Application Note

AN2204//D
Rev. 1, 2/2002

Fast NVM Programming for the MC9S12DP256

by **Stuart Robb**
Strategic Marketing & Systems (Europe),
Transportation and Standard Products Group

Introduction

In some applications, it may be necessary to program a number of bytes of data in non-volatile memory (NVM) at short notice and with the programming to be completed within a minimum time. In an automotive application for example, an application may be notified of a vehicle crash event and the application may be required to immediately save diagnostic data. The crash event may cause the electronic control unit (ECU) power supply to be disrupted either accidentally by mechanical damage to the wiring harness, or by design, by controlled switching of the 12V supply. In these cases, the ECU remains powered only by charge stored in capacitors contained within the ECU. The supply to the microcontroller (MCU) may remain above a minimum operating value for only a few milliseconds after the crash event is detected, and this will constitute the time available to save diagnostic data.

This application note presents two software routines written in the C programming language which will program a number of bytes of data to non-volatile memory on the MC9S12DP256 MCU in the minimum time required. One routine writes data to EEPROM and the other routine writes data to Flash memory. 32 bytes of data are typically programmed into EEPROM in under 700µs and into Flash memory in under 400µs.

MC9S12DP265

The MC9S12DP256 microcontroller unit is a high performance 16-bit CISC device composed of standard on-chip peripherals including a 16-bit central processing unit (CPU), 256k bytes of Flash memory, 12k bytes of RAM and 4k bytes of EEPROM. The MC9S12DP256 operates from 5V with an internal bus frequency of up to 25MHz. All internal memory can be accessed in a single bus cycle.

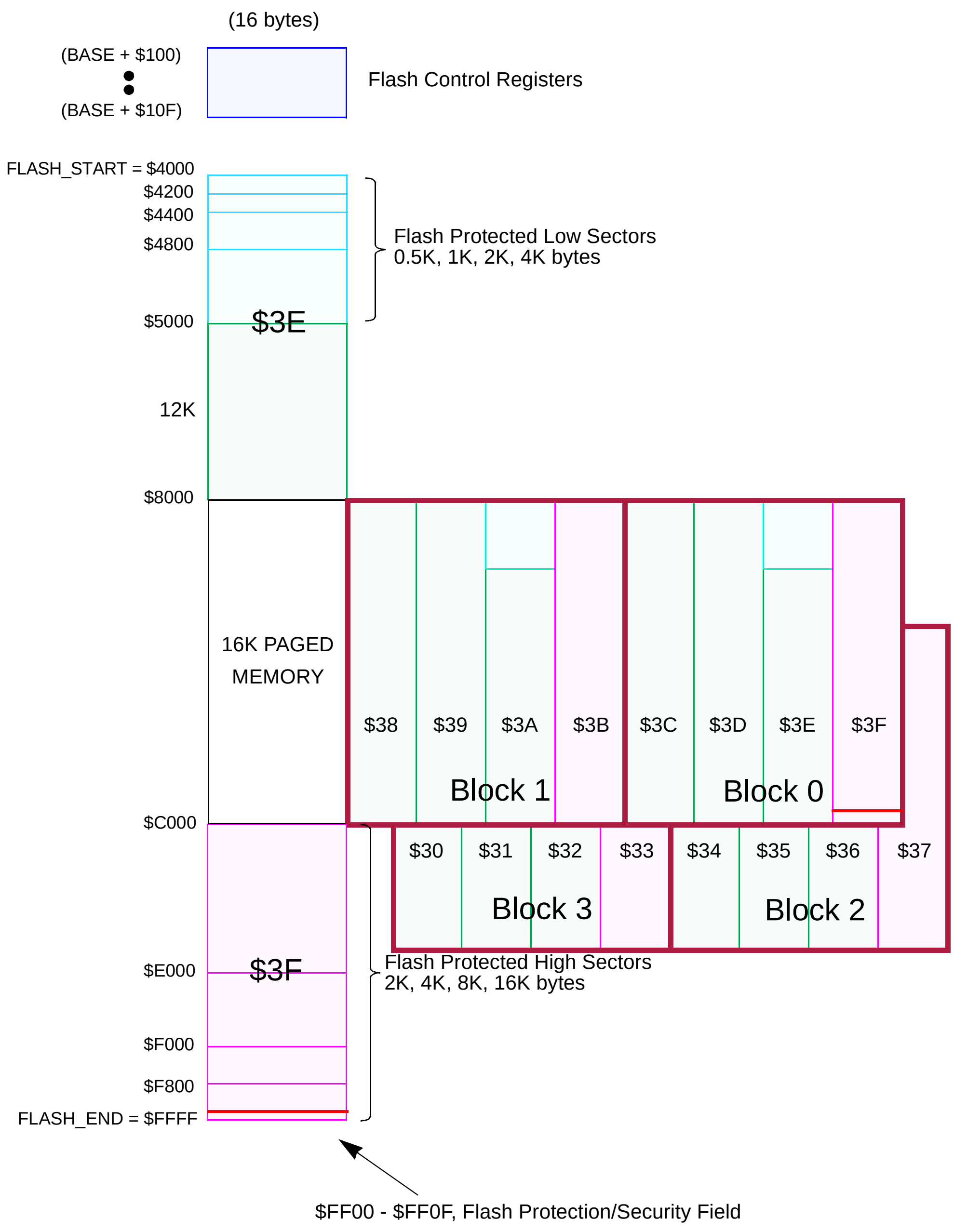
EEPROM

The MC9S12DP256 microcontroller includes 4k bytes of on-chip EEPROM. The EEPROM block uses a small sector Flash memory to emulate EEPROM functionality. The EEPROM block is organised as 2048 rows of 2 bytes. The EEPROM block’s minimum erase sector is 2 rows (4 bytes). Programming is performed on complete rows, i.e. on aligned words. The high voltage required to program and erase is generated internally by an on-chip charge pump. The program and erase operations are controlled by a state machine. The state machine requires a clock with a frequency of 150kHz to 200kHz which is obtained by dividing the oscillator clock by a programmable prescaler. Thereafter the state machine takes care of all timing requirements, the user has only to initiate the required commands. The rules for choosing the prescaler divider and the correct command sequences are described in the EEPROM Block User Guide.

The EEPROM block cannot be read whilst it is being programmed or erased.

Flash Memory

The MC9S12DP256 microcontroller includes 256k bytes of on-chip Flash memory. The Flash array is organised as 4 blocks of 64k bytes. Each block is organised as 1024 rows of 64 bytes. The Flash block’s minimum erase sector size is 8 rows (512 bytes). Each Flash block is subdivided into four pages of 16k bytes each. Each page is accessed by the CPU though a page "window" at $8000 to $BFFF by writing the appropriate page value to the PPAGE register. In addition, two pages are permanently located in the memory map and are described as "non-paged" memory. These are page $3E at $4000 to $7FFF and page $3F at $C000 to $FFFF.



Four Blocks of Flash IP

Note: $30-$3F correspond to the PPAGE register content

Figure 1. 256K Flash Memory Map

Programming is performed on aligned words. The high voltage required to program and erase is generated internally by on-chip charge pumps. The program and erase operations are controlled by a state machine. The state machine requires a clock with a frequency of 150kHz to 200kHz which is obtained by dividing the oscillator clock by a programmable prescaler. Thereafter the state machine takes care of all timing requirements, the user has only to initiate the required commands. The rules for choosing the prescaler divider and the correct command sequences are described in the Flash Block User Guide.

A Flash block cannot be read whilst it is being programmed or erased. Therefore the program and erase function cannot be executed from the flash block which is being operated on. If the program/erase function is located in non-paged memory ($4000 to $7FFF and $C000 to $FEFF) then pages $30 to $3B can be programmed/erased. Alternatively, if the program/erase function is located in paged memory (pages $30 to $3B) then only non-paged memory can be programmed/erased. In the latter case it is very important to disable all interrupts when programming or erasing, as the interrupt vectors are located in non-paged memory and will not be accessible. This is the approach taken in the code example provided in this application note. In order to be able to program/erase all Flash without restriction, the program/erase function must be copied to and executed from a non-flash memory location, RAM or EEPROM for example.

Software Description

stdtypes.h

This header file contains type definitions for the basic data types as well as macros for TRUE and FALSE.

mcucfg.h

This header file contains definitions for the following macros:

OSCCLK_FREQ_KHZ	This is used to define the frequency of the crystal oscillator in units of kHz. This constant must be defined as a long type by appending 'L' to the value.
REFDV	This is used to define the value which is put in the CRG Reference Divider Register.
SYNR	This is used to define the value which is put in the CRG Synthesizer Register.
BUSCLK_FREQ_KHZ	This calculates the bus clock frequency in kHz from the values of OSCCLK_FREQ_KHZ, REFDV and SYNR. Note that this calculation is correct *only* if the PLL is used. If the PLL is not used, the bus frequency is half the oscillator frequency.
REG_BASE	This is used to define the base address of the registers. The default value is 0.

s12_eectl.h

This header file contains typedefs for all of the EEPROM registers, macro for register bits and prototypes for the functions in progEeprom.c. It also contains the macro for EECLK_PRESCALER. This calculates the required value for the EEPROM Clock Divider Register from the values defined in mcucfg.h. OSCCLK_FREQ_KHZ must be defined as a long integer type for this macro to work correctly.

s12_fectl.h

This header file contains typedefs for all of the Flash registers, macro for register bits and prototypes for the functions in progFlash.c. It also contains the macro for FCLK_PRESCALER. This calculates the required value for the Flash Clock Divider Register from the values defined in mcucfg.h. OSCCLK_FREQ_KHZ must be defined as a long integer type for this macro to work correctly.

ProgEeprom.c

This file creates the static variable 'eeprom' of type tEEPROM which is used to access the EEPROM registers, and contains the two functions ConfigECLKDIV and ProgEeprom.

ConfigECLKDIV

Arguments: none

Return: none

Description: this function simply writes the value calculated by the EECLK_PRESCALER macro to the EEPROM Clock Divider Register if the register has not previously been written. The EEPROM Clock Divider Register is a 'write once' register and its unwritten status is indicated by the EDIVLD bit.

NOTE: *This function must be called once before the ProgEeprom function is called.*

ProgEeprom

Arguments: progAdr: pointer to the start of the destination EEPROM location to be programmed.
bufferPtr: pointer to the start of the source data.
size: number of words to be programmed.

Return: FAIL: if progAdr does not point to an aligned word, or the ACCERR bit or PVIOL bit is set during programming. Refer to the EEPROM Block User Guide for conditions under which the error flags become set.
PASS: if not FAIL.

Description: This function programs the specified number of words to EEPROM. EEPROM must be programmed in aligned words, so the progAdr pointer is first checked for an even address. After clearing the error flags in the EEPROM Status Register, the routine then enters a loop which programs one word at a time until either all words have been programmed or an error flag becomes set. Refer to the EEPROM Block User Guide for conditions under which the error flags become set.

NOTE: *This function does not check whether the EEPROM is erased before programming, nor does it verify that programming has been successful.*

ProgFlash.c

This file creates the static variable 'flash' of type tFLASH which is used to access the Flash registers, and contains the two functions ConfigFCLKDIV and ProgFlash.

ConfigFCLKDIV	Arguments:	none
	Return:	none
	Description:	this function simply writes the value calculated by the FCLK_PRESCALER macro to the Flash Clock Divider Register if the register has not previously been written. The Flash Clock Divider Register is a 'write once' register and its unwritten status is indicated by the FDIVLD bit.

NOTE: *This function must be called once before the ProgFlash function is called.*

ProgFlash	Arguments:	progAdr: pointer to the start of the destination Flash location to be programmed. This is a 16-bit pointer so the Flash destination must be un-paged, i.e. 0x4000 to 0x7FFE or 0xC000 to 0xFEFE bufferPtr: pointer to the start of the source data. size: number of words to be programmed.
	Return:	FAIL: if progAdr does not point to an aligned word, or the ACCERR bit or PVIOL bit is set during programming. Refer to the Flash Block User Guide for conditions under which the error flags become set. PASS: if not FAIL.
	Description:	This function programs the specified number of words to Flash. Flash must be programmed in aligned words, so the progAdr pointer is first checked for an even address. Because this function programs Flash Block 0 which contains the interrupt vectors, interrupts must be masked during programming. An inline assembly statement first copies the Condition Codes Register to a local variable and then sets the I Mask bit. After clearing the error flags in all four of the Flash Status Registers, the routine then enters a loop which programs one word at a time until either all words have been programmed or an error flag becomes set. Refer to the Flash Block User Guide for conditions under which the error flags become set. The final action before returning is to restore the Condition Codes Register to its original value.

NOTE: *This function does not check whether the Flash is erased before programming, nor does it verify that programming has been successful. This function must be located in pages $30 to $3B, i.e. not in Flash Block 0, the Flash array which is programmed.*

Acronyms

CISC	Complex Instruction Set Computer
CPU	Central Processing Unit
ECU	Electronic Control Unit
EEPROM	Electrically Eraseable Programmable Read Only Memory
MCU	MicroController Unit
NVM	Non-Volatile Memory
RAM	Random Access Memory

Code Listing

```
/******************************************************************************
                                                 Copyright (c)
File Name                :      $RCSfile: stdtypes.h,v $

Engineer                 :      $Author: r27624 $

Location                 :      EKB

Date Created             :      28/06/2001

Current Revision         :      $Revision: 1.0 $

*******************************************************************************
Freescale reserves the right to make changes without further notice to any
product herein to improve reliability, function or design. Freescale does
not assume any liability arising out ot the application or use of any
product, circuit, or software described herein; neither does it convey any
license under its patent rights nor the rights of others. Freescale products
are not designed, intended, or authorized for use as components in systems
intended for surgical implant into the body, or other applications intended
to support life, or for any other application in which the failure of the
Freescale product could create a situation where personal injury or death may
occur. Should Buyer purchase or use Freescale products for any such unintended
or unauthorized application, Buyer shall idemnify and hold Freescale and its
officers, employees, subsidiaries, affiliates, and distributors harmless
against all claims costs, damages, and expenses, and reasonable attorney fees
arising out of, directly or indirectly, any claim of personal injury or death
associated with such unintended or unathorized use, even if such claim alleges
that Freescale was negligent regarding the design or manufacture of the part.
Freescale and the Freescale logo* are registered trademarks of Freescale Semiconductor, Inc.
******************************************************************************/

#ifndef STDTYPES_H
#define STDTYPES_H                /* this file */


/******************************************************************************
These defines allow for easier porting to other compilers. If porting change
these defines to the required values for the chosen compiler.
******************************************************************************/
typedef unsigned char          UINT8;           /* unsigned 8-bit */
typedef unsigned int           UINT16;          /* unsigned 16-bit */
typedef unsigned long          UINT32;          /* unsigned 32-bit */
typedef signed char            INT8;            /* signed 8-bit */
typedef int                    INT16;           /* signed 16-bit */
typedef long int               INT32;           /* signed 32-bit */
```

```
/*****************************************************************************
Standard Definitions
*****************************************************************************/

#ifndef FALSE
#define FALSE 0
#endif

#ifndef TRUE
#define TRUE 1
#endif

/*****************************************************************************
Standard Enumerations
*****************************************************************************/


#endif          /* End of Header file !defined */
```

```
/*************************************************************************
                                                 Copyright (c)
File Name                :      $RCSfile: mcucfg.h,v $

Engineer                 :      $Author: r27624 $

Location                 :      EKB

Date Created             :      05/07/2001

Current Revision         :      $Revision: 1.0 $

*************************************************************************
Freescale reserves the right to make changes without further notice to any
Product herein to improve reliability, function or design. Freescale does not
assume any liability arising out of the application or use of any product,
circuit, or software described herein; neither does it convey any license
under its patent rights nor the rights of others. Freescale products are not
designed, intended, or authorized for use as components in systems intended for
surgical implant into the body, or other applications intended to support life,
or for any other application in which the failure of the  Freescale product
could create a situation where personal injury or death may occur. Should
Buyer purchase or use Freescale products for any such unintended or
unauthorized application, Buyer shall idemnify and hold Freescale and its
officers, employees, subsidiaries, affiliates, and distributors harmless
against all claims costs, damages, and expenses, and reasonable attorney fees
arising out of, directly or indirectly, any claim of personal injury or death
associated with such unintended or unauthorized use, even if such claim alleges
that Freescale was negligent regarding the design or manufacture of the part.
Freescale and the Freescale logo* are registered trademarks of Freescale Semiconductor, Inc.
************************************************************************/

#ifndef MCUCFG_H
#define MCUCFG_H

/************************* System Include Files *****************************/

/************************* Project Include Files ****************************/
#include "stdtypes.h"

/************************* typedefs *****************************************/

/************************* Extern Variables *********************************/


/************************* #Defines *****************************************/
/* Oscillator frequency in kHz */
#define OSCCLK_FREQ_KHZ         4000L           /* must have "L" appended for
F/ECLK_PRESCALER macro */
/* register values for PLL */
#define REFDV                   0               /* fref = fosc / (REFDV+1) */
#define SYNR                    5               /* fbus = fref*(SYNR+1) */
/* Bus frequency in kHz */
#define BUSCLK_FREQ_KHZ         OSCCLK_FREQ_KHZ*(SYNR+1)/(REFDV+1)
```

```
/* register base address */
#define REG_BASE                0x0000

/************************* Macros *******************************************/

/************************* Prototypes ***************************************/


#endif          /* End of Header file !defined    */
```

```
/******************************************************************************
                                                  Copyright (c)
File Name                 :     $RCSfile: s12_eectl.h,v $

Engineer                  :     $Author: r27624 $

Location                  :     EKB

Date Created              :     28/06/2001

Current Revision          :     $Revision: 1.0 $

******************************************************************************
Freescale reserves the right to make changes without further notice to any
Product herein to improve reliability, function or design. Freescale does not
assume any liability arising out of the application or use of any product,
circuit, or software described herein; neither does it convey any license
under its patent rights nor the rights of others. Freescale products arenot
designed, intended, or authorized for use as components in systems intended for
surgical implant into the body, or other applications intended to support life,
or for any other application in which the failure of the  Freescale product
could create a situation where personal injury or death may occur. Should
Buyer purchase or use Freescale products for any such unintended or
unauthorized application, Buyer shall idemnify and hold Freescale and its
officers, employees, subsidiaries, affiliates, and distributors harmless
against all claims costs, damages, and expenses, and reasonable attorney fees
arising out of, directly or indirectly, any claim of personal injury or death
associated with such unintended or unauthorized use, even if such claim alleges
that Freescale was negligent regarding the design or manufacture of the part.
Freescale and the Freescale logo* are registered trademarks of Freescale Semiconductor, Inc.
*****************************************************************************/

#ifndef EECTL_H
#define EECTL_H

/************************** System Include Files *****************************/

/************************** Project Include Files ****************************/
#include "stdtypes.h"

/************************** typedefs *****************************************/
typedef union uECLKDIV
  {
  UINT8         byte;
  struct
    {
    UINT8 ediv           :6;                     /*clk divider */
    UINT8 ediv8          :1;                     /*clk /8 prescaler enable */
    UINT8 edivld         :1;                     /*clock divider loaded flag */
    }bit;
  }tECLKDIV;

typedef union uECNFG
  {
```

```
  UINT8         byte;
  struct
    {
    UINT8                :4;                     /*not used */
    UINT8 eswai          :1;                     /*eeprom stopped in wait mode */
    UINT8                :1;                     /*not used */
    UINT8 ccie           :1;                     /*command complete interrupt enable */
    UINT8 cbeie          :1;                     /*command buffer empty interrupt enable*/
    }bit;
  }tECNFG;

typedef union uEPROT
  {
  UINT8         byte;
  struct
    {
    UINT8 ep             :3;                     /*protection block size: (ep+1)*64 bytes */
    UINT8 epdis          :1;                     /*protection disable */
    UINT8                :3;                     /*contain value of equivalent bits in
protection byte */
    UINT8 eopen          :1;                     /*open block for program/erase */
    }bit;
  }tEPROT;

typedef union uESTAT
  {
  UINT8         byte;
  struct
    {
    UINT8                :2;                     /*not used */
    UINT8 blank          :1;                     /*blank verify flag */
    UINT8                :1;                     /*not used */
    UINT8 accerr         :1;                     /*access error flag */
    UINT8 pviol          :1;                     /*protection violation flag */
    UINT8 ccif           :1;                     /*command complete interrupt flag */
    UINT8 cbeif          :1;                     /*command buffer empty interrupt flag */
    }bit;
  }tESTAT;

typedef union uECMD
  {
  UINT8         byte;
  struct
    {
    UINT8 mass           :1;                     /*mass erase enable*/
        UINT8            :1;                     /*not used */
    UINT8 erver          :1;                     /*erase verify enable */
        UINT8            :2;                     /*not used */
    UINT8 prog           :1;                     /*word programming */
    UINT8 erase          :1;                     /*erase control */
        UINT8            :1;                     /*not used */
    }bit;
  }tECMD;

typedef struct                                   /*eeprom datastructure */
```

```
  {
  volatile tECLKDIV             eclkdiv;         /*eeprom clock divider register */
                UINT8           rsvee1[2];       /*reserved */
  volatile tECNFG               ecnfg;           /*eeprom configuration register */
  volatile tEPROT               eprot;           /*eeprom protection register */
  volatile tESTAT               estat;           /*eeprom status register */
  volatile tECMD                ecmd;            /*eeprom command buffer & status register */
           UINT8                rsvee2[5];       /*reserved */
  }tEEPROM;


/************************* Extern Variables **********************************/


/************************* #Defines ******************************************/
#define EDIV8            0x40   /*bit masks      */
#define EDIVLD           0x80

#define ESWAI            0x10   /*bit masks      */
#define CCIE             0x40
#define CCBIE            0x80

#define EP0              0x01   /*bit masks      */
#define EP1              0x02
#define EP2              0x04
#define EP               0x07   /*ep block mask */
#define EPDIS            0x08
#define EOPEN            0x80

#define BLANK            0x04   /*bit masks      */
#define ACCERR           0x10
#define PVIOL            0x20
#define CCIF             0x40
#define CBEIF            0x80

#define MASS             0x01   /*bit masks      */
#define ERVER            0x04
#define PROG             0x20
#define ERASE            0x40

/************************* Macros ********************************************/

   /* Macro that generates the EEPROM clock precaler as per data book.
      If the crystal frequency is above 12.8MHz then the EDIV bit must
      be set, this divides the OSCCLK frequency by 8 before the prescaler. */
#if (OSCCLK_FREQ_KHZ > 12800)
   /* This macro calculates the prescaler, but also implements the EDIV8 bit */
#if ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200)) % (1600 * BUSCLK_FREQ_KHZ) == 0)
#define EECLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (1600 * BUSCLK_FREQ_KHZ))
+ EDIV8 - 1)
#else
#define EECLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (1600 * BUSCLK_FREQ_KHZ))
+ EDIV8)
#endif /* OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) % (1600 * BUSCLK_FREQ_KHZ) == 0 */
   /* Make sure EECLK is within specified range. */
```

```
#define EECLK_FREQ_KHZ (OSCCLK_FREQ_KHZ / (8 * (1 + EECLK_PRESCALER - EDIV8)))
#if ((EECLK_FREQ_KHZ < 150) || (EECLK_FREQ_KHZ > 200) || (EECLK_PRESCALER > 0x7F))
#error EEPROM prescaler or clock out of range.
#endif /* Incorrect EECLK frequency. */

#else
   /* This macro calculates the prescaler. */
#if ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200)) % (200 * BUSCLK_FREQ_KHZ) == 0)
#define EECLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (200 * BUSCLK_FREQ_KHZ))
- 1)
#else
#define EECLK_PRESCALER (OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (200 * BUSCLK_FREQ_KHZ))
#endif /* OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) % (200 * BUSCLK_FREQ_KHZ) == 0 */
   /* Make sure ECLK is within specified range. */
#define EECLK_FREQ_KHZ (OSCCLK_FREQ_KHZ / (1 + EECLK_PRESCALER))
#if ((EECLK_FREQ_KHZ < 150) || (EECLK_FREQ_KHZ > 200) || (EECLK_PRESCALER > 0x3F))
#error EEPROM prescaler or clock out of range.
#endif /* Incorrect EECLK frequency. */
#endif /* OSCCLK_FREQ_KHZ > 12800 */

/************************ Prototypes ***************************************/
void  ConfigECLKDIV(void);
UINT8 ProgEeprom(UINT16*, UINT16*, UINT16);

#endif          /* End of Header file !defined    */
```

```
/*****************************************************************************
                                                 Copyright (c)
File Name                 :      $RCSfile: s12_fectl.h,v $

Engineer                  :      $Author: r27624 $

Location                  :      EKB

Date Created              :      09/01/01

Current Revision          :      $Revision: 1.0 $

******************************************************************************
Freescale reserves the right to make changes without further notice to any
Product herein to improve reliability, function or design. Freescale does not
assume any liability arising out of the application or use of any product,
circuit, or software described herein; neither does it convey any license
under its patent rights nor the rights of others. Freescale products are not
designed, intended, or authorized for use as components in systems intended for
surgical implant into the body, or other applications intended to support life,
or for any other application in which the failure of the  Freescale product
could create a situation where personal injury or death may occur. Should
Buyer purchase or use Freescale products for any such unintended or
unauthorized application, Buyer shall idemnify and hold Freescale and its
officers, employees, subsidiaries, affiliates, and distributors harmless
against all claims costs, damages, and expenses, and reasonable attorney fees
arising out of, directly or indirectly, any claim of personal injury or death
associated with such unintended or unauthorized use, even if such claim alleges
that Freescale was negligent regarding the design or manufacture of the part.
Freescale and the Freescale logo* are registered trademarks of Freescale Semiconductor, Inc.
*****************************************************************************/

#ifndef FECTL_H
#define FECTL_H

/************************ System Include Files *****************************/

/************************ Project Include Files ****************************/
#include "stdtypes.h"

/************************ typedefs *****************************************/
typedef union uFCLKDIV
  {
  UINT8         byte;
  struct
    {
    UINT8 fdiv          :6;                     /*clk divider */
    UINT8 fdiv8         :1;                     /*clk /8 prescaler enable */
    UINT8 fdivld        :1;                     /*clock divider loaded flag */
    }bit;
  }tFCLKDIV;

typedef union uFSEC
  {
```

```
  UINT8         byte;
  struct
    {
    UINT8 sec0          :2;                     /*memory security bit */
    UINT8 nv            :5;                     /*user non volatile flag bits */
    UINT8 keyen         :1;                     /*security key access enable */
    }bit;
  }tFSEC;

typedef union uFCNFG
  {
  UINT8         byte;
  struct
    {
    UINT8 bksel         :2;                     /*register bank select */
    UINT8               :3;                     /*not used */
    UINT8 keyacc        :1;                     /*security key writing enable */
    UINT8 ccie          :1;                     /*command complete interrupt enable */
    UINT8 cbeie         :1;                     /*command buffer empty interrupt enable*/
    }bit;
  }tFCNFG;

typedef union uFPROT
  {
  UINT8         byte;
  struct
    {
    UINT8 fpls          :2;                     /*flash protection lower address size */
    UINT8 fpldis        :1;                     /*flash protection lower address range
disable */
    UINT8 fphs          :2;                     /*flash protection higher address size */
    UINT8 fphdis        :1;                     /*flash protection higher address range
disable */
    UINT8               :1;                     /*contains value of equivalent bit in
protection byte */
    UINT8 fopen         :1;                     /*open block for program/erase control */
    }bit;
  }tFPROT;

typedef union uFSTAT
  {
  UINT8         byte;
  struct
    {
    UINT8               :2;                     /*not used */
    UINT8 blank         :1;                     /*blank verify flag */
    UINT8               :1;                     /*not used */
    UINT8 accerr        :1;                     /*access error flag */
    UINT8 pviol         :1;                     /*protection violation flag */
    UINT8 ccif          :1;                     /*command complete interrupt flag */
    UINT8 cbeif         :1;                     /*command buffer empty interrupt flag */
    }bit;
  }tFSTAT;

typedef union uFCMD
```

```
  {
  UINT8         byte;
  struct
    {
    UINT8 mass          :1;                     /*mass erase enable*/
    UINT8               :1;                     /*not used */
    UINT8 erver         :1;                     /*erase verify enable */
    UINT8               :2;                     /*not used */
    UINT8 prog          :1;                     /*word programming */
    UINT8 erase         :1;                     /*erase control */
    UINT8               :1;                     /*not used */
    }bit;
  }tFCMD;

typedef struct                                  /*flash datastructure */
  {
  volatile tFCLKDIV             fclkdiv;        /*flash clock divider register */
  volatile tFSEC                fsec;           /*flash security register */
             UINT8              rsvfee1;        /*reserved */
           tFCNFG               fcnfg;          /*flash configuration register */
  volatile tFPROT               fprot;          /*flash protection register */
  volatile tFSTAT               fstat;          /*flash status register */
             tFCMD              fcmd;           /*flash command buffer & status register */
  volatile UINT8                rsvfee2[9];     /*reserved */
  }tFLASH;

/************************* Extern Variables **********************************/

/************************* #Defines *****************************************/
#define FDIV8           0x40    /*bit masks      */
#define FDIVLD          0x80

#define SEC00           0x01    /*bit masks      */
#define SEC01           0x02
#define NV2             0x04
#define NV3             0x08
#define NV4             0x10
#define NV5             0x20
#define NV6             0x40
#define KEYEN           0x80

#define BKSEL0          0x01    /*bit masks      */
#define BKSEL1          0x02
#define BKSEL           0x03    /*bank select mask */
#define KEYACC          0x20
#define CCIE            0x40
#define CCBIE           0x80

#define FPLS0           0x01    /*bit masks      */
#define FPLS1           0x02
#define FPLS            0x03    /*fpls block size mask */
#define FPLDIS          0x04
#define FPHS0           0x08
#define FPHS1           0x10
#define FPHS            0x18    /*fphs block size mask */
```

```
#define FPHDIS              0x20
#define FOPEN               0x80

#define BLANK               0x04   /*bit masks      */
#define ACCERR              0x10
#define PVIOL               0x20
#define CCIF                0x40
#define CBEIF               0x80

#define MASS                0x01   /*bit masks      */
#define ERVER               0x04
#define PROG                0x20
#define ERASE               0x40

/************************* Macros ********************************************/

   /* Macro that generates the FLASH clock precaler as per data book.
      If the crystal frequency is above 12.8MHz then the FDIV bit must
      be set, this divides the OSCCLK frequency by 8 before the prescaler. */
#if (OSCCLK_FREQ_KHZ > 12800)
   /* This macro calculates the prescaler, but also implements the FDIV8 bit */
#if ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200)) % (1600 * BUSCLK_FREQ_KHZ) == 0)
#define FCLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (1600 * BUSCLK_FREQ_KHZ))
+ FDIV8 - 1)
#else
#define FCLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (1600 * BUSCLK_FREQ_KHZ))
+ FDIV8)
#endif /* OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) % (1600 * BUSCLK_FREQ_KHZ) == 0 */
   /* Make sure FCLK is within specified range. */
#define FCLK_FREQ_KHZ (OSCCLK_FREQ_KHZ / (8 * (1 + FCLK_PRESCALER - FDIV8)))
#if ((FCLK_FREQ_KHZ < 150) || (FCLK_FREQ_KHZ > 200) || (FCLK_PRESCALER > 0x7F))
#error FLASH prescaler or clock out of range.
#endif /* Incorrect FCLK frequency. */

#else
   /* This macro calculates the prescaler. */
#if ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200)) % (200 * BUSCLK_FREQ_KHZ) == 0)
#define FCLK_PRESCALER ((OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (200 * BUSCLK_FREQ_KHZ))
- 1)
#else
#define FCLK_PRESCALER (OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) / (200 * BUSCLK_FREQ_KHZ))
#endif /* OSCCLK_FREQ_KHZ * (BUSCLK_FREQ_KHZ + 200) % (200 * BUSCLK_FREQ_KHZ) == 0 */
   /* Make sure FCLK is within specified range. */
#define FCLK_FREQ_KHZ (OSCCLK_FREQ_KHZ / (1 + FCLK_PRESCALER))
#if ((FCLK_FREQ_KHZ < 150) || (FCLK_FREQ_KHZ > 200) || (FCLK_PRESCALER > 0x3F))
#error FLASH prescaler or clock out of range.
#endif /* Incorrect FCLK frequency. */
#endif /* OSCCLK_FREQ_KHZ > 12800 */

/************************* Prototypes ****************************************/
void  ConfigFCLKDIV(void);
UINT8 ProgFlash(UINT16*, UINT16*, UINT16);

#endif          /* End of Header file !defined    */
```

```
/***************************************************************************
                                                   Copyright (c)
File Name                :      $RCSfile: ProgEeprom.c,v $

Engineer                 :      $Author: r27624 $

Location                 :      EKB

Date Created             :      05/06/2001

Current Revision         :      $Revision: 1.2 $

****************************************************************************


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

***************** System Include Files *****************************/

***************** Project Include Files ****************************/
"stdtypes.h"
"mcucfg.h"
"s12_eectl.h"

***************** typedefs *****************************************/

***************** #Defines *****************************************/
      ALIGNED_WORD_MASK
LIGNED_WORD_MASK       0x0001


ASS
ASS                    1


AIL
AIL                    0
```

```
/************************* Global Variables **********************************/
static tEEPROM           eeprom                  @(REG_BASE + 0x110);

/************************* External Variables ********************************/

/******************************************************************************
Function Name              :      ConfigECLKDIV
Engineer                   :      r27624
Date                       :      17/9/2001

Arguments                  :      none

Return                     :      none

Notes                      :      This function configures the EEPROM clock prescaler
                                  in preparation for programming.
******************************************************************************/
void
ConfigECLKDIV(void)
{
        if(eeprom.eclkdiv.bit.edivld == 0)
        {                                         /* configure EEPROM clock prescaler */
                eeprom.eclkdiv.byte  = (UINT8)EECLK_PRESCALER;
        }
        return;
}

/******************************************************************************
Function Name:            ProgEeprom
Engineer         :        r27624
Date             :        28/06/2001

Arguments        :        progAdr                 Pointer to the start of the destination
                                                  EEPROM location to be programmed

                          bufferPtr               Pointer to the start of the source data

                          size                    Number of WORDS to be programmed

Return           :        status                  FAIL:
                                                  if progAdr does not point to an aligned word,
                                                  or the ACCERR bit is set during programming
                                                  sequence, or the PVIOL bit is set during
                                                  programming sequence.

                                                  PASS:
                                                  if not FAIL.

Notes            :        This function does not check if the EEPROM is erased.
                          This function does not verify that the data has been
                          successfully programmed.
******************************************************************************/
UINT8
ProgEeprom(UINT16* progAdr, UINT16* bufferPtr, UINT16 size)
{
        if(((UINT16)progAdr & ALIGNED_WORD_MASK) != 0)/* Check for aligned word */
        {
                return(FAIL);
        }
                                /* Clear error flags */
        eeprom.estat.byte = (ACCERR | PVIOL);
                                /* Word to program? */
        while(size != 0)
        {                       /* Is command buffer empty? */
                if(eeprom.estat.bit.cbeif == 1)
                {
                                /* Latch data and address */
                         *progAdr++ = *bufferPtr++;
                                /* Configure prog command */
                         eeprom.ecmd.byte = PROG;
                                /* Launch the command */
                         eeprom.estat.byte = CBEIF;
                                /* Was the access error flag set? */
                                /* Was the protection violation error flags set */
                         if((eeprom.estat.bit.accerr == 1) ||
                            (eeprom.estat.bit.pviol == 1))
                         {
                                return(FAIL);
                         }
                                /* next word */
                         size--;
                }

        }
                                /* Wait for last command to finish */
        while(eeprom.estat.bit.ccif != 1)
        {
        }
                                /* finished, no errors */
        return(PASS);
}
```

```
/******************************************************************************
                                                Copyright (c)
File Name               :      $RCSfile: ProgFlash.c,v $

Engineer                :      $Author: r27624 $

Location                :      EKB

Date Created            :      05/06/2001

Current Revision        :      $Revision: 1.2 $

*******************************************************************************
Freescale reserves the right to make changes without further notice to any
Product herein to improve reliability, function or design. Freescale does not
assume any liability arising out of the application or use of any product,
circuit, or software described herein; neither does it convey any license
under its patent rights nor the rights of others. Freescale products are not
designed, intended, or authorized for use as components in systems intended for
surgical implant into the body, or other applications intended to support life,
or for any other application in which the failure of the  Freescale product
could create a situation where personal injury or death may occur. Should
Buyer purchase or use Freescale products for any such unintended or
unauthorized application, Buyer shall idemnify and hold Freescale and its
officers, employees, subsidiaries, affiliates, and distributors harmless
against all claims costs, damages, and expenses, and reasonable attorney fees
arising out of, directly or indirectly, any claim of personal injury or death
associated with such unintended or unauthorized use, even if such claim alleges
that Freescale was negligent regarding the design or manufacture of the part.
Freescale and the Freescale logo* are registered trademarks of Freescale Inc.
******************************************************************************/

/************************* System Include Files ******************************/

/************************* Project Include Files *****************************/
#include "stdtypes.h"
#include "mcucfg.h"
#include "s12_fectl.h"

/************************* typedefs ******************************************/

/************************* #Defines ******************************************/
#ifndef ALIGNED_WORD_MASK
#define ALIGNED_WORD_MASK                               0x0001
#endif

#ifndef PASS
#define PASS                                            1
#endif

#ifndef FAIL
#define FAIL                                            0
#endif
```

```
/************************ Global Variables *********************************/
static tFLASH flash @(REG_BASE + 0x100);

/************************ External Variables ********************************/

/******************************************************************************
Function Name           :     ConfigFCLKDIV
Engineer                :     r27624
Date                    :     17/9/2001

Arguments               :     none

Return                  :     none

Notes                   :     This function configures the flash clock prescaler.
*****************************************************************************/
void
ConfigFCLKDIV(void)
{
        if(flash.fclkdiv.bit.fdivld == 0)
        {                                       /* configure flash clock prescaler */
                flash.fclkdiv.byte  = (UINT8)FCLK_PRESCALER;
        }

        return;
}

/******************************************************************************
Function Name           :     ProgFlash
Engineer                :     r27624
Date                    :     17/9/2001

Arguments               :     progAdr           Pointer to the start of the destination
                                                Flash location to be programmed

                              bufferPtr         Pointer to the start of the source data

                              size        Number of WORDS to be programmed

Return                  :     FAIL              if progAdr does not point to an aligned
word, or the ACCERR bit is set during programming sequence, or the PVIOL bit is set during
programming sequence.

                              PASS              if not FAIL.

Notes                   :     This function does not check if the flash is erased.
                              This function does not verify that the data has been
                              sucessfully programmed.
                              This function will program non-paged flash only
                              This function must NOT be located in pages $3C to $3F
*****************************************************************************/
UINT8
ProgFlash(UINT16* progAdr, UINT16* bufferPtr, UINT16 size)
{
        UINT8 CCRCopy;
```

```
if(((UINT16)progAdr & ALIGNED_WORD_MASK) != 0)/* Check for aligned word */
{
        return(FAIL);
}

asm
{
        TFR       CCR,A
        STAA      CCRCopy                    ;store CCR
        ORCC      #$10                       ;mask interrupts
}
                        /* Clear error flags for ALL arrays */
flash.fcnfg.byte = 3;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcnfg.byte = 2;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcnfg.byte = 1;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcnfg.byte = 0;   /* this array to be programmed */
flash.fstat.byte = (PVIOL | ACCERR);

while(size != 0)
{
                        /* Is the command buffer empty? */
        if(flash.fstat.bit.cbeif == 1)
        {
                        /* Write word to FLASH buffer. */
                 *progAdr++ = *bufferPtr++;
                        /* Initiate program command */
                 flash.fcmd.byte = PROG;
                        /* Clear command buffer empty flag by writing a 1 to it
                        This launches the command. */
                 flash.fstat.byte = CBEIF;
                        /* Was the access error flag set */
                        /* Was the protection violation error flags set */
                 if((flash.fstat.bit.accerr == 1) ||
                    (flash.fstat.bit.pviol == 1))
                 {
                        asm
                        {
                             LDAA         CCRCopy
                             TFR          A,CCR    ;restore CCR
                        }
                        /* Return status. */
                        return(FAIL);
                 }
                        /* One less word to program */
/* One less word to write */
                 size--;
        }
}
                        /* wait for last command to complete */
while(flash.fstat.bit.ccif != 1)
{
        }

        asm
        {
                LDAA     CCRCopy
                TFR      A,CCR                  ;restore CCR
        }
                                /* Return status. */
        return(PASS);
}

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

[redacted]

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor