



















```
/*
Standard Definitions
*/

#ifndef FALSE
#define FALSE 0
#endif

#ifndef TRUE
#define TRUE 1
#endif

/*
Standard Enumerations
*/

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

---



```
/* register base address */  
#define REG_BASE          0x0000  
  
/***** Macros *****/  
  
/***** Prototypes *****/  
  
#endif          /* End of Header file !defined */
```

---



```

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              rsveel[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 */
```

---



```

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 bkssel          :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          rsvfeel;     /*reserved */
        tFCNFG          fcng;        /*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 */

```



```

/***** 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) Motorola 2001
File Name      :      $RCSfile: ProgFlash.c,v $
Engineer       :      $Author: r27624 $
Location       :      EKB
Date Created   :      05/06/2001
Current Revision :      $Revision: 1.2 $
*/
```

```
*****
Motorola reserves the right to make changes without further notice to any
Product herein to improve reliability, function or design. Motorola 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. Motorola 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 Motorola product
could create a situation where personal injury or death may occur. Should
Buyer purchase or use Motorola products for any such unintended or
unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part.
Motorola and the Motorola logo* are registered trademarks of Motorola Ltd.
*****/
```

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

```
/* ***** Project Include Files ***** */
```

```
#include "stdtypes.h"
#include "mcucfg.h"
#include "sl2_lect1.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
                        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
                        PASS             if not FAIL.
                        word, or the ACCERR bit is set during programming sequence, or the PVIOL bit is set during
                        programming sequence.

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.fcncfg.byte = 3;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcncfg.byte = 2;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcncfg.byte = 1;
flash.fstat.byte = (PVIOL | ACCERR);
flash.fcncfg.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.fcncfg.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);  
}  
  
/*****/
```

---

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **E-mail:**

[support@freescale.com](mailto: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](mailto: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](mailto: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](mailto: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](mailto: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](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor 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 or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor 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 Semiconductor was negligent regarding the design or manufacture of the part.

