In-System Programming - Flash Library for T89C51RD2

1. Introduction

1.1. Overview

The T89C51RD2 provides a BootFlash which contains routines to allow an application to perform Read/Write operations on the internal Flash memory. The routine may be called at one entry point but with varying values in register R0, R1, ACC and DPTR0&1 to perform the following operations:

- Read and write bytes in the Flash memory
- Read and write the Security Bits
- Read the Software Boot Vector and Boot Status Byte
- Erase and write the Software Boot Vector and Boot Status Byte
- Read the Manufacturer ID, Device ID and bootloader version
- Read and write bytes in the EEPROM

The Flash_API library provides a mean to perform all operations by making function calls in C language. The Flash_API library provides a standard way to call these functions in C language. It has been done for Keil C-compilers parameter conventions but can be adapted for others. This library targets T89C51RD2 but it can be updated to support any future Flash ISP devices without changing the user's source programs.

The library also provides Macros and pre-defined values to ease certain operations.

1.2. Acronyms

ISP: In-System Programming

API: Application Program Interface

BSB: Boot Status Byte

SBV: Software Boot Vector

SSB: Software Security Bit

HSB: Hardware Security Bit

2. Library Usage

2.1. Adding to a Project

The library consist of two source files:

- flash_api.c
- flah_api.a51

and two C header file

- flash_api.h
- config.h

The library is dedicated to Keil 8051 compilers. (See file STATUS in the distribution).

To use the library simply add the source files "flash_api.c" and "flash_lib.a51" to your project and include "flash_api.h" in all C source files that use the library. The content of our config.h file, can easily be added to one of thue user include files. You also need to include the standard "t89c51rd2.h" file. If you use your own SFR description file, check that you correctly define the DPTR register as a 16bits SFR.

The library can be easily configured with the "flash_api.h" file. Thus, only the needed functions will be compiled.

Take care to define the correct value to EETIM_VALUE for EEPROM access and define the correct memory model used with your compiler (validate the "#define LARGE_MEMORY_MODEL" when using this compilation mode) in "flash_api.h" file.

2.2. Execution Environment

Each function in the library modifies the configuration of the microcontroller in the following ways during the function call (the configuration is restored at the end of the function call):

- All interrupts are disabled during write access to code flash or EEPROM data.
- The Hardware Watchdog Timer timer is not disable. Thus, the user has to take care of it before launching a Flash operation(flash writing process needs at least 10 ms). It is higly recommended to disable the Hardware Watchdog Timer before calling a function for write in Flash.

3. Description

3.1. Types Description

Uchar : unsigned char

Uint16 : unsigned short

3.2. Functions Resume

Function	Parameters	Return
`__api_rd_code_byte (macro)`	Uint16 address	Uchar value
`__api_wr_code_byte`	Uint16 address, Uchar value	Uchar state
`__api_wr_code_page`	Uint16 address, Uint16 pt_xram, Uchar nb_data	Uchar state
`__api_rd_BSB`	void	Uchar value
`__api_wr_BSB`	Uchar value	Uchar state
`__api_rd_SBV`	void	Uchar value
`__api_wr_SBV`	Uchar value	Uchar state
`__api_erase_SBV`	void	Uchar state
`__api_rd_SSB`	void	Uchar value
`__api_wr_SSB`	ssb_t value	Uchar state
`__api_rd_HSB`	void	Uchar value
`__api_rd_manufacterer`	void	Uchar value
`__api_rd_device_id1`	void	Uchar value
`__api_rd_device_id2`	void	Uchar value
`__api_rd_device_id3`	void	Uchar value
`__api_rd_bootloader_version`	void	Uchar value
`__api_eeprom_busy (macro)`	void	Uchar state
`__api_rd_eeprom_byte`	Uint16 address	Uchar value
`__api_wr_eeprom_byte`	Uint16 address, Uchar value	Uchar state
`__api_wr_eeprom_page`	Uint16 address, Uint16 pt_xram, Uchar value	Uchar state

3.3. Functions Description

3.3.1. __api_rd_code_byte

This function is used to read a byte value in Flash memory on given address. To use this function __API_RD_CODE_BYTE constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_code_byte (Uint16 address)

- Input:

  Uint16 address: address of the byte to read

- Output:

  Uchar return: value read at *address* in Flash memory

- Example:

```
Uchar read_value;
read_value = __api_rd_code_byte (0x100);
```

3.3.2. __api_wr_code_byte

This function is used to write a byte in Flash memory on given address. To use this function __API_WR_CODE_BYTE constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_code_byte (Uint16 address, Uchar value)

- Input:

  Uint16 address: address where byte must be wrote

  Uchar value: byte to write in Flash memory

- Output:

  Uchar return:

  return = 0x00 -> program success
  return != 0x00 -> program fail

- Example:

```
if(__api_wr_code_byte(0x500, 0x55)==0x00)
/* program succeded */
else
/* program failed */
```

3.3.3. __api_wr_code_page

This function is used to write until 128 bytes from XRAM to Flash memory on given start address. To use this function __API_WR_CODE_PAGE constant must be defined in flash_api.h file, otherwise the function is not compiled.

The only restriction is that all data must be in the same page. The page size is 128 bytes.

This function used Dual Data Pointer DPTR0&1. At the end of this function the DPTR = DPTR0.

- Prototype:

  Uchar __api_wr_code_page (Uint16 add_flash, Uint16 pt_xram, Uchar nb_data)

- Input:

  Uint16 add_flash: address in Flash where bytes must be written

  Uchar pt_xram*: pointer on the first XRAM data to write

  Uchar nb_data: number of byte to write in Flash memory

- Output:

  Uchar return:

  return = 0x00 -> program success
  return != 0x00 -> program fail

- Example:

```
xdata Uchar buffer[128] = {0x55,.., 0x55};

if(__api_wr_code_page(0x0000, &buffer[], 128)==0x00)
/* program succeded */
else
/* program failed */
```

3.3.4. __api_rd_BSB

This function is used to read BSB. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_BSB (void)

- Input:

  void

- Output:

  Uchar return: BSB read

- Example:

```
Uchar BSB_value;
BSB_value = __api_rd_BSB (void);
```

3.3.5. __api_wr_BSB

This function is used to write BSB. To use this function __API_FCT_SET_2 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_BSB (Uchar BSB)

- Input:

  Uchar BSB: value of BSB to write

- Output:

  Uchar state

- Example:

```
__api_wr_BSB (0x55);
```

3.3.6. __api_rd_SBV

This function is used to read SBV. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_SBV (void)

- Input:

  void

- Output:

  Uchar return: SBV read

- Example:

```
Uchar  SBV_value;
SBV_value  =  __api_rd_SBV(void);
```

3.3.7. __api_wr_SBV

This function is used to write SBV. To use this function __API_FCT_SET_2 constant must be defined flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_SBV (Uchar SBV)

- Input:

  Uchar SBV: value of SBV to write

- Output:

  Uchar state

- Example:

```
 __api_wr_SBV  (0x80);
```

3.3.8. __api_erase_SBV

This function is used to erase SBV. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_erase_SBV (void)

- Input:

  void

- Output:

  Uchar state

- Example:

```
 __api_erase_SBV  ();
```

3.3.9. __api_rd_SSB

This function is used to read SSB. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_SSB (void)

- Input:

  void

- Output:

  Uchar return: SSB read

- Example:

```
Uchar SSB_value;
SSB_value = __api_rd_SSB (void);
```

3.3.10. __api_wr_SSB

This function is used to write SSB. To use this function __API_FCT_SET_1 constant must be defined flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_SSB (ssb_t SSB)

- Input:

  ssb_t SSB: value of SSB to write (LEVEL1, LEVEL2, LEVEL2_1, LEVEL1_0)

- Output:

  Uchar state

- Example:

```
__api_prg_SSB (LEVEL2);
```

3.3.11. __api_rd_HSB

This function is sed to read HSB byte. To use this function __API_FCT_SET_1 constant must be defined flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_SSB (ssb_t SSB)

- Input:

  ssb_t SSB: value of SSB to write (LEVEL1, LEVEL2, LEVEL2_1, LEVEL1_0)

- Output:

  Uchar state

- Example:

```
__api_prg_SSB (LEVEL2);
```

3.3.12. __api_rd_manufacturer

This function is used to read manufacturer ID. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_manufacturer (void)

- Input:

  void

- Output:

  Uchar return: manufacturer id read

- Example:

```
Uchar  manufacturer_value;
manufacturer_value  =  __api_rd_manufacturer  (void);
```

3.3.13. __api_rd_device_id1

This function is used to read device id1. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_device_id1 (void)

- Input:

  void

- Output:

  Uchar return: device id1 read

- Example:

```
Uchar  device_id1_value;
device_id1_value  =  __api_rd_device_id1  (void);
```

3.3.14. __api_rd_device_id2

This function is used to read device id2. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_device_id2 (void)

- Input:

  void

- Output:

  Uchar return: device id2 read

- Example:

```
Uchar  device_id2_value;
device_id2_value  =  __api_rd_device_id2  (void);
```

3.3.15. __api_rd_device_id3

This function is used to read device id3. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_device_id3 (void)

- Input:

  void

- Output:

  Uchar return: device id3 read

- Example:

```
Uchar  device_id3_value;
device_id3_value  =  __api_rd_device_id3  (void);
```

3.3.16. __api_rd_bootloader_version

This function is used to read bootloader version. To use this function __API_FCT_SET_1 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_bootloader_version (void)

- Input:

  void

- Output:

  Uchar return: version of bootloader

- Example:

```
Uchar  bootloader_version;
bootloader_version  =  __api_rd_bootloader_version  (void);
```

3.3.17. __api_eeprom_busy

This function is used to read the state of eeprom. To use this function __API_EEPROM_BUSY constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_eeprom_busy (void)

- Input:

  void

- Output:

  eeprom_state_t return = EEPROM_BUSY or EEPROM_NOT_BUSY

- Example:

```
if(__api_eeprom_busy!=EEPROM_BUSY)
/* eeprom access allowed */
else
/* eeprom access forbidden*/
```

3.3.18. __api_rd_eeprom_byte

This function is used to read a byte value in Eeprom memory on given address. To use this function __API_RD_EEPROM_BYTE constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_rd_eeprom_byte (Uint16 address)

- Input:

  Uint16 address: address of the byte to read

- Output:

  Uchar return: value read at *address* in Eeprom memory

- Example:

```
Uchar read_value;

if(__api_eeprom_busy!=EEPROM_BUSY)
read_value = __api_rd_eeprom_byte (0x100);
```

3.3.19. __api_wr_eeprom_byte

This function is used to write a byte in Eeprom memory on given address. To use this function __API_WR_EEPROM_BYTE constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_wr_eeprom_byte (Uint16 address, Uchar value)

- Input:

  Uint16 address: address where byte must be wrote

  Uchar value: byte to write in Eeprom memory

- Output

  Uchar state

- Example:

```
if(__api_eeprom_busy!=EEPROM_BUSY)
 __api_wr_eeprom_byte (0x100,  0x55);
```

4. Library Run Time Requirements

4.1. Code Size Per Function

Function	Size (bytes) small / large	Ram / Xram small / large	Stack
__api_wr_code_byte	18 / 26	1 / 0	3
__api_wr_code_page	35 / 57	3 / 2	3
__api_rd_BSB	13 / 22		3
__api_wr_BSB	20 / 25	1 / 0	3
__api_rd_SBV	13 / 22		3
__api_wr_SBV	20 / 25	1 / 0	3
__api_erase_SBV	13 / 22		3
__api_rd_SSB	13 / 22		3
__api_wr_SSB	13 / 22		3
__api_rd_HSB	13 / 22		3
__api_rd_manufacturer	13 / 22		3
__api_rd_device_id1	13 / 22		3
__api_rd_device_id2	13 / 22		3
__api_rd_device_id3	13 / 22		3
__api_rd_bootloader_version	13 / 22		3
__api_rd_eeprom_byte	17 / 17		3
__api_wr_eeprom_byte	25 / 25		3
__api_wr_eeprom_page	61 / 87	2 / 2	3

Note : For all Write functions (Flash and EEPROM Data), one ram bit is used to save and restore the EA bit state.

4.2. Total Code Size

All functions using generic function by #define __API_FCT_SET_1 reserved only one time 13 or 22 bytes and all functions using generic function by #define __API_FCT_SET_2 reserved only one time 20 or 25 bytes

5. Source Code

```
/*****************************************************************************
*
*(c) ATMEL-Wireless and Microcontrollers 2001
*
*
*****************************************************************************/

/*H*************************************************************************
*  NAME: config.h
*----------------------------------------------------------------------------
*  AUTHORS :Jean-sebastien Berthy && Raphaël L'etendu
*****************************************************************************/

#ifndef _CONFIG_H_
#define _CONFIG_H_


/************   Type Definition ************/


typedef unsigned char                           Uchar;
typedef unsigned int                            Uint16;
typedef int                                     Int16;
typedef float                                   Float16;
typedef unsigned long int                       Uint32;
typedef long int                                Int32;
typedef unsigned char                           Bool;


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

#include "89c51rd2.h"
#include "flash_api.h"


#endif /* _CONFIG_H_ */
```

```
/*****************************************************************************
*
*                  (c) ATMEL-Wireless and Microcontrollers 2001
*
*
*****************************************************************************/

/*H*************************************************************************
* NAME: flash_api.h
*----------------------------------------------------------------------------
* CREATED_BY:     Jean-sebastien Berthy && Raphaël L'etendu
* CREATION_DATE:  1/03/2000
* AUTHOR:         $Author: jberthy $
* REVISION       $Revision: 1.1.1.1 $
* DATE:          $Date: 2001/03/02 10:11:24 $
*----------------------------------------------------------------------------
*************************************************************************/


#ifndef _FLASH_API_H_
#define _FLASH_API_H_

/*_____ C O N F I G U R A T I O N ____________________________________*/

/* define here the needed functions */
#define   __API_FCT_SET_1
#define   __API_FCT_SET_2

#define   __API_WR_CODE_BYTE
#define   __API_WR_CODE_PAGE

#define   __API_RD_EEPROM_BYTE
#define   __API_WR_EEPROM_BYTE
#define   __API_WR_EEPROM_PAGE


/*****************************************************************************/
/*****************************************************************************/
/***                                                                       ***/
/***          WARNING :TAKE CARE OF THESE TWO DEFINES                      ***/
/***                                                                       ***/
/*****************************************************************************/
/*****************************************************************************/

#define EETIM_VALUE0x37        /* For exact EETIM value : see T8951RD2 Data Sheet */
                  // Example :value of EETIM   mhz*5=0x37 @ 11 mhz = 0x3C @ 12 mhz

/* IF NEEDED ( LARGE MEMORY MODEL USED ) uncomment this line : */

//   #define LARGE_MEMORY_MODEL


/*_____ E N D   C O N F I G U R A T I O N ____________________________*/


#ifdef __API_FCT_SET_1
Uchar __api_fct_set_1 (Uchar _R1, Uint16 _DPTR);

   #define __api_rd_manufacturer()                  __api_fct_set_1(0, 0)
#define __api_rd_device_id1()                       __api_fct_set_1(0, 1)
#define __api_rd_device_id2()                       __api_fct_set_1(0, 2)
#define __api_rd_device_id3()                       __api_fct_set_1(0, 3)
#define __api_erase_SBV()                           __api_fct_set_1(4, 0)

#defineSSB_ALLOW_WR            ((Uchar)0xFE)
#defineSSB_SECURITY_WR((Uchar)0x00)
#defineSSB_SECURITY_RD((Uchar)0x01)

#define __api_wr_SSB(SSB)                           __api_fct_set_1(5, SSB)

#define __api_wr_SSB_RD_WR_SECURITY()               __api_wr_SSB(SSB_SECURITY_RD)
#define __api_wr_SSB_WR_SECURITY()                  __api_wr_SSB(SSB_SECURITY_WR)
#define __api_wr_SSB_NO_SECURITY()                  __api_wr_SSB(SSB_ALLOW_WR)

#define __api_rd_SSB()                              __api_fct_set_1(7, 0)
#define __api_rd_BSB()                              __api_fct_set_1(7, 1)
#define __api_rd_SBV()                              __api_fct_set_1(7, 2)
#define __api_rd_HSB()                              __api_fct_set_1(7, 3)
#define __api_rd_bootloader_version()               __api_fct_set_1(8, 0)
```

```
#endif


#ifdef __API_FCT_SET_2
Uchar __api_fct_set_2 (Uchar _ACC, Uchar _DPL);

#define __api_wr_BSB(BSB)                          __api_fct_set_2(BSB, 0)
#define __api_wr_SBV(SBV)                          __api_fct_set_2(SBV, 1)

#endif

#define __api_rd_code_byte(address)   (*((Uchar code*) (address)))

#ifdef   __API_WR_CODE_BYTE
Uchar__api_wr_code_byte(Uint16 , Uchar);
#endif

#ifdef   __API_WR_CODE_PAGE
Uchar__api_wr_code_page(Uint16 , Uint16, Uchar);
#endif


/*------------------------------ EEPROM ------------------------------*/
#defineEEPROM_NOT_BUSY          ((Uchar) 0)
#defineEEPROM_BUSY              ((Uchar) 1)


#define   __api_eeprom_busy()   (EECON&1)

#ifdef    __API_RD_EEPROM_BYTE
Uchar      __api_rd_eeprom_byte (Uint16);
#endif

#ifdef    __API_WR_EEPROM_BYTE
Uchar __api_wr_eeprom_byte     (Uint16, Uchar);
#endif

#ifdef   __API_WR_EEPROM_PAGE
Uchar__api_wr_eeprom_page      (Uint16 , Uint16, Uchar);
#endif


#endif /* _FLASH_API_H_ */
```

```
/*****************************************************************************
*
*   (c) ATMEL-Wireless and Microcontrollers 2000
*
*
*****************************************************************************/

/*C*************************************************************************
* NAME: flash_api.c
*----------------------------------------------------------------------------
* CREATED_BY:     Jean-sebastien Berthy && Raphaël L'etendu
* CREATION_DATE:   1/03/2001
* AUTHOR:          $Author: jberthy $
* REVISION         $Revision: 1.1.1.1 $
* DATE:            $Date: 2001/03/02 10:11:24 $
*----------------------------------------------------------------------------*/



/*_____ I N C L U D E - F I L E S ____________________________________*/
#include "config.h"


/*_____ G L O B A L S ________________________________________________*/

#ifdef LARGE_MEMORY_MODEL
Uint16 data data_addr_xram;
Uint16 datadata_addr_flash;
Uchar   datadata_nb_data;
#endif

Uchar datadata_value;

/*_____ L O C A L S __________________________________________________*/

/*_____   F U N C T I O N S - D E C L A R A T I O N_____*/

extern void ASM_MOV_R1_A(void);
extern void __API_FLASH_ENTRY_POINT(void);

/*F*************************************************************************
* NAME: __api_wr_data_byte
*----------------------------------------------------------------------------
* AUTHOR: Jean-sebastien Berthy && Raphaël L'etendu
*----------------------------------------------------------------------------
* PARAMS:
*  Uint16 address : address to program
*  Uchar value : data to write
*  Uchar return :
*                   return   = 0x00 -> pass
*                   return  != 0x00 -> fail
*----------------------------------------------------------------------------
* PURPOSE:
*   Program data byte in Flash memory
*
*****************************************************************************/
#ifdef __API_WR_CODE_BYTE
Uchar __api_wr_code_byte (Uint16 address, Uchar value)
{
#ifdef LARGE_MEMORY_MODEL
   data_addr_flash=address;
   data_value=value;
   DPTR = data_addr_flash;
   ACC = 0x02;
   ASM_MOV_R1_A();
   ACC = data_value;
   __API_FLASH_ENTRY_POINT();
   return (ACC);
#else
   DPTR = address;
   ACC = 0x02;
   ASM_MOV_R1_A();
   ACC = value;
   __API_FLASH_ENTRY_POINT();
   return (ACC);
#endif


}
#endif
```

```
/*F****************************************************************************
*  NAME:  __api_wr_code_page
*-----------------------------------------------------------------------------
*  AUTHOR:  Jean-sebastien Berthy && Raphaël L'etendu
*-----------------------------------------------------------------------------
*  PARAMS:
*   Uint16 add_flash  : address of the first byte to program in the Flash
*   Uint16 add_xram   : address in XRAM of the first data to program
*   Uchar  nb_data  : number of bytes to program
*   Uchar  return  :
*                            return = 0x00 -> pass
*                            return != 0x00 -> fail
*-----------------------------------------------------------------------------
*  PURPOSE:
*    Program until 128 Datas in Flash memory.
*  Number of bytes to program is limited such as the Flash write
*  remains in a single 128 bytes page.
*
*****************************************************************************/
#ifdef  __API_WR_CODE_PAGE
Uchar  __api_wr_code_page (Uint16 add_flash,  Uint16 add_xram,  Uchar  nb_data)
{
#ifdef  LARGE_MEMORY_MODEL
data_addr_flash=add_flash;
data_addr_xram=add_xram;
data_nb_data=nb_data;
AUXR1  &=  ~0x01;            /*  Set  DPTR=DPTR0  */
DPTR  =  data_addr_flash;
AUXR1++;             /*  DPTR=DPTR1  */
DPTR  =  data_addr_xram;
ACC  =  0x09;
ASM_MOV_R1_A();
ACC  =  data_nb_data;        /*  Number  of  bytes  to  program  */
   __API_FLASH_ENTRY_POINT();
AUXR1++;             /*  Set  DPTR=DPTR0  */
return  (ACC);
#else
AUXR1  &=  ~0x01;            /*  Set  DPTR=DPTR0  */
DPTR  =  add_flash;
AUXR1++;             /*  DPTR=DPTR1  */
DPTR  =  add_xram;
ACC  =  0x09;
ASM_MOV_R1_A();
   ACC  =  nb_data;
   __API_FLASH_ENTRY_POINT();
AUXR1++;             /*  Set  DPTR=DPTR0  */
return  (ACC);
#endif

}
#endif


/*F****************************************************************************
*  NAME:  __api_fct_set_1
*-----------------------------------------------------------------------------
*  AUTHOR:  Jean-sebastien Berthy && Raphaël L'etendu
*-----------------------------------------------------------------------------
*  PARAMS:
*   Uchar  _R1  :
*   Uint16  _DPTR  :
*   Uchar  return  :
*-----------------------------------------------------------------------------
*****************************************************************************/
#ifdef  __API_FCT_SET_1
Uchar  __api_fct_set_1 (Uchar  _R1,  Uint16  _DPTR)
{
#ifdef  LARGE_MEMORY_MODEL
data_value=_R1;
data_addr_flash=_DPTR;
DPTR  =  data_addr_flash;
ACC  =  data_value;
ASM_MOV_R1_A();
__API_FLASH_ENTRY_POINT();
return  (ACC);
#else
DPTR  =  _DPTR;
ACC  =  _R1;
ASM_MOV_R1_A();
__API_FLASH_ENTRY_POINT();
```

```
return (ACC);
#endif
}
#endif


/*F****************************************************************************
* NAME: __api_fct_set_2
*-------------------------------------------------------------------------------
* AUTHOR: Jean-sebastien Berthy && Raphaël L'etendu
*-------------------------------------------------------------------------------
* PARAMS:
*  Uchar _ACC :
*  Uchar _DPL :
*  Uchar return :
*-------------------------------------------------------------------------------
*******************************************************************************/
#ifdef __API_FCT_SET_2
Uchar __api_fct_set_2 (Uchar _ACC, Uchar _DPL)
{
#ifdef LARGE_MEMORY_MODEL
data_value=_ACC;
data_nb_data=_DPL;
DPH = 0x00;
DPL = data_nb_data;
ACC = 0x06;
ASM_MOV_R1_A();
ACC = data_value;
__API_FLASH_ENTRY_POINT();
#else
DPH = 0x00;
DPL = _DPL;
ACC = 0x06;
ASM_MOV_R1_A();
ACC = _ACC;
__API_FLASH_ENTRY_POINT();
#endif
return 1;
}
#endif


/*F****************************************************************************
* NAME: api_rd_eeprom_byte
*-------------------------------------------------------------------------------
* AUTHOR: Jean-sebastien Berthy && Raphaël L'etendu
*-------------------------------------------------------------------------------
* PARAMS:
*                 Uint16 address :
*                 Uchar return :
*-------------------------------------------------------------------------------
* PURPOSE: Read a byte in Eeprom
*
*******************************************************************************/
#ifdef __API_RD_EEPROM_BYTE
Uchar __api_rd_eeprom_byte(Uint16 address)
{
register Uchar data_value;
bit ea_save;
ea_save = EA;
EA=0;
EECON |= 0x02;
data_value    = *((Uchar xdata*) address);
EECON--;          // &= 0xFD;
EA=ea_save;       // restore interrupt state
return data_value;
}
#endif
```

```
/*F****************************************************************************
*  NAME: api_wr_eeprom_byte
*-----------------------------------------------------------------------------
*  AUTHOR: Jean-sebastien Berthy && Raphaël L'etendu
*-----------------------------------------------------------------------------
*  PARAMS:
*                  Uint16 address :
*                  Uchar value :
*                  Uchar return :
*-----------------------------------------------------------------------------
*  PURPOSE: Program a byte in Eeprom
*
*****************************************************************************/
#ifdef __API_WR_EEPROM_BYTE
Uchar __api_wr_eeprom_byte(Uint16 address, Uchar value)
{
bit ea_save;
ea_save = EA;
EETIM = EETIM_VALUE;
EA=0;
EECON |= 0x02;
*((Uchar xdata*)address) = value;/* addr is a pointer to external data mem */
EECON =  0x50;   // write 5x, clear 2nd LSb
EECON ^= 0x50;   // write Ax
EA=ea_save;      // restore interrupt state
return 1;
}
#endif


/*F****************************************************************************
*  NAME: __api_wr_eeprom_page
*-----------------------------------------------------------------------------
*  AUTHOR: Jean-sebastien Berthy && Raphaël L'etendu
*-----------------------------------------------------------------------------
*  PARAMS:
*       Uint16 add_eeprom : address of the first byte to program in the Eeprom
*                  Uint16 add_xram   : address in XRAM of the first data to program
*       Uchar nb_data : number of bytes to program
*                  Uchar return :
*-----------------------------------------------------------------------------
*  PURPOSE: Program until 64 bytes in Eeprom memory.
*  Number of bytes to program is limited such as the Eeprom write remains in a
*  single 64 bytes page.
*
*****************************************************************************/
#ifdef __API_WR_EEPROM_PAGE
Uchar __api_wr_eeprom_page (Uint16 add_eeprom, Uint16 add_xram, Uchar nb_data)
{
bit ea_save;

#ifdef LARGE_MEMORY_MODEL
//init
data_addr_flash = add_eeprom;
data_addr_xram  = add_xram;
data_nb_data     = nb_data;
ea_save = EA;
EA = 0;

// write column latch
DPTR = data_addr_xram;
AUXR1++;           /* DPTR=DPTR1 */
DPTR = data_addr_flash;
AUXR1++;
for (;data_nb_data;data_nb_data--)
{
   ACC=*((Uchar xdata*)DPTR);
   AUXR1++;
   EECON |= 0x02;
   *((Uchar xdata*)DPTR) = ACC;
   DPL++;
   EECON--;
   AUXR1++;
   DPTR++;
}
// start programmation
EETIM = EETIM_VALUE;
EECON = 0x50; // write 5x, clear 2nd LSb
EECON^= 0x50; // write Ax
// restore
```

```
EA=ea_save;   // restore interrupt state
#else
ea_save = EA;
EA=0;
// write column latch
DPTR = add_xram;
AUXR1++;            /* DPTR=DPTR1 */
DPTR = add_eeprom;
AUXR1++;
for (;nb_data;nb_data--)
{
   ACC=*((Uchar xdata*)DPTR);
   AUXR1++;
   EECON |= 0x02;
   *((Uchar xdata*)DPTR) = ACC;
   DPL++;
   EECON--;
   AUXR1++;
   DPTR++;
}
// start programmation
EETIM = EETIM_VALUE;
EECON = 0x50; // write 5x, clear 2nd LSb
EECON^= 0x50; // write Ax
// restore
EA=ea_save;   // restore interrupt state
#endif
return 1;
}
#endif
```

```
;*******************************************************************************
;*
;*                 (c)  ATMEL-Wireless  and  Microcontrollers  2001
;*
;*
;*******************************************************************************/

 NAME  FLASH_LIB;
;*A51************************************************************************
;  FILE_NAME             :  FLASH_LIB.a51
;-------------------------------------------------------------------------
;  FILE_CREATED_BY     :  Jean-sebastien Berthy && Raphaël L'etendu
;  FILE_CREATION_DATE:  14/01/00
;-------------------------------------------------------------------------
;  FILE_PURPOSE:  low  level  function  for  API
;*******************************************************************************

USING  0

PUBLIC  ASM_MOV_R1_A
PUBLIC  __API_FLASH_ENTRY_POINT


AUXR1  EQU0A2h

START  SEGMENT  CODE
RSEG  START

;*******************************************************************************
;  FUNCTION_NAME:  ASM_MOV_A_R1
;*******************************************************************************
ASM_MOV_R1_A:
   Mov  R1,  A
RET

;*******************************************************************************
;  FUNCTION_NAME:  __API_FLASH_ENTRY_POINT
;*******************************************************************************
__API_FLASH_ENTRY_POINT:
PUSHAR2
PUSHAR4
PUSHAR6
LCALL  0FFF0h
POP AR6
POP AR4
POP AR2
RET

END
```

6. Bibliography

Datasheet T89C51RD2