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.

This application note is dedicated for the "Flash library T89C51RD2 Rev 1.0.2" source code.

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 the 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 the flash_api execution.
- The Hardware Watchdog Timer is not disable.
  If the user application use the watchdog, some precaution must be take:
  Use the watchdog with a minimun period of 30 ms and must be serviced just before called an api.
  exemple:
  WDTRST = 0x1E;
  WDTRST = 0xE1;
  __api_wr_code_byte(0x500, 0x55);

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

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 up to 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	40 / 42	1 / 0	3
__api_wr_code_page	52 / 68	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

Note : 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. Bibliography

Datasheet T89C51RD2