C Flash Drivers for the T89C51CC02CA for Keil Compilers

This application note describes C routines for Keil® compiler to perform In-application Programming/Self programming according to the “T89C51CC02CA CAN Bootloader Datasheet”.

The T89C51CC02CA provides on-chip CAN Bootloader which contains routines to perform In-Application Programming/Self Programming.

The routine may be called at one entry point but with varying values in globals variables to perform the following operations:

- Read/Write Flash and EEPROM memory
- Read/Write Device ID
- Block Erase
- Read/Write Configuration Byte
- Bootloader start

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

Rev. 4213A–CAN–12/02

Interface with CAN Bootloader

The interface is decribed in the T89C51CC02CA CAN Bootloader Datasheet (see section In-Application Programming/Self Programming in the T89C51CC02CA CAN Bootloader Datasheet).

Library Usage

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.

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 the config.h can easily be added to one of your include files. You also need to include the standart "t89c51cc02.h" file.

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

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 (see T89C51CC02CA Datasheet).

Description

Types Description

Uchar : unsigned char

Uint16 : unsigned short

ssb_t : enum

NO_SECURITY
WR_SECURITY
RD_WR_SECURITY

eeprom_t : enum

EEPROM_NOT_BUSY
EEPROM_BUSY

block_t : enum

BLOCK_0
BLOCK_1
BLOCK_2

Functions

Function	Parameters	Return
__api_rd_code_byte	adress	value
__api_wr_code_byte	address, value	state
__api_wr_code_page	pt_code, pt_xram, nb_data	state
__api_erase_block	num_block	state
__api_rd_HSB	-	value
__api_set_X2	-	state
__api_clr_X2	-	state
__api_set_BLJB	-	state
__api_clr_BLJB	-	state
__api_rd_BSB	-	value
__api_wr_BSB	value	state
__api_rd_SBV	-	value
__api_wr_SBV	value	state
__api_erase_SBV	-	state
__api_rd_SSB	-	value
__api_wr_SSB	value	state
__api_rd_EB	-	value
__api_wr_EB	value	state
__api_rd_CANBTC1	-	value
__api_wr_CANBTC1	value	state

Function	Parameters	Return
__api_rd_CANBTC2	-	value
__api_wr_CANBTC2	value	state
__api_rd_CANBTC3	-	value
__api_wr_CANBTC3	value	state
__api_rd_NNB	-	value
__api_wr_NNB	value	state
__api_rd_CRIS	-	value
__api_wr_CRIS	value	state
__api_rd_manufacturer	-	value
__api_rd_device_id1	-	value
__api_rd_device_id2	-	value
__api_rd_device_id3	-	value
__api_rd_bootloader_version	-	value
__api_eeprom_busy)	-	state
__api_rd_eeprom_byte	address	value
__api_wr_eeprom_byte	address, value	state
__api_start_bootloader	-	-
__api_start_isp	-	-

Functions Description

__api_rd_code_byte

This macro is used to read a byte value in Flash memory on a 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 (Uchar code * pt_address)
- Input:
  - pt_address: address of the byte to read in the Flash memory
- Output:
  - return value read
- Example:

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

__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 (Uchar xdata* address, Uchar value)
- Input:
  - pt_address: address where byte must be written
  - value: byte to write in Flash memory
- Output:
  - return = 0x00 -> program success
    return != 0x00 -> program fail
- Example:

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

__api_wr_code_page

This function writes up to 128 bytes from XRAM to Flash memory from a 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.

All data must be in the same page (see section “program code/memory” in the T89C51CC02CA Datasheet).

- Prototype:
  - Uchar __api_wr_code_page (Uchar xdata * pt_code,Uchar xdata * pt_xram, Uchar nb_data)
- Input:
  - pt_code: pointer on the first Flash memory address to write
  - pt_xram: pointer on the first XRAM data to read
  - nb_data: number of byte to write in Flash memory
    No more than 128 bytes.

- Output:
  - 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 */
```

__api_erase_block

This function is used to erase one of the 3 blocks in Flash memory:

BLOCK_0: 0000h to 1FFFh

BLOCK_1: 2000h to 3FFFh

BLOCK_2: 4000h to 7FFFh

To use this function __API_ERASE_BLOCK constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - Uchar __api_erase_block (block_t num_block)
- Input:
  - num_block: BLOCK_0, BLOCK_1 or BLOCK_2
- Output:
  - state: not used
- Example:

```
__api_erase_block (BLOCK_1);
```

Generic Reading Function

The following functions used the same generic reading function:

__api_rd_HSB
__api_rd_BSB
__api_rd_SBV
__api_rd_SSB
__api_rd_EB
__api_rd_CANBTC1
__api_rd_CANBTC2
__api_rd_CANBTC3
__api_rd_CRIS
__api_rd_NNB
__api_rd_manufacturer
__api_rd_device_id1
__api_rd_device_id2
__api_rd_device_id3
__api_rd_bootloader

To use one of these functions the corresponding constant (same name than the function writes in uppercase) must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - Uchar __api_rd_XX (void)
- Input:
  - void
- Output:
  - XX read
- Example:

```
Uchar XX_value;
XX_value = __api_rd_XX ();
```

Generic Writing Function

The following functions use the same generic writing function:

__api_wr_HSB
__api_wr_BSB
__api_wr_SBV
__api_wr_SSB
__api_wr_EB
__api_wr_CANBTC1
__api_wr_CANBTC2
__api_wr_CANBTC3
__api_wr_CRIS
__api_wr_NNB

To use one of these functions the corresponding constant (same name that the function writes in uppercase) must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - Uchar __api_wr_XX (Uchar XX)
- Input:
  - XX: value of XX to write
- Output:
  - state: not used
- Example:

```
__api_wr_XX (0x80);
```

__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:
  - state: not used
- Example:

```
__api_erase_SBV ();
```

__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:
  - state: not used
- Example:

```
__api_set_X2 ();
```

__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:
  - state: not used
- Example:

```
__api_clr_X2 ();
```

__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:
  - state: not used
- Example:

```
__api_set_BLJB ();
```

__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:
  - state: not used
- Example:

```
__api_clr_BLJB ();
```

__api_rd_eeprom_byte

This function is used to read a byte in the EEPROM Data. 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 (Uchar xdata* address)
- Input:
  - address to read the byte in EEPROM data
- Output:
  - value read
- Example:

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

__api_eeprom_busy

This function is used to check if an EEPROM write cycle is ongoing.

This function return EEPROM_BUSY if the EEPROM is programming. To use this function __API_EEPROM_BUSY constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - eeprom_t __api_eeprom_busy (void)
- Input:
  - void
- Output:
  - eeprom state: EEPROM_BUSY or EEPROM_NOT_BUSY
- Example:

```
while(__api_eeprom_busy ()==EEPROM_BUSY);
```

__api_wr_eeprom_byte

This function is used to write a byte in the EEPROM Data.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 (Uchar xdata * address, Uchar value)
- Input:
  - address to write
  - value to write
- Output:
  - state: not used
- Example:

```
if(__api_eeprom_busy ()==EEPROM_NOT_BUSY)
{
   __api_wr_eeprom_byte(0x25, 0x55);
}
```

__api_start_bootloader

This function is used to start the bootloader at address F800h. To use this function __API_START_BOOTLOADER constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - void __api_start_bootloader (void)
- Input:
  - void
- Output:
  - void
- Example:

```
__api_start_bootloader();
```

__api_start_isp

This function is used to start the bootloader at address FF00h. To use this function __API_START_ISP constant must be defined in flash_api.h file, otherwise the function is not compiled.

- Prototype:
  - void __api_start_isp (void)
- Input:
  - void
- Output:
  - void
- Example:

```
__api_start_isp();
```

Note: There are two possible start bootloaders:

- The __api_start_booloader routine allows to start at the beginning of the bootloader as after a reset. After calling this routine the regular boot process is performed and the communication must be open before any action.
- The __api_start_isp routine allows to start the bootloader with the CAN bit configuration of the application and start with the state "communication open". That means the bootloader will return the message “id_select_node” with the field com port open.

References

- Datasheet T89C51CC02CA
- Datasheet Bootloader CAN T89C51CC02CA

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Data-com
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

4213A–CAN–12/02 /0M