R8C/10 GROUP

SAMPLE PROGRAM THREE MINUTES TIMER

REJ05B0272-0100Z
Rev.1.00
Sep 09, 2003

1. Abstract

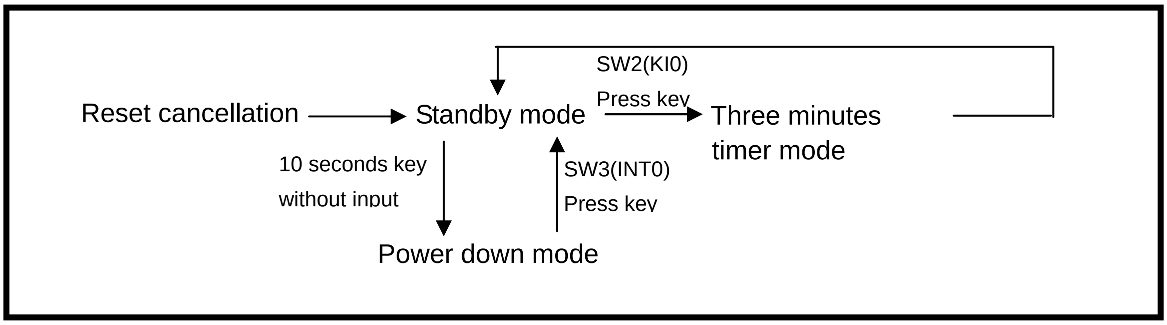
Three minutes timer uses below functions.

- Timer X (Timer mode)
- Timer Z (Programmable waveform generation mode)
- Stop mode
- INT0 interrupt

2. Introduction

Operation (1) Three minutes timer is divided by three mode.

- Standby mode
- Three minutes timer mode
- Power down mode



(2) After the reset cancellation, the microcomputer changes to standby mode.
(3) When pressing SW2(K10) in standby mode, the microcomputer changes to three minutes timer mode.
(4) The microcomputer executes three minutes count operation in three minutes timer mode.
(5) When the microcomputer completes three minutes count operation, buzzer sounds two times and the microcomputer places in standby mode. (Buzzer detailed operation is described as following)
(6) When the microcomputer can not confirm the SW2 (K10) or SW3 (INT0) key press for 10 seconds and changes to power down mode.
(7) The microcomputer stops all clock oscillations in power down mode. (change to stop mode).
(8) When the microcomputer acknowledges SW3 (INT0) in power down mode, it places in standby mode.
(9) The key input samples by 10ms periods and confirms by three times match (chattering removal)
(10) When the microcomputer places in standby mode by INT0 interrupt in power down mode, it samples key input by 10ms periods after return and confirms the SW3 (INT0) key press by three times match. When it can not confirm the key press, it changes to power down mode.
(11) The input key, SW2 (KI0) is L active (L: with pressing H: without pressing)

(12) In three minutes timer mode, the LED indicates as below.

LED indication for elapsed time :

	LED1(Red)	LED2(Green)	LED3(Green)	LED4(Green)
More than 2 minutes left	Turn off	500ms Period blinking	500msPeriod blinking	500ms Period blinking
More than 1 minute left	Turn off	500ms Period blinking	500msPeriod blinking	Turn off
More than 10 seconds left	Turn off	500ms Period blinking	Turn off	Turn off
Less than 10 seconds left	Turn off	250ms Period blinking	Turn off	Turn off
0 second left (during buzzer sounds)	Turn on	Turn off	Turn off	Turn off

(13) In standby mode and power down mode, the LED indicates as below.

LED indication for each mode :

	LED1(Red)	LED2(Green)	LED3(Green)	LED4(Green)
In standby mode	Turn on	Turn off	Turn off	Turn off
In power down mode	Turn off	Turn off	Turn off	Turn off

(14) LED1, LED2, LED3 and LED4 are L active (L: turn on H: Turn off)

(15) The buzzer output uses the timer Z programmable waveform mode and outputs from P31 (TZOUT)

(16) Waveform outputs in 1 kHz (50% Duty) and the buzzer output timing repeats ON and OFF twice by 200ms periods. Finally again, it turns OFF for 200ms once.

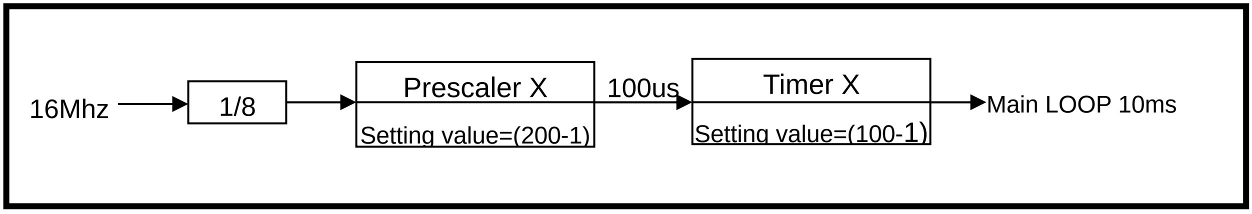
Buzzer output timing sequence :

Sequence NO	1	2	3	4
Buzzer status	ON(Operation)	OFF(Stop)	ON(Operation)	OFF(Stop)
Output time , stop time	200ms	200ms	200ms	400ms

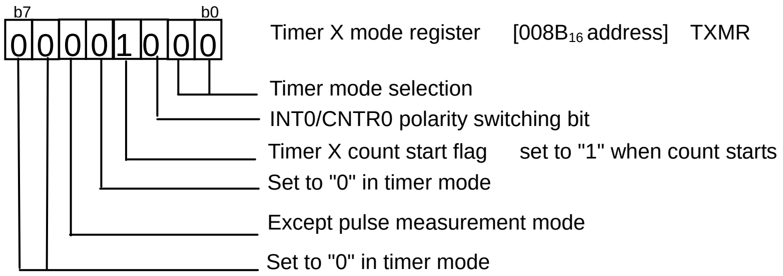
3. Program Reference

3.1 Timer X (Timer mode)

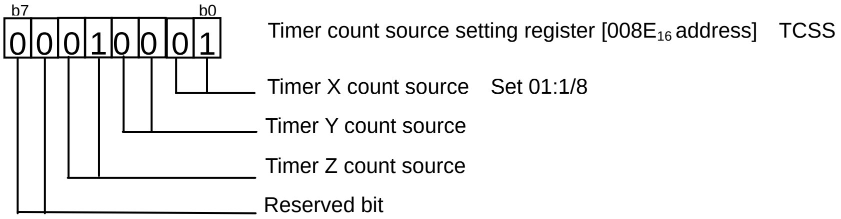
This sample program uses the timer X (timer mode) and stabilizes the main LOOP period.
The setting value of the timer X shows as below.



Timer X mode register setting


b7
b0
0 0 0 0 1 0 0 0
Timer X mode register [008B16 address] TXMR
Timer mode selection
INT0/CNTR0 polarity switching bit
Timer X count start flag set to "1" when count starts
Set to "0" in timer mode
Except pulse measurement mode
Set to "0" in timer mode

Set timer count source setting register

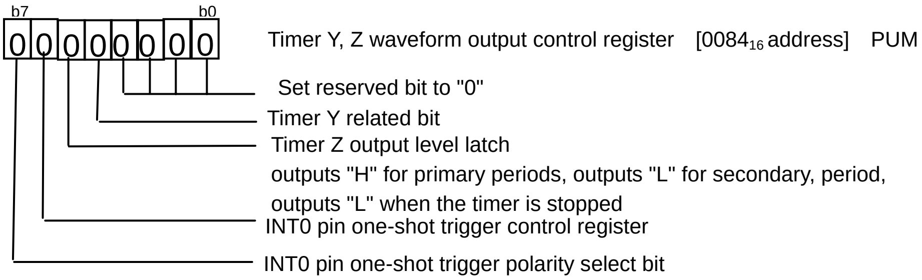

b7
b0
0 0 0 1 0 0 0 1
Timer count source setting register [008E16 address] TCSS
Timer X count source Set 01:1/8
Timer Y count source
Timer Z count source
Reserved bit

When setting as above, the timer X interrupt request bit is set to "1" by 10 ms periods.
Before this sample program executes process, it confirms to pass 10 ms by the timer X interrupt request bit.
When it confirms to pass 10 ms, it sets the timer X interrupt request bit to "0" and executes main process.
When it doesn't confirm to pass 10 ms, it waits for the timer X interrupt request bit to be set to "1".

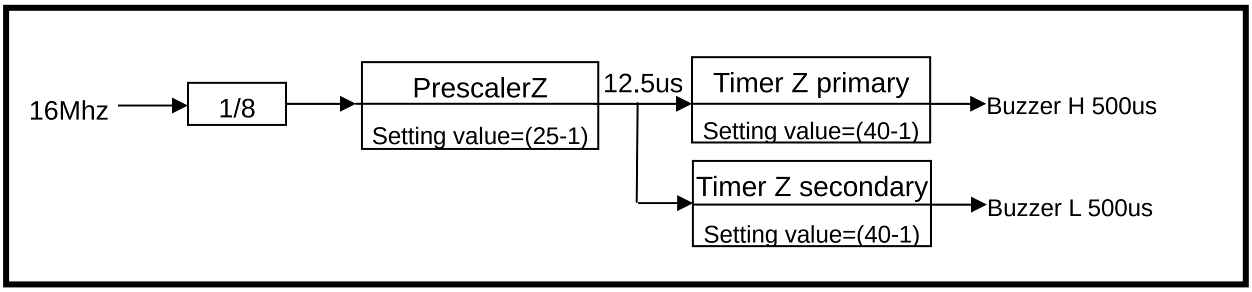
3.2 Timer Z (Programmable waveform generation mode)

The buzzer makes sounds using the timer Z (programmable waveform generation mode).
When setting as below, 1 KHz (50% Duty) is enabled to output. This program sets output level in the timer Y and the timer Z output level latch (TZOPL) of the Z waveform output control register (PUM).
Setting the timer Z output level latch to "0" causes outputs "H" for primary period, output "L" for secondary period and outputs "L" when the timer is stopped.

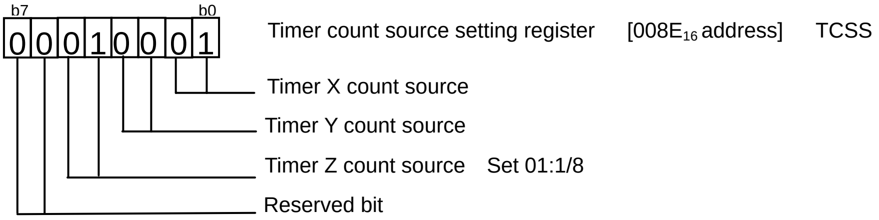
Timer Y, Z waveform output control register setting


b7
b0
0 0 0 0 0 0 0 0
Timer Y, Z waveform output control register [0084$_{16}$ address] PUM
Set reserved bit to "0"
Timer Y related bit
Timer Z output level latch
outputs "H" for primary periods, outputs "L" for secondary, period,
outputs "L" when the timer is stopped
INT0 pin one-shot trigger control register
INT0 pin one-shot trigger polarity select bit

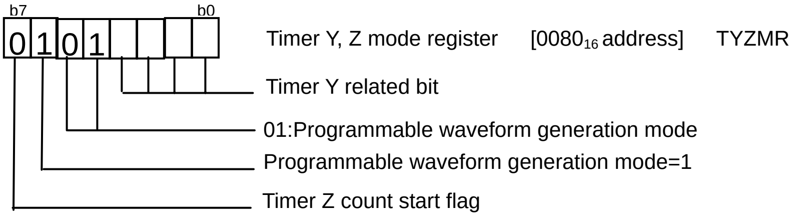
When this function starts counting, it outputs "H" for primary period and "L" for secondary period from P31 (TZOUT).
When this function stops counting, it outputs "L" from P31 (TZOUT).
The setting value of the timer Z shows as below.



Setting of timer count source setting register


b7
b0
0 0 0 1 0 0 0 1
Timer count source setting register [008E$_{16}$ address] TCSS
Timer X count source
Timer Y count source
Timer Z count source Set 01:1/8
Reserved bit

Timer Y, Z mode register setting

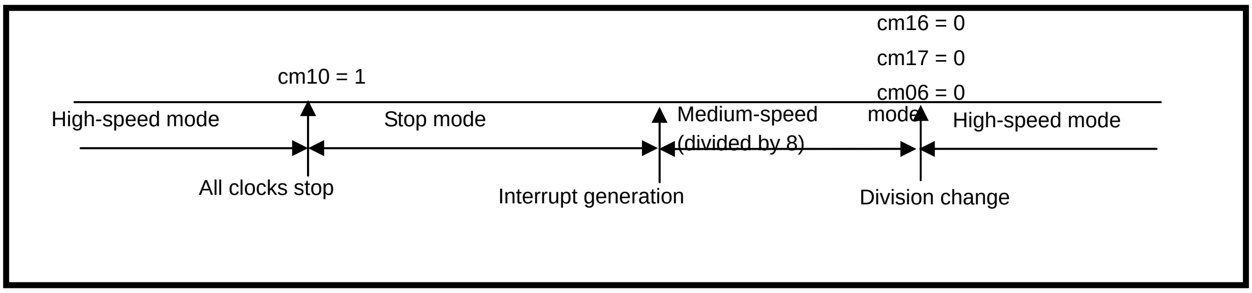

b7
b0
0 1 0 1
Timer Y, Z mode register [0080$_{16}$ address] TYZMR
Timer Y related bit
01:Programmable waveform generation mode
Programmable waveform generation mode=1
Timer Z count start flag

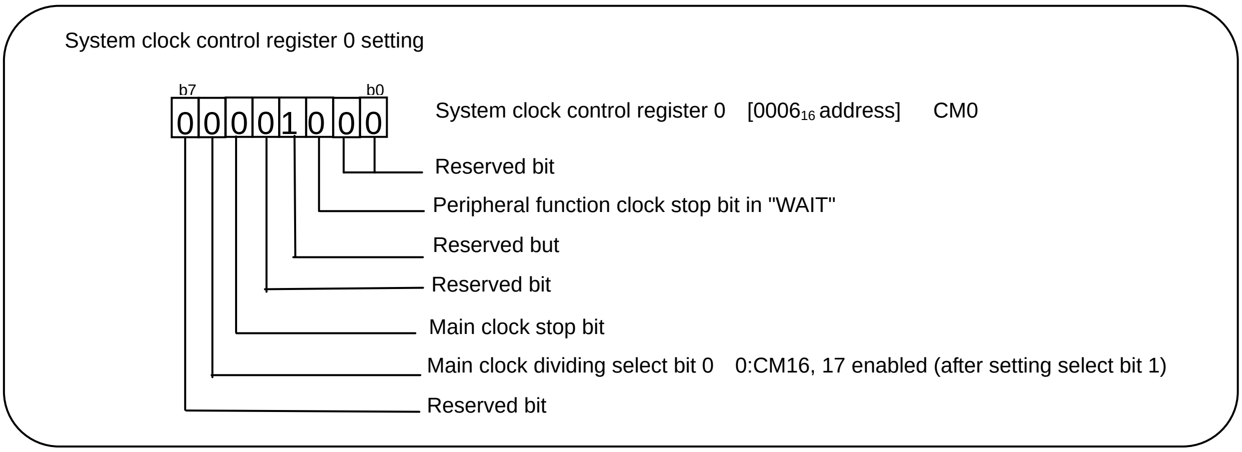
3.3 Stop Mode

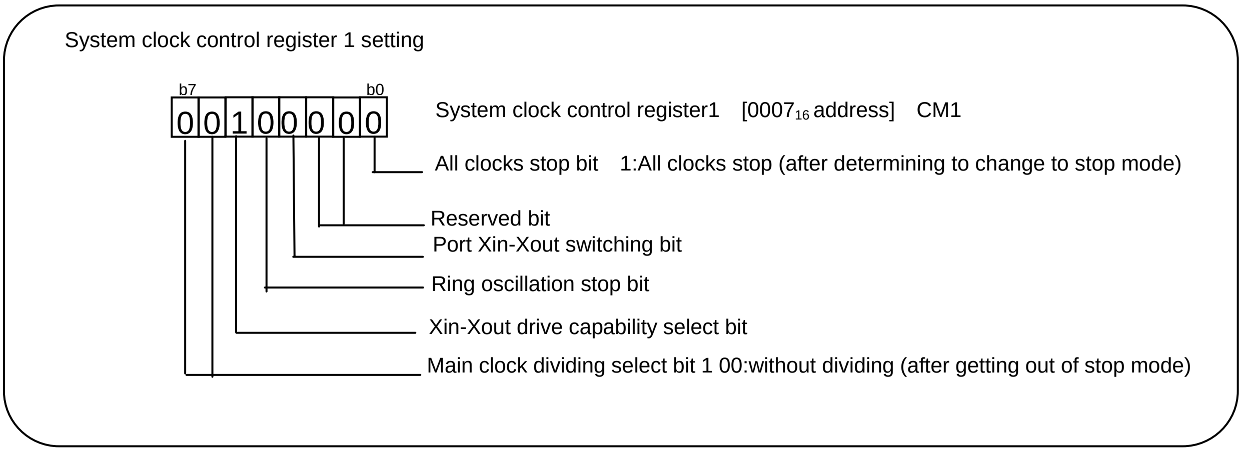
The microcomputer operates in high-speed mode for three minutes timer. When the microcomputer can not confirm the key pressing for ten seconds, SW2 (K10) or SW3 (INT0) in standby mode, it changes to stop mode. It uses INT0 interrupt by SW3 key pressing to get out of stop mode. When users set all clocks stop bit (CM10) of the system clock control register 1 (CM1) to "1:stop", all clocks are stopped (= changing to stop mode).

Because the microcomputer changes to medium-speed mode (divided by 8) at the rising edge from stop mode, this program executes to change to high-speed mode for three minutes timer. After it changes the main clock dividing select bit 1 (CM16, CM17), it changes the main clock dividing select bit 0 (CM06)

This program sets the main clock dividing select bit 1(CM16, CM17) to "00: without division" and sets the main clock dividing select bit 0 (CM06) to "0: the main clock dividing select bit 1 enabled".



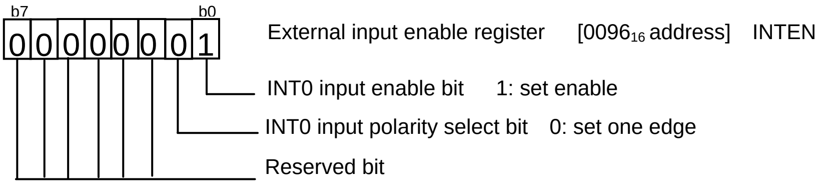

System clock control register 0 setting
b7
b0
0 0 0 0 1 0 0 0
System clock control register 0 [0006₁₆ address] CM0
Reserved bit
Peripheral function clock stop bit in "WAIT"
Reserved but
Reserved bit
Main clock stop bit
Main clock dividing select bit 0 0:CM16, 17 enabled (after setting select bit 1)
Reserved bit


System clock control register 1 setting
b7
b0
0 0 1 0 0 0 0 0
System clock control register1 [0007₁₆ address] CM1
All clocks stop bit 1:All clocks stop (after determining to change to stop mode)
Reserved bit
Port Xin-Xout switching bit
Ring oscillation stop bit
Xin-Xout drive capability select bit
Main clock dividing select bit 1 00:without dividing (after getting out of stop mode)

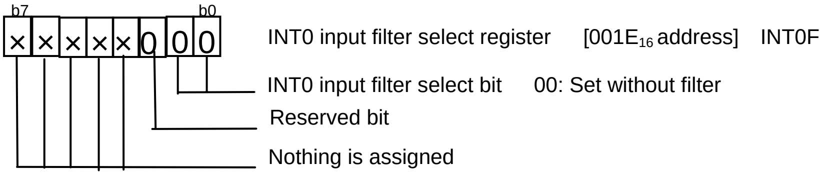
3.4 INT0 Interrupt (getting out of stop mode)

The three minutes timer program uses INT0 interrupt by the SW3 key pressing to get out of stop mode.
It enables INT0 interrupt before changing to stop mode.
Setting contents show as below.

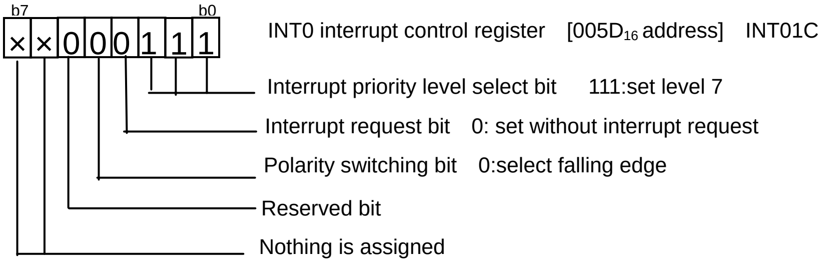
Setting of external input enabled register


b7
b0
0 0 0 0 0 0 0 1
External input enable register [0096$_{16}$ address] INTEN
INT0 input enable bit 1: set enable
INT0 input polarity select bit 0: set one edge
Reserved bit

INT0 input filter select register


b7
b0
× × × × × 0 0 0
INT0 input filter select register [001E$_{16}$ address] INT0F
INT0 input filter select bit 00: Set without filter
Reserved bit
Nothing is assigned

INT0 interrupt Control Register Setting


b7
b0
× × 0 0 0 1 1 1
INT0 interrupt control register [005D$_{16}$ address] INT01C
Interrupt priority level select bit 111:set level 7
Interrupt request bit 0: set without interrupt request
Polarity switching bit 0:select falling edge
Reserved bit
Nothing is assigned

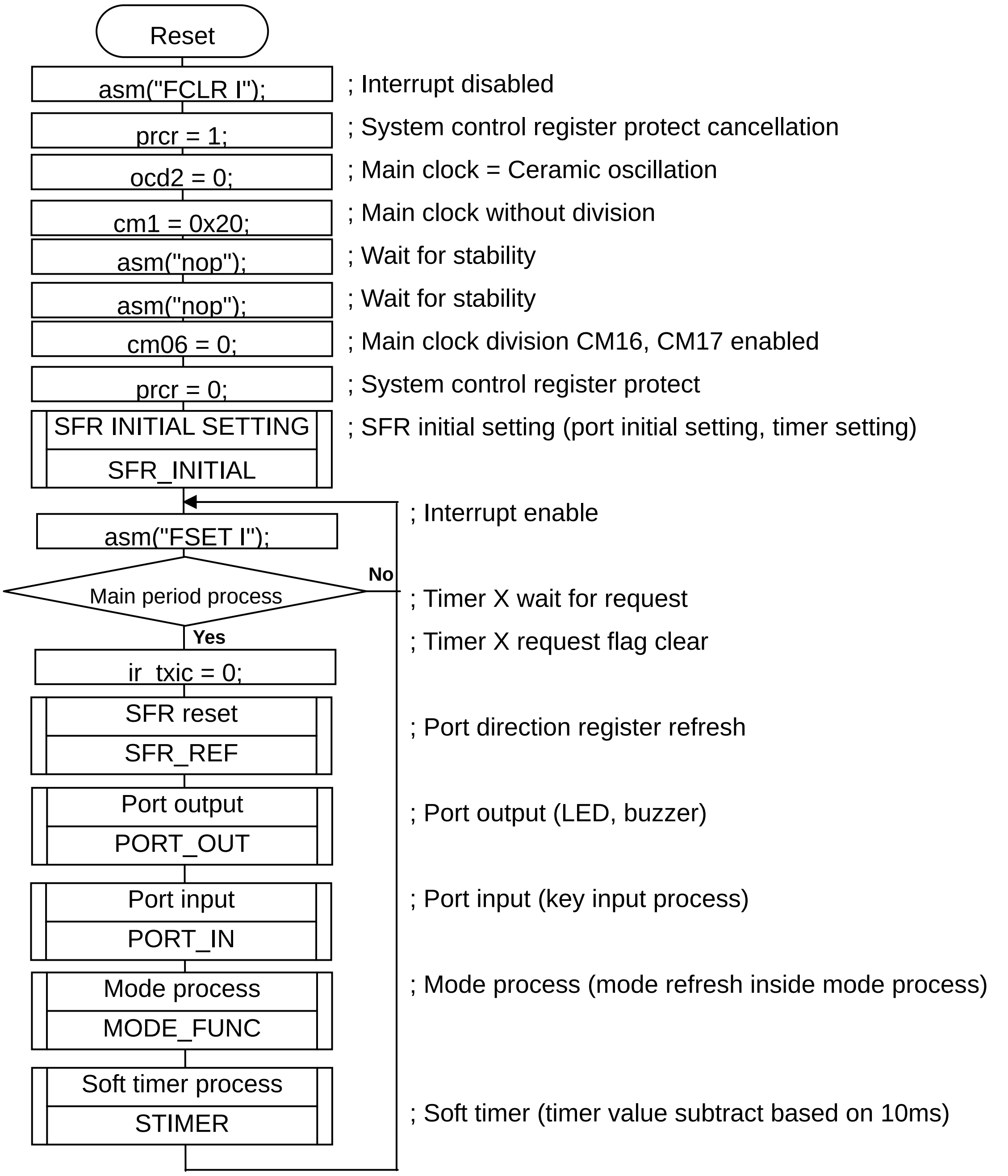
Setting above causes to judge input from INT0, detect one falling edge and generates interrupt.

3.5 Chattering Removal

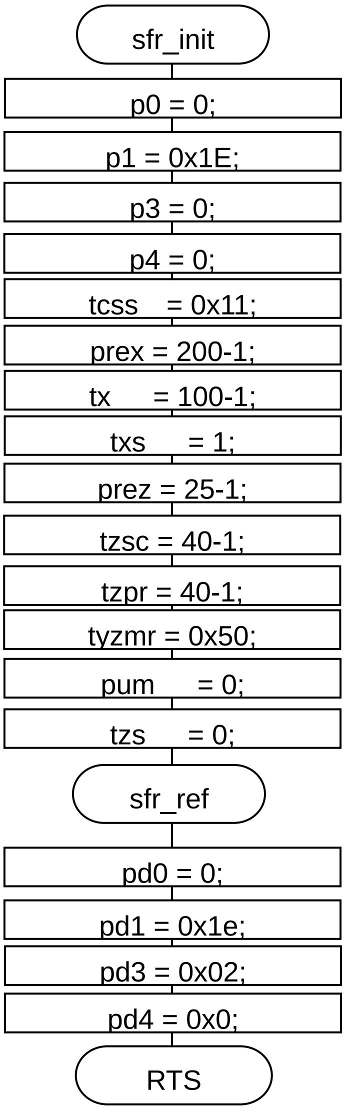
To confirm the level match needs to read the input port (SW2, SW3) a few times for the countmeasure against noise. The three minutes timer samples by 10ms periods and confirms the SW press three times matching.

4. Flow Chart

4.1 Initial Operation and Main LOOP

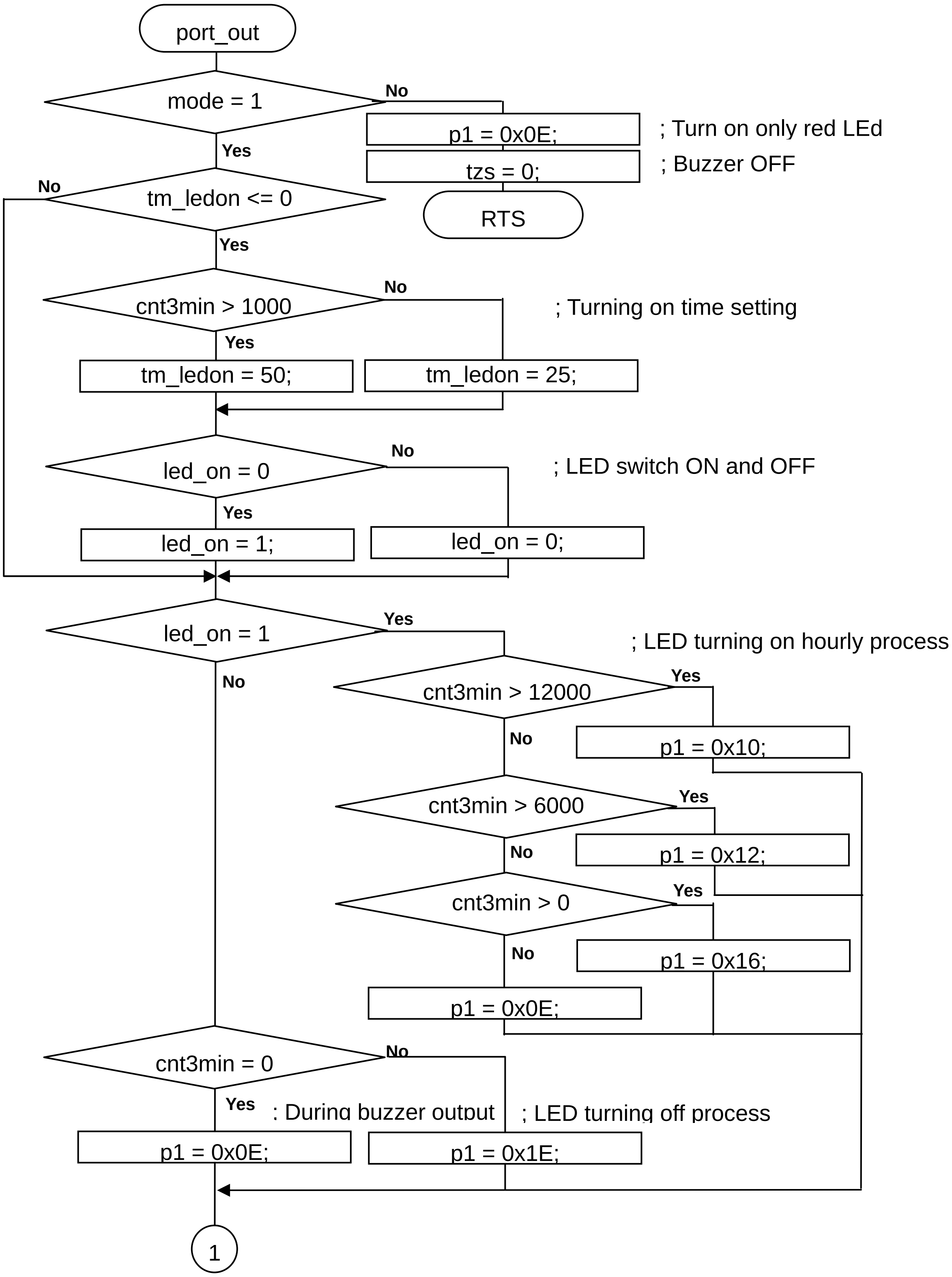

Reset
asm("FCLR I");
; Interrupt disabled
prcr = 1;
; System control register protect cancellation
ocd2 = 0;
; Main clock = Ceramic oscillation
cm1 = 0x20;
; Main clock without division
asm("nop");
; Wait for stability
asm("nop");
; Wait for stability
cm06 = 0;
; Main clock division CM16, CM17 enabled
prcr = 0;
; System control register protect
SFR INITIAL SETTING
SFR_INITIAL
; SFR initial setting (port initial setting, timer setting)
asm("FSET I");
; Interrupt enable
Main period process
No
Yes
; Timer X wait for request
ir_txic = 0;
; Timer X request flag clear
SFR reset
SFR_REF
; Port direction register refresh
Port output
PORT_OUT
; Port output (LED, buzzer)
Port input
PORT_IN
; Port input (key input process)
Mode process
MODE_FUNC
; Mode process (mode refresh inside mode process)
Soft timer process
STIMER
; Soft timer (timer value subtract based on 10ms)

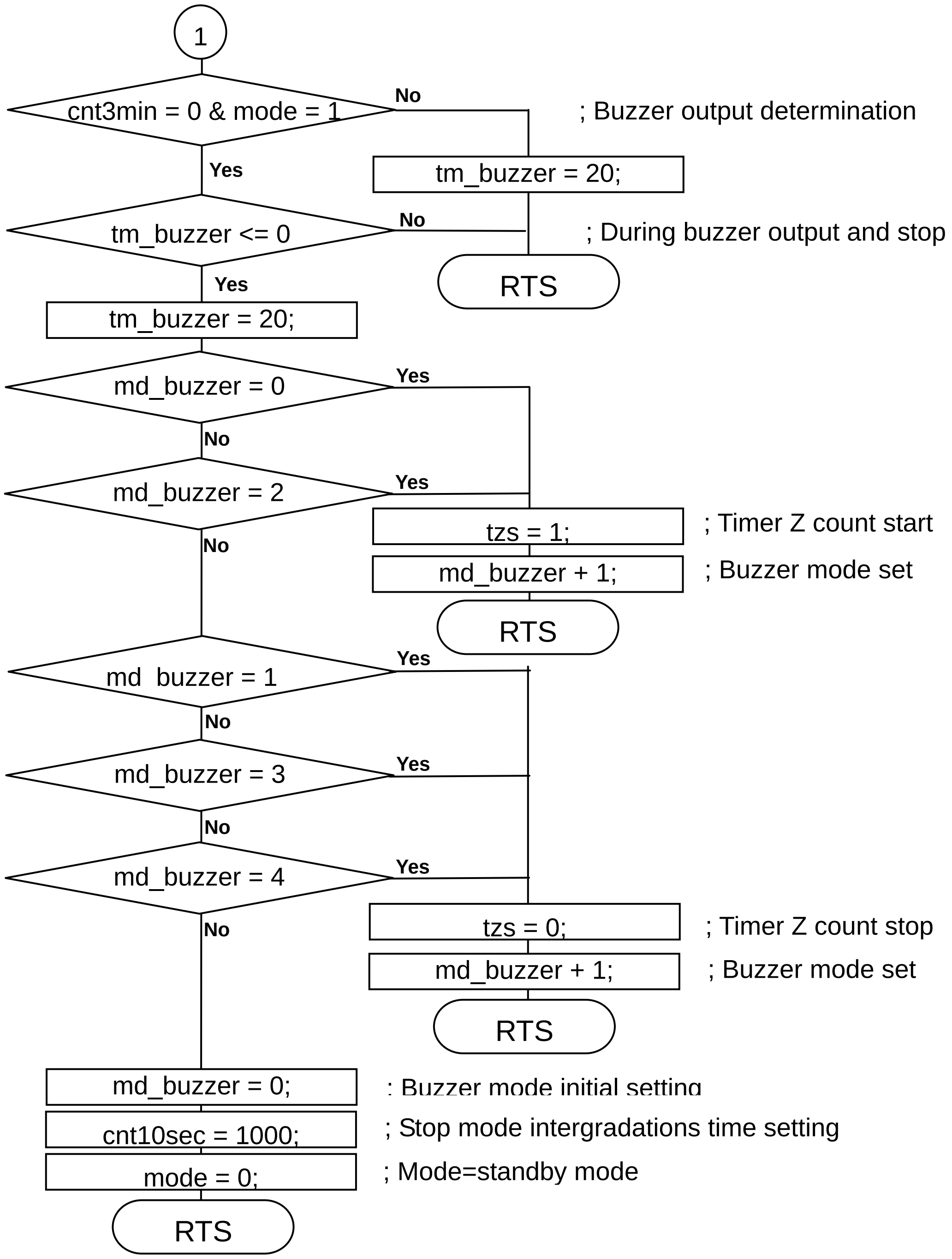
4.2 SFR Initial Setting



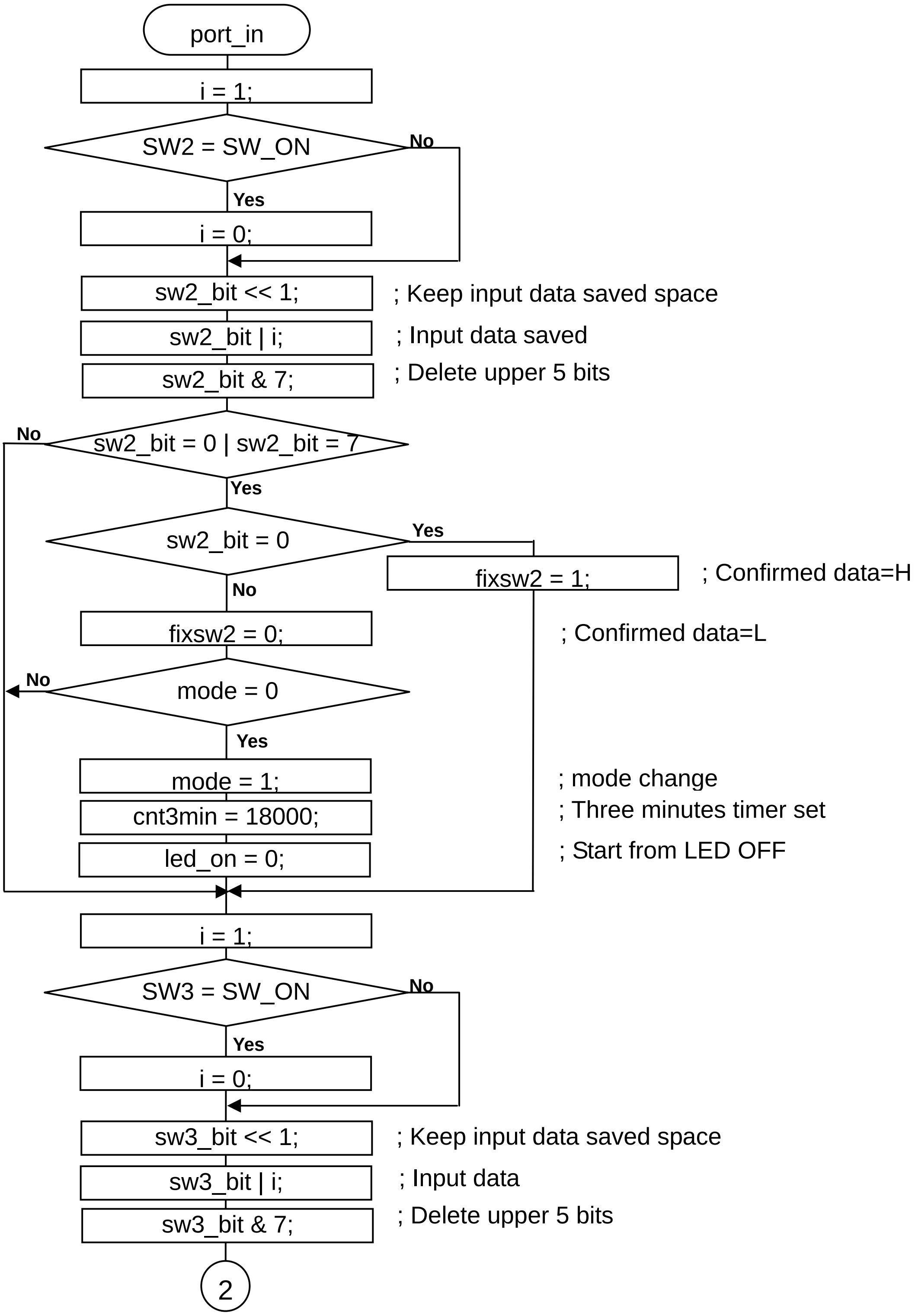
; Port 0 initial setting
; Port 1 initial setting
; Port 3 initial setting
; Port 4 initial setting
; Timer X=divided by 1/8, Timer Z=divided by 1/8
; Prescaler X=200
; Timer X=100
; Timer X count start
; Prescaler Z=25
; Timer Z secondary=40
; Timer Z primary=40
; Timer Z mode=programmable waveform generation mode
; "H" for primary period, "L" for secondary period, "L" when the timer is stopped
; Timer Z count stop
; SFR refresh
; Port 0 direction register initial setting, refresh
; Port 1 direction register initial setting, refresh
; Port 3 direction register initial setting refresh
; Port 4 direction register initial setting, refresh

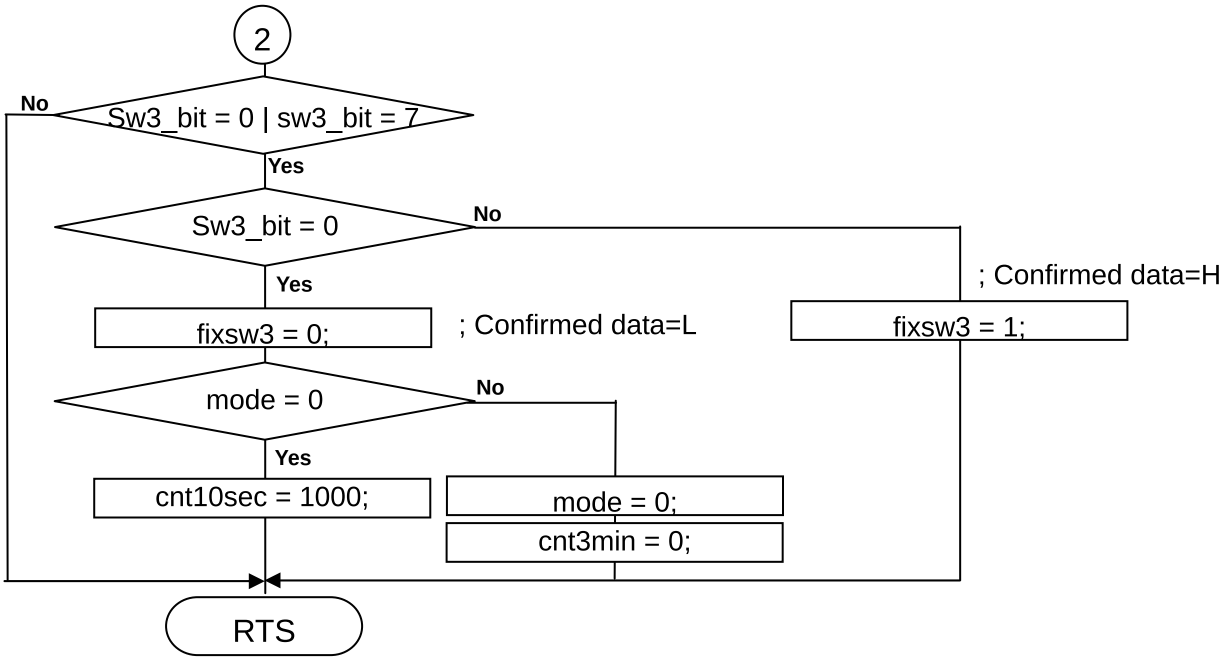
4.3 Port Output



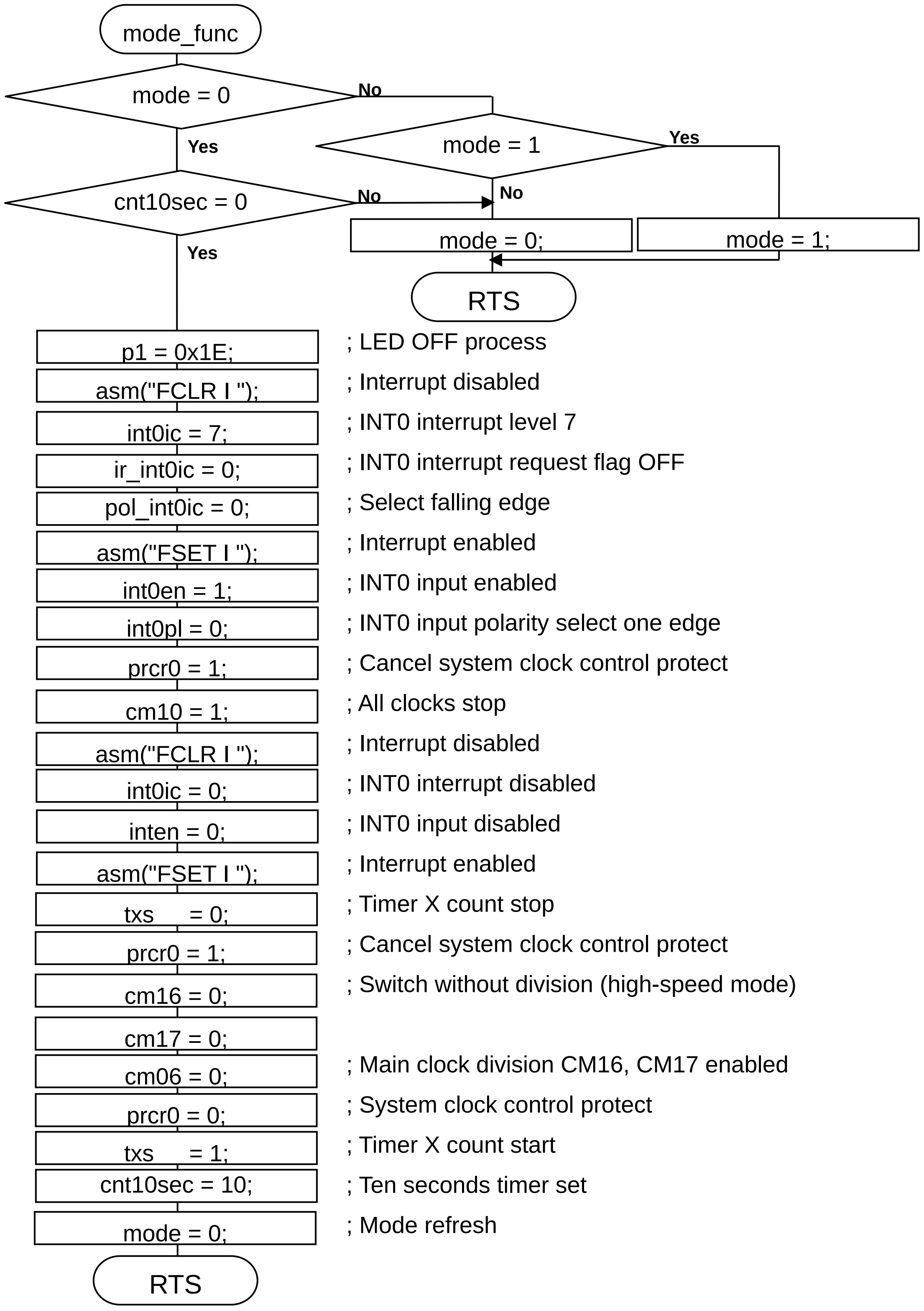

1
cnt3min = 0 & mode = 1
No
; Buzzer output determination
Yes
tm_buzzer = 20;
tm_buzzer <= 0
No
; During buzzer output and stop
Yes
RTS
tm_buzzer = 20;
md_buzzer = 0
Yes
No
md_buzzer = 2
Yes
tzs = 1;
; Timer Z count start
No
md_buzzer + 1;
; Buzzer mode set
RTS
md_buzzer = 1
Yes
No
md_buzzer = 3
Yes
No
md_buzzer = 4
Yes
tzs = 0;
; Timer Z count stop
No
md_buzzer + 1;
; Buzzer mode set
RTS
md_buzzer = 0;
; Buzzer mode initial setting
cnt10sec = 1000;
; Stop mode intergradations time setting
mode = 0;
; Mode=standby mode
RTS

4.4 Port Input

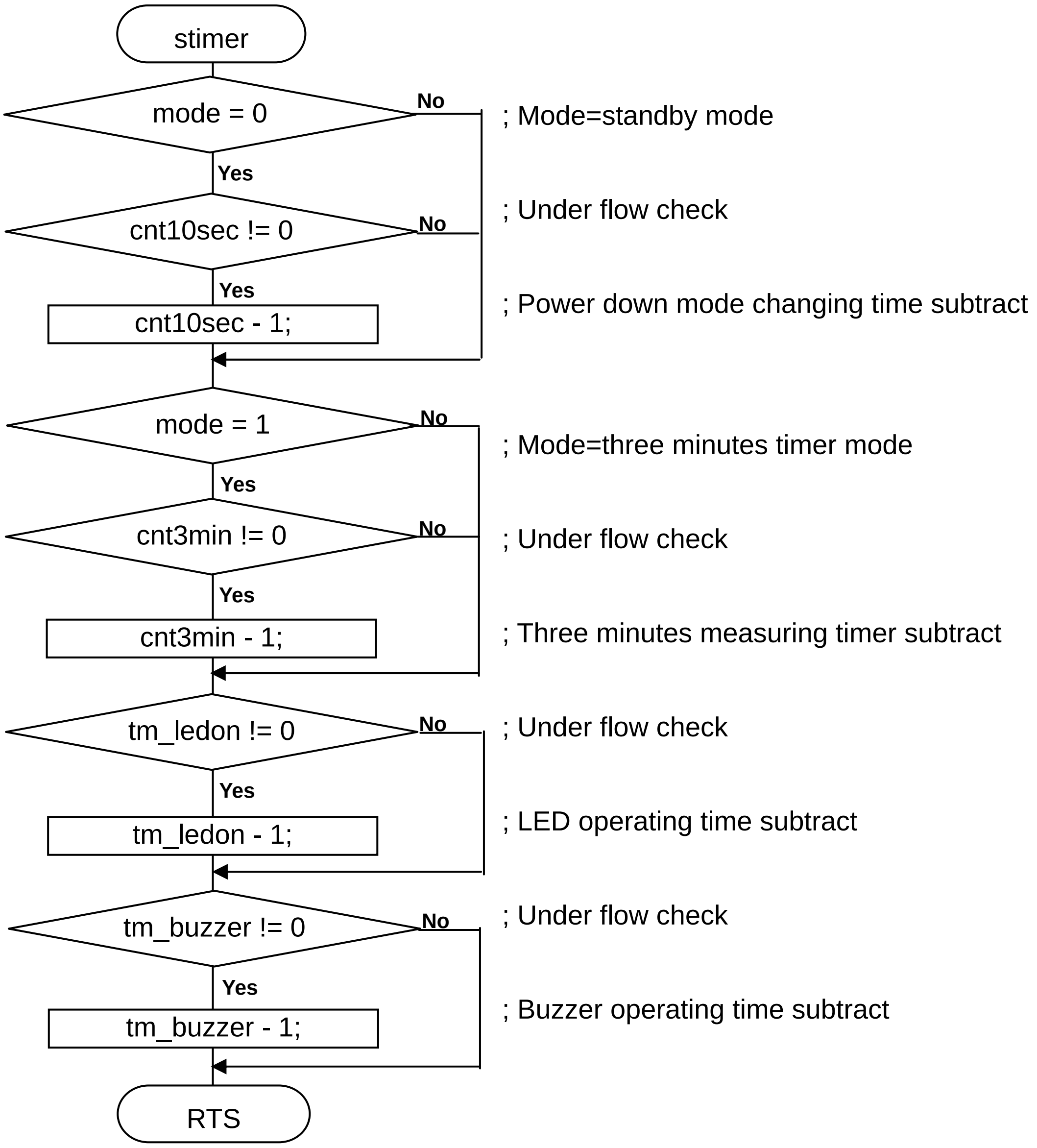



2
Sw3_bit = 0 | sw3_bit = 7
No
Yes
Sw3_bit = 0
No
Yes
fixsw3 = 0;
; Confirmed data=L
; Confirmed data=H
fixsw3 = 1;
mode = 0
No
Yes
cnt10sec = 1000;
mode = 0;
cnt3min = 0;
RTS

4.5 Mode Process


mode_func
mode = 0
No
Yes
mode = 1
Yes
No
cnt10sec = 0
No
Yes
mode = 0;
mode = 1;
RTS
p1 = 0x1E; ; LED OFF process
asm("FCLR I "); ; Interrupt disabled
int0ic = 7; ; INT0 interrupt level 7
ir_int0ic = 0; ; INT0 interrupt request flag OFF
pol_int0ic = 0; ; Select falling edge
asm("FSET I "); ; Interrupt enabled
int0en = 1; ; INT0 input enabled
int0pl = 0; ; INT0 input polarity select one edge
prcr0 = 1; ; Cancel system clock control protect
cm10 = 1; ; All clocks stop
asm("FCLR I "); ; Interrupt disabled
int0ic = 0; ; INT0 interrupt disabled
inten = 0; ; INT0 input disabled
asm("FSET I "); ; Interrupt enabled
txs = 0; ; Timer X count stop
prcr0 = 1; ; Cancel system clock control protect
cm16 = 0; ; Switch without division (high-speed mode)
cm17 = 0;
cm06 = 0; ; Main clock division CM16, CM17 enabled
prcr0 = 0; ; System clock control protect
txs = 1; ; Timer X count start
cnt10sec = 10; ; Ten seconds timer set
mode = 0; ; Mode refresh
RTS

4.6 Soft Timer Process



5. Program

```
/************************************************************************
*                                                                       *
*   File Name   : timer3min.h                                           *
*   Contents    : definition of R8C/11 Group SFR                        *
*   Copyright, 2003 RENESAS TECHNOLOGY CORPORATION                      *
*                 AND RENESAS SOLUTIONS CORPORATION                     *
*   Version     : 1.00                                                  *
*   note        :                                                       *
*                                                                       *
************************************************************************/

/* Definition of RAM area */

char mode       = 0;                    /* Mode number */
char sw2_bit    = 0;                    /* Input SW2 data */
char fixsw2     = 1;                    /* Input SW2 Settlement data */
char sw3_bit    = 0;                    /* Input SW3 data */
char fixsw3     = 1;                    /* Input SW3 Settlement data */

char md_buzzer;                     /* Buzzer control mode */
char led_on;                        /* Display on-off */
unsigned int tm_ledon;              /* LED control timer */
unsigned int tm_buzzer;             /* Buzzer control timer */
unsigned long int cnt3min;          /* Counting area in the 3 minute timer */
unsigned long int cnt10sec = 1000;  /* Counting area in the 10 seconds timer */

/* Declaration of function prototype */

extern void sfr_init(void);         /* Initial setting of SFR registers */
extern void sfr_ref(void);          /* Refresh of SFR registers */
extern void port_out(void);         /* Port output */
extern void port_in(void);          /* Port input */
extern void mode_func(void);        /* Mode processing */
extern void stimer(void);           /* Software timer */

/* Definition of base section */

#define SW_ON   0
#define SW_OFF  1
#define LOW     0
#define HIGH    1

/* Definition of port */

#define GREEN_LED2  p1_1
#define GREEN_LED1  p1_2
#define GREEN_LED0  p1_3
#define RED_LED     p1_4
#define SW2         p1_0
#define SW3         p4_5
#define BUZZER      p3_1
```

```
/************************************************************************
*                                                                       *
*   File Name   : main.c                                                *
*   Contents    : definition of R8C/11 Group SFR                        *
*   Copyright, 2003 RENESAS TECHNOLOGY CORPORATION                      *
*               AND RENESAS SOLUTIONS CORPORATION                       *
*   Version     : 1.00                                                  *
*   note        :                                                       *
*                                                                       *
************************************************************************/
#include "SFR_R811.h"           /* Definition of the R8C/11 SFR */
#include "timer3min.h"          /* Definition of processing for 3 minutes timer */

#include <stdio.h>
#include <stdlib.h>

main(){
    asm("FCLR I");              /* Interrupt enable */
    prcr = 1;                   /* Protect off */
    cm13 = 1;                   /* X-in X-out */
    cm15 = 1;                   /* XCIN-XCOUT drive capacity select bit : HIGH */
    asm("nop");
    asm("nop");
    cm05 = 0;                   /* X-in on */
    asm("nop");
    asm("nop");
    cm16 = 0;                   /* Main clock = No division mode */
    cm17 = 0;
    asm("nop");
    asm("nop");
    cm06 = 0;                   /* CM16 and CM17 enable */
    asm("nop");
    asm("nop");
    ocd2 = 0;                   /* Main clock change */
    prcr = 0;                   /* Protect on */
    sfr_init();                 /* Initial setting of SFR registers */
    while(1){                   /* Main processing */
        asm("FSET I");          /* Interrupt disable */
        while(ir_txic == 0){}   /* Main cycle 10ms */
        ir_txic = 0;
        sfr_ref();              /* Refreshment of SFR registers */
        port_out();             /* Port output */
        port_in();              /* Port input */
        mode_func();            /* Mode processing */
        stimer();               /* Software timer */
    }
}
```

```
/*****************************************************************************
Name:           sfr_init
Parameters:     None
Returns:        None
Description:    Initial setting of SFR registers
*****************************************************************************/
void sfr_init(void){
    /* Setting port registers */
    p0 = 0;
    p1 = 0x1E;              /* p14-11 = H(Led 4.3.2.1) */
    p3 = 0;                 /* p31 = L(Buzzer) */
    p4 = 0;

    tcss = 0x11;            /* division = X:1/8,Z:1/8 */
    /* Setting main cycle timer */
    /* 16Mhz* 1/8 * 200 * 100 =10ms */
    prex = 200 - 1;         /* Setting Prescaler X register */
    tx = 100 - 1;           /* Setting timer X register */
    txs = 1;                /* Timer X count start flag = start */
    /* Setting Buzzer output timer */
    /* the period of Timer Z primary = 16Mhz* 1/8 * 25 * 40 = 0.5ms */
    /* the period of Timer Z secondary = 16Mhz* 1/8 * 25 * 40 = 0.5ms */
    prez = 25 - 1;          /* Setting Prescaler Z register */
    tzsc = 40 - 1;          /* Setting timer Z secondary register */
    tzpr = 40 - 1;          /* Setting timer Z Primary register */
    tyzmr = 0x50;           /* Timer Z mode = Programmable waveform generation mode */
    pum = 0;                /* Timer Y, Z waveform output control register */
    tzs = 0;                /* Timer Z count start flag = stop */

    sfr_ref();              /* Refreshment of SFR registers */
}
/*****************************************************************************
Name:           sfr_ref
Parameters:     None
Returns:        None
Description:    Refresh of SFR registers
*****************************************************************************/
void sfr_ref(void){
    /* Setting port direction registers */
    pd0 = 0;
    pd1 = 0x1E;
    pd3 = 0x02;
    pd4 = 0x0;
}
```

```
/*******************************************************************************
Name:           port_out
Parameters:     None
Returns:        None
Description:    Port output
*******************************************************************************/
void port_out(void){
    if (mode == 1){
        /* LED output processing */
        if (tm_ledon <= 0){
            if (cnt3min > 1000){    /* Change of LED control timer */
                tm_ledon = 50;      /* Display during 500ms */
            }else{
                tm_ledon = 25;      /* Display during 250ms */
            }
            if (led_on == 0){       /* Change of display on-off */
                led_on = 1;         /* Display start */
            }else{
                led_on = 0;         /* Display stoppage */
            }
        }
        if (led_on == 1){
            if (cnt3min > 12000){   /* The remainder time is more than 2 minutes */
                p1 = 0x10;
            }else if (cnt3min > 6000){/* The remainder time is more than 1 minutes */
                p1 = 0x12;
            }else if (cnt3min > 0){ /* The remainder time is less than 1 minutes */
                p1 = 0x16;
            }else{                  /* The remainder time is nothing */
                p1 = 0x0E;
            }
        }else{
            if (cnt3min == 0){      /* Counting end */
                p1 = 0x0E;
            }else{
                p1 = 0x1E;          /* LED turn-off */
            }
        }
        /* Buzzer output processing */
        if (cnt3min == 0 && mode == 1){
            if (tm_buzzer <= 0){
                tm_buzzer = 20;     /* Buzzer control timer = 200ms */
                switch (md_buzzer){
                case 0:
                case 2:
                    tzs = 1;        /* Counting start of timer Z */
                    md_buzzer = md_buzzer + 1;
                    break;
                case 1:
                case 3:
                case 4:
                    tzs = 0;            /* Counting stoppage of timer Z */
                    md_buzzer = md_buzzer + 1;
                    break;
                default:
                    mode = 0;           /* Move to the standby mode */
                    md_buzzer = 0;
                    cnt10sec = 1000;
                }
            }
        }else{
            tm_buzzer = 20;             /* Buzzer output of 200ms */
        }
    }else{
        p1 = 0x0E;
        tzs = 0;                        /* Timer Z stops Counting */
    }
}
```

```
/******************************************************************************
Name:           port_in
Parameters:     None
Returns:        None
Description:    Port input
******************************************************************************/
void port_in(void){

 unsigned char i;

    /* Determination of input level SW2 */
    i = 1;
    if (SW2 == SW_ON)i = 0;                 /* Now determination SW2 */

    sw2_bit = sw2_bit << 1;     /* Check pulses matching a trigger input level 3
times */
    sw2_bit = sw2_bit | i;
    sw2_bit = sw2_bit & 7;

    if (sw2_bit == 0 || sw2_bit == 7){      /* Determinate input SW2 */
        if (sw2_bit == 0){
            fixsw2 = 0;                     /* Input on */
            /* Counting start of the 3 minutes timer */
            if (mode == 0){
                mode = 1;                   /* Setting the 3 minute timer mode */
                cnt3min = 18000;            /*  Initialization of the 3 minutes
timer */
                led_on = 0;                 /* Display start */
            }
        }else{
            fixsw2 = 1;                     /* Input off */
        }
    }

    /* Determination of input level SW3 */
    i = 1;
    if (SW3 == SW_ON)i = 0;                 /* Now determination SW3 */

    sw3_bit = sw3_bit << 1;     /* Check pulses matching a trigger input level 3
times */
    sw3_bit = sw3_bit | i;
    sw3_bit = sw3_bit & 7;

    if (sw3_bit == 0 || sw3_bit == 7){      /* Determinate input SW2 */
        if (sw3_bit == 0){
            fixsw3 = 0;                     /* Input on */
            if (mode == 0){
                cnt10sec = 1000;    /* Initialization of the move time to the stop
mode */
            }else{
                mode = 0;                   /* Setting the standby mode */
                cnt3min = 0;
            }
        }else{
            fixsw3 = 1;                     /* Input off */
        }
    }
}
```

```
/*******************************************************************************
Name:           mode_func
Parameters:     mode
Returns:        None
Description:    Mode processing
*******************************************************************************/
void mode_func(void){
    switch(mode){
    case 0:                     /* Standby mode */
        if (cnt10sec == 0){
            /* To Sstop mode */
            p1 = 0x1E;          /* LED turn-off */
            asm("FCLR I");      /* Interrupt enable */
            int0ic = 7;         /* INT0 interruption enable */
            ir_int0ic = 0;
            pol_int0ic = 0;
            asm("FSET I");      /* Interrupt disable */
            int0en = 1;
            int0pl = 0;
            prcr0 = 1;          /* Protect off */
            cm10 = 1;
            /* From Stop mode */
            asm("FCLR I");      /* Interrupt enable */
            int0ic = 0;         /* INT0 interruption disable */
            inten = 0;
            asm("FSET I");      /* Interrupt disable */
            txs = 0;            /* Timer X count start flag = stop */
            prcr0 = 1;          /* Protect off */
            cm16 = 0;           /* Main clock = No division mode */
            cm17 = 0;
            cm06 = 0;           /* CM16 and CM17 enable */
            prcr0 = 0;          /* Protect on */
            txs = 1;            /* Timer X count start flag = start */
            cnt10sec = 10;      /* Setting SW3 input settlement waiting time */
        }
        mode = 0;               /* Setting the standby mode */
        break;
    case 1:                     /* The 3 minute timer mode */
        mode = 1;               /* Setting the 3 minutes timer mode */
        break;
    default:
        mode = 0;               /* Refresh of the mode */
        break;
    }
}
```

```
/******************************************************************************
Name:           stimer
Parameters:     None
Returns:        None
Description:    Software timer
******************************************************************************/
void stimer(void){
    /* Countdown of 10 seconds timer */
    if (mode == 0){
        if (cnt10sec != 0){
            cnt10sec = cnt10sec - 1;
        }
    }
    /* Countdown of 3 minutes timer */
    if (mode == 1){
        if (cnt3min != 0){
            cnt3min = cnt3min - 1;
        }
    }
    /* Countdown of LED control timer */
    if (tm_ledon != 0){
        tm_ledon = tm_ledon - 1;
    }
    /* Countdown of buzzer control timer */
    if (tm_buzzer != 0){
        tm_buzzer = tm_buzzer - 1;
    }
}
```

6. Reference Documents

Hardware Manual

R8C/Tiny Series Hardware Manual

(For the most current version, please visit Renesas Technology Home Page)

7. Home Page and Support Information Window

Renesas Technology Home Page
http://www.renesas.com/

M16C Family MCU Technical Support Information Window
support_apl@renesas.com

DEVISION HISTORY	R8C/10 Group Three minutes Program Application Note

Rev.	Date	Description	
		Page	Point
		-	

Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
  Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
  The information described here may contain technical inaccuracies or typographical errors.
  Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
  Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.