this point, the boot swap takes effect. Blocks 0-3 of flash memory, containing the BOOTA program, are swapped to location 00002000H, and blocks 4-7 of the flash memory containing the BOOTB program are swapped to location 00000000H. Upon reset, the BOOTB program runs. BOOTB shows a “b1” in the LED, prints its sign-on message, and shows the prompt.

```
L,R,B,S,I or D --
BootVer B1.0
L,R,B,S,I or D –
```

e. Enter the “I” command to get boot swap information.

```
L,R,B,S,I or D -- BootInfo
BootSwap=00000001
SecFlags=03FFFFFF
```

This shows that the current state of the boot swap bit is now 1, indicating the boot areas have been swapped. Note that the security flags still show bit zero as a 1.

f. Enter “S” to swap back the boot areas.

```
L,R,B,S,I or D -- Swap Boot
BootSwap=00000001
SecFlags=03FFFFFF
Set secflag=03FFFFFF
00000000
BootSwap=00000001
SecFlags=03FFFFFF

BootVer B1.0
```

The FSP_BootSwap routine shows the current boot swap bit as 1, and the value of the security flags. To swap the boot areas back, bit zero of the security flag value is set to 1, following the current value of the boot swap bit, and the security flag value is written.

After the security flags are written, the boot swap bit is still 1, since the swap has not yet taken place. The routine jumps to the Reset vector, and the sign-on message still shows the BOOTB program.

g. Enter the “I” command to see the boot information:

```
L,R,B,S,I or D -- BootInfo
BootSwap=00000001
SecFlags=03FFFFFF
```

The boot swap bit still is 1, reflecting swapped boot areas, with BOOTB active.

h. Press the SW1 RESET switch on the M-Station, and release it. This performs a hardware RESET of the device. At this point, the boot swap takes effect. Blocks 4-7 of flash memory, currently located at 00000000H and containing the BOOTB program, are swapped back to location 00002000H; blocks 0-3 of flash memory, currently located at 00002000H and containing the BOOTA program, are swapped back to location 00000000H. Upon reset, the BOOTA program runs. BOOTA shows an “A1” in the LED, prints its sign-on message, and shows the prompt.

```
L,R,B,S,I or D --
BootVer A1.0
L,R,B,S,I or D --
```

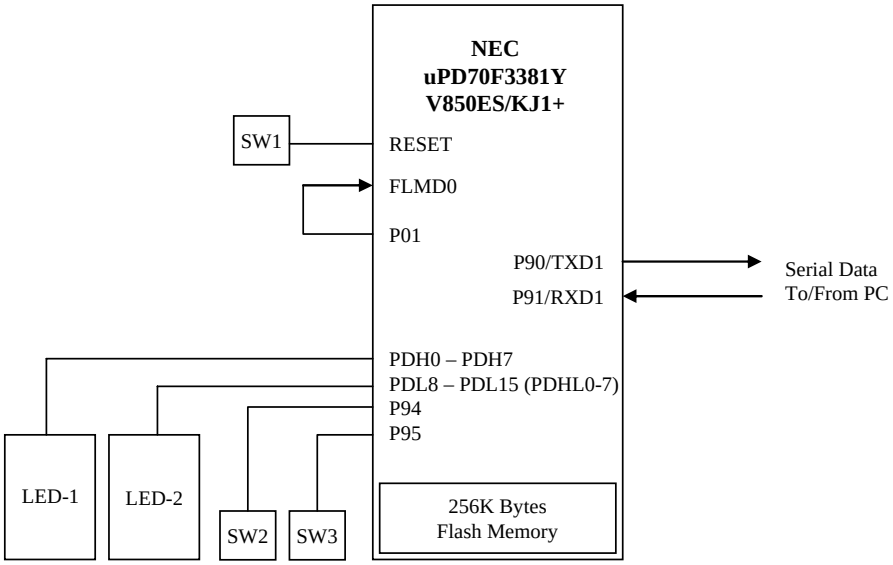
i. Enter the “I” command to see the boot information:

```
L,R,B,S,I or D -- BootInfo
BootSwap=00000000
SecFlags=03FFFFFF
```

The boot swap bit is now zero again, showing that the boot program area and alternate boot program area are not swapped.

1.11 Hardware

Figure 27. Hardware Block Diagram



When the M-V850ES-KJ1 micro-board is mounted on the M-Station II, the default configuration of Please consult the user's manual for the M-V850ES-KJ1 and M-Station II for further details on signal interconnection.

Note: on the M-V850ES-KJ1, it is necessary to insert jumper JP2, to connect the P01 pin to the FLMD0 input of the processor, to allow the processor to drive FLMD0 high and activate flash self-programming mode.

1.12 Software Modules

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

The listings for these files are located in the next section.

1.12.1 BOOT Demonstration Program Except Flash Self-Programming Library Files

File	Purpose	Generated By Applilet	Modified By User
Main.c	Main program	Yes	Yes
Macrodriver.h	General definitions used by Applilet	Yes	No
Crte.s	Reset vector, program startup code	Yes	Yes[Note 1]

File	Purpose	Generated By Applilet	Modified By User
Inttab.s	Interrupt vectors	Yes	Yes[Note 1]
System.inc	Clock-related definitions	Yes	No
System.s	Clock_Init() function	Yes	Yes[Note 2]
System_user.c	File for System interrupt	Yes	No
Systeminit.c	SystemInit() and hdwinit() functions	Yes	No
Serial.h	UART1 and CSI00-related definitions	Yes	Yes[Note 3]
Serial.c	UART1 and CSI00-related functions	Yes	Yes[Note 3]
Serial_user.c	User code in UART1, CSI00 callback routines	Yes	Yes[Note 3]
Timer.h	Timer-related definitions	Yes	Yes[Note 4]
Timer.c	Timer-related functions	Yes	No
Timer_user.c	User code for timer interrupt service	Yes	Yes[Note 4]
850_splib.dir	Link directive file	Yes	Yes[Note 1]
Fsp.h	Flash self-programming definitions	No	--
Fsp.c	Flash self-programming functions	No	--
Ihexin.h	Intel hex input definitions	No	--
Ihexin.c	Intel hex input functions	No	--
Led_vkj1.h	LED definitions	No	--
Led_vkj1.c	LED functions	No	--

Notes:

1. Crte.s, inttab.s, and 850_splib.dir (modified from 850.dir) were modified to increase the amount of space available for the boot program, by eliminating unused interrupt vectors.

2. System.s was modified to correct an error in the Clock_init() routine which resulted in excessive time spent waiting for the PLL to stabilize.

3. Serial.h was modified to add the declarations of new UART1 values and functions, defined in Serial_user.c. Serial.c was modified to change the initialization of UART1, and to remove unused code. Serial_user.c had several variables and functions added, to support special interrupt-driven reception of data.

4. Timer_user.c was modified to add the code to handle the periodic INTTM000 interrupt in the MD_INTTM000() routine, and to add routines for millisecond timing. Timer.h was modified to add the declarations for the millisecond timing functions.

Please see the appropriate places in section 1.4 above, and the listings for notes on modifications.

1.12.2 Flash Self-Programming Library Files

The following table shows the files used from the NEC Electronics flash self-programming library, and which were modified for use in the demonstration program.

Table 2.

File	Purpose	Modified By User
df3381.h	Definitions of µPD70F3318 I/O register locations	No
Nec_types.h	Definitions of data types	No
SelfLib.h	Definitions of self-programming library functions and values	No
SelfLibSpecific.h	Device-specific definitions	No
Selfprog.h	Definitions of default self-programming function	No
Target.h	Target hardware specific definitions	Yes[Note 1]
SelfLibAsm.s	Basic self-programming functions in assembly language	No
SelfLibBrom.c	Low-level interface to assembly language functions	No
SelfLibCommands.c	User interface functions of the self-programming library	No
SelfLibDebug.c	Debug functions (not compiled)	No
SelfLibEnvironment.h	Definitions for device-dependant flash programming items	No
SelfLibGlobal.h	Definitions for functions and global variables used by library	No
SelfLibInit.c	Initialization functions for self-programming library	No
userFuncInt.s	Code placed at start of internal RAM for interrupt handling	No
userFunc.c	User interrupt handling in flash self-programming mode	Yes[Note 2]

Notes:

1. Target.h was modified to specify the output line used for FLMD0 control, and to comment out unused hardware initialization or control functions.

2. userFunc.c was modified to provide handling for the INTTM000 and INTSR1 interrupts during flash self-programming mode. The routines ram_inttm000() and ram_intsr1() were copied to internal RAM for this purpose. The code for these routines, located in the original flash memory locations, was used for interrupt handling when not in flash self-programming mode.

1.12.3 User Application Program UA_1 and UA_2 Files

The following table shows the files used for the user application programs, UA_1 and UA_2.

These programs were written from scratch rather than by modification of template files, except for the crte_8000.s startup file and the UA_8000.dir link directive file, which were modified from the standard crte.s and 850.dir template files, supplied with the NEC Electronics C compiler for V850ES devices.

Table 3. User Application Program UA_1

File	Description
Crte_8000.s	Reset vector, entry vector, startup code
Ua_1.c	Main() routine, hdwinit() routine
Sw_vkj1.h	Pushbutton switch definitions
Sw_vkj1.c	Pushbutton switch functions

File	Description
Led_vkj1.h	LED definitions
Led_vkj1.c	LED functions
UA_8000.dir	Link directive file

Table 4. User Application Program UA_2

File	Purpose
Crte_8000.s	Reset vector, Entry vector, startup code
Ua_2.c	Main() routine, hdwinit() routine
Sw_vkj1.h	Pushbutton switch definitions
Sw_vkj1.c	Pushbutton switch functions
Led_vkj1.h	LED definitions
Led_vkj1.c	LED functions
UA_8000.dir	Link directive file

Except for the ua_1.c and ua_2.c files, other files used for these two programs were identical.

The led_vkj1.c file as used by the user application programs is slightly different than that used by the boot program. In the boot program, unused routines were commented out for space saving, and display data table constants were placed in a special SCONST segment.

1.12.4 Notes on Program Build: Compile and Hex Output Constraints

The following constraints need to be noted on the program build process for the boot program and user application program.

1. For the boot program, the code for the flash self-programming library which is copied to RAM for execution must execute completely in RAM, with no access to ROM functions. This means that this code cannot access standard library functions, or the standard prologue and epilogue functions normally used by C functions

2. To have no calls to prologue and epilogue functions, the compiler options “-Ot” (optimize for speed) and “-Xpro_epi_runtime=off” (inline prologue and epilogue code) must be set for all of the files in the flash self-programming library.

3. In the PM+ build environment for the NEC Electronics C compiler, this is done by selecting the “Special compiler options” for these source files.

4. On the **General** tab, select Optimization Level setting “Level 2 Advanced Opt. (Exec. Speed)(-Ot)”.

5. On the “Output Code” tab, select Prologue Epilogue Runtime setting “Not Use(-Xpro_epi_runtime=off)”.

6. For both the boot program and the user application program, to have the Intel hexadecimal output file produce records that are compatible with the input process in the FSP_LoadProgram() function, settings need to be made for hexadecimal output.

7. In the PM+ build environment, select **Tools → Hexa Converter Options…**.

8. On the **Option** tab, in the **Format** box, select “Intel Extended” to produce Intel hexadecimal-formatted records. In the **Maximum Length of Block/Record** box, enter “0x10” to create records that have at most 16 bytes per line.

9. To properly assemble hex input line bytes into words, the input line should be constructed of a number of bytes divisible by four. The default length (31 bytes) will cause errors in the FSP_LoadProgram() function, because word data will break across two hex records.

2. DEVELOPMENT TOOLS

The following software and hardware tools were used in the development of this application note.

Figure 28. Software Tools

Tool	Version	Description
Applilet	E1.46c	Source code generation tool for NEC Electronics devices
V850ESKX1H.mcu	V1.33	Applilet MCU configuration for µPD70F3318Y (V850ES/KJ1+)
PM Plus	V6.10	Project Manager for program compilation and linking
CA850	V3.00	C compiler/assembler/linker for NEC Electronics V850ES devices
DF3318Y.800	V1.01	Device file for µPD70F3318Y (V850ES/KJ1+) device
HyperTerminal emulation program	V6.3	Terminal emulation program for the Windows operating system

Figure 29. Hardware Tools

Tool	Version	Description
M-Station 2	V2.1E	Base platform for micro-board demonstration
M-V850ES-KJ1	V1.0	Micro-board for V850ES/KJ1+ CPU chip is µPD70F3318YGJ

Figure 30. Flash Self-Programming Libraries

Tool	Version	Description
U16929EE3V1AN00	March 2006	Application note, manual for flash self-programming libraries
NEC_SingleVoltage_SelfLib_V110.exe	V1.10	Executable program containing flash self-programming libraries
KX1+(3318) Library	V1.04	Flash self-programming library for V850ES/KJ1+ device

3. SOFTWARE LISTINGS

3.1 BOOTA and BOOTB Loader Program

The BOOTA loader program and the BOOTB program use the same source files. In main.c, a #define directive was added:

```
#define BOOTA  1
```

With this defined as shown above, compiling and linking the program will produce the BOOTA version. Changing this definition to:

```
#define BOOTA  0
```

and compiling and linking the program will produce the BOOTB version. The only difference in the two versions is the version display in the LED and the sign-on message.

3.1.1 Main.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 : 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 "timer.h"
#include "serial.h"
```

```
/* include files for M-Station I/O */
#include "led_vkj1.h"   /* functions for M-Station LED output */

/* include files for this application */
#include "fsp.h"        /* include Flash Self-Programming */

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

/* define BOOTA = 1 for BOOTA version, BOOTA = 0 for BOOTB version */
#define BOOTA     1

#if (BOOTA == 1)
#define BOOT_LETTER           'A'   /* letter identifying boot A or B */
#else
#define BOOT_LETTER           'B'
#endif

#define BOOT_VER_MAJOR  1     /* major version */
#define BOOT_VER_MINOR  0     /* minor version */

/* constant strings - allocate in special section, for space reduction */
#pragma section sconst "S_SCONST0" begin
#if (BOOTA == 1)
const char s_BootVer[]        = "\r\nBootVer A1.0";
#else
const char s_BootVer[]        = "\r\nBootVer B1.0";
#endif
const char s_Prompt[]   = "\r\nL,R,B,S,I or D -- ";
const char s_Load[]           = "Load\r\n";
const char s_BootLoad[] = "BLoad\r\n";
const char s_QQ[]       = "??\r\n";
#pragma section sconst "S_SCONST0" end

/* rest of strings go in standard SCONST section */
const char s_BootInfo[] = "BootInfo\r\n";

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**          main function
**
**  Parameters:
**          None
**
**  Returns:
**          None
**
**-------------------------------------------------------------------------
*/
```

```
#if 0
/* buffer for strings in echo line debugging test */
char buf[128];
#endif

void  main( void )
{
MD_STATUS status;
unsigned char sw_val;
BOOL sw2first = MD_FALSE;
char inchar;
int i;

      led_init();             /* initialize LED */
      led_dig_left(BOOT_LETTER - 'A' + 0xA);          /* display letter and
major version number */
      led_dig_right(BOOT_VER_MAJOR);

      UART1_Rx_Clear(); /* clear any UART1 input */
      TM00_Start();           /* start millisecond timer */

      FSP_Init();             /* initialize Flash Self-Programming */

      UART1_TxStr((char *)s_BootVer);           /* display version and prompt
on UART */
      UART1_TxStr((char *)s_Prompt);

      while (1) {
            if (UART1_Rx_Chk() == MD_TRUE) {
                  /* got a character, act on it */
                  inchar = UART1_Rx();
                  if ((inchar >= 'a') && (inchar <= 'z')) {
                        /* make uppercase */
                        inchar = inchar - ('a' - 'A');
                  }
                  switch (inchar) {
                  case 'L':
                        UART1_TxStr((char *)s_Load);
                        led_out_left(0xC7);           /* show 'L' */
                        led_out_right(0xBF);    /* turn on segment G */
                        FSP_LoadProgram(FSP_PROG_SADDR, FSP_PROG_EADDR, 0);
                        break;

                  case 'R':
                        FSP_RunProgram();
                        break;

                  case 'B':
                        UART1_TxStr((char *)s_BootLoad);
                        led_out_left(0xF8);           /* show '7' (upside
down L) */
                        led_out_right(0xBF);    /* turn on segment G */
                        FSP_LoadProgram(FSP_BOOT_SADDR, FSP_BOOT_EADDR,
FSP_BOOT_SADDR);
```

```
                        break;

                case 'S':
                        FSP_Bootswap();
                        break;

                case 'D':
                        led_out_left(0xFE);           /* turn on segment A in
both */
                        led_out_right(0xFE);
                        FSP_DemoIsr();
                        break;

#if 1 /* for debugging */
                case 'I':
                        UART1_TxStr((char *)s_BootInfo);
                        FSP_BootInfo();
                        break;
#endif

#if 0 /* for debugging */
                case 'E':
                        UART1_TxStr("Echo lines received\r\n");
                        while (1) {
                                UART1_RxFgets(buf, sizeof(buf) - 1, 0);
                                SetMsecTimer(1000);
                                while (!CheckMsecTimer())
                                        ;
                                UART1_TxStr(buf);
                                UART1_TxCRLF();
                        }
                        break;
                case 'e':
                        UART1_TxStr("Echo characters received\r\n");
                        while (1) {
                                if (UART1_Rx_Chk()) {
                                        inchar = UART1_Rx();
                                        if ((inchar >= 'a') && (inchar <= 'z'))
                                                inchar = inchar - ('a' - 'A');
                                        SetMsecTimer(200);
                                        while (!CheckMsecTimer())
                                                ;
                                        UART1_Tx(inchar);
                                }
                        }
                        break;
#endif

                default:
                        UART1_TxStr((char *)s_QQ);
                        break;
                }
                UART1_TxStr((char *)s_Prompt);
                led_dig_left(BOOT_LETTER - 'A' + 0xA);          /* letter
and major version number */
```

```
                led_dig_right(BOOT_VER_MAJOR);
            } /* end of if UART1_ReceiveDone true */

      } /* end of while (1) loop */

} /* end of main() */
```

3.1.2 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 */
#define LockInt( ) { __asm("stsr 5,r10"); __asm("push r10"); __asm("di"); }
#define UnlockInt( ) {  __asm("pop r10"); __asm("ldsr r10,5"); }

/*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
```

3.1.3 Crte.s

```
#  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 : crte.s
#  Abstract : start file for CA850
#  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
#
#                  |        :        |
#                  |        :        |
#         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     _SystemInit
      .extern     _main
      .extern     _Clock_Init

.if 0 -- remove argv and argc, not used by 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
.endif -- remove argv and argc

#-----------------------------------------------------------------------------
-
#     dummy data declaration for creating sbss section
#-----------------------------------------------------------------------------
-
      .sbss
```

```
      .lcomm      __sbss_dummy, 0, 0


#------------------------------------------------------------------------------
-
#     system stack
#------------------------------------------------------------------------------
-
# -- for FSP App Note, increase stack size for complex ISR
#     .set  STACKSIZE, 0x200
      .set  STACKSIZE, 0x400
      .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
#     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 warning

      mov    1, r11           -- on-chip debug mode
      set1  5, PMC0[r0]
      set1  5, P0[r0]
      st.b   r11, PRCMD[r0]
      st.b   r11, OCDM[r0]
```

```
      nop
      nop
      nop
      nop
      nop
      mov   0x1, r11
      st.b r11, VSWC[r0]                  --mainclock over 16.6MHz

      jarl  _Clock_Init, lp         -- call Clock_Init function

      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)

      -- IRAM clean up --
      mov 0x3ffd800, r10      -- IRAM start address
      mov 0x3fff000, r11      -- IRAM end address
_clear_loop:                  -- IRAM clean up
      st.w r0, 0x0[r10]
      add 4, r10
      cmp r11,r10
      jnz _clear_loop

.if 0 -- remove argc and argv setting
      ld.w  $__argc, r6       -- set argc
      movea $__argv, gp, r7         -- set argv
.endif
      jarl  _SystemInit, lp         -- call SystemInit function
      jarl  _main,lp          -- call main function
__exit:
      halt                    -- end of program

.if 1 -- add section for marker at end of program
      .section "S_ROM_END", const
```

```
      .word 0x12345678  -- special marker
.endif
```

3.1.4 Inttab.s

```
--/*
--
*****************************************************************************
--**
--**  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 : inttab.s
--**  Abstract : This file implements interrupt vector table
--**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
--**
--
*****************************************************************************
--*/

--INT vector

------------------------------------------------------------------------
--    variable initiate
------------------------------------------------------------------------

      --.section "RESET", text
      --jr      __start

      .section "NMI", text               --nmi pin input
      reti

      .section "INTWDT1", text           --WDT1 OVF nonmaskable
      reti

      .section "INTWDT2", text           --WDT2 OVF nonmaskable
      reti

      .section "TRAP00", text            --TRAP instruction
      .globl      __trap00
__trap00:
      reti

      .section "TRAP10", text            --TRAP instruction
      .globl      __trap01
__trap01:
```

```
      reti

      .section "ILGOP", text              --illegal op code
      .globl      __ilgop
__ilgop:
      reti

.if 0 -- eliminate unused interrupts between ILGOP (0x0060) and last
interrupt
      -- program will use unused interrupt area for storing data, to save
boot space
      -- if debugger used, debugger uses RESET (0000), NMI (0010), and ILGOP
(0060)
      -- if debugger used, also uses INTSR0 (0190); see bottom of file for
this vector

      -- note that Applilet did not generate dummy vectors for following
interrupts,
      -- expected to be provided by "#pragma interrupt <int name>
<function>":
      -- INTTM000 (0100)
      -- INTSR1   (01C0)
      -- INTST1   (01D0)

      .section "INTWDTM1", text           --WDT1OVF maskable
      reti

      .section "INTP0", text              --INTP0 pin
      reti

      .section "INTP1", text              --INTP1 pin
      reti

      .section "INTP2", text              --INTP2 pin
      reti

      .section "INTP3", text              --INTP3 pin
      reti

      .section "INTP4", text              --INTP4 pin
      reti

      .section "INTP5", text              --INTP5 pin
      reti

      .section "INTP6", text              --INTP6 pin
      reti

      .section "INTTM001", text           --TM00 and CR001 match
      reti

      .section "INTTM010", text           --TM01 and CR010 match
      reti

      .section "INTTM011", text           --TM01 and CR011 match
```

```
        reti

        .section "INTTM50", text            --TM50 and CR50 match
        reti

        .section "INTTM51", text            --TM51 and CR51 match
        reti

        .section "INTCSI00", text           --CSI00 transfer complete
        reti

        .section "INTCSI01", text           --CSI01 transfer complete
        reti

        .section "INTSRE0", text            --UART0 reception error occurence
        reti

        .section "INTSR0", text             --UART0 reception completion
        reti

        .section "INTST0", text             --UART0 translation completion
        reti

        .section "INTSRE1", text            --UART1 reception error occurence
        reti

        .section "INTTMH0", text            --TMH0 and CMP00/CMP01 match
        reti

        .section "INTTMH1", text            --TMH1 and CMP10/CMP11 match
        reti

        .section "INTCSIA0", text           --CSIA0 transfer completion
        reti

        .section "INTIIC0", text            --IIC0 transfer completion
        reti

        .section "INTAD", text              --AD conversion end
        reti

        .section "INTKR", text              --key return interrupt
        reti

        .section "INTWTI", text             --watchtimer interval
        reti

        .section "INTWT", text              --watchtimer referemce time
        reti

        .section "INTBRG", text             --watchtimer counter BRG and PRSCM
match
        reti

        .section "INTTM020", text           --TM02 and CR020 match
```

```
      reti

      .section "INTTM021", text           --TM02 and CR021 match
      reti

      .section "INTTM030", text           --TM03 and CR030 match
      reti

      .section "INTTM031", text           --TM03 and CR031 match
      reti

      .section "INTCSIA1", text           --CSIA1 transfer completion
      reti

      .section "INTTM040", text           --TM04 and CR040 match
      reti

      .section "INTTM041", text           --TM04 and CR041 match
      reti

      .section "INTTM050", text           --TM05 and CR050 match
      reti

      .section "INTTM051", text           --TM05 and CR051 match
      reti

      .section "INTCSI02", text           --CSI02 transfer completion
      reti

      .section "INTSRE2", text            --UART2 reception error occurence
      reti

      .section "INTSR2", text             --UART2 reception completion
      reti

      .section "INTST2", text             --UART2 translation completion
      reti

      .section "INTIIC1", text            --IIC1 transfer completion
      reti
.endif      -- eliminated vectors

.if 1 -- add specific vectors here, rather than using pragma

      -- special section for INTTM000 (0100)
      .section "S_INTTM000", text
      jr _MD_INTTM000

      -- special section for INTSR0 (0190), to reserve spot (debugger uses)
      .section "S_INTSR0", text
      reti

      -- special section for INTSR1 (01C0)
      .section "S_INTSR1", text
      jr _MD_INTSR1
```

```
.endif

-- end of file
```

3.1.5 System.inc

```
--/*
--
*******************************************************************************
--**
--**  This device driver was created by Applilet for the V850ES/FE2,
V850ES/FF2,V850ES/FG2
--** and V850ES/FJ2 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 : system.inc
--**  Abstract : This file implements a device driver for the SYSTEM module
--**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
--**
--  Device:  uPD70F3318Y
--
--  Compiler:  NEC/CA850
--
--
*******************************************************************************
--*/
.set  CG_Mainosc, 0x5
.set  CG_SECURITY0,     0xff
.set  CG_SECURITY1,     0xff
.set  CG_SECURITY2,     0xff
.set  CG_SECURITY3,     0xff
.set  CG_SECURITY4,     0xff
.set  CG_SECURITY5,     0xff
.set  CG_SECURITY6,     0xff
.set  CG_SECURITY7,     0xff
.set  CG_SECURITY8,     0xff
.set  CG_SECURITY9,     0xff
```

3.1.6 System.s

```
--/*
--
*******************************************************************************
--
--  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 : system.s
--  Abstract : This file implements a device driver for the SYSTEM module
--  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
--
--
--  Compiler:  NEC/CA850
--
--
*****************************************************************************
--*/
      .include "system.inc"
      .section "OPTION_BYTES", text
      .byte  0                      --Set to option byte (Ring-OSC cannot be
stopped)
      .byte  0
      .byte  0
      .byte  0
      .byte  0
      .byte  0

      .section "SECURITY_ID", text
      .byte CG_SECURITY0            -- Security ID head
      .byte CG_SECURITY1
      .byte CG_SECURITY2
      .byte CG_SECURITY3
      .byte CG_SECURITY4
      .byte CG_SECURITY5
      .byte CG_SECURITY6
      .byte CG_SECURITY7
      .byte CG_SECURITY8
      .byte CG_SECURITY9            -- Security ID tail

      .text
      .globl      _Clock_Init
      .align      4
--/*
--**-------------------------------------------------------------------------
--
--**
--**  Abstract:
--**  Init the Clock Generator and Watchdog timer
--**
--**  Parameters:
--**  None
--**
--**  Returns:
```

```
--**  None
--**
--**-------------------------------------------------------------------------
--
--*/
_Clock_Init:
      add    -8, sp
      st.w  r11, 0[sp]
      st.w  r12, 4[sp]

      -- disable interrupt
      stsr  5, r11
      ori         0x80, r11, r11
      ldsr  r11, 5

      clr1  0, SYS[r0]              -- reset SYS register

      mov   r0, r11
      ld.b  PCC[r0], r12
      andi  0xf8, r12, r12
      or          r12, r11
      st.b  r11, PRCMD[r0]
      st.b  r11, PCC[r0]

      nop
      nop
      nop
      nop
      nop
      -- PLL start
      set1  0, PLLCTL[r0]
      -- PLL work
.if 1 -- fix bad code generated by Applilet
-- need to set r11 to some value before starting this loop!
      -- Lock 200 us
      movea 0x800, r0, r11
.endif
__CG_LOOP4:
      nop
      nop
      nop

      addi  -1, r11, r11
      cmp   r0, r11
      bnz   __CG_LOOP4
      set1  1, PLLCTL[r0]
      -- enable interrupt
      stsr  5, r11
      andi  0x7f, r11, r11
      ldsr  r11, 5
      -- pop
      ld.w  0[sp], r11
      ld.w  4[sp], r12
      add   8, sp
      --disable watchdog timer 2
```

```
      mov   0x1f, r11
      st.b  r11, WDTM2[r0]

      jmp [lp]
```

3.1.7 System_user.c

```
/*
***********************************************************************
**
**  This device driver was created by Applilet for the V850ES/FE2,
V850ES/FF2,V850ES/FG2
**  and V850ES/FJ2 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 : system_user.c
**  Abstract : This file implements a device driver for the SYSTEM interrupt
service routine
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
**
**  Compiler:  NEC/CA850
**
***********************************************************************
*/
/*
** ********************************************************************
** Include files
** ********************************************************************
*/

#include "macrodriver.h"
/*
** ********************************************************************
** MacroDefine
** ********************************************************************
*/
```

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 "timer.h"
#include "serial.h"
/*
** **********************************************************************
** MacroDefine
** **********************************************************************
*/
extern unsigned long _S_romp;

/*
**-----------------------------------------------------------------------
**
**  Abstract:
**    Init every Macro
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------
*/
void  SystemInit( void )
{

      _rcopy(&_S_romp, -1);

      __asm("di");                        /* disable interrupt */

      UART1_Init( );                      /* UART1 initiate */
```

```
      TM00_Init( );                         /* TM00 initiate */
      __asm("ei");                          /* enable interrupt */
}
```

3.1.8 Serial.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 : serial.h
**  Abstract : This file implements a device driver for the SERIAL module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*****************************************************************************
*/
#ifndef _MDSERIAL_
#define _MDSERIAL_

#define ADR_CSIA0B0           0xfffffe00  /* CSIA0 automatic transfer RAM
address */
#define ADR_CSIA1B0           0xfffffe20  /* CSIA1 automatic transfer RAM
address */

#define CSIA_AUTORAMSIZE            32    /* CSIA automatic transfer RAM size
*/
#define IIC_RECEIVEBUFSIZE          32

#define     UART_BAUDRATE_M1  0x1
#define     UART_BAUDRATE_K1  0x82

void UART1_Init( void );
#if 0 /* eliminate Applilet definitions and functions not used */
MD_STATUS UART1_SendData( UCHAR* txbuf, USHORT txnum );
MD_STATUS UART1_ReceiveData( UCHAR* rxbuf, USHORT rxnum );

enum TransferMode { Send, Receive };
#endif

/* added definitions and declarations for flash self-programming App Note */
/* definitions */
```

```
#define U1_RXBUF_SIZE   32          /* size of buffer */
#define U1_XOFF_MARK    (U1_RXBUF_SIZE - 20)    /* point where XOFF issued */
#define U1_XON_MARK           1           /* point to start sending again */
#define XOFF      0x13              /* XOFF or Ctrl-S */
#define XON       0x11              /* XON  or Ctrl-Q */

/* function declarations */
void UART1_Rx_Init(void);
void UART1_Rx_Isr(void);
BOOL UART1_Rx_Chk(void);
char UART1_Rx(void);
void UART1_Rx_Clear();
char * UART1_RxFgets(char *s, int n, int timeout);
void UART1_Tx(char c);              /* polled transmit output */
void UART1_TxStr(char * str); /* polled output string */
void UART1_TxNibble(unsigned char nib);
void UART1_TxByte(unsigned char byte);
void UART1_TxHalfword(unsigned short hword);
void UART1_TxWord(unsigned long word);
void UART1_TxCRLF(void);

#endif      /* _MDSERIAL_ */
```

3.1.9 Serial.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 : serial.c
**  Abstract : This file implements a device driver for the SERIAL module
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
*/

#include "macrodriver.h"
#include "serial.h"

#if 0 /* remove pragmas for interrupt vector generation - special sections in
inttab.s */
```

```
#pragma interrupt INTSR1 MD_INTSR1
#pragma interrupt INTST1 MD_INTST1
#endif
#if 0 /* remove unused variable definitions */
/* UART1 Transmission */
UCHAR *UART1_TX_ADDRESS;                /* uart1 transmit buffer address */
USHORT UART1_TX_CNT;                    /* uart1 transmit data number */

/* UART1 Reception */
UCHAR *UART1_RX_ADDRESS;                /* uart1 receive buffer address */
USHORT UART1_RX_CNT;                    /* uart1 recieve data number */
USHORT UART1_RX_LEN;
#endif

/*
**-------------------------------------------------------------------------
-------------------------------
**
**  Abstract:
**    Initialises UART1 interface for each channel; the application is
responsible for
**    set work mode, transfer speed, data bit length, stop bit length, parity
type and
**    interrupt pority setting.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
-------------------------------
*/
void UART1_Init( void )
{
      ASIM1 = 0;
      SetIORBit(PMC9L, 0x03);           /* Set UART1 work in
Receive/Transmit mode */
      SetIORBit(PFC9L, 0x03);
      CKSR1 = UART_BAUDRATE_M1;         /* Baudrate selection */
      BRGC1 = UART_BAUDRATE_K1;
      SetIORBit(SRIC1, Lowest);         /* Set reception interrupt priority
Lowest */
      SetIORBit(STIC1, Lowest);         /* Set transmission interrupt
priority Lowest */
      SetIORBit(ASIM1, 0x80);
#if 1 /* Applilet error - following lines left out of generated code to set
data length and stop bits */
      SetIORBit(ASIM1, 0x04);           /* CL1 = 1, Data length is 8 bits
*/
      ClrIORBit(ASIM1, 0x02);           /* SL1 = 0, Stop length is 1 bits
*/
#endif
#if 0 /* for this program, do not use transmit interrupt */
```

```
      ClrIORBit(STIC1, 0x40);             /* Set UART transmission completion
interrupt enable */
#else
      SetIORBit(STIC1, 0x40);             /* Set UART transmission completion
interrupt disable */
#endif
      ClrIORBit(SRIC1, 0x40);             /* Set UART reception completion
interrupt enable */
      SetIORBit(ASIM1, 0x01);             /* Generate a reception completion
interrupt request (INTSRn) as an interrupt when an error occurs. */
      ClrIORBit(ASIM1, 0x18);             /* Parity mode is None parity */
      SetIORBit(ASIM1, 0x60);             /* Receive/Transmit enable */
#if 1 /* after hardware initialize, add setup for receive interrupt variables
*/
      UART1_Rx_Init();
#endif
      return;
}

#if 0 /* Remove unused Applilet routines - no transmit via isr */
/*
**-------------------------------------------------------------------------
-------------------------------
**
**  Abstract:
**    This function is responsible for UART1 data transfering. and call back
**    function is provided as interface to high level user.
**
**  Parameters:
**    txbuf:            Header pointer of transfer buffer.
**    txnum:            The number of data to transmit(frame number).
**
**  Returns:
**    MD_ERROR:   When an error argument input.
**    MD_OK:      When transfer success.
**
**-------------------------------------------------------------------------
-------------------------------
*/
MD_STATUS UART1_SendData(UCHAR* txbuf, USHORT txnum)
{
      if(txnum < 1){
            return MD_ERROR;              /* 1 frame data should be
transfered at least */
      }

      UART1_TX_ADDRESS = txbuf;
      UART1_TX_CNT = txnum;
      if( txnum == 1) {                   /* Only 1 frame data to transfer
doesn't need contiously transfer */
            if( (ASIF1 & 0x02) == 0){
                  TXB1 = *UART1_TX_ADDRESS;
            }
            else{
                  return MD_DATAEXISTS;
```

```
            }
      }
      else {
            if( (ASIF1 & 0x02) == 0){
                  TXB1 = *UART1_TX_ADDRESS ++ ;

                  while((ASIF1 & 0x2) == 2);    /* wait */
                  UART1_TX_CNT--;
                  TXB1 = *UART1_TX_ADDRESS ++ ;
            }
            else
                  return MD_DATAEXISTS;
      }
      return MD_OK;
}
#endif

#if 0 /* Remove unused Applilet routines - different receive routine via isr
used */
/*
**-------------------------------------------------------------------------
------------------------------
**
**  Abstract:
**    This function is responsible for UART1 data receiving. and call back
**    functions are provided as interface to user.
**
**  Parameters:
**    rxbuf:            Header point of receive buffer.
**    rxnum:            The size of receive buffer.
**
**  Returns:
**    MD_ERROR:   When an error argument input.
**    MD_OK:      When transfer success.
**
**-------------------------------------------------------------------------
------------------------------
*/
MD_STATUS UART1_ReceiveData(UCHAR* rxbuf, USHORT rxnum)
{
      if(rxnum < 1){
            return MD_ERROR;
      }

      UART1_RX_CNT = 0;
      UART1_RX_LEN = rxnum;
      UART1_RX_ADDRESS = rxbuf;

      return MD_OK;
}
#endif

#if 0 /* Remove unused Applilet routines - no transmit isr */
/*
```

```
**------------------------------------------------------------------------------
------------------------------
**
**  Abstract:
**    This function is the high level language interrupt handler
**    for the UART1 transmission completion interrupt (INTST1).
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**------------------------------------------------------------------------------
----------------------------------
*/
__interrupt void MD_INTST1( void )
{
      if(UART1_TX_CNT == 0){
            /* send data complete */
            CALL_UART1_Send();
            return;
      }

      if((ASIF1 & 0x02) == 0 && (ASIM1 & 0x80) == 0x80){
            UART1_TX_CNT--;
            if(UART1_TX_CNT != 0){
                  TXB1 = *UART1_TX_ADDRESS ++ ;
            }
      }
}
#endif

/* Keep Applilet shell for receive isr, but call our function instead */
/*
**------------------------------------------------------------------------------
------------------------------
**
**  Abstract:
**    This function is the high level language interrupt handler
**    for the UART1 receive end interrupt (INTSR1).
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**------------------------------------------------------------------------------
----------------------------------
*/
__interrupt void MD_INTSR1( void )
{
#if 1 /* call replacement function */
      UART1_Rx_Isr();
```

```
#else /* removed Applilet code, not used */
      UCHAR err_type;

      err_type = ASIS1;

      if( err_type & 0x07 ){              /* status check */
            return;
      }

      /* The interrut generated by receive end */
      UART1_RX_CNT ++ ;
      if(UART1_RX_LEN >= UART1_RX_CNT){

            *UART1_RX_ADDRESS ++ = RXB1;

            if(UART1_RX_LEN == UART1_RX_CNT){
                  /* receive data complete, add user own coding */
            }
      }
#endif
}
```

3.1.10 Serial_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 : serial_user.c
**  Abstract : This file gives callback functions for serial module.
**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
**
  Device:  uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
************************************************************************
*/
/*
** ********************************************************************
** Include files
** ********************************************************************
*/
#include "macrodriver.h"
#include "serial.h"
```

```
volatile int u1_dropcount;          /* count of dropped characters */

/* local variables */
volatile char u1_rxbuf[U1_RXBUF_SIZE];    /* buffer to hold received
characters */
volatile char * pu1_rxin;                                   /* pointer to put
next char in buffer */
volatile char * pu1_rxout;                                  /* pointer to get
next char from buffer */
volatile int u1_rxcount;                                    /* count of
characters in buffer */
volatile char u1_xoff_sent;                                 /* 0 if no XOFF
issued, 1 if XOFF issued */
volatile unsigned char last_ASIS1_err_val;            /* last error seen */

/* initialize UART1 for RxIsr */
/* this is called from UART1_Init(), called from SystemInit() */
/* note interrupts are disabled at this point */
void UART1_Rx_Init(void)
{
int i;
      for (i = 0; i < U1_RXBUF_SIZE; i++)
            u1_rxbuf[i] = 0;        /* clear buffer */
      pu1_rxin = u1_rxbuf;          /* pointers to start of buffer */
      pu1_rxout = u1_rxbuf;
      u1_rxcount = 0;                     /* count is zero */
      u1_dropcount = 0;             /* clear count of dropped characters */
      u1_xoff_sent = 0;             /* XOFF not issued */
      last_ASIS1_err_val = 0;       /* clear error location */
}

#if 0 /* for space saving, will use ROM-located code */
void UART1_Rx_push(char c)
{
      /* if room, put in buffer */
      if (u1_rxcount < U1_RXBUF_SIZE) {
            *pu1_rxin++ = c;  /* store in buffer */
            u1_rxcount++;           /* increment count */
            if (pu1_rxin == u1_rxbuf + U1_RXBUF_SIZE) {
                  pu1_rxin = u1_rxbuf;
            }
      } else {          /* if buffer is full, */
            u1_dropcount++;         /* count dropped characters */
      }
}
#endif

extern unsigned int ram_int;
void ram_intsr1( void );

/* interrupt service routine for UART1 character receive */
void UART1_Rx_Isr(void)
{
```

```
#if 1 /* to save space, call ROM-located code for RAM ISR */
      ram_int = 0;
      ram_intsr1();
#else
char c,tmp;
#if 1 /* should not be needed */
      SRIF1 = 0;        /* clear interrupt flag */
      SREIF1 = 0;
#endif
      tmp = ASIS1;      /* get ASIS error status */
      c = RXB1;         /* read character now, in case error */
      if ((tmp & 0x07) != 0) {
            /* error bits set, save error */
            last_ASIS1_err_val = tmp;
            return;                 /* ignore reception */
      }

      UART1_Rx_push(c); /* store in buffer */
      /* send XOFF if buffer has reached this point and not sent yet */
      if (u1_xoff_sent == 0) {
            if (u1_rxcount == U1_XOFF_MARK) {
                  while ((ASIF1 & 0x02) != 0) {
                        /* while waiting for transmit buffer to be available,
*/
                        /* check for additional characters received, and
store */
                        if (SRIF1 == 1) {
                              c = RXB1;
                              SRIF1 = 0;
                              UART1_Rx_push(c);
                        }
                  }
                  TXB1 = XOFF;
                  u1_xoff_sent = 1;
            }
      }
#endif
}

/* check for a character on UART1 */
BOOL UART1_Rx_Chk(void)
{
char c;
      if (u1_rxcount == 0)
            return MD_FALSE;  /* no character received */
      else
            return MD_TRUE;         /* character available */
}

/* get character input */
char UART1_Rx(void)
{
char c;
      if (u1_rxcount == 0)
            return 0;         /* no character received */
```

```
      __asm("di");                          /* disable interrupt while
accessing and updating rx variables */
      c = *pu1_rxout++; /* get character from buffer */
      u1_rxcount--;           /* decrement count */
      if (pu1_rxout == u1_rxbuf + U1_RXBUF_SIZE) {
            pu1_rxout = u1_rxbuf;
      }
      __asm("ei");            /* re-enable interrupts */
      if (u1_xoff_sent != 0) {
            if (u1_rxcount == U1_XON_MARK) {
                  UART1_Tx(XON);
                  u1_xoff_sent = 0;
            }
      }
      return c;
}

/* clear all UART1 RX input */
void UART1_Rx_Clear(void)
{
int i = 50000;
    UART1_Tx(XON);      /* send XON in case XOFF has been sent */
      while (i != 0) {
            if (UART1_Rx_Chk()) {
                  /* if character, read it, reset timeout */
                  UART1_Rx();
                  i = 50000;
            } else {
                  i--;
            }
      }
}

/* equivalent of fgets(char *s, int n, stdin) */
char * UART1_RxFgets(char *s, int n, int timeout) {
int to = timeout;
BOOL done = MD_FALSE;
char c;
char *lp = s;

      if (s == 0) {
            return 0;
      }
      /* return start of string if count is less than 1 */
      if (n < 1) {
            return s;
      }
      do {
            if (UART1_Rx_Chk() == MD_FALSE) {
                  if (timeout != 0) {
                        to--;
                        if (to == 0) {
                              *lp = '\0';       /* mark end of string as
timeout */
```

```
                        done = MD_TRUE;
                    }
                }
            } else {
                c = UART1_Rx();
                to = timeout;
                if ( (c == '\0') || (c == '\r') ) {
                    /* ignore nulls and CR */
                } else if (c == '\n') {
                    *lp = '\0'; /* terminate string */
                    done = MD_TRUE;
                } else {
                    *lp++ = c;
                    n--;
                    if (n == 1) {
                        *lp = '\0';
                        done = MD_TRUE;
                    }
                }
            }

    } while (done == MD_FALSE);

    if ( (timeout != 0) && (to == 0) && (lp == s) ) {
        /* no characters received, timed out */
        return (char *)0;
    } else {
        return s;
    }
}

/* void UART1_Tx(char) */
/*    transmit a character on UART1 */
void UART1_Tx(char c)
{
    while ((ASIF1 & 0x02) != 0)
        ;           /* wait for transmit buffer available */
    TXB1 = c;   /* write character to buffer */
}

/* void UART1_TxStr(char *) */
/*    transmit a string on UART1 */
void UART1_TxStr(char *str)
{
    while (*str != 0) {
        UART1_Tx(*str++);
    }
}

/* send hex character representing low four bits */
void UART1_TxNibble(unsigned char nib)
{
    nib &= 0x0F;
    if (nib > 9)
        nib += (('A' - '0') - 10);
```

```
      nib += '0';
      UART1_Tx((char)nib);
}

/* send two hex characters representing low eight bits */
void UART1_TxByte(unsigned char byte)
{
      UART1_TxNibble(byte >> 4);
      UART1_TxNibble(byte);
}

/* send four hex characters representing low 16-bits */
void UART1_TxHalfword(unsigned short hword)
{
      UART1_TxByte(hword >> 8);
      UART1_TxByte(hword);
}

/* send eight hex characters representing 32 bits */
void UART1_TxWord(unsigned long word)
{
      UART1_TxByte(word >> 24);
      UART1_TxByte(word >> 16);
      UART1_TxByte(word >> 8);
      UART1_TxByte(word);
}

/* send carriage return and linefeed */
void UART1_TxCRLF(void)
{
      UART1_Tx('\r');
      UART1_Tx('\n');
}
```

3.1.11 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     0x0
#define     TM_TMP0_INTERVALVALUE   0x00
#define     TM_TMP0_INTERVALVALUE2  0x00
#define     TM_TMP0_ONESHOTOUTPUTCYCLE    0x00
#define     TM_TMP0_ONESHOTOUTPUTDELAY    0x00
#define     TM_TMP0_EXTTRIGGERCYCLE 0x00
#define     TM_TMP0_EXTTRIGGERDELAY 0x00
#define     TM_TMP0_PWMCYCLE  0x00
#define     TM_TMP0_PWMWIDTH  0x00
#define     TM_TMP0_CCR0COMPARE     0x00
#define     TM_TMP0_CCR1COMPARE     0x00
#define     TM00_Clock  0x0
#define     TM00_INTERVALVALUE      0x270f
#define     TM00_SQUAREWIDTH  0x270f
#define     TM00_PPGCYCLE     0x270f
#define     TM00_PPGWIDTH     0x00
#define     TM00_ONESHOTCYCLE 0x270f
#define     TM00_ONEPULSEDELAY      0x00
#define     TM01_Clock  0x0
#define     TM01_INTERVALVALUE      0x00
#define     TM01_SQUAREWIDTH  0x00
#define     TM01_PPGCYCLE     0x00
#define     TM01_PPGWIDTH     0x00
#define     TM01_ONESHOTCYCLE 0x00
#define     TM01_ONEPULSEDELAY      0x00
#define     TM02_Clock  0x0
#define     TM02_INTERVALVALUE      0x00
#define     TM02_SQUAREWIDTH  0x00
#define     TM02_PPGCYCLE     0x00
#define     TM02_PPGWIDTH     0x00
#define     TM02_ONESHOTCYCLE 0x00
#define     TM02_ONEPULSEDELAY      0x00
#define     TM03_Clock  0x0
#define     TM03_INTERVALVALUE      0x00
#define     TM03_SQUAREWIDTH  0x00
#define     TM03_PPGCYCLE     0x00
#define     TM03_PPGWIDTH     0x00
#define     TM03_ONESHOTCYCLE 0x00
#define     TM03_ONEPULSEDELAY      0x00
#define     TM04_Clock  0x0
#define     TM04_INTERVALVALUE      0x00
```

```
#define     TM04_SQUAREWIDTH  0x00
#define     TM04_PPGCYCLE     0x00
#define     TM04_PPGWIDTH     0x00
#define     TM04_ONESHOTCYCLE 0x00
#define     TM04_ONEPULSEDELAY      0x00
#define     TM05_Clock  0x0
#define     TM05_INTERVALVALUE      0x00
#define     TM05_SQUAREWIDTH  0x00
#define     TM05_PPGCYCLE     0x00
#define     TM05_PPGWIDTH     0x00
#define     TM05_ONESHOTCYCLE 0x00
#define     TM05_ONEPULSEDELAY      0x00
#define     TM50_Clock  0x5
#define     TM50_INTERVALVALUE      0x1e
#define     TM50_SQUAREWIDTH  0x1e
#define     TM50_PWMACTIVEVALUE     0x1e
#define     TM51_Clock  0x5
#define     TM51_INTERVALVALUE      0x1e
#define     TM51_SQUAREWIDTH  0x1e
#define     TM51_PWMACTIVEVALUE     0x1e
#define     TMH0_Clock  0x3
#define     TMH0_INTERVALVALUE      0x7c
#define     TMH0_SQUAREWIDTH  0x7c
#define     TMH0_PWMCYCLE     0x7c
#define     TMH0_PWMDELAY     0x3d
#define     TMH0_CARRIERDELAY 0x7c
#define     TMH0_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 TM00_Init( void );

/*timer00 to 05 free running start,50,51,H0,H1 timer start*/
void TM00_Start( void );

/*timer00 to 05,50,51,H0,H1 timer stop*/
void TM00_Stop( void );
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
__interrupt void MD_INTTM000( void );

/* added functions in timer_user.c for millisecond timer */
void SetMsecTimer(int time);  /* set the timer */
BOOL CheckMsecTimer(void);          /* check the timer */
void SetLEDSpin(BOOL onoff);  /* set whether to spin the LED in ISR */

#endif
```

3.1.12 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 TM00, select founction and input parameter
**    count clock selection, INT init
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**------------------------------------------------------------------------
*/
void TM00_Init( void )
```

```
{
      TMC00 = 0x0;                                        /* stop TM00 */
      ClrIORBit(PRM00, 0x3);
      ClrIORBit(SELCNT1, 0x1);

      SELCNT1 |= ( TM00_Clock&0x4)>>2;    /* internal count clock */
      PRM00 |=( TM00_Clock&0x3);

      /* INTTM000 setting */
      TM0IC00 = Lowest;
      SetIORBit(TM0IC00, 0x40);
      /* TM00 interval */
      ClrIORBit(CRC00, 0x01);
      CR000 = TM00_INTERVALVALUE;
      CR001 = 0xffff;
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    start the TM00 counter
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**
**-------------------------------------------------------------------------
*/
void TM00_Start( void )
{
      TMC00 = 0x0c;                         /* interval timer start */
      ClrIORBit(TM0IC00, 0x40);             /* enable INTTM000 */
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    stop the TM00 counter and clear the count register
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void TM00_Stop( void )
{
      TMC00 = 0x0;                                  /* stop TM00 */
```

```
      SetIORBit(TM0IC00, 0x40);           /* disable INTTM000 */
}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    Change TM00 condition.
**
**  Parameters:
**    USHORT*:     array_reg
**    USHORT:       array_num
**  Returns:
**    MD_OK
**    MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
   switch (array_num){
         case 2:
                  CR001=*(array_reg + 1);
         case 1:
                  CR000=*(array_reg + 0);
                  break;
         default:
                  return MD_ERROR;
         }
        return MD_OK;
}
```

3.1.13 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"
#if 0 /* remove pragma statement for INTTM000 - special section set in
inttab.s */
#pragma interrupt INTTM000 MD_INTTM000
#endif

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

/* counter for millisecond timer */
volatile unsigned int milliseconds;

/* counter for spinning LED in timer ISR */
volatile int msecCount;
BOOL ledSpin;

/*
**-----------------------------------------------------------------------
----------
**
**  Abstract:
**    TM00 INTTM000 Interrupt service routine
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------
----------
*/
extern unsigned int ram_int;
void ram_inttm00(void);

__interrupt void MD_INTTM000( void )
{
#if 1 /* share ROM-located code for RAM isr */
      ram_int = 0;
      ram_inttm00();
#else
unsigned char tmp;
```

```
      /* count down millisecond timer */
      if (milliseconds > 0)
            milliseconds--;
      /* if we are spinning the LED, count out and do it */
      if (ledSpin == MD_TRUE) {
            if (msecCount == 0) {
                  msecCount = 333;
                  tmp = ~PDLH;
                  tmp = tmp << 1;
                  if (tmp == 0)
                        tmp = 1;
                  if (tmp == 0x40)
                        tmp = 1;
                  PDLH = ~tmp;
            } else {
                  msecCount--;
            }
      }
#endif
}

/* set the millisecond timer */
void SetMsecTimer(int time)
{
      milliseconds = time;
}

/* check the millisecond timer */
BOOL CheckMsecTimer(void)
{
      if (milliseconds > 0)
            return MD_FALSE;
      return MD_TRUE;
}

/* turn LED spinning on or off in ISR */
void SetLEDSpin(BOOL onoff)
{
      ledSpin = onoff;
}
```

3.1.14 850_splib.dir

```
#*
#*****************************************************************************
*
#**
#**  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 : 850.dir
#**  Abstract : This is the link file for CA850
#**  APIlib:  V850ESKX1H.lib V1.33 [24 Sep 2004]
#**
#*************************************************************************
*
#*
#******* NOTE ********
#* Original 850.dir from Applilet saved to 850_orig.dir; this file modified
#* changed order of segments, to provide SCONST at 0x400, followed by CONST,
followed by TEXT
#******* NOTE ********
#* 850_splib.dir modified to add Self-Programming Library sections


OPT     : !LOAD ?R V0x7a{
          .opt        = $PROGBITS ?A .opt;
    };

# Special SCONST segments, inside interrupt area
# S_SCONST0 between SECURITY/OPT (0070-007F) and INTTM000 (0100)
S_SCONST0     : !LOAD ?R V0x0080{
      S_SCONST0.sconst        = $PROGBITS ?A S_SCONST0.sconst;
      };

# Specific section for INTTM000, instead of pragma
S_INTTM000 : !LOAD ?RX V0x0100{
          S_INTTM000  = $PROGBITS ?AX S_INTTM000 ;
    };

# Special SCONST segments, inside interrupt area
# S_SCONST1 between INTTM000 (0100) and INTSR0 (0190)
S_SCONST1     : !LOAD ?R V0x0104{
      S_SCONST1.sconst        = $PROGBITS ?A S_SCONST1.sconst;
      };

# Specific section for INTSR0 (used by debugger)
S_INTSR0 : !LOAD ?RX V0x0190{
          S_INTSR0  = $PROGBITS ?AX S_INTSR0 ;
    };

# Special SCONST segments, inside interrupt area
# S_SCONST1 between INTSR0 (0190) and INTSR1 (01C0)
S_SCONST2     : !LOAD ?R V0x0194{
      S_SCONST2.sconst        = $PROGBITS ?A S_SCONST2.sconst;
      };

# Specific section for INTSR1, instead of pragma
S_INTSR1 : !LOAD ?RX V0x01C0{
          S_INTSR1  = $PROGBITS ?AX S_INTSR1 ;
    };
```

```
# Locate start of ROM at 0x1C4, inside unused interrupt vectors, for space
reasons
# Normally, should be at 0x400
SCONST     : !LOAD ?R V0x1C4{
      .sconst        = $PROGBITS ?A .sconst;
      };

CONST     : !LOAD ?R {
      .const        = $PROGBITS ?A .const;
      };

TEXT  : !LOAD ?RX {
      .pro_epi_runtime = $PROGBITS  ?AX;
      .text        = $PROGBITS      ?AX;
};

# Separate segment for text for SelfLib, because rompsec placed after text
# Must specify an address in ROM, with enough space for rompsec
# End of TEXTSL must come before end of boot block (0x0001FFF)
# Check link map and romp map for possible conflicts
# Address = 0x0001700
TEXTSL      : !LOAD ?RX V0x0001780 {
#       SelfLib ROM sections
            SelfLib_Rom.text            = $PROGBITS    ?AX SelfLib_Rom.text;
            SelfLib_ToRamUsrInt.text    = $PROGBITS    ?AX
SelfLib_ToRamUsrInt.text;
            SelfLib_ToRamUsr.text       = $PROGBITS    ?AX
SelfLib_ToRamUsr.text;
            SelfLib_ToRam.text          = $PROGBITS    ?AX
SelfLib_ToRam.text;
            SelfLib_RomOrRam.text       = $PROGBITS    ?AX
SelfLib_RomOrRam.text;
};

# Specify place to load word at end of boot ROM
S_ROM_END : !LOAD ?R V0x0001FFC {
      S_ROM_END   = $PROGBITS ?A S_ROM_END;
};

  DATA      : !LOAD ?RW V0xffffe000 {

      .data       = $PROGBITS ?AW;
      .sdata            = $PROGBITS ?AWG;
      .sbss       = $NOBITS   ?AWG;
      .bss        = $NOBITS   ?AW;
};

STACK : !LOAD ?RW V0xffffee00{
      .stack            = $PROGBITS     ?AW     .stack;
};

# replaced following definitions with those from SelfLibApp.dir
#__tp_TEXT @ %TP_SYMBOL{TEXT};
#__gp_DATA @ %GP_SYMBOL{DATA} &__tp_TEXT{DATA};
```

```
#__ep_DATA @ %EP_SYMBOL;
# these are the replacements
__tp_TEXT @ %TP_SYMBOL;
#__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA SELFLIB_RAM_SECTION};
__ep_DATA @ %EP_SYMBOL;
```

3.1.15 Fsp.h

```
/*
***************************************************************************
**
**
**  This file was created for the NEC V850ES Flash Self-Program Application
Note
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :    fsp.h
**  Abstract :    This file implements header for fsp.c
**
**  Device:    uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
***************************************************************************
**
*/
#ifndef     _FSP_H_
#define     _FSP_H_
/*
***************************************************************************
**
**  MacroDefine
***************************************************************************
**
*/

/* Flash Self-Program Error (FSPE) code returns from routines */
#define MD_FSPE_ERROR   (MD_ERROR)
#define MD_FSPE_NOINIT  (MD_FSPE_ERROR + 1)     /* could not init flash */

#if 0
#define MD_FSPE_PARAM   (MD_FSPE_ERROR + 1)     /* parameter error */
#define MD_FSPE_NODATA  (MD_FSPE_ERROR + 2)     /* no data found */
#define MD_FSPE_NOBLOCK (MD_FSPE_ERROR + 3)     /* no valid block found */
#define MD_FSPE_NOWCOMP (MD_FSPE_ERROR + 5)     /* readback of written data
does not compare */
#define MD_FSPE_NOCHANGE      (MD_FSPE_ERROR + 6)     /* UseChangeBlock
failed */
```

```
#define MD_FSPE_NOADDR  (MD_FSPE_ERROR + 7)     /* WriteTopSearch failed */
#define MD_FSPE_NOERASE (MD_FSPE_ERROR + 8)     /* could not erase */
#define MD_FSPE_NOWRITE (MD_FSPE_ERROR + 9)     /* could not write flash */
#endif

#define MD_FSPE_NOTIMP  (MD_FSPE_ERROR + 15)    /* function not implemented
yet */

/* define Flash memory block size */
#define FSP_BLOCK_SIZE  2048  /* 2K blocks */
#define FSP_BLOCK_WGRAN 1           /* word granularity in block */

/* define memory locations for User Program area */
#define FSP_PROG_SADDR  0x00008000  /* entry address for program */
#define FSP_PROG_SIZE   0x8000            /* 32K size for loaded user program
*/
#define FSP_PROG_EADDR  ((FSP_PROG_SADDR) + (FSP_PROG_SIZE))      /* end
address + 1 */
#define FSP_PROG_BLK_S  ((FSP_PROG_SADDR) / (FSP_BLOCK_SIZE))     /* starting
block */
#define FSP_PROG_BLK_E  ((FSP_PROG_EADDR) / (FSP_BLOCK_SIZE))     /* end
block + 1 */

#define FSP_BOOT_SADDR  0x00002000  /* start address for boot swap area */
#define FSP_BOOT_SIZE   0x2000            /* size of boot */
#define FSP_BOOT_EADDR  ((FSP_BOOT_SADDR) + (FSP_BOOT_SIZE))      /* end
address + 1 */
#define FSP_BOOT_BLK_S  ((FSP_BOOT_SADDR) / (FSP_BLOCK_SIZE))     /* start
block */
#define FSP_BOOT_BLK_E  ((FSP_BOOT_EADDR) / (FSP_BLOCK_SIZE))     /* end
block + 1 */

/* Flash Self-Program functions */
MD_STATUS FSP_Init(void);                 /* initialize Flash Self-
Programming */
MD_STATUS FSP_LoadProgram(unsigned long saddr, unsigned long eaddr, unsigned
long offset );    /* load subprogram */
MD_STATUS FSP_RunProgram(void);           /* run loaded subprogram */
void      FSP_DemoIsr(void);        /* demonstrate ISR in FSP mode */
MD_STATUS FSP_Bootswao(void);       /* swap in alternate boot program */

#endif /* _FSP_H_ */
```

3.1.16 Fsp.c

```
/*
************************************************************************
**
**
**  This file was created for the NEC V850ES Flash Self-Program Application
Note
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
```

```
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :    fsp.c
**  Abstract :    This file implements functions for flash self-programming
**
**  Device:    uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*****************************************************************************
**
*/
/*
*****************************************************************************
**
**  Include files
*****************************************************************************
**
*/
#include "macrodriver.h"
#include "fsp.h"
#include "serial.h"           /* for UART1_TxStr() */
#include "timer.h"            /* for timer functions */
#include "ihexin.h"           /* for Intel Hex format input */
#include "led_vkj1.h"   /* for LED output */

/* include file and definition for Self-Programming Library */
#define _DECLARE_VARS
#include "selfprog.h"

/*
**-------------------------------------------------------------------------
--
**  Abstract:
**    Function: do Flash Self-Program Initialization
**
**  Parameters: None
**  Returns:
**    MD_OK if initialize is successful, MD_ERROR if fails
**
**-------------------------------------------------------------------------
--
*/
MD_STATUS FSP_Init(void)
{
MD_STATUS status = MD_FSPE_NOINIT;
u32 retval;
      /* set output mode of FLMD0 control pin, and set off initially */
      SET_MODE_FLMD0();
      SET_FLMD0_OFF();

      /* initialize the self programming library */
```

```
    retval = SelfLib_Init ( (void *)SELFLIB_RAMADD,
                  FREQUENCY,
                  (SECTION_COPY)SELFLIB_SECTIONS );
      if (retval == SELFLIB_OK) {
            status = MD_OK;
      } else {
            status = MD_FSPE_NOINIT;
      }
      return status;    /* if neither define above is true */
}

/* strings here so we can assign locations */
/* note S_SCONST0 is shared with other files */
#pragma section sconst "S_SCONST0" begin
const char s_Run[]                  = "Run...\r\n";
const char s_DemoISR[]        = "Demo ISR\r\n";
const char s_BootSwapEq[]     = "BootSwap=";
const char s_SecFlagsEq[]     = "SecFlags=";
const char s_BadSecFlagEq[]   = "BAD SECFLAG=";
#pragma section sconst "S_SCONST0" end

#pragma section sconst "S_SCONST1" begin
const char s_StartEq[]        = "Start = ";
const char s_EndEq[]          = ", End = ";
const char s_RetBlkChk[]      = "\r\nBlank Check...";
const char s_Erasing[]        = "Erasing...";
const char s_ErrorEq[]        = " Error = ";
const char s_Erased[]         = "Erased\r\n";
const char s_Blank[]          = "Blank\r\n";
const char s_Aborting[]       = "Aborting\r\n";
const char s_SendFile[]       = "Send file...";
const char s_FLineErr[]       = "First line Error\r\n";
const char s_Loading[]        = "loading....\r\n";
const char s_WnAddr[]         = "Wn: addr ";
#pragma section sconst "S_SCONST1" end

#pragma section sconst "S_SCONST2" begin
const char s_LtStart[]        = " < start - ignored\r\n";
const char s_GtEnd[]          = " > end - ignored\r\n";
#pragma section sconst "S_SCONST2" end

/* other strings will go in regular SCONST section */
const char s_AddrEq[]         = "Addr=";
const char s_NbytesEq[]       = ", Nbytes=";
const char s_WaddrEq[]        = ", Waddr=";
const char s_FlashProgErr[]   = "Flash Prog Err ";
const char s_ErrLine[]        = "Err Line ";
const char s_Code[]                 = ", code ";
const char s_LoadOK[]         = "Load OK\r\n";
const char s_LoadError[]      = "Load Error\r\n";
const char s_VerifyOK[]       = "Verify OK\r\n";
const char s_VerifyError[]    = "Verify Error\r\n";
const char s_SwapBoot[]     = "Swap Boot\r\n";
const char s_SetSecFlagEq[] = "Set secflag=";
```

```
static HEX_LINE hexline;                  /* structure to hold parsed hex
line */
static u32 warray[FSP_BLOCK_WGRAN]; /* array of words for programming */
static u32 waddr;                         /* word address for programming */

MD_STATUS FSP_LoadProgram(unsigned long saddr, unsigned long eaddr, unsigned
long offset)
{
MD_STATUS status = MD_OK;
unsigned long writeaddr;
u32 ret, sb, eb;  /* return code, start block, end block */
unsigned char *lpnt,*bpnt,*barray_end;
int error;
int i;

      TM00_Stop();            /* don't run timer while loading */
    /* clear remaining UART input */
    UART1_Rx_Clear();

      /* report start and end, and tell Blank Check starting */
      UART1_TxStr((char *)s_StartEq);
      UART1_TxWord(saddr);
      UART1_TxStr((char *)s_EndEq);
      UART1_TxWord(eaddr - 1);
      UART1_TxStr((char *)s_RetBlkChk);

      /* set starting and ending block to block offset into flash block area
*/
      sb = saddr / FSP_BLOCK_SIZE;                          /* set starting
block */
      eb = ((eaddr + (FSP_BLOCK_SIZE - 1)) / FSP_BLOCK_SIZE) - 1;       /*
set last block number */

      /* see if block is blank */
    SET_FLMD0_ON();           /* Switch on voltage at FLMD0 pin */
    ret = SelfLib_BlankCheck( sb, eb );
    SET_FLMD0_OFF();           /* Switch off voltage at FLMD0 pin */

    if ( SELFLIB_OK != ret ) {
        /* Blocks are not blank, erase them */
        UART1_TxStr((char *)s_Erasing);
          SET_FLMD0_ON();           /* Switch on voltage at FLMD0 pin */
            ret = SelfLib_Erase( sb, eb );
          SET_FLMD0_OFF();           /* Switch off voltage at FLMD0 pin */
          if ( SELFLIB_OK != ret ) {
            /* erase failed */
            status = MD_ERROR;
            UART1_TxStr((char *)s_ErrorEq);
            UART1_TxWord(ret);
            UART1_TxCRLF();
          } else {
            UART1_TxStr((char *)s_Erased);
          }
    } else {
            /* blank check returned zero, no erase needed */
```

```
      UART1_TxStr((char *)s_Blank);
    }
      if (status != MD_OK) {
            UART1_TxStr((char *)s_Aborting);
            TM00_Start();     /* restart timer */
            return status;
      }

      /* now progRAM with data from file */
      UART1_TxStr((char *)s_SendFile);
      hexline.saddr = 0;      /* set segment address to zero */
      error = ihex_fget_hline(&hexline, 0);     /* get first line with no
timeout */

      if (error != 0) {
            status = MD_ERROR;
            UART1_TxStr((char *)s_FLineErr);
      } else {
            UART1_TxStr((char *)s_Loading);
      }

      while ((status == MD_OK) && (hexline.type != HRT_EOF)) {
            writeaddr = hexline.hexaddr + offset;
            if (writeaddr < saddr) {
                  UART1_TxStr((char *)s_WnAddr);
                  UART1_TxWord(hexline.hexaddr);
                  UART1_TxStr((char *)s_LtStart);
            } else if ((writeaddr + hexline.nbytes) > eaddr) {
                  UART1_TxStr((char *)s_WnAddr);
                  UART1_TxWord((hexline.hexaddr + hexline.nbytes) - 1);
                  UART1_TxStr((char *)s_GtEnd);
            } else {
                  /* address in bounds, show address and bytes */
                  UART1_TxStr((char *)s_AddrEq);
                  UART1_TxWord(hexline.hexaddr);
                  UART1_TxStr((char *)s_NbytesEq);
                  UART1_TxHalfword(hexline.nbytes);
                  UART1_TxStr((char *)s_WaddrEq);
                  UART1_TxWord(writeaddr);
                  UART1_TxCRLF();
                  /* write the data to flash memory */
                  lpnt = hexline.bytes;         /* source of bytes from line
*/
                  while ((hexline.nbytes != 0) && (status == MD_OK)) {
                        /* read current memory to warray */
                        waddr = writeaddr & 0xFFFFFFFC;           /* form
word address */
                        for (i=0; i < FSP_BLOCK_WGRAN; i++) {
                              warray[i] = *((u32 *)waddr);  /* read words to
array */
                              waddr = waddr + 4;
                        }
                        waddr = writeaddr & 0xFFFFFFFC;           /* reform
word address */
                        /* copy data from hex input to warray */
```

```
                        bpnt = (unsigned char *)(&warray[0]);           /*
point to start of array */
                        barray_end = bpnt + (FSP_BLOCK_WGRAN * 4);      /*
point past end of array */
                        bpnt = bpnt + (writeaddr & 0x03);         /* adjust
for alignment */
                        while ((bpnt < barray_end) && (hexline.nbytes != 0))
{
                              *bpnt++ = *lpnt++;
                              hexline.nbytes--;
                              writeaddr++;
                        }
                        /* now write words */
                      SET_FLMD0_ON();            /* Switch on voltage at
FLMD0 pin */
#if 1
                        ret = SelfLib_Write((void *)warray, (void *)waddr,
FSP_BLOCK_WGRAN);
#else /* to debug file loading and parsing, do flash operation but don't
write */
                        ret = SelfLib_BlankCheck( blockNo_S, blockNo_E );
#endif
                      SET_FLMD0_OFF();            /* Switch off voltage at
FLMD0 pin */
                      if ( SELFLIB_OK != ret ) {
                        UART1_TxStr((char *)s_FlashProgErr);
                        UART1_TxWord(ret);
                        UART1_TxCRLF();
                        status = MD_ERROR;
                        break;      /* exit loop */
                      }
                  } /* end of while nbytes != 0 */
            } /* end of if-else-else for address check */

            if (status == MD_OK) {
                  /* get next line */
                  error = ihex_fget_hline(&hexline, 0x10000);     /* timeout
on next send */
                  if (error != 0) {
                        UART1_TxStr((char *)s_ErrLine);
                        UART1_TxWord(hexline.lineno);
                        UART1_TxStr((char *)s_Code);
                        UART1_TxWord(error);
                        UART1_TxCRLF();
                        status = MD_ERROR;
                        break;
                  }
            }
      }

      if (status == MD_OK) {
            UART1_TxStr((char *)s_LoadOK);
            /* do an internal verify of the programmed area */
          SET_FLMD0_ON();            /* Switch on voltage at FLMD0 pin */
            ret = SelfLib_IVerify( sb, sb);
```

```
          SET_FLMD0_OFF();            /* Switch off voltage at FLMD0 pin */
            if ( SELFLIB_OK != ret ) {
                  UART1_TxStr((char *)s_VerifyError);
                  status = MD_ERROR;
            } else {
                  UART1_TxStr((char *)s_VerifyOK);
            }
      } else {
            UART1_TxStr((char *)s_LoadError);
      }

    /* clear remaining UART input */
    UART1_Rx_Clear();
    TM00_Start();       /* restart the timer */

      return status;
}

MD_STATUS FSP_RunProgram(void)
{
MD_STATUS status = MD_OK;
void (*fnptr)(void);

      UART1_TxStr((char *)s_Run);
      fnptr = (void(*)(void)) FSP_PROG_SADDR;
      fnptr();
      /* program does not return */

      return status;
}

/*
**-------------------------------------------------------------------------
--
**  Abstract:
**    Function: demonstrate interrupts during Flash Self-Program operations
**
**  Parameters: None
**  Returns:    None
**
**-------------------------------------------------------------------------
--
*/
void FSP_DemoIsr(void)
{
u32 ret;
int i;
      UART1_TxStr((char *)s_DemoISR);
      /* first show ISR in non-FSP mode */
      /* Set to spin right LED in timer ISR */
      SetLEDSpin(MD_TRUE);

      SetMsecTimer(5000);           /* wait 5 seconds */
      while (!CheckMsecTimer())
            ;
```

```
      /* now do operations in flash self program mode */
      /* left LED should spin if interrupt happens if FSP mode */
    SET_FLMD0_ON();            /* Switch on voltage at FLMD0 pin */
      for (i=0; i<300; i++) { /* do enough times to make visible */
            /* see if first block of user program area is blank */
          ret = SelfLib_BlankCheck( FSP_PROG_BLK_S, FSP_PROG_BLK_S );
      }
    SET_FLMD0_OFF();           /* Switch off voltage at FLMD0 pin */

      /* now wait in non-FSP mode - right LED should spin again */

      SetMsecTimer(5000);          /* wait 5 seconds */
      while (!CheckMsecTimer())
            ;

      /* Stop LED spin in timer ISR */
      SetLEDSpin(MD_FALSE);
}

/*
**-------------------------------------------------------------------------
--
**  Abstract:
**    Function: read and report boot program information
**
**  Parameters: None
**  Returns:    None
**
**-------------------------------------------------------------------------
--
*/
void FSP_BootInfo(void)
{
u32 bootswap, secflag;

    SET_FLMD0_ON();            /* Switch on voltage at FLMD0 pin */
      bootswap = SelfLib_GetInfo_BootSwap();
      UART1_TxStr((char *)s_BootSwapEq);
      UART1_TxWord(bootswap);
      UART1_TxCRLF();

#if 0
      ret = SelfLib_GetInfo_BlockCnt();
      UART1_TxStr("GetInfo_BlockCnt returns ");
      UART1_TxWord(ret);
      UART1_TxCRLF();

      ret = SelfLib_GetInfo_BlockEndAdd((u32) 3);
      UART1_TxStr("GetInfo_BlockEndAdd(3) returns ");
      UART1_TxWord(ret);
      UART1_TxCRLF();
#endif

      secflag = SelfLib_GetInfo_SecFlags();
```

```
      UART1_TxStr((char *)s_SecFlagsEq);
      UART1_TxWord(secflag);
      UART1_TxCRLF();
    SET_FLMD0_OFF();            /* Switch off voltage at FLMD0 pin */
}

/*
**-------------------------------------------------------------------------
--
**  Abstract:
**    Function: swap boot program from alternate boot page
**
**  Parameters: None
**  Returns:
**    MD_OK if operation is successful, MD_ERROR if fails
**
**-------------------------------------------------------------------------
--
*/
MD_STATUS FSP_Bootswap(void)
{
MD_STATUS status = MD_OK;
u32 ret, blockNo_S, blockNo_E;
u32 secflag, bootswap;

      UART1_TxStr((char *)s_SwapBoot);

      /* report status before boot swap set */
      FSP_BootInfo();

    SET_FLMD0_ON();            /* Switch on voltage at FLMD0 pin */

      secflag = SelfLib_GetInfo_SecFlags();     /* get security flags */
      secflag = secflag & 0x03FFFFFF;                /* set boot cluster
bits to 03 */
      bootswap = SelfLib_GetInfo_BootSwap() & 0x00000001;   /* get state of
boot swap */
      if (bootswap == 0) {
            secflag = secflag & 0xFFFFFFFE;          /* set bit 0 of
secflags to opposite state */
      } else {
            secflag = secflag | 0x00000001;
      }
      if ( ((secflag & 0x20)==0) ||
           ((secflag & 0x02)==0) ) {
            /* NEVER clear bits 5 (boot cluster protect) or 1 (disable chip
erase) !! */
            UART1_TxStr((char *)s_BadSecFlagEq);
            UART1_TxWord(secflag);
            UART1_TxCRLF();
      } else {
            /* WARNING! Clearing bits 5 or 1 will make chip unable to be
programmed again! */
            /* DO NOT clear these bits ! */
            UART1_TxStr((char *)s_SetSecFlagEq);
```

```
            UART1_TxWord(secflag);
            UART1_TxCRLF();
            ret = SelfLib_SetSecFlags(secflag);
            UART1_TxWord(ret);                  /* report return value */
            UART1_TxCRLF();
      }

    SET_FLMD0_OFF();            /* Switch off voltage at FLMD0 pin */

      /* report status after boot swap set */
      FSP_BootInfo();

      ret &= 0xFF;
      led_dig_left(ret >> 4);
      led_dig_right(ret & 0x0F);    /* show result in LEDs */
      /* delay to allow digits to be seen */
      ret = 1000000;
      while (ret != 0) {
            __asm("nop");
            __asm("nop");
            ret--;
      }
      /* restart program with jump to location 00000000 */
      __asm("jmp [r0]");

      return status;
}
```

3.1.17 Ihexin.h

```
/*
*********************************************************************************
**
**
**  This file was created for the NEC V850ES Flash Self-Program Application
Note
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :    fsp.h
**  Abstract :    This file implements header for ihexin.c
**
**  Device:    uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*********************************************************************************
**
*/
```

```
/*
      Based on ihexin.h from MSI VAIO2::E:\GA\USBMST\MSTDLL\DLL, 02-25-2004
*/

#ifndef     _IHEXIN_H_
#define _IHEXIN_H_

/* definitions */

typedef unsigned char BYTE;         /* same as UCHAR */

#ifndef FALSE
  #ifdef MD_FALSE
    #define FALSE MD_FALSE
  #else
    #define FALSE 0
  #endif
#endif

#ifndef TRUE
  #ifdef MD_TRUE
    #define TRUE  MD_TRUE
  #else
    #define TRUE  1
  #endif
#endif

#define     MAX_HEX_BYTES     256                     /* allow full line */
#define     MAX_HEX_LINE      ((MAX_HEX_BYTES)*2 + 12 + 10) /* 2 per byte +
record + slop */

/* error codes returned */
#define     LHF_ERROR                     -1
#define     LHF_FILE_NOT_FOUND            -2
#define     LHF_EOF                             -3
#define     LHF_ILLEGAL_CHAR        -4
#define     LHF_ILLEGAL_HEX_RECORD  -5
#define     LHF_BAD_CHECKSUM        -6
#define     LHF_WRITE_MEM_ERROR           -7
#define     LHF_USER_ABORT                -8
#define     LHF_UNDEFINED_ERROR           -9    /* this should be last and
highest number */

typedef enum {
      HRT_EOF = 0,
      HRT_DATA,
      HRT_EXTADDR,
      HRT_STARTADDR
} HEX_REC_TYPE;

typedef struct {
      int                     lineno;           /* line number in file */
      HEX_REC_TYPE      type;       /* record type read */
      unsigned long     hexaddr;    /* extended address of data read */
```

```
      unsigned long     saddr;            /* segment address (set this to 0
before first read) */
      int                     nbytes;           /* number of bytes read */
      unsigned char     bytes[MAX_HEX_BYTES];   /* array to hold bytes read
*/
} HEX_LINE;


/* function prototypes */

/* return error string for given int error */
char * ehex_fget_hline(int err);

/* read hex line from input into hexline structure */
int ihex_fget_hline(HEX_LINE * hline, int timeout);

#endif      /* _IHEXIN_H_ */
```

3.1.18 Ihexin.c

```
/*
*************************************************************************
**
**
**  This file was created for the NEC V850ES Flash Self-Program Application
Note
**
**  Copyright(C) NEC Electronics Corporation 2002 - 2006
**  All rights reserved by NEC Electronics Corporation.
**
**  This program should be used on your own responsibility.
**  NEC Electronics Corporation assumes no responsibility for any losses
**  incurred by customers or third parties arising from the use of this file.
**
**  Filename :    ihexin.c
**  Abstract :    This file implements input of Intel HEX data
**
**  Device:    uPD70F3318Y
**
**  Compiler:  NEC/CA850
**
*************************************************************************
**
*/
/*
    Based on MSI ihexin.cpp, VAIO2::E:\GA\USBMST\MSTDLL\DLL 02-25-2004
*/

/*
*************************************************************************
**
**  Include files
*************************************************************************
**
*/
```

```
#include "macrodriver.h"
#include "ihexin.h"
#include "serial.h"           /* for UART1_TxStr() */
#include "stdio.h"

/* load a hex file to specified memory area */

#if 0
/* strings and function for error lookup and report, if desired */
char  *lhf_errmsg[] = {
      "LHF_ERROR: General hex error",
      "LHF_FILE_NOT_FOUND: Hex file not found",
      "LHF_EOF: End of file before end hex record",
      "LHF_ILLEGAL_CHAR: Illegal character in hex value",
      "LHF_ILLEGAL_HEX_RECORD: Unrecognized hex record type",
      "LHF_BAD_CHECKSUM: Checksum for hex line incorrect",
      "LHF_WRITE_MEM_ERROR",
      "LHF_USER_ABORT",
      "LHF_UNDEFINED_ERROR"
      };

char *ehex_fget_hline(int err)
{
      if (err < LHF_UNDEFINED_ERROR || err >= 0)
            err = LHF_UNDEFINED_ERROR;
      err = -err - 1;
      return lhf_errmsg[err];
}
#endif

static BYTE lhf_checksum;     /* line checksum */
static char *lp;              /* pointer to input line */
static int  lhf_error;        /* line error flag */

BYTE
get_nibble(void)
{
char  c;
      c = *lp++;
      if ( (c >= '0') && (c <= '9') ) {
            return (c - '0');
      }
      if ( (c >= 'A') && (c <= 'F') ) {
            return ((c - 'A') + 10);
      }
      /* allow lowercase hex digits */
      if ( (c >= 'a') && (c <= 'f') ) {
            return (c - 'a' + 10);
      }
      lhf_error = LHF_ILLEGAL_CHAR;
      return 0;
}

BYTE
get_byte(void)
```

```
{
BYTE  b;
      b = get_nibble() << 4;
      if (!lhf_error)
            b = b | get_nibble();
      lhf_checksum += b;
      return (b);
}

int
ihex_fget_hline(HEX_LINE *hline, int timeout)
{
char  line[MAX_HEX_LINE];           /* input buffer for lines */
BYTE  *bp;                          /* pointer to bytes */
int   hcount, i;                    /* count of bytes in line */
UINT  haddr;                              /* hex address */
BYTE  htype;                              /* record type */
BOOL  done = FALSE;

      do {
            lhf_error = 0;                /* set line error to zero */
            lhf_checksum = 0;       /* set line checksum to zero */
            bp = hline->bytes;            /* point at start of byte buffer */

        hline->hexaddr = 0;
        hline->nbytes = 0;
            /* get line of data */
            lp = UART1_RxFgets(line, MAX_HEX_LINE, timeout);      /* get hex
line with no timeout */
#if 1 /* debug echo of line */
            if (lp) {
                  UART1_TxStr(lp);
                  UART1_TxCRLF();
            }
#endif
            hline->lineno++;
            if (!lp)
                  lhf_error = LHF_EOF;

            /* find start of record */
            if (!lhf_error) {
                  while (*lp != ':' && *lp != '\0')
                        lp++;
                  if (*lp == '\0')
                        continue;               /* skip lines with no colon
*/
                  }
            lp++;                               /* point past colon */

            /* get byte count */
            if (!lhf_error)
                  hcount = get_byte();
            else
                  hcount = 0;
            if (!lhf_error)
```

```
                haddr = get_byte();
            if (!lhf_error)
                haddr = (haddr << 8) | get_byte();
            if (!lhf_error) {
                htype = get_byte();
                switch (htype) {
                      case 2:           /* type 2 is extended address
record */
                            hline->type = HRT_EXTADDR;
                            break;
                      case 3:           /* type 3 is set start address
record */
                            hline->type = HRT_STARTADDR;
                            break;

                      case 0:           /* type 0 is data record - return
every line */
                            hline->type = HRT_DATA;
                            done = TRUE;
                            break;

                      case 1:
                            hline->type = HRT_EOF;
                            done = TRUE;      /* type 1 is end-of-file
record */
                            break;

                      default:
                            lhf_error = LHF_ILLEGAL_HEX_RECORD;
                            break;
                      }
                }
            /* get the bytes to byte array */
            i = hcount;
            if (!lhf_error && (hcount !=0))
                while (i-- && !lhf_error)
                      *bp++ = get_byte();
            /* get the checksum */
            if (!lhf_error) {
                get_byte();
                if (lhf_checksum != 0)
                      lhf_error = LHF_BAD_CHECKSUM;
                }
            if (!lhf_error) {
                if (htype == 0) {
                      /* update the structure */
                            hline->hexaddr = (unsigned long)haddr + hline-
>saddr;
                    hline->nbytes = hcount;
                      }
                if (htype == 2) {
                      /* set the segment address */
                            hline->saddr = hline->bytes[0];     /* high
byte of address */
```

```
                              hline->saddr = (hline->saddr << 8) + hline-
>bytes[1];  /* shift, add low byte */
                              hline->saddr = hline->saddr << 4;   /* shift up
by 16 for segment */
                        }

                  if (htype == 3) {
                        /* set the start address */
                  /* not implemented at this time */
                        }
            }

      } while (!done && !lhf_error);

      return lhf_error;
}
```

3.1.19 Led_vkj1.h

```
/* led_vkj1.h     */
/*    header for M-V850ES-KJ1 CPU board for LED digit display */
/* Version 1.1    05-08-2006
      */

#ifndef _LED_VKJ1_H
#define _LED_VKJ1_H

/*********************************************************************/
/* 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

/*****************************************************************/
/* Export functions                                              */
/*****************************************************************/
extern void led_init(void);                                  /* init ports for
LED output  */
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
      */
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_dig(unsigned char num);              /* display number as
hex         */
extern void led_dig_bcd(unsigned char bcdnum);  /* display number as BCD
      */

#endif      /* _LED_KJ1_H */
```

3.1.20 Led_vkj1.c

```
/*    led_vkj1.c - routines for LED                    */
/*    for M-V850ES-KJ1 CPU board on M-Station base board    */
/*  Version:      1.1   05-08-2006                              */

/*          PDL8-PDL15 = output to right digit (LED2)       */
/*          PDH0-PDH7  = output to left digit  (LED1)       */

/*    To connect ports to LEDs on M-Station 1.1, make the
      following jumper connections between ROW1 and ROW2.
      To connect ports to LEDs on M-Station 2, make sure
      the default SBxx connections are inserted.
      Port  LED         M-Station 1.1     M-Station 2.2
      ----  ---         -------------     -------------
      PDL8        2-A         R1.25 - R2.25     SB27
      PDL9        2-B         R1.26 - R2.26     SB28
      PDL10       2-C         R1.27 - R2.27     SB29
      PDL11       2-D         R1.28 - R2.28     SB30
      PDL12       2-E         R1.29 - R2.29     SB31
      PDL13       2-F         R1.30 - R2.30     SB32
      PDL14       2-G         R1.31 - R2.31     SB33
      PDL15       2-DP  R1.32 - R2.32     SB34
```

```
      PDH0        1-A         R1.17 - R2.17     SB35
      PDH1        1-B         R1.18 - R2.18     SB36
      PDH2        1-C         R1.19 - R2.19     SB37
      PDH3        1-D         R1.20 - R2.20     SB38
      PDH4        1-E         R1.21 - R2.21     SB39
      PDH5        1-F         R1.22 - R2.22     SB40
      PDH6        1-G         R1.23 - R2.23     SB41
      PDH7        1-DP  R1.24 - R2.24     SB42
*/

/* NOTE: on M-Station Base V1.0 prototype, PDH0-PDH7 are    */
/*  located at ROW4.1-8, and need to be wirewrapped to      */
/*  connect to ROW2.17-24 to drive LED1.                    */

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

#include "led_vkj1.h"

#pragma section sconst "S_SCONST0" begin
/* table of bit patterns for seven-segment digits */
const 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 */
};
#pragma section sconst "S_SCONST0" end

/*    void led_init(void) */
/*          set up ports for display of LED digits */
void led_init(void)
{
#if 1 /* ports initialized in Port_Init() by Applilet */
      PMCDH = 0x00;           /* set port DH to port mode */
      PMDH = 0x00;            /* set port DH to output */

      PMCDLH = 0x00;          /* set port DL high 8-bits to port mode */
      PMDLH = 0x00;           /* set port DL high 8-bits to output */
#endif
}

/* void led_out_right(unsigned char val) */
```

```
/*          output raw data to right LED */
void led_out_right(unsigned char val)
{
      PDLH = val;
}

/* void led_out_left(unsigned char val) */
/*          output raw data to left LED */
void led_out_left(unsigned char val)
{
      PDH = val;
}

/* void led_dig_right(unsigned char num) */
/*    display number in right LED */
void led_dig_right(unsigned char num)
{
      if (num > 0x0F) {
            led_out_right(LED_PAT_BLANK);
            return;
      }
      led_out_right(dig_tab[num]);
}

/* void led_dig_left(unsigned char num) */
/*    display number in left LED */
void led_dig_left(unsigned char num)
{
      if (num > 0x0F) {
            led_out_left(LED_PAT_BLANK);
            return;
      }
      led_out_left(dig_tab[num]);
}

#if 0 /* removed for space savings */
/* void led_dig(unsigned char num) */
/*          display number as hex digits */
/*          num - number to display */
/*                bits 0-3 in right digit */
/*                bits 4-7 in left digit */
void led_dig(unsigned char num)
{
      led_out_right(dig_tab[num & 0x0F]);
      led_out_left(dig_tab[(num >> 4) & 0x0F]);
}

/* void led_dig_bcd(unsigned char bcdnum) */
/*          display two digits of BCD coded bcdnum */
/*          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 */
void led_dig_bcd(unsigned char bcdnum)
{
```

```
unsigned char tens_dig;

      if (bcdnum > 99) {
            led_out_right(LED_PAT_BLANK); /* display both digits blank */
            led_out_left(LED_PAT_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++;       /* while counting up the tens digit */
      } while (bcdnum >= 10);
      /* now tens_dig has ten's place */
      /* and bcdnum has remainder */
      led_out_right(dig_tab[bcdnum]);
      led_out_left(dig_tab[tens_dig]);
}
#endif
```

3.2 Flash Self-Programming Library Files

The files below are provided in the NEC Flash Self-Programming Library for the uPD70F3318 (V850ES/KJ1+) device.

The file target.h has been modified for the hardware environment of the demonstration program.

The file userfunc.c has been modified to provide interrupt handling for the INTTM000 and INTSR1 interrupts used by the boot program.

3.2.1 df3381.h

```
/***********************************************************************/
/*                          df3318.h                                   */
/*                                                                     */
/*  NEC V850 microcontroller device uPD70F3318                         */
/*                                                                     */
/*    Declarations of I/O Registers                                      */
/*                                                                     */
/*                                                                     */
/*  Copyright (C) NEC Corporation 2004                                 */
/*  This file was created from device file df3318.800       [E1.00a]   */
/*  by DeFiX V1.07b                                                    */
/*                                                                     */
/*  This file is only intended as a sample supplement to NEC tools.   */
/*  Feel free to adapt it to your own needs.                           */
/*  This File is provided 'as is' without warranty of any kind.        */
/*  Neither NEC nor their sales representatives can be held liable     */
/*  of any inconvenience or problem caused by its contents.            */
/***********************************************************************/

#ifndef __uPD70F3318_H
#define __uPD70F3318_H

struct bitf {
      unsigned char bit00:1;
      unsigned char bit01:1;
      unsigned char bit02:1;
      unsigned char bit03:1;
      unsigned char bit04:1;
      unsigned char bit05:1;
      unsigned char bit06:1;
      unsigned char bit07:1;
      unsigned char bit08:1;
      unsigned char bit09:1;
      unsigned char bit10:1;
      unsigned char bit11:1;
      unsigned char bit12:1;
```

```
      unsigned char bit13:1;
      unsigned char bit14:1;
      unsigned char bit15:1;
};


/* I/O register */

#define     PDL         *((volatile unsigned short *)0xfffff004)
#define     PDLL        *((volatile unsigned char  *)0xfffff004)
#define     PDLH        *((volatile unsigned char  *)0xfffff005)
#define     PDH         *((volatile unsigned char  *)0xfffff006)
#define     PCS         *((volatile unsigned char  *)0xfffff008)
#define     PCT         *((volatile unsigned char  *)0xfffff00a)
#define     PCM         *((volatile unsigned char  *)0xfffff00c)
#define     PCD         *((volatile unsigned char  *)0xfffff00e)
#define     PMDL        *((volatile unsigned short *)0xfffff024)
#define     PMDLL       *((volatile unsigned char  *)0xfffff024)
#define     PMDLH       *((volatile unsigned char  *)0xfffff025)
#define     PMDH        *((volatile unsigned char  *)0xfffff026)
#define     PMCS        *((volatile unsigned char  *)0xfffff028)
#define     PMCT        *((volatile unsigned char  *)0xfffff02a)
#define     PMCM        *((volatile unsigned char  *)0xfffff02c)
#define     PMCD        *((volatile unsigned char  *)0xfffff02e)
#define     PMCDL       *((volatile unsigned short *)0xfffff044)
#define     PMCDLL      *((volatile unsigned char  *)0xfffff044)
#define     PMCDLH      *((volatile unsigned char  *)0xfffff045)
#define     PMCDH       *((volatile unsigned char  *)0xfffff046)
#define     PMCCS       *((volatile unsigned char  *)0xfffff048)
#define     PMCCT       *((volatile unsigned char  *)0xfffff04a)
#define     PMCCM       *((volatile unsigned char  *)0xfffff04c)
#define     BSC         *((volatile unsigned short *)0xfffff066)
#define     VSWC        *((volatile unsigned char  *)0xfffff06e)
#define     DSA0L       *((volatile unsigned short *)0xfffff080)
#define     DSA0H       *((volatile unsigned short *)0xfffff082)
#define     DDA0L       *((volatile unsigned short *)0xfffff084)
#define     DDA0H       *((volatile unsigned short *)0xfffff086)
#define     DSA1L       *((volatile unsigned short *)0xfffff088)
#define     DSA1H       *((volatile unsigned short *)0xfffff08a)
#define     DDA1L       *((volatile unsigned short *)0xfffff08c)
#define     DDA1H       *((volatile unsigned short *)0xfffff08e)
#define     DSA2L       *((volatile unsigned short *)0xfffff090)
#define     DSA2H       *((volatile unsigned short *)0xfffff092)
#define     DDA2L       *((volatile unsigned short *)0xfffff094)
#define     DDA2H       *((volatile unsigned short *)0xfffff096)
#define     DSA3L       *((volatile unsigned short *)0xfffff098)
#define     DSA3H       *((volatile unsigned short *)0xfffff09a)
#define     DDA3L       *((volatile unsigned short *)0xfffff09c)
#define     DDA3H       *((volatile unsigned short *)0xfffff09e)
#define     DBC0        *((volatile unsigned short *)0xfffff0c0)
#define     DBC1        *((volatile unsigned short *)0xfffff0c2)
#define     DBC2        *((volatile unsigned short *)0xfffff0c4)
#define     DBC3        *((volatile unsigned short *)0xfffff0c6)
#define     DADC0       *((volatile unsigned short *)0xfffff0d0)
#define     DADC1       *((volatile unsigned short *)0xfffff0d2)
```

```
#define     DADC2       *((volatile unsigned short *)0xfffff0d4)
#define     DADC3       *((volatile unsigned short *)0xfffff0d6)
#define     DCHC0       *((volatile unsigned char  *)0xfffff0e0)
#define     DCHC1       *((volatile unsigned char  *)0xfffff0e2)
#define     DCHC2       *((volatile unsigned char  *)0xfffff0e4)
#define     DCHC3       *((volatile unsigned char  *)0xfffff0e6)
#define     IMR0        *((volatile unsigned short *)0xfffff100)
#define     IMR0L       *((volatile unsigned char  *)0xfffff100)
#define     IMR0H       *((volatile unsigned char  *)0xfffff101)
#define     IMR1        *((volatile unsigned short *)0xfffff102)
#define     IMR1L       *((volatile unsigned char  *)0xfffff102)
#define     IMR1H       *((volatile unsigned char  *)0xfffff103)
#define     IMR2        *((volatile unsigned short *)0xfffff104)
#define     IMR2L       *((volatile unsigned char  *)0xfffff104)
#define     IMR2H       *((volatile unsigned char  *)0xfffff105)
#define     IMR3        *((volatile unsigned short *)0xfffff106)
#define     IMR3L       *((volatile unsigned char  *)0xfffff106)
#define     IMR3H       *((volatile unsigned char  *)0xfffff107)
#define     WDT1IC      *((volatile unsigned char  *)0xfffff110)
#define     PIC0        *((volatile unsigned char  *)0xfffff112)
#define     PIC1        *((volatile unsigned char  *)0xfffff114)
#define     PIC2        *((volatile unsigned char  *)0xfffff116)
#define     PIC3        *((volatile unsigned char  *)0xfffff118)
#define     PIC4        *((volatile unsigned char  *)0xfffff11a)
#define     PIC5        *((volatile unsigned char  *)0xfffff11c)
#define     PIC6        *((volatile unsigned char  *)0xfffff11e)
#define     TM0IC00     *((volatile unsigned char  *)0xfffff120)
#define     TM0IC01     *((volatile unsigned char  *)0xfffff122)
#define     TM0IC10     *((volatile unsigned char  *)0xfffff124)
#define     TM0IC11     *((volatile unsigned char  *)0xfffff126)
#define     TM5IC0      *((volatile unsigned char  *)0xfffff128)
#define     TM5IC1      *((volatile unsigned char  *)0xfffff12a)
#define     CSI0IC0     *((volatile unsigned char  *)0xfffff12c)
#define     CSI0IC1     *((volatile unsigned char  *)0xfffff12e)
#define     SREIC0      *((volatile unsigned char  *)0xfffff130)
#define     SRIC0       *((volatile unsigned char  *)0xfffff132)
#define     STIC0       *((volatile unsigned char  *)0xfffff134)
#define     SREIC1      *((volatile unsigned char  *)0xfffff136)
#define     SRIC1       *((volatile unsigned char  *)0xfffff138)
#define     STIC1       *((volatile unsigned char  *)0xfffff13a)
#define     TMHIC0      *((volatile unsigned char  *)0xfffff13c)
#define     TMHIC1      *((volatile unsigned char  *)0xfffff13e)
#define     CSIAIC0     *((volatile unsigned char  *)0xfffff140)
#define     ADIC        *((volatile unsigned char  *)0xfffff144)
#define     KRIC        *((volatile unsigned char  *)0xfffff146)
#define     WTIIC       *((volatile unsigned char  *)0xfffff148)
#define     WTIC        *((volatile unsigned char  *)0xfffff14a)
#define     BRGIC       *((volatile unsigned char  *)0xfffff14c)
#define     TM0IC20     *((volatile unsigned char  *)0xfffff14e)
#define     TM0IC21     *((volatile unsigned char  *)0xfffff150)
#define     TM0IC30     *((volatile unsigned char  *)0xfffff152)
#define     TM0IC31     *((volatile unsigned char  *)0xfffff154)
#define     CSIAIC1     *((volatile unsigned char  *)0xfffff156)
#define     TM0IC40     *((volatile unsigned char  *)0xfffff158)
#define     TM0IC41     *((volatile unsigned char  *)0xfffff15a)
```

```
#define     TM0IC50     *((volatile unsigned char  *)0xfffff15c)
#define     TM0IC51     *((volatile unsigned char  *)0xfffff15e)
#define     CSI0IC2     *((volatile unsigned char  *)0xfffff160)
#define     SREIC2      *((volatile unsigned char  *)0xfffff162)
#define     SRIC2       *((volatile unsigned char  *)0xfffff164)
#define     STIC2       *((volatile unsigned char  *)0xfffff166)
#define     LVIIC       *((volatile unsigned char  *)0xfffff170)
#define     PIC7        *((volatile unsigned char  *)0xfffff172)
#define     TP0OVIC     *((volatile unsigned char  *)0xfffff174)
#define     TP0CCIC0    *((volatile unsigned char  *)0xfffff176)
#define     TP0CCIC1    *((volatile unsigned char  *)0xfffff178)
#define     DMAIC0      *((volatile unsigned char  *)0xfffff17a)
#define     DMAIC1      *((volatile unsigned char  *)0xfffff17c)
#define     DMAIC2      *((volatile unsigned char  *)0xfffff17e)
#define     DMAIC3      *((volatile unsigned char  *)0xfffff180)
#define     ISPR        *((volatile unsigned char  *)0xfffff1fa)
#define     PRCMD       *((volatile unsigned char  *)0xfffff1fc)
#define     PSC         *((volatile unsigned char  *)0xfffff1fe)
#define     ADM         *((volatile unsigned char  *)0xfffff200)
#define     ADS         *((volatile unsigned char  *)0xfffff201)
#define     PFM         *((volatile unsigned char  *)0xfffff202)
#define     PFT         *((volatile unsigned char  *)0xfffff203)
#define     ADCR        *((volatile unsigned short *)0xfffff204)
#define     ADCRH       *((volatile unsigned char  *)0xfffff205)
#define     DACS0       *((volatile unsigned char  *)0xfffff280)
#define     DACS1       *((volatile unsigned char  *)0xfffff282)
#define     DAM         *((volatile unsigned char  *)0xfffff284)
#define     KRM         *((volatile unsigned char  *)0xfffff300)
#define     SELCNT0     *((volatile unsigned char  *)0xfffff308)
#define     SELCNT1     *((volatile unsigned char  *)0xfffff30a)
#define     NFC         *((volatile unsigned char  *)0xfffff318)
#define     P0          *((volatile unsigned char  *)0xfffff400)
#define     P1          *((volatile unsigned char  *)0xfffff402)
#define     P3          *((volatile unsigned short *)0xfffff406)
#define     P3L         *((volatile unsigned char  *)0xfffff406)
#define     P3H         *((volatile unsigned char  *)0xfffff407)
#define     P4          *((volatile unsigned char  *)0xfffff408)
#define     P5          *((volatile unsigned char  *)0xfffff40a)
#define     P6          *((volatile unsigned short *)0xfffff40c)
#define     P6L         *((volatile unsigned char  *)0xfffff40c)
#define     P6H         *((volatile unsigned char  *)0xfffff40d)
#define     P7          *((volatile unsigned short *)0xfffff40e)
#define     P7L         *((volatile unsigned char  *)0xfffff40e)
#define     P7H         *((volatile unsigned char  *)0xfffff40f)
#define     P8          *((volatile unsigned char  *)0xfffff410)
#define     P9          *((volatile unsigned short *)0xfffff412)
#define     P9L         *((volatile unsigned char  *)0xfffff412)
#define     P9H         *((volatile unsigned char  *)0xfffff413)
#define     PM0         *((volatile unsigned char  *)0xfffff420)
#define     PM1         *((volatile unsigned char  *)0xfffff422)
#define     PM3         *((volatile unsigned short *)0xfffff426)
#define     PM3L        *((volatile unsigned char  *)0xfffff426)
#define     PM3H        *((volatile unsigned char  *)0xfffff427)
#define     PM4         *((volatile unsigned char  *)0xfffff428)
#define     PM5         *((volatile unsigned char  *)0xfffff42a)
```

```
#define     PM6         *((volatile unsigned short *)0xfffff42c)
#define     PM6L        *((volatile unsigned char  *)0xfffff42c)
#define     PM6H        *((volatile unsigned char  *)0xfffff42d)
#define     PM8         *((volatile unsigned char  *)0xfffff430)
#define     PM9         *((volatile unsigned short *)0xfffff432)
#define     PM9L        *((volatile unsigned char  *)0xfffff432)
#define     PM9H        *((volatile unsigned char  *)0xfffff433)
#define     PMC0        *((volatile unsigned char  *)0xfffff440)
#define     PMC3        *((volatile unsigned short *)0xfffff446)
#define     PMC3L       *((volatile unsigned char  *)0xfffff446)
#define     PMC3H       *((volatile unsigned char  *)0xfffff447)
#define     PMC4        *((volatile unsigned char  *)0xfffff448)
#define     PMC5        *((volatile unsigned char  *)0xfffff44a)
#define     PMC6        *((volatile unsigned short *)0xfffff44c)
#define     PMC6L       *((volatile unsigned char  *)0xfffff44c)
#define     PMC6H       *((volatile unsigned char  *)0xfffff44d)
#define     PMC8        *((volatile unsigned char  *)0xfffff450)
#define     PMC9        *((volatile unsigned short *)0xfffff452)
#define     PMC9L       *((volatile unsigned char  *)0xfffff452)
#define     PMC9H       *((volatile unsigned char  *)0xfffff453)
#define     PFC3        *((volatile unsigned char  *)0xfffff466)
#define     PFC4        *((volatile unsigned char  *)0xfffff468)
#define     PFC5        *((volatile unsigned char  *)0xfffff46a)
#define     PFC6H       *((volatile unsigned char  *)0xfffff46d)
#define     PFC8        *((volatile unsigned char  *)0xfffff470)
#define     PFC9        *((volatile unsigned short *)0xfffff472)
#define     PFC9L       *((volatile unsigned char  *)0xfffff472)
#define     PFC9H       *((volatile unsigned char  *)0xfffff473)
#define     DWC0        *((volatile unsigned short *)0xfffff484)
#define     AWC         *((volatile unsigned short *)0xfffff488)
#define     BCC         *((volatile unsigned short *)0xfffff48a)
#define     TMHMD0      *((volatile unsigned char  *)0xfffff580)
#define     TMCYC0      *((volatile unsigned char  *)0xfffff581)
#define     CMP00       *((volatile unsigned char  *)0xfffff582)
#define     CMP01       *((volatile unsigned char  *)0xfffff583)
#define     TMHMD1      *((volatile unsigned char  *)0xfffff590)
#define     TMCYC1      *((volatile unsigned char  *)0xfffff591)
#define     CMP10       *((volatile unsigned char  *)0xfffff592)
#define     CMP11       *((volatile unsigned char  *)0xfffff593)
#define     TP0CTL0     *((volatile unsigned char  *)0xfffff5a0)
#define     TP0CTL1     *((volatile unsigned char  *)0xfffff5a1)
#define     TP0IOC0     *((volatile unsigned char  *)0xfffff5a2)
#define     TP0IOC1     *((volatile unsigned char  *)0xfffff5a3)
#define     TP0IOC2     *((volatile unsigned char  *)0xfffff5a4)
#define     TP0OPT0     *((volatile unsigned char  *)0xfffff5a5)
#define     TP0CCR0     *((volatile unsigned short *)0xfffff5a6)
#define     TP0CCR1     *((volatile unsigned short *)0xfffff5a8)
#define     TP0CNT      *((volatile unsigned short *)0xfffff5aa)
#define     TM5         *((volatile unsigned short *)0xfffff5c0)
#define     TM50        *((volatile unsigned char  *)0xfffff5c0)
#define     TM51        *((volatile unsigned char  *)0xfffff5c1)
#define     CR5         *((volatile unsigned short *)0xfffff5c2)
#define     CR50        *((volatile unsigned char  *)0xfffff5c2)
#define     CR51        *((volatile unsigned char  *)0xfffff5c3)
#define     TCL5        *((volatile unsigned short *)0xfffff5c4)
```

```
#define     TCL50       *((volatile unsigned char  *)0xfffff5c4)
#define     TCL51       *((volatile unsigned char  *)0xfffff5c5)
#define     TMC5        *((volatile unsigned short *)0xfffff5c6)
#define     TMC50       *((volatile unsigned char  *)0xfffff5c6)
#define     TMC51       *((volatile unsigned char  *)0xfffff5c7)
#define     TM00        *((volatile unsigned short *)0xfffff600)
#define     CR000       *((volatile unsigned short *)0xfffff602)
#define     CR001       *((volatile unsigned short *)0xfffff604)
#define     TMC00       *((volatile unsigned char  *)0xfffff606)
#define     PRM00       *((volatile unsigned char  *)0xfffff607)
#define     CRC00       *((volatile unsigned char  *)0xfffff608)
#define     TOC00       *((volatile unsigned char  *)0xfffff609)
#define     TM01        *((volatile unsigned short *)0xfffff610)
#define     CR010       *((volatile unsigned short *)0xfffff612)
#define     CR011       *((volatile unsigned short *)0xfffff614)
#define     TMC01       *((volatile unsigned char  *)0xfffff616)
#define     PRM01       *((volatile unsigned char  *)0xfffff617)
#define     CRC01       *((volatile unsigned char  *)0xfffff618)
#define     TOC01       *((volatile unsigned char  *)0xfffff619)
#define     TM02        *((volatile unsigned short *)0xfffff620)
#define     CR020       *((volatile unsigned short *)0xfffff622)
#define     CR021       *((volatile unsigned short *)0xfffff624)
#define     TMC02       *((volatile unsigned char  *)0xfffff626)
#define     PRM02       *((volatile unsigned char  *)0xfffff627)
#define     CRC02       *((volatile unsigned char  *)0xfffff628)
#define     TOC02       *((volatile unsigned char  *)0xfffff629)
#define     TM03        *((volatile unsigned short *)0xfffff630)
#define     CR030       *((volatile unsigned short *)0xfffff632)
#define     CR031       *((volatile unsigned short *)0xfffff634)
#define     TMC03       *((volatile unsigned char  *)0xfffff636)
#define     PRM03       *((volatile unsigned char  *)0xfffff637)
#define     CRC03       *((volatile unsigned char  *)0xfffff638)
#define     TOC03       *((volatile unsigned char  *)0xfffff639)
#define     TM04        *((volatile unsigned short *)0xfffff640)
#define     CR040       *((volatile unsigned short *)0xfffff642)
#define     CR041       *((volatile unsigned short *)0xfffff644)
#define     TMC04       *((volatile unsigned char  *)0xfffff646)
#define     PRM04       *((volatile unsigned char  *)0xfffff647)
#define     CRC04       *((volatile unsigned char  *)0xfffff648)
#define     TOC04       *((volatile unsigned char  *)0xfffff649)
#define     TM05        *((volatile unsigned short *)0xfffff650)
#define     CR050       *((volatile unsigned short *)0xfffff652)
#define     CR051       *((volatile unsigned short *)0xfffff654)
#define     TMC05       *((volatile unsigned char  *)0xfffff656)
#define     PRM05       *((volatile unsigned char  *)0xfffff657)
#define     CRC05       *((volatile unsigned char  *)0xfffff658)
#define     TOC05       *((volatile unsigned char  *)0xfffff659)
#define     WTM         *((volatile unsigned char  *)0xfffff680)
#define     OSTS        *((volatile unsigned char  *)0xfffff6c0)
#define     WDCS        *((volatile unsigned char  *)0xfffff6c1)
#define     WDTM1       *((volatile unsigned char  *)0xfffff6c2)
#define     WDTM2       *((volatile unsigned char  *)0xfffff6d0)
#define     WDTE        *((volatile unsigned char  *)0xfffff6d1)
#define     RTBL0       *((volatile unsigned char  *)0xfffff6e0)
#define     RTBH0       *((volatile unsigned char  *)0xfffff6e2)
```

```
#define     RTPM0       *((volatile unsigned char  *)0xfffff6e4)
#define     RTPC0       *((volatile unsigned char  *)0xfffff6e5)
#define     RTBL1       *((volatile unsigned char  *)0xfffff6f0)
#define     RTBH1       *((volatile unsigned char  *)0xfffff6f2)
#define     RTPM1       *((volatile unsigned char  *)0xfffff6f4)
#define     RTPC1       *((volatile unsigned char  *)0xfffff6f5)
#define     PFCE3       *((volatile unsigned char  *)0xfffff706)
#define     SYS         *((volatile unsigned char  *)0xfffff802)
#define     PLLCTL      *((volatile unsigned char  *)0xfffff806)
#define     RCM         *((volatile unsigned char  *)0xfffff80c)
#define     DTFR0       *((volatile unsigned char  *)0xfffff810)
#define     DTFR1       *((volatile unsigned char  *)0xfffff812)
#define     DTFR2       *((volatile unsigned char  *)0xfffff814)
#define     DTFR3       *((volatile unsigned char  *)0xfffff816)
#define     PSMR        *((volatile unsigned char  *)0xfffff820)
#define     PCC         *((volatile unsigned char  *)0xfffff828)
#define     CCLS        *((volatile unsigned char  *)0xfffff82e)
#define     CORAD0      *((volatile unsigned long  *)0xfffff840)
#define     CORAD0L     *((volatile unsigned short *)0xfffff840)
#define     CORAD0H     *((volatile unsigned short *)0xfffff842)
#define     CORAD1      *((volatile unsigned long  *)0xfffff844)
#define     CORAD1L     *((volatile unsigned short *)0xfffff844)
#define     CORAD1H     *((volatile unsigned short *)0xfffff846)
#define     CORAD2      *((volatile unsigned long  *)0xfffff848)
#define     CORAD2L     *((volatile unsigned short *)0xfffff848)
#define     CORAD2H     *((volatile unsigned short *)0xfffff84a)
#define     CORAD3      *((volatile unsigned long  *)0xfffff84c)
#define     CORAD3L     *((volatile unsigned short *)0xfffff84c)
#define     CORAD3H     *((volatile unsigned short *)0xfffff84e)
#define     RNZC        *((volatile unsigned char  *)0xfffff860)
#define     CLM         *((volatile unsigned char  *)0xfffff870)
#define     CORCN       *((volatile unsigned char  *)0xfffff880)
#define     RESF        *((volatile unsigned char  *)0xfffff888)
#define     LVIM        *((volatile unsigned char  *)0xfffff890)
#define     LVIS        *((volatile unsigned char  *)0xfffff891)
#define     PRSM        *((volatile unsigned char  *)0xfffff8b0)
#define     PRSCM       *((volatile unsigned char  *)0xfffff8b1)
#define     OCDM        *((volatile unsigned char  *)0xfffff9fc)
#define     PEMU1       *((volatile unsigned char  *)0xfffff9fe)
#define     ASIM0       *((volatile unsigned char  *)0xfffffa00)
#define     RXB0        *((volatile unsigned char  *)0xfffffa02)
#define     ASIS0       *((volatile unsigned char  *)0xfffffa03)
#define     TXB0        *((volatile unsigned char  *)0xfffffa04)
#define     ASIF0       *((volatile unsigned char  *)0xfffffa05)
#define     CKSR0       *((volatile unsigned char  *)0xfffffa06)
#define     BRGC0       *((volatile unsigned char  *)0xfffffa07)
#define     ASICL0      *((volatile unsigned char  *)0xfffffa08)
#define     ASIM1       *((volatile unsigned char  *)0xfffffa10)
#define     RXB1        *((volatile unsigned char  *)0xfffffa12)
#define     ASIS1       *((volatile unsigned char  *)0xfffffa13)
#define     TXB1        *((volatile unsigned char  *)0xfffffa14)
#define     ASIF1       *((volatile unsigned char  *)0xfffffa15)
#define     CKSR1       *((volatile unsigned char  *)0xfffffa16)
#define     BRGC1       *((volatile unsigned char  *)0xfffffa17)
#define     ASIM2       *((volatile unsigned char  *)0xfffffa20)
```

```
#define     RXB2        *((volatile unsigned char  *)0xfffffa22)
#define     ASIS2       *((volatile unsigned char  *)0xfffffa23)
#define     TXB2        *((volatile unsigned char  *)0xfffffa24)
#define     ASIF2       *((volatile unsigned char  *)0xfffffa25)
#define     CKSR2       *((volatile unsigned char  *)0xfffffa26)
#define     BRGC2       *((volatile unsigned char  *)0xfffffa27)
#define     P0NFC       *((volatile unsigned char  *)0xfffffb00)
#define     P1NFC       *((volatile unsigned char  *)0xfffffb04)
#define     INTF0       *((volatile unsigned char  *)0xfffffc00)
#define     INTF3       *((volatile unsigned char  *)0xfffffc06)
#define     INTF9H      *((volatile unsigned char  *)0xfffffc13)
#define     INTR0       *((volatile unsigned char  *)0xfffffc20)
#define     INTR3       *((volatile unsigned char  *)0xfffffc26)
#define     INTR9H      *((volatile unsigned char  *)0xfffffc33)
#define     PU0         *((volatile unsigned char  *)0xfffffc40)
#define     PU1         *((volatile unsigned char  *)0xfffffc42)
#define     PU3         *((volatile unsigned char  *)0xfffffc46)
#define     PU4         *((volatile unsigned char  *)0xfffffc48)
#define     PU5         *((volatile unsigned char  *)0xfffffc4a)
#define     PU6         *((volatile unsigned short *)0xfffffc4c)
#define     PU6L        *((volatile unsigned char  *)0xfffffc4c)
#define     PU6H        *((volatile unsigned char  *)0xfffffc4d)
#define     PU8         *((volatile unsigned char  *)0xfffffc50)
#define     PU9         *((volatile unsigned short *)0xfffffc52)
#define     PU9L        *((volatile unsigned char  *)0xfffffc52)
#define     PU9H        *((volatile unsigned char  *)0xfffffc53)
#define     PF3H        *((volatile unsigned char  *)0xfffffc67)
#define     PF4         *((volatile unsigned char  *)0xfffffc68)
#define     PF5         *((volatile unsigned char  *)0xfffffc6a)
#define     PF6         *((volatile unsigned short *)0xfffffc6c)
#define     PF6L        *((volatile unsigned char  *)0xfffffc6c)
#define     PF6H        *((volatile unsigned char  *)0xfffffc6d)
#define     PF8         *((volatile unsigned char  *)0xfffffc70)
#define     PF9H        *((volatile unsigned char  *)0xfffffc73)
#define     CSIM00      *((volatile unsigned char  *)0xfffffd00)
#define     CSIC0       *((volatile unsigned char  *)0xfffffd01)
#define     SIRB0       *((volatile unsigned short *)0xfffffd02)
#define     SIRB0L      *((volatile unsigned char  *)0xfffffd02)
#define     SOTB0       *((volatile unsigned short *)0xfffffd04)
#define     SOTB0L      *((volatile unsigned char  *)0xfffffd04)
#define     SIRBE0      *((volatile unsigned short *)0xfffffd06)
#define     SIRBE0L     *((volatile unsigned char  *)0xfffffd06)
#define     SOTBF0      *((volatile unsigned short *)0xfffffd08)
#define     SOTBF0L     *((volatile unsigned char  *)0xfffffd08)
#define     SIO00       *((volatile unsigned short *)0xfffffd0a)
#define     SIO00L      *((volatile unsigned char  *)0xfffffd0a)
#define     CSIM01      *((volatile unsigned char  *)0xfffffd10)
#define     CSIC1       *((volatile unsigned char  *)0xfffffd11)
#define     SIRB1       *((volatile unsigned short *)0xfffffd12)
#define     SIRB1L      *((volatile unsigned char  *)0xfffffd12)
#define     SOTB1       *((volatile unsigned short *)0xfffffd14)
#define     SOTB1L      *((volatile unsigned char  *)0xfffffd14)
#define     SIRBE1      *((volatile unsigned short *)0xfffffd16)
#define     SIRBE1L     *((volatile unsigned char  *)0xfffffd16)
#define     SOTBF1      *((volatile unsigned short *)0xfffffd18)
```

```
#define     SOTBF1L     *((volatile unsigned char  *)0xfffffd18)
#define     SIO01       *((volatile unsigned short *)0xfffffd1a)
#define     SIO01L      *((volatile unsigned char  *)0xfffffd1a)
#define     CSIM02      *((volatile unsigned char  *)0xfffffd20)
#define     CSIC2       *((volatile unsigned char  *)0xfffffd21)
#define     SIRB2       *((volatile unsigned short *)0xfffffd22)
#define     SIRB2L      *((volatile unsigned char  *)0xfffffd22)
#define     SOTB2       *((volatile unsigned short *)0xfffffd24)
#define     SOTB2L      *((volatile unsigned char  *)0xfffffd24)
#define     SIRBE2      *((volatile unsigned short *)0xfffffd26)
#define     SIRBE2L     *((volatile unsigned char  *)0xfffffd26)
#define     SOTBF2      *((volatile unsigned short *)0xfffffd28)
#define     SOTBF2L     *((volatile unsigned char  *)0xfffffd28)
#define     SIO02       *((volatile unsigned short *)0xfffffd2a)
#define     SIO02L      *((volatile unsigned char  *)0xfffffd2a)
#define     CSIMA0      *((volatile unsigned char  *)0xfffffd40)
#define     CSIS0       *((volatile unsigned char  *)0xfffffd41)
#define     CSIT0       *((volatile unsigned char  *)0xfffffd42)
#define     BRGCA0      *((volatile unsigned char  *)0xfffffd43)
#define     ADTP0       *((volatile unsigned char  *)0xfffffd44)
#define     ADTI0       *((volatile unsigned char  *)0xfffffd45)
#define     SIOA0       *((volatile unsigned char  *)0xfffffd46)
#define     ADTC0       *((volatile unsigned char  *)0xfffffd47)
#define     CSIMA1      *((volatile unsigned char  *)0xfffffd50)
#define     CSIS1       *((volatile unsigned char  *)0xfffffd51)
#define     CSIT1       *((volatile unsigned char  *)0xfffffd52)
#define     BRGCA1      *((volatile unsigned char  *)0xfffffd53)
#define     ADTP1       *((volatile unsigned char  *)0xfffffd54)
#define     ADTI1       *((volatile unsigned char  *)0xfffffd55)
#define     SIOA1       *((volatile unsigned char  *)0xfffffd56)
#define     ADTC1       *((volatile unsigned char  *)0xfffffd57)
#define     CSIA0B0     *((volatile unsigned short *)0xfffffe00)
#define     CSIA0B0L    *((volatile unsigned char  *)0xfffffe00)
#define     CSIA0B0H    *((volatile unsigned char  *)0xfffffe01)
#define     CSIA0B1     *((volatile unsigned short *)0xfffffe02)
#define     CSIA0B1L    *((volatile unsigned char  *)0xfffffe02)
#define     CSIA0B1H    *((volatile unsigned char  *)0xfffffe03)
#define     CSIA0B2     *((volatile unsigned short *)0xfffffe04)
#define     CSIA0B2L    *((volatile unsigned char  *)0xfffffe04)
#define     CSIA0B2H    *((volatile unsigned char  *)0xfffffe05)
#define     CSIA0B3     *((volatile unsigned short *)0xfffffe06)
#define     CSIA0B3L    *((volatile unsigned char  *)0xfffffe06)
#define     CSIA0B3H    *((volatile unsigned char  *)0xfffffe07)
#define     CSIA0B4     *((volatile unsigned short *)0xfffffe08)
#define     CSIA0B4L    *((volatile unsigned char  *)0xfffffe08)
#define     CSIA0B4H    *((volatile unsigned char  *)0xfffffe09)
#define     CSIA0B5     *((volatile unsigned short *)0xfffffe0a)
#define     CSIA0B5L    *((volatile unsigned char  *)0xfffffe0a)
#define     CSIA0B5H    *((volatile unsigned char  *)0xfffffe0b)
#define     CSIA0B6     *((volatile unsigned short *)0xfffffe0c)
#define     CSIA0B6L    *((volatile unsigned char  *)0xfffffe0c)
#define     CSIA0B6H    *((volatile unsigned char  *)0xfffffe0d)
#define     CSIA0B7     *((volatile unsigned short *)0xfffffe0e)
#define     CSIA0B7L    *((volatile unsigned char  *)0xfffffe0e)
#define     CSIA0B7H    *((volatile unsigned char  *)0xfffffe0f)
```

```
#define     CSIA0B8     *((volatile unsigned short *)0xfffffe10)
#define     CSIA0B8L    *((volatile unsigned char  *)0xfffffe10)
#define     CSIA0B8H    *((volatile unsigned char  *)0xfffffe11)
#define     CSIA0B9     *((volatile unsigned short *)0xfffffe12)
#define     CSIA0B9L    *((volatile unsigned char  *)0xfffffe12)
#define     CSIA0B9H    *((volatile unsigned char  *)0xfffffe13)
#define     CSIA0BA     *((volatile unsigned short *)0xfffffe14)
#define     CSIA0BAL    *((volatile unsigned char  *)0xfffffe14)
#define     CSIA0BAH    *((volatile unsigned char  *)0xfffffe15)
#define     CSIA0BB     *((volatile unsigned short *)0xfffffe16)
#define     CSIA0BBL    *((volatile unsigned char  *)0xfffffe16)
#define     CSIA0BBH    *((volatile unsigned char  *)0xfffffe17)
#define     CSIA0BC     *((volatile unsigned short *)0xfffffe18)
#define     CSIA0BCL    *((volatile unsigned char  *)0xfffffe18)
#define     CSIA0BCH    *((volatile unsigned char  *)0xfffffe19)
#define     CSIA0BD     *((volatile unsigned short *)0xfffffe1a)
#define     CSIA0BDL    *((volatile unsigned char  *)0xfffffe1a)
#define     CSIA0BDH    *((volatile unsigned char  *)0xfffffe1b)
#define     CSIA0BE     *((volatile unsigned short *)0xfffffe1c)
#define     CSIA0BEL    *((volatile unsigned char  *)0xfffffe1c)
#define     CSIA0BEH    *((volatile unsigned char  *)0xfffffe1d)
#define     CSIA0BF     *((volatile unsigned short *)0xfffffe1e)
#define     CSIA0BFL    *((volatile unsigned char  *)0xfffffe1e)
#define     CSIA0BFH    *((volatile unsigned char  *)0xfffffe1f)
#define     CSIA1B0     *((volatile unsigned short *)0xfffffe20)
#define     CSIA1B0L    *((volatile unsigned char  *)0xfffffe20)
#define     CSIA1B0H    *((volatile unsigned char  *)0xfffffe21)
#define     CSIA1B1     *((volatile unsigned short *)0xfffffe22)
#define     CSIA1B1L    *((volatile unsigned char  *)0xfffffe22)
#define     CSIA1B1H    *((volatile unsigned char  *)0xfffffe23)
#define     CSIA1B2     *((volatile unsigned short *)0xfffffe24)
#define     CSIA1B2L    *((volatile unsigned char  *)0xfffffe24)
#define     CSIA1B2H    *((volatile unsigned char  *)0xfffffe25)
#define     CSIA1B3     *((volatile unsigned short *)0xfffffe26)
#define     CSIA1B3L    *((volatile unsigned char  *)0xfffffe26)
#define     CSIA1B3H    *((volatile unsigned char  *)0xfffffe27)
#define     CSIA1B4     *((volatile unsigned short *)0xfffffe28)
#define     CSIA1B4L    *((volatile unsigned char  *)0xfffffe28)
#define     CSIA1B4H    *((volatile unsigned char  *)0xfffffe29)
#define     CSIA1B5     *((volatile unsigned short *)0xfffffe2a)
#define     CSIA1B5L    *((volatile unsigned char  *)0xfffffe2a)
#define     CSIA1B5H    *((volatile unsigned char  *)0xfffffe2b)
#define     CSIA1B6     *((volatile unsigned short *)0xfffffe2c)
#define     CSIA1B6L    *((volatile unsigned char  *)0xfffffe2c)
#define     CSIA1B6H    *((volatile unsigned char  *)0xfffffe2d)
#define     CSIA1B7     *((volatile unsigned short *)0xfffffe2e)
#define     CSIA1B7L    *((volatile unsigned char  *)0xfffffe2e)
#define     CSIA1B7H    *((volatile unsigned char  *)0xfffffe2f)
#define     CSIA1B8     *((volatile unsigned short *)0xfffffe30)
#define     CSIA1B8L    *((volatile unsigned char  *)0xfffffe30)
#define     CSIA1B8H    *((volatile unsigned char  *)0xfffffe31)
#define     CSIA1B9     *((volatile unsigned short *)0xfffffe32)
#define     CSIA1B9L    *((volatile unsigned char  *)0xfffffe32)
#define     CSIA1B9H    *((volatile unsigned char  *)0xfffffe33)
#define     CSIA1BA     *((volatile unsigned short *)0xfffffe34)
```

```
#define     CSIA1BAL    *((volatile unsigned char  *)0xfffffe34)
#define     CSIA1BAH    *((volatile unsigned char  *)0xfffffe35)
#define     CSIA1BB     *((volatile unsigned short *)0xfffffe36)
#define     CSIA1BBL    *((volatile unsigned char  *)0xfffffe36)
#define     CSIA1BBH    *((volatile unsigned char  *)0xfffffe37)
#define     CSIA1BC     *((volatile unsigned short *)0xfffffe38)
#define     CSIA1BCL    *((volatile unsigned char  *)0xfffffe38)
#define     CSIA1BCH    *((volatile unsigned char  *)0xfffffe39)
#define     CSIA1BD     *((volatile unsigned short *)0xfffffe3a)
#define     CSIA1BDL    *((volatile unsigned char  *)0xfffffe3a)
#define     CSIA1BDH    *((volatile unsigned char  *)0xfffffe3b)
#define     CSIA1BE     *((volatile unsigned short *)0xfffffe3c)
#define     CSIA1BEL    *((volatile unsigned char  *)0xfffffe3c)
#define     CSIA1BEH    *((volatile unsigned char  *)0xfffffe3d)
#define     CSIA1BF     *((volatile unsigned short *)0xfffffe3e)
#define     CSIA1BFL    *((volatile unsigned char  *)0xfffffe3e)
#define     CSIA1BFH    *((volatile unsigned char  *)0xfffffe3f)
#define     PUDL        *((volatile unsigned short *)0xffffff44)
#define     PUDLL       *((volatile unsigned char  *)0xffffff44)
#define     PUDLH       *((volatile unsigned char  *)0xffffff45)
#define     PUDH        *((volatile unsigned char  *)0xffffff46)
#define     PUCS        *((volatile unsigned char  *)0xffffff48)
#define     PUCT        *((volatile unsigned char  *)0xffffff4a)
#define     PUCM        *((volatile unsigned char  *)0xffffff4c)
#define     PUCD        *((volatile unsigned char  *)0xffffff4e)
#define     EXIMC       *((volatile unsigned char  *)0xffffffbe)


/* I/O register bit */

#define     _E00        ((volatile struct bitf *)0xfffff0e0)->bit00
#define     _STG0       ((volatile struct bitf *)0xfffff0e0)->bit01
#define     _INIT0      ((volatile struct bitf *)0xfffff0e0)->bit02
#define     _TC0        ((volatile struct bitf *)0xfffff0e0)->bit07

#define     _E11        ((volatile struct bitf *)0xfffff0e2)->bit00
#define     _STG1       ((volatile struct bitf *)0xfffff0e2)->bit01
#define     _INIT1      ((volatile struct bitf *)0xfffff0e2)->bit02
#define     _TC1        ((volatile struct bitf *)0xfffff0e2)->bit07

#define     _E22        ((volatile struct bitf *)0xfffff0e4)->bit00
#define     _STG2       ((volatile struct bitf *)0xfffff0e4)->bit01
#define     _INIT2      ((volatile struct bitf *)0xfffff0e4)->bit02
#define     _TC2        ((volatile struct bitf *)0xfffff0e4)->bit07

#define     _E33        ((volatile struct bitf *)0xfffff0e6)->bit00
#define     _STG3       ((volatile struct bitf *)0xfffff0e6)->bit01
#define     _INIT3      ((volatile struct bitf *)0xfffff0e6)->bit02
#define     _TC3        ((volatile struct bitf *)0xfffff0e6)->bit07

#define     _WDT1MK     ((volatile struct bitf *)0xfffff110)->bit06
#define     _WDT1IF     ((volatile struct bitf *)0xfffff110)->bit07

#define     _PMK0       ((volatile struct bitf *)0xfffff112)->bit06
#define     _PIF0       ((volatile struct bitf *)0xfffff112)->bit07
```

```
#define     _PMK1       ((volatile struct bitf *)0xfffff114)->bit06
#define     _PIF1       ((volatile struct bitf *)0xfffff114)->bit07

#define     _PMK2       ((volatile struct bitf *)0xfffff116)->bit06
#define     _PIF2       ((volatile struct bitf *)0xfffff116)->bit07

#define     _PMK3       ((volatile struct bitf *)0xfffff118)->bit06
#define     _PIF3       ((volatile struct bitf *)0xfffff118)->bit07

#define     _PMK4       ((volatile struct bitf *)0xfffff11a)->bit06
#define     _PIF4       ((volatile struct bitf *)0xfffff11a)->bit07

#define     _PMK5       ((volatile struct bitf *)0xfffff11c)->bit06
#define     _PIF5       ((volatile struct bitf *)0xfffff11c)->bit07

#define     _PMK6       ((volatile struct bitf *)0xfffff11e)->bit06
#define     _PIF6       ((volatile struct bitf *)0xfffff11e)->bit07

#define     _TM0MK00    ((volatile struct bitf *)0xfffff120)->bit06
#define     _TM0IF00    ((volatile struct bitf *)0xfffff120)->bit07

#define     _TM0MK01    ((volatile struct bitf *)0xfffff122)->bit06
#define     _TM0IF01    ((volatile struct bitf *)0xfffff122)->bit07

#define     _TM0MK10    ((volatile struct bitf *)0xfffff124)->bit06
#define     _TM0IF10    ((volatile struct bitf *)0xfffff124)->bit07

#define     _TM0MK11    ((volatile struct bitf *)0xfffff126)->bit06
#define     _TM0IF11    ((volatile struct bitf *)0xfffff126)->bit07

#define     _TM5MK0     ((volatile struct bitf *)0xfffff128)->bit06
#define     _TM5IF0     ((volatile struct bitf *)0xfffff128)->bit07

#define     _TM5MK1     ((volatile struct bitf *)0xfffff12a)->bit06
#define     _TM5IF1     ((volatile struct bitf *)0xfffff12a)->bit07

#define     _CSI0MK0    ((volatile struct bitf *)0xfffff12c)->bit06
#define     _CSI0IF0    ((volatile struct bitf *)0xfffff12c)->bit07

#define     _CSI0MK1    ((volatile struct bitf *)0xfffff12e)->bit06
#define     _CSI0IF1    ((volatile struct bitf *)0xfffff12e)->bit07

#define     _SREMK0     ((volatile struct bitf *)0xfffff130)->bit06
#define     _SREIF0     ((volatile struct bitf *)0xfffff130)->bit07

#define     _SRMK0      ((volatile struct bitf *)0xfffff132)->bit06
#define     _SRIF0      ((volatile struct bitf *)0xfffff132)->bit07

#define     _STMK0      ((volatile struct bitf *)0xfffff134)->bit06
#define     _STIF0      ((volatile struct bitf *)0xfffff134)->bit07

#define     _SREMK1     ((volatile struct bitf *)0xfffff136)->bit06
#define     _SREIF1     ((volatile struct bitf *)0xfffff136)->bit07
```

```
#define     _SRMK1      ((volatile struct bitf *)0xfffff138)->bit06
#define     _SRIF1      ((volatile struct bitf *)0xfffff138)->bit07

#define     _STMK1      ((volatile struct bitf *)0xfffff13a)->bit06
#define     _STIF1      ((volatile struct bitf *)0xfffff13a)->bit07

#define     _TMHMK0     ((volatile struct bitf *)0xfffff13c)->bit06
#define     _TMHIF0     ((volatile struct bitf *)0xfffff13c)->bit07

#define     _TMHMK1     ((volatile struct bitf *)0xfffff13e)->bit06
#define     _TMHIF1     ((volatile struct bitf *)0xfffff13e)->bit07

#define     _CSIAMK0    ((volatile struct bitf *)0xfffff140)->bit06
#define     _CSIAIF0    ((volatile struct bitf *)0xfffff140)->bit07

#define     _ADMK       ((volatile struct bitf *)0xfffff144)->bit06
#define     _ADIF       ((volatile struct bitf *)0xfffff144)->bit07

#define     _KRMK       ((volatile struct bitf *)0xfffff146)->bit06
#define     _KRIF       ((volatile struct bitf *)0xfffff146)->bit07

#define     _WTIMK      ((volatile struct bitf *)0xfffff148)->bit06
#define     _WTIIF      ((volatile struct bitf *)0xfffff148)->bit07

#define     _WTMK       ((volatile struct bitf *)0xfffff14a)->bit06
#define     _WTIF       ((volatile struct bitf *)0xfffff14a)->bit07

#define     _BRGMK      ((volatile struct bitf *)0xfffff14c)->bit06
#define     _BRGIF      ((volatile struct bitf *)0xfffff14c)->bit07

#define     _TM0MK20    ((volatile struct bitf *)0xfffff14e)->bit06
#define     _TM0IF20    ((volatile struct bitf *)0xfffff14e)->bit07

#define     _TM0MK21    ((volatile struct bitf *)0xfffff150)->bit06
#define     _TM0IF21    ((volatile struct bitf *)0xfffff150)->bit07

#define     _TM0MK30    ((volatile struct bitf *)0xfffff152)->bit06
#define     _TM0IF30    ((volatile struct bitf *)0xfffff152)->bit07

#define     _TM0MK31    ((volatile struct bitf *)0xfffff154)->bit06
#define     _TM0IF31    ((volatile struct bitf *)0xfffff154)->bit07

#define     _CSIAMK1    ((volatile struct bitf *)0xfffff156)->bit06
#define     _CSIAIF1    ((volatile struct bitf *)0xfffff156)->bit07

#define     _TM0MK40    ((volatile struct bitf *)0xfffff158)->bit06
#define     _TM0IF40    ((volatile struct bitf *)0xfffff158)->bit07

#define     _TM0MK41    ((volatile struct bitf *)0xfffff15a)->bit06
#define     _TM0IF41    ((volatile struct bitf *)0xfffff15a)->bit07

#define     _TM0MK50    ((volatile struct bitf *)0xfffff15c)->bit06
#define     _TM0IF50    ((volatile struct bitf *)0xfffff15c)->bit07

#define     _TM0MK51    ((volatile struct bitf *)0xfffff15e)->bit06
```

```
#define     _TM0IF51    ((volatile struct bitf *)0xfffff15e)->bit07

#define     _CSI0MK2    ((volatile struct bitf *)0xfffff160)->bit06
#define     _CSI0IF2    ((volatile struct bitf *)0xfffff160)->bit07

#define     _SREMK2     ((volatile struct bitf *)0xfffff162)->bit06
#define     _SREIF2     ((volatile struct bitf *)0xfffff162)->bit07

#define     _SRMK2      ((volatile struct bitf *)0xfffff164)->bit06
#define     _SRIF2      ((volatile struct bitf *)0xfffff164)->bit07

#define     _STMK2      ((volatile struct bitf *)0xfffff166)->bit06
#define     _STIF2      ((volatile struct bitf *)0xfffff166)->bit07

#define     _LVIMK      ((volatile struct bitf *)0xfffff170)->bit06
#define     _LVIIF      ((volatile struct bitf *)0xfffff170)->bit07

#define     _PMK7       ((volatile struct bitf *)0xfffff172)->bit06
#define     _PIF7       ((volatile struct bitf *)0xfffff172)->bit07

#define     _TP0OVMK    ((volatile struct bitf *)0xfffff174)->bit06
#define     _TP0OVIF    ((volatile struct bitf *)0xfffff174)->bit07

#define     _TP0CCMK0   ((volatile struct bitf *)0xfffff176)->bit06
#define     _TP0CCIF0   ((volatile struct bitf *)0xfffff176)->bit07

#define     _TP0CCMK1   ((volatile struct bitf *)0xfffff178)->bit06
#define     _TP0CCIF1   ((volatile struct bitf *)0xfffff178)->bit07

#define     _DMAMK0     ((volatile struct bitf *)0xfffff17a)->bit06
#define     _DMAIF0     ((volatile struct bitf *)0xfffff17a)->bit07

#define     _DMAMK1     ((volatile struct bitf *)0xfffff17c)->bit06
#define     _DMAIF1     ((volatile struct bitf *)0xfffff17c)->bit07

#define     _DMAMK2     ((volatile struct bitf *)0xfffff17e)->bit06
#define     _DMAIF2     ((volatile struct bitf *)0xfffff17e)->bit07

#define     _DMAMK3     ((volatile struct bitf *)0xfffff180)->bit06
#define     _DMAIF3     ((volatile struct bitf *)0xfffff180)->bit07

#define     _ISPR0      ((volatile struct bitf *)0xfffff1fa)->bit00
#define     _ISPR1      ((volatile struct bitf *)0xfffff1fa)->bit01
#define     _ISPR2      ((volatile struct bitf *)0xfffff1fa)->bit02
#define     _ISPR3      ((volatile struct bitf *)0xfffff1fa)->bit03
#define     _ISPR4      ((volatile struct bitf *)0xfffff1fa)->bit04
#define     _ISPR5      ((volatile struct bitf *)0xfffff1fa)->bit05
#define     _ISPR6      ((volatile struct bitf *)0xfffff1fa)->bit06
#define     _ISPR7      ((volatile struct bitf *)0xfffff1fa)->bit07

#define     _STP        ((volatile struct bitf *)0xfffff1fe)->bit01
#define     _INTM       ((volatile struct bitf *)0xfffff1fe)->bit04
#define     _NMI0M      ((volatile struct bitf *)0xfffff1fe)->bit05
#define     _NMI2M      ((volatile struct bitf *)0xfffff1fe)->bit07
```

```
#define     _ADCS2      ((volatile struct bitf *)0xfffff200)->bit00
#define     _ADCS       ((volatile struct bitf *)0xfffff200)->bit07

#define     _PFCM       ((volatile struct bitf *)0xfffff202)->bit06
#define     _PFEN       ((volatile struct bitf *)0xfffff202)->bit07

#define     _DACE0      ((volatile struct bitf *)0xfffff284)->bit00
#define     _DACE1      ((volatile struct bitf *)0xfffff284)->bit02

#define     _ISEL00     ((volatile struct bitf *)0xfffff308)->bit00

#define     _TOEN0      ((volatile struct bitf *)0xfffff580)->bit00
#define     _TOLEV0     ((volatile struct bitf *)0xfffff580)->bit01
#define     _TMHE0      ((volatile struct bitf *)0xfffff580)->bit07

#define     _NRZ0       ((volatile struct bitf *)0xfffff581)->bit00

#define     _TOEN1      ((volatile struct bitf *)0xfffff590)->bit00
#define     _TOLEV1     ((volatile struct bitf *)0xfffff590)->bit01
#define     _TMHE1      ((volatile struct bitf *)0xfffff590)->bit07

#define     _NRZ1       ((volatile struct bitf *)0xfffff591)->bit00

#define     _TP0CE      ((volatile struct bitf *)0xfffff5a0)->bit07

#define     _TP0EEE     ((volatile struct bitf *)0xfffff5a1)->bit05
#define     _TP0EST     ((volatile struct bitf *)0xfffff5a1)->bit06

#define     _TP0OVF     ((volatile struct bitf *)0xfffff5a5)->bit00

#define     _TOE50      ((volatile struct bitf *)0xfffff5c6)->bit00
#define     _LVR50      ((volatile struct bitf *)0xfffff5c6)->bit02
#define     _LVS50      ((volatile struct bitf *)0xfffff5c6)->bit03
#define     _TCE50      ((volatile struct bitf *)0xfffff5c6)->bit07

#define     _TOE51      ((volatile struct bitf *)0xfffff5c7)->bit00
#define     _LVR51      ((volatile struct bitf *)0xfffff5c7)->bit02
#define     _LVS51      ((volatile struct bitf *)0xfffff5c7)->bit03
#define     _TCE51      ((volatile struct bitf *)0xfffff5c7)->bit07

#define     _OVF00      ((volatile struct bitf *)0xfffff606)->bit00

#define     _TOE00      ((volatile struct bitf *)0xfffff609)->bit00
#define     _LVR00      ((volatile struct bitf *)0xfffff609)->bit02
#define     _LVS00      ((volatile struct bitf *)0xfffff609)->bit03
#define     _OSPE00     ((volatile struct bitf *)0xfffff609)->bit05
#define     _OSPT00     ((volatile struct bitf *)0xfffff609)->bit06

#define     _OVF01      ((volatile struct bitf *)0xfffff616)->bit00

#define     _TOE01      ((volatile struct bitf *)0xfffff619)->bit00
#define     _LVR01      ((volatile struct bitf *)0xfffff619)->bit02
#define     _LVS01      ((volatile struct bitf *)0xfffff619)->bit03
#define     _OSPE01     ((volatile struct bitf *)0xfffff619)->bit05
#define     _OSPT01     ((volatile struct bitf *)0xfffff619)->bit06
```

```
#define     _OVF02      ((volatile struct bitf *)0xfffff626)->bit00

#define     _TOE02      ((volatile struct bitf *)0xfffff629)->bit00
#define     _LVR02      ((volatile struct bitf *)0xfffff629)->bit02
#define     _LVS02      ((volatile struct bitf *)0xfffff629)->bit03
#define     _OSPE02     ((volatile struct bitf *)0xfffff629)->bit05
#define     _OSPT02     ((volatile struct bitf *)0xfffff629)->bit06

#define     _OVF03      ((volatile struct bitf *)0xfffff636)->bit00

#define     _TOE03      ((volatile struct bitf *)0xfffff639)->bit00
#define     _LVR03      ((volatile struct bitf *)0xfffff639)->bit02
#define     _LVS03      ((volatile struct bitf *)0xfffff639)->bit03
#define     _OSPE03     ((volatile struct bitf *)0xfffff639)->bit05
#define     _OSPT03     ((volatile struct bitf *)0xfffff639)->bit06

#define     _OVF04      ((volatile struct bitf *)0xfffff646)->bit00

#define     _TOE04      ((volatile struct bitf *)0xfffff649)->bit00
#define     _LVR04      ((volatile struct bitf *)0xfffff649)->bit02
#define     _LVS04      ((volatile struct bitf *)0xfffff649)->bit03
#define     _OSPE04     ((volatile struct bitf *)0xfffff649)->bit05
#define     _OSPT04     ((volatile struct bitf *)0xfffff649)->bit06

#define     _OVF05      ((volatile struct bitf *)0xfffff656)->bit00

#define     _TOE05      ((volatile struct bitf *)0xfffff659)->bit00
#define     _LVR05      ((volatile struct bitf *)0xfffff659)->bit02
#define     _LVS05      ((volatile struct bitf *)0xfffff659)->bit03
#define     _OSPE05     ((volatile struct bitf *)0xfffff659)->bit05
#define     _OSPT05     ((volatile struct bitf *)0xfffff659)->bit06

#define     _WTM0       ((volatile struct bitf *)0xfffff680)->bit00
#define     _WTM1       ((volatile struct bitf *)0xfffff680)->bit01

#define     _RUN1       ((volatile struct bitf *)0xfffff6c2)->bit07

#define     _RTPOE0     ((volatile struct bitf *)0xfffff6e5)->bit07

#define     _RTPOE1     ((volatile struct bitf *)0xfffff6f5)->bit07

#define     _PRERR      ((volatile struct bitf *)0xfffff802)->bit00

#define     _PLLON      ((volatile struct bitf *)0xfffff806)->bit00
#define     _SELPLL     ((volatile struct bitf *)0xfffff806)->bit01
#define     _RTOST0     ((volatile struct bitf *)0xfffff806)->bit02
#define     _RTOST1     ((volatile struct bitf *)0xfffff806)->bit03

#define     _RSTOP      ((volatile struct bitf *)0xfffff80c)->bit00

#define     _DF0        ((volatile struct bitf *)0xfffff810)->bit07

#define     _DF1        ((volatile struct bitf *)0xfffff812)->bit07
```

```
#define     _DF2        ((volatile struct bitf *)0xfffff814)->bit07

#define     _DF3        ((volatile struct bitf *)0xfffff816)->bit07

#define     _PSM        ((volatile struct bitf *)0xfffff820)->bit00

#define     _CK3        ((volatile struct bitf *)0xfffff828)->bit03
#define     _CLS        ((volatile struct bitf *)0xfffff828)->bit04
#define     _MCK        ((volatile struct bitf *)0xfffff828)->bit06

#define     _CLME       ((volatile struct bitf *)0xfffff870)->bit00

#define     _COREN0     ((volatile struct bitf *)0xfffff880)->bit00
#define     _COREN1     ((volatile struct bitf *)0xfffff880)->bit01
#define     _COREN2     ((volatile struct bitf *)0xfffff880)->bit02
#define     _COREN3     ((volatile struct bitf *)0xfffff880)->bit03

#define     _LVIF       ((volatile struct bitf *)0xfffff890)->bit00
#define     _LVIMD      ((volatile struct bitf *)0xfffff890)->bit01
#define     _LVION      ((volatile struct bitf *)0xfffff890)->bit07

#define     _BGCE       ((volatile struct bitf *)0xfffff8b0)->bit04

#define     _OCDM0      ((volatile struct bitf *)0xfffff9fc)->bit00

#define     _RXE0       ((volatile struct bitf *)0xfffffa00)->bit05
#define     _TXE0       ((volatile struct bitf *)0xfffffa00)->bit06
#define     _UARTE0     ((volatile struct bitf *)0xfffffa00)->bit07

#define     _TXSF0      ((volatile struct bitf *)0xfffffa05)->bit00
#define     _TXBF0      ((volatile struct bitf *)0xfffffa05)->bit01

#define     _UDIR0      ((volatile struct bitf *)0xfffffa08)->bit01
#define     _SBRT0      ((volatile struct bitf *)0xfffffa08)->bit06
#define     _SBRF0      ((volatile struct bitf *)0xfffffa08)->bit07

#define     _RXE1       ((volatile struct bitf *)0xfffffa10)->bit05
#define     _TXE1       ((volatile struct bitf *)0xfffffa10)->bit06
#define     _UARTE1     ((volatile struct bitf *)0xfffffa10)->bit07

#define     _TXSF1      ((volatile struct bitf *)0xfffffa15)->bit00
#define     _TXBF1      ((volatile struct bitf *)0xfffffa15)->bit01

#define     _RXE2       ((volatile struct bitf *)0xfffffa20)->bit05
#define     _TXE2       ((volatile struct bitf *)0xfffffa20)->bit06
#define     _UARTE2     ((volatile struct bitf *)0xfffffa20)->bit07

#define     _TXSF2      ((volatile struct bitf *)0xfffffa25)->bit00
#define     _TXBF2      ((volatile struct bitf *)0xfffffa25)->bit01

#define     _CSOT0      ((volatile struct bitf *)0xfffffd00)->bit00
#define     _DIR0       ((volatile struct bitf *)0xfffffd00)->bit04
#define     _TRMD0      ((volatile struct bitf *)0xfffffd00)->bit06
#define     _CSI0E0     ((volatile struct bitf *)0xfffffd00)->bit07
```

```
#define     _CSOT1      ((volatile struct bitf *)0xfffffd10)->bit00
#define     _DIR1       ((volatile struct bitf *)0xfffffd10)->bit04
#define     _TRMD1      ((volatile struct bitf *)0xfffffd10)->bit06
#define     _CSI0E1     ((volatile struct bitf *)0xfffffd10)->bit07

#define     _CSOT2      ((volatile struct bitf *)0xfffffd20)->bit00
#define     _DIR2       ((volatile struct bitf *)0xfffffd20)->bit04
#define     _TRMD2      ((volatile struct bitf *)0xfffffd20)->bit06
#define     _CSI0E2     ((volatile struct bitf *)0xfffffd20)->bit07

#define     _DIRA0      ((volatile struct bitf *)0xfffffd40)->bit01
#define     _RXEA0      ((volatile struct bitf *)0xfffffd40)->bit02
#define     _TXEA0      ((volatile struct bitf *)0xfffffd40)->bit03
#define     _CSIAE0     ((volatile struct bitf *)0xfffffd40)->bit07

#define     _ATSTA0     ((volatile struct bitf *)0xfffffd42)->bit00
#define     _ATSTP0     ((volatile struct bitf *)0xfffffd42)->bit01

#define     _DIRA1      ((volatile struct bitf *)0xfffffd50)->bit01
#define     _RXEA1      ((volatile struct bitf *)0xfffffd50)->bit02
#define     _TXEA1      ((volatile struct bitf *)0xfffffd50)->bit03
#define     _CSIAE1     ((volatile struct bitf *)0xfffffd50)->bit07

#define     _ATSTA1     ((volatile struct bitf *)0xfffffd52)->bit00
#define     _ATSTP1     ((volatile struct bitf *)0xfffffd52)->bit01


#endif /* __uPD70F3318_H */
```

3.2.2 Nec_types.h

```
/*============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLib.h
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
/*============================================================================
*/
/*                                  COPYRIGHT
*/
/*============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*============================================================================
*/
/* Purpose:
*/
/*                  This file contains the coding-standard type definitions.
*/
```

```
/*=============================================================================
*/
/* Environment:
*/
/*               Devices:       ALL
*/
/*               IDE's:         ALL
*/
/*=============================================================================
*/

#ifndef _NEC_TYPES_H
#define _NEC_TYPES_H

/*=============================================================================
*/
/* Standard definitions
*/
/*=============================================================================
*/
#ifndef NULL
    #define NULL            0x00000000
#endif

typedef enum { false=0, true=1 } bit;

/*=============================================================================
*/
/* NEC Standard variable types
*/
/*=============================================================================
*/

typedef unsigned char   u08;
typedef unsigned short  u16;
typedef unsigned long   u32;

typedef signed char     s08;
typedef signed short    s16;
typedef signed long     s32;

typedef float           flt;
typedef double          dbl;
typedef long double     lgd;


#endif
```

3.2.3 SelfLib.h

```
/*=============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
```

```
/* Module       = SelfLib.h
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
/*============================================================================
*/
/*                                  COPYRIGHT
*/
/*============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*============================================================================
*/
/* Purpose:
*/
/*              Header file which includes the prototypes of the
*/
/*              selfprogramming library functions and the return parameter
*/
/*              defines
*/
/*============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*============================================================================
*/

#ifndef SELFLIB_H
#define SELFLIB_H

/*============================================================================
*/
/* Global compiler definition
*/
/*============================================================================
*/
#define GHS     1
#define IAR     2
#define NEC     3

#if defined (__IAR_SYSTEMS_ASM__)
    #define SL_COMPILER IAR
#elif defined (__IAR_SYSTEMS_ICC__)
    #define SL_COMPILER IAR
#elif defined(__v850__)
```

```
    #define SL_COMPILER NEC
#else /*GHS */
    #define SL_COMPILER GHS
#endif



/*============================================================================
*/
/* Definition used for the SelfLib_Init() function parameters
*/
/*============================================================================
*/
/* Section copy */
#define SELFLIB_COPY                1
#define SELFLIB_DONT_COPY           2

typedef enum { DONT_COPY =  SELFLIB_DONT_COPY,
               COPY =       SELFLIB_COPY
             } SECTION_COPY;

/*============================================================================
*/
/* Self-programming library function headers
*/
/*============================================================================
*/
u32  SelfLib_Init(                  void            *destAddSelfLib,
                                    u32             frequency,
                                    SECTION_COPY    copy );
u32  SelfLib_LibVersion(            void );
u32 SelfLib_FctNewAddress(          void            *addFct,
                                    void            *destAddSelfLib );
u32  SelfLib_BlankCheck(            u32             BlockNoStart,
                                    u32             BlockNoEnd );
u32  SelfLib_Erase(                 u32             BlockNoStart,
                                    u32             BlockNoEnd );
u32  SelfLib_Write(                 void            *AddSrc,
                                    void            *AddDest,
                                    u32             Length );
u32  SelfLib_IVerify(               u32             BlockNoStart,
                                    u32             BlockNoEnd );
u32 SelfLib_Read(                   void            *readAddress,
                                    void            *destAddress,
                                    u32             length );
u32  SelfLib_ModeCheck(             void );
u32  SelfLib_SetSecFlags(           u32             SecFlags );
u32  SelfLib_BootSwap(              void );

u32  SelfLib_GetInfo_Device(        void );
u32  SelfLib_GetInfo_BlockCnt(      void );
u32  SelfLib_GetInfo_SecFlags(      void );
u32  SelfLib_GetInfo_BootSwap(      void );
u32  SelfLib_GetInfo_BlockEndAdd(   u32             BlockNo );
```

```
#if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
    u32  SelfLib_StatusCheck        (   void );
    u32  SelfLib_FlashEnv_Activate  (   void );
    void SelfLib_FlashEnv_Deactivate(   void );
#endif

/*============================================================================
*/
/* Error Codes returned by SelfLib functions
*/
/*============================================================================
*/

/* Selfprogramming function terminated successfully */
#define SELFLIB_OK                  0x00

/* Unsupported operation */
#define SELFLIB_ERR_UNSUPPORTED     0x01

/* A parameter of the SelfLib function call was wrong or the command is not
*/
/* supported by the device */
#define SELFLIB_ERR_PARAMETER       0x05

/* Flash protected against the operation */
#define SELFLIB_ERR_PROTECTION      0x10

/* No Vdd detected at the FLMD0 pin */
#define SELFLIB_ERR_FLMD0           0x18

/* Different Flash processing errors */
#define SELFLIB_ERR_FLASHPROC0      0x1A
#define SELFLIB_ERR_FLASHPROC1      0x1B
#define SELFLIB_ERR_FLASHPROC2      0x1C

/* SelfLib Status check - busy */
#define SELFLIB_BUSY                0xFF


#endif
```

3.2.4 SelfLibSpecific.h

```
/*============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfSpecific.h
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
```

```
/*=============================================================================
*/
/*                                  COPYRIGHT
*/
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              Header file defines the API of the selflib
*/
/*              SELFLIB_API_BACKCOMP defines a UC2 backward compatible API
*/
/*              SELFLIB_DEFAULT defines the new API with extended parameters
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

#ifndef SELFLIBSPECIFIC_H
#define SELFLIBSPECIFIC_H

/*-----------------------------------------------------------------------------
*/
/*------------- DO NOT CHANGE !!! ---------------------------------------------
*/
/*-----------------------------------------------------------------------------
*/

    /*-------------------------------------------------------------------------
*/
    /*   Basic defines
*/
    /*-------------------------------------------------------------------------
*/
    /* Device Specific library extensions */
    #define SELFLIB_DEFAULT             1
    #define SELFLIB_CZ6                 2
    #define SELFLIB_KX1_V1              3
    #define SELFLIB_PHOENIX_F           4

    /* Status check handling */
    #define STATUS_CHECK_FW             1
```

```
    #define STATUS_CHECK_USER           2
    #define STATUS_CHECK_SELFLIB        3

    /*-------------------------------------------------------------------------
*/
    /*   Currently just status check handling by the firmware is
*/
    /*   implemented. So do not change
*/
    /*-------------------------------------------------------------------------
*/
    #define SELFLIB_STATUS_CHECK        STATUS_CHECK_FW


/*-------------------------------------------------------------------------
*/
/*------------- USER CHANGABLE SECTION ----------------------------------
*/
/*-------------------------------------------------------------------------
*/

    /*-------------------------------------------------------------------------
*/
    /*   Device specific library compile options
*/
    /*   Please refer to the Self-Programming app.-note for single voltage
*/
    /*   SST Flash and to the device reference list for the correct settings
*/
    /*   NOTE:
*/
    /*   Wrong setting results in wrong or non functioning Self-Programming
*/
    /*-------------------------------------------------------------------------
*/
    #define SELFLIB_SPECIFIC_OPTIONS    SELFLIB_KX1_V1

#endif
```

3.2.5 Selfprog.h

```
/*=========================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = selfprog.h
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
/*=========================================================================
*/
/*                                  COPYRIGHT
*/
```

```
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              Header for selfprogramming functions
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

#ifndef SELFPROG_H
#define SELFPROG_H


/*=============================================================================
*/
/* Other header includes
*/
/*=============================================================================
*/
#include "nec_types.h"
#include "selflib.h"
#include "target.h"

#if SL_COMPILER == IAR
    __intrinsic void    __disable_interrupt(void);
    __intrinsic void    __enable_interrupt(void);

    #define __DI()      __disable_interrupt()
    #define __EI()      __enable_interrupt()
#endif

/*=============================================================================
*/
/* Function headers
*/
/*=============================================================================
*/
extern u32 SelfProg_Control    ( void );


#endif
```

3.2.6 Target.h

```
/*=============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = target.h
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
/*=============================================================================
*/
/*                                  COPYRIGHT
*/
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              Header for basic I/O functions
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

/* NOTE: 8/08/2006 - For use in NEC Flash Self-Programming Application Note
*/
/* This file modified for M-Station + M-V850ES-KJ1 hardware          */

#ifndef TARGET_H
#define TARGET_H

/*=============================================================================
*/
/* Include the correct device file
*/
/*=============================================================================
*/

#include "df3318.h"
```

```
/*============================================================================
*/
/* Set-up the CPU, frquency, watchdog, ...
*/
/*============================================================================
*/

#if 0 /* for Application Note, this is not used */
#define SET_FCPU_AND_FPERIPHERALS
\
        PRCMD = 0x00;           /* Prepare to write specific register */
\
        PCC   = 0x00;           /* Select cpu clock fxx = foscl       */
\
        PLLCTL= 0x03;           /* Switch to PLL clock                */
\
        WDTM2 = 0x1F;           /* Stop WDT                           */
\
                                /* !!! Option byte 0x7a set to 0x00!!! */
\
        VSWC  = 0x01;           /* Set NPB speed */
\
        P0    = 0;              /* added for M-Station */
\
            PM0   = 0x02;                /* Set P01 = output, low, for FLMD0
drive */
#endif

/*============================================================================
*/
/* Activate clockout
*/
/*============================================================================
*/

#if 0 /* for Application Note, this is not used */
#define SET_CLOCKOUT_ON
\
        PMCM  = 0xFD;           /* Switch PCM1 to output */
\
        PMCCM = 0x02;           /* Use CLKOUT output     */
#endif

/*============================================================================
*/

/* set Vdd for the FLMD0 pin
*/
/* *** modified for M-V850ES-KJ1 and M-Station II - use JP2 P01->FLMD0
*/
/*============================================================================
*/

#if 0 /* original code */
```

```
#define SET_VOLTAGE_FLMD0(a)
while(SELFLIB_ERR_FLMD0==SelfLib_ModeCheck());
#else
      /* for M-Station and M-V850ES-KJ1, P01 controls FLMD0 */
#define SET_MODE_FLMD0()                    \
            PM0 = PM0 & 0xFD; /* set P01 in output mode */
#define SET_FLMD0_OFF()       \
            P0 = P0 & 0xFD;         /* set P01 low to turn off FLMD0 */
#define     SET_FLMD0_ON()          \
            P0 = P0 | 0x02;         /* set P01 high to turn on FLMD0 */
#endif

/*=============================================================================
*/
/* User function section
*/
/*=============================================================================
*/

/* Definition required for the interrupt vector table entry */
#include "selflib.h"
#if SL_COMPILER == NEC
      #ifdef _DECLARE_VARS
#if 0 /* modified for Flash Self-Program App Note program */
          #pragma interrupt INTTP0CC0 userIntHdr
#else /* not necessary to define specific interrupt vector f0r userIntHdr()
*/
#endif
      #endif
#else
      #define USERFUNC_INT_HDR    0x3B0
#endif

#if 0 /* for Application Note, this is not used */
#define USER_FUNC_OUTPUT_INIT
\
        /* port DL to output for port toggling */
\
        ( (volatile struct bitf *)(&PDL) )->bit15 = 0;
\
        ( (volatile struct bitf *)(&PMDL) )->bit15 = 0;
#endif

#if 0 /* for Application Note, this is not used */
#define USER_FUNC_INTSRC_INIT
\
        /* timer P0 setup */
\
        TP0CTL0 = 0x85;         /* Fxx/32. 1.6us/tick */
\
        TP0CCR0 = 0x06;
\
        TP0CTL1 = 0x00;         /* Interval timer */
\
```

```
        TP0OPT0 = 0x00;         /* compare */
\

\
        /* timer P0 interrupt enable */
\
        TP0CCIC0 = 0x00;        /* enable interrupts */
#endif

#if 0 /* for Application Note, this is not used */
#define USER_FUNC_OUTPUT
\
        /* port DL to output for port toggling */
\
        ( (volatile struct bitf *)(&PDL) )->bit15
\
            = ~( (volatile struct bitf *)(&PDL) )->bit15;
#endif

/*=============================================================================
*/
/*  Defines to setup the execution environment
*/
/*  !!! Please refer to the application note Self-Programming for single
*/
/*      voltage Flash !!!
*/
/*=============================================================================
*/

#define SELFLIB_RAMADD      0xFFFFB000      /* SelfLib destination add. */

#define FREQUENCY           20000000        /* CPU Operation frequency */
#define SELFLIB_SECTIONS    SELFLIB_COPY    /* Copy SelfLib Sections */




#endif
```

3.2.7 SelfLibAsm.s

```
#
===============================================================================
#  Project      = Selfprogramming Library Single Voltage Flash
#  Module       = SelfLibAsm.s85
#  Version      = 1.02
#  Date         = 21.03.2005
#
===============================================================================
#                                   COPYRIGHT
#
===============================================================================
#  Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
```

```
#
================================================================================
#  Purpose:
#               - Misc basic assembler functions required for the selfLib
#               - Device firmware interface (Trap entry )
#
================================================================================
#  Environment:
#               Devices:        V850E(S) Single Voltage SST Flash
#               IDE's:          GHS V3.5.1, GHS V4.0.5,
#                               IAR V2.3, IAR V3.10, NEC V2.61
#
================================================================================
      .set    PSW_NMI_INT_DISABLE, 0x00A0  --NMI Flag=1, ID=1
      .set    PSW_REGISTER_NUMBER, 0x05
      .set    TRAP_SELFPROGRAMMING, 0x1f


#
================================================================================
# Depending on the device either SelfLib_BasFct_WaitRo or
# SelfLib_BasFct_Wait is required. The other function may be deleted
# Please refer to the module SelfLibBrom.c, function
# SelfLib_BasFct_FlashEnv_Deactivate
#
================================================================================

    #
================================================================================
    #  Function name:   SelfLib_BasFct_WaitRo
    #  IN:              -
    #  OUT:             -
    #  Description:     WAIT a fixed time for Flash activation. Timing is
    #                   based on an internal ring oscillator. So no
    #                   operation frequency dependancy
    #
================================================================================
    .set    FT1, 0xfffff8da
    .set    FT2, 0xfffff8c2
    .set    FT3, 0xfffff8c3
    .set    FT4, 0xfffff8db

    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_WaitRo

    _SelfLib_BasFct_WaitRo:
        ld.b    FT4[zero], r6
        ori     0x08, r6, r6
        st.b    r6, FT4[zero]
        ld.b    FT4[zero], r6
        ori     0x60, r6, r6
        st.b    r6, FT4[zero]
        st.b    zero, FT3[zero]
          movea   0x80, zero, r6
        st.b    r6, FT2[zero]
```

```
          movea   0x32, zero, r6
        st.b    r6, FT1[zero]
        ld.b    FT1[zero], r6
        ori     0x40, r6, r6
        st.b    r6, FT1[zero]
        ld.b    FT1[zero], r6
        ori     0x80, r6, r6
        st.b    r6, FT1[zero]
    L1: ld.b    FT1[zero], r6
        andi    0x20, r6, r6
        addi    0xFFFFFFE0, r6, r7
        bz      L1
        ld.b    FT1[zero], r6
        andi    0xbf, r6, r6
        st.b    r6, FT1[zero]
        ld.b    FT1[zero], r6
        andi    0x7f, r6, r6
        st.b    r6, FT1[zero]
        ld.b    FT4[zero], r6
        andi    0xf7, r6, r6
        st.b    r6, FT4[zero]
        ld.b    FT4[zero], r6
        andi     0x9f, r6, r6
        st.b    r6, FT4[zero]

        jmp     [lp]

    #
================================================================================
    #  Function name:   SelfLib_BasFct_Wait
    #  IN:              time (us)           (r6)
    #                   frequency (Hz)      (r7)
    #  OUT:             -
    #  Describtion:     WAIT function
    #                   clk = t(us)*f(MHz)
    #                   clk/loop = 8
    #
================================================================================
    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_Wait

    _SelfLib_BasFct_Wait:
        shr     13, r7          -- r7 /= (8*1000) to be in 32bit range
        mulu    r6, r7, zero    -- r7  = r7 * time
        shr     10, r7          -- r7 /= (8* (1000/8))

    L2: nop     -- 8clk/loop on V850ES
        nop
        addi    -1, r7, r7
        nop
        nop
        nop
        nop
        bgt     L2
```

```
        jmp     [lp]
#
================================================================================


#
================================================================================
#  Function name:   SelfLib_BasFct_IntNmiDisable
#  IN:              -
#  OUT:             old PSW
#  Describtion:     disable INT and NMI and return original PSW value for
#                   later restore
#
================================================================================
    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_IntNmiDisable

    _SelfLib_BasFct_IntNmiDisable:

        #  get current PSW into return register
        stsr    PSW_REGISTER_NUMBER, r10

        #  Disable NMI and Interrupt
        movea   PSW_NMI_INT_DISABLE, zero, r6
        ldsr    r6, PSW_REGISTER_NUMBER

        jmp     [lp]

#
================================================================================
#  Function name:   SelfLib_BasFct_RestorePSW
#  IN:              old saved PSW (r6)
#  OUT:             -
#  Describtion:     restore PSW using the passed parameter
#
================================================================================
    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_RestorePSW

    _SelfLib_BasFct_RestorePSW:

        #  restore old PSW
        ldsr    r6, PSW_REGISTER_NUMBER
        jmp     [lp]


#
================================================================================
#  Function name:   SelfLib_BasFct_WriteSecReg
#  IN:              Dest. Register      (r6)
#                   Security Register   (r7)
#                   Value               (r8)
#  OUT:             -
#  Describtion:     Write sfr, that is secured by a special sequence
```

```
#
================================================================================
    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_WriteSecReg

    _SelfLib_BasFct_WriteSecReg:
        add     -8, sp
        st.w    r20, 0[sp]
        st.w    r21, 4[sp]

        not     r8, r20
        movea   0xa5, zero, r21

        st.b    r21, 0[r7]
        st.b    r8, 0[r6]
        st.b    r20, 0[r6]
        st.b    r8, 0[r6]

        ld.w    0[sp], r20
        ld.w    4[sp], r21
        add     8, sp

        jmp     [lp]


#
================================================================================
#  Function name:   SelfLib_BasFct_TrapEntry
#  IN:              Parameter 1~4 passed to the firmware in registers r6~r9
#                   Parameter 5 passed to the firmware in ep
#  OUT:             Result of the trap function (no result in case of
#                   ..._void
#  Describtion:     Device firmware interface, using a TRAP
#
================================================================================
    .section "SelfLib_ToRam.text", text
    .globl _SelfLib_BasFct_TrapEntry
    .globl _SelfLib_BasFct_TrapEntry_void

    _SelfLib_BasFct_TrapEntry:
    _SelfLib_BasFct_TrapEntry_void:

        addi    -16, sp, sp
        st.w    r2,   0[sp]
        st.w    r5,   4[sp]
        st.w    ep,   8[sp]
        st.w    lp,   12[sp]

        # call values already in r6~r9

        # Set ep new, 5th parameter taken from the stack
        ld.w    16[sp], ep

        # Call the BROM functions with parameter in r6,r7,r8,r9
        trap    TRAP_SELFPROGRAMMING
```

```
        # Restore regs from stack
        ld.w    0[sp],  r2
        ld.w    4[sp],  r5
        ld.w    8[sp], ep
        ld.w    12[sp], lp
        addi    16, sp, sp

        jmp     [lp]



#
================================================================================
#  Function name:   SelfLib_BasFct_ExeOffset
#  IN:              -
#  OUT:             Execution offset in Bytes
#  Describtion:     Calculate offset between SelfLib execution address and
#                   link address
#
================================================================================
    .section "SelfLib_Rom.text", text
    .globl _SelfLib_BasFct_ExeOffset

_SelfLib_BasFct_ExeOffset:
        jarl    L0, r10                  -- store exe address of 'L0' to r10
      L0:   movea       lo(L0), zero, r6  -- store link address of 'L0' to r6
            movhi hi(L0), r6, r6
            sub         r6, r10                    -- build offset and store
it to r10

        jmp     [lp]



#
================================================================================
#  Function name:   SelfLib_BasFct_SectionAdd_...
#  IN:              -
#  OUT:             secStart:   Start address of the section
#                   secSize:    Size of the section
#  Describtion:     The initialisation routine requires section addresses.
#                   Getting this information is different on all compilers.
#
================================================================================
    .section "SelfLib_Rom.text", text
    .globl _SelfLib_BasFct_SectionAdd_ToRamUsrInt
    .globl _SelfLib_BasFct_SectionAdd_ToRamUsr
    .globl _SelfLib_BasFct_SectionAdd_ToRam
    .globl _SelfLib_BasFct_SectionAdd_RomOrRam

    # Parameter: u32 *secStart, u32 *secSize

    _SelfLib_BasFct_SectionAdd_ToRamUsrInt:
        #  Generate section start
```

```
        movhi   hi1 (#__sSelfLib_ToRamUsrInt.text), zero, r10
        movea   lo  (#__sSelfLib_ToRamUsrInt.text), r10, r10
        #  Generate section size
        movhi   hi1 (#__eSelfLib_ToRamUsrInt.text), zero, r11
        movea   lo  (#__eSelfLib_ToRamUsrInt.text), r11, r11
        br SelfLib_BasFct_SectionAdd_End

    _SelfLib_BasFct_SectionAdd_ToRamUsr:
        #  Generate section start
        movhi   hi1 (#__sSelfLib_ToRamUsr.text), zero, r10
        movea   lo  (#__sSelfLib_ToRamUsr.text), r10, r10
        #  Generate section size
        movhi   hi1 (#__eSelfLib_ToRamUsr.text), zero, r11
        movea   lo  (#__eSelfLib_ToRamUsr.text), r11, r11
        br SelfLib_BasFct_SectionAdd_End

    _SelfLib_BasFct_SectionAdd_ToRam:
        #  Generate section start
        movhi   hi1 (#__sSelfLib_ToRam.text), zero, r10
        movea   lo  (#__sSelfLib_ToRam.text), r10, r10
        #  Generate section size
        movhi   hi1 (#__eSelfLib_ToRam.text), zero, r11
        movea   lo  (#__eSelfLib_ToRam.text), r11, r11
        br SelfLib_BasFct_SectionAdd_End

    _SelfLib_BasFct_SectionAdd_RomOrRam:
        #  Generate section start
        movhi   hi1 (#__sSelfLib_RomOrRam.text), zero, r10
        movea   lo  (#__sSelfLib_RomOrRam.text), r10, r10
        #  Generate section size
        movhi   hi1 (#__eSelfLib_RomOrRam.text), zero, r11
        movea   lo  (#__eSelfLib_RomOrRam.text), r11, r11
        br SelfLib_BasFct_SectionAdd_End

    SelfLib_BasFct_SectionAdd_End:
        st.w    r10, 0[r6]
        sub     r10, r11
        st.w    r11, 0[r7]

        jmp     [lp]
```

3.2.8 SelfLibBrom.c

```
/*============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibBrom.h
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
```

```
/*=============================================================================
*/
/*                                  COPYRIGHT
*/
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              File providing the low level interface to the assembly
*/
/*              functions
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

#include "SelfLibGlobal.h"

/*=============================================================================
*/
/* Function name:   SelfLib_BasFct_Env_Init
*/
/* IN:              -
*/
/* OUT:             -
*/
/* Describtion:     Initialises the firmware environment
*/
/*                  -- Technology specific --
*/
/*=============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

void SelfLib_BasFct_Env_Init( void )
{
```

```
#if SELFLIB_SPECIFIC_OPTIONS == SELFLIB_KX1_V1
    TT_OUT2BYTESBROM(FNO_ENV_INIT,1);
    SelfLib_BasFct_TrapEntry_void( FNO_ENV_INIT,
                                   u32SelfLib_frequency,
                                   0,
                                   0,
                                   sysData );
    TT_OUT2BYTESBROM(0,0);

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_F
    /* Do nothing */

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_FS
    /* Do nothing */

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_CZ6
    TT_OUT2BYTESBROM(FNO_ENV_INIT,1);
    SelfLib_BasFct_TrapEntry_void( FNO_ENV_INIT,
                                   u32SelfLib_frequency,
                                   0,
                                   0,
                                   sysData  );
    TT_OUT2BYTESBROM(0,0);

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_DEFAULT
    /* Do nothing */

#endif
}


/*============================================================================
*/
/* Function name:   SelfLib_BasFct_FlashEnv_Activate
*/
/* IN:              -
*/
/* OUT:             -
*/
/* Describtion:     Activates the Flash environment and the device firmware
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

void SelfLib_BasFct_FlashEnv_Activate( void )
{
    u08 ioReg;
    u32 pswTmp;
```

```
    /* Activate BROM firmware */
    pswTmp = SelfLib_BasFct_IntNmiDisable();

    ioReg = FLPMC;
    ioReg |= FLSPM_MASK;
    SelfLib_BasFct_WriteSecReg ( (u08 *)(&FLPMC), (u08 *)(&PFCMD), ioReg );

    SelfLib_BasFct_RestorePSW ( pswTmp );

    /* Initialise environment */
    SelfLib_BasFct_Env_Init();
}


/*=============================================================================
*/
/* Function name:   SelfLib_BasFct_FlashEnv_Deactivate
*/
/* IN:              -
*/
/* OUT:             -
*/
/* Describtion:     Deactivates the Flash environment and the device
firmware*/
/*                  Waits the deactivation time for Flash to come up again
*/
/*=============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

void SelfLib_BasFct_FlashEnv_Deactivate( void )
{
    u08 ioReg;
    u32 pswTmp;

    /* No interrupt or NMI execution during Deinit environment, as neither */
    /* BROM nor Flash is available that time */
    pswTmp = SelfLib_BasFct_IntNmiDisable();

    ioReg = FLPMC;
    ioReg &= ( ~FLSPM_MASK );
    SelfLib_BasFct_WriteSecReg ( (u08 *)(&FLPMC), (u08 *)(&PFCMD), ioReg );

    #if SELFLIB_SPECIFIC_OPTIONS==SELFLIB_PHOENIX_F
        SelfLib_BasFct_WaitRo();
    #elif SELFLIB_SPECIFIC_OPTIONS==SELFLIB_PHOENIX_FS
        SelfLib_BasFct_WaitRo();
    #else
```

```
        SelfLib_BasFct_Wait( TIME_US_ACTIVATE_FLASH, u32SelfLib_frequency );
    #endif

    SelfLib_BasFct_RestorePSW ( pswTmp );
}

/*============================================================================
*/
/* Function name:   SelfLib_BasFct_BromFlashFunc
*/
/* IN:              functionNo:             Number of the function to be
*/
/*                                          executed by the firmware
*/
/*                  param1                  1st Parameter for the firmware
*/
/*                                          function
*/
/*                  param2                  2nd Parameter for the firmware
*/
/*                                          function
*/
/*                  param3                  3rd Parameter for the firmware
*/
/*                                          function
*/
/*                  statCheck               Defines, if a firmware operation
*/
/*                                          has only be initiated and status
*/
/*                                          check is after execution of the
*/
/*                                          firmware function
*/
/* OUT:                                     Errors according to the firmware
*/
/*                                          functions are:
*/
/*                  SELFLIB_OK:             function terminated sucessfully
*/
/*                  SELFLIB_ERR_PARAMETER:  wrong parameters
*/
/*                  SELFLIB_ERR_PROTECTION: operation prohibited
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/* Describtion:     Firmware interface of the selfprogramming library. The
*/
/*                  Flash programming firmware is called and control is
*/
/*                  handed over to the firmware. Return with the result
*/
/*                  after operation completion
*/
```

```
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

u32 SelfLib_BasFct_BromFlashFunc (  u32                         functionNo,
                                    u32                         param1,
                                    u32                         param2,
                                    u32                         param3,
                                    SELFLIB_STATCHECK           statCheck )
{
    u32 ret;

    ret = SELFLIB_OK;

#ifdef SELFLIB_POLLEMU
    if( functionNo != FNO_STATUS_CHECK )
    {
#endif

    TF_CRLF;TF_TAB;
    TF_3CHAR( 'T', 'r', 'E' );
    TF_HWORD( functionNo );
    TF_HWORD( param1 );
    TF_HWORD( param2 );
    TF_HWORD( param3 );

#ifdef SELFLIB_POLLEMU
    }

    if( functionNo == FNO_STATUS_CHECK )
    {
        SelfLibPollEmu_StatusCheck();

        if( SelfLibEmu_Ret == 0xFF )
        {
            TF_CHAR( 'C' );
        }
        else
        {
            TF_TAB;
            TF_HWORD( SelfLibEmu_Ret );
            TF_CRLF;
        }

        return SelfLibEmu_Ret;
    }
#endif
```

```
/*=========================================================================*/
    /* Explicit Flash environment activation and deactivation are treated
*/
    /* different from other Flash functions. However need to be handled
*/
    /* by this function, as this is the only interface between the
*/
    /* SelfLib_RomOrRam and SelfLib_ToRAM sections
*/

/*=========================================================================*/
    #if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
        if( functionNo == FNO_SELFLIB_INTERNAL_ENV_ACT )
        {
            SelfLib_BasFct_FlashEnv_Activate();
            return ret;
        }

        if ( functionNo == FNO_SELFLIB_INTERNAL_ENV_DEACT )
        {
            SelfLib_BasFct_FlashEnv_Deactivate();
            return ret;
        }

    #endif


/*=========================================================================*/
    /* Activate Flash environment
*/

/*=========================================================================*/
    #if SELFLIB_STATUS_CHECK!=STATUS_CHECK_USER
        SelfLib_BasFct_FlashEnv_Activate();
    #endif


/*=========================================================================*/
    /* Call the firmware function
*/

/*=========================================================================*/
    TT_OUT2BYTESBROM(functionNo, param1);

    ret = SelfLib_BasFct_TrapEntry ( functionNo,
                                     param1,
                                     param2,
                                     param3,
                                     sysData );

    TT_OUT2BYTESBROM(0,0);


/*=========================================================================*/
```

```
    /* Do the Status check, if required
*/
    /*
*/
    /* - Some devices do the status check inside the firmware, so no
*/
    /*   operation in the SelfLib
*/
    /* - In Polling mode the status check is done by the user program
*/
    /* - Otherwise, do it here
*/

/*=========================================================================*/
    #if SELFLIB_STATUS_CHECK==STATUS_CHECK_SELFLIB
        if ( SELFLIB_OK == ret )
        {
            if ( statCheck == STATCHECK_ON )
            {
                do
                {
                    ret = SelfLib_BasFct_TrapEntry ( FNO_STATUS_CHECK,
                                                     0,
                                                     0,
                                                     0,
                                                     sysData  );

                } while ( SELFLIB_BUSY == ret );
            }
        }

    #endif

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;


/*=========================================================================*/
    /* Dectivate Flash environment
*/

/*=========================================================================*/
    #if SELFLIB_STATUS_CHECK!=STATUS_CHECK_USER
        SelfLib_BasFct_FlashEnv_Deactivate();
    #endif

    return ret;
}
```

3.2.9 SelfLibCommands.c

```
/*=============================================================================
*/
```

```
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibCommands.c
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
/*============================================================================
*/
/*                                  COPYRIGHT
*/
/*============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*============================================================================
*/
/* Purpose:
*/
/*              User interface functions of the SelfLib
*/
/*============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*============================================================================
*/

#define  DECLARE_VARS
#include "SelfLibGlobal.h"
#undef   DECLARE_VARS


/*============================================================================
*/
/* Function name:   SelfLib_BasFct_CallBromFlashFunc
*/
/* IN:              See SelfLib_BasFct_BromFlashFunc in SelfLibBrom.c
*/
/* OUT:             See SelfLib_BasFct_BromFlashFunc in SelfLibBrom.c
*/
/* Describtion:     Calls the SelfLib_BasFct_BromFlashFunc function either
*/
/*                  via function pointer or direct, depending on the memory
*/
/*                  scenario
*/
```

```
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_BasFct_CallBromFlashFunc ( u32                functionNo,
                                       u32                param1,
                                       u32                param2,
                                       u32                param3,
                                       SELFLIB_STATCHECK  statCheck )
{
    u32 ret;

    if( pfctSelfLib_BasFct_BromFlashFunc == NULL )
    {
        ret = SelfLib_BasFct_BromFlashFunc(    functionNo,
                                               param1,
                                               param2,
                                               param3,
                                               statCheck );
    }
    else
    {
        ret = pfctSelfLib_BasFct_BromFlashFunc(functionNo,
                                               param1,
                                               param2,
                                               param3,
                                               statCheck );
    }

    return ret;
}



#if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER

/*============================================================================
*/
/* Function name:   SelfLib_FlashEnv_Activate
*/
/* IN:              -
*/
/* OUT:             SELFLIB_OK              Flash erased correctly
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/* Describtion:     Activates the Flash environment and the device firmware
*/
```

```
/*============================================================================
*/
    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_RomOrRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_RomOrRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_RomOrRam"
    #endif

    u32 SelfLib_FlashEnv_Activate( void )
    {
        u32 ret;

        TF_3CHAR('F','E','A');
        TF_CRLF;

        ret = SelfLib_BasFct_CallBromFlashFunc( FNO_SELFLIB_INTERNAL_ENV_ACT,
                                                0,
                                                0,
                                                0,
                                                STATCHECK_OFF );

        TF_TAB;
        TF_HWORD( ret );
        TF_CRLF;

        return ( ret );
    }


/*============================================================================
*/
/* Function name:   SelfLib_FlashEnv_Deactivate
*/
/* IN:              -
*/
/* OUT:             -
*/
/* Describtion:     Deactivates the Flash environment and the device
firmware*/
/*                  Waits the deactivation time for Flash to come up again
*/
/*============================================================================
*/
    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_RomOrRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_RomOrRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_RomOrRam"
    #endif

    void SelfLib_FlashEnv_Deactivate( void )
    {
```

```
        u32 ret;

        TF_3CHAR('F','E','D');
        TF_CRLF;

        /* returned value is a dummy value*/
        ret = SelfLib_BasFct_CallBromFlashFunc(
FNO_SELFLIB_INTERNAL_ENV_DEACT,
                                               0,
                                               0,
                                               0,
                                               STATCHECK_OFF );

        /* just to get rid of the warning of non used return value */
        ret = ret;
    }

#endif


/*=============================================================================
*/
/* Function name:   SelfLib_BlankCheck
*/
/* IN:              blockNoStart            first block to be checked
*/
/*                  blockNoEnd              last block to be checked
*/
/* OUT:             SELFLIB_OK              Block is blank
*/
/*                  SELFLIB_ERR_FLASHPROCn  (n=0x00~0x02)
*/
/*                                          Block is not blank
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters
*/
/* Describtion:     Blank check on the selected block
*/
/*=============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_BlankCheck( u32 blockNoStart,
                         u32 blockNoEnd )
{

    u32 ret;

    TF_3CHAR('B','C',' ');
```

```
    TF_HWORD ( blockNoStart );
    TF_HWORD ( blockNoEnd );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_BLOCK_BLANKCHECK,
                                            blockNoStart,
                                            blockNoEnd,
                                            0,
                                            STATCHECK_ON );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_Write
*/
/* IN:              addSrc:                 Data source address
*/
/*                  addDest:                Destination address in the Flash
*/
/*                  length:                 number of WORDS to be written
*/
/* OUT:             SELFLIB_OK              Flash written correctly
*/
/*                  SELFLIB_ERR_FLASHPROCn (n=0x00~0x02)
*/
/*                                          Write error
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters
*/
/*                  SELFLIB_ERR_PROTECTION  Flash protection error
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/* Describtion:     Write data to the Flash
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_Write( void *addSrc, void *addDest, u32  length )
{
```

```
    u32 ret;

    TF_3CHAR('W',' ',' ');
    TF_HWORD( (u32) addSrc );
    TF_HWORD( (u32) addDest );
    TF_HWORD( length );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_WORD_WRITE,
                                            (u32)addDest,
                                            (u32)addSrc,
                                            length,
                                            STATCHECK_ON );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_IVerify
*/
/* IN:              blockNoStart            first block to be checked
*/
/*                  blockNoEnd              last block to be checked
*/
/* OUT:             SELFLIB_OK              Flash written correctly
*/
/*                  SELFLIB_ERR_FLASHPROCn (n=0x00~0x02)
*/
/*                                          Verify error
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters
*/
/* Describtion:     Do an internal verify on a Flash block. Necessary to
*/
/*                  ensure the specified data retention after writing data
*/
/*                  into a Flash block
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_IVerify( u32 blockNoStart,
```

```
                      u32 blockNoEnd )
{

    u32 ret;

    TF_3CHAR('I','V',' ');
    TF_HWORD ( blockNoStart );
    TF_HWORD ( blockNoEnd );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_BLOCK_IVERIFY,
                                            blockNoStart,
                                            blockNoEnd,
                                            0,
                                            STATCHECK_ON );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_Erase
*/
/* IN:              blockNoStart            first block to be erased
*/
/*                  blockNoEnd              last block to be erased
*/
/* OUT:             SELFLIB_OK              Flash erased correctly
*/
/*                  SELFLIB_ERR_FLASHPROCn (n=0x00~0x02)
*/
/*                                          Erase error
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters
*/
/*                  SELFLIB_ERR_PROTECTION  Flash protection error
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/*                                          !!! FLMD0 is not checked during
*/
/*                                              erase on CZ6 devices !!!
*/
/* Describtion:     Erase a flash block
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
```

```
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_Erase ( u32 blockNoStart,
                    u32 blockNoEnd )
{

    u32 ret;

    TF_3CHAR('E','R','S');
    TF_HWORD ( blockNoStart );
    TF_HWORD ( blockNoEnd );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_BLOCK_ERASE,
                                            blockNoStart,
                                            blockNoEnd,
                                            0,
                                            STATCHECK_ON );
    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_Read
*/
/* IN:              readAddress:            Source address in flash
*/
/*                  destAddress:            Destination address in RAM where
*/
/*                                          the data is copied
*/
/*                  length:                 Number of words to be read
*/
/* OUT:             SELFLIB_OK              Flash erased correctly
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters or function
*/
/*                                          not supported
*/
/* Describtion:     Read out a block of data from Flash
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
```

```
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_Read( void *readAddress,
                  void *destAddress,
                  u32  length )
{
    u32 ret;

    TF_3CHAR('R','E','A');
    TF_HWORD( (u32)readAddress );
    TF_HWORD( (u32)destAddress );
    TF_HWORD( length );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_WORD_READ,
                                            (u32)readAddress,
                                            (u32)destAddress,
                                            length,
                                            STATCHECK_OFF );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_ModeCheck
*/
/* IN:              -
*/
/* OUT:             SELFLIB_OK              Flash erased correctly
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/* Describtion:     Check Vdd on the FLMD0 pin
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_ModeCheck( void )
{
    u32 ret;
```

```
    TF_3CHAR('M','C',' ');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_MODE_CHECK,
                                            0,
                                            0,
                                            0,
                                            STATCHECK_OFF );

    /* Change the error response*/
    if( ret == SELFLIB_ERR_FLMD0_MODECHECK )
    {
        ret = SELFLIB_ERR_FLMD0;
    }

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_SetSecFlags
*/
/* IN:              SecFlags:              Security flags to be set:
*/
/*                      Bit 0:             0: Block swapped to 0x00000000
*/
/*                                         1: Block not swapped to
0x0000000*/
/*                      Bit 1:             0: Chip erase disabled
*/
/*                                         1: Chip erase enabled
*/
/*                      Bit 2:             0: Block erase disabled
*/
/*                                         1: Block erase enabled
*/
/*                      Bit 3:             0: Write disabled
*/
/*                                         1: Write enabled
*/
/*                      Bit 4:             0: Read disabled
*/
/*                                         1: Read enabled
*/
/*                      Bit 5:             0: Boot cluster protected
*/
/*                                         1: Boot cluster not protected
*/
```

```
/*                      Bit 23~31:          Last boot block number (= Number
*/
/*                                          of blocks in the boot cluster-1)
*/
/* OUT:             SELFLIB_OK              Flags set correctly
*/
/*                  SELFLIB_ERR_FLASHPROCn (n=0x00~0x02)
*/
/*                                          Flag setting error
*/
/*                  SELFLIB_ERR_PARAMETER   Incorrect parameters
*/
/*                  SELFLIB_ERR_PROTECTION  Flash protection error
*/
/*                  SELFLIB_ERR_FLMD0       Vdd on FLMD0 not applied
*/
/* Describtion:     Sets the security flags in a selected block
*/
/*=============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_SetSecFlags( u32  SecFlags )
{
    u32 ret;

    TF_3CHAR('S','S','F');
    TF_HWORD( SecFlags );
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_FLASHINFO_SET,
                                            SecFlags,
                                            0,
                                                      0,
                                            STATCHECK_ON );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*=============================================================================
*/
/* Function name:   SelfLib_BootSwap
*/
```

```
/* IN:              -
*/
/* OUT:             SELFLIB_OK              Boot swap executed successfully
*/
/*                  SELFLIB_ERR_UNSUPPORTED The operation is not supported
*/
/*                  SELFLIB_ERR_PROTECTION  Flash protection error
*/
/* Describtion:     Swap the boot blocks (clusters). The function does not
*/
/*                  affect the security flags.
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32  SelfLib_BootSwap( void )
{
    u32 ret;

    TF_3CHAR('S','W','P');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_BOOTBLOCK_SWAP,
                                            0,
                                            0,
                                            0,
                                            STATCHECK_OFF );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_GetInfo_Device
*/
/* IN:              -
*/
/* OUT:             CPU number in decimal format
*/
/*                  e.g. uPD70F3166 (V850ES/SA3) = 3166 (=0x0C5E)
*/
/* Describtion:     Get the CPU number in decimal format
*/
```

```
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_GetInfo_Device( void )
{
    u32 ret;

    TF_3CHAR('G','I','D');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_FLASHINFO_READ,
                                            OPT_FLASHINFO_DEVCODE_BLKCNT,
                                            0,
                                            0,
                                            STATCHECK_OFF );
    ret = ret >> 16;

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_GetInfo_BlockCnt
*/
/* IN:              -
*/
/* OUT:             number of flash blocks
*/
/* Describtion:     Get the number of flash blocks on the device
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_GetInfo_BlockCnt( void )
{
    u32 ret;
```

```
    TF_3CHAR('G','I','B');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_FLASHINFO_READ,
                                            OPT_FLASHINFO_DEVCODE_BLKCNT,
                                            0,
                                            0,
                                            STATCHECK_OFF );

    ret = ret & 0xFFFF;

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*=============================================================================
*/
/* Function name:   SelfLib_GetInfo_SecFlags
*/
/* OUT:             Security flags setting:
*/
/*                      Bit 0:              0: Block swapped to 0x00000000
*/
/*                                          1: Block not swapped to
0x0000000*/
/*                      Bit 1:              0: Chip erase disabled
*/
/*                                          1: Chip erase enabled
*/
/*                      Bit 2:              0: Block erase disabled
*/
/*                                          1: Block erase enabled
*/
/*                      Bit 3:              0: Write disabled
*/
/*                                          1: Write enabled
*/
/*                      Bit 4:              0: Read disabled
*/
/*                                          1: Read enabled
*/
/*                      Bit 5:              0: Boot cluster protected
*/
/*                                          1: Boot cluster not protected
*/
/*                      Bit 24~31:          Last boot block number (= Number
*/
/*                                          of blocks in the boot cluster-1)
*/
```

```
/* Describtion:     Get the current security flag setting
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_GetInfo_SecFlags( void )
{
    u32 ret;

    TF_3CHAR('G','I','F');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_FLASHINFO_READ,
                                            OPT_FLASHINFO_SEC_STAT,
                                            0,
                                            0,
                                            STATCHECK_OFF );

    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_GetInfo_BootSwap
*/
/* IN:              -
*/
/* OUT:             0: Boot swapping is not performed
*/
/*                  1: Boot swapping is performed
*/
/* Describtion:     Check whether boot swap is executed on the next RESET
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif
```

```
u32 SelfLib_GetInfo_BootSwap( void )
{
    u32 ret;

    TF_3CHAR('G','I','S');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc( FNO_FLASHINFO_READ,
                                            OPT_FLASHINFO_SWAP_STAT,
                                            0,
                                            0,
                                            STATCHECK_OFF );
    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ( ret );
}


/*============================================================================
*/
/* Function name:   SelfLib_GetInfo_BlockEndAdd
*/
/* IN:              blockNo:                requested block number
*/
/* OUT:             End address of the selected block
*/
/* Describtion:     get a block end address
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_RomOrRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_RomOrRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_RomOrRam"
#endif

u32 SelfLib_GetInfo_BlockEndAdd( u32 blockNo )
{
    u32 ret;

    TF_3CHAR('G','I','E');
    TF_CRLF;

    ret = SelfLib_BasFct_CallBromFlashFunc(
                FNO_FLASHINFO_READ,
                OPT_FLASHINFO_BLKENDADD_OFF + blockNo,
                0,
                0,
                STATCHECK_OFF );
    TF_TAB;
    TF_HWORD( ret );
```

```
    TF_CRLF;

    return ( ret );
}


#if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
/*============================================================================
*/
/* Function name:   SelfLib_StatusCheck
*/
/* IN:              -
*/
/* OUT:             SELFLIB_OK              Flash operation terminated
*/
/*                                          sucessfully
*/
/*                  SELFLIB_ERR_FLASHPROCn (n=0x00~0x02) operation error
*/
/* Describtion:     Checks the status of background flash operations (erase,
*/
/*                  write, ... ) if user application is set to polling
*/
/*============================================================================
*/
    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_RomOrRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_RomOrRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_RomOrRam"
    #endif

    u32 SelfLib_StatusCheck( void )
    {
        u32 ret;

    #ifndef SELFLIB_POLLEMU
        TF_3CHAR('S','t','C');
        TF_CRLF;
    #endif

        ret = SelfLib_BasFct_CallBromFlashFunc( FNO_STATUS_CHECK,
                                                0,
                                                0,
                                                0,
                                                STATCHECK_OFF );
    #ifndef SELFLIB_POLLEMU
        TF_TAB;
        TF_HWORD( ret );
        TF_CRLF;
    #endif

        return ( ret );
    }
```

```
#endif
```

3.2.10 SelfLibDebug.c

```
/*============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibDebug.c
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
/*============================================================================
*/
/*                                  COPYRIGHT
*/
/*============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*============================================================================
*/
/* Purpose:
*/
/*              File containing the debug output functions. Please note that
*/
/*              the basic functions for the output (init and acutal output)
*/
/*              are hardware dependent and have to be placed in the project
*/
/*              which is using this library. This ensures that the library
*/
/*              remains hardware independent.
*/
/*============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*============================================================================
*/

#include "selflibglobal.h"


#define CR              13
#define LF              10
```

```
/*============================================================================
*/
/* Place Test routines to SelfLib_ToRAM section, so org location can be
*/
/* accessed by Flash routines and copied location can be accessed by
*/
/* SelfLib_ToRam routines
*/
/*============================================================================
*/



/*============================================================================
*/
/* Test FUNCTION
*/
/*============================================================================
*/
#if SELFLIB_TEST == TEST_FUNCTION
    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_ToRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_ToRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_ToRam"
    #endif

    void Selflib_Output3Char( unsigned char ucC1, unsigned char ucC2,
                          unsigned char ucC3 )
    {
        Selflib_OutputChar(ucC1);
        Selflib_OutputChar(ucC2);
        Selflib_OutputChar(ucC3);
        Selflib_OutputChar(' ');
    }


    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_ToRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_ToRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_ToRam"
    #endif

    void Selflib_OutputTab( void )
    {
        Selflib_OutputChar(' ');
        Selflib_OutputChar(' ');
        Selflib_OutputChar(' ');
        Selflib_OutputChar(' ');
    }
```

```
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

void Selflib_OutputCrLf( void )
{
    Selflib_OutputChar( CR );
    Selflib_OutputChar( LF );
}

#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
#endif

void Selflib_OutputHByte( unsigned char ucData )
{
   unsigned char ucTmp;

   ucTmp = ucData>>4;

   if (ucTmp > 9)
   {
       Selflib_OutputChar( 'A' 10 );
   }
   else
   {
       Selflib_OutputChar( '0'  );
   }

   ucData &= 0x0F;
   if (ucData > 9)
   {
       Selflib_OutputChar( 'A'  10 );
   }
   else
   {
       Selflib_OutputChar( '0'   );
   }
}


#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRam"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRam"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRam"
```

```
    #endif

    void Selflib_OutputHWord( u32 ucData )
    {
        Selflib_OutputHByte ((char)(ucData >> 24));
        Selflib_OutputHByte ((char)(ucData >> 16));
        Selflib_OutputHByte ((char)(ucData >> 8));
        Selflib_OutputHByte ((char)ucData );
        Selflib_OutputChar(' ');
    }


    #if   SL_COMPILER == GHS
        #pragma ghs section text=".SelfLib_ToRam"
    #elif SL_COMPILER == IAR
        #pragma location = "SelfLib_ToRam"
    #elif SL_COMPILER == NEC
        #pragma text "SelfLib_ToRam"
    #endif

    void Selflib_OutputRet( u32 iTab, u32 iRet )
    {
        int i;

        for (i=0; i<iTab; )
        {
            Selflib_Output3Char( ' ', ' ', ' ' );
        }
        Selflib_Output3Char( 'R', 'e', 't' );

        Selflib_OutputHWord( iRet );
        Selflib_OutputChar( CR ); Selflib_OutputChar( LF );
    }

#endif
```

3.2.11 SelfLibEnvironment.h

```
/*=============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibEnvironment.h
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
/*=============================================================================
*/
/*                                   COPYRIGHT
*/
/*=============================================================================
*/
```

```
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*================================================================================
*/
/* Purpose:
*/
/*              File which contains all the technology and device family
*/
/*              dependent settings
*/
/*================================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*================================================================================
*/

#ifndef SELFLIBENVIRONMENT_H
#define SELFLIBENVIRONMENT_H


/*================================================================================
*/
/* Flash Type dependant definitions
*/
/*================================================================================
*/
#define FLASHREG_BASE                                   0xfffff8c0

#define PFCMD       (*((volatile unsigned char  *)(FLASHREG_BASE+0x10)))
#define FLPMC       (*((volatile unsigned char  *)(FLASHREG_BASE+0x14)))
#define FLSPM_MASK                                      0x02

/* Time to activate the FLASH after programming (us) */
#define TIME_US_ACTIVATE_FLASH                          20

/*================================================================================
*/
/*  Function numbers
*/
/*================================================================================
*/
#define FNO_ENV_INIT                                    0
#define FNO_BLOCK_ERASE                                 3
#define FNO_WORD_WRITE                                  4
#define FNO_BLOCK_IVERIFY                               6
#define FNO_BLOCK_BLANKCHECK                            8
#define FNO_FLASHINFO_READ                              9
    #define OPT_FLASHINFO_DEVFIR_VER                    0x0
```

```
    #define OPT_FLASHINFO_SUP_FUNC                  0x1
        #define FLASHINFO_CHECKSUP_USRHDR(a)        ((a>>13)&1)

    #define OPT_FLASHINFO_DEVCODE_BLKCNT            0x2
    #define OPT_FLASHINFO_SEC_STAT                  0x3
    #define OPT_FLASHINFO_SWAP_STAT                 0x4
    #define OPT_FLASHINFO_BLKENDADD_OFF             0x5
#define FNO_FLASHINFO_SET                           10
#define FNO_STATUS_CHECK                            11
#define FNO_BOOTBLOCK_SWAP                          12
#define FNO_INT_REGISTRATION                        13
#define FNO_MODE_CHECK                              14
#define FNO_WORD_READ                               15
#define FNO_SELFLIB_INTERNAL_ENV_ACT                254
#define FNO_SELFLIB_INTERNAL_ENV_DEACT              255


/*=============================================================================
*/
/*  Internal error number
*/
/*=============================================================================
*/
#define SELFLIB_ERR_FLMD0_MODECHECK                 0x01

#endif
```

3.2.12 SelfLibGlobal.h

```
/*=============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibGlobal.h
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
/*=============================================================================
*/
/*                                  COPYRIGHT
*/
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              Header file which includes the global defines and variables
*/
/*              used by the library.
*/
```

```
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/


#ifndef SELFLIBGLOBAL_H
#define SELFLIBGLOBAL_H

#include "nec_types.h"
#include "SelfLibEnvironment.h"
#include "selflib.h"
#include "SelfLibSpecific.h"

/*=============================================================================
*/
/* NEC internal define for debugging purpose (Setup of debug output)
*/
/* ----- May be removed in the target application -----
*/
/*=============================================================================
*/
#include "target.h"
/*=============================================================================
*/

/*=============================================================================
*/
/* NEC test function default define
*/
/*=============================================================================
*/
#ifndef SELFLIB_TEST
    #define TEST_OFF        0
    #define TEST_FUNCTION   1
    #define TEST_TIMING     2
    #define TEST_TIMINGBROM 3

    #define SELFLIB_TEST    TEST_OFF
#endif

/*=============================================================================
*/
/* Library version
*/
/*=============================================================================
*/
```

```
#define LIBRARY_VERSION     0x0104

/*============================================================================
*/
/* Enumerations
*/
/*============================================================================
*/
typedef enum { STATCHECK_OFF, STATCHECK_ON } SELFLIB_STATCHECK;

/*============================================================================
*/
/* BROM access function prototype
*/
/*============================================================================
*/
u32     SelfLib_BasFct_BromFlashFunc(    u32   functionNo,
                                         u32   param1,
                                         u32   param2,
                                         u32   param3,
                                         SELFLIB_STATCHECK statCheck );


/*============================================================================
*/
/* Device Interface (asm-code) functions prototypes
*/
/*============================================================================
*/
#if SELFLIB_SPECIFIC_OPTIONS==SELFLIB_PHOENIX_F
    void    SelfLib_BasFct_WaitRo(       void );
#elif SELFLIB_SPECIFIC_OPTIONS==SELFLIB_PHOENIX_FS
    void    SelfLib_BasFct_WaitRo(       void );
#else
    void    SelfLib_BasFct_Wait(         u32   timeUs,
                                         u32   frequency );
#endif

s32     SelfLib_BasFct_ExeOffset(        void );
u32     SelfLib_BasFct_IntNmiDisable(    void );
void    SelfLib_BasFct_RestorePSW(       u32   psw );
void    SelfLib_BasFct_WriteSecReg(      u08   *regDest,
                                         u08   *regProtection,
                                         u08   value );
u32     SelfLib_BasFct_TrapEntry(        u32   a,
                                         u32   b,
                                         u32   c,
                                         u32   d,
                                         u08   *ep );
u32     SelfLib_BasFct_TrapEntry_void(   u32   a,
                                         u32   b,
                                         u32   c,
                                         u32   d,
                                         u08   *ep );
s32     SelfLib_Div321632(               s32   a,
```

```
                                        s16   b );
void    SelfLib_BasFct_SectionAdd_ToRamUsrInt(
                                        u32 *secStart,
                                        u32 *secSize  );
void    SelfLib_BasFct_SectionAdd_ToRamUsr(
                                        u32 *secStart,
                                        u32 *secSize  );
void    SelfLib_BasFct_SectionAdd_ToRam(
                                        u32 *secStart,
                                        u32 *secSize  );
void    SelfLib_BasFct_SectionAdd_RomOrRam(
                                        u32 *secStart,
                                        u32 *secSize  );


/*==========================================================================
*/
/* Variables
*/
/*==========================================================================
*/
#ifdef DECLARE_VARS
    #undef DECLARE_VARS

    #if   SL_COMPILER == GHS
        #pragma ghs section bss   = ".SelfLib_Ram"
        #define NOINIT
    #elif SL_COMPILER == IAR
        #pragma    dataseg = SelfLib_Ram
        #define NOINIT __no_init
    #elif SL_COMPILER == NEC
        #define NOINIT
        #pragma section data "SelfLib_RAM" begin
    #endif

    NOINIT u32  (*pfctSelfLib_BasFct_BromFlashFunc)(
                                            u32   functionNo,
                                            u32   param1,
                                            u32   param2,
                                            u32   param3,
                                            SELFLIB_STATCHECK statCheck
);

    /* SysData is Flash technology dependant */
    #if SELFLIB_SPECIFIC_OPTIONS == SELFLIB_KX1_V1
        NOINIT u08 sysData[0x54];
        NOINIT u32 u32SelfLib_frequency;

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_F
        NOINIT u08 sysData[0x54];

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_FS
        NOINIT u08 sysData[0x54];

    #elif SELFLIB_SPECIFIC == SELFLIBSPECIFIC_CZ6
```

```
        NOINIT u08 sysData[0x54];
        NOINIT u32 u32SelfLib_frequency;

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_DEFAULT
    #endif

    #ifdef SELFLIB_POLLEMU
        NOINIT u08 SelfLibEmu_FWStack[0x200];
        NOINIT u32 SelfLibEmu_Ret;
        NOINIT u32 SelfLibEmu_Status;
        NOINIT u32 SelfLibEmu_AppSP;
        NOINIT u32 SelfLibEmu_FWSP;
    #endif

    #if   SL_COMPILER == GHS
    #elif SL_COMPILER == IAR
        #pragma    dataseg   =   default
    #elif SL_COMPILER == NEC
        #pragma section data "SelfLib_RAM" end
    #endif

#else
    extern u32  (*pfctSelfLib_BasFct_BromFlashFunc)(
                                                u32   functionNo,
                                                u32   param1,
                                                u32   param2,
                                                u32   param3,
                                                SELFLIB_STATCHECK statCheck
);

    #if SELFLIB_SPECIFIC_OPTIONS == SELFLIB_KX1_V1
        extern u08 sysData[];
        extern u32 u32SelfLib_frequency;

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_F
        extern u08 sysData[];

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_FS
        extern u08 sysData[];

    #elif SELFLIB_SPECIFIC == SELFLIBSPECIFIC_CZ6
        extern u08 sysData[];
        extern u32 u32SelfLib_frequency;

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_DEFAULT
        #define sysData NULL

    #endif

    #ifdef SELFLIB_POLLEMU
        extern u08 SelfLibEmu_FWStack[];
        extern u32 SelfLibEmu_Ret;
        extern u32 SelfLibEmu_Status;
        extern u32 SelfLibEmu_AppSP;
        extern u32 SelfLibEmu_FWSP;
```

```
    #endif

#endif


/*============================================================================
*/
/* Test & Debug functions
*/
/*============================================================================
*/

#if SELFLIB_TEST == TEST_FUNCTION
    void Selflib_OutputChar         ( u08 data );
    void Selflib_Output3Char        ( u08 c1, u08 c2, u08 c3 );
    void Selflib_OutputTab          ( void );
    void Selflib_OutputCrLf         ( void );
    void Selflib_OutputHByte        ( u08 data);
    void Selflib_OutputHWord        ( u32 data);
    void Selflib_OutputRet          ( u32 tab, u32 ret );


    #define TF_HWORD(a)             Selflib_OutputHWord(a)
    #define TF_CHAR(a)              Selflib_OutputChar(a)
    #define TF_3CHAR(a,b,c)         Selflib_Output3Char(a,b,c)
    #define TF_CRLF                 Selflib_OutputCrLf()
    #define TF_TAB                  Selflib_OutputTab()
    #define TF_RET(a,b)             Selflib_OutputRet (a,b)

#else
    #define TF_HWORD(a)
    #define TF_CHAR(a)
    #define TF_3CHAR(a,b,c)
    #define TF_CRLF
    #define TF_TAB
    #define TF_RET(a,b)

#endif

#if SELFLIB_TEST == TEST_TIMING
    #define TT_OUT2BYTES(a,b)       TT_TG_OUT2BYTES(a,b)
    #define TT_OUT2BYTESBROM(a,b)

#elif SELFLIB_TEST == TEST_TIMINGBROM
    #define TT_OUT2BYTES(a,b)
    #define TT_OUT2BYTESBROM(a,b)   TT_TG_OUT2BYTES(a,b)

#else
    #define TT_OUT2BYTES(a,b)
    #define TT_OUT2BYTESBROM(a,b)

#endif

#endif
```

3.2.13 SelfLibInit.c

```
/*=============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = SelfLibInit.c
*/
/* Version      = 1.04
*/
/* Date         = 22.03.2006
*/
/*=============================================================================
*/
/*                                  COPYRIGHT
*/
/*=============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*=============================================================================
*/
/* Purpose:
*/
/*              File providing the all the required initialization functions
*/
/*              for the selfprogramming
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

#include "SelfLibGlobal.h"

/*=============================================================================
*/
/* Function name:   SelfLib_BasFct_SysDataInit
*/
/* IN:              -
*/
/* OUT:             SELFLIB_OK              No error
*/
/*                  SELFLIB_ERR_PARAMETER   frequency!=0 on non CZ6 devices
*/
/*
*/
```

```
/* Describtion:     For some device types additional initialisation, like
*/
/*                  for the SysData structure is required. This is done here
*/
/*                  Furthermore a crosscheck of the SELFLIB_SPECIFIC_OPTIONS
*/
/*                  settings is done here for non CZ6 devices. The frequency
*/
/*                  setting must be 0 in that case
*/
/*=============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_Rom"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_Rom"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_Rom"
#endif

u32 SelfLib_BasFct_SpecificInit( u32   frequency )
{
    u32 ret = SELFLIB_OK;

    #if SELFLIB_SPECIFIC_OPTIONS == SELFLIB_KX1_V1
    {
        /* --------------- SysData Init ------------------ */
        /* 0x00, 0xe1, 0xf5, 0x05, 0x00, 0xe1, 0xf5, 0x05, */
        /* 0x20, 0x4e, 0x00, 0x00, 0xe8, 0x03, 0x00, 0x00, */
        /* 0x10, 0x27, 0x00, 0x00, 0x50, 0xc3, 0x00, 0x00, */
        /* 0x20, 0x4e, 0x20, 0x4e, 0x20, 0x4e, 0xd0, 0x07, */
        /* 0xff, 0xff                                      */

        u32 *pSysData = (u32 *)sysData;

        (*pSysData++) = 0x05f5e100;
        (*pSysData++) = 0x05f5e100;
        (*pSysData++) = 0x00004e20;
        (*pSysData++) = 0x000003e8;
        (*pSysData++) = 0x00002710;
        (*pSysData++) = 0x0000c350;
        (*pSysData++) = 0x4e204e20;
        (*pSysData++) = 0x07d04e20;
        (*pSysData++) = 0x0000ffff;
        (*pSysData++) = 0x00000000;

        *(u32 *)&sysData[0x40] = frequency;

        /* Set the system frequency */
        u32SelfLib_frequency = frequency;

        if( ( frequency<100000 ) || ( frequency>300000000 ) )
        {
            ret = SELFLIB_ERR_PARAMETER;
        }
```

```
}

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_F
{
    /* --------------- SysData Init ----------------*/
    /* 88,   00,   80,   00,   80,   00,   14,   5, */
    /* 6,    2,    0x0e, 97,   255,  0,    7,    0, */
    /* 3,    0,    2,    38,   2,    2,    2,    8, */
    /* 9,    9,    19,   0x17, 0x16, 16,   4,    4, */
    /* 1,    1,    1,    1,    1,    0,    0,    0  */

    u32 *pSysData = (u32 *)sysData;

    (*pSysData++) = 0x00500058;
    (*pSysData++) = 0x050e0050;
    (*pSysData++) = 0x610e0206;
    (*pSysData++) = 0x000700ff;
    (*pSysData++) = 0x26020003;
    (*pSysData++) = 0x08020202;
    (*pSysData++) = 0x17130909;
    (*pSysData++) = 0x04041016;
    (*pSysData++) = 0x01010101;
    (*pSysData++) = 0x00000001;

    /* --- Emulate Polling mode by ROM correction------*/
    /* --- ROM Correction setup -----------------------*/
    #if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
        CORAD7H = 0x0000;
        CORAD7L = 0x31be;   /* Start Sequencer
                               - CheckCommand Call replaced */
        CORCN   = 0x80;
    #endif

    if( frequency != 0 )
    {
        ret = SELFLIB_ERR_PARAMETER;
    }
}

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_PHOENIX_FS
{
    /* --------------- SysData Init ----------------*/
    /* 88,   00,   80,   00,   80,   00,   14,   5, */
    /* 3,    2,    0x0e, 97,   255,  0,    7,    0, */
    /* 3,    0,    2,    38,   2,    2,    2,    8, */
    /* 9,    9,    19,   0x17, 0x16, 16,   4,    4, */
    /* 1,    1,    1,    1,    1,    0,    0,    0  */

    u32 *pSysData = (u32 *)sysData;

    (*pSysData++) = 0x00500058;
    (*pSysData++) = 0x050e0050;
    (*pSysData++) = 0x610e0203;
    (*pSysData++) = 0x000700ff;
    (*pSysData++) = 0x26020003;
```

```
    (*pSysData++) = 0x08020202;
    (*pSysData++) = 0x17130909;
    (*pSysData++) = 0x04041016;
    (*pSysData++) = 0x01010101;
    (*pSysData++) = 0x00000001;

    if( frequency != 0 )
    {
        ret = SELFLIB_ERR_PARAMETER;
    }
}

#elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_RS2
{
    /* --------------- SysData Init -----------------*/
    /* 88,   00,   80,   00,   80,   00,   14,   5, */
    /* 3,    2,    0x0e, 156,  254,  0,    4,    0, */
    /* 3,    0,    2,    0,    0,    0,    8,    9, */
    /* 9,    99,   99,   0,    0,    0,    4,    4, */
    /* 1,    1,    0,    0,    1,    0,    0,    0  */

    /* --------------- SysData Init -------------------*/
    /* 0x58, 0x00, 0x50, 0x00, 0x50, 0x00, 0x0e, 0x05, */
    /* 0x03, 0x02, 0x0e, 0x9C, 0xFE, 0x00, 0x04, 0x00, */
    /* 0x03, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x08, */
    /* 0x09, 0x63, 0x63, 0x00, 0x00, 0x00, 0x04, 0x04, */
    /* 0x01, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00  */

    u32 *pSysData = (u32 *)sysData;

    (*pSysData++) = 0x00500058;
    (*pSysData++) = 0x050e0050;
    (*pSysData++) = 0x9c0e0203;
    (*pSysData++) = 0x000400FE;
    (*pSysData++) = 0x00020003;
    (*pSysData++) = 0x08000000;
    (*pSysData++) = 0x00636309;
    (*pSysData++) = 0x04040000;
    (*pSysData++) = 0x00000101;
    (*pSysData++) = 0x00000001;

    if( frequency != 0 )
    {
        ret = SELFLIB_ERR_PARAMETER;
    }
}

#elif SELFLIB_SPECIFIC == SELFLIBSPECIFIC_CZ6
    u32 *pSysData = (u32 *)sysData;

    (*pSysData++) = 0x00000000;
    (*pSysData++) = 0x00000000;
    (*pSysData++) = 0x00000000;
    (*pSysData++) = 0x00000000;
    (*pSysData++) = 0x00000000;
```

```
        (*pSysData++) = 0x00000000;
        (*pSysData++) = 0x00000000;
        (*pSysData++) = 0x00000000;
        (*pSysData++) = 0x00000000;
        (*pSysData++) = 0x00000000;

        /* Set the system frequency */
        u32SelfLib_frequency = frequency;

        if( ( frequency<100000 ) || ( frequency>300000000 ) )
        {
            ret = SELFLIB_ERR_PARAMETER;
        }

    #elif SELFLIB_SPECIFIC_OPTIONS == SELFLIB_DEFAULT
        if( frequency != 0 )
        {
            ret = SELFLIB_ERR_PARAMETER;
        }

    #else
        #error wrong SELFLIB_SPECIFIC_OPTIONS definition
    #endif

    /* Check correct USER_FCT_HANDLING definition by the preprocessor */
    #if SELFLIB_STATUS_CHECK==STATUS_CHECK_FW
    #elif SELFLIB_STATUS_CHECK==STATUS_CHECK_SELFLIB
    #elif SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
    #else
        #error wrong SELFLIB_STATUS_CHECK definition
    #endif

    return ret;
}


/*============================================================================
*/
/* Function name:   SelfLib_BasFct_CopySection
*/
/* IN:              srcAdd:                 Source address
*/
/*                  destAdd:                destination address
*/
/*                  size:                   number of bytes to be copied
*/
/* OUT:             -
*/
/* Describtion:     Copies a number of bytes
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_Rom"
#elif SL_COMPILER == IAR
```

```
    #pragma location = "SelfLib_Rom"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_Rom"
#endif

u32 SelfLib_BasFct_CopySection( void *srcAdd, void *destAdd, u32 size )
{
    u08 *src, *dest;

    src = (u08*)srcAdd;
    dest = (u08*)destAdd;

    /* Copy Program execution address offset to the linked address */
    src = (u08*)( (u32)src + SelfLib_BasFct_ExeOffset() );

    /* Copy the section */
    for ( ; size>0; size-- )
    {
        (*dest++) = (*src++);
    }

    /* Align the destination address to word boundary */
    dest = (u08 *)( ( (u32)dest + 0x02 ) & (~0x03) );
    return ( (u32)dest );
}


/*============================================================================
*/
/* Function name:   SelfLib_Init
*/
/* IN:              destAddSelfLib:         Destination address of the
*/
/*                                          SelfLib sections to be copied
*/
/*                                          (e.g. internal RAM)
*/
/*                  frequency:              CPU operation frequency
*/
/*                                          Only req. for CZ6HSF devices!
*/
/*                                          Set to 0 otherwise
*/
/*                  copy:                   COPY: Copy the sections
*/
/*                                                ...ToRamUsrInt,
*/
/*                                                ...ToRamUsr
*/
/*                                                and ...ToRam to the
*/
/*                                                dest. adress
*/
/*                                          DONT_COPY: Don't copy sections
*/
```

```
/* OUT:             SELFLIB_OK              No SelfLib configuration error
*/
/*                  SELFLIB_ERR_PARAMETER   specific parameter does not
*/
/*                                          match the configuration in
*/
/*                                          SelflibSpecific.h
*/
/* Describtion:     Initializes the SelfLib and copies the the sections to
*/
/*                  their destination address (See application note self-
*/
/*                  programming of single voltage Flash for further
*/
/*                  information)
*/
/*=========================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_Rom"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_Rom"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_Rom"
#endif

u32  SelfLib_Init( void             *destAddSelfLib,
                   u32              frequency,
                   SECTION_COPY     sectionCopy )
{

    void *srcAdd,
         *destAdd;
    u32  size,
         ret;


      TF_CRLF;
      TF_3CHAR('I','N','I');
      TF_CRLF;

    pfctSelfLib_BasFct_BromFlashFunc = NULL;


/*=========================================================================*/
    /* Init SysData if required by the device
*/

/*=========================================================================*/
    ret = SelfLib_BasFct_SpecificInit( frequency );


/*=========================================================================*/
    /* Copy SelfLib locations to RAM if required
*/
```

```
/*====================================================================*/

    if( sectionCopy == COPY )
    {
        destAdd = destAddSelfLib;


/*====================================================================*/
        /* Copy the User interrupt sections, if the size is >0
*/

/*====================================================================*/
        SelfLib_BasFct_SectionAdd_ToRamUsrInt( (u32 *)&srcAdd, &size );
        if( size > 0 )
        {
            destAdd = (void *)SelfLib_BasFct_CopySection( srcAdd,
                                                          destAdd,
                                                          size );

            TF_TAB;
            TF_3CHAR('U', 's', '1');
          TF_HWORD((u32)srcAdd);
            TF_HWORD((u32)destAdd);
          TF_HWORD(size);
            TF_CRLF;
        }

        SelfLib_BasFct_SectionAdd_ToRamUsr( (u32 *)&srcAdd, &size );
        if( size > 0 )
        {
            destAdd = (u08 *)SelfLib_BasFct_CopySection( srcAdd, destAdd,
size );

            TF_TAB;
            TF_3CHAR('U', 's', 'r');
          TF_HWORD((u32)srcAdd);
            TF_HWORD((u32)destAdd);
          TF_HWORD(size);
            TF_CRLF;
        }


/*====================================================================*/
        /* Copy the SelfLib_ToRam section
*/

/*====================================================================*/
        SelfLib_BasFct_SectionAdd_ToRam( (u32 *)&srcAdd, &size );
        destAdd = (u08 *)SelfLib_BasFct_CopySection( srcAdd, destAdd, size );

        TF_TAB;
        TF_3CHAR('R', 'a', 'm');
        TF_HWORD((u32)srcAdd);
        TF_HWORD((u32)destAdd);
```

```
        TF_HWORD(size);
        TF_CRLF;


/*=====================================================================*/
        /* Copy the SelfLib_RomOrRam section
*/
        /* Only necessary if status polling is done by the user. In this
*/
        /* case the user self-programming control program shall be located
*/
        /* in the section ..._RomOrRam
*/

/*=====================================================================*/
          #if SELFLIB_STATUS_CHECK==STATUS_CHECK_USER
              SelfLib_BasFct_SectionAdd_RomOrRam( (u32 *)&srcAdd, &size );
            destAdd = (u08 *)SelfLib_BasFct_CopySection( srcAdd, destAdd,
size );

            TF_TAB;
            TF_3CHAR('R', 'o', 'R');
            TF_HWORD((u32)srcAdd);
            TF_HWORD((u32)destAdd);
            TF_HWORD(size);
            TF_CRLF;

        #else
            /* The pointer to the BROM interface function is required only,
*/
            /* if the section SelfLib_ToRam is copied and the rest is
*/
            /* executed on the original address. That means COPY and
*/
            /* not STATUS_CHECK_USER
*/
            /* Otherwise the interface is called direct (SelfLibCommands.c)
*/
            pfctSelfLib_BasFct_BromFlashFunc
                = (u32(*)(u32 functionNo, u32 param1, u32 param2, u32 param3,
                          SELFLIB_STATCHECK statCheck))
                              SelfLib_FctNewAddress(
                                  (void*)SelfLib_BasFct_BromFlashFunc,
                                  destAddSelfLib );
        #endif
    }

    TF_TAB;
      TF_3CHAR( 'p', 'F', 'F' );
      TF_HWORD( (u32)pfctSelfLib_BasFct_BromFlashFunc );
      TF_CRLF;

      return ret;
}
```

```
/*============================================================================
*/
/* Function name:   SelfLib_FctNewAddress
*/
/* IN:              addFct:                 Original address of the function
*/
/*                  destAddSelfLib:         Destination address of the
*/
/*                                          SelfLib sections to be copied by
*/
/*                                          SelfLib_Init
*/
/* OUT:             Execution address of the passed function
*/
/*                  0x00000000, if the function is outside the SelfLib
*/
/*                  sections
*/
/* Describtion:     Calculates the execution address (new address after
*/
/*                  copying sections by SelfLib_Init) of a function which
*/
/*                  original address is passed by addFct.
*/
/*                  !!! The function must be in the sections
*/
/*                      SelfLib_RomOrRam or in SelfLib_ToRamUsr !!!
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_Rom"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_Rom"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_Rom"
#endif

u32 SelfLib_FctNewAddress ( void *addFct,
                            void *destAddSelfLib )
{
    u32 secAdd;
    u32 secSize;
    u32 destAdd;
    u32 exeOff;

    TF_3CHAR('F','N','A');
      TF_CRLF;
      TF_TAB;
      TF_HWORD((u32)addFct);
      TF_HWORD((u32)destAddSelfLib);

    /* Add SelfLib_ToRamUsr1st section size */
    SelfLib_BasFct_SectionAdd_ToRamUsrInt( &secAdd, &secSize );
```

```
    secSize = ( secSize + 0x02 ) & (~0x03);     /* Align to word boundary */
    destAdd = (u32)destAddSelfLib + secSize;

    /* Check, if the function is in user section */
    SelfLib_BasFct_SectionAdd_ToRamUsr( &secAdd, &secSize );
    if( ( (u32)addFct >= secAdd ) && ( (u32)addFct < secAdd+secSize ) )
    {
        destAdd += ( (u32)addFct - secAdd );
    }
    else
    {
        secSize = ( secSize + 0x02 ) & (~0x03); /* Align to word boundary */
        destAdd += secSize;

        /* check, if the function is in the SelfLib_ToRam section */
        SelfLib_BasFct_SectionAdd_ToRam( &secAdd, &secSize );
        if( ( (u32)addFct >= secAdd ) && ( (u32)addFct < secAdd+secSize ) )
        {
            destAdd += ( (u32)addFct - secAdd );
        }
        else
        {
            secSize = ( secSize + 0x02 ) & (~0x03);/* Align to word boundary
*/
            destAdd += secSize;

            /* check, if the function is in the SelfLib_RomOrRam section */
            SelfLib_BasFct_SectionAdd_RomOrRam( &secAdd, &secSize );
            if( ( (u32)addFct >= secAdd ) && ( (u32)addFct < secAdd+secSize )
)
            {
                destAdd += ( (u32)addFct - secAdd );
            }
            else
            {
                destAdd = 0x00000000; /*Error function address not found */
            }
        }
    }

    /* If linked address is != execution address, consider execution */
    /* offset in the result */
    exeOff = SelfLib_BasFct_ExeOffset();
    destAdd -= exeOff;

    TF_HWORD( exeOff );
      TF_TAB;
      TF_HWORD((u32)destAdd );
      TF_CRLF;

    return destAdd;
}
```

```
/*============================================================================
*/
/* Function name:   SelfLib_GetInfo_LibVersion
*/
/* IN:              none
*/
/* OUT:             int       (r10)
*/
/* Description:     returns Version of Library
*/
/*============================================================================
*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_Rom"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_Rom"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_Rom"
#endif

u32 SelfLib_LibVersion( void )
{
    u32 ret;

    ret = LIBRARY_VERSION;

    TF_CRLF;
      TF_3CHAR('L','i','b');
    TF_3CHAR('V', 'e', 'r');
    TF_TAB;
    TF_HWORD( ret );
    TF_CRLF;

    return ret;
}
```

3.2.14 UserFuncInt.s

```
#
;===============================================================================
#  Project      = Selfprogramming Library Single Voltage Flash
#  Module       = SelfLibAsm.s85
#  Version      = 1.02
#  Date         = 21.03.2005
#
;===============================================================================
#                                    COPYRIGHT
#
;===============================================================================
#  Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
#
;===============================================================================
#  Purpose:
#               - Misc basic assembler functions required for the selfLib
#               - Device firmware interface (Trap entry )
```

```
#
================================================================================
#  Environment:
#               Devices:        V850E(S) Single Voltage SST Flash
#               IDE's:         GHS V3.5.1, GHS V4.0.5,
#                               IAR V2.3, IAR V3.10, NEC V2.61
#
================================================================================

#
================================================================================
#  Function name:   -
#  IN:              -
#  OUT:             -
#  Describtion:     Interrupt entry point
#                   Calls interrupt handler routine
#
================================================================================
.section "SelfLib_ToRamUsrInt.text",text
.globl _intEntry

_intEntry:
    nop
    nop
    jr  _userIntHdr
```

3.2.15 UserFunc.c

```
/*============================================================================
*/
/* Project      = Selfprogramming Library Single Voltage Flash
*/
/* Module       = userFunc.c
*/
/* Version      = 1.02
*/
/* Date         = 21.03.2005
*/
/*============================================================================
*/
/*                                  COPYRIGHT
*/
/*============================================================================
*/
/* Copyright (c) 2004 by NEC Electronics (Europe) GmbH. All rights reserved.
*/
/*============================================================================
*/
/* Purpose:
*/
/*              This module contains the user interrupt functions, that are
*/
/*              located in the SelfLib_eRam1st section.
*/
```

```
/*              The interrupt functions are called via hooks, located in the
*/
/*              module userFunc.s !!!
*/
/*=============================================================================
*/
/* Environment:
*/
/*              Devices:        V850E(S) Single Voltage SST Flash
*/
/*              IDE's:          GHS V3.5.1, GHS V4.0.5,
*/
/*                              IAR V2.3, IAR V3.10, NEC V2.61
*/
/*=============================================================================
*/

#include "selfprog.h"
#include "macrodriver.h"
#include "serial.h"

/*=============================================================================*/
/* Function name:   UserFunc                                                   */
/* IN:              -                                                          */
/* OUT:             -                                                          */
/* Describtion:     Simple routine, just toggling a pin or port. Called        */
/*                  by IntHdr                                                  */
/*=============================================================================*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRamUsr"
#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRamUsr"
#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRamUsr"
#endif

void ram_intsr1(void);
void ram_inttm00(void);

unsigned int ecr_val;               /* save location to get ECR value */
volatile unsigned int ram_int;      /* flag that interrupt happened in ram
service */

void userFunc ( void )
{
unsigned int vec;
      __asm("stsr 4,r6");                 /* get ECR to r6 */
      __asm("st.w r6,$_ecr_val");   /* save in variable */

      ram_int = 1;            /* set flag so ISRs can know if interrupt
happened in FSP mode */
      vec = ecr_val & 0x0000FFFF;   /* mask to EICC, interrupt condition code
*/
      switch (vec) {
       case 0x0100:           /* interrupt was INTTM00 */
```

```
            TM0IC00 = TM0IC00 & 0x7F;     /* clear interrupt flag (necessary
if RAM) */
            ram_inttm00();
            break;
       case 0x01C0:           /* interrupt was INTSR1 */
            SRIC1 = SRIC1 & 0x7F;         /* clear interrupt flag */
            ram_intsr1();
            break;
      }
}


/*=====================================================================*/
/* Function name:   UserFunc1_Int                                      */
/* IN:              -                                                  */
/* OUT:             -                                                  */
/* Describtion:     Interrupt routine for normal operation, calling the  */
/*                  user function                                      */
/*=====================================================================*/
#if   SL_COMPILER == GHS
    #pragma ghs section text=".SelfLib_ToRamUsr"
    __interrupt void userIntHdr ( void );
    #pragma intvect userIntHdr USERFUNC_INT_HDR

#elif SL_COMPILER == IAR
    #pragma location = "SelfLib_ToRamUsr"
    #pragma vector = USERFUNC_INT_HDR

#elif SL_COMPILER == NEC
    #pragma text "SelfLib_ToRamUsr"

#endif

__interrupt void userIntHdr ( void )
{
    userFunc();
}


extern int u1_dropcount;            /* count of dropped characters */

/* local variables */
extern char u1_rxbuf[U1_RXBUF_SIZE];      /* buffer to hold received
characters */
extern char * pu1_rxin;                                 /* pointer to put next
char in buffer */
extern char * pu1_rxout;                                      /* pointer to get
next char from buffer */
extern int u1_rxcount;                                  /* count of characters
in buffer */
extern char u1_xoff_sent;                                     /* 0 if no XOFF
issued, 1 if XOFF issued */
extern unsigned char last_ASIS1_err_val;        /* last error seen */

volatile int intsr1_count;
```

```
void ram_intsr1_push(char c)
{
      /* if room, put in buffer */
      if (u1_rxcount < U1_RXBUF_SIZE) {
            *pu1_rxin++ = c;  /* store in buffer */
            u1_rxcount++;           /* increment count */
            if (pu1_rxin == u1_rxbuf + U1_RXBUF_SIZE) {
                  pu1_rxin = u1_rxbuf;
            }
      } else {          /* if buffer is full, */
            u1_dropcount++;         /* count dropped characters */
      }
}

/* ISR in RAM for INTSR1 while Flash Self-programming */
void ram_intsr1( void )
{
unsigned char tmp;
char c;
      if (ram_int != 0) {
            /* if interrupt happened in RAM, spin right LED backwards to
indicate */
            ram_int = 0;
            tmp = ~PDLH;
            tmp = tmp >> 1;
            if (tmp == 0)
                  tmp = 0x40;
            PDLH = ~tmp;
            intsr1_count++;         /* keep count for debugging */
      }
      c = RXB1;
      ram_intsr1_push(c);
      /* send XOFF if buffer has reached this point and not sent yet */
      if (u1_xoff_sent == 0) {
            if (u1_rxcount == U1_XOFF_MARK) {
                  while ((ASIF1 & 0x02) != 0) {
                        /* while waiting for transmit buffer to be available,
*/
                        /* check for additional characters received, and
store */
                        if (SRIF1 == 1) {
                              c = RXB1;
                              SRIF1 = 0;
                              ram_intsr1_push(c);
                        }
                  }
                  TXB1 = XOFF;
                  u1_xoff_sent = 1;
            }
      }
}

extern volatile unsigned int milliseconds;
extern volatile int msecCount;
```

```
extern BOOL ledSpin;

/* ISR in RAM for INTTM00 while Flash Self-programming */
void ram_inttm00( void )
{
unsigned char tmp;
#if 1
      /* count down millisecond timer */
      if (milliseconds > 0)
            milliseconds--;
      /* if we are spinning the LED, count out and do it */
      if (ledSpin == MD_TRUE) {
            if (msecCount == 0) {
                  if (ram_int != 0) {
                        /* interrupt in RAM, spin left LED faster */
                        msecCount = 100;
                        tmp = ~PDH;
                        tmp = tmp << 1;
                        if (tmp == 0)
                              tmp = 1;
                        if (tmp == 0x40)
                              tmp = 1;
                        PDH = ~tmp;
                  } else {
                        /* interrupt not in RAM, spin right LED slower */
                        msecCount = 333;
                        tmp = ~PDLH;
                        tmp = tmp << 1;
                        if (tmp == 0)
                              tmp = 1;
                        if (tmp == 0x40)
                              tmp = 1;
                        PDLH = ~tmp;
                  }
            } else {
                  msecCount--;
            }
      }
#else /* old code */
      if (msecCount == 0) {
            msecCount = 100;
            tmp = ~PDH;
            tmp = tmp << 1;
            if (tmp == 0)
                  tmp = 1;
            if (tmp == 0x40)
                  tmp = 1;
            PDH = ~tmp;
      } else {
            msecCount--;
      }
#endif
}
```

3.3 UA_1 User Program

The UA_1 and UA_2 user application programs share the same files, except for the file containing the main() function. This is ua_1.c for UA_1, and ua_2.c for UA_2.

3.3.1 Crte_8000.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_8000.s -  start up module for ca850(V850E)
#    modified from crtE.s to add additional section with branch to main at
0x8000
#
# 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

#-------------------------------------------------------------------------------
-
#     ENTRY handler
#-------------------------------------------------------------------------------
-

      .section        "ENTRY.text", 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 -------------------#
#                                                 #
```

3.3.2 Ua_1.c

```
/* UA_1.C - User Application #1 for M-V850ES-KJ1 CPU board */
/*    on M-Station base board, for flash self-programming AppNote */
/*  06/21/2006 */

/* pragma */

#pragma ioreg
/* #pragma di */
/* #pragma ei */
/* #pragma nop */
/* #pragma asm */

/* include files */

#include    "led_vkj1.h"      /* definitions for LED output routines */
#include    "sw_vkj1.h"       /* definitions for switch input routines */

/* function prototypes in this file */

void hdwinit( void );         /*initialize */
void main( void );                  /* main loop */

/* defines */
#define COUNT_INCR      1           /* amount to count up */
#define COUNT_DECR      2           /* amount to count down */
#define COUNT_MAX 15          /* maximum count */

/* RAM */

/* void hdwinit(void) - set up initial hardware */
void hdwinit ()
{
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");

      __asm("ei");                        /* enable interrupt */
}

extern unsigned long _S_romp; /* label used for ROM to RAM transfer */
```

```
/* main function */
void main( void )
{
unsigned char count;
unsigned char swval;

      _rcopy(&_S_romp,-1);    /* copy ROM data to RAM */

      hdwinit();        /* initialize hardware */

      count = 0;

      /* initialize board hardware */
      led_init();                   /* set up LEDs */
      sw_init();                    /* set up switches */
      sw_set_debounce(64);    /* initial debounce value */

      led_dig_bcd(count);           /* display count */

      /* main loop for test selection and execution */
      while(1){
            swval = sw_get(); /* get a debounced switch value */
            switch (swval) {
            case SW_LU_RU:
                  break;            /* do nothing if both up */

            case SW_LD_RU:    /* left switch (SW2) down, count down */
                  count = count - COUNT_DECR;
                  if (count > COUNT_MAX) {
                        /* went below zero */
                        count = COUNT_MAX;
                  }
                  led_dig_bcd(count);           /* display new count */
                  while (SW_LU_RU != sw_get())  /* wait for switches up */
                        ;
                  break;

            case SW_LU_RD:    /* right switch (SW3) down, count up */
                  count = count + COUNT_INCR;
                  if (count > COUNT_MAX) {
                        count = 0;
                  }
                  led_dig_bcd(count);           /* display new count */
                  while (SW_LU_RU != sw_get())  /* wait for switches up */
                        ;
                  break;

            case SW_LD_RD:    /* both switches down */
                  break;            /* ignore at this level */
            } /* end switch (swval) */
      } /* end while(1) */
}
```

3.3.3 Sw_vkj1.h

```
/* sw_vkj1.h      */
/*    header for M-V850ES-KJ1 CPU board for base board switch reading */

#ifndef _SW_VKJ1_H
#define _SW_VKJ1_H

/*****************************************************************/
/* 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 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 deboune count */

#endif      /* _SW_VKJ1_H */
```

3.3.4 Sw_vkj1.c

```
/*    sw_vkj1.c - routines for switch input                     */
/*    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)                        */

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

#include "sw_vkj1.h"

/* 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 */

/*    void sw_init(void) */
/*          set up ports for switch input */
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 */
}

/* unsigned char sw_chk(void) */
/*    return input from switches, undebounced */
unsigned char sw_chk(void)
{
      return P9L & 0x30;
}

/* void sw_set_debounce(unsigned char count) */
/*    set the debounce counter value */
void sw_set_debounce(unsigned char count)
{
      sw_deb_value = count;   /* set new debounce counter value */
      sw_deb_count = count;   /* set counter to max */
}

/* unsigned char sw_get(void) */
/*    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, and returns quickly */

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;
}
```

3.3.5 Led_vkj1.h

```
/* led_vkj1.h     */
/*    header for M-V850ES-KJ1 CPU board for LED digit display */
/* Version 1.1    05-08-2006
      */

#ifndef _LED_VKJ1_H
#define _LED_VKJ1_H

/*********************************************************************/
/* 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

/*********************************************************************/
/* Export functions                                                  */
/*********************************************************************/
extern void led_init(void);                                  /* init ports for
LED output  */
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
      */
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_dig(unsigned char num);              /* display number as
hex         */
extern void led_dig_bcd(unsigned char bcdnum);  /* display number as BCD
      */

extern void led_dp_left(unsigned char on);           /* turn on or off left
DP */
extern void led_dp_right(unsigned char on);          /* turn on or off right
DP */

#endif      /* _LED_KJ1_H */
```

3.3.6 Led_vkj1.c

```
/*    led_vkj1.c - routines for LED                  */
/*    for M-V850ES-KJ1 CPU board on M-Station base board    */
/*  Version:      1.1   05-08-2006                              */
/*  Version:    1.2 06-08-2006 added dp routines    */

/*          PDL8-PDL15 = output to right digit (LED2)       */
/*          PDH0-PDH7  = output to left digit  (LED1)       */

/*    To connect ports to LEDs on M-Station 1.1, make the
```

```
      following jumper connections between ROW1 and ROW2.
      To connect ports to LEDs on M-Station 2, make sure
      the default SBxx connections are inserted.
      Port  LED         M-Station 1.1     M-Station 2.2
      ----  ---         -------------     -------------
      PDL8        2-A         R1.25 - R2.25     SB27
      PDL9        2-B         R1.26 - R2.26     SB28
      PDL10       2-C         R1.27 - R2.27     SB29
      PDL11       2-D         R1.28 - R2.28     SB30
      PDL12       2-E         R1.29 - R2.29     SB31
      PDL13       2-F         R1.30 - R2.30     SB32
      PDL14       2-G         R1.31 - R2.31     SB33
      PDL15       2-DP  R1.32 - R2.32     SB34

      PDH0        1-A         R1.17 - R2.17     SB35
      PDH1        1-B         R1.18 - R2.18     SB36
      PDH2        1-C         R1.19 - R2.19     SB37
      PDH3        1-D         R1.20 - R2.20     SB38
      PDH4        1-E         R1.21 - R2.21     SB39
      PDH5        1-F         R1.22 - R2.22     SB40
      PDH6        1-G         R1.23 - R2.23     SB41
      PDH7        1-DP  R1.24 - R2.24     SB42
*/

/* NOTE: on M-Station Base V1.0 prototype, PDH0-PDH7 are    */
/*  located at ROW4.1-8, and need to be wirewrapped to      */
/*  connect to ROW2.17-24 to drive LED1.                    */

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

#include "led_vkj1.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 */
};

/*    void led_init(void) */
/*          set up ports for display of LED digits */
```

```
void led_init(void)
{
#if 1 /* ports initialized in Port_Init() by Applilet */
      PMCDH = 0x00;           /* set port DH to port mode */
      PMDH = 0x00;            /* set port DH to output */

      PMCDLH = 0x00;          /* set port DL high 8-bits to port mode */
      PMDLH = 0x00;           /* set port DL high 8-bits to output */
#endif
}

/* void led_out_right(unsigned char val) */
/*          output raw data to right LED */
void led_out_right(unsigned char val)
{
      PDLH = val;
}

/* void led_out_left(unsigned char val) */
/*          output raw data to left LED */
void led_out_left(unsigned char val)
{
      PDH = val;
}

/* void led_dp_left(unsigned char on) */
/* turn on or off left DP */
void led_dp_left(unsigned char on)
{
      if (on == 0)
            PDH = PDH | 0x80; /* set bit 7 high to turn off */
      else
            PDH = PDH & 0x7f; /* set bit 7 low to turn on */
}

/* void led_dp_right(unsigned char on) */
/* turn on or off right DP */
void led_dp_right(unsigned char on)
{
      if (on == 0)
            PDLH = PDLH | 0x80;     /* set bit 7 high to turn off */
      else
            PDLH = PDLH & 0x7f;     /* set bit 7 low to turn on */

}

/* void led_dig_right(unsigned char num) */
/*    display number in right LED */
void led_dig_right(unsigned char num)
{
      if (num > 0x0F) {
            led_out_right(LED_PAT_BLANK);
            return;
      }
      led_out_right(dig_tab[num]);
```

```
}

/* void led_dig_left(unsigned char num) */
/*    display number in left LED */
void led_dig_left(unsigned char num)
{
      if (num > 0x0F) {
            led_out_left(LED_PAT_BLANK);
            return;
      }
      led_out_left(dig_tab[num]);
}

/* void led_dig(unsigned char num) */
/*          display number as hex digits */
/*          num - number to display */
/*                bits 0-3 in right digit */
/*                bits 4-7 in left digit */
void led_dig(unsigned char num)
{
      led_out_right(dig_tab[num & 0x0F]);
      led_out_left(dig_tab[(num >> 4) & 0x0F]);
}

/* void led_dig_bcd(unsigned char bcdnum) */
/*          display two digits of BCD coded bcdnum */
/*          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 */
void led_dig_bcd(unsigned char bcdnum)
{
unsigned char tens_dig;

      if (bcdnum > 99) {
            led_out_right(LED_PAT_BLANK); /* display both digits blank */
            led_out_left(LED_PAT_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++;       /* while counting up the tens digit */
      } while (bcdnum >= 10);
      /* now tens_dig has ten's place */
      /* and bcdnum has remainder */
      led_out_right(dig_tab[bcdnum]);
```

```
      led_out_left(dig_tab[tens_dig]);
}
```

3.3.7 UA_8000.dir

```
#   Sample link directive file (not use RTOS/use internal memory only)
#
#   Copyright (C) NEC Electronics Corporation 2002
#   All rights reserved by NEC Electronics Corporation.
#
#   This is a sample file.
#   NEC Electronics assumes no responsibility for any losses incurred by
customers or
#   third parties arising from the use of this file.
#
#   Generated      : PM+ V6.10  [ 4 Apr 2005]
#   Sample Version : E1.00b [12 Jun 2002]
#   Device         : uPD70F3318Y (C:\Program Files\NEC Electronics
Tools\DEV\DF3318Y.800)
#   Internal RAM   : 0x3ffb000 - 0x3ffefff
#
#   NOTICE:
#        Allocation of SCONST, CONST and TEXT depends on the user program.
#
#        If interrupt handler(s) are specified in the user program then
#        the interrupt handler(s) are allocated from address 0 and
#        SCONST, CONST and TEXT are allocated after the interrupt handler(s).

# modified to load at 0x8000, and use RAM starting as 0x3FFD000
#SCONST  : !LOAD ?R {
SCONST  : !LOAD ?R V0x0008000 {
        .sconst         = $PROGBITS     ?A .sconst;
};

#CONST   : !LOAD ?R {
CONST   : !LOAD ?R V0x0008000 {
        .const          = $PROGBITS     ?A .const;
};

ENTRY    : !LOAD ?RX V0x0008000 {
        ENTRY.text      = $PROGBITS    ?AX ENTRY.text;
};

# Begin TEXT at word boundary after ENTRY
#TEXT    : !LOAD ?RX {
TEXT    : !LOAD ?RX V0x0008010 {
        .pro_epi_runtime = $PROGBITS    ?AX .pro_epi_runtime;
        .text            = $PROGBITS    ?AX .text;
};

#SIDATA  : !LOAD ?RW V0x3ffb000 {
SIDATA  : !LOAD ?RW V0x3ffd000 {
        .tidata.byte    = $PROGBITS     ?AW .tidata.byte;
        .tibss.byte     = $NOBITS       ?AW .tibss.byte;
        .tidata.word    = $PROGBITS     ?AW .tidata.word;
```

```
        .tibss.word     = $NOBITS       ?AW .tibss.word;
        .tidata         = $PROGBITS     ?AW .tidata;
        .tibss          = $NOBITS       ?AW .tibss;
        .sidata         = $PROGBITS     ?AW .sidata;
        .sibss          = $NOBITS       ?AW .sibss;
};

#DATA    : !LOAD ?RW V0x3ffb100 {
DATA    : !LOAD ?RW V0x3ffd100 {
        .data           = $PROGBITS     ?AW  .data;
        .sdata          = $PROGBITS     ?AWG .sdata;
        .sbss           = $NOBITS       ?AWG .sbss;
        .bss            = $NOBITS       ?AW  .bss;
};

__tp_TEXT @ %TP_SYMBOL;
__gp_DATA @ %GP_SYMBOL &__tp_TEXT{DATA};
__ep_DATA @ %EP_SYMBOL;
```

3.4 UA_2.c User Program

The UA_1 and UA_2 user application programs share the same files, except for the file containing the main() function. This is ua_1.c for UA_1, and ua_2.c for UA_2. Only ua_2.c is shown here; see the listings for UA_1 for the other files.

```
/* UA_2.C - User Application #2 for M-V850ES-KJ1 CPU board */
/*    on M-Station base board, for flash self-programming AppNote */
/*  06/21/2006 */

/* pragma */

#pragma ioreg
/* #pragma di */
/* #pragma ei */
/* #pragma nop */
/* #pragma asm */

/* include files */

#include    "led_vkj1.h"      /* definitions for LED output routines */
#include    "sw_vkj1.h"       /* definitions for switch input routines */

/* function prototypes in this file */

void hdwinit( void );         /*initialize */
void main( void );                  /* main loop */

/* defines */
#define COUNT_INCR      2           /* amount to count up */
#define COUNT_DECR      3           /* amount to count down */
#define COUNT_MAX 37          /* maximum count */

/* RAM */
```

```
/* void hdwinit(void) - set up initial hardware */
void hdwinit ()
{
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");

      __asm("ei");                         /* enable interrupt */
}

extern unsigned long _S_romp; /* label used for ROM to RAM transfer */

/* main function */
void main( void )
{
unsigned char count;
unsigned char swval;

      _rcopy(&_S_romp,-1);    /* copy ROM data to RAM */

      hdwinit();        /* initialize hardware */

      count = 0;

      /* initialize board hardware */
      led_init();                  /* set up LEDs */
      sw_init();                   /* set up switches */
      sw_set_debounce(64);    /* initial debounce value */

      led_dig(count);              /* display count */

      /* main loop for test selection and execution */
      while(1){
            swval = sw_get(); /* get a debounced switch value */
            switch (swval) {
            case SW_LU_RU:
                  break;            /* do nothing if both up */

            case SW_LD_RU:    /* left switch (SW2) down, count down */
                  count = count - COUNT_DECR;
                  if (count > COUNT_MAX) {
```

```
                        /* went below zero */
                        count = COUNT_MAX;
                  }
                  led_dig(count);         /* display new count */
                  while (SW_LU_RU != sw_get())  /* wait for switches up */
                        ;
                  break;

            case SW_LU_RD:    /* right switch (SW3) down, count up */
                  count = count + COUNT_INCR;
                  if (count > COUNT_MAX) {
                        count = 0;
                  }
                  led_dig(count);         /* display new count */
                  while (SW_LU_RU != sw_get())  /* wait for switches up */
                        ;
                  break;

            case SW_LD_RD:    /* both switches down */
                  break;            /* ignore at this level */
            } /* end switch (swval) */
      } /* end while(1) */
}
```

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

The information in this document is current as of October 2006. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.

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).