

---

# API Program Examples

## References

- Atmel 8051 Microcontrollers Hardware Manual



---

**8051  
Microcontrollers**

---

**Application Note**

Rev. 4365A–80C51–07/04





## 1. Introduction

This Application Note provides to customers C program examples for Api usages.

## 2. API for Standard C51

### 2.1 flash\_eeprom\_api.c

```
/*C*****
* NAME:          flash_eeprom_api.c
* -----
* Copyright (c) 2004 Atmel.
* -----
* RELEASE:
* REVISION:      1.0
* -----
* PURPOSE:
* Read/Write flash
* CAUTION : add #define ONCHIP_EEPROM for on-chip eeprom products
*          (defined by default in the standard delivery)
*****/

/*_____ I N C L U D E - F I L E S _____*/
#include "reg_C51.h"

/*_____ D E C L A R A T I O N _____*/
#define ONCHIP_EEPROM //define it only for on-chip eeprom products
/*---- API for FLASH access -----*/
/*****/
#define __api_rd_code_byte(address) ((*((unsigned char code*) (address)))
unsigned char __api_wr_code_byte(int , unsigned char)small;
unsigned char __api_wr_code_page(int , int, unsigned char)small;
/*---- API for EEPROM access -----*/
/*****/
#ifdef ONCHIP_EEPROM
    void __api_wr_eeprom_byte(unsigned int adr, unsigned char value);
    unsigned char __api_rd_eeprom_byte(unsigned int adr);
#endif

/*_____ G L O B A L S _____*/
sfr16 DPTR    =    0x82;

/*_____ L O C A L S _____*/
#define MSK_AUXR1_ENBOOT0x20
#define MSK_AUXR_M00x20
#define MAP_BOOT_AUXR1 |= MSK_AUXR1_ENBOOT;
#define UNMAP_BOOTAUXR1 &= ~MSK_AUXR1_ENBOOT;

/*_____ E X T E R N A L - 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_code_byte
* -----
* PARAMS:
*   int address : address to program
*   unsigned char value : data to write
*   unsigned char return :
*       return = 0x00 -> pass
*       return != 0x00 -> fail
* -----
* PURPOSE:
*   Program data byte in Flash memory
*****/
unsigned char __api_wr_code_byte (int address, unsigned char value) small
{
    bit ea_save;
    ea_save = EA;
    EA = 0;
    DPTR = address;
    ACC = 0x02;
    ASM_MOV_R1_A();
    ACC = value;
    MAP_BOOT;
    __API_FLASH_ENTRY_POINT();
    UNMAP_BOOT;
    EA = ea_save;    // restore interrupt state
    return (ACC);
}

/*F*****
* NAME: __api_wr_code_page
* -----
* PARAMS:
*   int add_flash : address of the first byte to program in the Flash
*   int add_xram : address in XRAM of the first data to program
*   unsigned char nb_data : number of bytes to program
*   unsigned char 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.
*****/
unsigned char __api_wr_code_page (int add_flash, int add_xram, unsigned char
nb_data) small
{
    unsigned char save_auxr1;
    bit ea_save;

```

```

    ea_save = EA;
    EA = 0;
    save_auxr1 = AUXR1;
    AUXR1 &= ~0x01;          // Set DPTR=DPTR0
    DPTR = add_flash;
    AUXR1++;                 // DPTR = DPTR1
    DPTR = add_xram;
    ACC = 0x09;
    ASM_MOV_R1_A();
    ACC = nb_data;
    AUXR1 &= ~0x01;          // Set DPTR = DPTR0
    MAP_BOOT
    __API_FLASH_ENTRY_POINT();
    UNMAP_BOOT;
    AUXR1 = save_auxr1;
    EA = ea_save;           // restore interrupt state
return (ACC);
}

/*F*****
* NAME: __api_rd_eeprom_byte
* -----
* PARAMS:
* unsigned int adr: The EEDATA memory location to read
* return: value
* -----
* PURPOSE:
* This function reads one byte in the on-chip EEPROM data.
* -----
* EXAMPLE:
* val=__api_rd_eeprom_byte(128);
* -----
* NOTE:
* -----
* REQUIREMENTS:
*****/
unsigned char __api_rd_eeprom_byte(unsigned int adr)
{
    unsigned char val;
    bit ea_save;
    while (EECON&1); //Eeprom_busy()
    ea_save=EA;
    EA=0;
    EECON |= 0x02; //Enable eeprom data;
    val=(unsigned char xdata*)adr;
    EECON &= ~0x02; //Disable eeprom data;
    EA=ea_save;
    return val;
}

```

```

/*F*****
* NAME: __api_wr_eeprom_byte
* -----
* PARAMS:
* unsigned int adr: The EEDATA memory location to read
* unsigned char value: The data byte to write
* return: none
* -----
* PURPOSE:
* This function writes one byte in the on-chip EEPROM data.
* -----
* EXAMPLE:
* -----
* NOTE:
* -----
* REQUIREMENTS:
*****/
void __api_wr_eeprom_byte(unsigned int adr, unsigned char value)
{
    bit ea_save;
    while(EECON & 0x01); // wait bit busy
    ea_save=EA;
    EA=0;
    EECON |= 0x02; //Enable eeprom data
    *(unsigned char xdata*)adr=value;
    EECON &= ~0x02; //Disable eeprom data
    EA=ea_save;
}

```

## 2.2 flash\_lib.a51

```
NAME FLASH_LIB;

; *A51*****
; FILE_NAME      : FLASH_LIB.a51
; -----
; -----
; FILE_PURPOSE: low level function for API
; *****

USING 0

PUBLIC ASM_MOV_R1_A
PUBLIC __API_FLASH_ENTRY_POINT

AUXR1 EQU 0A2h

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
    POPAR6
    POPAR4
    POPAR2
Ret

END
```

## 3. API for USB products

### 3.1 flash\_eeprom\_api.c

```

/*C*****
* NAME:          flash_eeprom_api.c
* -----
* Copyright (c) 2004 Atmel.
* -----
* RELEASE:
* REVISION:
* -----
* PURPOSE:
* This file contains whole of functions to access AT89C5131 Flash and
* EEPROM and AT89C51SND1.
* CAUTION : add #define ONCHIP_EEPROM for on-chip eeprom products
*           (defined by default in the standard delivery)
*****/

/*_____ I N C L U D E S _____*/
#include "reg_C51.h"
/*_____ D E F I N I T I O N _____*/
#define ONCHIP_EEPROM //define it only for build-in eeprom chips

unsigned char  data api_command      _at_ 0x1C;
unsigned char  data api_value        _at_ 0x1D;

#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 ))

/*_____ D E C L A R A T I O N _____*/
/*---- API for FLASH access -----*/
/*****/
unsigned char __api_rd_code_byte (unsigned char code * pt_address);
unsigned char __api_wr_code_byte (unsigned char xdata* , unsigned char);
unsigned char __api_wr_code_page (unsigned char xdata* pt_code,
                                unsigned char xdata* pt_xram,
                                unsigned char nb_data);

/*---- API for EEPROM access -----*/
/*****/
#ifdef ONCHIP_EEPROM
    unsigned char __api_rd_eeprom_byte(unsigned char xdata *);
    unsigned char __api_wr_eeprom_byte(unsigned char xdata *, unsigned char);
#endif

```

```

/*F*****
* NAME: __api_rd_code_byte
* -----
* PARAMS:
* unsigned int address : address in flash memory to read
* return:
* unsigned char device : read value
* -----
* PURPOSE:
* This function allows to read a flash memory byte.
* -----
* EXAMPLE:
* -----
* NOTE:
* -----
* REQUIREMENTS:
*****/
unsigned char __api_rd_code_byte (unsigned char code * pt_address)
{
    return(*pt_address);
}

/*F*****
* NAME: __api_wr_code_byte
* -----
* PARAMS:
* unsigned int address : address to program
* unsigned char value : data to write
* return:
* unsigned char return :
*     return = 0x00 -> pass
*     return != 0x00 -> fail
* -----
* PURPOSE:
* This function allows to program data byte in Flash memory.
* -----
* EXAMPLE:
* -----
* NOTE:
* -----
* REQUIREMENTS:
*****/
unsigned char __api_wr_code_byte (unsigned char xdata * pt_address,
                                unsigned char value)
{
    bit ea_save;

    ea_save = EA;
    EA = 0;
    api_command = 0x0D; //_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);
}

/*F*****
* NAME: __api_wr_code_page
* -----
* PARAMS:
* unsigned int add_flash : address of the first byte to program
*                      in the Flash
* unsigned int add_xram  : address in XRAM of the first data to program
* unsigned char nb_data  : number of bytes to program
* return:
* unsigned char 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.
* -----
* EXAMPLE:
* -----
* NOTE:
* This function used Dual Data Pointer DPTR0&1. At the end of this
* function.
* DPTR = DPTR0.
* -----
* REQUIREMENTS:
*****/
unsigned char __api_wr_code_page (unsigned char xdata * pt_code,
                                unsigned char xdata * pt_xram,
                                unsigned char nb_data)
{
    unsigned char data i, temp, temp_nb_data;
    bit ea_save;
    unsigned int  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 = 0x0D;
    for (i=0 ; i< temp_nb_data; i++,add_pt_xram++,add_pt_code++)
    {
        temp = *(unsigned char xdata *)add_pt_xram;
        FCON = 0x08;
        *(unsigned char xdata *)add_pt_code = temp;
        FCON = 0x00;
    }

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

    return(api_value);
}

#ifdef ONCHIP_EEPROM
/*F*****
* NAME: api_rd_eeprom
*-----
* PARAMS:
* unsigned char xdata *address : address to read
* return:
*-----
* PURPOSE:
* This function allows to read a byte in Eeprom.
*-----
* EXAMPLE:
*-----
* NOTE:
*-----
* REQUIREMENTS: The EEPROM mustn't be busy to perform the read access.
*               eeprom status : (EECON & 0x01)=1 busy, =0 free
*****/
unsigned char __api_rd_eeprom_byte(unsigned char xdata *address)
{
    unsigned char val;
    bit ea_save;

    ea_save = EA;
    EA = 0;
    EECON = 0x02;
    val = *address;
    EECON = 0x00;
    EA = ea_save;
    return (val);
}

```

```

/*F*****
* NAME: api_wr_eeprom_byte
* -----
* PARAMS:
* unsigned char xdata* address : address to read
* unsigned char value : data to write
* return:
* -----
* PURPOSE:
* This function allows to program a byte in Eeprom.
* -----
* EXAMPLE:
* -----
* NOTE:
* -----
* REQUIREMENTS: The EEPROM mustn't be busy to perform the read access.
*               eeprom status : (EECON & 0x01)=1 busy, =0 free
*****/
unsigned char __api_wr_eeprom_byte (unsigned char xdata *address,
                                   unsigned char value)
{
    bit ea_save;
    while(EECON & 0x01); // wait bit busy
    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

```

## 4. Example

### 4.1 test\_api.c

```
/*C*****
* NAME:          test_api.c
*-----
* Copyright (c) 2004 Atmel.
*-----
* RELEASE:
* REVISION:      1.0
*-----
* PURPOSE: usage example of flash_eeprom_api.c
*****/

/*_____ I N C L U D E S _____*/
#include "flash_eeprom_api.c"

/*F*****
* NAME: main
*-----
* PARAMS:
*-----
* PURPOSE: usage example of flash_eeprom_api.c
*-----
* EXAMPLE:
*-----
* NOTE:
*-----
* REQUIREMENTS:
*****/
void main (void)
{
    int address;
    char i=0;
    char data_tmp;

    /* write code page example */
    for(address=0x0000;address<0x007F;address++)
    {
        /* write 0x55 between 0x0000 and 0x007F of xram */
        *((unsigned char xdata*) address)=0x55;
    }
    /* copy xram page to flash at 0x1100 */
    __api_wr_code_page(0x1100,0x0000,0x7F);

    /* write code byte example */
    i=0;
    for(address=0x1000;address<0x1006;address++)
    {
```

```

/* write "ABCDEF" at 0x1000 */
__api_wr_code_byte(address,0x41+i++);
}

/* read and write code byte example */
for(address=0x1000;address<0x1006;address++)
{
/* copy 0x1000-0x1006 to 0x1010-0x1016 in flash memory */
data_tmp = __api_rd_code_byte(address);
__api_wr_code_byte(address+0x0010,data_tmp);
}

#ifdef ONCHIP_EEPROM
/* write eeprom byte example */
i=0;
for (address=0x000;address<0x400;address++) /* write 1Ko of eeprom */
{
/* write a byte, write time = 10ms to 20ms */
__api_wr_eeprom_byte(address,i++);
}

/* read eeprom byte example */
for (address=0x000;address<0x400;address++)
{
/* copy previous writed data from eeprom to flash at 0x1200 */
data_tmp=__api_rd_eeprom_byte(address);
__api_wr_code_byte(0x1200+address,data_tmp);
}

#endif
while(1); /* endless */
}

```

## 4.2 SFR Register Definition

```
/*H*****
**
* NAME: AT89C51XD2.h
*-----
-
* PURPOSE: SFR Description file for AT89C51xD2 products
*         ON KEIL compiler
*****
*/

#define Sfr(x, y)  sfr x = y
#define Sbit(x, y, z)  sbit x = y^z
#define Sfr16(x,y)    sfr16 x = y

/*-----*/
/* Include file for 8051 SFR Definitions */
/*-----*/

/* BYTE Register */
Sfr (P0 , 0x80);

Sbit (P0_7 , 0x80, 7);
Sbit (P0_6 , 0x80, 6);
Sbit (P0_5 , 0x80, 5);
Sbit (P0_4 , 0x80, 4);
Sbit (P0_3 , 0x80, 3);
Sbit (P0_2 , 0x80, 2);
Sbit (P0_1 , 0x80, 1);
Sbit (P0_0 , 0x80, 0);

Sfr (P1 , 0x90);

Sbit (P1_7 , 0x90, 7);
Sbit (P1_6 , 0x90, 6);
Sbit (P1_5 , 0x90, 5);
Sbit (P1_4 , 0x90, 4);
Sbit (P1_3 , 0x90, 3);
Sbit (P1_2 , 0x90, 2);
Sbit (P1_1 , 0x90, 1);
Sbit (P1_0 , 0x90, 0);

Sfr (P2 , 0xA0);
Sbit (P2_7 , 0xA0, 7);
Sbit (P2_6 , 0xA0, 6);
Sbit (P2_5 , 0xA0, 5);
Sbit (P2_4 , 0xA0, 4);
Sbit (P2_3 , 0xA0, 3);
```

```
Sbit (P2_2 , 0xA0, 2);
Sbit (P2_1 , 0xA0, 1);
Sbit (P2_0 , 0xA0, 0);
```

```
Sfr (P3 , 0xB0);
```

```
Sbit (P3_7 , 0xB0, 7);
Sbit (P3_6 , 0xB0, 6);
Sbit (P3_5 , 0xB0, 5);
Sbit (P3_4 , 0xB0, 4);
Sbit (P3_3 , 0xB0, 3);
Sbit (P3_2 , 0xB0, 2);
Sbit (P3_1 , 0xB0, 1);
Sbit (P3_0 , 0xB0, 0);
```

```
Sbit (RD , 0xB0, 7);
Sbit (WR , 0xB0, 6);
Sbit (T1 , 0xB0, 5);
Sbit (T0 , 0xB0, 4);
Sbit (INT1 , 0xB0, 3);
Sbit (INT0 , 0xB0, 2);
Sbit (TXD , 0xB0, 1);
Sbit (RXD , 0xB0, 0);
```

```
Sfr (P4 , 0xC0);
```

```
Sbit (P4_7 , 0xC0, 7);
Sbit (P4_6 , 0xC0, 6);
Sbit (P4_5 , 0xC0, 5);
Sbit (P4_4 , 0xC0, 4);
Sbit (P4_3 , 0xC0, 3);
Sbit (P4_2 , 0xC0, 2);
Sbit (P4_1 , 0xC0, 1);
Sbit (P4_0 , 0xC0, 0);
```

```
Sfr (P5 , 0xE8);
```

```
Sbit (P5_7 , 0xE8, 7);
Sbit (P5_6 , 0xE8, 6);
Sbit (P5_5 , 0xE8, 5);
Sbit (P5_4 , 0xE8, 4);
Sbit (P5_3 , 0xE8, 3);
Sbit (P5_2 , 0xE8, 2);
Sbit (P5_1 , 0xE8, 1);
Sbit (P5_0 , 0xE8, 0);
```

```
Sfr (PSW , 0xD0);
```

```
Sbit (CY , 0xD0 , 7);
Sbit (AC , 0xD0 , 6);
```

```

Sbit (F0 , 0xD0 , 5);
Sbit (RS1 , 0xD0 , 4);
Sbit (RS0 , 0xD0 , 3);
Sbit (OV , 0xD0 , 2);
Sbit (UD , 0xD0 , 1);
Sbit (P , 0xD0 , 0);

Sfr (ACC , 0xE0);
Sfr (B , 0xF0);
Sfr (SP , 0x81);
Sfr (DPL , 0x82);
Sfr (DPH , 0x83);

Sfr (PCON , 0x87);
Sfr (CKCON0 , 0x8F);
Sfr (CKCON1 , 0xAF);

/*----- TIMERS registers -----*/
Sfr (TCON , 0x88);
Sbit (TF1 , 0x88, 7);
Sbit (TR1 , 0x88, 6);
Sbit (TF0 , 0x88, 5);
Sbit (TR0 , 0x88, 4);
Sbit (IE1 , 0x88, 3);
Sbit (IT1 , 0x88, 2);
Sbit (IE0 , 0x88, 1);
Sbit (IT0 , 0x88, 0);

Sfr (TMOD , 0x89);

Sfr (T2CON , 0xC8);
Sbit (TF2 , 0xC8, 7);
Sbit (EXF2 , 0xC8, 6);
Sbit (RCLK , 0xC8, 5);
Sbit (TCLK , 0xC8, 4);
Sbit (EXEN2 , 0xC8, 3);
Sbit (TR2 , 0xC8, 2);
Sbit (C_T2 , 0xC8, 1);
Sbit (CP_RL2 , 0xC8, 0);

Sfr (T2MOD , 0xC9);
Sfr (TL0 , 0x8A);
Sfr (TL1 , 0x8B);
Sfr (TL2 , 0xCC);
Sfr (TH0 , 0x8C);
Sfr (TH1 , 0x8D);
Sfr (TH2 , 0xCD);
Sfr (RCAP2L , 0xCA);
Sfr (RCAP2H , 0xCB);
Sfr (WDTRST , 0xA6);

```

```

Sfr (WDTPRG , 0xA7);

/*----- UART registers -----*/
Sfr (SCON , 0x98);
Sbit (SM0 , 0x98, 7);
Sbit (FE , 0x98, 7);
Sbit (SM1 , 0x98, 6);
Sbit (SM2 , 0x98, 5);
Sbit (REN , 0x98, 4);
Sbit (TB8 , 0x98, 3);
Sbit (RB8 , 0x98, 2);
Sbit (TI , 0x98, 1);
Sbit (RI , 0x98, 0);

Sfr (SBUF , 0x99);
Sfr (SADEN , 0xB9);
Sfr (SADDR , 0xA9);

/*----- Internal Baud Rate Generator -----*/
Sfr (BRL , 0x9A);
Sfr (BDRCON , 0x9B);

/*----- IT registers -----*/
Sfr (IEN0 , 0xA8);
Sfr (IEN1 , 0xB1);
Sfr (IPH0 , 0xB7);
Sfr (IPH1 , 0xB3);
Sfr (IPL0 , 0xB8);
Sfr (IPL1 , 0xB2);

/* IEN0 */
Sbit (EA , 0xA8, 7);
Sbit (EC , 0xA8, 6);
Sbit (ET2 , 0xA8, 5);
Sbit (ES , 0xA8, 4);
Sbit (ET1 , 0xA8, 3);
Sbit (EX1 , 0xA8, 2);
Sbit (ET0 , 0xA8, 1);
Sbit (EX0 , 0xA8, 0);

/*----- PCA registers -----*/
Sfr (CCON , 0xD8);
Sfr (CMOD , 0xD9);
Sfr (CH , 0xF9);

```

```

Sfr (CL , 0xE9);
Sfr (CCAP0H , 0xFA);
Sfr (CCAP0L , 0xEA);
Sfr (CCAPM0 , 0xDA);
Sfr (CCAP1H , 0xFB);
Sfr (CCAP1L , 0xEB);
Sfr (CCAPM1 , 0xDB);
Sfr (CCAP2H , 0xFC);
Sfr (CCAP2L , 0xEC);
Sfr (CCAPM2 , 0xDC);
Sfr (CCAP3H , 0xFD);
Sfr (CCAP3L , 0xED);
Sfr (CCAPM3 , 0xDD);
Sfr (CCAP4H , 0xFE);
Sfr (CCAP4L , 0xEE);
Sfr (CCAPM4 , 0xDE);
/* CCON */
Sbit (CF , 0xD8, 7);
Sbit (CR , 0xD8, 6);

Sbit (CCF4 , 0xD8, 4);
Sbit (CCF3 , 0xD8, 3);
Sbit (CCF2 , 0xD8, 2);
Sbit (CCF1 , 0xD8, 1);
Sbit (CCF0 , 0xD8, 0);

/*----- T W I registers -----*/
Sfr (SSCON , 0x93);
Sfr (SSCS , 0x94);
Sfr (SSDAT , 0x95);
Sfr (SSADR , 0x96);
Sfr (PI2, 0xF8);
Sbit (PI2_1 , 0xF8, 1);
Sbit (PI2_0 , 0xF8, 0);

/*----- OSC control registers -----*/
Sfr (CKSEL , 0x85 );
Sfr (OSCCON , 0x86 );
Sfr (CKRL , 0x97 );

/*----- Keyboard control registers -----*/
Sfr (KBLS , 0x9C );
Sfr (KBE , 0x9D );
Sfr (KBF , 0x9E );
/*----- SPI -----*/
Sfr (SPCON, 0xC3 );
Sfr (SPSTA, 0xC4 );
Sfr (SPDAT, 0xC5 );

```



```
/*----- Misc -----*/  
Sfr( AUXR , 0x8E);  
Sfr ( AUXR1, 0xA2);  
Sfr ( FCON, 0xD1);  
  
/*----- E data -----*/  
  
Sfr ( EECON, 0xD2 );
```



## **Atmel Corporation**

2325 Orchard Parkway  
San Jose, CA 95131  
Tel: 1(408) 441-0311  
Fax: 1(408) 487-2600

## **Regional Headquarters**

### **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 Tsimshatsui  
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 Datacom**

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](mailto:literature@atmel.com)

### **Web Site**

<http://www.atmel.com>

**Disclaimer:** Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

©Atmel Corporation 2004. All rights reserved. Atmel, the Atmel logo, and combinations thereof are registered trademarks of Atmel Corporation or its subsidiaries. Windows® Windows 98™, Windows XP™, and Windows 2000™ are trademarks and/or registered trademark of Microsoft Corporation. Other terms and product names in this document may be the trademarks of others.



Printed on recycled paper.