

## TMP92FD54AIF

## 1. Outline and Device Characteristics

TMP92FD54AI is high-speed advanced 32-bit micro-controller developed for controlling equipment which processes mass data.

TMP92FD54AI is a micro-controller which has a high-performance CPU (900/H1 CPU) and various built-in I/Os. TMP92FD54AI is housed in a 100-pin mini flat package.

Device characteristics are as follows:

## (1) CPU : 32-bit CPU(900/H1 CPU)

Compatible with TLCS-900,900/L,900/L1,900/H,900/H2's instruction code

16Mbytes of linear address space

General-purpose register and register banks

Micro DMA : 8channels (250ns / 4bytes at  $f_c = 20\text{MHz}$ , best case)

Minimum instruction execution time : 50ns(at 20MHz)

Internal data bus : 32-bit

## (2) Internal memory

Internal RAM : 32K-byte

Internal ROM : 512K-byte Flash E2PROM

3K-byte Mask ROM (for Flash boot mode)

## RESTRICTIONS ON PRODUCT USE

060116EBP

- The information contained herein is subject to change without notice. 021023\_D
- 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. 021023\_A
- 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. 021023\_B
- The products described in this document shall not be used or embedded to any downstream products of which manufacture, use and/or sale are prohibited under any applicable laws and regulations. 060106\_Q
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others. 021023\_C
- The products described in this document are subject to the foreign exchange and foreign trade laws. 021023\_E
- For a discussion of how the reliability of microcontrollers can be predicted, please refer to Section 1.3 of the chapter entitled Quality and Reliability Assurance/Handling Precautions. 030619\_S

- (3) External memory expansion
  - 16M-byte linear address space (memory mapped I/O)
  - External data bus : 8bit(for external I/O expansion)
  - \* Can't use upper address bus when built-in I/Os are selected
- (4) Memory controller (MEMC)
  - Chip select output : 1 channel
- (5) 8-bit timer : 8 channels
  - 8-bit interval timer mode (8 channels)
  - 16-bit interval timer mode (4 channels)
  - 8-bit programmable pulse generation (PPG) output mode (4 channels)
  - 8-bit pulse width modulation (PWM) output mode (4 channels)
- (6) 16-bit timer : 2 channels
  - 16-bit interval timer mode
  - 16-bit event counter mode
  - 16-bit programmable pulse generation (PPG) output mode
  - Frequency measurement mode
  - Pulse width measurement mode
  - Time differential measurement mode
- (7) Serial interface (SIO) : 2 channels
  - I/O interface mode
  - Universal asynchronous receiver transmitter (UART) mode
- (8) Serial expansion interface (SEI) : 1 channel
  - Baud rate 4/2/0.5Mbps at  $f_c=20\text{MHz}$ .
- (9) Serial bus interface (SBI) : 3 channels
  - Clocked-synchronous 8-bit serial interface mode
  - I<sup>2</sup>C bus mode
- (10) CAN controller : 1channel
  - Supports CAN version 2.0B.
  - 16 mailboxes
- (11) 10-bit A/D converter (ADC) : 12 channels
  - A/D conversion time 8 $\mu\text{sec}$  @ $f_c=20\text{MHz}$ .
  - Total tolerance  $\pm 3\text{LSB}$  (excluding quantization error)
  - Scan mode for all 12channels
- (12) Watch dog timer (WDT)
- (13) Timer for real-time clock (RTC)
  - Can operate with only low frequency oscillator.
- (14) Interrupt controller (INTC) : 60 interrupt sources
  - 9 interrupts from CPU
  - 42 internal interrupt vectors
  - 9 external interrupt vectors
- (15) I/O Port : 68pins
- (16) Standby mode
  - Four modes : IDLE3,IDLE2,IDLE1 and STOP
  - STOP mode can be released by 9 external inputs.
- (17) Internal voltage detection flag (RAMSTB)

- (18) Power supply voltage
  - VCC5 = 4.5V to 5.25V
  - VCC3 = 3.3V (VCC3 Connect to REGOUT; built-in voltage regulator.)
- (19) Operating temperature : -40 to 85 degree C
- (20) Package : P-LQFP100-1414-0.50C

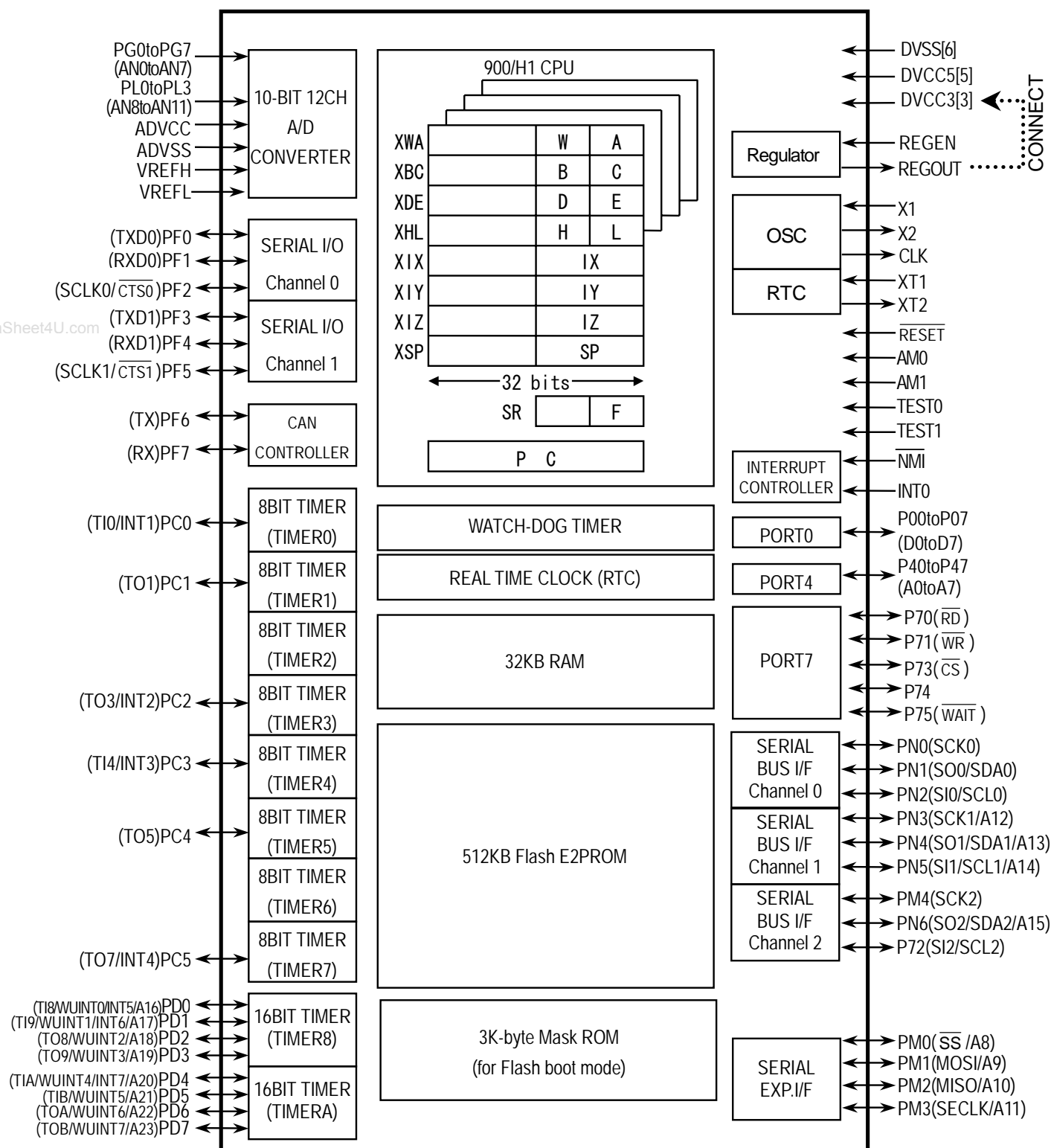
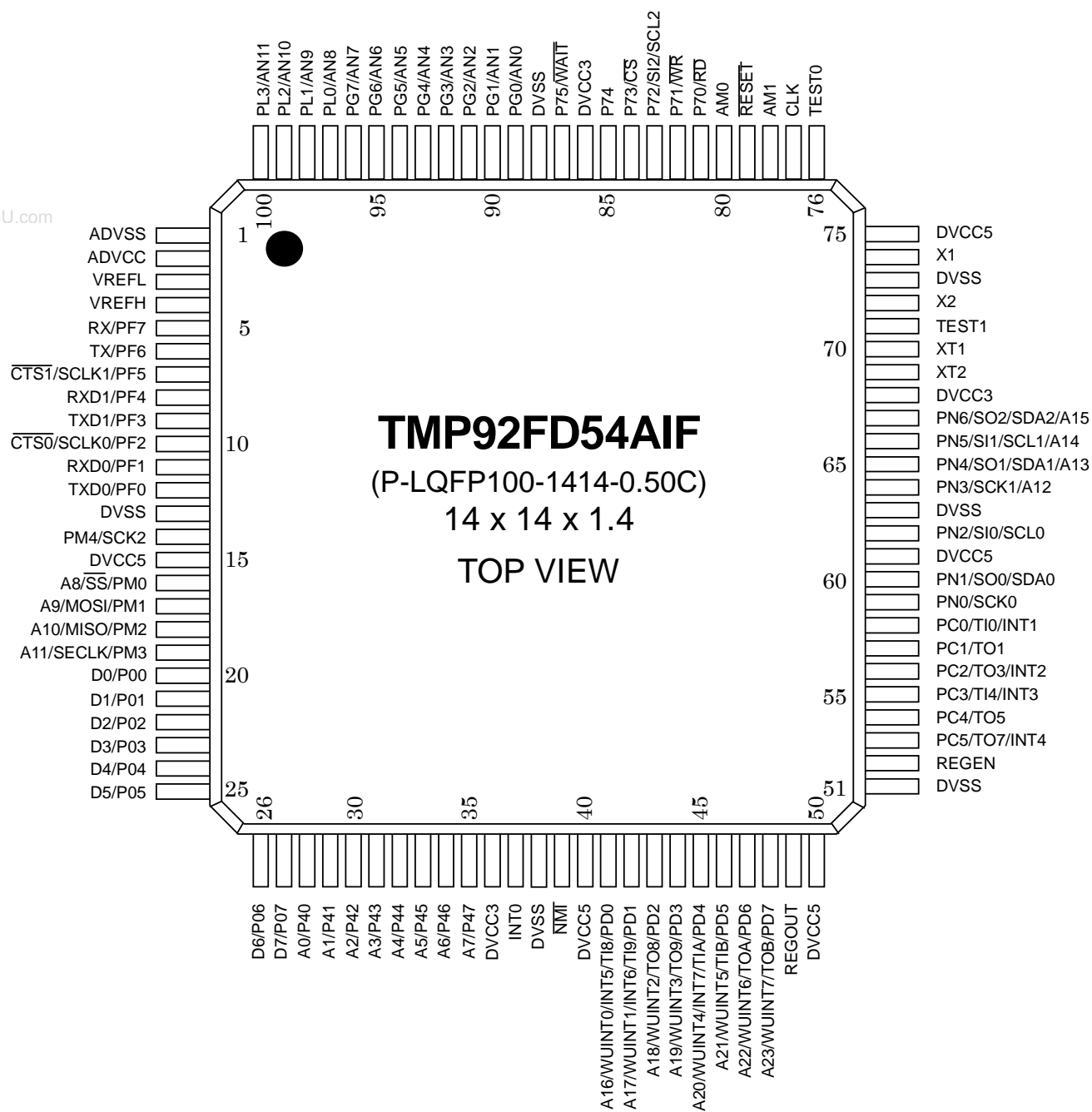


Figure 1 TMP92FD54AI block diagram

## 2. Pin Assignment and Functions

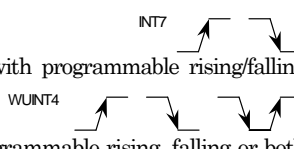
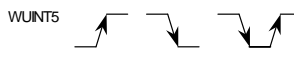
### 2.1 Pin Assignment

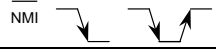


## 2.2 Pin names and functions

The following table shows the names and functions of the input/output pins.

Pin name	Pin number	Number of pins	In/Out	Function
P00..P07 D0..D7	20 <sup>th</sup> ...27 <sup>th</sup>	8 (CMOS) (TTL)	in/out in/out	Port 0: I/O port. Input or output specifiable in units of bits. Data: Data bus 0 to 7.
P40..P47 A0..A7	28 <sup>th</sup> ...35 <sup>th</sup>	8	in/out out	Port4: I/O port. Input or output specifiable in units of bits. Address: Address bus 0 to 7.
P70 $\overline{\text{RD}}$	81 <sup>st</sup>	1	in/out out	Port70: I/O port. Read: Outputs strobe signal to read external memory.
P71 $\overline{\text{WR}}$	82 <sup>nd</sup>	1	in/out out	Port 71: I/O port. Write: Output strobe signal to write external memory.
P72 SI2 SCL2	83 <sup>rd</sup>	1	in/out	Port 72: I/O port. SBI channel 2: Input data at SIO mode SBI channel 2: Clock input/output at I <sup>2</sup> C mode
P73 $\overline{\text{CS}}$	84 <sup>th</sup>	1	in/out out	Port 73: I/O port. Chip select: Outputs "low" if address is within specified address area.
P74	85 <sup>th</sup>	1	in/out	Port 74: I/O port.
P75 $\overline{\text{WAIT}}$	87 <sup>th</sup>	1	in/out in	Port 75: I/O port. Wait: Signal used to request CPU bus wait.
PC0 TI0 INT1	58 <sup>th</sup>	1	in/out in in	Port C0: I/O port. Timer input 0: Input pin for timer 0. Interrupt request pin 1: Rising-edge interrupt request pin. 
PC1 TO1	57 <sup>th</sup>	1	in/out out	Port C1: I/O port. Timer output 1: Output pin for timer 1.
PC2 TO3 INT2	56 <sup>th</sup>	1	in/out out in	Port C2: I/O port. Timer output 3: Output pin for timer 3. Interrupt request pin 2: Rising-edge interrupt request pin. 
PC3 TI4 INT3	55 <sup>th</sup>	1	in/out in in	Port C3: I/O port. Timer input 4: Input pin for timer 4. Interrupt request pin 3: Rising-edge interrupt request pin. 
PC4 TO5	54 <sup>th</sup>	1	in/out out	Port C4: I/O port. Timer output 5: Output pin for timer 5.
PC5 TO7 INT4	53 <sup>rd</sup>	1	in/out out in	Port C5: I/O port. Timer output 7: Output pin for timer 7. Interrupt request pin 4: Rising-edge interrupt request pin. 
PD0 TI8 INT5 A16 WUINT0	41 <sup>st</sup>	1	in/out in in out in	Port D0: I/O port. Timer input 8: Input pin for timer 8. Interrupt request pin 5: Interrupt request pin with programmable rising/falling edge.  Address: Address bus 16. Wake up input 0: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD1 TI9 INT6 A17 WUINT1	42 <sup>nd</sup>	1	in/out in in out in	Port D1: I/O port. Timer input 9: Input pin for timer 9.  Interrupt request pin 6: Rising-edge interrupt request pin.  Address: Address bus 17. Wake up input 1: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD2 TO8 A18 WUINT2	43 <sup>rd</sup>	1	in/out out out in	Port D2: I/O port. Timer output 8: Output pin for timer 8  Address: Address bus 18. Wake up input 2: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD3 TO9 A19 WUINT3	44 <sup>th</sup>	1	in/out out out in	Port D3: I/O port. Timer output 9: Output pin for timer 9  Address: Address bus 19. Wake up input 3: Wake up request pin with programmable rising, falling or both falling and rising edge. 

Pin name	Pin number	Number of pins	In/Out	Function
PD4 TIA INT7  A20 WUINT4	45 <sup>th</sup>	1	in/out in in  out in	Port D4: I/O port. Timer input A: Input pin for timer A. Interrupt request pin 7: Interrupt request pin with programmable rising/falling edge. Address: Address bus 20. Wake up input 4: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD5 TIB A21 WUINT5	46 <sup>th</sup>	1	in/out in out in	Port D5: I/O port. Timer input B: Input pin for timer B. Address: Address bus 21. Wake up input 5: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD6 TOA A22 WUINT6	47 <sup>th</sup>	1	in/out out out in	Port D6: I/O port. Timer output A: Output pin for timer A. Address: Address bus 22. Wake up input 6: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PD7 TOB A23 WUINT7	48 <sup>th</sup>	1	in/out out out in	Port D7: I/O port. Timer output B: Output pin for timer B. Address: Address bus 23. Wake up input 7: Wake up request pin with programmable rising, falling or both falling and rising edge. 
PF0 TXD0	12 <sup>th</sup>	1	in/out out	Port F0: I/O port. Serial interface channel 0: Transmission data.
PF1 RXD0	11 <sup>th</sup>	1	in/out in	Port F1: I/O port. Serial interface channel 0: Receive data.
PF2 SCLK0 CTS0	10 <sup>th</sup>	1	in/out in/out in	Port F2: I/O port. Serial interface channel 0: Clock input/output. Serial interface channel 0: Data ready to send. (Clear-to-send)
PF3 TXD1	9 <sup>th</sup>	1	in/out out	Port F3: I/O port. Serial interface channel 1: Transmission data.
PF4 RXD1	8 <sup>th</sup>	1	in/out in	Port F4: I/O port. Serial interface channel 1: Receive data.
PF5 SCLK1 CTS1	7 <sup>th</sup>	1	in/out in/out in	Port F5: I/O port. Serial interface channel 1: Clock input/output. Serial interface channel 1: Data ready to send. (Clear-to-send)
PF6 TX	6 <sup>th</sup>	1	in/out out	Port F6: I/O port. CAN: Transmission data.
PF7 RX	5 <sup>th</sup>	1	in/out in	Port F7: I/O port. CAN: Receive data.
PG0..PG7 AN0..AN7	89 <sup>th</sup> ...96 <sup>th</sup>	8	in in	Port G: Input-only port. Analog input 0 to 7: AD converter input pins.
PL0..PL3 AN8..AN11	97 <sup>th</sup> ...100 <sup>th</sup>	4	in in	Port L0 to L3: Input-only port. Analog input 8 to 11: AD converter input pins.
PM0 SS A8	16 <sup>th</sup>	1	in/out in out	Port M0: I/O port. SEI: Slave select input. Address: Address bus 8.
PM1 MOSI A9	17 <sup>th</sup>	1	in/out in/out out	Port M1: I/O port. SEI: Master output, slave input. Address: Address bus 9.
PM2 MISO A10	18 <sup>th</sup>	1	in/out in/out out	Port M2: I/O port. SEI: Master input, slave output. Address: Address bus 10.
PM3 SECLK A11	19 <sup>th</sup>	1	in/out in/out out	Port M3: I/O port. SEI: Clock input/output. Address: Address bus 11.
PM4 SCK2	14 <sup>th</sup>	1	in/out in/out	Port M4: I/O port. SBI channel 2: Clock input/output at SIO mode.
PN0 SCK0	59 <sup>th</sup>	1	in/out in/out	Port N0: I/O port. SBI channel 0: Clock input/output at SIO mode.

Pin name	Pin number	Number of pins	In/Out	Function
PN1 SO0 SDA0	60 <sup>th</sup>	1	in/out out in/out	Port N1: I/O port. SBI channel 0: Output data input/output at SIO mode SBI channel 0: Data input/output at I <sup>2</sup> C mode
PN2 SI0 SCL0	62 <sup>nd</sup>	1	in/out in in/out	Port N2: I/O port. SBI channel 0: Input data at SIO mode SBI channel 0: Clock input/output at I <sup>2</sup> C mode
PN3 SCK1 A12	64 <sup>th</sup>	1	in/out in/out out	Port N3: I/O port. SBI channel 1: Clock input/output at SIO mode Address: Address bus 12.
PN4 SO1 SDA1 A13	65 <sup>th</sup>	1	in/out out in/out out	Port N4: I/O port. SBI channel 1: Output data at SIO mode SBI channel 1: Data input/output at I <sup>2</sup> C mode Address: Address bus 13.
PN5 SI1 SCL1 A14	66 <sup>th</sup>	1	in/out in in/out out	Port N5: I/O port. SBI channel 1: Input data at SIO mode SBI channel 1: Clock input/output at I <sup>2</sup> C mode Address: Address bus 14
PN6 SO2 SDA2 A15	67 <sup>th</sup>	1	in/out out	Port N6: I/O port. SBI channel 2: Output data at SIO mode SBI channel 2: data input output at I <sup>2</sup> C mode Address: Address bus 15.
$\overline{\text{NMI}}$	39 <sup>th</sup>	1	in	Non-maskable interrupt: Interrupt request pin with programmable falling or both falling and rising edge. 
INT0	37 <sup>th</sup>	1	in	Interrupt request pin 0: Interrupt request pin with programmable level or rising-edge. 
AM0,1	80 <sup>th</sup> , 78 <sup>th</sup>	2	in	Address mode pins: These pins are set as following, (Single-Chip mode) AM0 = L, AM1 = H (Single-Boot mode) AM0 = H, AM1 = H
TEST0,1	76 <sup>th</sup> , 71 <sup>st</sup>	2	in	Test mode pins: These pins are set as following, (Single Chip & Single Boot mode) TEST0 = L, TEST1 = L
CLK	77 <sup>th</sup>	1	out	Programmable clock output (with pull-up register).
X1/X2	74 <sup>th</sup> , 72 <sup>nd</sup>	2	in/out	Oscillator connecting pins
XT1/XT2	70 <sup>th</sup> , 69 <sup>th</sup>	2	in/out	Low frequency oscillator connecting pins. Crystal or ceramic resonator is connected. RC oscillation is also possible depending on MASK option.
$\overline{\text{RESET}}$	79 <sup>th</sup>	1	in	Reset: Initializes LSI (with pull-up register).
VREFH	4 <sup>th</sup>	1	in	AD reference voltage high
VREFL	3 <sup>rd</sup>	1	in	AD reference voltage low
ADVCC	2 <sup>nd</sup>	1	-	Power supply pin for AD converter (+5V): Connect ADVCC pin to 5V power supply.
ADVSS	1 <sup>st</sup>	1	-	GND pin for AD converter: Connect ADVSS pin to GND (0V).
DVCC5	15 <sup>th</sup> , 40 <sup>th</sup> , 50 <sup>th</sup> , 61 <sup>st</sup> , 75 <sup>th</sup>	5	-	Power supply pins (+5V): Connect all DVCC5 pins to 5V power supply.
DVCC3	36 <sup>th</sup> , 68 <sup>th</sup> , 86 <sup>th</sup>	3	-	Power supply pins (+3.3V): Connect all DVCC3 pins to REGOUT pin.
DVSS	13 <sup>th</sup> , 38 <sup>th</sup> , 51 <sup>st</sup> , 63 <sup>rd</sup> , 73 <sup>rd</sup> , 88 <sup>th</sup>	6	-	GND: Connect all DVSS pins to GND (0V).
REGOUT	49 <sup>th</sup>	1	out	Regulator output 3.3V: Connect capacitor to stabilize the regulator output.
REGEN	52 <sup>nd</sup>	1	in	Regulator enable pin: Should be set to H or OPEN (with pull-up register).

### 3. OPERATION

This section describes the basic components, functions and operation of TMP92FD54AI.

#### 3.1 CPU

TMP92FD54AI contains an advanced high-speed 32-bit CPU (900/H1 CPU)

##### 3.1.1 CPU Outline

900/H1 CPU is high-speed and high-performance CPU based on 900/H CPU. 900/H1 CPU has expanded 32-bit internal data bus to process Instructions more quickly.

Outline of 900/H1 CPU are as follows:

	900/H1 CPU	
Width of CPU Address Bus	24-bit	
Width of CPU Data Bus	32-bit	
Internal Operating Frequency	16 to 20MHz (@fosc=8 to 10MHz)	
Minimum Bus Cycle (Internal RAM)	1-clock access (50ns@fosc=10MHz)	
Internal RAM	32-bit 1-clock access	
Internal ROM	32-bit interleave 2-1-1-1-clock access	
Internal I/O	8/16-bit 2-clock access	PORT, INTC, MEMC
	8/16-bit 5 to 6-clock access	SEI, SIO, WDT, 8-bit Timer, 16-bit Timer, RTC, 10-bit ADC, SBI, CAN
External Device	8-bit 2-clock access (can insert some waits)	
Minimum Instruction Execution Cycle	1-clock(50ns@fosc=10MHz)	
Conditional Jump	2-clock(100ns@fosc=10MHz)	
Instruction Queue Buffer	12-byte	
Instruction Set	Compatible with TLCS-900, 900/H, 900/L, 900/L1 and 900/H2 (NORMAL, MIN, MAX and LDX instruction is deleted)	
Micro DMA	8-channels	

## 4. Functional Description

This section describes the hardware configuration of the TMP92FD54AI and how it operates. This device is a modified version of the TMP92CD54I with an internal 512K-byte flash memory instead of an internal 512K-byte Mask ROM. These internal RAM are same size, 32K-byte. In all other respects the hardware configuration and the functionality of the TMP92FD54AI are identical to those of the TMP92CD54I.

### 4.1 Flash Memory

#### 4.1.1 Features

##### 1) Memory capacity

TMP92FD54AI contains a 4-Mbit (512K-byte) flash memory. This flash memory consists of 10 blocks (64 K-bytes  $\times$  6 blocks, 56 K-bytes  $\times$  2 blocks, 8 K-bytes  $\times$  2 blocks), each of which can be independently protected from being programmed or erased. The CPU accesses the flash memory using a 32-bit wide data bus. A writer rewrites the flash memory in units of 16 bits.

##### 2) Flash memory access

Interleaved access (2-1-1-1-clock access)

##### 3) Program/erase time

Programming time (including Verify): 6 seconds per chip (typ.)

30  $\mu$ s per long word (typ.)

Erase time (including Verify) 12 seconds per chip (typ.)

Note: The above values are typical times and do not include data transfer time.

The actual time per chip varies according to the method used by the user to perform rewriting.

##### 4) Programming methods

On-board programming mode, in which the memory can be rewritten while mounted on the user PCB.

- On-board programming mode
  - 1) User Boot Mode  
Supports user-defined rewriting methods.
  - 2) Single Boot Mode  
Supports rewriting by serial transfer (unique to Toshiba).

##### 5) Rewriting method

With a few exceptions, the functions of the device's internal flash memory conform to JEDEC standards. Therefore, even if the user system uses flash memory as external memory, it can easily be adapted to use this LSI. Furthermore, since the device's flash memory has built-in circuits which can automatically write to the flash memory and erase the chip, the user does not have to code a program to perform the program and erase operations.

The Block Protect function, which inhibits writing to or erasing the flash memory, only supports software protection (i.e. protection by a command setting), and does not support hardware protection (i.e. protection by the application of a voltage of 12 V).

JEDEC-based functions	Functions which have been changed, added or removed
Auto Program Auto Chip Erase Auto Block Erase Data Polling / Toggle Bit	<Changed> Block Protect (supports only software protection.) <Removed> Erase Resume/Suspend function

#### 4.1.2 Block diagram of the flash unit

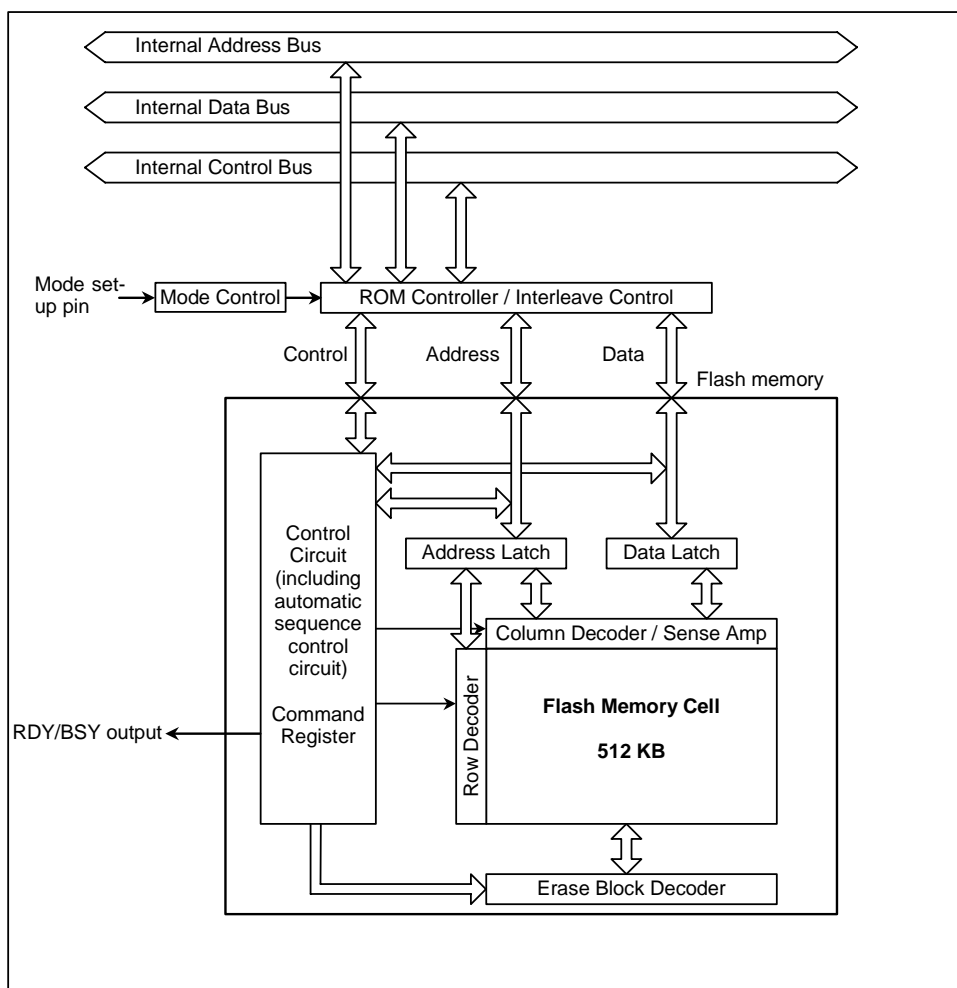


Figure 4.1.1 Block diagram of the flash unit

## 4.2 Operation Modes

TMP92FD54AI has three operation modes.

Table 4.2.1 Operation modes

Operation Mode Name	Description
<u>Single-Chip Mode</u>	After a reset the device restarts from the internal flash memory. Single-Chip Mode is further sub-divided into the following two modes: Normal Mode and User Boot Mode. The means for selecting between these two modes can be set by the user as desired. For example, it can be set so that if Port 00 = 1, Normal Mode is selected, and if Port 00 = 0, User Boot Mode is selected. The user application program must incorporate a routine to handle mode switching.
Normal Mode	The user's application program is executed.
User Boot Mode	The flash memory on the user PCB board is rewritten.
<u>Single Boot Mode</u>	After a Reset the device starts up from the internal boot ROM (the mask ROM). The boot ROM has an algorithm which allows memory on the user PCB to be rewritten via the device's serial port. The internal flash memory can be rewritten by connecting the PCB to the external host via TMP92FD54AI's serial port and transferring data using TMP92FD54AI data transfer protocol.


Of the modes listed in the table above, there are two operation modes in which the flash memory can be programmed: User Boot Mode and Single Boot Mode.

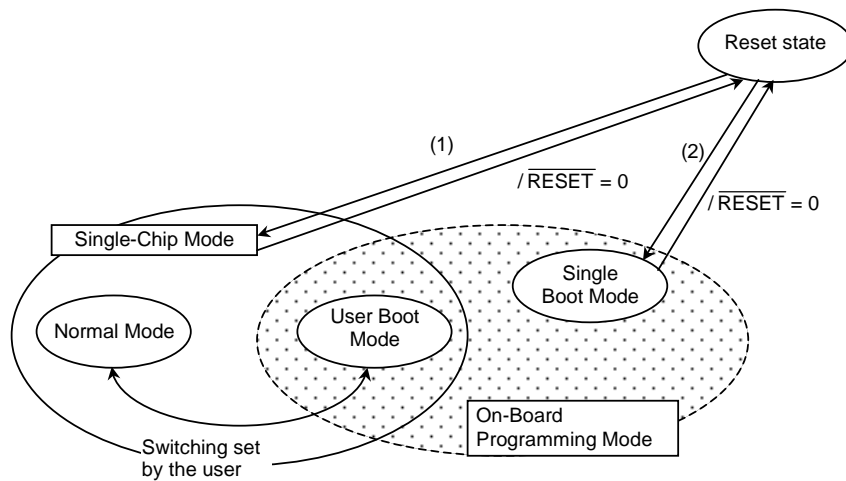
At User Boot Mode (in Single-Chip mode) and Single Boot Mode, the internal flash memory can be rewritten while mounted on the user PCB. These two modes are collectively referred to as On-Board Programming Mode.

The input pin levels on AM0, AM1, TEST0 and TEST1, which are set externally during a Reset, determine the operation mode - either Single-Chip Mode, Single-Boot Mode.

For all operation modes, the CPU will start operating in the selected mode on completion of the reset. The input pin levels are set before the reset is completed. Once the mode has been set, make sure that the input pin levels do not change during operation. The following table shows how each operation mode is set. The state diagram beneath the table shows the various possible mode transitions.

Table 4.2.2 Operation mode pin settings

	Operation Mode	Input Pins				
		$\overline{\text{RESET}}$	AM1	AM0	TEST1	TEST0
(1)	Single-Chip Mode (Normal & User Boot)		1	0	0	0
(2)	Single Boot Mode		1	1	0	0



Numbers in ( ) indicate the column row in the above table in which the input pin levels for the mode setting are shown.

Figure 4.2.1 Mode transition diagram

#### 4.2.1 Reset operation

To reset the device, hold the  $\overline{\text{RESET}}$  input Low (= 0) for at least 20 system clock periods while the power supply voltage is within the rated operating range and the internal high-frequency oscillator is oscillating stably. When the device is operating at 20 MHz, this period will be equal to 4  $\mu\text{s}$ . For details please refer to the section entitled Reset Operation in TMP92CD54I User's Manual.

#### 4.2.2 Memory map for each mode

The memory map for the device varies according to the operation mode. The following diagram shows the memory map for each operation mode.

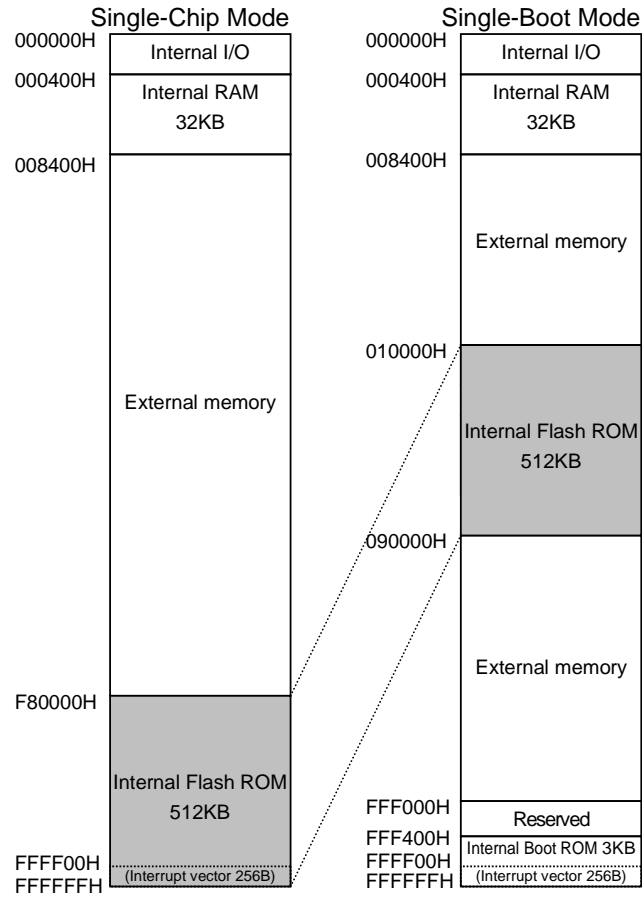


Figure 4.2.2 TMP92FD54AI memory map for each mode

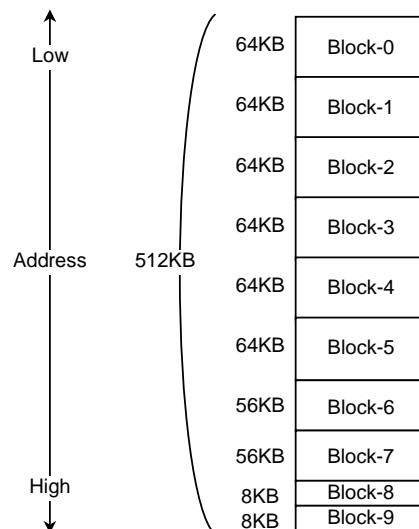


Figure 4.2.3 Block allocation

Table 4.2.3 Block address range by mode

	Single-Chip Mode	Single-Boot Mode
Block-0 (64 KB)	F80000H to F8FFFFH	010000H to 01FFFFH
Block-1 (64 KB)	F90000H to F9FFFFH	020000H to 02FFFFH
Block-2 (64 KB)	FA0000H to FAFFFFH	030000H to 03FFFFH
Block-3 (64 KB)	FB0000H to FBFFFFH	040000H to 04FFFFH
Block-4 (64 KB)	FC0000H to FCFFFFH	050000H to 05FFFFH
Block-5 (64 KB)	FD0000H to FDFFFFH	060000H to 06FFFFH
Block-6 (56 KB)	FE0000H to FEDFFFFH	070000H to 07DFFFFH
Block-7 (56 KB)	FEE000H to FFBFFFFH	07E000H to 08BFFFFH
Block-8 ( 8 KB)	FFC000H to FFDFFFFH	08C000H to 08DFFFFH
Block-9 ( 8 KB)	FFE000H to FFFFFFFH	08E000H to 08FFFFFFH

### 4.3 On-Board Programming Mode (User-Boot and Single-Boot)

On-board Programming Mode allows flash memory to be rewritten while it is mounted on the user's target system board. There are two versions of this mode: Single Boot Mode, which supports the Toshiba proprietary method of rewriting via serial I/O, and User Boot Mode (in Single-Chip mode), which allows user-defined rewriting methods to be defined in Single-Chip Mode.

#### 4.3.1 User Boot Mode (in Single-Chip Mode)

User Boot Mode enables a user-defined flash memory rewrite routine to be used. This mode is used when the data transfer bus which the user application flash memory rewrite program uses is not serial. Rewriting is performed in Single-Chip Mode. For this reason the operation mode within Single-Chip Mode must be changed, from Normal Mode (the mode in which the user application program normally operates) to User Boot Mode, in which mode the flash memory can be rewritten. This requires that a condition-judging program be incorporated into the user application reset-processing program.

Any mode-switching condition may be set using the device's I/Os as long as it is suitable for the user system. Also, the user's exclusive flash memory rewrite routine, for use in User Boot Mode, may be programmed into the user application in advance, so that it can be used to rewrite the flash memory after the device has entered User Boot Mode. Furthermore, since the processor cannot read data from the internal flash memory during an erase/program operation, the rewrite routine must be stored somewhere outside the flash memory so that it can be executed. While the on-chip flash memory is being rewritten in User Boot mode, all interrupts including the non-maskable interrupt must be disabled.

The pages which follow explain the procedure for rewriting the flash memory using two example case studies. In one case the rewrite routine is put into the internal flash memory (1-A); in the other the rewrite routine is transferred from an external source (1-B). For details of how to program/erase the flash memory, refer to Section 4.4, Programming/Erasing Flash Memory during on-Board Programming.

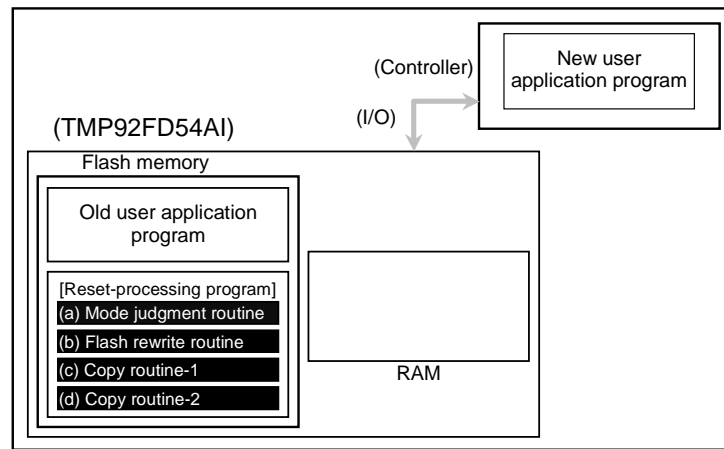
**User Boot Mode**

(1-A) Example procedure in which the rewrite routine is stored in the internal flash memory

**(Step 1)**

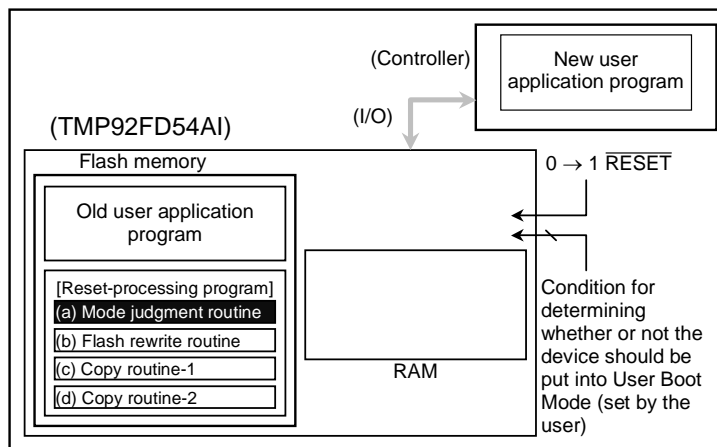
First, the conditions (e.g. the pin status) under which the program will enter User Boot Mode must be set and the I/O bus which the program will use to transfer data must be selected. Then a circuit must be designed and a corresponding program written. Before mounting this device on the board, the following four programs must be written into one of the blocks of the flash memory using a programmer or similar device.

- (a) Mode judgment routine: code to determine when rewriting of the device will commence
- (b) Flash rewrite routine: code to read the rewrite data from an external source and then rewrite the flash memory
- (c) Copy routine-1: code to copy (a) to (d) into the internal RAM or to external memory
- (d) Copy routine-2: code to copy (a) to (d) to the flash memory from the internal RAM or the external memory.

**(Step 2)**

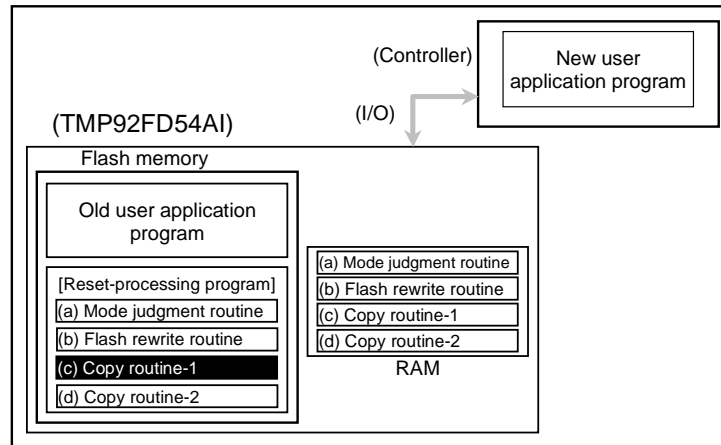
The following explanation of the proceeding step assumes that the routines described above have been included in the Reset-processing program.

After a Reset the Reset-processing program must first determine whether or not the device should enter User Boot Mode. If the condition for entering User Boot Mode is true, the program must put the device into User Boot Mode, the mode in which the flash memory can be rewritten. (Once the device has entered User Boot Mode, no interrupts — not even NMIs — should be used.)

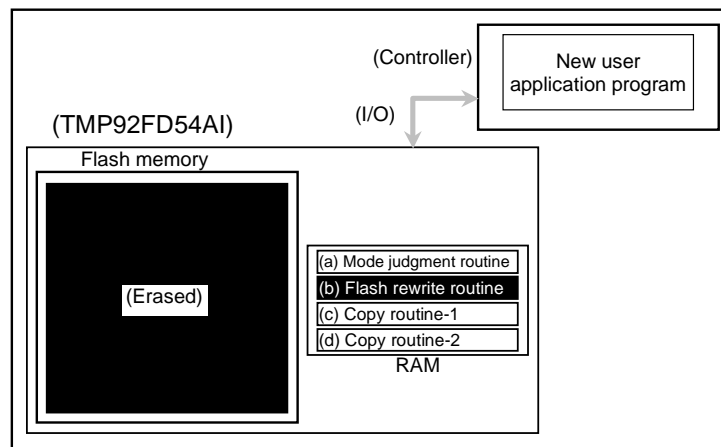


*(Step 3)*

After the device has entered User Boot Mode, (c), the copy routine, must copy (a) to (d), the flash rewrite routine, into the internal RAM or into external memory. (In the example shown below the routine is copied into the internal RAM.)

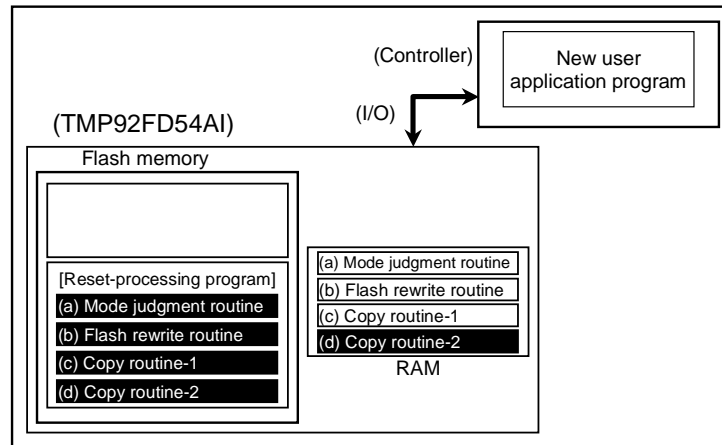
*(Step 4)*

The program jumps to the rewrite routine in RAM, erases all blocks (chip-erase).

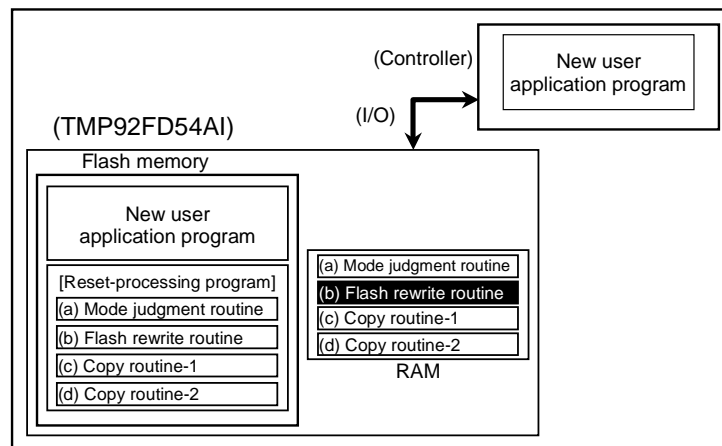


**(Step 5)**

The copy routine-2 in the RAM is executed and writes (a) to (d) into the flash memory.

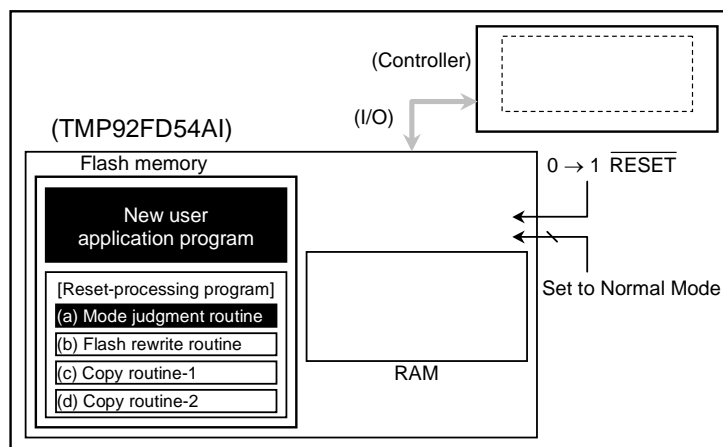
**(Step 6)**

Next, the rewrite routine in the RAM is executed and writes the new user application program into the erased area of the flash memory after loading it from the source of transfer (the controller).



*(Step 7)*

The  $\overline{\text{RESET}}$  input pin must be driven Low (= 0) to reset the device and to set it to Normal Mode. After de-assertion of the reset, the processor will start running the new user application program.



(1-B) Example procedure for when the rewrite routine is transferred from an external source.

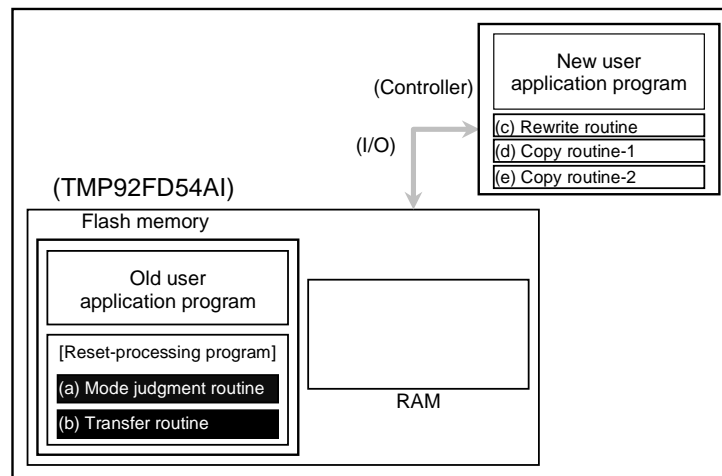
**(Step 1)**

First, the condition (e.g. the pin status) under which the program will enter User Boot Mode must be set and the I/O bus which the program will use to transfer data must be selected. Then a circuit must be designed and a corresponding program written. Before mounting this device on the board, the following two programs must be written into one of the blocks of the flash memory using a programmer or similar device.

- (a) Mode judgment routine: Code to determine when rewriting of the device will commence
- (b) Transfer routine: Program for reading the rewrite program from an external source

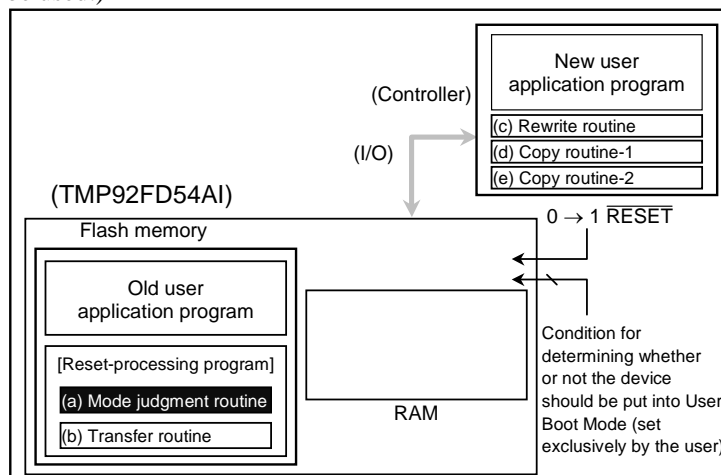
In addition, the programs shown below should be present in the host:

- (c) Rewrite routine: Program for rewriting the flash memory
- (d) Copy routine-1: Code to copy (a) to (d) into the internal RAM or to the external ROM.
- (e) Copy routine-2: Code to copy (a) to (d) into the flash memory from the internal RAM or the external ROM



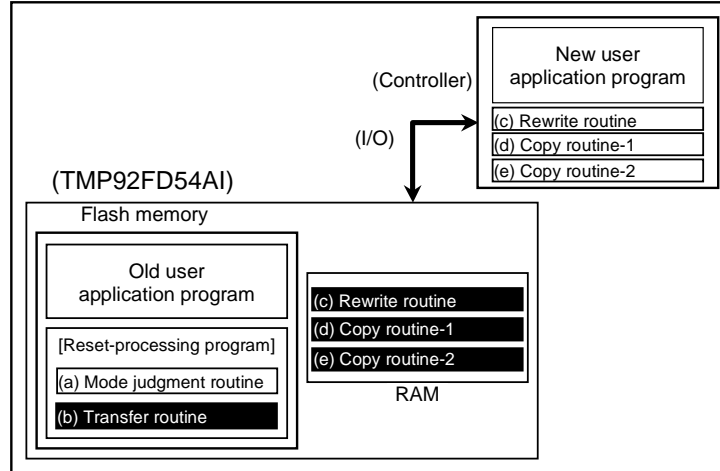
**(Step 2)**

The following explanation of the proceeding step assumes that the routines described above have been included in the Reset processing program. After reset the Reset-processing program must first determine whether or not the device should enter User Boot Mode. If the condition for entering User Boot Mode is true, the program must put the device into User Boot Mode, the mode in which to the flash memory can be rewritten. (Once the device has entered User Boot Mode, no interrupts — not even NMIs — should be used.)

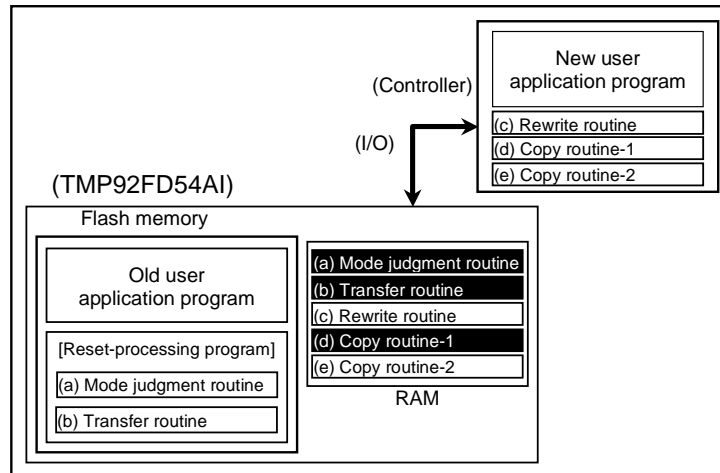


**(Step 3)**

After the device has entered User Boot Mode, (c) Rewrite routine, (d) Copy routine-1 and (e) Copy routine-2, must be loaded from the source of transfer (the controller) into the internal RAM or into external memory using (b), the transfer routine. (In the example shown below the routine is copied into the internal RAM.)

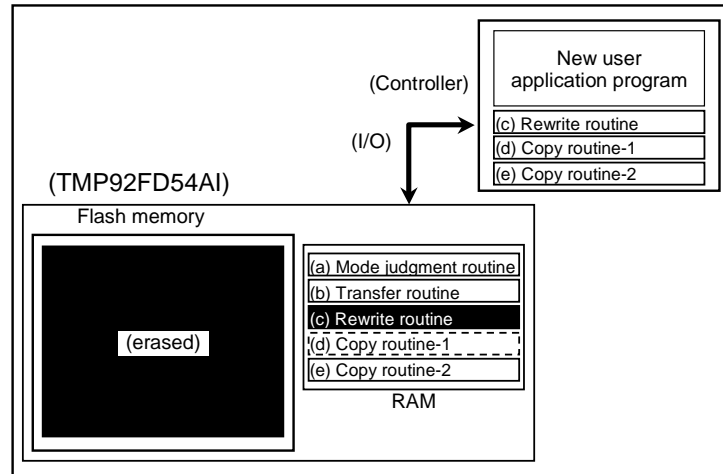
**(Step 4)**

The program jumps to the copy routine-1 in RAM, copy (a) and (b) into the internal RAM or into external memory. (In the example shown below the routine is copied into the internal RAM.)

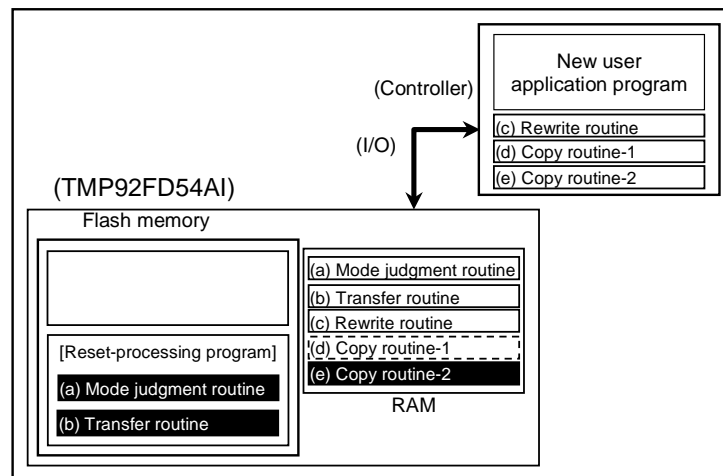


**(Step 5)**

The rewrite routine in the RAM is executed, erase all block of the flash memory (Chip-erase).

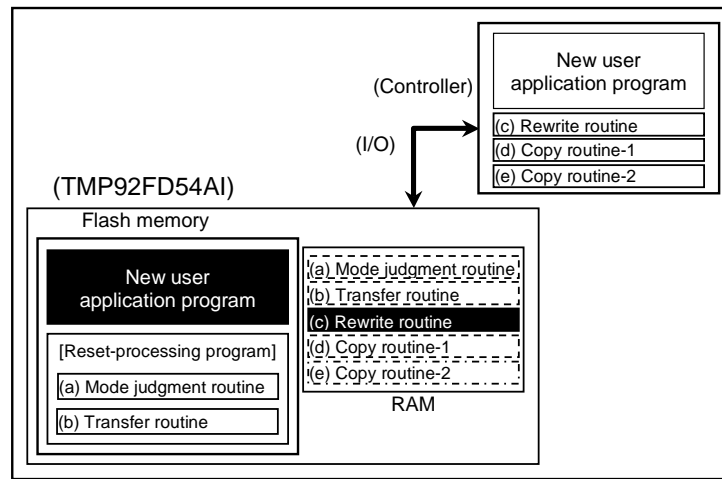
**(Step 6)**

The copy routine-2 in the RAM is executed, copy (a) and (b) into the flash memory.

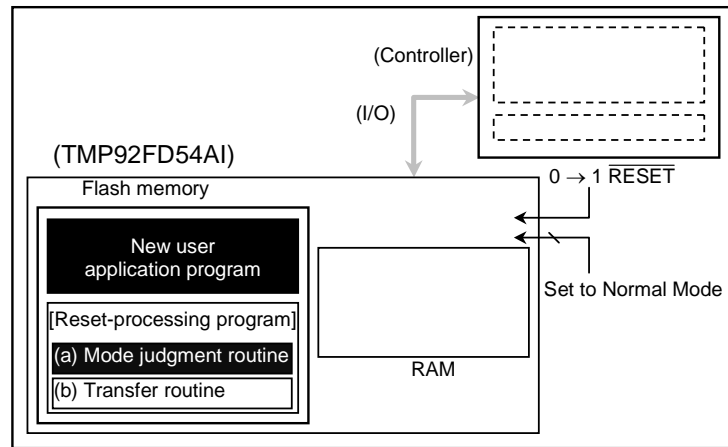


**(Step 7)**

Next, the rewrite routine in the RAM is executed and writes the new user application program into the erased area of the flash memory after loading it from the source of transfer (the controller).

**(Step 8)**

The  $\overline{\text{RESET}}$  input pin must be driven Low (= 0) to reset the device and to set it to Normal Mode. After de-assertion of the reset, the processor will start running the new user application program.



#### 4.3.2 Single-Boot Mode

This mode involves activation of the internal boot ROM (mask ROM) so that the flash memory can be rewritten using a program stored in the boot ROM. In this mode the internal boot ROM is mapped into an area containing the interrupt vector table, in which the boot ROM program is executed. The flash memory is mapped into an address space different from the one into which the boot ROM is mapped (see Figure 4.2.2).

In Single-Boot Mode the flash memory is rewritten by serial transfer of commands and data.

First, the device's SIO (SIO1) and the external host are connected and the rewrite program is copied from the external host into the device's internal RAM; the rewrite program is then executed in RAM and the flash memory rewritten. The rewrite routine is initiated by sending commands and rewrite data from the host. The device must communicate with the host following the protocol described later. Before the program can be transferred into RAM, the user password is checked to ensure the security of the user ROM data. If the password does not match, the program is not transferred into RAM. Interrupts must be disabled when peripheral functions such as the SIO is utilized in Single Boot mode. Nonetheless, the code for an interrupt cause is recorded in the INTES1 register ; thus, for example, data receptions or transmissions in the SIO can be confirmed by keeping track of value changes of the INTES1 register. The NMI interrupt must be disabled at this time.

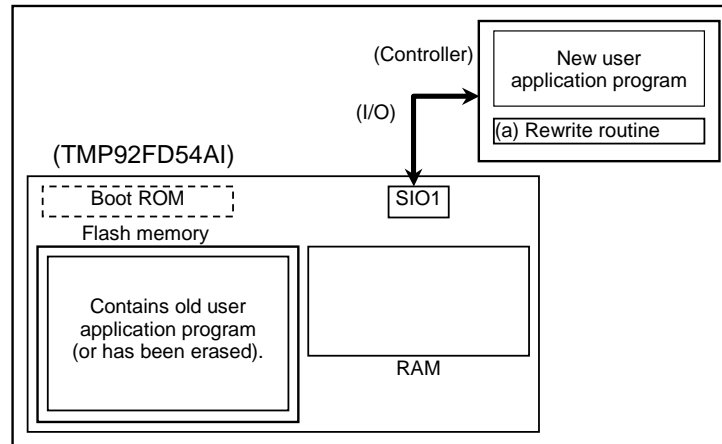
To prevent the contents of flash memory inadvertently rewritten in Single-Chip Mode (Normal Mode), it is recommended that, on completion of rewriting, write-protect be set for any blocks which the user wishes to protect. For details of how to program/erase the flash memory, please refer to Section 4.4, Programming/Erasing Flash Memory during On-Board Programming.

**Single Boot Mode**

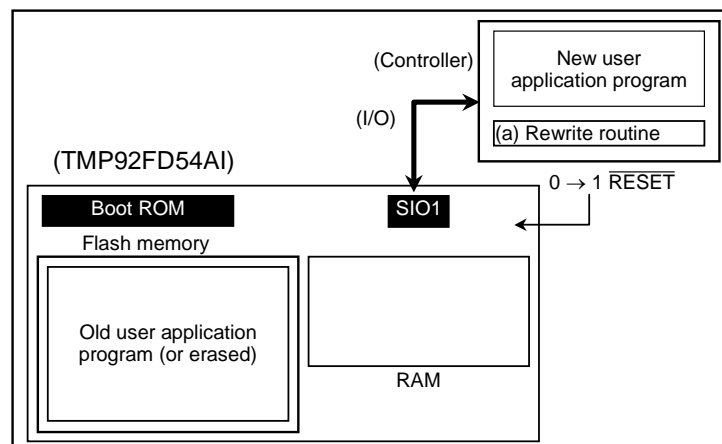
(2-A) Using the rewrite algorithm in the internal boot ROM

**(Step 1)**

The state of the flash memory is not important; it may contain the old version of the user program or it may have been erased. Since the rewriting routine and rewrite data are transferred via SIO (SIO1), the device's SIO (SIO1) and the external controller should be connected on the board. The user must ensure that (a), the rewrite routine, is present on the controller.

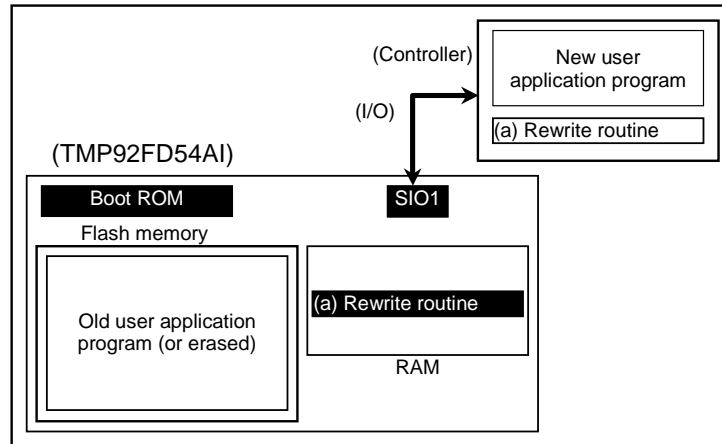
**(Step 2)**

To turn the boot ROM on, Reset must be de-asserted by setting the input pins accordingly. (a), the rewriting routine, must be transferred from the source (the controller) into the device via SIO. Before this is carried out, however, the password entered by the user must be checked against the password registered on the user application program. (When the flash memory has been erased, erase data ("0xFF" 12byte) is used as the password.)

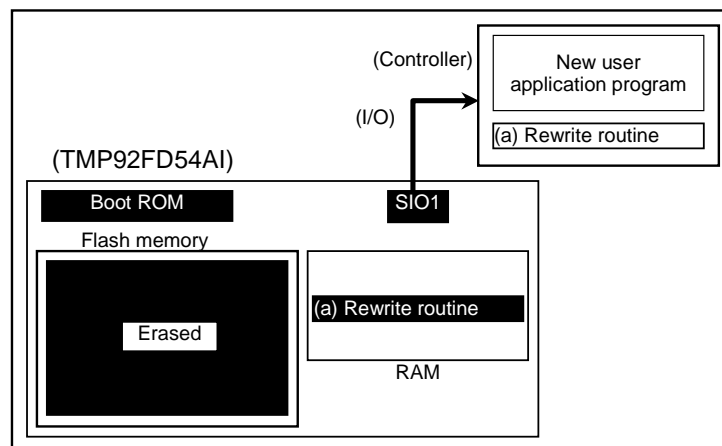


*(Step 3)*

After the password has been checked, (a), the rewrite routine, must be transferred from the source (the controller). The boot ROM then loads the routine into the internal RAM. It must be ensured, however, that the routine is stored within the RAM address area 000400H to 006BFFH.

*(Step 4)*

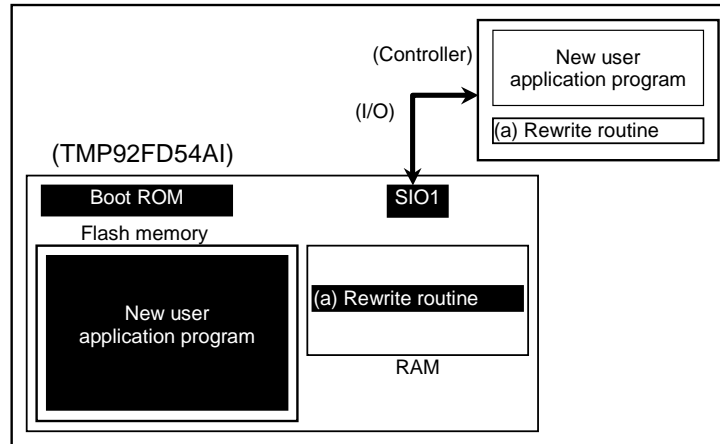
The CPU jumps to (a), the rewrite routine, in RAM and the rewrite routine erases the old user application program area (all the blocks at once).



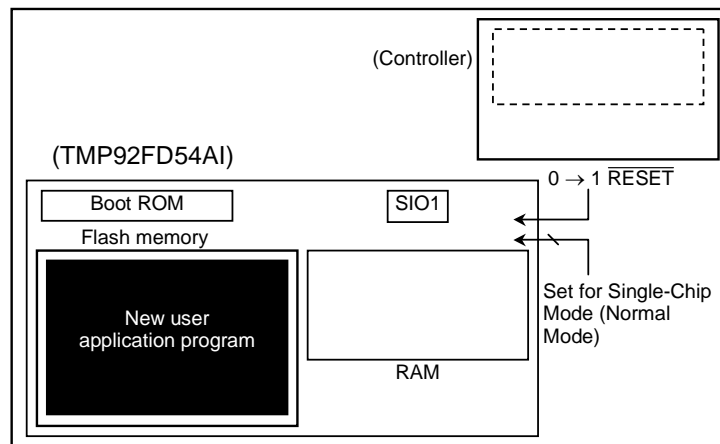
*(Step 5)*

Next, the Rewrite routine in the RAM is executed and writes the new user application program into the erased area of the flash memory after loading it from the source of transfer (the controller). When the processor has finished writing the new user application program, write-protect for the user program area should be turn on.

In the example below the rewrite data is transferred from the same host via the same SIO as the rewrite routine. However, once normal program operation has started in RAM, any other data bus and any other transfer source may be used as desired. The user must design the board hardware and write the rewrite routine accordingly.

*(Step 6)*

When writing to the device has been completed, turn the power to the board off and remove the cable connecting the device and host. To execute the new user application program, turn the power back on again and start the device in Single-Chip Mode (Normal Mode).



## (1) Typical connection scheme for Single Boot Mode

In Single Boot Mode the flash memory is rewritten by means of a serial transfer. Therefore on-board programming is carried out by connecting the device's SIO (Channel1) and the controller (the programming tool) and sending commands from the controller to the target board. Figure 4.3.1 shows an example connection scheme for a programming controller and a target board.

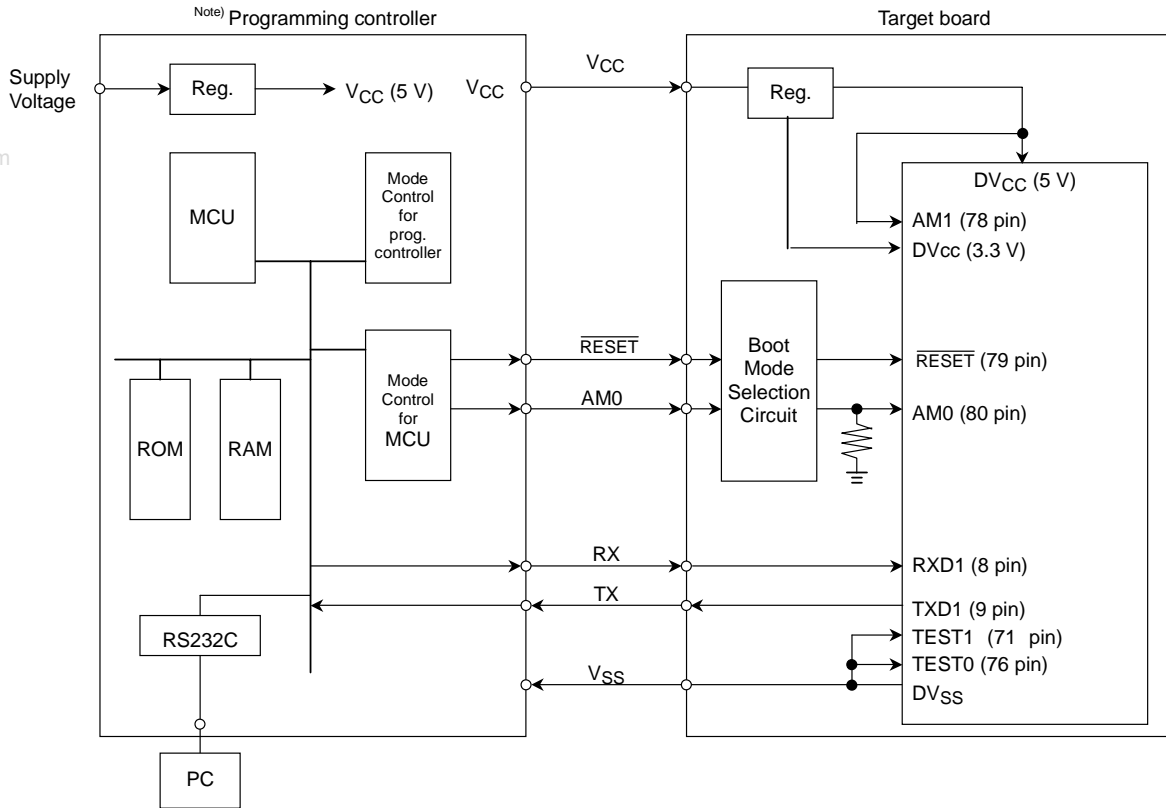


Figure 4.3.1 Example of connection board and external controller in Single Boot Mode (with communication via UART)

## (2) Mode settings

Before on-board programming can be performed, the device must be started up in Single Boot Mode. The input pins must be set as follows in order to start the device up in Single Boot Mode.

AM0	= 1
AM1	= 1
TEST0	= 0
TEST1	= 0
$\overline{\text{RESET}}$	= 0 $\rightarrow$ 1

Set the  $\overline{\text{RESET}}$  input pin Low (= 0) and set the AM0, AM1, TEST0 and TEST1 pins as shown above. Then de-assert  $\overline{\text{RESET}}$  (i.e. set it to 1). The device will start up in Single Boot Mode.

## (3) Memory map

Figure 4.3.2 compares the memory maps for Single-Chip Mode and Single-Boot Mode. As shown, in Single-Boot Mode the internal flash memory is mapped into 010000H to 08FFFFH. The boot ROM (mask ROM) is mapped into FFF400H to FFFFFFFH.

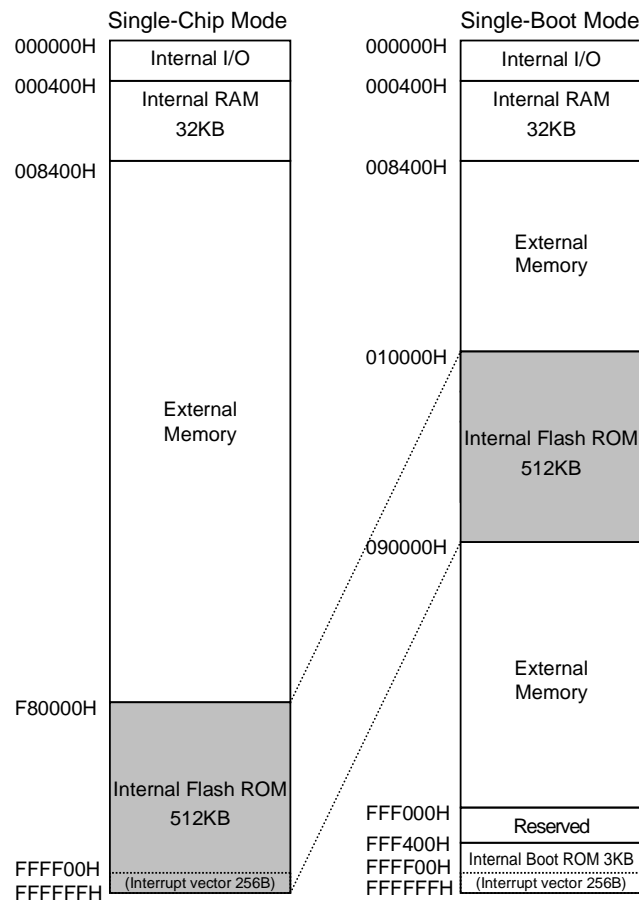


Figure 4.3.2 Memory maps for Single-Chip Mode and Single-Boot Mode

## (4) Interface specifications

SIO communication specifications for Single Boot Mode are shown below. The device supports a serial operation mode: UART (Asynchronous). Before on-board programming can be executed, the programming controller must be set to the same communications format as the device side.

- Communicating in UART Mode
  - Communications channel: SIO Channel1
  - Serial transfer mode: UART (Asynchronous) Mode, full-duplex communication
  - Data length: 8 bits
  - Parity bits: None
  - STOP bits: 1
  - Baud rate: Desired baud rate (See Table 4.3.1(a) )

Table 4.3.1(a) Baud Rate Selection Table

Baud Rate (bps)	38400	19200	9600	4800	2400
-----------------	-------	-------	------	------	------

Table 4.3.1(b) Pin connections

Pin	UART
Power supply pins	DVCC (3.3 V)
	DVSS
Mode-setting pin	AM1, AM0, TEST1, TEST0
Reset pin	RESET
Communications pins	TXD1
	RXD1

## (5) Data transfer format

The operation commands and data transfer formats for each operation mode are shown in Table 4.3.2 to Table 4.3.67. For related information please refer to subsection (6), entitled *Boot program*, on page 37 of Section 4.3.2, *Single-Boot Mode*.

Table 4.3.2 Operation command data

Operation Command Data	Operation Mode
10H	RAM transfer
20H	Flash memory SUM
30H	Read product information
40H	Auto Chip Erase & UnProtect

Table 4.3.3 Single Boot program transfer format (for RAM transfer)

	Number of bytes transferred	Data transferred from the controller to the device	Baud rate	Data transferred from the device to the controller
BOOT ROM	1st byte	Serial operation mode and baud rate settings For UART 86H	Desired baud rate (Table 4.3.1(a))	—
	2nd byte	—		ACK response for Serial Operation Mode For UART If normal (can be set) 86H (If the device determines that the baud rate cannot be set, it will stop operating.)
	3rd byte	Operation command data (10H)		—
	4th byte	—		ACK response for operation command *1 If normal 10H If abnormal × 1H If communications error occurs × 8H
	5th byte	PASSWORD data (12 bytes)		—
	~			
	16th byte	(08FEF4H to 08FEFFH)		—
	17th byte	CHECKSUM value for bytes 5 to 16		—
	18th byte	—		ACK response to CHECKSUM value *1 If normal 10H If abnormal 11H If communications error occurs 18H
	19th byte	RAM storage start address 31 to 24		—
	20th byte	RAM storage start address 23 to 16		—
	21st byte	RAM storage start address 15 to 8		—
	22nd byte	RAM storage start address 7 to 0		—
	23rd byte	RAM storage bytes 15 to 8		—
	24th byte	RAM storage bytes 7 to 0		—
	25th byte	CHECKSUM value for bytes 19 to 24		—
	26th byte	—		ACK response for CHECKSUM value *1 If normal 10H If abnormal 11H If communications error occurs 18H
	27th byte	RAM storage data		—
	~			
	mth byte			—
	(m + 1) th byte	CHECKSUM value for bytes 27 to m		ACK response for CHECKSUM value *1 If normal 10H If abnormal 11H If communications error occurs 18H
	(m + 2) th byte	—		—
RAM	(m + 3) th byte	—		JUMP RAM storage start address

\*1: After returning an abnormal response the device waits for the operation command (third byte). In I/O Interface Mode this ACK response is not generated.

\*2: Write your program so that the data in bytes 19 to 25 is stored within the RAM address area 000400H to 0083FFH.

Table 4.3.4 Boot program transfer format (for flash memory SUM)

	Number of bytes transferred	Data transferred from the controller to the device	Baud rate	Data transferred from the device to the controller
BOOT ROM	1st byte	Serial operation mode and baud rate settings For UART 86H	Desired baud rate (Table 4.3.1(a))	—
	2nd byte	—		ACK response for Serial Operation Mode For UART If normal (can be set) 86H (If the device determines that the baud rate cannot be set, it will stop operating.)
	3rd byte	Operation command data (20H)		—
	4th byte	—		ACK response for operation command *1 If normal 20H If abnormal X1H If communication error occurs X8H
	5th byte	—		SUM (upper)
	6th byte	—		SUM (lower)
	7th byte	—		CHECKSUM value for bytes 5 to 6
	8th byte	(Wait for the next operation command.)		—

\*1: After returning an abnormal response the controller waits for the operation command (third byte).

Table 4.3.5 Boot program transfer format (for reading product information) (1/2)

	Number of bytes transferred	Data transferred from the controller to the device	Baud rate	Data transferred from the device to the controller
BOOT ROM	1st byte	Serial operation mode and baud rate settings For UART 86H	Desired baud rate (Table 4.3.1(a))	—
	2nd byte	—		ACK response for Serial Operation Mode For UART If normal (can be set) 86H (If the device determines that the baud rate cannot be set, it will stop operating.)
	3rd byte	Operation command data (30H)		—
	4th byte	—		ACK response for operation command *1 If normal 30H If abnormal X1H If communication error occurs X8H
	5th byte	—		Flash memory data (address 08FEF0H)
	6th byte	—		Flash memory data (address 08FEF1H)
	7th byte	—		Flash memory data (address 08FEF2H)
	8th byte	—		Flash memory data (address 08FEF3H)
	9th byte ~ 20th byte	—		Product name (ASCII code, 12 bytes) "TMP92FD54AI " beginning at the 9th byte
	21st byte ~ 24th byte	—		Password comparison start address (4 bytes) F4H, FEH, 08H and 00H beginning at the 21st byte
	25th byte ~ 28th byte	—		RAM start address (4 bytes) 00H, 04H, 00H and 00H beginning at the 25th byte
	29th byte ~ 32nd byte	—		RAM user area end address (8 bytes) FFH, 6BH, 00H and 00H beginning at the 29th byte
	33rd byte ~ 36th byte	—		RAM end address (4 bytes) FFH, 83H, 00H and 00H beginning at the 33rd byte
	37th byte ~ 44th byte	—		Dummy data (8 bytes) 00H, 00H, 00H and 00H 00H, 00H, 00H and 00H
	45th byte ~ 46th byte	—		Dummy data (2 bytes) 00H and 03H beginning at the 45th byte
	47th byte ~ 50th byte	—		Flash memory start address (4 bytes) 00H, 00H, 01H and 00H beginning at the 47th byte
	51st byte ~ 54th byte	—		Flash memory end address (4 bytes) FFH, FFH, 08H and 00H beginning at the 51st byte
	55th byte ~ 56th byte	—		Flash memory block divisions information (2 bytes) 0AH and 00H beginning at the 55th byte
	57th byte ~ 60th byte	—		Start address of flash memory block of the same size (4 bytes) 00H, 00H, 01H and 00H beginning at the 57th byte

Table 4.3.6 Boot program transfer format (for reading product information) (2/2)

	Number of bytes transferred	Data transferred from the controller to the device	Baud rate	Data transferred from the device to the controller
BOOT ROM	61st byte ~ 64th byte	—		Size (in half words) of flash memory blocks of the same size (4 bytes) 00H, 80H, 00H and 00H beginning at the 61st byte
	65th byte	—		Number of flash memory blocks of the same size (1 byte) 06H
	66th byte ~ 69th byte	—		Start address of flash memory block of the same size (4 bytes) 00H, 00H, 07H and 00H beginning at the 66th byte
	70th byte ~ 73rd byte	—		Size (in half words) of flash memory block of the same size (4 bytes) 00H, 70H, 00H and 00H beginning at the 70th byte
	74th byte	—		Number of flash memory blocks of the same size (1 byte) 02H
	75th byte ~ 78th byte	—		Start address of flash memory block of the same size (4 bytes) 00H, C0H, 08H and 00H beginning at the 75th byte
	79th byte ~ 82nd byte	—		Size (in half words) of flash memory block of the same size (4 bytes) 00H, 10H, 00H and 00H beginning at the 79th byte
	83rd byte	—		Number of flash memory blocks of the same size (1 byte) 02H
	84th byte	—		CHECKSUM value for bytes 5 to 83
	85th byte	(Wait for the next operation command data)		—

\*1: After returning abnormal response, the device waits for the operation command (third byte).

Table 4.3.7 Boot program transfer format (for Auto Chip Erase &amp; UnProtect)

	Number of bytes transferred	Data transferred from the controller to the device	Baud rate	Data transferred from the device to the controller
BOOT ROM	1st byte	Serial operation mode and baud rate settings For UART 86H	Desired baud rate (Table 4.3.1(a))	—
	2nd byte	—		ACK response for Serial Operation Mode For UART If normal (can be set) 86H (If the device determines that the baud rate cannot be set, it will stop operating.)
	3rd byte	Operation command data (40H)		—
	4th byte	—		ACK response for operation command *1 If normal 40H If abnormal X1H If communication error occurs X8H
	5th byte	—		Erase Operation result response data If normal 4FH If abnormal 4CH
	6th byte	—		ACK response If normal B1H If abnormal B4H
	7th byte	(Wait for the next operation command.)		—

\*1: After returning an abnormal response the controller waits for the operation command (third byte).

(6) Boot program

When the device is turned on in Single Boot Mode, the boot program starts up. These functions are detailed in the explanations “1, RAM Transfer command” to “4, Auto Chip Erase & UnProtect command) on the following pages.

1. RAM Transfer command

RAM transfer refers to the storing of data received from the controller in the device's internal RAM. If the transfer terminates normally, the boot program will start running the user program. The maximum size of the user program is 26Kbytes, and the address at which program execution starts is the RAM storage start address.

This RAM transfer function allows unique on-board programming by executing a rewrite program created by the user. In order for on-board programming to be executed as part of the user program, the flash memory command sequence described later in Section 4.5 must be used. The RAM transfer command examines the result of the password matching check before it starts operation. If the password is found not to match, RAM transfer is not performed. Once the RAM transfer command has been completed, access to the entire on-chip RAM is free.

2. Flash Memory SUM command

This command calculates the sum of 512 Kbytes of flash memory and returns the result. There is no operation command available to the boot program which will enable it to read data from the entire area of the flash memory. Instead, this flash memory SUM command must be used. Reading SUM enables revision management of the application program to be carried out.

3. Product Information Read command

This command returns the product name, information about the device's memory and various other details. Specifically, this command returns data from a part of the flash memory area (addresses 08FEF0H to 08FEF3H). Using this data as well as the above Flash Memory SUM command enables revision management of the application program to be carried out.

4. Auto Chip Erase & UnProtect command

This command erases all the blocks of flash memory. Even when protect function is effective, all the blocks of flash memory are erased and protect function is initialized. It is not necessary to perform comparison of a password in this operation.

## 1) RAM Transfer command (See Table 4.3.3.)

1. The data in the first byte is used to set the serial operation mode. For details of how to set the serial operation mode, please refer to part 6), entitled *Determination of Serial Operation mode*, on page 47 of subsection 4.3.2 (6), *Boot program*. If the serial operation mode is found to be UART Mode, the device checks to see if the baud rate can be set. So UART receiving function needs to be disabled (SC1MOD0<RXE> = 0) in the first byte.

- To communicate in UART Mode  
Send the data value 86H from the controller to the target board in UART mode at the desired baud rate. If the serial operation mode is found to be UART Mode, the device checks to see whether the baud rate can be set. If the device finds that the baud rate cannot be set, it stops operation, and thus can no longer be communicated with.

2. The data in the second byte is the ACK response returned by the device for the serial operation mode which is set by the data in the first byte. If the data in the first byte is found to signify UART Mode and the baud rate can be set, the device will return 86H.

- UART Mode  
The device checks to see if the baud rate can be set. If it is found that the baud rate can be set, the device rewrites the BR1CR and BR1ADD registers and returns 86H. If it is found that the baud rate cannot be set, the device stops operation and hence does not send any information. The controller sets a time-out time (5 seconds) after it has finished sending the first byte of data. If the data (86H) cannot be received normally within the time-out time, the device should be considered unable to communicate. Reception is enabled (SC1MOD0 <RXE> = 1) before the data (86H) is written to the transmission buffer.

3. The data received in the third byte is the operation command data. In this case it is the data for the RAM Transfer command (10H).

4. The data in the fourth byte is the ACK response data returned by the device for the operation command data in the third byte. The first, the device checks if the received data in the third byte contains an error. If a Receive error is found, the device returns the ACK response data for a Communications error (bit 3) X8H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

Next, if the data received in the third byte corresponds to one of the operation command values given in Table 4.3.2, the device echoes back the received data (the ACK response data for normal reception). In this case the data value 10H is echoed back and the program branches to the RAM transfer processing routine. After branching to this routine, the device checks the data in the password area. For details of how to check the data in the password area, refer

to part 7), entitled *The password*, on page 50 of subsection 4.3.2 (6), *Boot program*. If the received data does not correspond to any operation command, the device returns the ACK response data for an Operation Command error (bit 0) X1H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

5. The data received in bytes 5 to 16 is password data (12 bytes). The data in bytes 5 to 16 is compared with the data in flash memory address 08FEF4H to 08FEFFH respectively. If the password does not match, the device sets the Password Error flag.
6. The data received as the 17th byte is checksum data. For details of how to calculate CHECKSUM data, please refer to part 9), entitled *Calculating CHECKSUM*, on page 51 of subsection 4.3.2 (6), *Boot program*. The checksum sent by the controller should be equal to the 2's complement of the unsigned 8-bit value obtained by summing the value of bytes 5 to 16, ignoring any overflow.
7. The data in the 18th byte is the ACK response data returned by the device for the data received as bytes 5 to 17 (the ACK response for the CHECKSUM value). The device first checks to see whether the data received as bytes 5 to 17 contains any Receive errors. If a Receive error is found, the device returns the ACK response data for a Communications error (bit 3) 18H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are the four high-order bits of the immediately preceding operation command data, so the value of these bits is always 1.

Next, the device checks the CHECKSUM data in the 17th byte. This check is made to see if the low-order 8-bit value obtained by adding eight bits of the transmission data to bytes 5 to 17 not including the sign (ignoring overflow) is 00H. If the value is not 00H, the device returns the ACK response data for a CHECKSUM error (bit 0) 11H and waits for the next operation command (third byte).

Finally, the device examines the result of the password matching check. In the following cases the device returns ACK response data for a Password error (bit 0) 11H and waits for the next operation command (third byte).

- Irrespective of the result of the password matching check for bytes 5 to 17, all 12 bytes of data in the password area will have the same value (a value other than FFH).
- None of the password data in bytes 5 to 17 matches the corresponding flash memory data.

If all data is found to be normal as a result of the above check, the device returns the ACK response data for normal reception, 10H.

8. The data received as bytes 19 to 22 indicates the start address in RAM in

which the block-transferred data is to be stored. The 19th byte corresponds to address bits 31 to 24 and the 22nd byte corresponds to address bits 7 to 0.

9. The data received as bytes 23 to 24 as bytes 19 to 22 the number of block-transferred bytes. The 23rd byte corresponds to bits 15 to 8 and the 24th byte to bits 7 to 0 of the transfer byte count.
10. The data received as the 25th byte is checksum data. The checksum data sent by the controller should be equal to the 2's complement of the unsigned 8-bit value obtained by summing the values of bytes 19 to 24 ignoring any overflow. For details of how to calculate checksum data, please refer to part 9), entitled *Calculating CHECKSUM*, on page 51 of subsection 4.3.2 (6), *Boot program*.
11. The data in the 26th byte is the ACK response data returned by the device for the data received as bytes 19 to 25 (the ACK response for CHECKSUM value). The device first checks to see whether the data received as bytes 19 to 25 contains any Receive errors. If a Receive error is found, the device returns the ACK response data for a Communications error (bit 3) 18H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are the four high-order bits of the immediately preceding operation command data, so the value of these bits is always 1.

Next, the device checks the CHECKSUM data in the 25th byte. This check is made to see if the low-order 8-bit value obtained by adding eight bits of the transmission data to bytes 19 to 25 not including the sign (ignoring overflow) is 00H. If the value is not 00H, the device returns the ACK response data for a CHECKSUM error (bit 0) 11H and waits for the next operation command (third byte).

- Write your program so that the data in bytes 19 to 25 are stored within a RAM address area from 000400H to 006BFFH.

If all data is found to go normal as a result of the above check, the device returns the ACK response data for normal reception, 10H.

12. The data received as bytes 27 to m is the data which is to be stored in RAM. This data is written to RAM starting at the start address specified by the data received as bytes 19 to 22 and up until the number of bytes specified by the data received as bytes 23 to 24 in the 23rd to the 24th bytes has been reached.
13. The data received as the m + 1'th byte is checksum data. The checksum sent by the controller should be equal to the 2's complement of the unsigned 8-bit value obtained by summing the values of bytes 27 to m, ignoring any overflow. For details of how to calculate checksum data, please refer to part 9), entitled *Calculating CHECKSUM*, on page 51 of subsection 4.3.2 (6), *Boot program*.
14. The data in the m + 2'th byte is the ACK response data returned by the device for the data received as bytes 27 to m + 1 (the ACK response for the CHECKSUM value). The device first checks to see whether the data received as bytes 27 to m + 1 contains any Receive error. If a receive error is found, the

device returns the ACK response data for a Communications error (bit 3) 18H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are the four high-order bits of the immediately preceding operation command data, so the value of these bits is always 1.

Next, the device checks the checksum data in the  $m + 1$ 'th byte. This check is made to see whether the low-order 8-bit value obtained by adding eight bits of the transmission data to 27 to  $m + 1$  bytes not including the sign (ignoring overflow) is 00H. If the value is not 00H, the device returns the ACK response data for a CHECKSUM error (bit 0) 11H and waits for the next operation command (third byte). If all data is found to be normal as a result of the above check, the device returns the ACK response data for normal reception, 10H.

15. If the ACK response data in the  $(m + 2)$ th byte is a normal ACK response, the device sends the normal ACK response data 10H and then branches to the address specified by the data received as bytes 19 to 22.

## 2) Flash Memory SUM command (See Table 4.3.4.)

1. The transmitted/received data in the first and second bytes are the same as for the RAM Transfer command.
2. The data received in the third byte is the operation command data. In this case it is flash memory SUM command data (20H).
3. The data in the fourth byte is the ACK response data returned by the device for the operation command data in the third byte. The first, the device checks if the received data in the third byte contains an error. If a Receive error is found, the device returns the ACK response data for, Communication error (bit 3) X8H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

Next, if the data received in the third byte corresponds to one of the operation command values given in Table 4.3.2, the device echoes back the received data (the ACK response data for normal reception). In this case the data value 20H is echoed back and the program branches to the flash memory SUM processing routine. If the received data does not correspond to any operation command the device returns the ACK response data for an Operation Command error (bit 0) X1H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

4. The data in the fifth and the sixth bytes sent by the device are the high-order and the low-order SUM data respectively. For details of how to calculate SUM, please refer to part 8), entitled *Calculating SUM*, on page 51 of subsection 4.2.2 (6), *Boot program*.
5. The data in the seventh byte sent by the device is CHECKSUM data. This is the low-order 8-bit value in 2's complement representation obtained by adding eight bits of transmit data in the fifth to the sixth bytes not including the sign (ignoring overflow).
6. The data received as the eighth byte is the data value for the next operation command.

## 3) Product Information Read command (See Table 4.3.5.)

1. The transmitted/received data in the first and second bytes are the same as for the RAM Transfer command.
2. The data received in the third byte is the operation command data. In this case it is product information read command data (30H).
3. The data in the fourth byte is the ACK response data returned by the device for the operation command data in the third byte. The first, the device checks if the received data in the third byte contains an error. If a Receive error is found, the device returns ACK response data for a Communication error (bit 3) X8H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

Next, if the data received in the third byte corresponds to one of the operation command values given in Table 4.3.2, the device echoes back the received data (the ACK response data for normal reception). In this case the data value 30H is echoed back, and the program branches to the product information read processing routine. If the received data does not correspond to any operation command data, the device returns the ACK response data for an Operation Command error (bit 0) X1H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

4. The data in bytes 5 to 8 sent by the device is flash memory data (the data at addresses 08FEF0H to 08FEF3H). These addresses can be used to write ID information for the software, facilitating version management.
5. The data in bytes 9 to 20 sent by the device denotes the product name. This data is "TMP92FD54AI" and is transmitted as a sequence of ASCII codes starting from the ninth byte.
6. The data in bytes 21 to 24 sent by the device is the password comparison start address. This data consists of the values F4H, FEH, 08H and 00H and is transmitted starting from the 21st byte.
7. The data in bytes 25 to 28 sent by the device is the RAM start address. This data consists of the values 00H, 04H, 00H and 00H and is transmitted starting from the 25th byte.
8. The data in bytes 29 to 32 sent by the device is the RAM user area end address. This data consists of the values 006BFFH and is transmitted starting from the 29th byte.
9. The data in bytes 33 to 36 sent by the device is the RAM end address. This data consists of the values FFH, 83H, 00H and 00H and is transmitted starting from the 33rd byte.

10. The data in bytes 37 to 44 sent by the device is the dummy data.
11. The data in bytes 45 to 46 sent by the device is the dummy data
12. The data in bytes 47 to 50 sent by the device is the flash memory start address.  
This data consists of the values 00H, 00H, 01H and 00H and is transmitted starting from the 47th byte.
13. The data in bytes 51 to 54 sent by the device is the flash memory end address.  
This data consists of the values FFH, FFH, 08H and 00H and is transmitted starting from the 51st byte.
14. The data in bytes 55 to 56 sent by the device is the data which indicates how many blocks the flash memory is divided into. This data consists of the values 0AH and 00H and is transmitted starting from the 55th byte.
15. The data in bytes 57 to 92 sent by the device is the data containing information on flash memory blocks. With consecutive blocks of a given size starting at the flash memory start address comprising one unit, this information gives the start address of the first block, the size of the block (in half words) and the number of blocks in each unit of a given size.  
  
The data in bytes 57 to 65 sent by the device gives the above information for the 64K-byte blocks (Block 0 to Block 5). Similarly, the data in bytes 66 to 74 gives the information for the 56K-byte blocks (Block 6 and Block 7), the data in bytes 75 to 83 gives the information for the 8-Kbyte blocks (Block 8 and Block 9). For a description of this data, please refer to Table 4.3.5.
16. The data in the 84th byte sent by the device is checksum data. This is the low-order 8-bit value in 2's complement representation obtained by adding eight bits of transmit data in the fifth to the 83rd bytes not including the sign (ignoring overflow).
17. The received data in the 85th byte is the next operation command data.

## 4) Auto Chip Erase &amp; UnProtect command (See Table 4.3.47.)

1. The transmitted/received data in the first and second bytes are the same as for the RAM Transfer command.
2. The data received in the third byte is the operation command data. In this case it is Auto Chip Erase & UnProtect command data (40H).
3. The data in the fourth byte is the ACK response data returned by the device for the operation command data in the third byte. The first, the device checks if the received data in the third byte contains an error. If a Receive error is found, the device returns the ACK response data for, Communication error (bit 3) X8H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

Next, if the data received in the third byte corresponds to one of the operation command values given in Table 4.3.2, the device echoes back the received data (the ACK response data for normal reception). In this case the data value 40H is echoed back and the program branches to the Auto Chip Erase & UnProtect processing routine. If the received data does not correspond to any operation command the device returns the ACK response data for an Operation Command error (bit 0) X1H and waits for the next operation command (third byte). The four high-order bits of the returned ACK response data are undefined. (They are the four high-order bits of the immediately preceding operation command data.)

4. The data in the fifth bytes sent by the device indicates whether collective erase has terminated normally. When collective erase has terminated normally, the device returns collective erase terminated normally code (4FH). If an erase error occurs, the device returns erase error code (4CH).
5. The data in the sixth bytes sent by the device are ACK response data. When collective erase has terminated normally, the device returns collective erase ACK response code (B1H).  
If an erase error occurs, the device returns erase error ACK response code (B4H).
6. The data received as the seventh byte is the data value for the next operation command.

## 5) ACK response data

The boot program notifies the controller of the processing status by sending various codes. Table 4.3.8 to Table 4.3.101 show the ACK response data returned by the device for each item of received data. The four high-order bits of the ACK response data are a direct reflection of the four high-order bits of the operation command data. Bit 3 indicates a Receive error, and bit 0 indicates an Operation Command error, CHECKSUM error or Password Error. Bits 1 and 2 are always 0.

Table 4.3.8 ACK response data to data for determining the serial operation mode

Returned Data	Meaning of Data Returned by the Device
86H	It is found that device can communicate in UART Mode.

www.DataSheet4U.com

Table 4.3.9 ACK response data to operation command data

Returned Data	Meaning of Data Returned by the Device
x8H *1	A Receive error occurred in the operation command data.
x1H *1	Undefined operation command data was received normally.
10H	Data was found to be a RAM Transfer command.
20H	Data was found to be a Flash Memory SUM command.
30H	Data was found to be a Product Information Read command.
40H	Data was found to be an Auto Chip Erase & UnProtect command.

\*1: The four high-order bits are a direct reflection of the four high-order bits of the immediately preceding operation command data.

Table 4.3.10 ACK response data to CHECKSUM data

Returned Data	Meaning of Data Returned by the Device
18H	An error had occurred in the received data.
11H	A CHECKSUM error or password error occurred.
10H	The CHECKSUM was found to be the correct value.

Table 4.3.11 ACK response data to Auto Chip Erase &amp; UnProtect data

Returned Data	Meaning of Data Returned by the Device
4FH	The erase operation has terminated normally
4CH	The erase error occurred.
B1H	Reconfirmation of erase operation
B4H	Reconfirmation of erase error operation

## 6) Determination of Serial Operation mode

When communicating in UART Mode, the controller should transmit the data value 86H as the first byte at the desired baud rate.

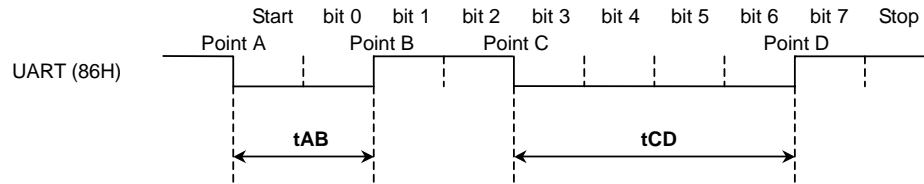


Figure 4.3.3 Data for determining the serial operation mode

The boot program disables reception of the serial operation method recognition data (86H) received from the controller as the first byte after a Reset and finds the durations of  $t_{AB}$ ,  $t_{AC}$  and  $t_{AD}$  shown in Figure 4.3.3 using the procedure shown in the flowchart in Figure 4.3.4. As shown in the flowchart in Figure 4.3.4, when the CPU detects a level change on the receive pin while monitoring the pin status, it latches the current timer value. For this reason, the timer values for  $t_{AB}$ ,  $t_{AC}$  and  $t_{AD}$  are somewhat erroneous. Also, if the baud rate is fast, the CPU may not be able to detect a level change on the receive pin.

As shown in the flowchart in Figure 4.3.5, the serial operation mode is determined by the relative magnitude of the durations  $t_{AB}$  and  $t_{CD}$  when the receive pin level is Low. When  $t_{AB} < t_{CD}$ , the serial operation mode is found to be UART Mode and the boot program determines from the duration of  $t_{AD}$  whether or not the baud rate can be automatically set. Since the timer values for  $t_{AB}$ ,  $t_{AC}$  and  $t_{AD}$  include some error, as mentioned earlier, if the baud rate is fast and the operating frequency slow, the timer values will not be long enough to enable the boot program to determine the serial operation mode correctly as intended.

For example, it may be erroneously determined that the serial operation mode is not UART Mode when in fact the controller intends to communicate in UART Mode. To avoid this kind of problem, if the controller is trying to communicate in UART Mode, a time-out time after transmission of the first byte of data should be set for the controller; if the data 86H cannot be received normally from the device within the time-out time, this can be used to determine that the device cannot communicate.

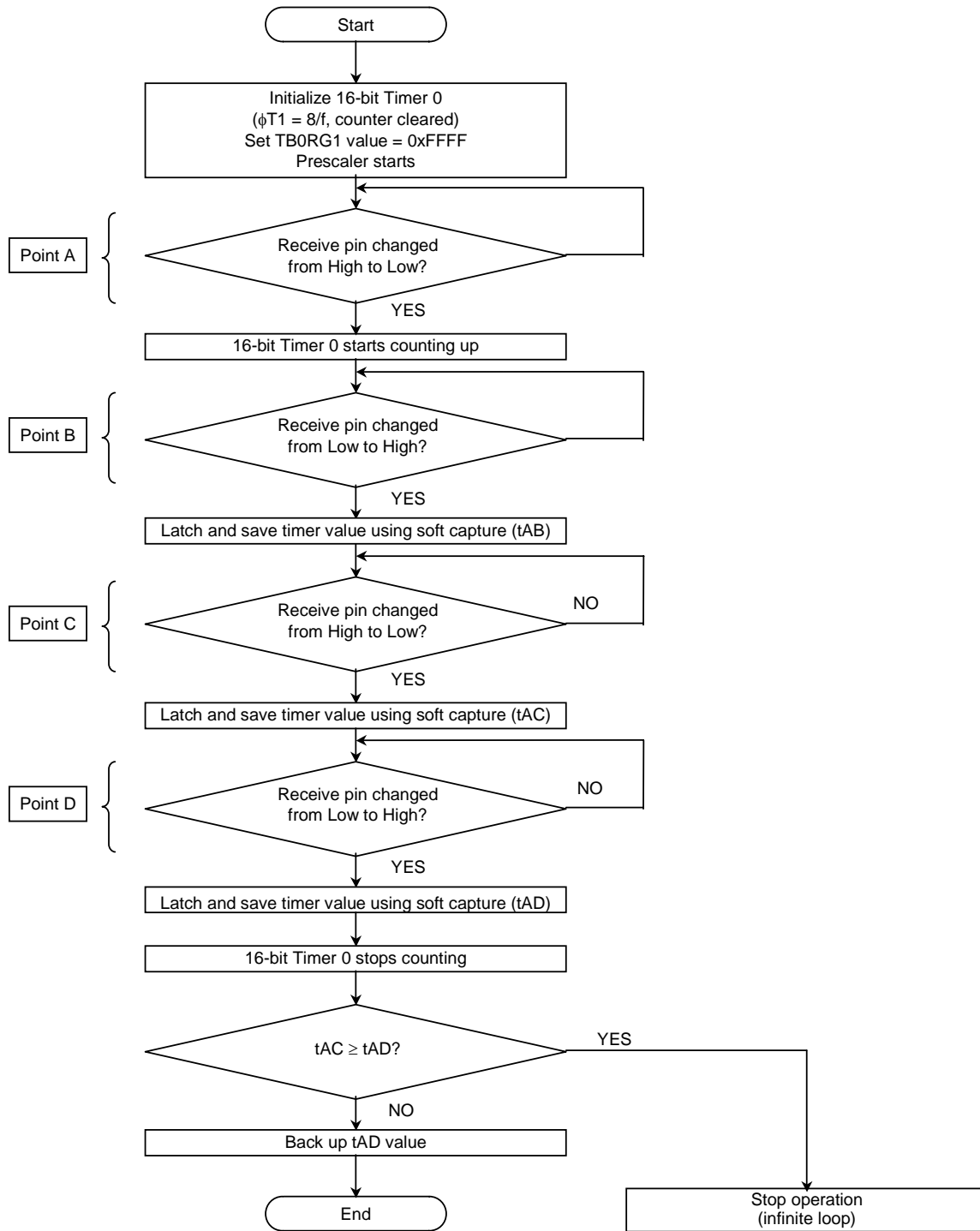


Figure 4.3.4 Serial operation mode reception flowchart

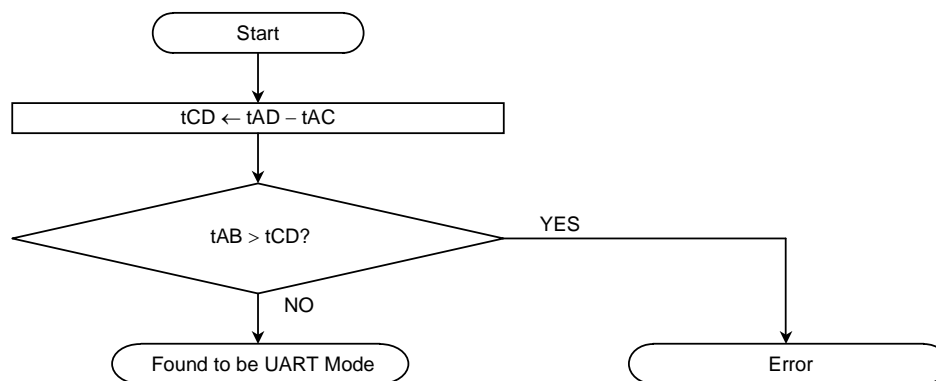


Figure 4.3.5 Flowchart for determination of serial operation mode

## 7) The password

If the received operation command data is a RAM Transfer command (10H), the boot program checks the password. First, after echoing back the received operation command data (10H), the boot program checks the data (12 bytes) in the password area (addresses 08FEF4H to 08FEFFH).

Next, the boot program checks the received data in bytes 5 to 16 (the password data) against the corresponding data in the flash memory. The relationship between the received data and the flash memory data which is compared with it is shown in Table 4.3.5. Unless all of 12 bytes of the data match, a Password error is assumed to have occurred. In this case, the ACK response returned for the CHECKSUM value in the 17th byte will signify a Password error.

Table 4.3.12 Relationship between data received and data in flash memory with which it is compared

Received Data	Flash Memory Data for Comparison
5th byte	Data at address 08FEF4H
6th byte	Data at address 08FEF5H
7th byte	Data at address 08FEF6H
8th byte	Data at address 08FEF7H
9th byte	Data at address 08FEF8H
10th byte	Data at address 08FEF9H
11th byte	Data at address 08FEFAH
12th byte	Data at address 08FEFBH
13th byte	Data at address 08FEFCH
14th byte	Data at address 08FEFDH
15th byte	Data at address 08FEFEH
16th byte	Data at address 08FEFFH

## 8) Calculating SUM

SUM is word value which the boot program returns after summing the values of all the data bytes which it has read. The resulting SUM is sent to the controller high-order byte first and low-order byte last. All the data in the flash memory (512 K-bytes) is included in the calculation of SUM. The following example illustrates how the flash memory SUM command calculates the SUM value which is returned.

Example:

A1H
B2H
C3H
D4H

When the calculation is performed on the four data entries shown on the left, the value of SUM is:

$$A1H + B2H + C3H + D4H = 02EAH$$

Hence the high-order data value for SUM is 02H and the low-order data value is EAH. Therefore SUM is sent to the controller as the value 02H followed by the value EAH.

## 9) Calculating CHECKSUM

CHECKSUM is obtained by taking the 2's-complement of the unsigned 8-bit value obtained by summing the values of the bytes received, ignoring any overflow. The CHECKSUM returned when the Flash Memory SUM command or Product Information Read command is executed is calculated in this way. The controller should also use this calculation method when transmitting CHECKSUM values.

Example: Calculating CHECKSUM for Flash Memory SUM command

In this example the CHECKSUM value is found for when the high-order byte of SUM is E5H and the low-order byte is F6H.

First, the bytes are summed to yield an unsigned result.

$$E5H + F6H = 1DBH$$

Take the 2s complement of the low-order byte of the resulting value as shown below. The result is the CHECKSUM value. Consequently, the value 25H is sent to the controller.

$$0 - DBH = 25H$$

## (7) Boot program general flowchart

Figure 4.3.6 shows an overall flowchart for the boot program.

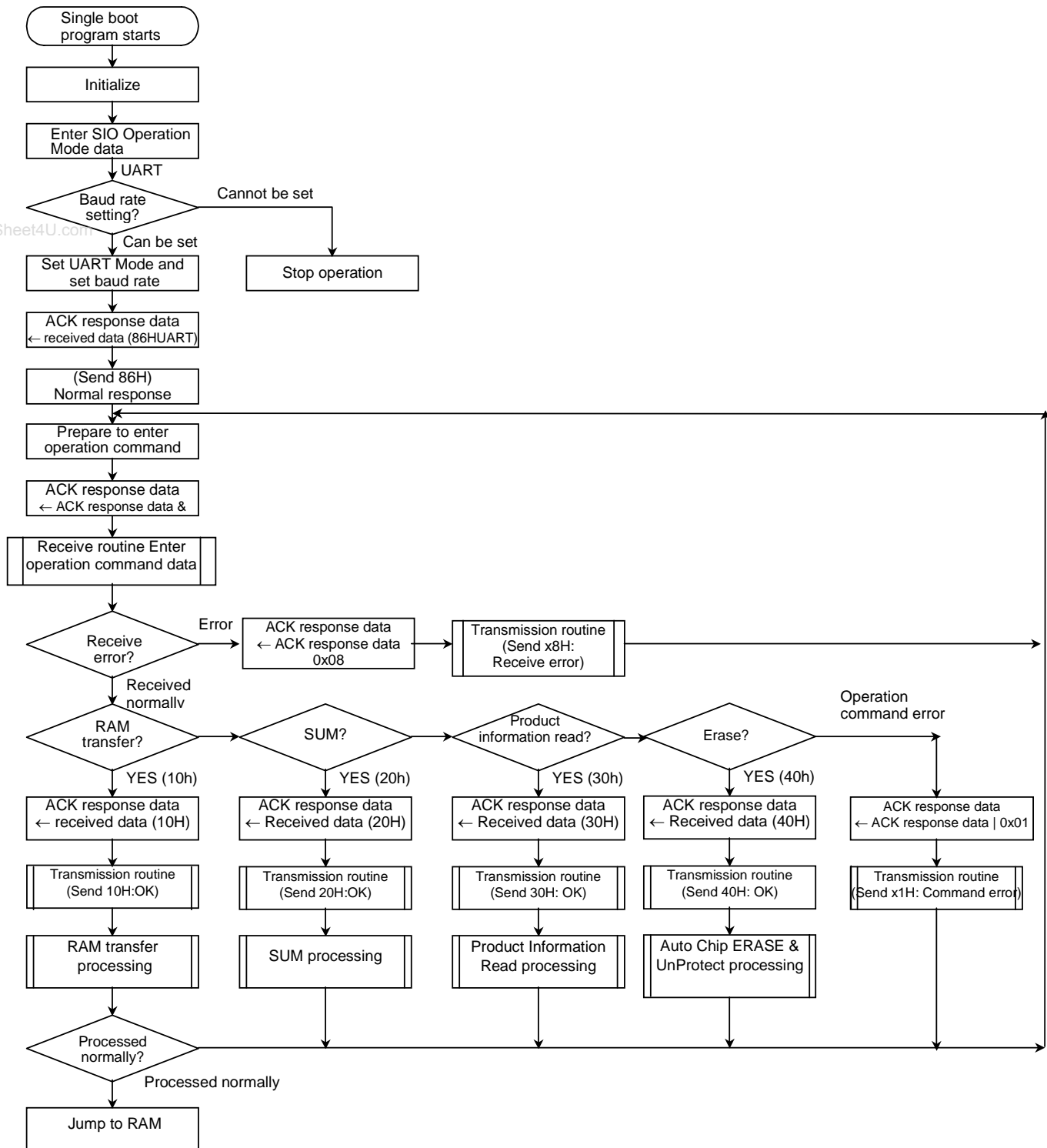


Figure 4.3.6 Overall flowchart for boot program

## 4.4 Programming/Erasing Flash Memory during on-Board Programming

The method used for controlling flash memory commands complies with JEDEC standards for standard E<sup>2</sup>PROMs. For this reason, during RAM transfer operations in User Boot Mode or Single Boot Mode the flash memory is programmed/erased by software commands executed by the CPU. The program for controlling this program/erase operation must be written in advance by the user. Because the flash memory cannot be read while it is being programmed or erased, the program/erase control program must be placed in and executed from the internal RAM or external memory after the device has entered User Boot Mode or after a RAM transfer.

### 4.4.1 Flash memory

With a few exceptions, the functions for programming and erasing the flash memory are based on JEDEC-compliant commands. To program or erase the flash memory, enter a command to the flash memory using the CPU's LD instruction. When a command is entered, the flash memory will automatically be programmed or erased internally. Furthermore, to erase the flash memory, several different methods can be used: all the blocks can be erased together, each block can be erased individually.

Table 4.4.1 Flash memory functions

Main Functions	Description
Auto Program	Automatically writes data and verifies written data in long word units.
Auto Chip Erase	Automatically erases and verifies all the blocks of the flash memory together.
Auto Block Erase	Automatically erases and verifies in units of blocks.
Hardware Sequence flags	Monitoring flags such as a data polling or Toggle Bit flag help to confirm whether the flash memory is being programmed or erased.
Block Protect	Block Protect function prevents individual blocks of flash memory from being programmed or erased.

As will be described later, in User Boot Mode or during RAM transfer the method of address specification for operation commands differs from that for standard commands for reasons of interfacing with the CPU. Also, unless otherwise stated, the flash memory is written to in units of 32 bits. When writing to flash memory, use 32-bit (long word) data transfer instructions. To enter commands, byte (8-bit) transfer instructions can be used.

## (1) Block configuration

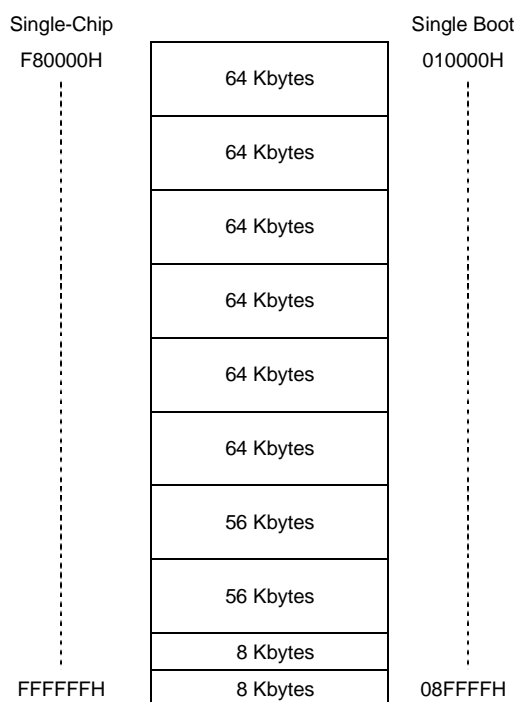


Figure 4.4.1 Flash memory block configuration

## (2) Flash memory interface during on-board programming

Figure 4.4.2 is a conceptual diagram of the flash memory's internal interface. Note, however, that this diagram is only a schematic representation of the interface between the CPU and flash memory, and does not represent the actual circuit.

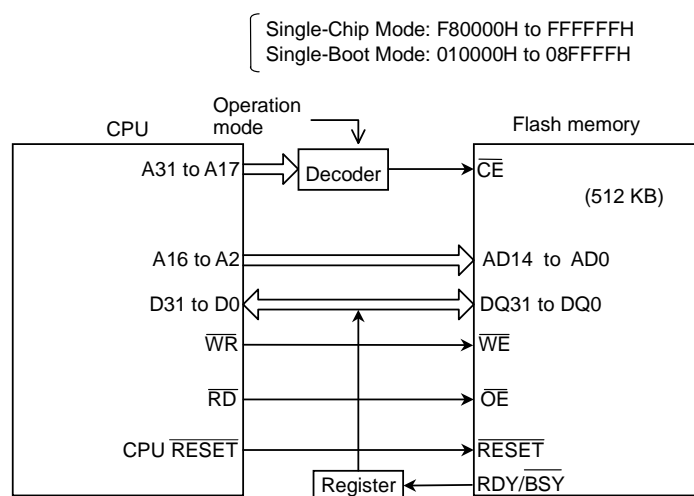


Figure 4.4.2 Flash memory internal interface

### (3) Basic operation

Essentially the flash memory on the TMP92FD54AI has the following two modes of operation:

- Mode in which data is read from memory (Read Mode)
- Mode in which memory data is automatically erased/rewritten (Auto Operation Mode)

Auto Operation Mode can be entered from Read Mode by the execution of a command sequence. In Auto Operation Mode no data can be read from the flash memory.

#### 1) Read

To read data, set the flash memory to Read Mode. The flash memory is automatically placed in Read Mode immediately after power-on, after the CPU is reset or when auto operation is terminated normally. If auto operation is terminated abnormally, or if you want to return to Read Mode from another mode, either use the Read/Reset command (software reset) or a hardware reset. These are described below.

#### 2) Writing commands

The method used for controlling flash memory commands complies with JEDEC standards for standard E2PROMs. To write to the Command Register, execute a command sequence on the flash memory. When the flash memory has latched the input address and data into the Command Register, the program will execute the instruction (see Tables 4.4.3 and 4.4.4).

Except for the fourth Read cycle of a Read/Reset and for the fourth Write cycle of an Auto Program, only DQ0 to DQ7 are used to enter commands. Commands can therefore be entered in byte units.

If you want to cancel command entry in the middle of a command sequence, enter the Reset command. On accepting the Reset command, the flash memory resets the Command Register and enters Read Mode. Likewise, if an erroneous command sequence is entered, the flash memory resets the Command Register and enters Read Mode.

#### 3) Reset

- Read/Reset command (software reset)

When auto operation terminates abnormally, the flash memory does not automatically re-enter Read Mode. In this case use the Read/Reset command to return the flash memory to Read Mode. Likewise, to cancel command entry in the middle of a command sequence, use the Read/Reset command to clear the contents of the Command Register.

- Hardware reset

As shown in Figure 4.4.2, the flash memory has a reset input of its own. This input is connected to the CPU's Reset signal. Hence, if the CPU is reset because the device's RESET input pin is pulled Low (VIL) or because the Watchdog timer has overflowed, the current flash memory operation, whether a read operation (write or erase), will stop and the device will either remain in, or re-enter, Read Mode.

Also, if auto operation terminates abnormally or if the mode which has been set is cleared by the issuing of a command, the device will re-enter Read Mode due to the CPU being reset. Note that if the device is halted by a hardware reset during auto operation, the flash memory data cannot be rewritten correctly. In this case it is necessary to rewrite the data.

For details of the CPU's Reset operation, please refer to Section 4.2.1, *Reset operation*. After being reset by a designated reset input, the CPU reads reset vector data from the flash memory and starts post-Reset operation.

#### 4) Auto Program

Writing to the flash memory involves changing the contents of cells containing 1 to 0. You cannot change 0 to 1. The only way to write 1 to a cell is to erase all the cells.

In User Boot Mode or during RAM transfer, data is written to the flash memory in units of 32 bits (long words). Auto Program operation is begun when a program address and program data (in units of long words) are latched in the fourth Bus Write cycle of the Command cycle. Auto programming starts when the flash memory has latched the program data. Since data is written in units of 32 bits, the program address must be incremented by 4.

Once program operation has begun, programming and program verification are automatically performed inside the chip. The status of an Auto Program operation can be checked by monitoring the Hardware Sequence flags (see Table 4.4.6). During Auto Program operation, the device will not accept any command sequences input. To stop operation, use a hardware reset. Note that if operation is interrupted, data cannot be written to the flash memory correctly.

Data cannot be written to addresses within a protected block. If this is attempted, the device will not execute Auto Program but will enter Read Mode. If Auto Program terminates normally, the device will automatically re-enter Read Mode. If Auto Program terminates in error, the flash memory will be locked into that mode and will not re-enter Read Mode. This status can be confirmed by inspecting the Hardware Sequence flag.

To return to Read Mode, it is necessary to reset the flash memory or the device using the Reset command or a hardware reset. In this case, because a write to the protected block address generated an error, it is recommended that the block which contains the bad address not be used or that the device itself not be used.

#### 5) Auto Chip Erase

Auto Chip Erase operation starts after the sixth Bus Write cycle of the Command cycle has ended. Once it has started, all flash memory addresses inside the chip are automatically programmed to 0; this is followed by the execution of Erase and Erase Verify. The status of an Auto Chip Erase operation can be checked by monitoring the Hardware Sequence flags (see Table 4.4.5). During an Auto Chip Erase operation, the device cannot accept any command sequences input. To stop operation, use a hardware reset. Note that if operation is interrupted, data cannot be erased correctly and hence the Auto Chip Erase must be re-executed.

Protected blocks, if any, cannot be erased. If all blocks are protected, the device will not execute Auto Chip Erase but will enter Read Mode.

In this case, the deletion operation other block of the following number of the protected block will be excuted.

If Auto Chip Erase terminates normally, the device will automatically re-enter Read Mode. If Auto Chip Erase terminates in error, the flash memory will be locked in that mode and will not re-enter Read Mode. The status can be checked by inspecting the Hardware Sequence flag.

To return to Read Mode, it is necessary to reset the flash memory or the device using the Reset command or a hardware reset. In this case it will be impossible to detect which block generated the error. Therefore, it is recommended that the bad block be located using the Block Erase function and that henceforth use the bad block not be used.

#### 6) Auto Block Erase

An Auto Block Erase starts the Erase Hold time after the sixth Bus Write cycle of the Command cycle has ended. Once it has started, all flash memory addresses in the selected block are automatically programmed to 0; this is followed by the execution of Erase and Erase Verify.

If any command sequence other than an Auto Block Erase is entered during the Erase Hold time, the flash memory will be reset and will enter Read Mode. The status of an Auto Block Erase operation can be checked by monitoring the Hardware Sequence flags (see Table 4.4.6). During an Auto Block Erase operation, the device cannot accept any command sequences input. To stop operation, use a hardware reset. Note that in this case, data cannot be erased correctly, and hence the Auto Block Erase must be re-executed.

Protected blocks, if any, cannot be erased. If all blocks are protected, the device will not execute Auto Block Erase but will enter Read Mode. If Auto Block Erase terminated in error, the flash memory will be locked in that mode and will not re-enter Read Mode. The status can be checked by inspecting the Hardware Sequence flag. To return to Read Mode, it is necessary to reset the flash memory or the device using the Reset command or a hardware reset.

## 7) Auto Block Protect

Auto Block Protect allows individual blocks to be protected from being written or erased. The table below shows the relationship between protection status and the rewrite operation.

Table 4.4.2 Block protection status and the Rewrite operation

Protection Status and Operation Executed	Result of executed operation
Execute Program on protected block	Not programmed and automatically returns to Read Mode
Execute Erase on protected block	Not erased and automatically returns to Read Mode
Execute Chip Erase when all blocks are protected	Not erased and automatically returns to Read Mode
Execute Chip Erase when any blocks are protected	Only the unprotected blocks are erased. Upon completion, the flash memory automatically returns to Read mode

Writing to the protect circuit is initiated by the address of the block to be protected in the sixth Bus Write cycle of the command sequence, and set data to "70H". (see Figure 4.4.8)

During automatic operation no other command sequence input can be accepted. To stop operation use a hardware reset.

When it erases Auto Block Protect, use Auto Chip Erase & Unprotect command. (see Table 4.4.3) In this case it erases all Auto Block Protect and all blocks of the flash memory.

## 8) Verify Block Protect

Verify Block Protect is used to check whether or not blocks are protected. Enter the address of the block you want to check in the fourth Bus Read cycle of the command sequence. At this time, specify an address where  $A3 = A6 = 0$ ,  $A4 = 1$ , and read the selected block in units of long words. If the block is protected, the data returned will be 00000001H. Conversely, if the block is unprotected, the data returned will be 00000000H.

To continue verifying the protection of other blocks, it is only necessary to repeating the fourth Bus Read cycle so as to read out data from each block. Change the block address as desired and read the data out in long words. To return to Flash Memory Read or any other command input after the Verify Block Protect has been completed, it is necessary to issue a Read/Reset command or a hardware reset.

## 9) Hardware Sequence flags (see Table 4.4.6)

The status of automatic execution of operations on flash memory can be checked by inspecting the Hardware Sequence flags. During automatic operation data can be read out with the same timing as that of Read Mode. The flash memory automatically returns to Read Mode after automatic operation has finished. During automatic execution it is possible to monitor the operating status by inspecting the Hardware Sequence flags. After automatic operation has finished, the operating status can be confirmed by checking the read data to see that it matches the cell data.

To inspect the Hardware Sequence flags during an Auto Program operation, specify the same address as the one which has been set by the rewrite routine (in which  $A0 = A1 = 0$ ) at the same time as reading out the flag. Also, to inspect the Hardware Sequence flags during an Auto Erase operation, specify an address in which  $A0 = A1 = 0$  at the same time as reading out the flag.

- **DQ7 (Data Polling)**  
The Data Polling function allows confirmation of the status of automatic operations on flash memory. Data polling output begins after the last Bus Write cycle of an automatic operation command sequence has ended. During an Auto Program operation the data written to DQ7 is output after being inverted. After an Auto Program operation, the cell data for DQ7 itself is output so that the operation status can be identified by reading DQ7. During an Auto Erase operation the data 0 is output on DQ7; after an Auto Erase operation the data 1 (the cell data) is output on DQ7. If an automatic operation resulted in an error, DQ7 will continue to output the data it held during the automatic operation. Therefore, this data can be used in combination with the value on DQ5 (the internal timer-out) to determine whether failure has occurred (see Figure 4.4.6).

Since the flash memory releases the address latch on completion of operation, to read out data it is necessary to specify either the address to which data has been written or the address of any unprotected erased block.

- **DQ6 (Toggle bit)**  
The Toggle bit function, same as the data polling function, allows confirmation of the status of automatic operations on flash memory. The Toggle output begins after the last Bus Write cycle of an automatic operation command sequence has ended on DQ6. Every read cycle, output reciprocally 0 and 1. If an automatic operation resulted in an error, DQ6 will continue to output the data it held during the automatic operation. Therefore, this data can be used in combination with the value on DQ5 (the internal timer-out) to determine whether failure has occurred (see Figure 4.4.6).
- **DQ5 (internal time-out)**  
Normally when automatic operation is in progress, the flash memory outputs 0 on DQ5. If the automatic operation exceeds the flash memory's internal time-out period, the DQ5 output will change state to 1. This indicates that automatic operation did not terminate normally and suggests that the flash memory may be faulty. Furthermore, since the flash memory can have its cells changed from data 1 to data 0 in Program Mode, but cannot have its cells changed from data 0 to data 1, if an attempt is made to write a data 1 to a data 0 cell, the device will not be able to write data correctly within the predetermined time. As a result, the device will output a 1 on DQ5, just as if the flash memory were found to be faulty.

In this case the value on DQ5 is not indicating that the flash memory is faulty, but that an attempt to perform an invalid operation was made. If automatic operation did not terminate normally, the flash memory will be locked and will not re-enter Read Mode. Therefore, reset the flash memory using the Reset command.

- **DQ3 (Block Erase timer)**  
Auto Block Erase starts the Block Erase Hold time after the sixth Bus Write cycle of the Command cycle has ended. The flash memory outputs a 0 on DQ3 during the Block Erase Hold time and outputs a 1 when it starts an Erase operation.

## 10) Flash Status Register

This is an 8-bit register which serves purposes of a flash memory status monitor.

Flash status register								
	7	6	5	4	3	2	1	0
Bit symbol	-	-	-	-	-	R/BSY	-	-
Read/Write	R/W	R/W	R/W	R/W		R		
After reset	0	0	0	0	-	1	-	-
FLSR (16EH)  Function	Set to 0.	Set to 0.	Set to 0.	Set to 0.		Ready /Busy flag 0:Busy (auto operation in progress) 1:Ready (auto operation finished)		

Figure 4.4.3(1) Flash Status Register

## Bit 2: Ready/Busy Flag bit

The device has a RDY/ $\overline{\text{BSY}}$  output which enables the status of automatic operation to be recognized by the host in Programmer Mode. This bit is used to monitor the RDY/BSY output from the CPU. When automatic operation is being performed on the flash memory, this bit outputs a 0, indicating that the flash memory is busy. When automatic operation has finished, it will output 1 and the flash memory will be ready to accept the next command. If automatic operation results in error, this bit will continue to output 0. The bit is reset to 1 by a hardware reset.

The bit outputs 0 starting from completion of the last Bus Write cycle of the automatic operation command sequence. For an Auto Block Erase, however, it starts outputting 0 the Erase Hold time after the said cycle has ended. If this bit = 0, no command sequence input can be accepted.

## 11) FLASH Security Write Enable Register

This is an 8-bit register which allows to enable Auto Chip Erase & Unprotect command.

FLASH Security Write Enable Resister								
	7	6	5	4	3	2	1	0
Bit symbol	-	-	-	-	-	-	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0
FSWE (16BH)  Function	C9H:Auto Chip Erase & Unprotect command Enable code Others: Auto Chip Erase & Unprotect command Disable code							

Figure 4.4.3(2) FLASH Security Write Enable Register

Pleas write it in the FSWE register when you execute the program with RAM.

## (4) Command sequence list

Table 4.4.3 Flash memory access by internal CPU

Command Sequence	Number of Bus Cycles	1st Bus Write Cycle		2nd Bus Write Cycle		3rd Bus Write Cycle		4th Bus Read/Write Cycle		5th Bus Write Cycle		6th Bus Write Cycle	
		Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data
Read/Reset	1	xXXX0H	F0H										
Read/Reset	3	xAAA8H	AAH	x5554H	55H	xAAA8H	F0H	RA	RD				
Auto Program	4	xAAA8H	AAH	x5554H	55H	xAAA8H	A0H	PA	PD				
Auto Chip Erase	6	xAAA8H	AAH	x5554H	55H	xAAA8H	80H	xAAA8H	AAH	x5554H	55H	xAAA8H	10H
Auto Block Erase	6	xAAA8H	AAH	x5554H	55H	xAAA8H	80H	xAAA8H	AAH	x5554H	55H	BA	30H
Auto Block Protect	6	xAAA8H	AAH	x5554H	55H	xAAA8H	9AH	xAAA8H	AAH	x5554H	55H	BA	70H
Verify Block Protect	4	xAAA8H	AAH	x5554H	55H	xAAA8H	90H	BPA	BD				
Auto Chip Erase & Unprotect *1	6	xAAA8H	AAH	x5554H	55H	xAAA8H	80H	xAAA8H	AAH	x5554H	55H	xAAA8H	10H

Note: There must be an interval of at least two instructions between each Bus Cycle.

\*1 When this command is operated, be sure to set FSWE.

The addresses to be accessed from the CPU are shown below.

Table 4.4.4 Relationship between addresses

Command Address	CPU Addresses: A23 to A0																
Addr.	A23 ~ A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0000H	Flash	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AAA8H	memory	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
5554H	address area	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0

## (5) Additional explanation

- F0H, AAH, 55H, A0H, 80H, 10H, 30H: Command data. Write data in units of bytes.
  - RA: Read address  
RD: Read data output
  - PA: Program address  
PD: Program data input
  - BA: Block address (BA0~BA6)
  - BPA: Verify Block Protect address
  - BD: Block Protect data
- Write data to addresses which are divisible by 4 in units of long word
- Please refer to Table 4.4., Block Erase addresses.
- Please refer to part 8), entitled *Verify Block Protect*, on page 58 of Section 3.4.1 (3), *Basic operation*.
- If the block is protected, the data 0x0000\_0001 will be returned; if it is not protected, the data 0x0000\_0000 will be returned.

Table 4.4.5 Block Erase addresses

Block	Address Area		Size
	User Boot Mode	Single Boot Mode	
BA0	F80000H to F8FFFFH	010000H to 01FFFFH	64Kbite
BA1	F90000H to F9FFFFH	020000H to 02FFFFH	64Kbite
BA2	FA0000H to FAFFFFH	030000H to 03FFFFH	64Kbite
BA3	FB0000H to FBFFFFH	040000H to 04FFFFH	64Kbite
BA4	FC0000H to FCFFFFH	050000H to 05FFFFH	64Kbite
BA5	FD0000H to FDFFFFH	060000H to 06FFFFH	64Kbite
BA6	FE0000H to FEFFFFH	070000H to 07DFFFFH	56Kbite
BA7	FEE000H to FFBFFFFH	07E000H to 08BFFFFH	56Kbite
BA8	FFC000H to FFDFFFFH	08C000H to 08DFFFFH	8Kbite
BA9	FFE000H to FFFFFFFH	08E000H to 08FFFFH	8Kbite

Enter an address within the address area of the block to be selected. This should be an address for which A0 = A1 = 0.

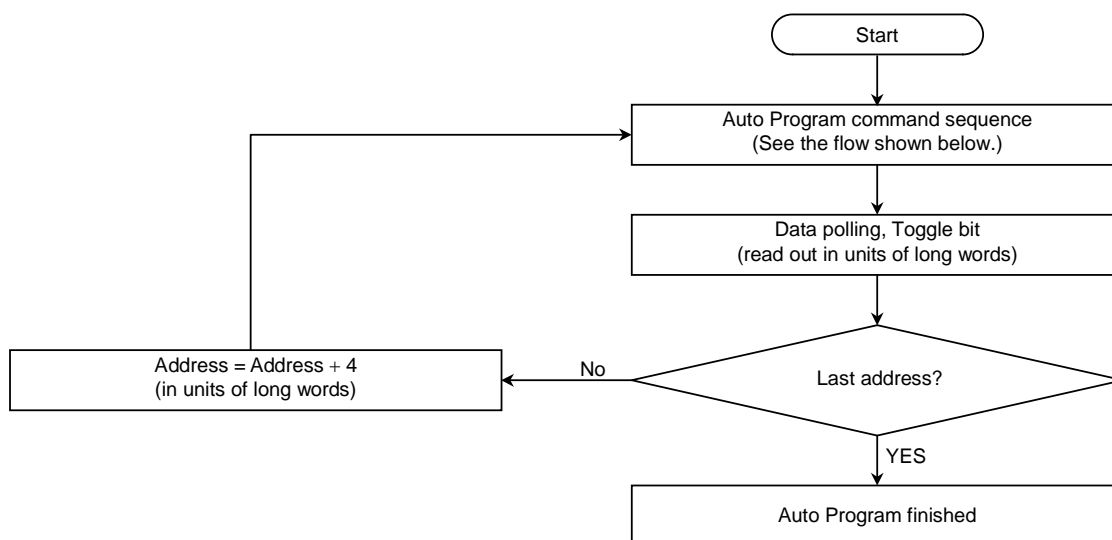
Example: To select BA0 in User Boot Mode, enter any address in the area F80000H to F8FFFFH.

Table 4.4.6 Hardware Sequence flags

Status		D7 (DQ7)	D6 (DQ6)	D5 (DQ5)	D3 (DQ3)
Automatic operation in progress	Auto Program	/D7	Toggle	0	0
	Auto Erase (during Erase Hold time)	0	Toggle	0	1
	Auto Protect Program	*	Toggle	0	0
	Auto Chip Erase & Unprotect	*	Toggle	0	1
Time-out (automatic operation failed)	Auto Program	/D7	Toggle	1	0
	Auto Erase	0	Toggle	1	1
	Auto Protect Program	*	Toggle	1	0
	Auto Chip Erase & Unprotect	*	Toggle	1	1
READ	Complete normally	Cell data	Cell data	Cell data	Cell data

Note: The settings for D31 to D8, D4 and D2 to D0 are "Don't care".

## (6) Flowchart



Auto Program command sequence (address/command)

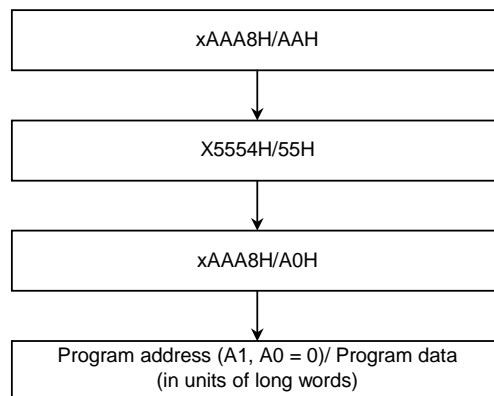


Figure 4.4.4 Auto Program

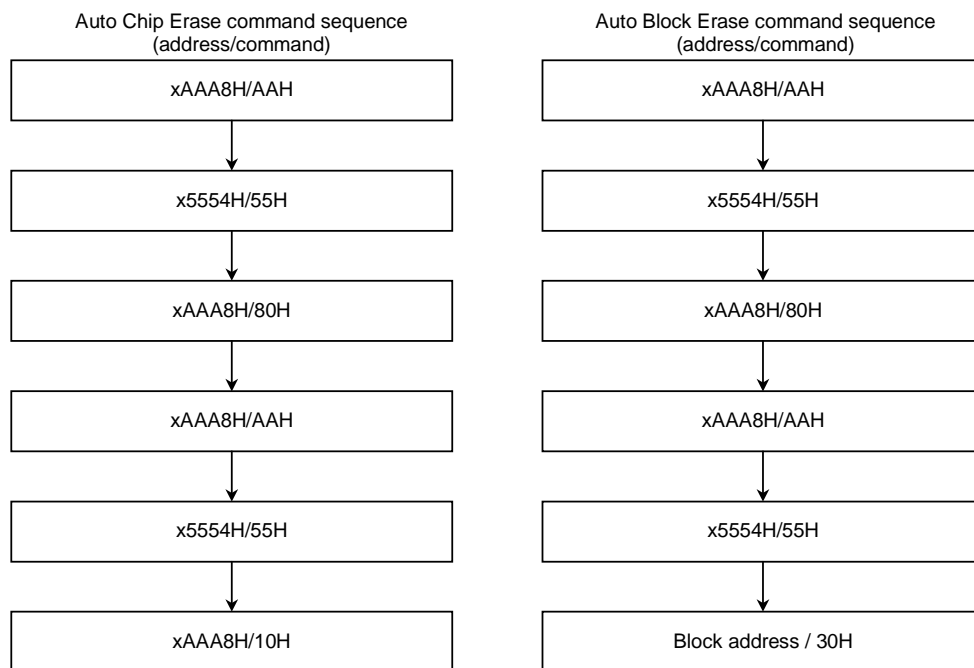
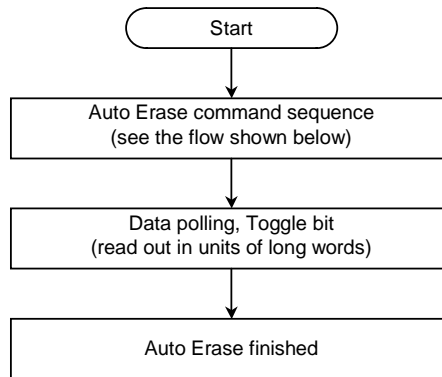


Figure 4.4.5 Auto Erase

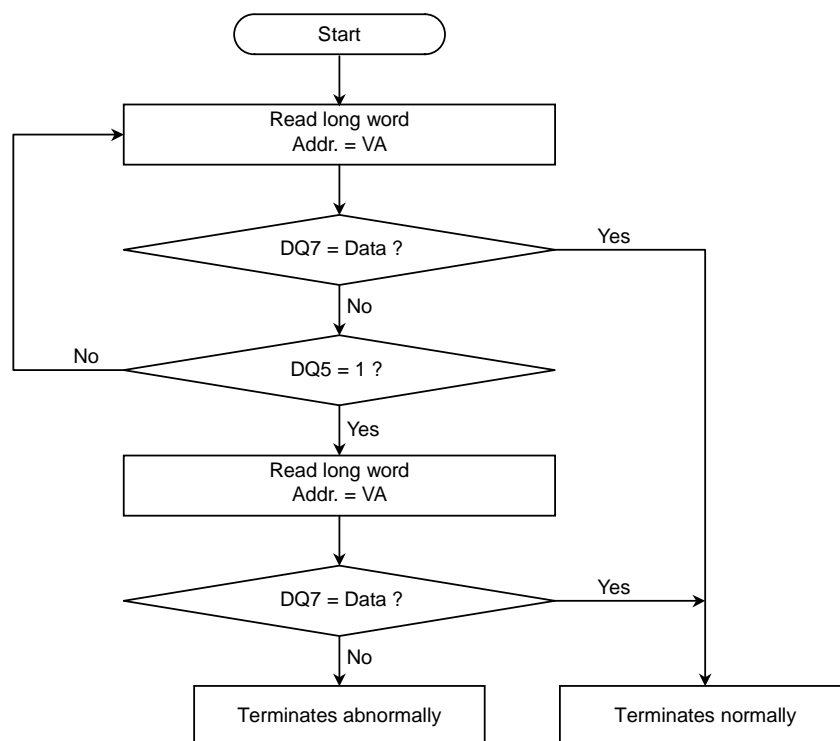


Figure 4.4.6 DQ7 data polling

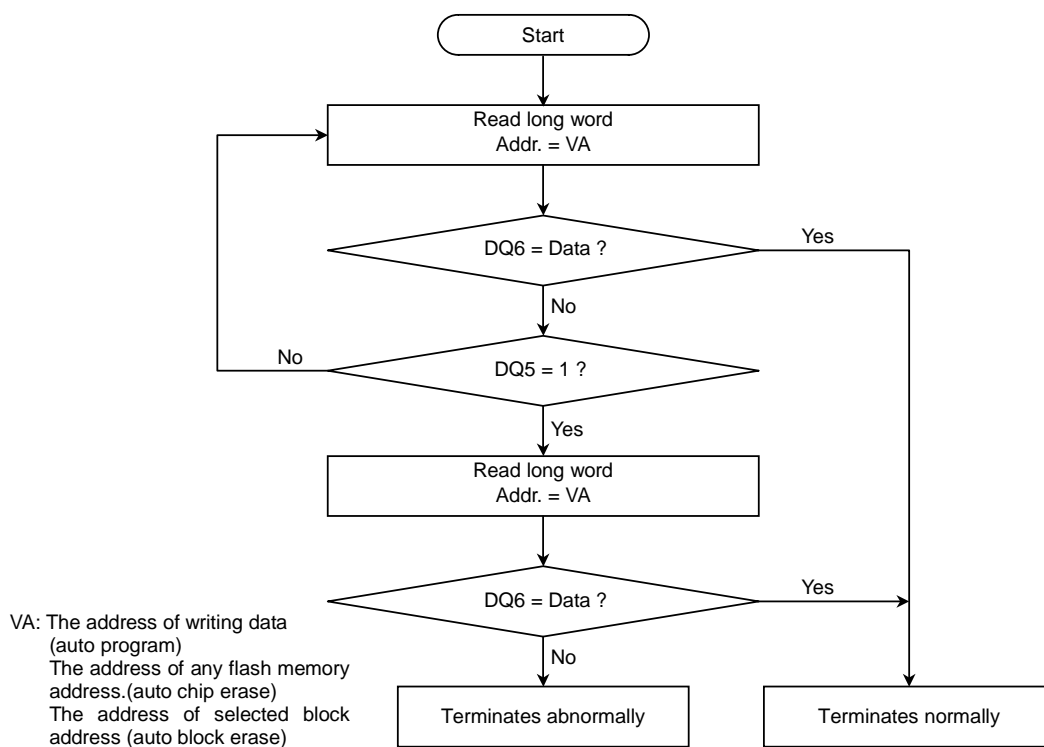


Figure 4.4.7 DQ6 toggle bit

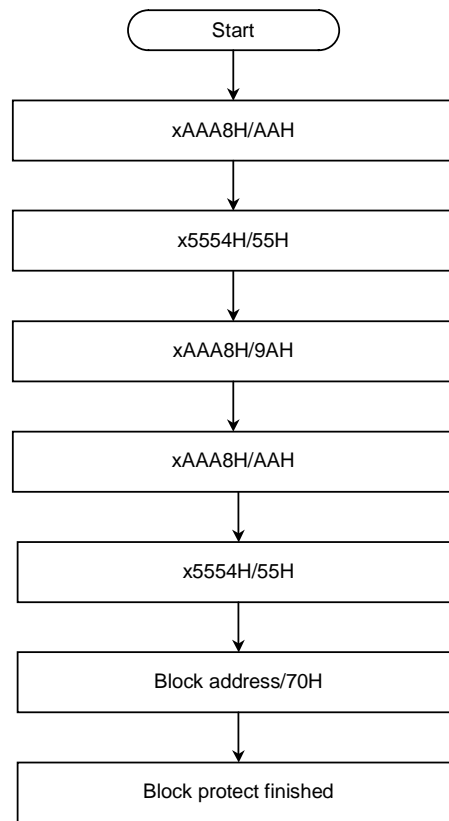


Figure 4.4.8 Auto Block Protect

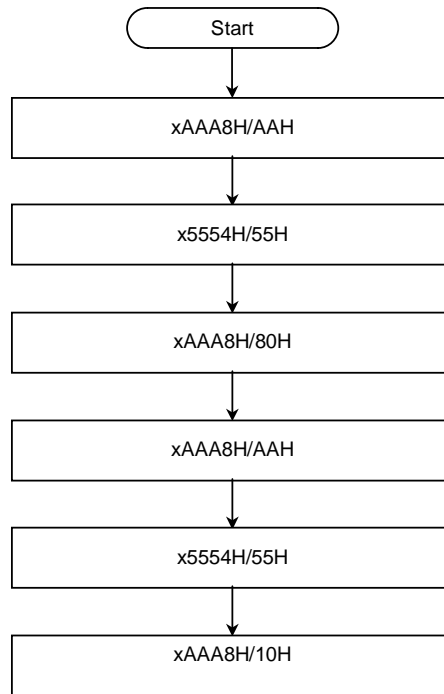


Figure 4.4.10 Auto Chip Erase &amp; Unprotect

## 5. Electrical Characteristics

### 5.1 Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Power Supply Voltage	V <sub>CC5</sub>	-0.5 to 6.0	V
Input Voltage	V <sub>IN</sub>	-0.5 to V <sub>CC5</sub> +0.5	V
Output Current(total)	$\Sigma I_{OL}$	100	mA
Output Current(total)	$\Sigma I_{OH}$	-100	mA
Power Dissipation(Ta=85degree C)	P <sub>D</sub>	600	mW
Soldering Temperature(10s)	T <sub>SOLDER</sub>	260	degree C
Storage Temperature	T <sub>STG</sub>	-65 to 150	degree C
Operation Temperature	T <sub>OPR</sub>	-40 to 85	degree C
Operation Temperature (Flash Program / Erase)		0 to 70	
The number of write erase cycles	N <sub>EW</sub>	100	Cycle

Note: The absolute maximum ratings are rated values that must not be exceeded during operation, even for an instant. Any one of the ratings must not be exceeded. If any absolute maximum rating is exceeded, a device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when designing products that include this device, ensure that no absolute maximum rating value will ever be exceeded.

## 5.2 DC Electrical Characteristics

Vcc5 = 4.5V to 5.25V / fc = 16 to 20MHz / Ta = -40 to 85 degree C

Parameter	Symbol	Condition	Min	Max	Unit
Supply Voltage	V <sub>CC5</sub>		4.5	5.25	V
Input Low Voltage P00 to P07(D0 to 7) PG0 to PG7 PL0 to PL3	V <sub>IL0</sub>		-0.3	0.8	V
Input Low Voltage P00 to P07(PORT) P40 to P47	V <sub>IL1</sub>		-0.3	0.3*V <sub>CC5</sub>	V
Input Low Voltage INT0 <u>NMI</u> RESET P70, P71, P73 to P75 PC0 to PC5 PD0 to PD7 PF0 to PF7 PM0 to PM4	V <sub>IL2</sub>		-0.3	0.25*V <sub>CC5</sub>	V
P72, PN0 to PN6	V <sub>IL6</sub>		-0.3	0.3*V <sub>CC5</sub>	V
Input Low Voltage AM0 to AM1 TEST0 to TEST1	V <sub>IL3</sub>		-0.3	0.3	V
Input Low Voltage X1, XT1 (Crystal)	V <sub>IL4</sub>	* Vcc3 = 3.3V	-0.3	0.2*V <sub>CC3</sub>	V
Input Low Voltage XT1 (CR)	V <sub>IL5</sub>	* Vcc3 = 3.3V	-0.3	0.2*V <sub>CC3</sub>	V
Input High Voltage P00 to P07(D0 to 7) PG0 to PG7 PL0 to PL3	V <sub>IH0</sub>		2.2	V <sub>CC5</sub> +0.3	V
Input High Voltage P00 to P07 P40 to P47	V <sub>IH1</sub>		0.7*V <sub>CC5</sub>	V <sub>CC5</sub> +0.3	V
Input High Voltage INT0 <u>NMI</u> RESET P70, P71, P73 to P75 PC0 to PC5 PD0 to PD7 PF0 to PF7 PM0 to PM4	V <sub>IH2</sub>		0.75*V <sub>CC5</sub>	V <sub>CC5</sub> +0.3	V
P72, PN0 to PN6	V <sub>IH6</sub>		0.7*V <sub>CC5</sub>	V <sub>CC5</sub> +0.3	V
Input High Voltage AM0 to AM1 TEST0 to TEST1	V <sub>IH3</sub>		V <sub>CC5</sub> -0.3	V <sub>CC5</sub> +0.3	V
Input High Voltage X1, XT1 (Crystal)	V <sub>IH4</sub>	* Vcc3 = 3.3V	0.8*V <sub>CC3</sub>	V <sub>CC3</sub> +0.3	V
Input High Voltage XT1 (CR)	V <sub>IH5</sub>	* Vcc3 = 3.3V	0.7*V <sub>CC3</sub>	V <sub>CC3</sub> +0.3	V

Parameter	Symbol	Condition	Min	Max	Unit
Output Low Voltage	$V_{OL}$	$I_{OL} = 3.0\text{mA}$		0.4	V
Output High Voltage	$V_{OH0}$	$I_{OH} = -400\mu\text{A}$	2.4		V
	$V_{OH1}$	$I_{OH} = -100\mu\text{A}$	$0.75 \cdot V_{CC5}$		
	$V_{OH2}$	$I_{OH} = -20\mu\text{A}$	$0.9 \cdot V_{CC5}$		
	$V_{OHn}$	$I_{OH} = -200\mu\text{A}$ , PF6(TX) pin	$0.82 \cdot V_{CC5}$		
Input Leakage Current	$I_{LI}$	$0.0 \leq V_{in} \leq V_{CC5}$	0.02(typ.)	+/- 5	$\mu\text{A}$
Output Leakage Current	$I_{LO}$	$0.2 \leq V_{in} \leq V_{CC5}-0.2$	0.05(typ.)	+/- 10	$\mu\text{A}$
Operating Current (Single Chip)*	$I_{CC5}$	$V_{CC5}=5.25\text{V}$ , X1=10MHz(Internal 20MHz)	80(typ)	100	mA
Operating Current (Stand-by)	$I_{CC5IDLE2}$	IDLE2 Mode $V_{CC5}=5.25\text{V}$ , X1=10MHz(Internal 20MHz)		90	mA
	$I_{CC5IDLE1}$	IDLE1 Mode $V_{CC5}=5.25\text{V}$ , X1=10MHz(Internal 20MHz)		30	
	$I_{CC5IDLE3}$	IDLE3 Mode $V_{CC5}=5.25\text{V}$ , $T_a = -40$ to $85$ degree C $V_{CC5}=5.25\text{V}$ , $T_a = -10$ to $55$ degree C		220 140	$\mu\text{A}$
	$I_{CC5STOP}$	STOP Mode $V_{CC5}=5.25\text{V}$ , $T_a = -40$ to $85$ degree C $V_{CC5}=5.25\text{V}$ , $T_a = -10$ to $55$ degree C		200 120	$\mu\text{A}$
Stand-by Voltage	$V_{STB5}$	$V_{CC3} < V_{CC5}$ , $V_{IH1} < V_{CC5}$ , $V_{IH2} < V_{CC5}$ , $V_{IH3} < V_{CC5}$	3.0	5.25	V
Pull-up Resistor	$R_{RST}$	RESET	60	220	K ohm
	$R_{CLK}$	CLK			
	$R_{REGEN}$	REGEN			
Schmitt Width	$V_{TH}$	INT0, NMI, RESET, P70 to P75, PC0 to PC5, PD0 to PD7, PF0 to PF7, PM0 to PM4, PN0 to PN6	0.4	1.0(typ.)	V

\*: On condition that external bus don't operate

Flash Single-boot Mode

V<sub>CC5</sub> = 4.5V to 5.25V / f<sub>c</sub> = 16 to 20MHz / T<sub>a</sub> = -40 to 85 degree C  
 ( T<sub>a</sub> = 0 to 70 degree C during programming or erasing of flash memory)

Parameter	Symbol	Condition	Min	Max	Unit
Mean operating current (during reading)	I <sub>DDO1</sub>	f <sub>c</sub> = 20 MHz	80	100	mA
Mean operating current (during programming)	I <sub>DDO2</sub>		—	100	mA
Mean operating current (during erasing)	I <sub>DDO3</sub>		—	110	mA
Standby current	I <sub>DDS</sub>	V <sub>CC5</sub> = 5.25V, T <sub>a</sub> = -40 to 85 degree C V <sub>CC5</sub> = 5.25V, T <sub>a</sub> = -10 to 55 degree C	—	200 120	μA

Note: Precautions when programming/erasing flash memory

- 1) In On-Board Programming Mode (Single-Boot Mode or User Boot Mode), inhibit all interrupts including NMI to allow the highest priority for program/erase operations.
- 2) To rewrite data in already programmed addresses, execute an Auto Erase before executing Auto Program.

## 5.3 AC Characteristics

## Read cycle

VCC5=4.5 to 5.25V±5%, TA=-40 to 85 degree C

No.	Parameter	Symbol	Min	Max	@20MHz	@16MHz	Unit
1	OSC period (X1/X2)	$t_{OSC}$	100	125	100	125	ns
2	System Clock period (=T)	$t_{CYC}$	50	62.5	50	62.5	ns
3	CLK Low Width	$t_{CL}$	$0.5 \times T-15$		10	16	ns
4	CLK High Width	$t_{CH}$	$0.5 \times T-15$		10	16	ns
5-1	A0 to A23 Valid → D0 to D7 Input @0WAIT	$t_{AD}$		$2.0 \times T-50$	50	75	ns
5-2	A0 to A23 Valid → D0 to D7 Input @1WAIT	$t_{AD3}$		$3.0 \times T-50$	100	138	ns
6-1	RD Fall → D0 to D7 Input @0WAIT	$t_{RD}$		$1.5 \times T-45$	30	49	ns
6-2	RD Fall → D0 to D7 Input @1WAIT	$t_{RD3}$		$2.5 \times T-45$	80	111	ns
7-1	RD Low Width @0WAIT	$t_{RR}$	$1.5 \times T-20$		55	74	ns
7-2	RD Low Width @1WAIT	$t_{RR3}$	$2.5 \times T-20$		105	136	ns
8	A0 to A23 Valid → RD Fall	$t_{AR}$	$0.5 \times T-20$		5	11	ns
9	RD Fall → CLK Fall	$t_{RK}$	$0.5 \times T-20$		5	11	ns
10	A0 to A23 Valid → D0 to D7 Hold	$t_{HA}$	0		0	0	ns
11	RD Rise → D0 to D7 Hold	$t_{HR}$	0		0	0	ns
12	A0 to A23 Valid → PORT Input	$t_{APR}$		$2.0 \times T-120$	-20	5	ns
13	A0 to A23 Valid → PORT Hold	$t_{APH}$	$2.0 \times T$		100	125	ns
14	WAIT Set-up Time	$t_{TK}$	15		15	15	ns
15	WAIT Hold Time	$t_{KT}$	5		5	5	ns

## Write cycle

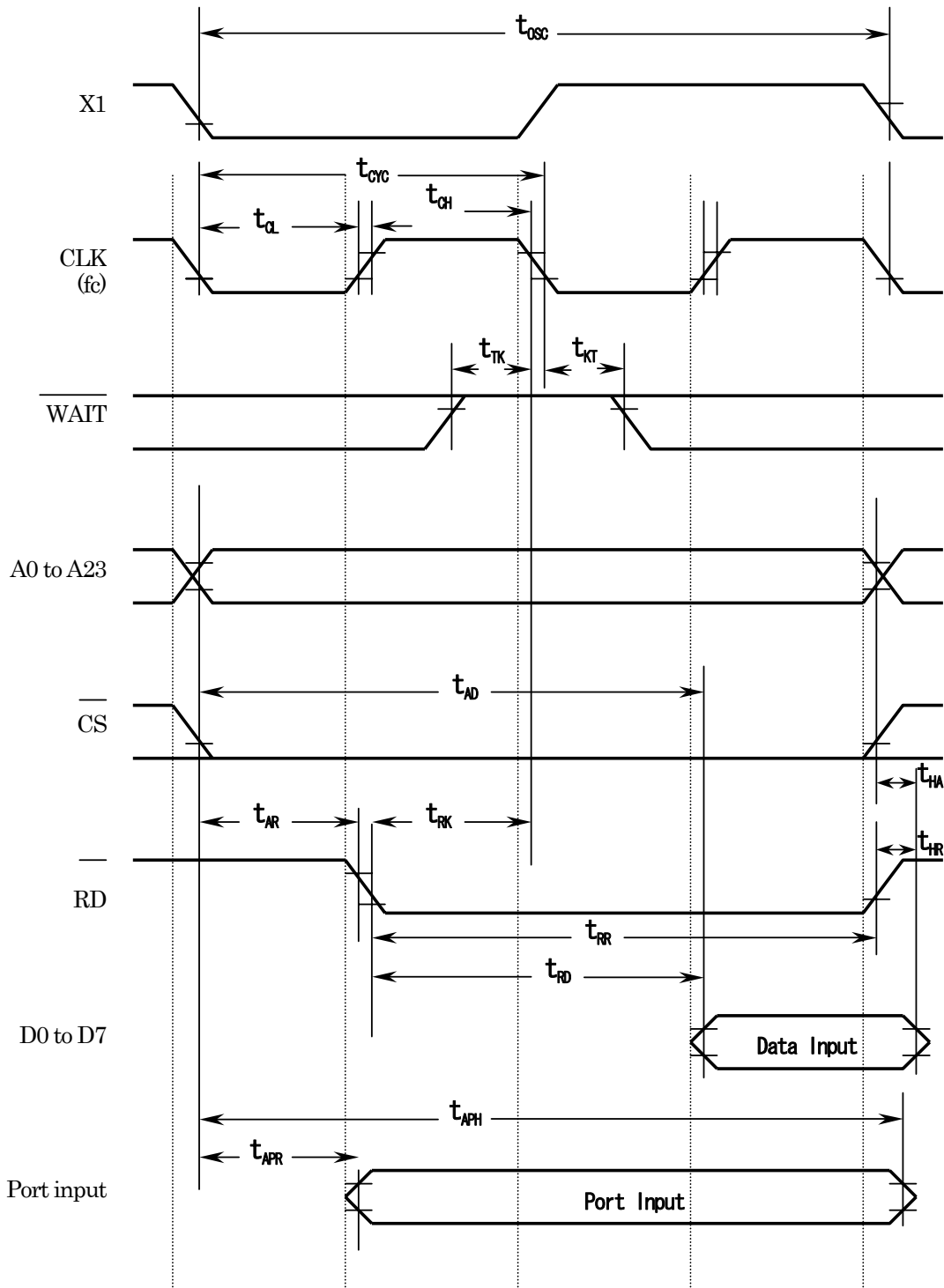
VCC5=5.0V±5%, TA=-40 to 85 degree C

No.	Parameter	Symbol	Min	Max	@20MHz	@16MHz	Unit
1	OSC period (X1/X2)	$t_{OSC}$	100	125	100	125	ns
2	System Clock period (=T)	$t_{CYC}$	50	62.5	50	62.5	ns
3	CLK Low Width	$t_{CL}$	$0.5 \times T-15$		10	16	ns
4	CLK High Width	$t_{CH}$	$0.5 \times T-15$		10	16	ns
5-1	D0 to D7 Valid → WR Rise @0WAIT	$t_{DW}$	$1.25 \times T-35$		28	43	ns
5-2	D0 to D7 Valid → WR Rise @1WAIT	$t_{DW3}$	$2.25 \times T-35$		78	106	ns
6-1	WR Low Width @0WAIT	$t_{WW}$	$1.25 \times T-30$		33	48	ns
6-2	WR Low Width @1WAIT	$t_{WW3}$	$2.25 \times T-30$		83	111	ns
7	A0 to A23 Valid → WR Fall	$t_{AW}$	$0.5 \times T-20$		5	11	ns
8	WR Fall → CLK Fall	$t_{WK}$	$0.5 \times T-20$		5	11	ns
9	WR Fall → A0 to A23 Hold	$t_{WA}$	$0.25 \times T-5$		8	11	ns
10	WR Fall → D0 to D7 Hold	$t_{WD}$	$0.25 \times T-5$		8	11	ns
11	A0 to A23 Valid → PORT Output	$t_{APW}$		$2.0 \times T+70$	170	195	ns
12	WAIT Set-up Time	$t_{TK}$	15		15	15	ns
13	WAIT Hold Time	$t_{KT}$	5		5	5	ns
14	RD Rise → D0 to D7 Output	$t_{RDO}$	$1.25 \times T-35$		20	26	ns

## AC Condition

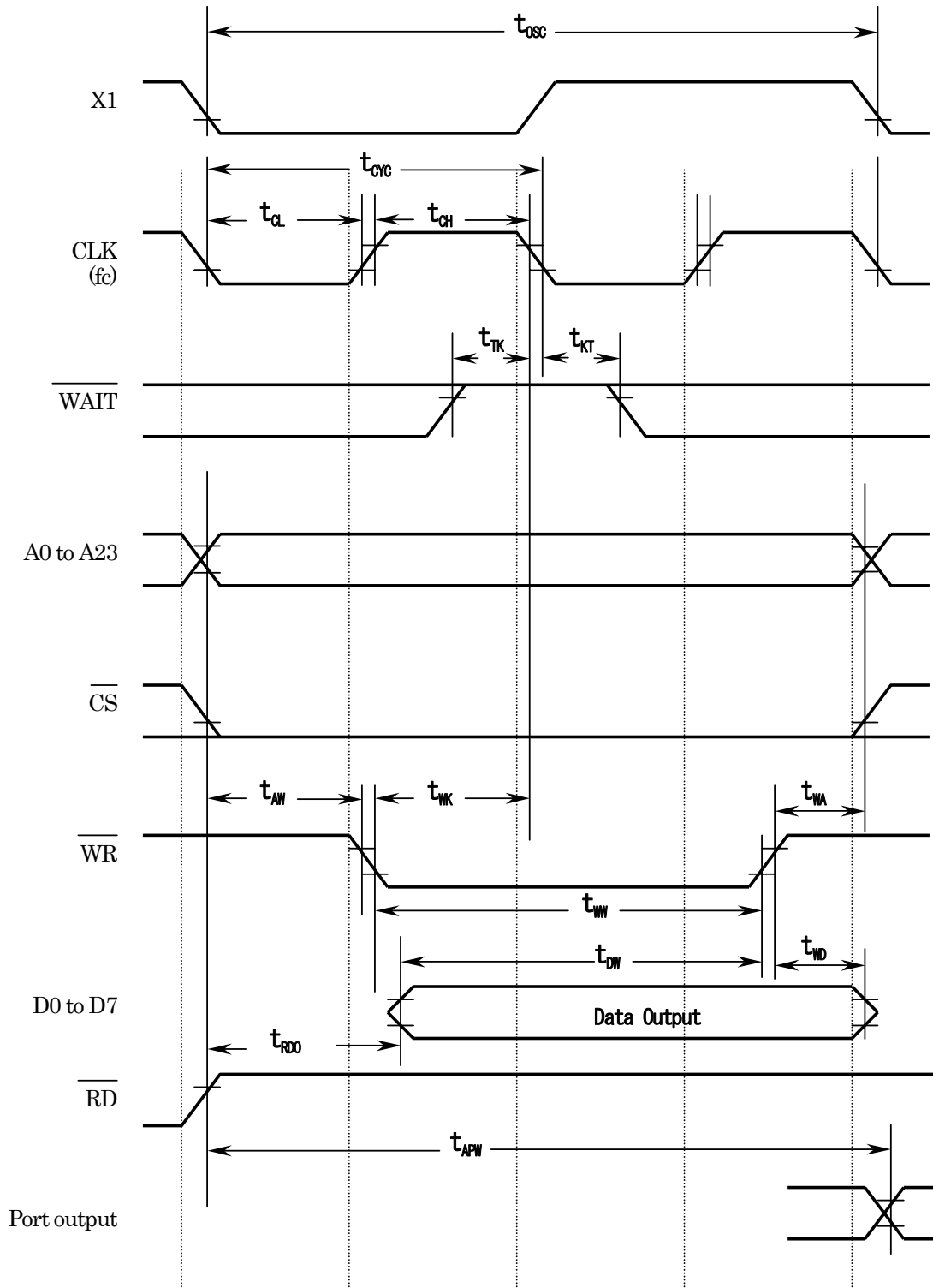
- Output : D0 to D7, A0 to A7, A8 to A15, A16 to A23, RD, WR  
High 2.0V, Low 0.8V, CL=50pF  
Others
- Input : D0 to D7  
High 2.0V, Low 0.8V, CL=50pF  
Others  
High  $0.8 \times VCC5$ , Low  $0.2 \times VCC5$

## (1) Read cycle (0 wait)



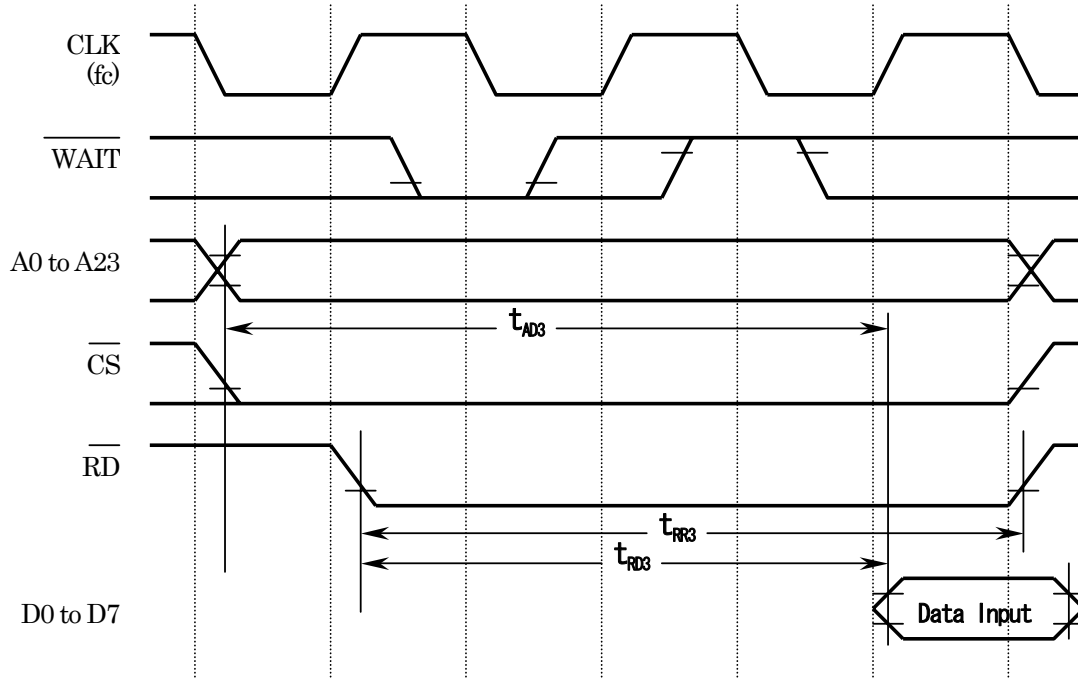
Note : The phase relation between X1 input signal and the other signals is unsettled.  
The timing chart above is an example.

## (2) Write cycle (0 wait)

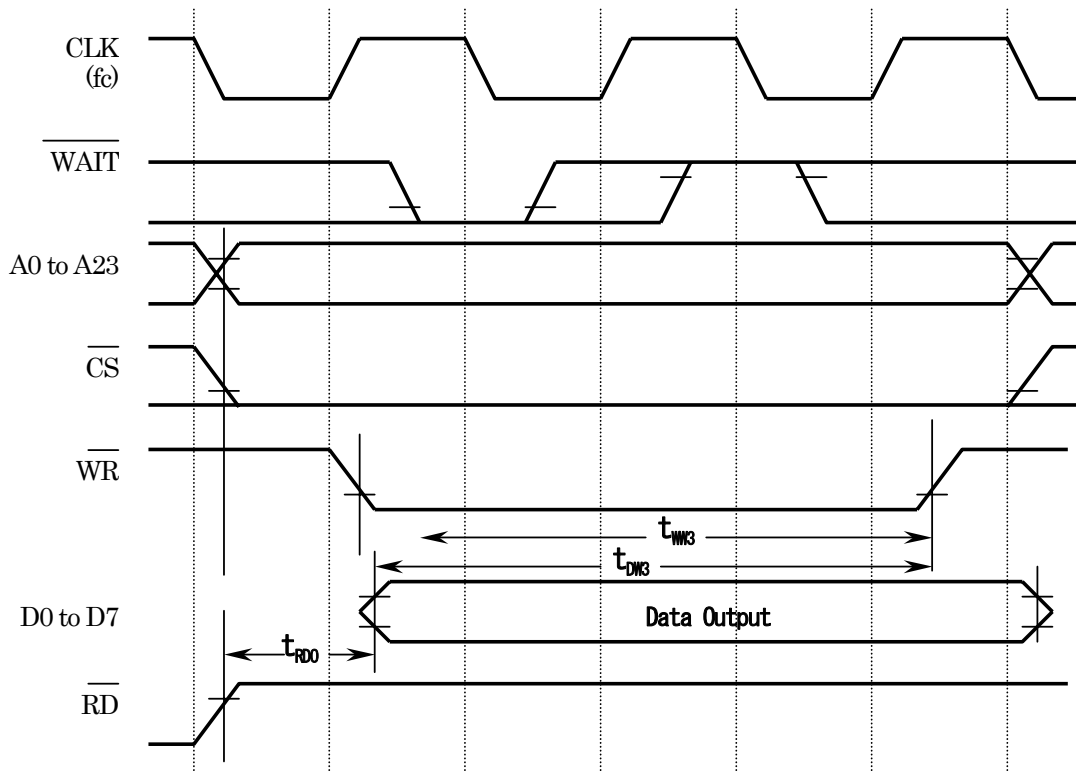


Note : The phase relation between X1 input signal and the other signals is unsettled.  
The timing chart above is an example.

(3) Read cycle (1 wait)



(4) Write cycle (1 wait)



## 5.4 AD Conversion Characteristics

Symbol	Parameter	Min	Typ	MAX	Unit
VREFH	Analog reference voltage(+)	VCC5-0.2	VCC5	VCC5	V
VREFL	Analog reference voltage(-)	VSS5	VSS5	VSS5	
AVCC	AD Converter Power Supply Voltage	VCC5-0.2	VCC5	VCC5	
AVSS	AD Converter Ground	VSS5	VSS5	VSS5	
AVIN	Analog Input Voltage	VREFL		VREFH	
IREF	Analog Current for analog reference voltage <VREFON>=1		0.8	1.2	mA
	<VREFON>=0		0.02	5	uA
ET	Total error (excluding quantize error)			±3.0	LSB

Note) "LSB" is the UNIT which means the resolution of AD CONVERTER. ( $\pm 3 \text{ LSB} = 3 * \text{VCC}/1024 = \pm 15\text{mV}$ )

## 5.5 Event Counter (TI0, TI4, TI8, TI9, TIA, TIB)

Parameter	Symbol	Variable		20MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
Clock Cycle	t <sub>CK</sub>	8T+100		500		600		ns
Clock Low Width	t <sub>CKL</sub>	4T+40		240		290		ns
Clock High Width	t <sub>CKH</sub>	4T+40		240		290		ns

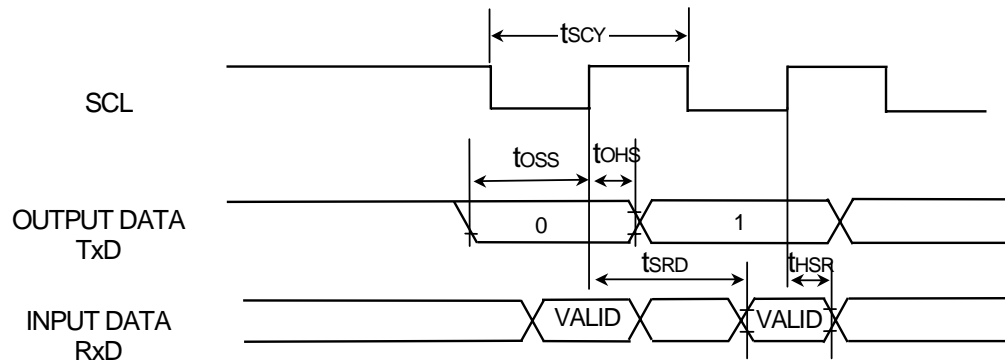
## 5.6 Serial Channel Timing

## (1) SCLK Input mode (I/O Interface mode)

Parameter	Symbol	Variable		20MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
SCLK Cycle	t <sub>SCY</sub>	16T		0.8		1.0		us
Output Data → SCLK Rise	t <sub>OSS</sub>	t <sub>SCY</sub> /2-4T -110		90		140		ns
SCLK Rise → Output Data Hold	t <sub>OHS</sub>	t <sub>SCY</sub> /2+2T		500		625		
SCLK Rise → Input Data Hold	t <sub>HSR</sub>	3T+10		160		197.5		
SCLK Rise → Input Data Valid	t <sub>SRD</sub>		t <sub>SCY</sub>		800		1000	

## (2) SCLK Output mode (I/O Interface mode)

Parameter	Symbol	Variable		20MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
SCLK Cycle (programmable)	t <sub>SCY</sub>	16T	8192T	0.8	409.6	1.0	512	us
Output Data → SCLK Rise	t <sub>OSS</sub>	t <sub>SCY</sub> /2-40		360		460		ns
SCLK Rise → Output Data Hold	t <sub>OHS</sub>	t <sub>SCY</sub> /2-40		360		460		
SCLK Rise → Input Data Hold	t <sub>HSR</sub>	0		0		0		
SCLK Rise → Input Data Valid	t <sub>SRD</sub>		t <sub>SCY</sub> /2-T -180		570		757.5	



www.DataSheet4U.com

## (3) SCLK Input mode (UART mode) (Preliminary)

Parameter	Symbol	Variable		20MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
SCLK Cycle	$T_{SCY}$	$4T + 20$		220		270		ns
SCLK Low level Pulse width	$T_{SCYL}$	$2T + 5$		105		130		
SCLK High level Pulse width	$T_{SCYH}$	$2T + 5$		105		130		

## 5.7 Interrupt Operation

Parameter	Symbol	Variable		20MHz		16MHz		Unit
		Min	Max	Min	Max	Min	Max	
NMI,INT0 Low Width	$T_{INTAL}$	$4T$		200		250		ns
NMI,INT0 High Width	$T_{INTAH}$	$4T$		200		250		
WUINT0 to WUINT7, INT1 to INT7 Low Width	$T_{INTBL}$	$8T+100$		500		600		
WUINT0 to WUINT7, INT1 to INT7 High Width	$T_{INTBH}$	$8T+100$		500		600		

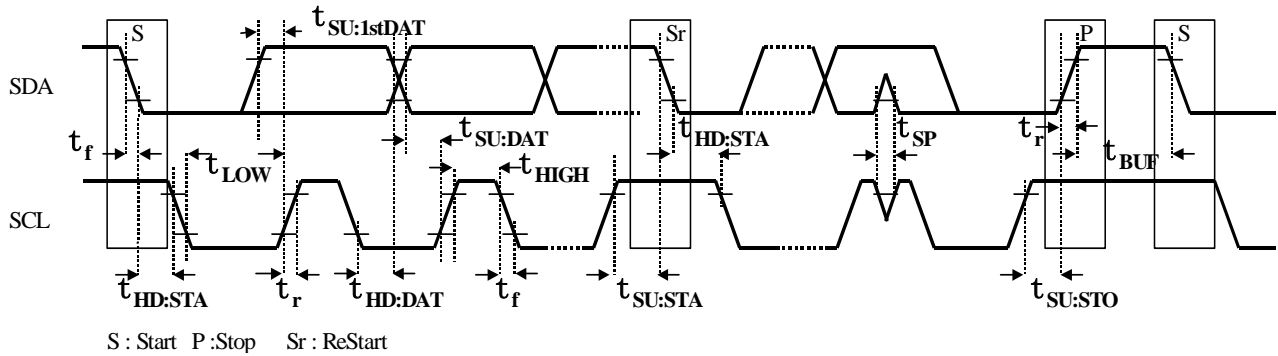
## 5.8 Serial bus interface

I2CBUS-AC-SPEC TABLE

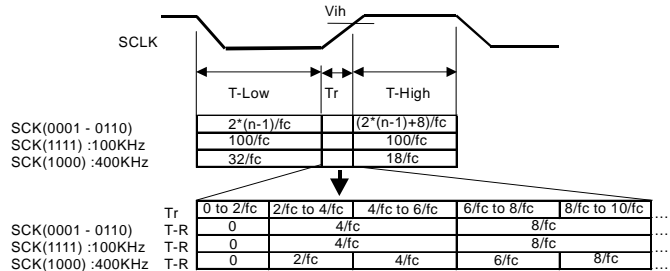
No	PARAMETER	SYMBOL	UNIT	(fc=20MHz)				(fc=System clock)	
				400KHz		100KHz		Existing rate	
				MIN	MAX	MIN	MAX	MIN	MAX
1	SCL clock frequency	f <sub>scl</sub>	KHz	0	400	0	100	0	fc/(2 <sup>n</sup> +8)
2	Hold time (repeated) START condition. After this period, the first clock pulse is generated.	t <sub>HD:STA</sub>	ns	650	-	4500	-	2 <sup>n-1</sup> /fc	-
3	LOW period of the SCL clock	t <sub>LOW</sub>	ns	1300	-	4700	-	2 <sup>n-1</sup> /fc	-
4	HIGH period of the SCL clock	t <sub>HIGH</sub>	ns	600	-	4000	-	(2 <sup>n-1</sup> +8)/fc	-
5	Set-up time for a repeated START condition	t <sub>SU:STA</sub>	ns	by software		by software		by software	
6	Data hold time: for CBUS compatible masters for I2C-bus devices	t <sub>HD:DAT</sub>	ns	0	900	0	3450	0	6/fc
7	Data set-up time	t <sub>SU:DAT</sub>	ns	100	-	250	-	(2 <sup>n-1</sup> -6)/fc	-
7	Data set-up time (The case in the first bit after transfer)	t <sub>SU:1stDAT</sub>	↑	↑	↑	↑	↑	(2 <sup>n-1</sup> -12)/fc	-
8	Rise time of both SDA and ACL signals (*1)	t <sub>r</sub>	ns	-	300 (receive)	-	1000 (receive)	-	
9	Fall time of both SDA and ACL signals	t <sub>f</sub>	ns	-	300	-	300	-	
10	Set-up time for STOP condition	t <sub>SU:STO</sub>	ns	950	-	4200	-	2 <sup>n-1</sup> +12/fc	-
11	Bus free time between a STOP and START condition	t <sub>BUF</sub>	ns	by software		by software		by software	
12	Capacitive load for each bus line	C <sub>b</sub>	pF	400		400		400	
13	Noise margin at the LOW level for each connected device (including hysteresis)	V <sub>nL</sub>	v	0.2V <sub>DD5</sub>	-	0.2V <sub>DD5</sub>	-	0.2V <sub>DD5</sub>	-
14	Noise margin at the HIGH level for each connected device (including hysteresis)	V <sub>nH</sub>	v	0.2V <sub>DD5</sub>	-	0.2V <sub>DD5</sub>	-	0.2V <sub>DD5</sub>	-
15	Pulse width of spikes which must be suppressed by the input filter	t <sub>sp</sub>	ns	0	50	n/a	n/a	n/a	n/a

## Note

1 All values referred to V<sub>IHmin</sub> and V<sub>ILmax</sub> levels.



## \*1) I2BUS CLK AC SPEC : Tr (Transmitter selection)



## T-period = T-Low + T-R + T-High

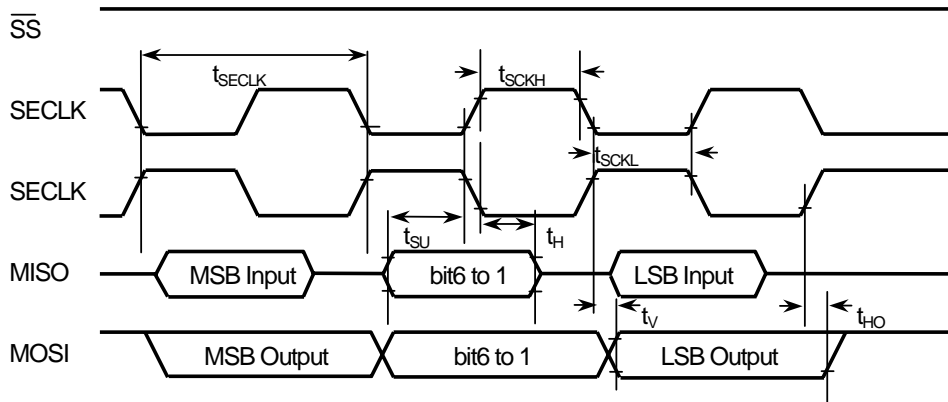
Example: in the case of fc=20MHz, SCK3,2,1,0=(0001), Tr=200ns

- 1) Tr=200ns so T-R=4/fc
- 2) T-period = 2\*(n-1)/fc + 4/fc + (2\*(n-1)+8)/fc = 76/fc = 3.8us

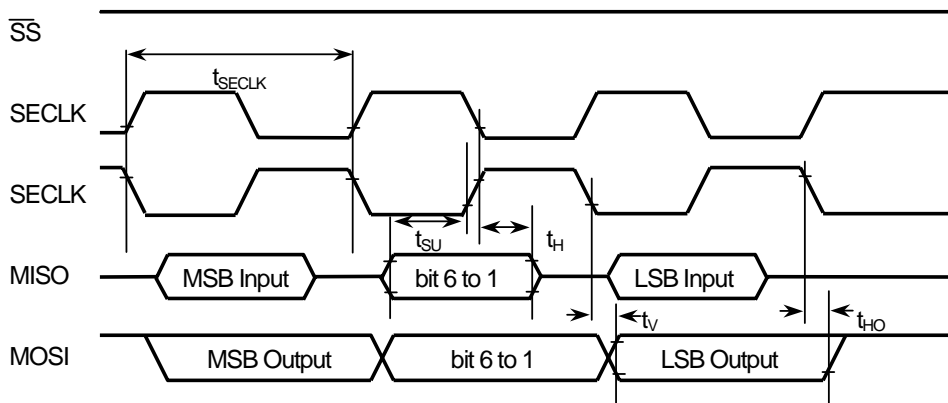
## 5.9 Serial Expansion Interface

Symbol	Parameter	Variable		20MHz		Unit
		Min	Max	Min	Max	
$t_{SECLK}$	SECLK Cycle	5T	40T	250	2000	ns
$t_{LEAD}$	SS fall $\rightarrow$ SECLK	4T		200		ns
$t_{LAG}$	SECLK $\rightarrow$ SS rise	4T		200		ns
$t_{SCKH}$	SECLK High Pulse Width	$t_{SECLK}/2-9$		116		ns
$t_{SCKL}$	SECLK Low Pulse Width	$t_{SECLK}/2-9$		116		ns
$t_{SU}$	Input Data Set-up	$t_{SECLK}/4-10$		52		ns
$t_H$	Input Data Hold	$t_{SECLK}/4$		62		ns
$t_V$	Output Data Valid		$t_{SECLK}/4$		62	ns
$t_{HO}$	Output Data Hold	0		0		ns

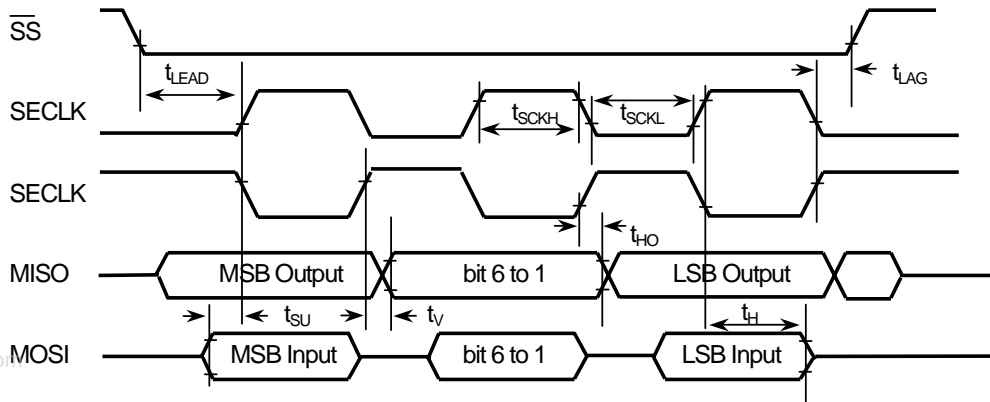
a) SEI Master (CPHA=0)



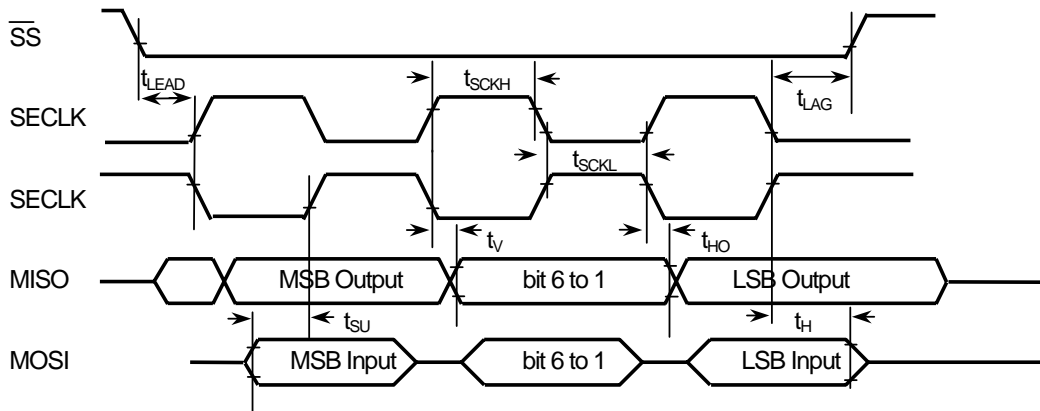
b) SEI Master (CPHA=1)



c) SEI Slave (CPHA=0)

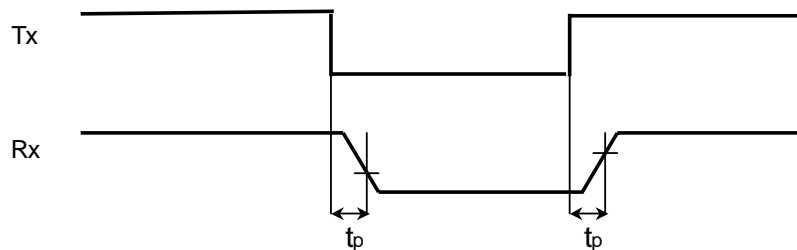


d) SEI Slave (CPHA=1)



## 5.10 Controller Area Network (CAN)

Symbol	Parameter	Variable		20MHz		Unit
		Min	Max	Min	Max	
$t_{clk}$	CAN Clock period	2T		100		ns
$t_p$	Tx edge $\rightarrow$ Rx Input		$2t_{clk}-20$		180	ns



## 5.11 Voltage regulator

## Voltage Regulator

Vcc5 = 4.5V to 5.25V / fc = 16 to 20MHz / Ta = -40 to 85 degree C

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit.
Output Voltage	REGOUT		3.0	—	3.6	V
Output Current	Iro	Vin-REGOUT=1.0V	0	—	150	mA
Quiescent Current	Iq	Iro ≤ 10 uA	30	50	100	μ A
	Iq1	10 uA < Iro < 100mA (Ta=25°C)	15	250	800	μ A
	Iop	Iro=150mA	6	8	10	mA
Standby Current	Is	REGEN=0 (Regulator Only)	—	0.1	0.2	μ A

0.5[Ohm] ≤ ESR ≤ 5.0[Ohm]

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit.
Stabilization capacitor	Cs	Cb=10uF, ESR=4.7 Ω	0.1	—	10	μ F
Bypass capacitor	Cb	Cs=10uF, ESR=4.7 Ω (Cs>=Cb)	0.1	—	10	μ F
Input capacitor	Cin (Note)	Cs=10uF, ESR=4.7 Ω	4.7	—	22	μ F
Equivalent Series Resistor	ESR	Cs=10uF Cb=0.1uF	0.5	—	5	Ω

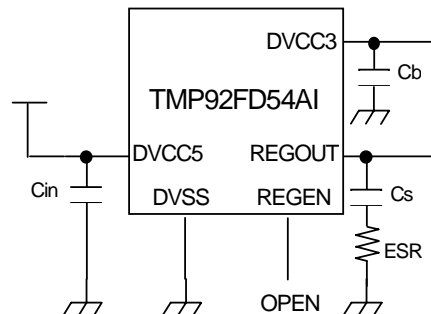
0.5[Ohm] ≤ ESR ≤ 50[Ohm]

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit.
Stabilization capacitor	Cs	Cb=0.6uF, ESR=47 Ω	0.1	—	10	μ F
Bypass capacitor	Cb	Cs=10uF, ESR=47 Ω (Cs>=Cb)	0.6	—	10	μ F
Input capacitor	Cin (Note)	Cs=10uF, ESR=47 Ω	4.7	—	22	μ F
Equivalent Series Resistor	ESR	Cs=10uF Cb=0.6uF	0.5	—	50	Ω

0.5[Ohm] ≤ ESR ≤ 100[Ohm]

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit.
Stabilization capacitor	Cs	Cb=1.0uF, ESR=100 Ω	0.1	—	10	μ F
Bypass capacitor	Cb	Cs=10uF, ESR=100 Ω (Cs>=Cb)	1.0	—	10	μ F
Input capacitor	Cin (Note)	Cs=10uF, ESR=100 Ω	4.7	—	22	μ F
Equivalent Series Resistor	ESR	Cs=10uF Cb=1.0uF	0.5	—	100	Ω

Note: Recommend Tantalum Capacitor.



## 6. Points to Note and Restrictions

### 6.1 Notation

- (1) The notation for built-in I/O registers is as follows register symbol <Bit symbol>

Example: TRUN01<T0RUN> denotes bit T0RUN of register TRUN01.

- (2) Read-modify-write instructions (RMW)

An instruction in which the CPU reads data from memory and writes the data to the same memory location in one instruction.

Example 1: SET 3, (TRUN01); Set bit3 of TRUN01.

Example 2: INC 1, (400H); Increment the data at 400H.

- Examples of read-modify-write instructions on the TLCS-900/H1

Exchange instruction

EX (mem), R

Arithmetic operations

ADD (mem), R/#      ADC (mem), R/#

SUB (mem), R/#      SBC (mem), R/#

INC #3, (mem)      DEC #3, (mem)

Logic operations

AND (mem), R/#      OR (mem), R/#

XOR (mem), R/#

Bit manipulation operations

STCF #3/A, (mem)      RES #3, (mem)

SET #3, (mem)      CHG #3, (mem)

TSET #3, (mem)

Rotate and shift operations

RLC (mem)      RRC (mem)

RL (mem)      RR (mem)

SLA (mem)      SRA (mem)

SLL (mem)      SRL (mem)

RLD (mem)      RRD (mem)

## 6.2 Points to Note

### (1) Watchdog timer

The watchdog timer starts operation immediately after a reset is released. When the watchdog timer is not to be used, disable it.

### (2) The stable time of the internal clock

When releasing the external reset using “built-in clock doubler” until the internal reset is released, the requiring time to stabilize the circuit is automatically set. See section 3.1.2 “Reset Operation” for details. Also when releasing standby mode in STOP mode using an interrupt until the internal circuit starts the operation, the stable time of the oscillator is automatically input. See section 3.4 “Standby Function (3) STOP mode” for details.

### (3) Undefined bit in the built-in I/O register

When reading the undefined bit in the built-in I/O register, the undefined value is output.

Thus, when creating program, it should not be depending on this bit condition.

### (4) Reserved address areas

The 16 bytes area (FFFFFF0H to FFFFFFFH) cannot be used for it is reserved as internal area. If using emulator, optional 64 Kbytes of 16M bytes area are used for control emulator. Therefore, if using emulator, its area cannot be used.

### (5) POP SR instruction

Execute the POP SR instruction during DI condition.

## Package Dimensions : P-LQFP100-1414-0.50C

Technical drawing of a square microchip package. The drawing includes three views: a top view, a side view, and a detail view of the lead profile.

**Top View Dimensions:**

- Overall width:  $16.0 \pm 0.2$
- Inner width:  $14.0 \pm 0.2$
- Overall height:  $16.0 \pm 0.2$
- Inner height:  $14.0 \pm 0.2$
- Pin pitch (top): 75
- Pin pitch (right): 51
- Pin pitch (bottom): 25
- Pin pitch (left): 100
- Pin pitch (left, inner): 76
- Pin pitch (right, inner): 50
- Pin pitch (bottom, inner): 26
- Pin pitch (left, inner): 1
- Pin pitch (right, inner): 25
- Pin pitch (bottom, inner): 1
- Pin pitch (left, inner): 1

**Side View Dimensions:**

- Overall width:  $15.0 \pm 0.2$
- Lead height:  $1.4 \pm 0.2$
- Lead thickness:  $0.1$
- Lead thickness (max):  $1.85 \text{ MAX}$
- Lead thickness (min):  $0.1$
- Lead thickness (max):  $0.15$
- Lead thickness (min):  $-0.1$

**Detail View Dimensions:**

- Lead thickness:  $0.125$
- Lead thickness (max):  $+0.1$
- Lead thickness (min):  $-0.05$
- Lead angle:  $0 \sim 10^\circ$
- Lead width:  $0.5 \pm 0.2$

**Callouts:**

- Top view:  $0.2 \pm 0.1$  (pin pitch),  $0.08$  (pin width),  $M$  (material)
- Side view:  $0.08$  (lead width)
- Detail view:  $0.08$  (lead width)