In-System Programming - Flash Library for T89C51CC01 with CAN bootloader

1. Introduction

1.1. Overview

The T89C51CC01 on chip Bootloader which contains routines to perform Read/Write operations from the Flash memory. The routine may be called at one entry point but with varying values in globales variables to perform the following operations:

- Read and write bytes from 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
- Read and write CAN information
- Start bootloader

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 T89C51CC01 with bootloader CAN 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 consists of one source file:

- flash_api.c

and two C header files

- flash_api.h
- config.h

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

To use the library simply add the source file "flash_api.c" to your project and include "flash_api.h" in all C source files that use the library. The content of our config.h can easily be added to one of your include files. You also need to include the standart "t89c51cc01.h" file. The library can be easily configured with the "flash_api.h" file. Thus, only the needed functions can be compiled.

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 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
ssb_t : enum
NO_SECURITY
WR_SECURITY
RD_WR_SECURITY
eeprom_state_t : enum
EEPROM_NOT_BUSY
EEPROM_BUSY

3.2. Functions Resume

Function	Parameters	Return
Flash Access		
`__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, Uchar* pt_xram, Uchar nb_data	Uchar state
Hardware byte Access		
`__api_rd_HSB`	void	Uchar value
`__api_set_X2`	void	Uchar state
`__api_clr_X2`	void	Uchar state
`__api_set_BLJB`	void	Uchar state
`__api_clr_BLJB`	void	Uchar state
Bootloader configuration Access		
`__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_EB`	void	Uchar value
`__api_wr_EB`	Uchar value	Uchar state
CAN Configuration Access		
`__api_rd_CANBTC1`	void	Uchar value
`__api_wr_CANBTC1`	Uchar value	Uchar state
`__api_rd_CANBTC2`	void	Uchar value
`__api_wr_CANBTC2`	Uchar value	Uchar state
`__api_rd_CANBTC3`	void	Uchar value

Function	Parameters	Return
__api_wr_CANBTC3	Uchar value	Uchar state
__api_rd_NNB	void	Uchar value
__api_wr_NNB	Uchar value	Uchar state
__api_rd_CRIS	void	Uchar value
__api_wr_CRIS	Uchar value	Uchar state
Special bytes Access		
__api_rd_manufacturer	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
EEPROM Access		
__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
Bootloader Access		
__api_start_bootloader	void	void
__api_start_isp	void	void

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.

- Prototype:

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

- Input:

  Uint16 add_flash: address in Flash where bytes must be wrote

  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_2 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 (NO_SECURITY , WR_SECURITY, RD_WR_SECURITY)

- Output:

  Uchar state

- Example:

```
 __api_wr_SSB  (WR_SECURITY);
```

3.3.11. __api_rd_EB

This function is used to read EB. 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_EB (void)

- Input:

  void

- Output:

  Uchar return: EB read

- Example:

```
Uchar  EB_value;
EB_value  =  __api_rd_EB  (void);
```

3.3.12. __api_wr_EB

This function is used to write EB. 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_EB (Uchar EB)
- Input:
  Uchar EB: value of EB to write
- Output:
  Uchar state
- Example:
```
__api_wr_EB (0x00);
```

3.3.13. __api_rd_CANBTC1/2/3

This function is used to read CANBTC1/2/3. 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_CANBTC1/2/3 (void)
- Input:
  void
- Output:
  Uchar return: CANBTC1/2/3 read
- Example:
```
Uchar CANBTC_value;
CANBTC_value = __api_rd_CANBTC1 (void);
```

3.3.14. __api_wr_CANBTC1/2/3

This function is used to write CANBTC1/2/3. 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_CANBTC1/2/3 (Uchar CANBTC1/2/3)
- Input:
  Uchar CANBTC1/2/3: value of CANBTC1/2/3 to write
- Output:
  Uchar state
- Example:
```
__api_wr_CANBTC1 (0x55);
```

3.3.15. __api_rd_NNB

This function is used to read NNB. 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_NNB (void)

- Input:

  void

- Output:

  Uchar return: NNB read

- Example:

```
Uchar NNB_value;
NNB_value = __api_rd_NNB (void);
```

3.3.16. __api_wr_NNB

This function is used to write NNB. 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_NNB (Uchar NNB)

- Input:

  Uchar NNB: value of NNB to write

- Output:

  Uchar state

- Example:

```
__api_wr_NNB (0x55);
```

3.3.17. __api_rd_CRIS

This function is used to read CRIS. 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_CRIS (void)

- Input:

  void

- Output:

  Uchar return: CRIS read

- Example:

```
Uchar CRIS_value;
CRIS_value = __api_rd_CRIS (void);
```

3.3.18. __api_wr_CRIS

This function is used to write CRIS. 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_CRIS (Uchar CRIS)

- Input:

  Uchar CRIS: value of CRIS to write

- Output:

  Uchar state

- Example:

```
__api_wr_CRIS (0x55);
```

3.3.19. __api_rd_HSB

This function is used to read HSB. 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_HSB (void)

- Input:

  void

- Output:

  Uchar return: HSB read

- Example:

```
Uchar HSB_value;
HSB_value = __api_rd_HSB (void);
```

3.3.20. __api_set_X2

This function is used to set bit X2 . To use this function __API_SET_X2 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_set_X2 (void)

- Input:

  void

- Output:

  Uchar state

- Example:

```
__api_set_X2 (void);
```

3.3.21. __api_clr_X2

This function is used to clear bit X2 . To use this function __API_CLR_X2 constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:

  Uchar __api_clr_X2 (void)

- Input:

  void

- Output:

  Uchar state

- Example:

```
__api_clr_X2 (void);
```

3.3.22. __api_set_BLJB

This function is used to set bit BLJB . To use this function __API_SET_BLJB constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  Uchar __api_set_BLJB (void)
- Input:
  void
- Output:
  Uchar state
- Example:
```
__api_set_BLJB (void);
```

3.3.23. __api_clr_BLJB

This function is used to clear bit BLJB . To use this function __API_CLR_BLJB constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  Uchar __api_clr_BLJB (void)
- Input:
  void
- Output:
  Uchar state
- Example:
```
__api_clr_BLJB (void);
```

3.3.24. __api_rd_manufacturer

This function is used to read manufacturer ID. To use this function __API_FC_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);
```

3.3.25. __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);
```

3.3.26. __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);
```

3.3.27. __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);
```

3.3.28. __api_rd_bootloader_version

This function is used to read version of bootloader. 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);
```

3.3.29. __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:

  eeprom_state_t __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*/
```

3.3.30. __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);
```

3.3.31. __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);
```

3.3.32. __api_start_isp

This function is used to start the bootloader with a comunication open. That means you can directly start an ISP communication with FLIP. To use this function __API_START_ISP constant must be defined in flash_api.h file, otherwise the function is not compiled.

This function shouldn't be called inside an interrupt routine.

- Prototype:

  void __api_start_isp (void)

- Input:

  void

- Output

  void

- Example:

```
__api_start_bootloader (void);
```

3.3.33. __api_start_booltoader

This function is used to start the bootloader at address 0xF800. To use this function __API_START_ISP constant must be defined in flash_api.h file, otherwise the function is not compiled.

This function shouldn't be called inside an interrupt routine.

- Prototype:

  void __api_start_bootloader (void)

- Input:

  void

- Output

  void

- Example:

```
__api_start_bootloader (void);
```

4. Library Run Time Requirements

4.1. Maximum Total Code Size in Small memory model

In Small memory model the maximum code size is 186bytes

4.2. Maximum Total Code Size inLarge memory model

In Large memory model the maximum code size is 312 bytes

5. Source Code

```
/*****************************************************************************
*
*             (c)  ATMEL-Wireless  and  Microcontrollers  2001
*
*
*****************************************************************************/
/*H**************************************************************************
*  NAME:  config.h
*----------------------------------------------------------------------------
*  PURPOSE:
*****************************************************************************/
#ifndef  _CONFIG_H_
#define  _CONFIG_H_

#include  "../../../lib/compiler.h"
#include  "../../../lib/t89c51cc01.h"
#include  "flash_api.h"


#endif  /*  _CONFIG_H_  */
```

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

/*H**************************************************************************
* NAME: flash_api.h
*----------------------------------------------------------------------------
*****************************************************************************/

#ifndef _FLASH_API_H_
#define _FLASH_API_H

/*_____ I N C L U D E S _____________________________________________________*/

/*_____ M A C R O S _________________________________________________________*/

/*
* These constants are used to compiled or not the corresponding function
*
* #define __API_RD_DEVICE_DATA   => function __api_rd_device_data compiled
* #undef   __API_RD_HSB=> function __api_rd_HSB not compiled
*/

#define   __API_RD_CODE_BYTE
#define   __API_WR_CODE_BYTE
#define   __API_WR_CODE_PAGE
#define   __API_ERASE_BLOCK

#define   __API_RD_BOOT_VERSION
#define   __API_RD_BSB
#define   __API_RD_SBV
#define   __API_RD_EB
#define   __API_RD_DEVICE_DATA
#define   __API_RD_DEVICE_ID1
#define   __API_RD_DEVICE_ID2
#define   __API_RD_DEVICE_ID3
#define   __API_RD_HSB
#define   __API_RD_MANUFACTURER
#define   __API_RD_SSB
#define   __API_RD_BTC_1
#define   __API_RD_BTC_2
#define   __API_RD_BTC_3
#define   __API_RD_NNB
#define   __API_RD_CRIS

#define   __API_WR_BSB
#define   __API_WR_SBV
#define   __API_WR_SSB
#define   __API_WR_EB
#define   __API_WR_BTC_1
#define   __API_WR_BTC_2
#define   __API_WR_BTC_3
#define   __API_WR_NNB
#define   __API_WR_CRIS

#define   __API_ERASE_SBV

#define   __API_SET_X2
#define   __API_CLR_X2
#define   __API_SET_BLJB
#define   __API_CLR_BLJB

#define   __API_EEPROM_BUSY
#define   __API_RD_EEPROM_BYTE
#define   __API_WR_EEPROM_BYTE

#define   __API_START_BOOTLOADER
#define   __API_START_ISP


// Constante value for api_command.
#define _COMMAND_ER_BLOCK          0
#define _COMMAND_WR_CODE_BYTE      13
#define _COMMAND_WR_CODE_PAGE      13
#define _COMMAND_WR_XAF            4
#define _COMMAND_RD_XAF            5
#define _COMMAND_WR_FUSE_BIT       7
#define _COMMAND_RD_HW             8
```

```
#define _COMMAND_RD_SPECIAL   14

/*_____ D E F I N I T I O N ________________________________________________*/

typedef enum {
   NO_SECURITY = 0xFF,
   WR_SECURITY = 0xFE,
   RD_WR_SECURITY = 0xFC
} ssb_t;

typedef enum {
      BLOCK_0 = 0x00,
      BLOCK_1 = 0x20,
      BLOCK_2 = 0x40,
      BLOCK_3 = 0x80,
      BLOCK_4 = 0xC0 } block_t;

typedef enum {
   EEPROM_NOT_BUSY,
   EEPROM_BUSY       }eeprom_t;


/*_____ D E C L A R A T I O N ______________________________________________*/



extern Uchar __api_rd_generic (Uchar command, Uchar dpl);
extern Uchar __api_wr_generic (Uchar command, Uchar value, Uchar dpl);
extern Uchar __api_wr_fuse (Uchar mask, Uchar filter);


/*---- API for Read access   ------------------------------------------------*/
/*****************************************************************************/
#ifdef __API_RD_BSB
   #define __api_rd_BSB()          __api_rd_generic(_COMMAND_RD_XAF, 0x00)
#endif

#ifdef __API_RD_SBV
   #define __api_rd_SBV()          __api_rd_generic(_COMMAND_RD_XAF, 0x01)
#endif

#ifdef __API_RD_SSB
   #define __api_rd_SSB()          __api_rd_generic(_COMMAND_RD_XAF, 0x05)
#endif

#ifdef __API_RD_EB
   #define __api_rd_EB()           __api_rd_generic(_COMMAND_RD_XAF, 0x06)
#endif

#ifdef __API_RD_BTC_1
   #define __api_rd_BTC_1()        __api_rd_generic(_COMMAND_RD_XAF, 0x1C)
#endif

#ifdef __API_RD_BTC_2
   #define __api_rd_BTC_2()        __api_rd_generic(_COMMAND_RD_XAF, 0x1D)
#endif

#ifdef __API_RD_BTC_3
   #define __api_rd_BTC_3()        __api_rd_generic(_COMMAND_RD_XAF, 0x1E)
#endif

#ifdef __API_RD_NNB
   #define __api_rd_NNB()          __api_rd_generic(_COMMAND_RD_XAF, 0x1F)
#endif

#ifdef __API_RD_CRIS
   #define __api_rd_CRIS()         __api_rd_generic(_COMMAND_RD_XAF, 0x20)
#endif

#ifdef __API_RD_HSB
   #define __api_rd_HSB()          __api_rd_generic(_COMMAND_RD_HW, 0x00)
#endif

#ifdef __API_RD_MANUFACTURER
   #define __api_rd_manufacturer() __api_rd_generic(_COMMAND_RD_XAF, 0x30)
#endif

#ifdef __API_RD_DEVICE_ID1
   #define __api_rd_device_id1()   __api_rd_generic(_COMMAND_RD_XAF, 0x31)
#endif
```

```
#ifdef __API_RD_DEVICE_ID2
   #define __api_rd_device_id2() __api_rd_generic(_COMMAND_RD_XAF, 0x60)
#endif

#ifdef __API_RD_DEVICE_ID3
   #define __api_rd_device_id3() __api_rd_generic(_COMMAND_RD_XAF, 0x61)
#endif

#ifdef __API_RD_BOOT_VERSION
   #define __api_rd_bootloader_version() __api_rd_generic(_COMMAND_RD_SPECIAL, 0x00)
#endif

extern Uchar __api_rd_code_byte (Uchar code * pt_address);


/*---- API for Write access  ------------------------------------------------*/
/*****************************************************************************/

extern Uchar __api_wr_code_byte (Uchar xdata* , Uchar);
extern Uchar __api_wr_code_page (Uchar xdata* pt_code, Uchar xdata* pt_xram, Uchar nb_data);

#ifdef __API_WR_BSB
   #define __api_wr_BSB(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0)
#endif

#ifdef __API_WR_SBV
   #define __api_wr_SBV(value) __api_wr_generic(_COMMAND_WR_XAF, value, 1)
#endif

#ifdef __API_WR_SSB
   #define __api_wr_SSB(value) __api_wr_generic(_COMMAND_WR_XAF, value, 5)
#endif

#ifdef __API_WR_EB
   #define __api_wr_EB(value)      __api_wr_generic(_COMMAND_WR_XAF, value, 6)
#endif

#ifdef __API_WR_BTC_1
   #define __api_wr_BTC_1(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0x1C)
#endif

#ifdef __API_WR_BTC_2
   #define __api_wr_BTC_2(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0x1D)
#endif

#ifdef __API_WR_BTC_3
   #define __api_wr_BTC_3(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0x1E)
#endif

#ifdef __API_WR_NNB
   #define __api_wr_NNB(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0x1F)
#endif

#ifdef __API_WR_CRIS
   #define __api_wr_CRIS(value) __api_wr_generic(_COMMAND_WR_XAF, value, 0x20)
#endif

#ifdef __API_ERASE_SBV
   #define __api_erase_SBV()       __api_wr_generic(_COMMAND_WR_XAF, 0xFF, 1)
#endif

#ifdef __API_SET_X2
   #define __api_set_X2()          __api_wr_fuse(0x80, 0x80)
#endif

#ifdef __API_CLR_X2
   #define __api_clr_X2()          __api_wr_fuse(0x80, 0x00)
#endif

#ifdef __API_SET_BLJB
   #define __api_set_BLJB()        __api_wr_fuse(0x40, 0x40)
#endif

#ifdef __API_CLR_BLJB
   #define __api_clr_BLJB()        __api_wr_fuse(0x40, 0x00)
#endif

extern Uchar __api_erase_block (block_t);

/*---- API for EEPROM access  -----------------------------------------------*/
/*****************************************************************************/
```

```
#ifdef __API_EEPROM_BUSY
   extern eeprom_t __api_eeprom_busy (void);
#endif
#ifdef __API_RD_EEPROM_BYTE
   extern Uchar __api_rd_eeprom_byte(Uchar xdata *);
#endif

#ifdef __API_WR_EEPROM_BYTE
   extern Uchar __api_wr_eeprom_byte(Uchar xdata *, Uchar);
#endif

/*---- API to start bootloader execution  ---------------------------------*/
/*************************************************************************/

#ifdef __API_START_BOOTLOADER
   extern void __api_start_bootloader (void);
#endif

#ifdef __API_START_ISP
   extern void __api_start_isp (void);
#endif

#endif
```

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

/*H*************************************************************************
* NAME: flash_api.c
*----------------------------------------------------------------------------
*****************************************************************************/

/*_____ I N C L U D E S ____________________________________________________*/

#include "config.h"

/*_____ M A C R O S ________________________________________________________*/

/*_____ D E F I N I T I O N ________________________________________________*/

Uchar   data api_command _at_ 0x1C;
Uchar   data api_value _at_ 0x1D;
Uchar   data api_dpl  _at_ 0x1F;
Uchar   data api_dph  _at_ 0x1E;


#define MSK_AUXR1_ENBOOT           0x20
#define MAP_BOOT                   AUXR1 |= MSK_AUXR1_ENBOOT;
#define UNMAP_BOOT                 AUXR1 &= ~MSK_AUXR1_ENBOOT;

#define __API_FLASH_ENTRY_POINT (*((const void(code*)(void)) 0xFFC0 ))
#define __API_JMP_ISP_START (*((const void(code*)(void)) 0xFF00 ))
#define __API_JMP_BOOTLOADER (*((const void(code*)(void)) 0xF800 ))

/*_____ D E C L A R A T I O N ______________________________________________*/


/*F*************************************************************************
* NAME: __api_rd_generic
*----------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*----------------------------------------------------------------------------
* PARAMS:
* Uchar command: api_command to call
* Uchar dpl      : api_dpl
* return:
* Uchar : read value
*----------------------------------------------------------------------------
* PURPOSE:
* This function allows to read xaf and special area
*****************************************************************************
* NOTE:
*****************************************************************************/
Uchar __api_rd_generic (Uchar command, Uchar dpl)
{
   api_command = command;
   api_dpl    = dpl;
   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;

   return(api_value);
}

/*F*************************************************************************
* NAME: __api_wr_generic
*----------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*----------------------------------------------------------------------------
* PARAMS:
* Uchar command: api_command to call
* Uchar value   : api_value
* Uchar dpl      : api_dpl
* return:
* Uchar return: command status (1 - ok)
*----------------------------------------------------------------------------
* PURPOSE:
* This function allows to write in xaf
*****************************************************************************
* NOTE:
*****************************************************************************/
```

```
Uchar __api_wr_generic (Uchar command, Uchar value, Uchar dpl)
{

   bit ea_save;

   ea_save = EA;
   EA = 0;
   api_command = command;
   api_dpl     = dpl;
   api_value   = value;
   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;
   EA = ea_save;      // restore interrupt state
   return(1);

}


/*F****************************************************************************
* NAME: __api_rd_code_byte
*-----------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*-----------------------------------------------------------------------------
* PARAMS:
* Uint16 address : address in flash memory to read
* return:
* Uchar device : read value
*-----------------------------------------------------------------------------
* PURPOSE:
* This function allows to read a flash memory byte.
*****************************************************************************
* NOTE:
* To use this function the constante __API_RD_CODE_BYTE must be define in
* C header file api_cc01.h.
*****************************************************************************/
#ifdef __API_RD_CODE_BYTE
Uchar __api_rd_code_byte (Uchar code * pt_address)
{
   return(*pt_address);
}
#endif

/*F****************************************************************************
* NAME: __api_wr_code_byte
*-----------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*-----------------------------------------------------------------------------
* PARAMS:
* Uint16 address : address to program
* Uchar value : data to write
* return:
* Uchar return :
*          return = 0x00 -> pass
*          return != 0x00 -> fail
*-----------------------------------------------------------------------------
* PURPOSE:
* This function allows to program data byte in Flash memory.
*****************************************************************************
* NOTE:
* To use this function the constante __API_WR_CODE_BYTE must be define in
* C header file api_cc01.h.
*****************************************************************************/
#ifdef __API_WR_CODE_BYTE
Uchar __api_wr_code_byte (Uchar xdata * pt_address, Uchar value)
{
   bit ea_save;

   ea_save = EA;
   EA = 0;
   api_command = _COMMAND_WR_CODE_BYTE;
   FCON = 0x08;

   *pt_address = value;

   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;
   EA = ea_save;      // restore interrupt state

   return(api_value);
```

```
}
#endif

/*F*****************************************************************************
* NAME: __api_wr_code_page
*------------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*------------------------------------------------------------------------------
* 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
* return:
* Uchar return :
*          return = 0x00 -> pass
*          return != 0x00 -> fail
*------------------------------------------------------------------------------
* PURPOSE:
* This function allows to 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.
*******************************************************************************
* NOTE:
* To use this function the constante __API_WR_CODE_PAGE must be define in
* C header file api_cc01.h.
* This function used Dual Data Pointer DPTR0&1. At the end of this function
* DPTR = DPTR0.
*******************************************************************************/
#ifdef __API_WR_CODE_PAGE
Uchar __api_wr_code_page (Uchar xdata * pt_code, Uchar xdata * pt_xram, Uchar nb_data)
{
   Uchar data i, temp, temp_nb_data;
   bit ea_save;
   Uint16   data add_pt_code, add_pt_xram;

   add_pt_xram = pt_xram;
   add_pt_code = pt_code;
   temp_nb_data = nb_data;
   ea_save = EA;
   EA = 0;
   api_command = _COMMAND_WR_CODE_BYTE;
   for (i=0 ; i< temp_nb_data; i++,add_pt_xram++,add_pt_code++)
   {
              temp = *(Uchar xdata *)add_pt_xram;
              FCON = 0x08;
              *(Uchar xdata *)add_pt_code = temp;
              FCON = 0x00;
   }

   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;
   EA = ea_save;      // restore interrupt state

   return(api_value);
}
#endif

/*F*****************************************************************************
* NAME: __api_wr_fuse
*------------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*------------------------------------------------------------------------------
* PARAMS:
* Uchar return :
*------------------------------------------------------------------------------
* PURPOSE:
*******************************************************************************
* NOTE:
*******************************************************************************/
Uchar __api_wr_fuse (Uchar mask, Uchar filter)
{
   Uchar value;
   bit ea_save;

   ea_save = EA;
   EA = 0;
   value   = __api_rd_HSB();
   value &= ~mask;
   api_value    = value | filter;
   api_command = _COMMAND_WR_FUSE_BIT;
```

```
   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;
   EA = ea_save;      // restore interrupt state

   return(1);
}


/*F****************************************************************************
*  NAME: api_erase_block
*-----------------------------------------------------------------------------
*  AUTHOR: Jean-Sebastien Berthy
*-----------------------------------------------------------------------------
*  PARAMS:
*  block_t num_block
*             num_block = BLOCK_0 (erase Flash between 0x0000-0x1FFF)
*             num_block = BLOCK_1 (erase Flash between 0x2000-0x3FFF)
*             num_block = BLOCK_2 (erase Flash between 0x3000-0x7FFF)
 return:
*-----------------------------------------------------------------------------
*  PURPOSE:
*  This function allows to erase Block in Flash.
*******************************************************************************
*  NOTE:
*  To use this function the constante __API_ERASE_BLOCK must be define in
*  C header file api_cc01.h.
*******************************************************************************/
#ifdef __API_ERASE_BLOCK
Uchar __api_erase_block (block_t num_block)
{
   bit ea_save;

   ea_save = EA;
   EA = 0;
   api_command = _COMMAND_ER_BLOCK;
   api_dph       = num_block;
   MAP_BOOT;
   __API_FLASH_ENTRY_POINT();
   UNMAP_BOOT;
   EA = ea_save;      // restore interrupt state

   return(1);
}
#endif

/*F****************************************************************************
*  NAME: api_eeprom_busy
*-----------------------------------------------------------------------------
*  AUTHOR: Jean-Sebastien Berthy
*-----------------------------------------------------------------------------
*  PARAMS:
*  return:
*  eeprom_t eep :
*           eep = EEPROM_NOT_BUSY
*           eep = EEPROM_BUSY
*-----------------------------------------------------------------------------
*  PURPOSE:
*  This function allows to check if eeprom is busy or not.
*******************************************************************************
*  NOTE:
*  To use this function the constante __API_EEPROM_BUSY must be define in
*  C header file api_cc01.h.
*******************************************************************************/
#ifdef __API_EEPROM_BUSY
eeprom_t __api_eeprom_busy (void)
{
   return(EECON & 0x01);
}
#endif

/*F****************************************************************************
*  NAME: api_rd_eeprom
*-----------------------------------------------------------------------------
*  AUTHOR: Jean-Sebastien Berthy
*-----------------------------------------------------------------------------
*  PARAMS:
*  Uchar xdata *address : address to read
*  return:
*-----------------------------------------------------------------------------
```

```
* PURPOSE:
* This function allows to read a byte in Eeprom.
*******************************************************************************
* NOTE:
* To use this function the constante __API_RD_EEPROM must be define in
* C header file api_cc01.h.
*******************************************************************************/
#ifdef __API_RD_EEPROM_BYTE
Uchar __api_rd_eeprom_byte(Uchar xdata *address)
{
   Uchar val;

   EECON = 0x02;
   val    = *address;
   EECON = 0x00;

   return (val);
}
#endif

/*F****************************************************************************
* NAME: api_wr_eeprom_byte
*-------------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*-------------------------------------------------------------------------------
* PARAMS:
* Uchar xdata* address : address to read
* Uchar value : data to write
* return:
*-------------------------------------------------------------------------------
* PURPOSE:
* This function allows to program a byte in Eeprom.
*******************************************************************************
* NOTE:
* To use this function the constante __API_WR_EEPROM_BYTE must be define in
* C header file api_cc01.h.
*******************************************************************************/
#ifdef __API_WR_EEPROM_BYTE
Uchar __api_wr_eeprom_byte (Uchar xdata *address, Uchar value)
{
   bit ea_save;

   ea_save = EA;
   EA = 0;
   EECON = 0x02;
   *address = value;/* addr is a pointer to external data mem */
   EECON = 0x50;
   EECON = 0xA0;
   EA = ea_save;

   return (1);
}
#endif

/*F****************************************************************************
* NAME: api_start_isp
*-------------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
*-------------------------------------------------------------------------------
* PARAMS:
* return:
*-------------------------------------------------------------------------------
* PURPOSE:
*******************************************************************************
* NOTE:
* To use this function the constante __API_START_ISP must be define in
* C header file api_cc01.h.
*******************************************************************************/
#ifdef __API_START_ISP
void __api_start_isp (void)
{
   EA = 0;
   MAP_BOOT;
   __API_JMP_ISP_START();
}
#endif

/*F****************************************************************************
* NAME: api_start_bootloader
*-------------------------------------------------------------------------------
* AUTHOR: Jean-Sebastien Berthy
```

```
*-------------------------------------------------------------------------
*  PARAMS:
*  return:
*-------------------------------------------------------------------------
*  PURPOSE:
***************************************************************************
*  NOTE:
*  To use this function the constante __API_START_BOOTLOADER must be define in
*  C header file.
***************************************************************************/
#ifdef __API_START_BOOTLOADER
void __api_start_bootloader (void)
{
   EA = 0;
   MAP_BOOT;
   __API_JMP_BOOTLOADER();
}
#endif
```

6. Bibliography

- ***Datasheet T89C51CC01***

- ***T89C51CC01 CAN Bootloader***
  "In-System Programming - T89C51CC01 CAN bootloader Description".