Toshiba Semiconductor and Storage - TMP91FW27UG(C,JZ) Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	900/L1
Core Size	16-Bit
Speed	27MHz
Connectivity	EBI/EMI, I²C, IrDA, UART/USART
Peripherals	DMA, WDT
Number of I/O	53
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	12K x 8
Voltage - Supply (Vcc/Vdd)	2.2V ~ 3.6V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/toshiba-semiconductor-and-storage/tmp91fw27ug-c-jz

Preface

Thank you very much for making use of Toshiba microcomputer LSI. Before using this LSI, refer to section “Points of Note and Restrictions”.

Especially, take care below cautions.

****CAUTION****

How to release the HALT mode

Usually, interrupts can release all halts stats. However, the interrupts = ($\overline{\text{NMI}}$, INT0, INTRTC), which can release the HALT mode may not be able to do so if they are input during the period CPU is shifting to the HALT mode (for about 5 clocks of f_{FPH}) with IDLE1 or STOP mode (IDLE2 is not applicable to this case). (In this case, an interupt request is kept on hold internally.)

If another interupt is generated after it has shifted to HALT mode completely, halt status can be released without difficultly. The priority of this interrupt is compare with that of the interrupt kept on hold internally, and the interrupt with higher priority is handled first followed by the other interrupt.

3.2.4 Single Boot Mode

In Single Boot mode, the internal boot ROM (mask ROM) is activated to transfer a program/erase routine (user-created boot program) from an external source into the internal RAM. This program/erase routine is then used to program/erase the flash memory. 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 3.2.3).

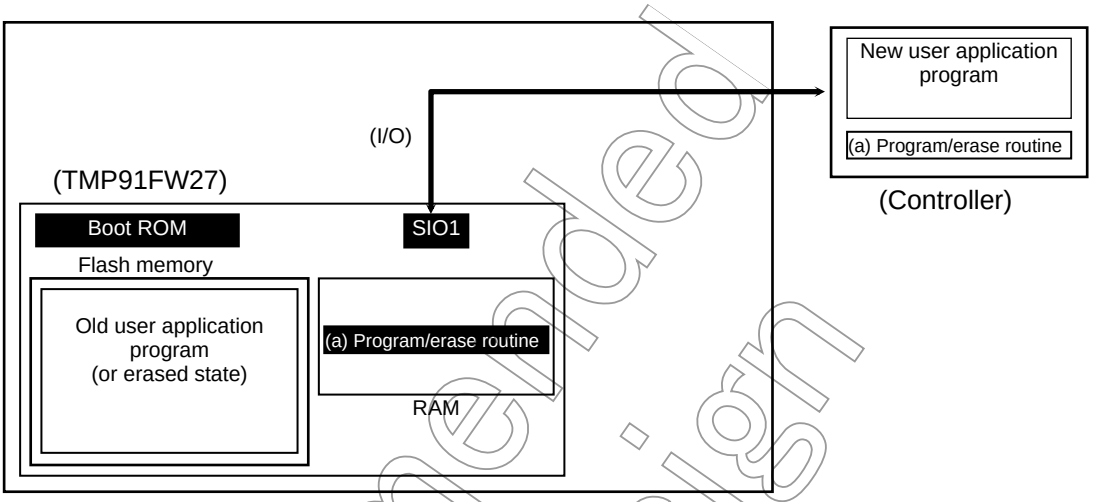
The device's SIO (SIO1) and the controller are connected to transfer the program/erase routine from the controller to the device's internal RAM. This program/erase routine is then executed to program/erase the flash memory.

The program/erase routine is executed by sending commands and write data from the controller. The communications protocol between the device and the controller is described later in this manual. Before the program/erase routine can be transferred to the RAM, user password verification is performed to ensure the security of user ROM data. If the password is not verified correctly, the RAM transfer operation cannot be performed. In Single Boot mode, disable interrupts and use the interrupt request flags to check for an interrupt request.

Note: In Single Boot mode, the boot-ROM programs are executed in Normal mode. Do not change to another operation mode in the program/erase routine.

(Step-3) Copying the program/erase routine to the RAM

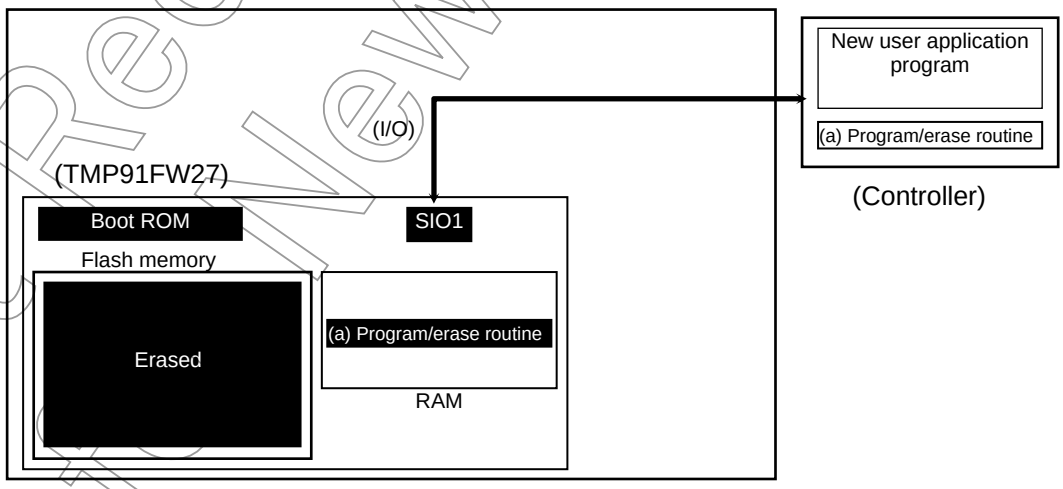
After password verification is completed, the boot ROM copies the program/erase routine (a) from the controller to the RAM using serial communications. The program/erase routine must be stored within the RAM address range of 001000H to 003DFFH.



(Step-4) Executing the program/erase routine in the RAM

Control jumps to the program/erase routine (a) in the RAM. If necessary, the old user application program is erased (sector erase or chip erase).

Note: The boot ROM is provided with an erase command, which enables the entire chip to be erased from the controller without using the program/erase routine. If it is necessary to erase data on a sector basis, incorporate the necessary code in the program/erase routine.



3.2.4.5 Interface Specifications

The SIO communications format in Single Boot mode is shown below. The device supports the UART (asynchronous communications) serial operation mode.

To perform on-board programming, the same communications format must also be set on the programming controller's side.

● UART (asynchronous) communications

- Communications channel : SIO channel 1 (For the pins to be used, see Table 3.2.4.)
- Serial transfer mode : UART (asynchronous communications) mode
- Data length : 8 bits
- Parity bit : None
- Stop bit : 1 bit
- Baud rate : See Table 3.2.5 and Table 3.2.6.

Table 3.2.4 Pin Connections

Pins		UART
Power supply pins	DVCC	○
	DVSS	○
Mode setting pins	AM1,AM0, $\overline{BOOT}$	○
Reset pin	$\overline{RESET}$	○
Communications pins	TXD1	○
	RXD1	○

Note: Unused pins are in the initial state after reset release.

Table 3.2.5 Baud Rate Table

SIO	Transfer Rate (bps)				
UART	115200	57600	38400	19200	9600

3.2.4.7 Boot Program

When the device starts up in Single Boot mode, the boot program is activated.

The following explains the commands that are used in the boot program to communicate with the controller when the device starts up in Single Boot mode. Use this information for creating a controller for using Single Boot mode or for building a user boot environment.

1. RAM Transfer command

   In RAM transfer, data is transferred from the controller and stored in the device's internal RAM. When the transfer completes normally, the boot program will start running the transferred user program. Up to 11.5 Kbytes of data can be transferred as a user program. (This limit is implemented in the boot program to protect the stack pointer area.) The user program starts executing from the RAM storage start address.
   This RAM transfer function enables a user-created program/erase routine to be executed, allowing the user to implement their own on-board programming method. To perform on-board programming with a user program, the flash memory command sequences (see section 3.2.6) must be used. After the RAM Transfer command has been completed, the entire internal RAM area can be used.
   If read protection or write protection is applied on the device or a password error occurs, this command will not be executed.

2. Flash Memory SUM command

   This command calculates the SUM of 128 Kbytes of data in the flash memory and returns the result. There is no operation command available to the boot program for reading data from the entire area of the flash memory. Instead, this Flash Memory SUM command can be used. Reading the SUM value enables revision management of the application program.

3. Product Information Read command

   This command returns the information about the device including its part number and memory details stored in the flash memory at addresses 02FEF0H to 02FEF3H. This command can also be used for revision management of the application program.

4. Flash Memory Chip Erase command

   This command erases all the sectors in the flash memory. If read protection or write protection is applied on the device, all the sectors in the flash memory are erased and the read protection or write protection is cleared.
   Since this command is also used to restore the operation of the boot program when the password is forgotten, it does not include password verification.

5. Flash Memory Protect Set command

   This command sets both read protection and write protection on the device. However, if a password error occurs, this command will not be executed.
   When read protection is set, the flash memory cannot be read in Programmer mode. When write protection is set, the flash memory cannot be written in Programmer mode.

10. From the controller to the device

    The data in the 25th byte is CHECKSUM data. The CHECKSUM data sent by the controller is the two's complement of the lower 8-bit value obtained by summing the data in the 19th to 24th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.2.4.17 "How to Calculate CHECKSUM ."

    Note: The data in the 19th to 25th bytes should be placed within addresses 001000H to 003DFFH (11.5 Kbytes) in the internal RAM.

11. From the device to the controller

    The data in the 26th byte is the ACK response data to the data in the 19th to 25th bytes (ACK response to the CHECKSUM value).

    The device first checks to see whether the data received in the 19th to 25th bytes contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 18H and waits for the next operation command (3rd byte). The upper four bits of the ACK response data are the upper four bits of the immediately preceding operation command data, so the value of these bits is "1".

    Next, the device checks the CHECKSUM data in the 25th byte. This check is made to see if the lower 8-bit value obtained by summing the data in the 19th to 25th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 11H and waits for the next operation command data (3rd byte).

12. From the controller to the device

    The data in the 27th to m'th bytes is the data to be stored in the RAM. This data is written to the RAM starting at the address specified in the 19th to 22nd bytes. The number of bytes to be written is specified in the 23rd and 24th bytes.

13. From the controller to the device

    The data in the (m+1)th byte is CHECKSUM data. The CHECKSUM data sent by the controller is the two's complement of the lower 8-bit value obtained by summing the data in the 27th to m'th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.2.4.17 "How to Calculate CHECKSUM."

3.2.4.9 Flash Memory SUM command (See Table 3.2.9)

1. The data in the 1st and 2nd bytes is the same as in the case of the RAM Transfer command.

2. From the controller to the device

   The data in the 3rd byte is operation command data. The Flash Memory SUM command data (20H) is sent here.

3. From the device to the controller

   The data in the 4th byte is the ACK response data to the operation command data in the 3rd byte.
   The device first checks to see if the data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)
   Then, if the data in the 3rd byte corresponds to one of the operation command values given in Table 3.2.7, the device echoes back the received data (ACK response for normal reception). In this case, 20H is echoed back and execution then branches to the flash memory SUM processing routine. If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

4. From the device to the controller

   The data in the 5th and 6th bytes is the upper and lower data of the SUM value, respectively. For details on SUM, see 3.2.4.16 “How to Calculate SUM .”

5. From the device to the controller

   The data in the 7th byte is CHECKSUM data. This is the two’s complement of the lower 8-bit value obtained by summing the data in the 5th and 6th bytes by unsigned 8-bit addition (ignoring any overflow).

6. From the controller to the device

   The data in the 8th byte is the next operation command data.

9. From the device to the controller

   The data in the 33rd to 36th bytes is the RAM end address. FFH, 3FH, 00H and 00H are sent starting from the 33rd byte.

10. From the device to the controller

    The data in the 37th to 44th bytes is dummy data.

11. From the device to the controller

    The data in the 45th and 46th bytes contains the protection status and sector division information of the flash memory.
    - Bit 0 indicates the read protection status.
      - 0: Read protection is applied.
      - 1: Read protection is not applied.
    - Bit 1 indicates the write protection status.
      - 0: Write protection is applied.
      - 1: Write protection is not applied.
    - Bit 2 indicates whether or not the flash memory is divided into sectors.
      - 0: The flash memory is divided into sectors.
      - 1: The flash memory is not divided into sectors.
    - Bits 3 to 15 are sent as "0".

12. From the device to the controller

    The data in the 47th to 50th bytes is the flash memory start address. 00H, 00H, 01H and 00H are sent starting from the 47th byte.

13. From the device to the controller

    The data in the 51st to 54th bytes is the flash memory end address. FFH, FFH, 02H and 00H are sent starting from the 51st byte.

14. From the device to the controller

    The data in the 55th and 56th bytes indicates the number of sectors in the flash memory. 20H and 00H are sent starting from the 55th byte.

15. From the device to the controller

    The data in the 57th to 65th bytes contains sector information of the flash memory. Sector information is comprised of the start address (starting from the flash memory start address), sector size and number of consecutive sectors of the same size. Note that the sector size is represented in word units.
    The data in the 57th to 65th bytes indicates 4 Kbytes of sectors (sector 0 to sector 31).
    For the data to be transferred, see Table 3.2.10 and Table 3.2.11.

16. From the device to the controller

    The data in the 66th byte is CHECKSUM data. This is the two's complement of the lower 8-bit value obtained by summing the data in the 5th to 65th bytes by unsigned 8-bit addition (ignoring any overflow).

17. From the controller to the device

    The data in the 67th byte is the next operation command data.

6. From the device to the controller

   The data in the 7th byte indicates whether or not the erase operation has completed successfully. If the erase operation has completed successfully, the device returns the end code (4FH). If an erase error has occurred, the device returns the error code (4CH).

7. From the device to the controller

   The data in the 8th byte is ACK response data. If the erase operation has completed successfully, the device returns the ACK response for erase completion (5DH). If an erase error has occurred, the device returns the ACK response for erase error (60H).

8. From the controller to the device

   The data in the 9th byte is the next operation command data.

6. From the device to the controller

   The data in the 18th byte is the ACK response data to the data in the 5th to 17th bytes (ACK response to the CHECKSUM value).
   The device first checks to see whether the data in the 5th to 17th bytes contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 68H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are the upper four bits of the immediately preceding operation command data, so the value of these bits is "6".
   Then, the device checks the CHECKSUM data in the 17th byte. This check is made to see if the lower 8 bits of the value obtained by summing the data in the 5th to 17th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 61H and waits for the next operation command data (3rd byte).
   Finally, the device examines the result of password verification. If all the data in the 5th to 16th bytes is not verified correctly, the device returns the ACK response data for password error (bit 0) 61H and waits for the next operation command data (3rd byte).
   If no error is found in the above checks, the device returns the ACK response data for normal reception 60H.

7. From the device to the controller

   The data in the 19th byte indicates whether or not the protect set operation has completed successfully. If the operation has completed successfully, the device returns the end code (6FH). If an error has occurred, the device returns the error code (6CH).

8. From the device to the controller

   The data in the 20th byte is ACK response data. If the protect set operation has completed successfully, the device returns the ACK response data for normal completion (31H). If an error has occurred, the device returns the ACK response data for error (34H).

9. From the device to the controller

   The data in the 21st byte is the next operation command data.

3.2.4.14 Determining Serial Operation Mode

To communicate in UART mode, the controller should transmit the data value 86H as the first byte at the desired baud rate. Figure 3.2.7 shows the waveform of this operation.

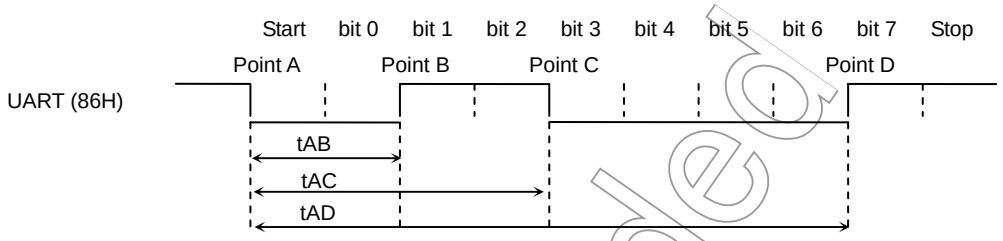


Figure 3.2.7 Data for Determining Serial Operation Mode

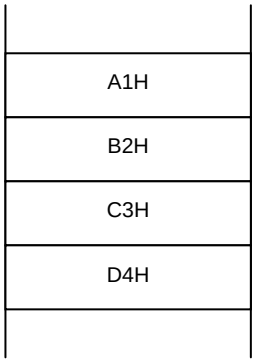
The boot program receives the first byte (86H) after reset release not as serial communications data. Instead, the boot program uses the first byte to determine the baud rate. The baud rate is determined by the output periods of tAB, tAC and tAD as shown in Figure 3.2.7 using the procedure shown in Figure 3.2.8.

The CPU monitors the level of the receive pin. Upon detecting a level change, the CPU captures the timer value to determine the baud rate.

3.2.4.16 How to Calculate SUM

SUM is calculated by summing the values of all data read from the flash memory by unsigned 8-bit addition and is returned as a word (16-bit) value. The resulting SUM value is sent to the controller in order of upper 8 bits and lower 8 bits. All the 128 Kbytes of data in the flash memory are included in the calculation of SUM. When the Flash Memory SUM command is executed, SUM is calculated in this way.

Example:



When SUM is calculated from the four data entries shown to the left, the result is as follows:

A1H + B2H + C3H + D4H = 02EAH
SUM upper 8 bits: 02H
SUM lower 8 bits: EAH

Thus, the SUM value is sent to the controller in order of 02H and EAH.

3.2.4.17 How to Calculate CHECKSUM

CHECKSUM is calculated by taking the two's complement of the lower 8-bit value obtained by summing the values of received data by unsigned 8-bit addition (ignoring any overflow). When the Flash Memory SUM command or the Product Information Read command is executed, CHECKSUM is calculated in this way. The controller should also use this CHECKSUM calculation method for sending CHECKSUM values.

Example: Calculating CHECKSUM for the Flash Memory SUM command

When the upper 8-bit data of SUM is E5H and the lower 8-bit data is F6H, CHECKSUM is calculated as shown below.

First, the upper 8 bits and lower 8 bits of the SUM value are added by unsigned operation.

E5H + F6H = 1DBH

Then, the two's complement of the lower 8 bits of this result is obtained as shown below. The resulting CHECKSUM value (25H) is sent to the controller.

0 – DBH = 25H

3.2.6.10 Programming the Flash Memory by the Internal CPU

The internal CPU programs the flash memory by using the command sequences and hardware sequence flags described above. However, since the flash memory cannot be read during auto operation mode, the program/erase routine must be executed outside of the flash memory.

The CPU can program the flash memory either by using Single Boot mode or by using a user-created protocol in Single Chip mode (User Boot).

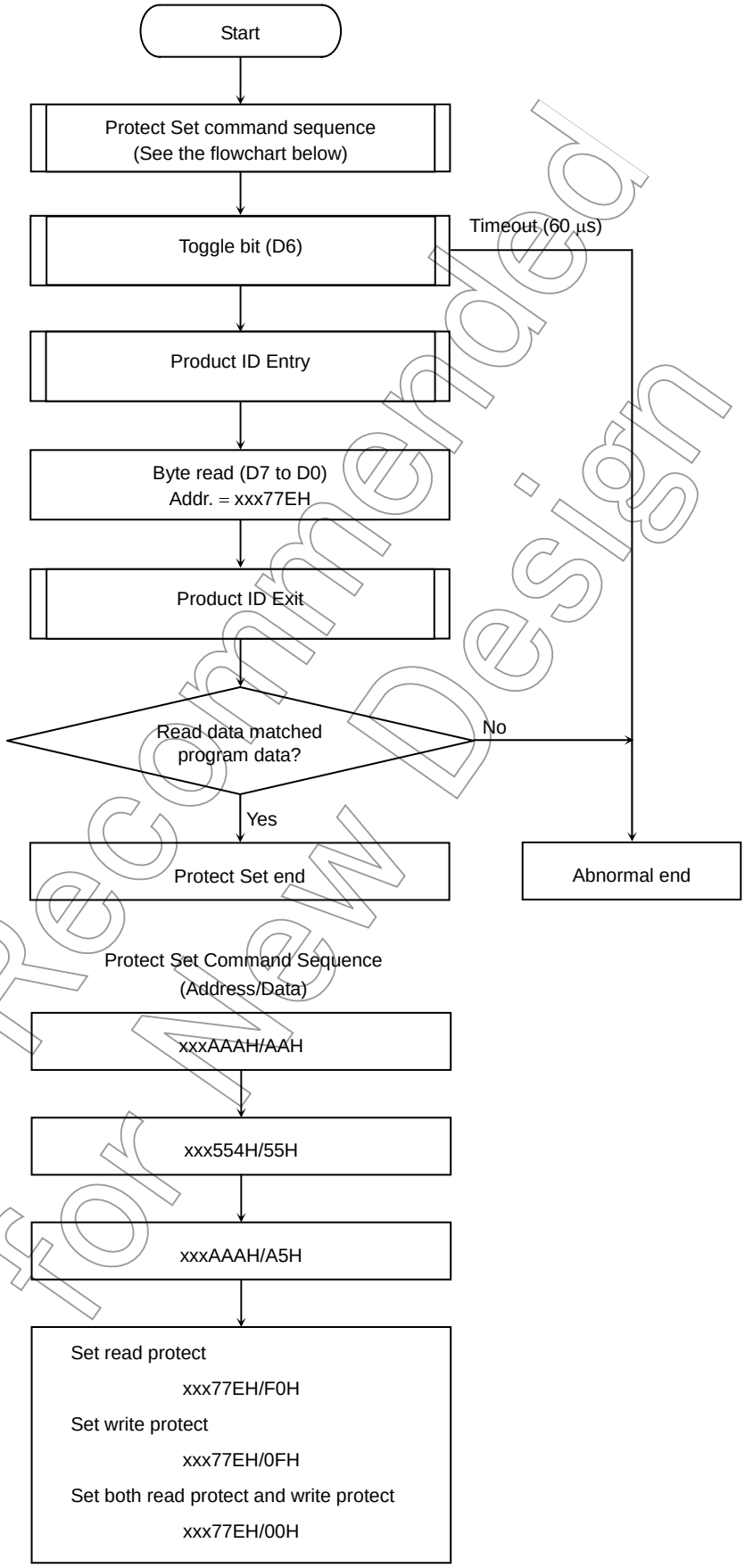
1) Single Boot:

The microcontroller is started up in Single Boot mode to program the flash memory by the internal boot ROM program. In this mode, the internal boot ROM is mapped to an area including the interrupt vector table, in which the boot ROM program is executed. The flash memory is mapped to an address area different from the boot ROM area. The boot ROM program loads data into the flash memory by serial transfer. In Single Boot mode, interrupts must be disabled including non-maskable interrupts ($\overline{\text{NMI}}$, etc.).
For details, see 3.2.4"Single Boot Mode"

2) User Boot:

In this method, the flash memory is programmed by executing a user-created routine in Single Chip mode (normal operation mode). In this mode, the user-created program/erase routine must also be executed outside of the flash memory. It is also necessary to disable interrupts including non-maskable interrupts.
The user should prepare a flash memory program/erase routine (including routines for loading write data and writing the loaded data into the flash memory). In the main program, normal operation is switched to flash memory programming operation to execute the flash memory program/erase routine outside of the flash memory area. For example, the flash memory program/erase routine may be transferred from the flash memory to the internal RAM and executed there or it may be prepared and executed in external memory.
For details, see 3.2.5"Flash Memory".

Read/Write Protect Set



(Example: Program to be loaded and executed in RAM)
Set read protection and write protection on the flash memory.

```
;#### Flash Memory Protect Set processing ####
        ld      XIX, 0xFE077E                   ; set protect address
PROTECT:
        ld      (0xFE0AAA), 0xAA                ; 1st bus write cycle
        ld      (0xFE0554), 0x55                ; 2nd bus write cycle
        ld      (0xFE0AAA), 0xA5                ; 3rd bus write cycle
        ld      (XIX), 0x00                     ; 4th bus write cycle

        cal     TOGGLECHK                       ; check toggle bit
        cal     PID_ENTRY                       ;
        ld      A, (XIX)                        ; read protected address
        cal     PID_EXIT                        ;
        cp      A, 0x00                         ;(0xFE077E)=0x00?
        j       ne, PROTECT_ERR                 ; protected?

PROTECT_END:
        j       PROTECT_END                     ; protect set operation completed

PROTECT_ERR:
        j       PROTECT_ERR                     ; protect set error


;#### Product ID Entry processing ####
PID_ENTRY:
        ld      (0xFE0AAA), 0xAA                ; 1st bus write cycle
        ld      (0xFE0554), 0x55                ; 2nd bus write cycle
        ld      (0xFE0AAA), 0x90                ; 3rd bus write cycle
        ; --- wait for 300 nsec or longer (execute NOP instruction [148nsec/@fFPH=27MHz] three times) ---
        nop
        nop
        nop                                     ; wait for 444 nsec
        ret


;#### Product ID Exit processing ####
PID_EXIT:
        ld      (0xFE0000), 0xF0                ; 1st bus write cycle
        ; --- wait for 300 nsec or longer (execute NOP instruction [148nsec/@fFPH=27MHz] three times) ---
        nop
        nop
        nop                                     ; wait for 444 nsec
        ret


;####  Toggle bit (D6) check processing ####
TOGGLECHK:
        ld      L, (XIX)
        and     L, 0y01000000                   ; check toggle bit (D6)
        ld      H, L                            ; save first toggle bit data
TOGGLECHK1:
        ld      L, (XIX)
        and     L, 0y01000000                   ; check toggle bit (D6)
        cp      L, H                            ; toggle bit = toggled?
        j       z, TOGGLECHK2                   ; if not toggled, end processing
        ld      H, L                            ; save current toggle bit state
        j       TOGGLECHK1                      ; recheck toggle bit
TOGGLECHK2:
        ret
```

(Example: Program to be loaded and executed in RAM)
Read data from address FE0000H.

```
;#### Flash memory read processing ####
READ:
        ld      WA, (0xFE0000)                  ; read data from flash memory
```

4. Electrical Characteristics

4.1 Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Power supply voltage	Vcc	−0.5 to 4.0	V
Input voltage	VIN	−0.5 to Vcc + 0.5	
Output current (1 pin)	IOL	2	mA
Output current (1 pin)	IOH	−2	
Output current (Total)	ΣIOL	80	
Output current (Total)	ΣIOH	−80	
Power dissipation (Ta = 85°C)	PD	600	mW
Soldering temperature (10 s)	TSOLDER	260	°C
Storage temperature	TSTG	−65 to 150	
Operation temperature	TOPR	−40 to 85	
Number of Times Program Erase	N_{EW}	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.

Solderability of lead free products

Test parameter	Test condition	Note
Solderability	Use of Sn−37Pb solder Bath Solder bath temperature =230°C, Dipping time = 5 seconds The number of times = one, Use of R-type flux	Pass: solderability rate until forming ≥ 95%
	Use of Sn−3.0Ag−0.5Cu solder bath Solder bath temperature = 245°C, Dipping time = 5 seconds The number of times = one, Use of R-type flux (use of lead free)	

4.4 AD Conversion Characteristics

AVCC = VCC, AVSS = VSS

Parameter	Symbol	Condition	Min	Typ.	Max	Unit
Analog input voltage	VAIN		AVSS		AVCC	V
Error (Not including quantization errors)	–	VCC = 2.2 V to 3.6 V		±1.0	±4.0	LSB

Note 1: 1 LSB = (AVCC − AVSS)/1024 [V]

Note 2: Minimum operation frequency:
The operaion of AD converter is guranteed only using fc (High frequency oscillator).
fs (Low frequency oscillator) is not guranteed. But When frequency of clock selected by clock gear is more than and eqaull 4 MHz in using fc, it is guranteed (fFPH ≥ 4 MHz).

Note 3: The value for Icc (Current of VCC pin) includes the current which flows through the AVCC pin.

4.5 Serial Channel Timing (I/O interface mode)

(1) SCLK input mode

Parameter	Symbol	Variable		10 MHz		27 MHz		Unit
		Min	Max	Min	Max	Min	Max	
SCLK period	t_{SCY}	16X		1.6		0.59		μs
Output data → SCLK rising/falling	t_{OSS}	$t_{SCY}/2 - 4X - 110$ (V_{CC} = 2.7 V to 3.6 V)		290		38		ns
		$t_{SCY}/2 - 4X - 180$ (V_{CC} = 2.2 V)		220		–		
SCLK rising/falling → Output data hold	t_{OHS}	$t_{SCY}/2 + 2X + 0$		1000		370		ns
SCLK rising /falling → Input data hold	t_{HSR}	3X + 10		310		121		ns
SCLK rising/falling → Valid data input	t_{SRD}		$t_{SCY} - 0$		1600		592	ns
Valid data input → SCLK rising/falling	t_{RDS}	0		0		0		ns

(2) SCLK ouptut mode

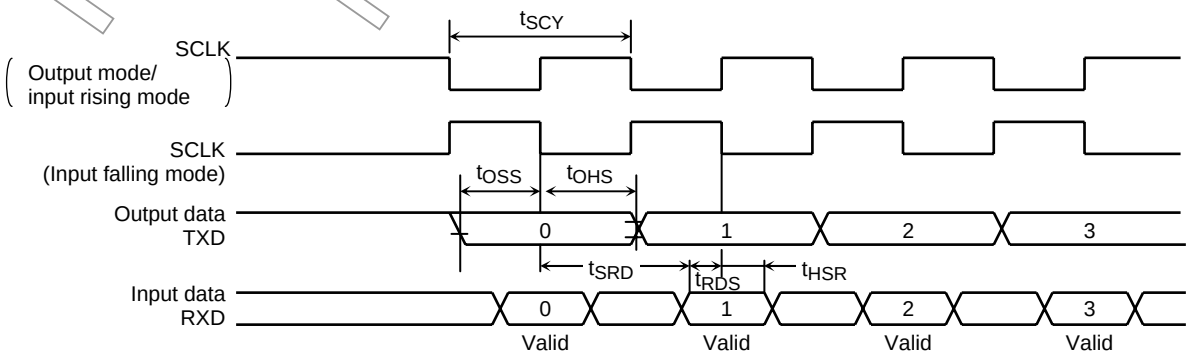
Parameter	Symbol	Variable		10 MHz		27 MHz		Unit
		Min	Max	Min	Max	Min	Max	
SCLK period	t_{SCY}	16X	8192X	1.6	819	0.59	303	μs
Output data → SCLK rising/falling	t_{OSS}	$t_{SCY}/2 - 40$		760		256		ns
SCLK rising/falling → Output data hold	t_{OHS}	$t_{SCY}/2 - 40$		760		256		ns
SCLK rising/falling → Input data hold	t_{HSR}	0		0		0		ns
SCLK rising/falling → Valid data input	t_{SRD}		$t_{SCY} - 1X - 180$		1320		375	ns
Valid data input → SCLK rising/falling	t_{RDS}	1X + 180		280		217		ns

Note 1: SCLK rising/falling: The rising edge is used in SCLK rising mode.
The falling edge is used in SCLK falling mode.

Note 2: 27 MHz and 10 MHz values are calculated from t_{SCY} = 16X case.

Note 3: Symbol [x] in the above table means the period of clock f_{FPH}. It's half period the system clock f_{SYS} for CPU core.

The period of clock f_{FPH} depends on the clock gear setting or the selection of high/low oscillator frequency.



5. Port Section Equivalent Circuit Diagrams

- Reading the circuit diagrams

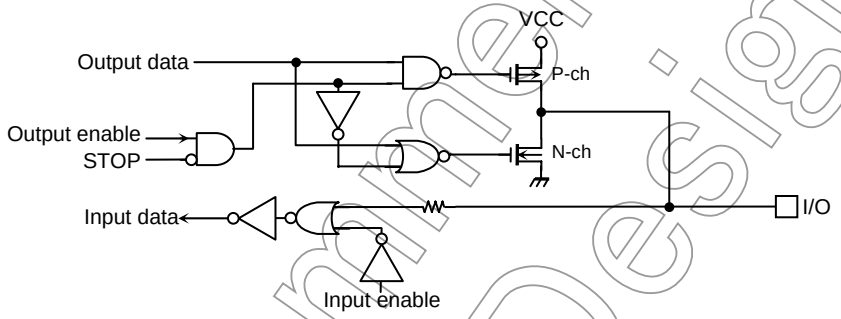
Basically, the gate symbols written are the same as those used for the standard CMOS logic IC [74HCXX] series.

The dedicated signal is described below.

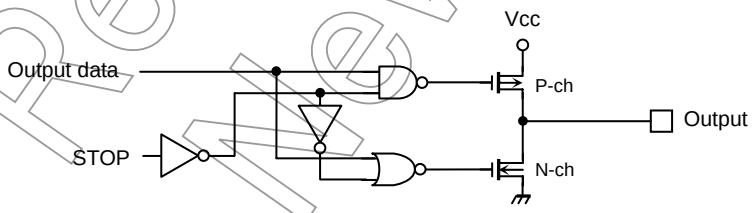
STOP : This signal becomes active 1 when the HALT mode setting register is set to the STOP mode (SYSCR2<HALTM1:0> = "01") and the CPU executes the HALT instruction. When the drive enable bit SYSCR2<DRVE> is set to "1", however STOP remains at "0".

- The input protection resistance ranges from several tens of ohms to several hundreds of ohms.

■ P0 (AD0~AD7), P1 (AD8~AD15, A8~A15), P2 (A16~A21, A0~A5), P60, P70~P74, P80~P83, P91~P92, P94~P95



■ P30 ($\overline{RD}$), P31 ($\overline{WR}$)



QFP64-P-1414-0.80A

Unit: mm