Flash programming in the TMP91FY42F

Used Features:

- ✓ Programming the on-chip flash memory
- ✓ Communications through the UART
- ✓ Running code in RAM
- ✓ Motorola S-record format

Application Note Category

- ☐ Software Algorithm
- ☐ MCU specific
- ☑ **System Solution**
- ☐ Basic Design Technique

Introduction

This application note describes how to erase and program the on-chip flash memory of the TMP91FY42 microcontroller.

It features a simple command line serial interface that can be used to re-program the on-chip flash memory. There are a number of commands available, including ones for erasing flash memory sectors, writing to flash memory, calculating checksums, and also returning the contents of flash memory. All program and read operations are carried out using Motorola S-records. Communications can be made with the microcontroller using a serial communications package such as Windows HyperTerminal.

The example is based on the TOPAS 900/L1 starter kit.

By:
Toshiba Electronics Europe GmbH
European LSI Design Eng. Center - **ELDEC**
Support-MCU@tee.toshiba.de

Functional Description

On reset, the application code is copied into RAM, and then run from there from that point onwards, with no further calls back to code in flash memory. This potentially means that all of the on-chip flash memory can be erased and programmed without any undue effect.

All flash memory operations are carried out through either serial I/O channel 0 or 1 on the microcontroller. This depends upon whether you are using a ROM monitor version of the software (_MONITOR is defined) or not. For a ROM monitor version, operations are carried out through channel 0, which corresponds to the serial communications interface on the board. Otherwise, operations are carried out through channel 1, which corresponds to the USB interface. In both cases the serial channel is configured in UART mode with a baud rate of 9600, 8 data bits, 1 stop bit and no parity. One command can be issued per line, and a line is terminated with the carriage return character (0x0d). Note that this is a consequence of the way Terminal programs behave. Any following new-line character (0x0a) is treated as white space. The line buffer can hold a maximum of 32 characters before overflow occurs. Should this happen an error is reported, and the line discarded.

Once a valid line has been received it is then processed. If at the end of this no errors have occurred, any action associated with the command is performed. The list of commands that the application accepts is described below. Command keywords are case insensitive, that is the keyword "CHECKSUM" is the same as the keyword "checksum". All addresses are specified in hexadecimal notation, with a preceding "0x" or "0X" character sequence. This is similar to how hexadecimal numbers are specified in C. The space (0x20), horizontal tab (0x09) and new-line (0x0a) characters are treated as white-space.

CHECKSUM COMMAND

Format: CHECKSUM *start-address end-address*
start-address: The start address of the checksum to calculate.
end-address: The end address of the checksum to calculate.

This command calculates the checksum for a section of flash memory, and returns it as a 4 digit hexadecimal number. The checksum is calculated by simply adding up all of the bytes in the section.

CHIPERASE COMMAND

Format: CHIPERASE

This command erases all of the flash memory. Unlike the ERASE command, this command also resets the flash memory read/write protection.

ERASE COMMAND

Format: ERASE *start-address* [*end-address*]
start-address: The start address of the flash memory sector(s) to erase.
end-address: The end address of the flash memory sector(s) to erase. If the *end-address* is omitted, then it is assumed to be the same as the *start-address*.

This command erases one or more sectors of flash memory, as specified by the address range. The start and end addresses do not need to line up exactly on sector address boundaries. If an address lies within a sector then it is erased. A flash memory sector in the TMP91FY42 is 4K bytes in size. With a flash size of 256K bytes, this gives a total of 64 sectors. The first sector is at address 0xFC0000 to 0xFC0FFF, the second is at address 0xFC1000 to 0xFC1FFF, all the way up to the sixty forth sector at address 0xFFF000 to 0xFFFFFF.

HELP COMMAND

Format: HELP

This command outputs a list of the commands supported by the application, along with a brief description of the command and parameters. This text is also output at start-up by the application.

ID COMMAND

Format: ID *product-id-address*
product-id-address: The address of the product ID to read.

This command reads a product ID from the flash memory. The list of product ID's, and their associated addresses are shown below.

Product ID	Address	Value
Vendor	0xYYY000	0xZZ98
Flash Macro	0xYYY002	0xZZ42
Flash Size	0xYYY004	0xZZ3F
Read/Write Protect Status	0xYYY77E	0xZZFF

The YYY specifies an address in flash memory. Only the lower 12-bits of the product ID address need to be specified in the command. The upper 12-bits (0xFFF000 in this case) are added automatically. The ZZ specifies an indeterminate value, although for the ID's it is typically 00 and for the read/write protect status it is FF. The value specified for the Read/Write Protect Status above is the default value. This can vary, depending upon whether read and/or write protection have been applied to the flash memory.

PROGRAM COMMAND

Format: PROGRAM

This command programs the flash memory with a following sequence of Motorola S-records. Each record must be separated by a new-line character, and programming is terminated when an S7, S8 or S9 record is encountered. At which point the command prompt returns.

PROTECT COMMAND

Format: PROTECT *read-write-protection*
read-write-protection: The start address of the flash memory to read.

This command sets the read/write protection for the flash memory. A read/write protection value of 0xF0 will give read protection, 0x0F will give write protection, and 0x00 will give both read and write protection. Read/write protection can only be reset by performing a chip erase.

READ COMMAND

Format: READ *start-address end-address*
start-address: The start address of the flash memory to read.
End-address: The end address of the flash memory to read.

This command reads the section of flash memory and returns it as a sequence of S3 Motorola S-records. The records are terminated by an S7 record.

MOTOROLA S-RECORD FORMAT

The Motorola S-record format is used to encode binary information in a printable (ASCII) format. Each line in a Motorola S-record format file contains exactly one S-record. An S record starts with an ASCII 'S'. This is followed by the record type, which consists of the ASCII digits '1' to '9'. Thereafter follows binary data, with each byte encoded as a 2-character hexadecimal number. The first (ASCII) character represents the high nibble of the byte, and the second (ASCII) character represents the low nibble. The binary data encompasses the record length, address, data and checksum fields of the record.

Field	Characters	Description
Type	2	The S-record type – S0, S1, S2, S3, S5, S7, S8 or S9.
Record length	2	A count of the address, data and checksum bytes.
Address	4, 6 or 8	The address at which the binary data is to be loaded into memory. This can be a 2, 3 or 4 byte address depending upon the S-record type.
Data	0 – 2n	From 0 to n bytes of binary data.
Checksum	2	The checksum. This is calculated by taking the one's complement of the sum of the record length, address and data bytes. When validating S-records, the sum of the record length, address, data and checksum fields should be 0xFF.

The table below gives some more detailed information on the various S-records.

Field	Description
S0	Specifies header information. 2 byte address is unused.
S1	A data record containing a 2 byte address and 0 to n bytes of data residing at that address.
S2	A data record containing a 3 byte address and 0 to n bytes of data residing at that address.
S3	A data record containing a 4 byte address and 0 to n bytes of data residing at that address.
S5	Specifies a count of the previous S1, S2 and S3 records in a 2 byte address.
S7	A terminating record for a file of S-records. Only one terminating record is allowed per file, and it must be on the last line of the file. This record contains a 4 byte address specifying the program entry point (typically 0x00000000). The record contains no data bytes.
S8	A terminating record for a file of S-records. Only one terminating record is allowed per file, and it must be on the last line of the file. This record contains a 3 byte address specifying the program entry point (typically 0x000000). The record contains no data bytes.
S9	A terminating record for a file of S-records. Only one terminating record is allowed per file, and it must be on the last line of the file. This record contains a 2 byte address specifying the program entry point (typically 0x0000). The record contains no data bytes.

An example of an S-record file is shown below. This specifies the text "Hello World!" and loads it at address 0xFD0000. It makes use of S3 and S7 records.

```
S31100FD000048656C6C6F20576F726C6421B4
S70500000000FA
```

WINDOWS HYPERTERMINAL

You can use the HyperTerminal application that comes with Windows, along with other serial communications programs, to communicate with the flash programming application

One important thing to be aware of is that when connecting a serial communications program to the USB interface, you may need to remove links LK2 and LK3 from the starter kit board. If not then the program can set the BOOT and RESET pins on the microcontroller into an undesirable state.

A simple test to perform with HyperTerminal is to write the "Hello World!" example above to flash memory at address 0xFD0000. First of all simply copy the Motorola S-records to a file and save them to disk. Make sure that the S7 end record has a trailing new-line! Then in HyperTerminal type the following command.

> READ 0xFD0000 0xFD001F

This should return three lines of Motorola S-records – two S3's and an S7 containing a lot of FF's and confirming that the section of memory is blank (if not issue an ERASE 0xFD0000 command). Then in HyperTerminal type the following command.

> PROGRAM

After you have pressed ENTER select the **Send Text File** command from the **Transfer** menu. In the dialog box that is displayed locate the "Hello World!" Motorola S-record file, select it and click **Open**. The file should now be downloaded to the microcontroller and programmed to flash memory. After this is complete you should receive a confirmatory message, and the command prompt should return. Then in HyperTerminal type another READ command.

> READ 0xFD0000 0xFD001F

Flash memory should now contain the text "Hello World!" at address 0xFD0000. You can also program other Motorola S-record files in this fashion. In particular, the Toshiba 900/L1 compiler generates Motorola S-record files containing program code by default.

Hardware Schematic

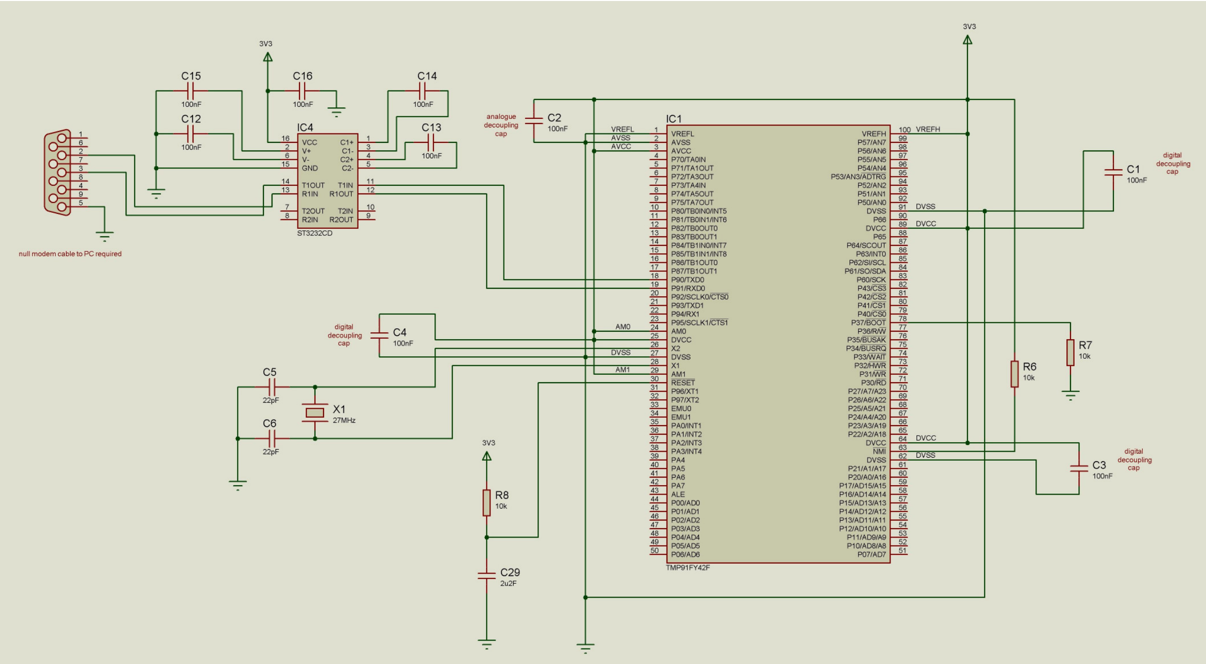

Figure 3 - The schematic.

Hardware Description

PROCESSOR BASICS

The TMP91FY42F processor is fed from the 3.3volt supply, its high-frequency clock oscillator is used with a 25 MHz crystal, giving a minimum processor instruction time of 0.16µs.

SERIAL INTERFACING

The RS232 connector on the starter kit board is connected via the RS232 level converter to pins P90/TXD0 and P91/RXD0 on the microcontroller. Serial I/O channel 0 is set to UART mode and is used to send and receive characters to and from the connector.

FLASH INTERFACING

As the Flash is internal to TMP91FY42F there is no hardware involved and no I/O pins are used up – this is the joy of on-chip Flash and an excellent reason for choosing the '42 as a host control element.

Software Description

OVERALL STRUCTURE

The C start-up code first of all copies the application code into RAM. This is performed in the macro processor file Stc91ml.mac. This ensures that the application is able to re-program all of the flash memory if required, including the memory containing itself. The _Initial function then makes a call to main, and program execution enters RAM. It will remain here until a reset (or an NMI) occurs. It is worth noting that the application from this point on cannot make any calls to standard C library functions. Any such functions used would be placed in flash memory, and hence subject to re-programming.

MAIN LOOP

The main function first initialises the UART and the communications handler. At this point, after initialisation, it would be common to enable interrupts. However, since the application does not make use of interrupts, and it is possible to re-program the flash memory containing the interrupt vectors and default interrupt handlers, they remain disabled.

The main function then outputs the start-up text and enters the main loop. The main loop then calls the Comms_OnHandle function repeatedly to process incoming commands. The command handler simply collects characters received from the UART in a buffer until it encounters a new line. If this forms a valid command, and no errors are encountered then the command, whatever it might be, is executed.

SOFTWARE FEATURES

This section describes some of the more interesting features of the application software that may be useful for designers of other or similar applications.

RE-PROGRAMMING THE ON-CHIP FLASH MEMORY

It is a necessity when programming the on-chip flash memory that the functions that are doing the programming (or erasing) are run from RAM. This application takes this one step further and copies the entire application to RAM. This ensures that all of the on-chip flash memory can be re-programmed. Another necessity when programming the on-chip flash memory is that interrupts are disabled. The occurrence and acceptance of an interrupt during a re-programming operation would cause the interrupt vector address to be looked up from flash memory. Not a good thing to happen.

The flash memory re-programming functions are located in the file Flash.c. If you have an application that say stores application data to one of the flash memory blocks, then it is only the code in this file that you would need to run from RAM. You would also need to remember to disable interrupts prior to calling the functions.

RUNNING CODE IN RAM

In order to run code in RAM you need to do three main things to your application source files.

1) You need to wrap the section of code that you want to run in RAM with a pair of **#pragma section code** and **#pragma section const** statements, and specify a name for both sections at the top. For example.

```
#pragma section code RAM_CODE
#pragma section const RAM_CONST
// functions (and constant data)
#pragma section const
#pragma section code
```

2) You need to specify the position in RAM where the code and constant data is to be mapped to. This is done in the linker command file. Below is an example linker command file that does this.

```
memory {
     IO(RW)       : origin=0x000000, length=0x001000
     IRAM(RWX)    : origin=0x001000, length=0x004000
     IROM(RX)     : origin=0xfc0000, length=0x03fef4
     PASSWORD(R)  : origin=0xfffef4, length=0x00000c
     INTTBL(R)    : origin=0xffff00, length=0x000100
}

sections {
     near_area    org=0x001000
                  : {*(n_area)}
     far_area     org=org(near_area)+sizeof(near_area)
                  : {*(f_area)}
     far_code     org=0xfc0000
                  : {*(f_code)}
     far_const    org=org(far_code)+sizeof(far_code)
                  : {*(f_const)}
     near_data    org=org=org(far_const)+sizeof(far_const)
                  addr=addr(far_area)+sizeof(far_area)
                  : {*(n_data)}
     far_data     org=org(near_data)+sizeof(near_data)
                  addr=addr(near_data)+sizeof(near_data)
                  : {*(f_data)}
     ram_code     org=org(far_data)+sizeof(far_data)
                  addr=addr(far_data)+sizeof(far_data)
                  : {*(RAM_CODE)}
     ram_const    org=org(ram_code)+sizeof(ram_code)
                  addr=addr(ram_code)+sizeof(ram_code)
                  : {*(RAM_CONST)}

     __NearDataAddr = addr(near_data);
     __FarDataAddr  = addr(far_data);
     __RamCodeAddr  = addr(ram_code);
     __RamConstAddr = addr(ram_const);
}
```

This should look fairly similar to other linker command files. There is though the addition of the extra ram_code and ram_const sections, and the definition of the __RamCodeAddr and __RamConstAddr labels. Firstly, the ram_code section states that the section will be placed just after the far data initialiser section in flash memory. This is the org part of the section definition. The ram_code section then states that the section will be relocated to just after the far data section in RAM. This is the addr part of the section definition. Note that the linker handles all of the address relocation for you regarding function calls automatically. Finally, the section will contain any code that is marked as RAM_CODE. Similarly, the ram_const section is located just after the ram_code section in both flash memory and RAM, and contains any code marked as RAM_CONST.
The __RamCodeAddr and __RamConstAddr labels then define the start addresses of the relocated code and constant data sections in RAM.

3) You need to copy the code and constant data sections from flash memory to RAM prior to calling any of the functions. This can be done anywhere, although in this case we will place the necessary code in the C start-up file, along with the near and far data section initialisation.

```
        extern  large __RamCodeAddr
        extern  large __RamConstAddr

RAM_CODE  section code    large align=2,2
RAM_CONST section romdata large align=2,2

        ; initialisation of ram code section
        ld      xde,__RamCodeAddr
        ld      xhl,startof(RAM_CODE)
        ld      xbc,sizeof(RAM_CODE)
        or      xbc,xbc
        j       z,RAM_CODE_1
        ldirb   (xde+),(xhl+)
        cp      qbc,0
        j       eq,RAM_CODE_1
        ld      wa,qbc
RAM_CODE_2:
        ldirb   (xde+),(xhl+)
        djnz    wa,RAM_CODE_2
RAM_CODE_1:

        ; initialisation of ram const section
        ld      xde,__RamConstAddr
        ld      xhl,startof(RAM_CONST)
        ld      xbc,sizeof(RAM_CONST)
        or      xbc,xbc
        j       z,RAM_CONST_1
        ldirb   (xde+),(xhl+)
        cp      qbc,0
        j       eq,RAM_CONST_1
        ld      wa,qbc
RAM_CONST_2:
        ldirb   (xde+),(xhl+)
        djnz    wa,RAM_CONST_2
RAM_CONST_1:
```

Here, in the RAM code section initialisation, XDE is loaded with the start address of the code in RAM, XHL is loaded with the start address of the code in flash memory, and XBC is loaded with the size of the section. The code is then copied from flash memory (XHL+) to RAM (XDE+) until the section size decrements to zero. This is repeatedly similarly for the RAM const section initialisation.

Disclaimer

TOSHIBA is continually working to improve the quality and reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to comply with the standards of safety in making a safe design for the entire system, and to avoid situations in which a malfunction or failure of such TOSHIBA products could cause loss of human life, bodily injury or damage to property. In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent TOSHIBA products specifications. Also, please keep in mind the precautions and conditions set forth in the "Handling Guide for Semiconductor Devices," or "TOSHIBA Semiconductor Reliability Handbook" etc..

The TOSHIBA products listed in this document are intended for usage in general electronics applications (computer, personal equipment, office equipment, measuring equipment, industrial robotics, domestic appliances, etc.). These TOSHIBA products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury ("Unintended Usage"). Unintended Usage include atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, medical instruments, all types of safety devices, etc.. Unintended Usage of TOSHIBA products listed in this document shall be made at the customer's own risk.

Toshiba assumes no liability for any damage or losses (including but not limited to, loss of business profit, business interruption, loss of business information and other pecuniary losses) occurring from the use of, or inability to use, this product.

The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA CORPORATION for any infringements of intellectual property or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any intellectual property or other rights of TOSHIBA CORPORATION or others.

The information contained herein is subject to change without notice.

OVERSEAS SUBSIDIARIES AND AFFILIATES

Toshiba Electronics Europe GmbH

Düsseldorf Head Office
Hansaallee 181, D-40549 Düsseldorf
Germany
Tel: (0211)5296-0 Fax: (0211)5296-400

München Office
Büro München Hofmannstrasse 52,
D-81378, München, Germany
Tel: (089)748595-0 Fax: (089)748595-42

Toshiba Electronics France SARL
Immeuble Robert Schumann 3 Rue de Rome,
F-93561, Rosny-Sous-Bois, Cedex, France
Tel: (1)48-12-48-12 Fax: (1)48-94-51-15

Toshiba Electronics Italiana S.R.L.
Centro Direzionale Colleoni
Palazzo Perseo Ingr. 2-Piano 6,
Via Paracelso n.12, 1-20041 Agrate Brianza
Milan, Italy
Tel: (039)68701 Fax:(039)6870205

Toshiba Electronics España, S.A
Parque Empresarial San Fernando Edificio Europa,
1a Planta, ES-28831 Madrid, Spain
Tel: (91)660-6700 Fax:(91)660-6799

Toshiba Electronics(UK) Limited
Riverside Way, Camberley Surrey,
GU15 3YA, U.K.
Tel: (01276)69-4600 Fax: (01276)69-4800

Toshiba Electronics Scandinavia AB
Gustavslundsvägen 12, 2nd Floor
S-161 15 Bromma, Sweden
Tel: (08)704-0900 Fax: (08)80-8459

Toshiba Electronics Asia

(Singapore) Pte. Ltd.
Singapore Head Office
438B Alexandra Road, #06-08/12 Alexandra
Technopark, Singapore 119968
Tel: (278)5252 Fax: (271)5155

Bangkok Office
135 Moo 5 Bangkadi Industrial Park,
Tivanon Rd.,Bangkadi Amphur Muang
Pathumthani, Bangkok, 12000, Thailand
Tel: (02)501-1635 Fax: (02)501-1638

Toshiba Electronics Trading
(Malaysia)Sdn. Bhd.
Kuala Lumpur Head Office
Suite W1203, Wisma Consplant, No.2,
Jalan SS 16/4, Subang Jaya, 47500 Petaling
Jaya, Selangor Darul Ehsan, Malaysia
Tel: (3)731-6311 Fax: (3)731-6307

Penang Office
Suite 13-1, 13th Floor, Menard Penang Garden,
26th Floor, Citibank Tower, Valero Street,
Makati, Manila, Philippines
Tel: (02)750-5510 Fax: (02)750-5511

Toshiba Electronics Philippines, Inc.
26th Floor, Citibank Tower, Valero Street,
Makati, Manila, Philippines
Tel: (02)750-5510 Fax: (02)750-5511

Toshiba America Electronic Components, Inc.

Headquarters-Irvine, CA
9775 Toledo Way, Irvine, CA 92618, U.S.A.
Tel: (949)455-2000 Fax: (949)859-3963

Boulder, CO
3100 Arapahoe Avenue, Ste. 500,
Boulder, CO 80303, U.S.A.
Tel: (303)442-3801 Fax: (303)442-7216

Boynton Beach, FL(Orlando)
11924 W. Forest Hill Blvd., Ste. 22-337,
Boynton Beach, FL 33414, U.S.A.
Tel: (561)374-6193 Fax: (561)374-6194

Deerfield, IL(Chicago)
One Pkwy., North, Suite 500, Deerfield,
IL 60015-2547, U.S.A.
Tel: (847)945-1500 Fax: (847)945-1044

Duluth, GA(Atlanta)
3700 Crestwood Parkway, Ste. 460,
Duluth, GA 30096, U.S.A.
Tel: (770)931-3363 Fax: (770)931-7602

Edison, NJ
2035 Lincoln Hwy. Ste. #3000, Edison
NJ 08817, U.S.A.
Tel: (732)248-8070 Fax: (732)248-8030

Orange County, CA
2 Venture Plaza, #500 Irvine, CA 92618, U.S.A.
Tel: (949)453-0224 Fax: (949)453-0125

Portland, OR
1700 NW 167th Place, #240,
Beaverton, OR 97006, U.S.A.
Tel: (503)629-0818 Fax: (503)629-0827

Richardson, TX(Dallas)
777 East Campbell Rd., Suite 650,
Richardson,
TX 75081, U.S.A.
Tel: (972)480-0470 Fax: (972)235-4114

San Jose Engineering Center, CA
1060 Rincon Circle, San Jose, CA 95131, U.S.A.
Tel: (408)526-2400 Fax:(408)526-2410

Wakefield, MA(Boston)
401 Edgewater Place, Suite #360, Wakefield,
MA 01880-6229, U.S.A.
Tel: (781)224-0074 Fax: (781)224-1095

Toshiba Do Brasil S.A.

Electronic Components Div.
Estrada Dos Alvarengas, 5. 500
09850-550-Sao Bernardo do campo - SP
Tel: (011)7689-7171 Fax: (011)7689-7189

Toshiba Electronics Asia, Ltd.

Hong Kong Head Office
Level 11, Top Glory Insurance Building,
Grand Century
Place, No.193, Prince Edward Road West,
Mong Kok, Kowloon, Hong Kong
Tel: 2375-6111 Fax: 2375-0969

Beijing Office
Rm 714, Beijing Fortune Building,
No.5 Dong San Huan Bei-Lu, Chao Yang
District, Beijing, 100004, China
Tel: (010)6590-8795 Fax: (010)6590-8791

Chengdu Office
Unit F, 18th Floor, New Times Plaza, 42
Wenwu Road, Xinhua Avenue, Chengdu,
610017, China
Tel: (028)675-1773 Fax: (028)675-1065

Shenzhen Office
Rm 3010-3012, Office Tower Shun Hing
Square, Di Wang Commercial Centre, 333
ShenNan East Road, Shenzhen, 518008,
China
Tel: (0755)246-1582 Fax: (0755)246-1581

Toshiba Electronics Korea Corporation
Seoul Head Office
14/F, KEC B/D, 257-7 Yangjae-Dong,
Seocho-ku, Seoul, Korea
Tel: (02)589-4334 Fax: (02)589-4302

Gumi Office
6/F, Ssangyong Investment Securities B/D,
56 Songjung-Dong, Gumi City
Kyeongbuk, Korea
Tel: (82)54-456-7613 Fax: (82)54-456-7617

Toshiba Technology Development (Shanghai) Co., Ltd.
23F, Shanghai Senmao International
Building, 101 Yin Cheng East Road, Pudong
New Area, Shanghai, 200120, China
Tel: (021)6841-0666 Fax: (021)6841-5002

Tsurong Xiamen Xiangyu Trading Co., Ltd.
8N, Xiamen SEZ Bonded Goods Market
Building, Xiamen, Fujian, 361006, China
Tel: (0592)562-3798 Fax: (0592)562-3799

Toshiba Electronics Taiwan Corporation
Taipei Head Office
17F, Union Enterprise Plaza Bldg. 109
Min Sheng East Rd., Section 3, 0446 Taipei,
Taiwan
Tel: (02)514-9988 Fax: (02)514-7892

Kaohsiung Office
16F-A, Chung-Cheng Bldg., Chung-Cheng
3Rd., 80027, Kaohsiung, Taiwan
Tel: (07)222-0826 Fax: (07)223-0046

TOSHIBA Semiconductor Websites

Europe: www.toshiba-components.com
Japan: www.semicon.toshiba.co.jp/eng/index.html
America: www.toshiba.com/taec/