
Timers Program Examples

1. Introduction

This Application Note provides to customers C and Assembler program examples for Timers.

These examples are developed for the different configuration modes of this feature.

1.1 References

- Atmel 8051 Microcontrollers Hardware Manual



**8051
Microcontrollers**

Application Note

Rev. 4345A–8051–06/04



2. C Examples

2.1 Timer 0

2.1.1 Mode 13 bits Timer Software Gated (not used)

```
/**
 * @file $RCSfile: t0_m0_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer0 in mode 0.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits timer)
 * with a software gate.
 * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
 * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0;      /* Timer 0 mode 0 with software gate */
    /* GATE0=0; C/T0#=0; M10=0; M00=0; */
    TH0 = 0x00;        /* init values */
    TL0 = 0x00;
    ET0=1;              /* enable timer0 interrupt */
    EA=1;               /* enable interrupts */
    TR0=1;              /* timer0 run */
    while(1);          /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0;           /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.2 Mode 13 bits Timer Hardware Gated

```
/**
 * @file $RCSfile: t0_m0_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits timer)
 * with a hardware gate.
 * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
 * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 0 with hardware gate */
    TMOD |= 0x08; /* GATE0=1; C/T0#=0; M10=0; M00=0; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by
hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.3 Mode 13 bits Counter Software Gated (not used)

```
/**
 * @file $RCSfile: t0_m0_c_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits counter)
 * with a software gate. The counter count up at each negative transition.
 * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
 * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
 * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 0 counter with software gate */
    TMOD |= 0x04; /* GATE0=0; C/T0#=1; M10=0; M00=0; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.4 Mode 13 bits Counter Hardware Gated

```
/**
 * @file $RCSfile: t0_m0_c_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits counter)
 * with a hardware gate. The counter count up at each negative transition.
 * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
 * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 *
 * P3.4(T0) controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 0 counter with hardware gate */
    TMOD |= 0x0C; /* GATE0=1; C/T0#=1; M10=0; M00=0; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.5 Mode 16 bits Timer Software Gated (not used)

```
/**
 * @file $RCSfile: t0_m1_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */

/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits timer)
 * with a software gate.
 * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
 * of TL0.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0;          /* Timer 0 mode 1 with software gate */
    TMOD |= 0x01;          /* GATE0=0; C/T0#=0; M10=0; M00=1; */
    TH0 = 0x00;           /* init values */
    TL0 = 0x00;
    ET0=1;                /* enable timer0 interrupt */
    EA=1;                 /* enable interrupts */
    TR0=1;                /* timer0 run */
    while(1);            /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0;             /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.6 Mode 16 bits Timer Hardware Gated

```
/**
 * @file $RCSfile: t0_m1_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits timer)
 * with a hardware gate.
 * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
 * of TL0.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 1 with hardware gate */
    TMOD |= 0x09; /* GATE0=1; C/T0#=0; M10=0; M00=1; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by
hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.7 Mode 16 bits Counter Software Gated (not used)

```
/**
 * @file $RCSfile: t0_m1_c_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits counter)
 * with a software gate. The counter count up at each negative transition.
 * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
 * of TL0.
 * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 1 counter with software gate */
    TMOD |= 0x05; /* GATE0=0; C/T0#=1; M10=0; M00=1; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.4(T0) period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.1.8 Mode 16 bits Counter Hardware Gated

```
/**
 * @file $RCSfile: t0_m1_c_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits counter)
 * with a hardware gate. The counter count up at each negative transition.
 * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
 * of TL0.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 *                  P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 0 counter with hardware gate */
    TMOD |= 0x0D; /* GATE0=1; C/T0#=1; M10=0; M00=1; */
    TH0 = 0x00; /* init values */
    TL0 = 0x00;
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.4(T0) period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.9 Mode 8 bits Auto Reload Timer Software Gated (not used)

```

/**
 * @file $RCSfile: t0_m2_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer0 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */
/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
 * timer) with a software gate.
 * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
 * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 2 with software gate */
    TMOD |= 0x02; /* GATE0=0; C/T0#=0; M10=1; M00=0; */
    TL0 = reload_value; /* init values */
    TH0 = reload_value; /* reload value */
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.1.10 Mode 8 bits Auto Reload Timer Hardware Gated

```
/**
 * @file $RCSfile: t0_m2_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer0 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */
/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
 * timer) with a hardware gate.
 * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
 * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 0 with hardware gate */
    TMOD |= 0x0A; /* GATE0=1; C/T0#=0; M10=1; M00=0; */
    TL0 = reload_value; /* init values */
    TH0 = reload_value; /* reload value */
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.11 Mode 8 bits Auto Reload Counter Software Gated (not used)

```

/**
 * @file $RCSfile: t0_m2_c_gs.c,v $
 * Copyright (c) 2004 Atmel.
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer0 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____*/

/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
 * timer) with a software gate. It count up at each negative transition.
 * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
 * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
 * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 2 counter with software gate */
    TMOD |= 0x06; /* GATE0=0; C/T0#=1; M10=1; M00=0; */
    TL0 = reload_value; /* init values */
    TH0 = reload_value; /* reload value */
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.4(T0)
 * period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.1.12 Mode 8 bits Auto Reload Counter Hardware Gated

```
/**
 * @file $RCSfile: t0_m2_c_gh.c,v $
 * Copyright (c) 2004 Atmel.
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer0 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____*/

/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
 * timer) with a hardware gate. It count up at each negative transition.
 * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
 * of TH0 for the reload value.TH0 is load in TL0 at timer0 overflow.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 *                  P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0; /* Timer 0 mode 2 counter with hardware gate */
    TMOD |= 0x0E; /* GATE0=1; C/T0#=1; M10=1; M00=0; */
    TL0 = reload_value; /* init values */
    TH0 = reload_value; /* reload value */
    ET0=1; /* enable timer0 interrupt */
    EA=1; /* enable interrupts */
    TR0=1; /* timer0 run */
    while(1); /* endless */
}

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.4(T0)
period
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.1.13 Mode Split Timer Software Gated (not used)

```
/**
 * @file $RCSfile: t0_m3_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 3.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer)
 * with a software gate. When timer 0 is placed in this mode, it essentially
 * becomes two separate 8-bits timers.
 * One consist of TL0 (8bits) and can be gated by software
 * The other consist of TH0 (8bits), is always in timer mode and cannot be
 * gated.
 * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1 always
 * running.
 * You can use this mode if you need to have two separate timers and,
 * additionally, a baud rate generator. In such case you can use the timer1 as
 * baud.
 * rate generator and use TH0/TL0 as two separate timers.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0;          /* Timer 0 mode 3 with software gate */
    TMOD |= 0x03;          /* GATE0=0; C/T0#=0; M10=1; M00=1; */

    TH0 = 0x00;           /* init values */
    TL0 = 0x00;
    ET0=1;                /* enable timer0 interrupt */
    ET1=1;                /* enable timer1 interrupt */
    EA=1;                 /* enable interrupts */
    TR0=1;                /* run TL0 */
    TR1=1;                /* run TH0 */

    while(1);            /* endless */
}
```

```

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
 * influenced by TH1
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_1 = ~P1_1; /* P1.1 toggle when interrupt. */
}

```

2.1.14 Mode Split Timer Hardware Gated

```

/**
 * @file $RCSfile: t0_m3_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 3.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"

/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer)
 * with a hardware gate. When timer 0 is placed in this mode, it essentially
 * becomes two separate 8-bits timers.
 * One consist of TL0 (8bits) and can be gated by software
 * The other consist of TH0 (8bits), is always in timer mode and cannot be
 * gated.
 * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1 always
 * running.
 * You can use this mode if you need to have two separate timers and,
 * additionally, a baud rate generator. In such case you can use the timer1 as
 * baud
 * rate generator and use TH0/TL0 as two separate timers.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0;          /* Timer 0 mode 3 with hardware gate */
    TMOD |= 0x0B;          /* GATE0=1; C/T0#=0; M10=1; M00=1; */
    TH0 = 0x00;            /* init values */
    TL0 = 0x00;
    ET0=1;                 /* enable timer0 interrupt */
    ET1=1;                 /* enable timer1 interrupt */
    EA=1;                  /* enable interrupts */
    TR0=1;                 /* run TL0 */
    TR1=1;                 /* run TH0 */
    while(1);              /* endless */
}

```

2.1.15 Mode Split Timer/counter Software Gated (not used)

```

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
 * influenced by TH1
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_1 = ~P1_1; /* P1.1 toggle when interrupt. */
}

/**
 * @file $RCSfile: t0_m3_c_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 3.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"

/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer/counter)
 * with a software gate. When timer 0 is placed in this mode, it essentially
 * becomes two separate 8-bits timers.
 * One consist of TL0 (8bits counter) and can be gated by software
 * The other consist of TH0 (8bits), is always in timer mode and cannot be
 * gated.
 * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1 always
 * running.
 * You can use this mode if you need to have two separate timers and,

```

```

* additionally, a baud rate generator. In such case you can use the timer1 as
* baud
* rate generator and use TH0/TL0 as two separate timers.
* FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
* FUNCTION_OUTPUTS: void
*/
void main(void)
{
    TMOD &= 0xF0;          /* Timer 0 mode 3 with software gate */
    TMOD |= 0x07;          /* GATE0=0; C/T0#=1; M10=1; M00=1; */

    TH0 = 0x00;           /* init values */
    TL0 = 0x00;
    ET0=1;                /* enable timer0 interrupt */
    ET1=1;                /* enable timer1 interrupt */
    EA=1;                 /* enable interrupts */
    TR0=1;                /* run TL0 */
    TR1=1;                /* run TH0 */

    while(1);            /* endless */
}

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0;             /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
 * influenced by TH1.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0;             /* reset interrupt flag (already done by hardware)*/
    P1_1 = ~P1_1; /* P1.1 toggle when interrupt. */
}

```

2.1.16 Mode Split Timer/Counter Hardware Gated

```
/**
 * @file $RCSfile: t0_m3_c_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer0 in mode 3.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"

/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer/counter)
 * with a hardware gate. When timer 0 is placed in this mode, it essentially
 * becomes two separate 8-bits timers.
 * One consist of TL0 (8bits counter) and can be gated by software
 * The other consist of TH0 (8bits), is always in timer mode and cannot be
 * gated.
 * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1 always
 * running.
 * You can use this mode if you need to have two separate timers and,
 * additionally, a baud rate generator. In such case you can use the timer1 as
 * baud.
 * rate generator and use TH0/TL0 as two separate timers.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 *                  P3.4(T0) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0xF0;      /* Timer 0 mode 3 with hardware gate */
    TMOD |= 0x0F;      /* GATE0=1; C/T0#=1; M10=1; M00=1; */
    TH0 = 0x00;        /* init values */
    TL0 = 0x00;
    ET0=1;              /* enable timer0 interrupt */
    ET1=1;              /* enable timer1 interrupt */
    EA=1;               /* enable interrupts */
    TR0=1;              /* run TL0 */
}
```

```

    TR1=1;          /* run TH0 */
    while(1);       /* endless */
}
/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer0(void) interrupt 1 /* interrupt address is 0x000b */
{
    TF0 = 0;        /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
 * influenced by TH1.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0;        /* reset interrupt flag (already done by hardware)*/
    P1_1 = ~P1_1; /* P1.1 toggle when interrupt. */
}

```

2.2 Timer 1

2.2.1 Mode 13 bits Timer Software Gated (not used)

```
/**
 * @file $RCSfile: t1_m0_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits timer)
 * with a software gate.
 * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
 * of TL1. The upper 3 bits of TL1 are undeterminate and are ignored.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F;      /* Timer 1 mode 0 with software gate */
                        /* GATE0=0; C/T0#=0; M10=0; M00=0; */
    TH1 = 0x00;        /* init values */
    TL1 = 0x00;
    ET1=1;              /* enable timer1 interrupt */
    EA=1;              /* enable interrupts */
    TR1=1;             /* timer1 run */
    while(1);          /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0;           /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.2 Mode 13 bits Timer Hardware Gated

```
/**
 * @file $RCSfile: t1_m0_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits timer)
 * with a hardware gate.
 * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
 * of TL1. The upper 3 bits of TL1 are undeterminate and are ignored.
 * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 0 with hardware gate */
    TMOD |= 0x80; /* GATE0=1; C/T0#=0; M10=0; M00=0; */
    TH1 = 0x00; /* init values */
    TL1 = 0x00;
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by
hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.3 Mode 13 bits Counter Software Gated (not used)

```
/**
 * @file $RCSfile: t1_m0_c_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits counter)
 * with a software gate. The counter count up at each negative transitions
 * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
 * of TL1. The upper 3 bits of TL1 are undeterminate and are ignored.
 * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 0 counter with software gate */
    TMOD |= 0x40; /* GATE0=0; C/T0#=1; M10=0; M00=0; */
    TH1 = 0x00; /* init values */
    TL1 = 0x00;
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.5(T1) period
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.4 Mode 16 bits Timer Software Gated (not used)

```
/**
 * @file $RCSfile: t1_m1_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits timer)
 * with a software gate.
 * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
 * of TL1.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F;      /* Timer 1 mode 1 with software gate */
    TMOD |= 0x10;      /* GATE0=0; C/T0#=0; M10=0; M00=1; */
    TH1 = 0x00;        /* init values */
    TL1 = 0x00;
    ET1=1;             /* enable timer1 interrupt */
    EA=1;              /* enable interrupts */
    TR1=1;             /* timer1 run */
    while(1);          /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0;          /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.5 Mode 16 bits Timer Hardware Gated

```
/**
 * @file $RCSfile: t1_m1_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits timer)
 * with a hardware gate.
 * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
 * of TL1.
 * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 0 mode 1 with hardware gate */
    TMOD |= 0x90; /* GATE0=1; C/T0#=0; M10=0; M00=1; */
    TH1 = 0x00; /* init values */
    TL1 = 0x00;
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by
hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.6 Mode 16 bits Counter Software Gated (not used)

```

/**
 * @file $RCSfile: t1_m1_c_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits counter)
 * with a software gate. The counter count up at each negative transition.
 * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
 * of TL1.
 * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 1 counter with software gate */
    TMOD |= 0x50; /* GATE0=0; C/T0#=1; M10=0; M00=1; */
    TH1 = 0x00; /* init values */
    TL1 = 0x00;
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.5(T1) period
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.2.7 Mode 16 bits Counter Hardware Gated

```
/**
 * @file $RCSfile: t1_m1_c_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits counter)
 * with a hardware gate. The counter count up at each negative transitions
 * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
 * of TL1.
 * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
 *                  P3.5(T1) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 0 counter with hardware gate */
    TMOD |= 0xD0; /* GATE0=1; C/T0#=1; M10=0; M00=1; */
    TH1 = 0x00; /* init values */
    TL1 = 0x00;
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.5(T1) period
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware) */
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.8 Mode 8 bits Auto Reload Timer Software Gated (not used)

```

/**
 * @file $RCSfile: t1_m2_t_gs.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer1 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */
/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
 * timer) with a software gate.
 * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
 * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 2 with software gate */
    TMOD |= 0x20; /* GATE0=0; C/T0#=0; M10=1; M00=0; */
    TL1 = reload_value; /* init values */
    TH1 = reload_value; /* reload value */
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.2.9 Mode 8 bits Auto Reload Timer Hardware Gated

```
/**
 * @file $RCSfile: t1_m2_t_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer1 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */
/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
 * timer) with a hardware gate.
 * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
 * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
 * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 0 with hardware gate */
    TMOD |= 0xA0; /* GATE0=1; C/T0#=0; M10=1; M00=0; */
    TL1 = reload_value; /* init values */
    TH1 = reload_value; /* reload value */
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}
/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}
```

2.2.10 Mode 8 bits Auto Reload Counter Software Gated (not used)

```

/**
 * @file $RCSfile: t1_m2_c_gs.c,v $
 * Copyright (c) 2004 Atmel.
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer1 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____*/

/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
 * timer) with a software gate. It count up at each negative transition.
 * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
 * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
 * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    TMOD &= 0x0F; /* Timer 1 mode 2 counter with software gate */
    TMOD |= 0x60; /* GATE0=0; C/T0#=1; M10=1; M00=0; */
    TL1 = reload_value; /* init values */
    TH1 = reload_value; /* reload value */
    ET1=1; /* enable timer1 interrupt */
    EA=1; /* enable interrupts */
    TR1=1; /* timer1 run */
    while(1); /* endless */
}

/**
 * FUNCTION_PURPOSE: timer1 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.5(T1)
period
 */
void it_timer1(void) interrupt 3 /* interrupt address is 0x001b */
{
    TF1 = 0; /* reset interrupt flag (already done by hardware)*/
    P1_0 = ~P1_0; /* P1.0 toggle when interrupt. */
}

```

2.2.11 Mode 8 bits Auto Reload Counter Hardware Gated

```
/**
 * @file $RCSfile: t1_m2_c_gh.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer1 in mode 2.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */
/**
 * This is a macro local to this file .
 */
#define reload_value 0x36 /* reload value example */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
 * timer) with a hardware gate. It count up at each negative transition.
 * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
 * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
 * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
 *
 * P3.5(T1) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
```

2.3 Timer 2

2.3.1 Mode 16 bits up/down Auto reload Timer

```

/**
 * @file $RCSfile: t2_m0_t.c,v $
 * Copyright (c) 2004 Atmel.
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer2 in mode 0.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____*/
/**
 * This is a macro local to this file .
 */
#define MSB_reload_value 0x36 /* msb reload value exemple */
#define LSB_reload_value 0x36 /* lsb reload value exemple */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits auto-reload
 * up/down counting timer).
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2. The EXF2 bit toggles when timer2 overflow or underflow occurs.
 * EXF2 does not generate interrupt. This bit can be used to provide 17-bit
 * resolution
 * FUNCTION_INPUTS: P1.1(T2EX)=0 for down counting or 1 for up counting.
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    T2MOD &= 0xFC; /* T2OE=0;DCEN=1; */
    T2MOD |= 0x01;
    EXF2=0; /* reset flag */
    TCLK=0;RCLK=0; /* disable baud rate generator */
    EXEN2=0; /* ignore events on T2EX */
    TH2=MSB_reload_value; /* Init msb_value */
    TL2=LSB_reload_value; /* Init lsb_value */
    RCAP2H=MSB_reload_value; /* reload msb_value */
    RCAP2L=LSB_reload_value; /* reload lsb_value */
    C_T2=0; /* timer mode */
    CP_RL2=0; /* reload mode */
    EA=1; /* interrupt enable */
    ET2=1; /* enable timer2 interrupt */
    TR2=1; /* timer2 run */
    while(1); /* endless */
}

```

2.3.2 Mode 16 bits up/down Auto reload Counter

```
/**
 * FUNCTION_PURPOSE: timer2 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * (65536-reload_value) cycles
 */
void it_timer2(void) interrupt 5 /* interrupt address is 0x002b */
{
    P1_2 = ~P1_2; /* P1.2 toggle when interrupt. */
    TF2 = 0;      /* reset interrupt flag */
}
```

```
/**
 * @file $RCSfile: t2_m0_c.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer2 in mode 0.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____ */

/**
 * This is a macro local to this file .
 */
#define MSB_reload_value 0x36 /* msb reload value exemple */
#define LSB_reload_value 0x36 /* lsb reload value exemple */

/* @section I N C L U D E S */
#include "reg_c51.h"
```

```

/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits auto-reload
 * up/down counter).
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2. The EXF2 bit toggles when timer2 overflow or underflow occurs.
 * EXF2 does not generate interrupt. This bit can be used to provide 17-bit
 * resolution
 * FUNCTION_INPUTS: P1.0(T2) must be controlled by an external clock
 *                  P1.1(T2EX)=0 for down counting or 1 for up counting.
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    T2MOD &= 0xFC;      /* T2OE=0;DCEN=1; */
    T2MOD |= 0x01;

    EXF2=0;              /* reset flag */
    TCLK=0;RCLK=0;       /* disable baud rate generator */
    EXEN2=0;             /* ignore events on T2EX */
    TH2=MSB_reload_value; /* Init msb_value */
    TL2=LSB_reload_value; /* Init lsb_value */
    RCAP2H=MSB_reload_value; /* reload msb_value */
    RCAP2L=LSB_reload_value; /* reload lsb_value */
    C_T2=1;              /* counter mode */
    CP_RL2=0;            /* reload mode */
    EA=1;                /* interrupt enable */
    ET2=1;               /* enable timer2 interrupt */
    TR2=1;               /* timer2 run */

    while(1);           /* endless */
}

/**
 * FUNCTION_PURPOSE: timer2 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * (65536-reload_value) * P1.0(T2)
 * period
 */
void it_timer2(void) interrupt 5 /* interrupt address is 0x002b */
{
    P1_2 = ~P1_2; /* P1.2 toggle when interrupt. */
    TF2 = 0;      /* reset interrupt flag */
}

```

2.3.3 Mode 16 bits Capture Timer Periodmeter application

```
/**
 * @file $RCSfile: t2_m1_t.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer2 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */

/* @section I N C L U D E S */
#include "reg_c51.h"
#include <stdio.h>

unsigned int RCAP2=0x0000; /*16 bits capture value*/
unsigned int nb_overflows=0x0000; /*number of overflow*/
unsigned int nb_overflows_old=0x0000; /*previous number of overflow*/
unsigned int RCAP2H_16=0x0000; /*msb capture value on 16 bits*/

unsigned int RCAP2_old=0x0000; /*previous 16 bits capture value*/
unsigned int RCAP2H_old=0x0000; /*previous msb capture value on 16 bits*/
unsigned char RCAP2L_old=0x00; /*previous lsb capture value*/

unsigned char RCAP2H_tmp=0x00; /*temp lsb capture value*/
unsigned char RCAP2L_tmp=0x00; /*temp lsb capture value*/
char first_passage=1;
float value; /*time between two negatives transitions*/
char nb_samples=0;
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits auto-reload
 * down counting timer).
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2.
 * FUNCTION_INPUTS: P1.1(T2EX) is a periodic signal : 50 us to 1 Hour
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    T2MOD &= 0xFC; /* T2OE=0;DCEN=0; */
    T2MOD |= 0x00;
    EXF2=0; /* reset flag */
    TCLK=0;RCLK=0; /* disable baud rate generator */
}
```

```

EXEN2=1;                /* enable events detect on T2EX */
C_T2=0;                 /* timer mode */
CP_RL2=1;               /* capture mode */
EA=1;                   /* interrupt enable */
ET2=1;                  /* enable timer2 interrupt */
TR2=1;                  /* timer2 run */

while(1)/* main task */
{
    if(first_passage==1 && nb_samples>1)
    {
        first_passage=0;
        RCAP2H_16=RCAP2H; /* convert 8 bits to 16 bits */
        RCAP2_old=((RCAP2H_old<<8)&0xFF00)|RCAP2L_old;
        RCAP2=((RCAP2H_16<<8)&0xFF00)|RCAP2L;
        value=RCAP2 + 0x10000 * nb_overflows_old - RCAP2_old;
        /* value = (signal period / cycle period) */
    }
}

/**
 * FUNCTION_PURPOSE: timer2 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void it_timer2(void) interrupt 5 /* interrupt address is 0x002b */
{
    if(TF2) nb_overflows++;
    if(EXF2)
    {
        nb_samples++;
        RCAP2L_old=RCAP2L_tmp;
        RCAP2H_old=RCAP2H_tmp;
        RCAP2L_tmp=RCAP2L;
        RCAP2H_tmp=RCAP2H;
        nb_overflows_old=nb_overflows;//save number of overflow
        nb_overflows=0x0000;//reset number of overflow
        if(nb_samples>1) EXEN2=0;//end of capture
    }
    EXF2=0;                /* reset flag */
    TF2 = 0;               /* reset interrupt flag */
}

```

2.3.4 Mode 16 bits Capture Counter

Frequency meter application

```
/**
 * @file $RCSfile: t2_m1_t.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 *
 * @brief This file is an example to use timer2 in mode 1.
 *
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 *
 * @version $Revision: 1.0 $ $Name: $
 */

/* @section I N C L U D E S */
#include "reg_c51.h"

unsigned int RCAP2=0x0000; /*16 bits capture value*/
unsigned int nb_overflows=0x0000; /*number of overflow*/
unsigned int nb_overflows_old=0x0000; /*previous number of overflow*/
unsigned int RCAP2H_16=0x0000; /*msb capture value on 16 bits*/

unsigned int RCAP2_old=0x0000; /*previous 16 bits capture value*/
unsigned int RCAP2H_old=0x0000; /*previous msb capture value on 16 bits*/
unsigned char RCAP2L_old=0x00; /*previous lsb capture value*/

unsigned char RCAP2H_tmp=0x00; /*temp lsb capture value*/
unsigned char RCAP2L_tmp=0x00; /*temp lsb capture value*/
char first_passage=1;
float value; /* period number of P1.0 between two negatives transitions of P1.1 */
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits capture
 * with hardware gate).
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2.
 * FUNCTION_INPUTS: P1.1(T2EX) is a 1Hz signal
 * P1.0(T2) is a périodique signal between 1Hz and 499Khz
 * FUNCTION_OUTPUTS: void
 */
void main(void)
{
    T2MOD &= 0xFC; /* T2OE=0;DCEN=0; */
    T2MOD |= 0x00;
    EXF2=0; /* reset flag */
    TCLK=0;RCLK=0; /* disable baud rate generator */
    EXEN2=1; /* enable events detect on T2EX */
}
```

```

C_T2=1; /* counter mode */
CP_RL2=1; /* capture mode */
EA=1; /* interrupt enable */
ET2=1; /* enable timer2 interrupt */
TR2=1; /* timer2 run */

while(1)/* main task */
{
    if(first_passage==1 && nb_overflows==0)
    {
        first_passage=0;
        RCAP2H_16=RCAP2H; /* convert 8 bits to 16 bits */
        RCAP2_old=((RCAP2H_old<<8)&0xFF00)|RCAP2L_old;
        RCAP2=((RCAP2H_16<<8)&0xFF00)|RCAP2L;
        value=RCAP2 + 0x10000 * nb_overflows_old - RCAP2_old;
        /* value = Freq P1.0 / Freq P1.1 */
    }
}

}

/**
 * FUNCTION_PURPOSE: timer2 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: void
 */
void it_timer2(void) interrupt 5 /* interrupt address is 0x002b */
{
    if(TF2) nb_overflows++;
    if(EXF2)
    {
        RCAP2L_old=RCAP2L_tmp;
        RCAP2H_old=RCAP2H_tmp;
        RCAP2L_tmp=RCAP2L;
        RCAP2H_tmp=RCAP2H;
        nb_overflows_old=nb_overflows;//save number of overflow
        nb_overflows=0x0000;//reset number of overflow
        first_passage=1;
    }
    EXF2=0; /* reset flag */
    TF2 = 0; /* reset interrupt flag */
}

```

2.3.5 Clock-out Mode

```
/**
 * @file $RCSfile: t2_m2.c,v $
 *
 * Copyright (c) 2004 Atmel.
 *
 * Please read file license.txt for copyright notice.
 * @brief This file is an example to use timer2 in mode 2.
 * This file can be parsed by Doxygen for automatic documentation
 * generation.
 * Put here the functional description of this file within the software
 * architecture of your program.
 * @version $Revision: 1.0.0 $ $Name: $
 */
/*_____ M A C R O S _____*/
/**
 * This is a macro local to this file .
 */
#define MSB_reload_value 0xFF /* msb reload value exemple */
#define LSB_reload_value 0xF2 /* lsb reload value exemple */
/* @section I N C L U D E S */
#include "reg_c51.h"
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 1 (clock-out mode and
 * negative transition detector).
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits of TL2.
 * TF2 does not generate interrupt.
 * A negative transition on P1.1(T2EX) generate an interrupt.
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0(T2) as clock output : Fout = Fperiph / (2*(65536-
 * RCAP2)).
 */
void main(void)
{
    T2MOD &= 0xFE;      /* T2OE=1;DCEN=0; */
    T2MOD |= 0x02;
    EXF2=0;              /* reset flag */
    TCLK=0;RCLK=0;        /* disable baud rate generator */
    EXEN2=1;              /* enable events on T2EX */
    TH2=MSB_reload_value; /* Init msb_value */
    TL2=LSB_reload_value; /* Init lsb_value */
    RCAP2H=MSB_reload_value; /* reload msb_value */
    RCAP2L=LSB_reload_value; /* reload lsb_value */
    C_T2=0;              /* timer mode */
    CP_RL2=0;            /* reload mode */
    EA=1;                /* interrupt enable */
    ET2=1;                /* enable timer2 interrupt */
    TR2=1;                /* timer2 run */
    while(1);            /* endless */
}
```

```
/**
 * FUNCTION_PURPOSE: timer2 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * P1.1(T2EX) period
 */
void it_timer2(void) interrupt 5 /* interrupt address is 0x002b */
{
    P1_2 = ~P1_2; /* P1.2 toggle when interrupt. */
    EXF2 = 0; /* reset interrupt flag */
}
```

2.4 SFR Register Definition

```
/*H*****
**
* NAME: reg_c51.h
* -----
-
* PURPOSE: SFR Description file for 8051 products
*         ON KEIL compiler
*****
*/

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

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

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

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

Sfr (P1 , 0x90);

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

Sfr (P2 , 0xA0);
Sbit (P2_7 , 0xA0, 7);
Sbit (P2_6 , 0xA0, 6);
Sbit (P2_5 , 0xA0, 5);
Sbit (P2_4 , 0xA0, 4);
Sbit (P2_3 , 0xA0, 3);
Sbit (P2_2 , 0xA0, 2);
```

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

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

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

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

```
Sfr (P4 , 0xC0);
Sbit (P4_7 , 0xC0, 7);
Sbit (P4_6 , 0xC0, 6);
Sbit (P4_5 , 0xC0, 5);
Sbit (P4_4 , 0xC0, 4);
Sbit (P4_3 , 0xC0, 3);
Sbit (P4_2 , 0xC0, 2);
Sbit (P4_1 , 0xC0, 1);
Sbit (P4_0 , 0xC0, 0);
```

```
Sfr (P5 , 0xE8);
Sbit (P5_7 , 0xE8, 7);
Sbit (P5_6 , 0xE8, 6);
Sbit (P5_5 , 0xE8, 5);
Sbit (P5_4 , 0xE8, 4);
Sbit (P5_3 , 0xE8, 3);
Sbit (P5_2 , 0xE8, 2);
Sbit (P5_1 , 0xE8, 1);
Sbit (P5_0 , 0xE8, 0);
```

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

```
Sbit (CY , 0xD0 , 7);
Sbit (AC , 0xD0 , 6);
Sbit (F0 , 0xD0 , 5);
```

```

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

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

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

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

Sfr (TMOD , 0x89);

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

Sfr (T2MOD , 0xC9);
Sfr (TL0 , 0x8A);
Sfr (TL1 , 0x8B);
Sfr (TL2 , 0xCC);
Sfr (TH0 , 0x8C);
Sfr (TH1 , 0x8D);
Sfr (TH2 , 0xCD);
Sfr (RCAP2L , 0xCA);
Sfr (RCAP2H , 0xCB);
Sfr (WDRST , 0xA6);
Sfr (WDTPRG , 0xA7);

```

```

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

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

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

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

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

/*----- PCA registers -----*/
Sfr (CCON , 0xD8);
Sfr (CMOD , 0xD9);
Sfr (CH , 0xF9);
Sfr (CL , 0xE9);

```

```

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

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

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

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

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

/*----- Misc -----*/

```

```
Sfr( AUXR , 0x8E);  
Sfr ( AUXR1, 0xA2);  
Sfr ( FCON, 0xD1);
```

```
/*----- E data -----*/
```

```
Sfr ( EECON, 0xD2 );
```

3. Assembler 51 Examples

3.1 Timer 0

3.1.1 Mode 13 bits Timer Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits timer)
; * with a software gate.
; * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
; * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; Timer 0 mode 0 with software gate
        ; GATE0=0; C/T0#=0; M10=0; M00=0;

    MOV TH0,#00h;          /* init values */
    MOV TL0,#00h;
    SETB ET0;              /* enable timer0 interrupt */
    SETB EA;               /* enable interrupts */
    SETB TR0;              /* timer0 run */
    JMP $;                 /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer0:
    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.2 Mode 13 bits Timer Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits timer)
; * with a hardware gate.
; * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
; * of TL0. The upper 3 bits of TL0 are undeterminate and are ignored.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h
begin:
    ANL TMOD,#0F0h;    /* Timer 0 mode 0 with hardware gate */
    ORL TMOD,#08h;      /* GATE0=1; C/T0#=0; M10=0; M00=0; */
    MOV TH0,#00h;       /* init values */
    MOV TL0,#00h;
    SETB ET0;          /* enable timer0 interrupt */
    SETB EA;            /* enable interrupts */
    SETB TR0;          /* timer0 run */
    JMP $;              /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer0:
    CLR TF0;          /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI
end

```

3.1.3 Mode 13 bits Counter Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits counter)
; * with a software gate. The counter count up at each negative transition.
; * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
; * of TL0. The upper 3 bits of TL0 are undetermined and are ignored.
; * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 0 counter with software gate */
    ORL TMOD,#04h;   /* GATE0=0; C/T0#=1; M10=0; M00=0; */

    MOV TH0,#00h;    /* init values */
    MOV TL0,#00h;
    SETB ET0;        /* enable timer0 interrupt */
    SETB EA;          /* enable interrupts */
    SETB TR0;        /* timer0 run */
    JMP $;            /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.4 Mode 13 bits Counter Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 0 (13 bits counter)
; * with a hardware gate. The counter count up at each negative transition.
; * The 13-bits register consist of all 8 bits of TH0 and the lower 5 bits
; * of TL0. The upper 3 bits of TL0 are undetermined and are ignored.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; *                  P3.4(T0) controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 0 counter with hardware gate */
    ORL TMOD,#0Ch;  /* GATE0=1; C/T0#=1; M10=0; M00=0; */

    MOV TH0,#00h;   /* init values */
    MOV TL0,#00h;
    SETB ET0;       /* enable timer0 interrupt */
    SETB EA;         /* enable interrupts */
    SETB TR0;        /* timer0 run */
    JMP $;           /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.1.5 Mode 16 bits Timer Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits timer)
; * with a software gate.
; * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
; * of TL0.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h;        /* Timer 0 mode 1 with software gate */
    ORL TMOD,#01h;         /* GATE0=0; C/T0#=0; M10=0; M00=1; */

    MOV TH0,#00h;          /* init values */
    MOV TL0,#00h;

    SETB ET0;              /* enable timer0 interrupt */
    SETB EA;                /* enable interrupts */
    SETB TR0;              /* timer0 run */
    JMP $;                  /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
; */
it_timer0:

    CLR TF0;               /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.6 Mode 16 bits Timer Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits timer)
; * with a hardware gate.
; * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
; * of TL0.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h;    /* Timer 0 mode 1 with hardware gate */
    ORL TMOD,#09h;     /* GATE0=1; C/T0#=0; M10=0; M00=1; */

    MOV TH0,#00h;      /* init values */
    MOV TL0,#00h;
    SETB ET0;          /* enable timer0 interrupt */
    SETB EA;            /* enable interrupts */
    SETB TR0;          /* timer0 run */
    JMP $;              /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
; */
it_timer0:

    CLR TF0;           /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.1.7 Mode 16 bits Counter Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits counter)
; * with a software gate. The counter count up at each negative transition.
; * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
; * of TL0.
; * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 1 counter with software gate */
    ORL TMOD,#05h; /* GATE0=0; C/T0#=1; M10=0; M00=1; */

    MOV TH0,#00h;      /* init values */
    MOV TL0,#00h;
    SETB ET0;          /* enable timer0 interrupt */
    SETB EA;            /* enable interrupts */
    SETB TR0;          /* timer0 run */
    JMP $;              /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.4(T0) period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.8 Mode 16 bits Counter Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 1 (16 bits counter)
; * with a hardware gate. The counter count up at each negative transition.
; * The 16-bits register consist of all 8 bits of TH0 and all 8 bits
; * of TL0.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; *                  P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 0 counter with hardware gate */
    ORL TMOD,#0Dh;  /* GATE0=1; C/T0#=1; M10=0; M00=1; */

    MOV TH0,#00h;    /* init values */
    MOV TL0,#00h;
    SETB ET0;        /* enable timer0 interrupt */
    SETB EA;          /* enable interrupts */
    SETB TR0;        /* timer0 run */
    JMP $;            /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.4(T0) period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.1.9 Mode 8 bits Auto Reload Timer Software Gated (not used)

```
#define reload_value 0x36; /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
; * timer) with a software gate.
; * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
; * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h;      /* Timer 0 mode 2 with software gate */
    ORL TMOD,#02h;      /* GATE0=0; C/T0#=0; M10=1; M00=0; */

    MOV TH1,reload_value; /* init values */
    MOV TH0,reload_value; /* reload value */
    SETB ET0; /* enable timer0 interrupt */
    SETB EA; /* enable interrupts */
    SETB TR0; /* timer0 run */
    JMP $; /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.10 Mode 8 bits Auto Reload Timer Hardware Gated

```
#define reload_value 0x36 /* reload value exemple */
$INCLUDE (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
; * timer) with a hardware gate.
; * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
; * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 0 with hardware gate */
    ORL TMOD,#0Ah; /* GATE0=1; C/T0#=0; M10=1; M00=0; */

    MOV TH1,reload_value; /* init values */
    MOV TH0,reload_value; /* reload value */
    SETB ET0; /* enable timer0 interrupt */
    SETB EA; /* enable interrupts */
    SETB TR0; /* timer0 run */
    JMP $; /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.11 Mode 8 bits Auto Reload Counter Software Gated (not used)

```
#define reload_value 0x36 /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
; * timer) with a software gate. The counter count up at each negative
; * transition.
; * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
; * of TH0 for the reload value.TH0 is loaded in TL0 at timer0 overflow.
; * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 2 counter with software gate */
    ORL TMOD,#06h; /* GATE0=0; C/T0#=1; M10=1; M00=0; */

    MOV TH1,reload_value; /* init values */
    MOV TH0,reload_value; /* reload value */
    SETB ET0; /* enable timer0 interrupt */
    SETB EA; /* enable interrupts */
    SETB TR0; /* timer0 run */
    JMP $; /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.4(T0)
; * period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.12 Mode 8 bits Auto Reload Counter Hardware Gated

```
#define reload_value 0x36 /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 2 (8 bits auto reload
; * timer) with a hardware gate. The counter count up at each negative
; * transition.
; * The 8-bits register consist of all 8 bits of TL0 and all 8 bits
; * of TH0 for the reload value.TH0 is load in TL0 at timer0 overflow.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; *
; * P3.4(T0) must be controled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h; /* Timer 0 mode 2 counter with hardware gate */
    ORL TMOD,#02h; /* GATE0=1; C/T0#=1; M10=1; M00=0; */

    MOV TH1,reload_value; /* init values */
    MOV TH0,reload_value; /* reload value */
    SETB ET0; /* enable timer0 interrupt */
    SETB EA; /* enable interrupts */
    SETB TR0; /* timer0 run */
    JMP $; /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.4(T0)
; * period
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.1.13 Mode Split Timer Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer)
; * with a software gate.When timer 0 is placed in this mode, it essentially
; * becomes two separate 8-bits timers.
; * One consist of TL0 (8bits) and can be gated by software
; * The other consist of TH0 (8bits),is always in timer mode and cannot be
gated.
; * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1
always running.
; * You can use this mode if you need to have two separate timers and,
; * additionally,a baud rate generator.In such case you can use the timer1 as
baud
; * rate generator and use TH0/TL0 as two separate timers.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h;        /* Timer 0 mode 3 with software gate */
    ORL TMOD,#03h;         /* GATE0=0; C/T0#=0; M10=1; M00=1; */

    MOV TH0,#00h;          /* init values */
    MOV TL0,#00h;
    SETB ET0;              /* enable timer0 interrupt */
    SETB ET1;              /* enable timer1 interrupt */
    SETB EA;               /* enable interrupts */
    SETB TR0;              /* run TL0 */
    SETB TR1;              /* run TH0 */

    JMP $;                 /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer0:
```

```
CLR TF0; /* reset interrupt flag (already done by hardware)*/
CPL P1.0; /* P1.0 toggle when interrupt. */
RETI
```

```
/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
influenced by TH1
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
it_timer1:
```

```
CLR TF1; /* reset interrupt flag (already done by hardware)*/
CPL P1.1; /* P1.1 toggle when interrupt. */
RETI
```

```
end
```

3.1.14 Mode Split Timer Hardware Gated

```
$INCLUDE (reg_c51.INC)
```

```
org 000h
ljmp begin
```

```
org 00Bh
ljmp it_timer0
```

```
org 01Bh
ljmp it_timer1
```

```
/**
 * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer)
 * with a hardware gate. When timer 0 is placed in this mode, it essentially
 * becomes two separate 8-bits timers.
 * One consist of TL0 (8bits) and can be gated by software
 * The other consist of TH0 (8bits), is always in timer mode and cannot be
gated.
 * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1
always running.
 * You can use this mode if you need to have two separate timers and,
 * additionally, a baud rate generator. In such case you can use the timer1 as
baud
 * rate generator and use TH0/TL0 as two separate timers.
 * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
 * FUNCTION_OUTPUTS: void
 */
org 0100h
```

```

begin:
    ANL TMOD,#0F0h;      /* Timer 0 mode 3 with hardware gate */
    ORL TMOD,#0Bh;       /* GATE0=1; C/T0#=0; M10=1; M00=1; */

    MOV TH0,#00h;        /* init values */
    MOV TL0,#00h;
    SETB ET0;           /* enable timer0 interrupt */
    SETB ET1;           /* enable timer1 interrupt */
    SETB EA;            /* enable interrupts */
    SETB TR0;           /* run TL0 */
    SETB TR1;           /* run TH0 */

    JMP $;              /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer0:

    CLR TF0; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

;/**
; * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
; * influenced by TH1
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.1; /* P1.1 toggle when interrupt. */
    RETI

end

```

3.1.15 Mode Split Timer/counter Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer/counter)
; * with a software gate.When timer 0 is placed in this mode, it essentially
; * becomes two separate 8-bits timers.
; * One consist of TL0 (8bits counter) and can be gated by software
; * The other consist of TH0 (8bits),is always in timer mode and cannot be
gated.
; * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1
always running.
; * You can use this mode if you need to have two separate timers and,
; * additionally,a baud rate generator.In such case you can use the timer1 as
baud
; * rate generator and use TH0/TL0 as two separate timers.
; * FUNCTION_INPUTS: P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0F0h;        /* Timer 0 mode 3 with software gate */
    ORL TMOD,#07h;         /* GATE0=0; C/T0#=1; M10=1; M00=1; */

    MOV TH0,#00h;         /* init values */
    MOV TL0,#00h;
    SETB ET0;             /* enable timer0 interrupt */
    SETB ET1;             /* enable timer1 interrupt */
    SETB EA;              /* enable interrupts */
    SETB TR0;             /* run TL0 */
    SETB TR1;             /* run TH0 */

    JMP $;                /* endless */

;/**
; * FUNCTION_PURPOSE: timer0 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
; */
it_timer0:

    CLR TF0;             /* reset interrupt flag (already done by hardware)*/
```

```

CPL P1.0; /* P1.0 toggle when interrupt. */
RETI

;/**
; * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
influenced by TH1
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
; */
it_timer1:

CLR TF1; /* reset interrupt flag (already done by hardware)*/
CPL P1.1; /* P1.1 toggle when interrupt. */
RETI

end

```

3.1.16 Mode Split Timer/Counter Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 00Bh
ljmp it_timer0

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 0 in mode 3 (Split Timer/counter)
; * with a hardware gate. When timer 0 is placed in this mode, it essentially
; * becomes two separate 8-bits timers.
; * One consist of TL0 (8bits counter) and can be gated by software
; * The other consist of TH0 (8bits), is always in timer mode and cannot be
gated.
; * TR0 bit is used to run TL0 and TR1 bit is used to run TH0 and timer1
always running.
; * You can use this mode if you need to have two separate timers and,
; * additionally, a baud rate generator. In such case you can use the timer1 as
baud
; * rate generator and use TH0/TL0 as two separate timers.
; * FUNCTION_INPUTS: P3.2(INT0)=1 : GATE Input
; *
P3.4(T0) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
ANL TMOD, #0F0h; /* Timer 0 mode 3 with hardware gate */
ORL TMOD, #0Fh; /* GATE0=1; C/T0#=1; M10=1; M00=1; */

```

```

MOV TH0,#00h;          /* init values */
MOV TL0,#00h;
SETB ET0;              /* enable timer0 interrupt */
SETB ET1;              /* enable timer1 interrupt */
SETB EA;               /* enable interrupts */
SETB TR0;              /* run TL0 */
SETB TR1;              /* run TH0 */

JMP $;                 /* endless */

/**
 * FUNCTION_PURPOSE: timer0 interrupt
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.4(T0) period
 */
it_timer0:

CLR TF0; /* reset interrupt flag (already done by hardware)*/
CPL P1.0; /* P1.0 toggle when interrupt. */
RETI

/**
 * FUNCTION_PURPOSE: timer1 interrupt is set at TH0 overflow and not
influenced by TH1
 * FUNCTION_INPUTS: void
 * FUNCTION_OUTPUTS: P1.1 toggle period = 2 * 8192 cycles
 */
it_timer1:

CLR TF1; /* reset interrupt flag (already done by hardware)*/
CPL P1.1; /* P1.1 toggle when interrupt. */
RETI

end

```

3.2 Timer 1

3.2.1 Mode 13 bits Timer Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits timer)
; * with a software gate.
; * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
; * of TL1. The upper 3 bits of TL1 are undeterminate and are ignored.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;        /* Timer 1 mode 0 with software gate */
                           /* GATE0=0; C/T0#=0; M10=0; M00=0; */

    MOV TH1,#00h;         /* init values */
    MOV TL1,#00h;
    SETB ET1;             /* enable timer1 interrupt */
    SETB EA;              /* enable interrupts */
    SETB TR1;             /* timer1 run */
    JMP $;                /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer1:

    CLR TF1;             /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.2 Mode 13 bits Timer Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits timer)
; * with a hardware gate.
; * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
; * of TL1. The upper 3 bits of TL1 are undeterminate and are ignored.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;    /* Timer 1 mode 0 with hardware gate */
    ORL TMOD,#80h;    /* GATE0=1; C/T0#=0; M10=0; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 cycles
; */
it_timer1:

    CLR TF1;    /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.2.3 Mode 13 bits Counter Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits counter)
; * with a software gate. The counter count up at each negative transitions
; * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
; * of TL1. The upper 3 bits of TL1 are undetermined and are ignored.
; * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 0 counter with software gate */
    ORL TMOD,#40h; /* GATE0=0; C/T0#=1; M10=0; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.5(T1) period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.4 Mode 13 bits Counter Hardware Gated

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 0 (13 bits counter)
; * with a hardware gate. The counter count up at each negative transitions
; * The 13-bits register consist of all 8 bits of TH1 and the lower 5 bits
; * of TL1. The upper 3 bits of TL1 are undetermined and are ignored.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; *                  P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 0 counter with hardware gate */
    ORL TMOD,#0C0h; /* GATE0=1; C/T0#=1; M10=0; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 8192 * P3.5(T1) period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.5 Mode 16 bits Timer Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits timer)
; * with a software gate.
; * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
; * of TL1.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;      /* Timer 1 mode 1 with software gate */
    ORL TMOD,#10h;      /* GATE0=0; C/T0#=0; M10=0; M00=1; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;

    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
; */
it_timer1:

    CLR TF1;          /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.6 Mode 16 bits Timer Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits timer)
; * with a hardware gate.
; * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
; * of TL1.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;    /* Timer 0 mode 1 with hardware gate */
    ORL TMOD,#90h;    /* GATE0=1; C/T0#=0; M10=0; M00=1; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 cycles
; */
it_timer1:

    CLR TF1;    /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.2.7 Mode 16 bits Counter Software Gated (not used)

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits counter)
; * with a software gate. The counter count up at each negative transition.
; * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
; * of TL1.
; * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 1 counter with software gate */
    ORL TMOD,#50h; /* GATE0=0; C/T0#=1; M10=0; M00=1; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.5(T1) period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.8 Mode 16 bits Counter Hardware Gated

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1

;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 1 (16 bits counter)
; * with a hardware gate. The counter count up at each negative transitions
; * The 16-bits register consist of all 8 bits of TH1 and all 8 bits
; * of TL1.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; *                  P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 0 counter with hardware gate */
    ORL TMOD,#0D0h; /* GATE0=1; C/T0#=1; M10=0; M00=1; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * 65536 * P3.5(T1) period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end

```

3.2.9 Mode 8 bits Auto Reload Timer Software Gated (not used)

```
#define reload_value 0x36; /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
; * timer) with a software gate.
; * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
; * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;      /* Timer 1 mode 2 with software gate */
    ORL TMOD,#20h;      /* GATE0=0; C/T0#=0; M10=1; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
; */
it_timer1:

    CLR TF1;          /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.10 Mode 8 bits Auto Reload Timer Hardware Gated

```
#define reload_value 0x36; /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
; * timer) with a hardware gate.
; * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
; * of TH1 for the reload value.TH1 is load in TL1 at timer1 overflow.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh;    /* Timer 1 mode 0 with hardware gate */
    ORL TMOD,#0A0h;    /* GATE0=1; C/T0#=0; M10=1; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) cycles
; */
it_timer1:

    CLR TF1;    /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.11 Mode 8 bits Auto Reload Counter Software Gated (not used)

```
#define reload_value 0x36; /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
; * timer) with a software gate. The counter count up at each negative
; * transition.
; * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
; * of TH1 for the reload value. TH1 is load in TL1 at timer1 overflow.
; * FUNCTION_INPUTS: P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 2 counter with software gate */
    ORL TMOD,#20h; /* GATE0=0; C/T0#=1; M10=1; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.5(T1)
; * period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.2.12 Mode 8 bits Auto Reload Counter Hardware Gated

```
#define reload_value 0x36; /* reload value exemple */

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 01Bh
ljmp it_timer1
;/**
; * FUNCTION_PURPOSE: This file set up timer 1 in mode 2 (8 bits auto reload
; * timer) with a hardware gate. The counter count up at each negative
; * transition.
; * The 8-bits register consist of all 8 bits of TL1 and all 8 bits
; * of TH1 for the reload value. TH1 is load in TL1 at timer1 overflow.
; * FUNCTION_INPUTS: P3.3(INT1)=1 : GATE Input
; *
; * P3.5(T1) must be controlled by an external clock
; * FUNCTION_OUTPUTS: void
; */
org 0100h

begin:
    ANL TMOD,#0Fh; /* Timer 1 mode 2 counter with hardware gate */
    ORL TMOD,#20h; /* GATE0=1; C/T0#=1; M10=1; M00=0; */

    MOV TH1,#00h;      /* init values */
    MOV TL1,#00h;
    SETB ET1;          /* enable timer1 interrupt */
    SETB EA;           /* enable interrupts */
    SETB TR1;          /* timer1 run */
    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer1 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0 toggle period = 2 * (256-reload_value) * P3.5(T1)
; * period
; */
it_timer1:

    CLR TF1; /* reset interrupt flag (already done by hardware)*/
    CPL P1.0; /* P1.0 toggle when interrupt. */
    RETI

end
```

3.3 Timer 2

3.3.1 Mode 16 bits up/down Auto reload Timer

```
#define MSB_reload_value 0x36 /* msb reload value exemple */
#define LSB_reload_value 0x36 /* lsb reload value exemple */
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 02Bh
ljmp it_timer2
/**
; * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits auto-reload
; * up/down counting timer).
; * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
; * of TL2. The EXF2 bit toggles when timer2 overflow or underflow occurs.
; * EXF2 does not generate interrupt. This bit can be used to provide 17-bit
resolution
; * FUNCTION_INPUTS: P1.1(T2EX)=0 for down counting or 1 for up counting.
; * FUNCTION_OUTPUTS: void
; */
org 0100h
begin:
    ANL T2MOD,#0FCh; /* T2OE=0;DCEN=1; */
    ORL T2MOD,#01h;
    CLR EXF2;          /* reset flag */
    CLR TCLK;
    CLR RCLK;          /* disable baud rate generator */
    CLR EXEN2;         /* ignore events on T2EX */
    MOV TH2,MSB_reload_value; /* Init msb_value */
    MOV TL2,LSB_reload_value; /* Init lsb_value */
    MOV RCAP2H,MSB_reload_value; /* reload msb_value */
    MOV RCAP2L,LSB_reload_value; /* reload lsb_value */
    CLR C_T2;          /* timer mode */
    CLR CP_RL2;        /* reload mode */
    SETB EA;           /* interrupt enable */
    SETB ET2;          /* enable timer2 interrupt */
    SETB TR2;          /* timer2 run */
    JMP $;             /* endless */
/**
; * FUNCTION_PURPOSE: timer2 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * (65536-reload_value) cycles
; */
it_timer2:
    CLR TF2; /* reset interrupt flag */
    CPL P1.2; /* P1.2 toggle when interrupt. */
    RETI

end
```



3.3.2 Mode 16 bits up/down Auto reload Counter

```
#define MSB_reload_value 0x36 /* msb reload value exemple */
#define LSB_reload_value 0x36 /* lsb reload value exemple */
$INCLUDE (reg_c51.INC)

org 000h
ljmp begin

org 02Bh
ljmp it_timer2
;/**
; * FUNCTION_PURPOSE: This file set up timer 2 in mode 0 (16 bits auto-reload
; * up/down counter).
; * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
; * of TL2. The EXF2 bit toggles when timer2 overflow or underflow occurs.
; * EXF2 does not generate interrupt. This bit can be used to provide 17-bit
resolution
; * FUNCTION_INPUTS: P1.0(T2) must be controlled by an external clock
; * P1.1(T2EX)=0 for down counting or 1 for up counting.
; * FUNCTION_OUTPUTS: void
; */
org 0100h
begin:
ANL T2MOD,#0FCh; /* T2OE=0;DCEN=1; */
ORL T2MOD,#01h;
CLR EXF2; /* reset flag */
CLR TCLK;
CLR RCLK; /* disable baud rate generator */
CLR EXEN2; /* ignore events on T2EX */
MOV TH2,MSB_reload_value; /* Init msb_value */
MOV TL2,LSB_reload_value; /* Init lsb_value */
MOV RCAP2H,MSB_reload_value; /* reload msb_value */
MOV RCAP2L,LSB_reload_value; /* reload lsb_value */
SETB C_T2; /* counter mode */
CLR CP_RL2; /* reload mode */
SETB EA; /* interrupt enable */
SETB ET2; /* enable timer2 interrupt */
SETB TR2; /* timer2 run */
JMP $; /* endless */
;/**
; * FUNCTION_PURPOSE: timer2 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * (65536-reload_value) * P1.0(T2)
period
; */
it_timer2:
CLR TF2; /* reset interrupt flag */
CPL P1.2; /* P1.2 toggle when interrupt. */
RETI

end
```

3.3.3 Mode 16 bits Capture Timer

```
$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 02Bh
ljmp it_timer2
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 1 (capture mode)
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2.
 * FUNCTION_INPUTS: P1.1(T2EX) is a periodic signal
 * FUNCTION_OUTPUTS: void
 */
begin:
    ANL T2MOD, #0FCh; /* T2OE=0;DCEN=1; */
    CLR EXF2;          /* reset flag */
    CLR TCLK;
    CLR RCLK;          /* disable baud rate generator */

    SETB EXEN2;        /* enable events detect on T2EX */
    CLR C_T2;          /* timer mode */
    SETB CP_RL2;       /* capture mode */
    SETB EA;           /* interrupt enable */
    SETB ET2;          /* enable timer2 interrupt */
    SETB TR2;          /* timer2 run */

    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer2 interrupt.
; * EXF2 is set at each negative transitions on P1.1(T2EX) and load TL2 in
; * RCAP2L and load TH2 in RCAP2H
; * TF2 is set at each timer2 overflows
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
it_timer2:
    CLR TF2; /* reset interrupt flag */
    CLR EXF2; /* reset interrupt flag */
; You can calculate a time between two negative transitions an P1.1(T2EX)
; time = (second_capture - first capture + 65536 * number_of_overflows) cycles

    RETI

end
```

3.3.4 Mode 16 bits Capture Counter

```

$INCLUDE    (reg_c51.INC)

org 000h
ljmp begin

org 02Bh
ljmp it_timer2
/**
 * FUNCTION_PURPOSE: This file set up timer 2 in mode 1 (capture mode)
 * The 16-bits register consist of all 8 bits of TH2 and all 8 bits
 * of TL2.
 * FUNCTION_INPUTS: P1.1(T2EX) is a periodic signal
 *                  P1.0(T2) must be controlled by an external clock
 * FUNCTION_OUTPUTS: void
 */
begin:
    ANL T2MOD, #0FCh; /* T2OE=0; DCEN=1; */
    CLR EXF2;          /* reset flag */
    CLR TCLK;
    CLR RCLK;          /* disable baud rate generator */

    SETB EXEN2;        /* enable events detect on T2EX */
    SETB C_T2;         /* counter mode */
    SETB CP_RL2;       /* capture mode */
    SETB EA;           /* interrupt enable */
    SETB ET2;          /* enable timer2 interrupt */
    SETB TR2;          /* timer2 run */

    JMP $;             /* endless */

;/**
; * FUNCTION_PURPOSE: timer2 interrupt.
; * EXF2 is set at each negative transitions on P1.1(T2EX) and load TL2 in
; * RCAP2L and load TH2 in RCAP2H
; * TF2 is set at each timer2 overflows
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
it_timer2:
    CLR TF2; /* reset interrupt flag */
    CLR EXF2; /* reset interrupt flag */
; You can calculate a time between two negative transitions on P1.1(T2EX)
; time = (second_capture - first capture + 65536 * number_of_overflows) *
; P1.0(T2)period

    RETI

end

```

3.3.5 Clock-out Mode

```
#define MSB_reload_value 0x36 /* msb reload value exemple */
#define LSB_reload_value 0x36 /* lsb reload value exemple */
$INCLUDE (reg_c51.INC)

org 000h
ljmp begin

org 02Bh
ljmp it_timer2
/**
; * FUNCTION_PURPOSE: This file set up timer 2 in mode 1 (clock-out mode and
; * negative transition detector).
; * The 16-bits register consist of all 8 bits of TH2 and all 8 bits of TL2.
; * TF2 does not generate interrupt.
; * A negative transition on P1.1(T2EX) generate an interrupt.
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.0(T2) as clock output : Fout = Fperiph / (2*(65536-
RCAP2)).
; */
org 0100h
begin:
    ANL T2MOD,#0FEh;      /* T2OE=1;DCEN=0; */
    ORL T2MOD,#02h;
    CLR EXF2;              /* reset flag */
    CLR TCLK;
    CLR RCLK;              /* disable baud rate generator */
    SETB EXEN2;           /* enable events on T2EX */
    MOV TH2,MSB_reload_value; /* Init msb_value */
    MOV TL2,LSB_reload_value; /* Init lsb_value */
    MOV RCAP2H,MSB_reload_value; /* reload msb_value */
    MOV RCAP2L,LSB_reload_value; /* reload lsb_value */
    CLR C_T2;              /* timer mode */
    CLR CP_RL2;            /* reload mode */
    SETB EA;               /* interrupt enable */
    SETB ET2;              /* enable timer2 interrupt */
    SETB TR2;              /* timer2 run */
    JMP $;                 /* endless */
/**
; * FUNCTION_PURPOSE: timer2 interrupt
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: P1.2 toggle period = 2 * P1.1(T2EX) period
; */
it_timer2:
    CLR TF2; /* reset interrupt flag */
    CPL P1.2; /* P1.2 toggle when interrupt. */
    RETI

end
```

3.4 SFR Register Definition

```
$SAVE
$NOLIST
```

```
P0      DATA    80H
TCONDATA88H
;--- TCON Bits ---
TF1      BIT      8FH
TR1      BIT      8EH
TF0      BIT      8DH
TR0      BIT      8CH
IE1      BIT      8BH
IT1      BIT      8AH
IE0      BIT      89H
IT0      BIT      88H

P1      DATA    90H

SCON     DATA    98H
;--- SCON Bits ----
SM0      BIT      9FH
SM1      BIT      9EH
SM2      BIT      9DH
REN      BIT      9CH
TB8      BIT      9BH
RB8      BIT      9AH
TI       BIT      99H
RI       BIT      98H

P2      DATA    0A0H
IEN0     DATA    0A8H
;--- IEN0 Bits -----
EA       BIT0AFH
EC       BIT0AEH
ET2      BIT0ADH
ES       BIT0ACH
ET1      BIT0ABH
EX1      BIT0AAH
ET0      BIT0A9H
EX0      BIT0A8H

P3      DATA    0B0H
;--- P3 Bits -----
RD       BIT      0B7H
WR       BIT      0B6H
T1       BIT      0B5H
T0       BIT      0B4H
INT1     BIT      0B3H
```

INT0	BIT	0B2H
TXD	BIT	0B1H
RXD	BIT	0B0H

P4	DATA	0C0H
P5	DATA	0E8H

```

IPL0DATA0B8H
;--- IPL0 Bits -----
PPCL  BIT0BEH
PT2L  BIT0BDH
PSL   BIT0BCH
PT1L  BIT0BBH
PX1L  BIT0BAH
PT0L  BIT0B9H
PX0L  BIT0B8H

```

T2CON	DATA	0C8H
-------	------	------

```

;--- T2CON bits ----
TF2    BIT    0CFH
EXF2   BIT    0CEH
RCLK   BIT    0CDH
TCLK   BIT    0CCH
EXEN2  BIT    0CBH
TR2    BIT    0CAH
C_T2   BIT    0C9H
CP_RL2 BIT    0C8H

```

PSW	DATA	0D0H
-----	------	------

```

;--- PSW bits -----
CY     BIT    0D7H
AC     BIT    0D6H
F0     BIT    0D5H
RS1    BIT    0D4H
RS0    BIT    0D3H
OV     BIT    0D2H
P      BIT    0D0H

```

```

CCONDATA0D8H
;--- CCON bits -----
CF     BIT    0DFH
CR     BIT    0DEH
CCF4   BIT    0DCH
CCF3   BIT    0DBH
CCF2   BIT    0DAH
CCF1   BIT    0D9H
CCF0   BIT    0D8H

```

ACC DATA 0E0H
B DATA 0F0H

SP DATA 81H
DPL DATA 82H
DPH DATA 83H
PCON DATA 87H

TMOD DATA 89H
TL0 DATA 8AH
TL1 DATA 8BH
TH0 DATA 8CH
TH1 DATA 8DH
AUXRDATA08EH
CKCON0DATA08Fh

SBUF DATA 99H
;-- Baud Rate generator
BRL DATA09AH
BDRCON DATA 09BH
;--- Keyboard
KBLSDATA09CH
KBEDATA09DH
KBFDATA09EH

;--- Watchdog timer
WDRSTDATA0A6H
WDTPRG DATA0A7H

SADDRDATA0A9H
CKCON1DATA0AFH

IEN1DATA0B1H
IPL1DATA0B2H
IPH1DATA0B3H
IPH0DATA0B7H

SADENDATA0B9H

```

T2MODDATA    0C9h
RCAP2L  DATA    0CAH
RCAP2H  DATA    0CBH
TL2     DATA    0CCH
TH2     DATA    0CDH

```

```

CMODDATA0D9H
CCAPM0DATA0DAH
CCAPM1DATA0DBH
CCAPM2DATA0DCH
CCAPM3DATA0DDH
CCAPM4DATA0DEH

```

```

CHDATA0F9H
CCAP0HDATA0FAH
CCAP1HDATA0FBH
CCAP2HDATA0FCH
CCAP3HDATA0FDH
CCAP4HDATA0FEH

```

```

CLDATA0E9H
CCAP0LDATA0EAH
CCAP1LDATA0EBH
CCAP2LDATA0ECH
CCAP3LDATA0EDH
CCAP4LDATA0EEH

```

```

; SPI
SPCON    DATA    0C3H
SPSTA    DATA    0C4H
SPDAT    DATA    0C5H

```

```

; TWI
PI2DATA    0F8h
SSCONDATA093H
SSCSDATA094H
SSDATDATA095H
SSADRDATA096H
PI2_0BIT0F8H
PI2_1BIT0F9H

```

```

; Clock Control
OSCCONDATA086H
CKSELDATA085H
CKRLDATA097H

```

```

; MISC
AUXR1DATA0A2H

```



```
; Flash control  
FCON DATA 0D1H
```

```
;EEData  
EECONDATA0D2H
```

```
$RESTORE
```

Table of Contents

1. Introduction	1
1.1 References	1
2. C Examples.....	2
Mode Split Timer Hardware Gated 16	
Mode Split Timer/Counter Hardware Gated 19	
2.2 Timer 1	21
2.3 Timer 2	32
2.4 SFR Register Definition	41
3. Assembler 51 Examples.....	47
Mode 16 bits Timer Hardware Gated 70	
Mode 16 bits Counter Software Gated (not used) 71	
3.3 Timer 2	77



Atmel Corporation

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

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

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

© **Atmel Corporation 2004. All rights reserved.** Atmel® and combinations thereof are the registered trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be the trademarks of others.



Printed on recycled paper.