NEC

Application Note

The Timer Array Unit of the 78K0R Microcontrollers

Document No. U18771EE2V0AN00

Date published September 2009

Printed in Germany

Legal Notes

- **The information in this document is current as of July, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

- The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime

systems, safety equipment and medical equipment (not specifically designed for life support).
"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives anddistributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and otherlegal issues may also vary from country to country.

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668, Japan
Tel: 044 4355111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554,
U.S.A.
Tel: 408 5886000
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211 65030
http://www.eu.necel.com/

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908 691133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01 30675800

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 6387200

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02 667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven,
The Netherlands
Tel: 040 2654010

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27
ZhiChunLu Haidian District,
Beijing 100083, P.R.China
Tel: 010 82351155
http://www.cn.necel.com/

NEC Electronics Shanghai Ltd.
Room 2511-2512, Bank of China
Tower,
200 Yincheng Road Central,
Pudong New Area,
Shanghai 200120, P.R. China
Tel: 021 58885400
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886 9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R.O.C.
Tel: 02 27192377

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253 8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku, Seoul,
135-080, Korea Tel: 02-558-3737
http://www.kr.necel.com/

Preface

The 78K0R 16 bit microcontroller family introduces a new timer structure for the classical timing applications. The **Timer Array Unit** is composed of eight 16-bit timers. Each 16-bit timer is called a channel and can be used as an independent timer. In addition, two or more “channels” can be used to create a high-accuracy timer.

The following timing functions can be realised with the different channels:

Independent Operation Function	Combination Operation Function
• Interval timer • Square wave output • External event counter • Divider function • Input pulse interval measurement • Measurement of high-/low-level width of input signal	• PWM output • One-shot pulse output • Multiple PWM output

Channel 7 of the TAU also has an alternative role: it can be used to realise LIN-bus reception processing in combination with UART3 of serial array unit 1.

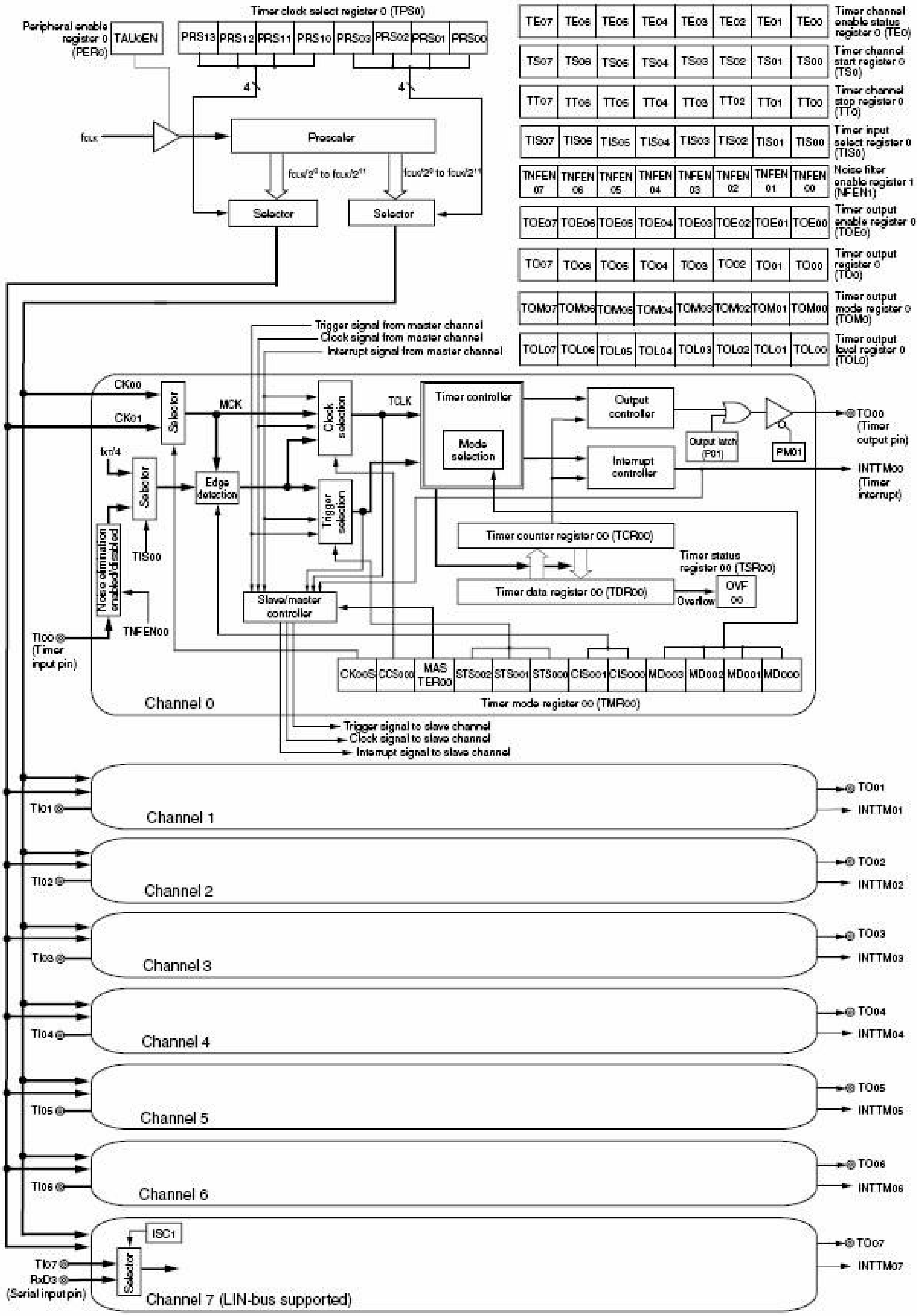


Figure -1 Block Diagram of Timer Array Unit

System Clock Settings

Before using the **TAU** the clock operation needs to be configured. This program function shows a general initialization of the clock, which could be done in many applications.

```
//-----------------------------------------------------------------------------------
// Module:   ClockGeneratorSetting
// Function:  general initialization of clock
//-----------------------------------------------------------------------------------
void ClockGeneratorSetting (void) {

        OSMC = 0x01;        // Operation speed mode control register
                            // frequency higher than 10MHz

        OSTS = 0x07;        // Set osc. stabilization time selection to 2^18/fx

        CMC = 0x51;         // Clock operation mode register
                            // X1 osc. mode, XT1 osc. mode, fx > 10MHz

        CKC = 0x08;         // System clock control register : fclk = fih

        CSC   = 0x00;       // Enable X1 , XT1 operation

        while(OSTC < 0xFF) {        // Wait until fX1 clock stabilization time has been elapsed
        __no_operation();
        }

        CKC = 0x18;         // System clock control register fclk = fmx = 20MHz

        CSC = 0x01;         // Stop internal high speed oscillator

}
//-----------------------------------------------------------------------------
```

Default settings of TAU registers

All Special Function Registers are initialized at their default values in this function. They will be changed depending of the used timer(s).

```
// -----------------------------------------------------------------------------------------------
// Module :  Default_TAU_Setting
// -----------------------------------------------------------------------------------------------
void Default_TAU_Setting (void) {

        // CONTROL REGISTERS OF T.A.U SETTING BLOCK
        PER0_bit.no0 = 0;    // Disable input clock for timer array

        TPS0 = 0x0000;       // Timer clock select register
                             // By default, the main clock is used > No prescaler

        TS0 = 0x0000;        // Timer channel start register
                             // By default, the start triggers do not need to be generated

        TT0 = 0x0000;        // Timer channel stop register
                             // By default, the stop triggers do not need to be generated

        TIS0 = 0x00;         // Timer input select register
                            // By default, all the inputs for the timers are the timer input pins.

        TOE0 = 0x0000;       // Timer output enable register
                             // By default, the timer output pins are controlled by software, so
                             // the value of the register TO0 can be written

        TO0 = 0x0000;        // Timer output register
                             // By default, the timer output values are "0", these values can be
                             // choose only when the output values are controlled by software.
```

```
        TOL0 = 0x0000;          // Timer output level register
                                // By default, the timer output levels are active high

        TOM0 = 0x0000;          // Timer output mode register
                                // By default, the toggle outputs are produced only with the
                                // interrupt, and do not depend of the state of the other channels

        NFEN1 = 0x00;           // Noise filter enable register 1
                                // By default, the noise filters are not used for the timer inputs


        // CONTROL REGISTERS OF EACH CHANNEL
        // Timer mode registers > These are used to set the operation mode of each channel
        TMR00 = 0x0000;
        TMR01 = 0x0000;
        TMR02 = 0x0000;
        TMR03 = 0x0000;
        TMR04 = 0x0000;
        TMR05 = 0x0000;
        TMR06 = 0x0000;
        TMR07 = 0x0000;

        // Input switch control register for channel 7 only
        // By default, the channel 7 is not used to implement LIN-bus communication
        ISC = 0x00;

        // Timer data registers > These are used to set the values that need to be counted.
        TDR00 = 0x0000;
        TDR01 = 0x0000;
        TDR02 = 0x0000;
        TDR03 = 0x0000;
        TDR04 = 0x0000;
        TDR05 = 0x0000;
        TDR06 = 0x0000;
        TDR07 = 0x0000;

}
// ------------------------------------------------------------------------------------------------
```

Table of Contents

Chapter 1 Operation of the Timer Array Unit as Independent Channels

All 16-bit timers of the **TAU** can be used in the following modes. For all the modes detailed below, only one timer channel is sufficient:

- Interval timer
- Square wave output
- External event counter
- Divider function
- Input pulse interval measurement
- Measurement of high-/low-level width of input signal

Each mode will be detailed by using an example.

1.1 Interval Timer

1.1.1 Program description

This program shows the configuration of the 16-bit timer TM02 and TM03 in order to operate in the interval timer mode. To count a specific period, the timer will be started by software and the interval update controlled by an interrupt service routine.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

Timer Mode Registers TMR02 and TMR03 are used to select the chosen clock and to configure the timer for the interval mode.

Timer Data Register TDR02 and TDR03 are used to set the period of the timers.

$$\textit{Interval\_measured} = \textit{Period}_{CKS00/CKS01} \times (\textit{Value}_{TDR02/TDR03} + 1)$$

When all the registers are configured, the timers can be started with the register TS0 setting bits TS0L_bit.no2 and TS0L_bit.no3 .

To count a time interval without using interrupts, after timer TM02 have been started, it is necessary to poll the interrupt flag: "*while (!TMIF02)*" and then to clear this flag and stop the timer.

To count a time interval using interrupts, the first thing to do is to enable the interrupt for the timer TM03 clearing TMMK03 after having started the timer.

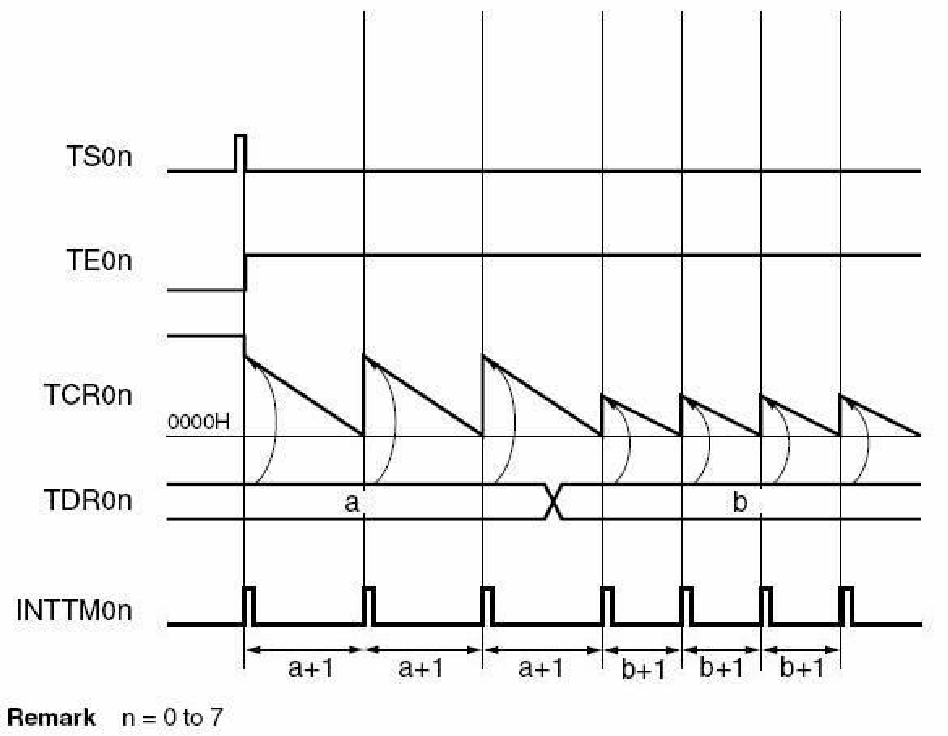
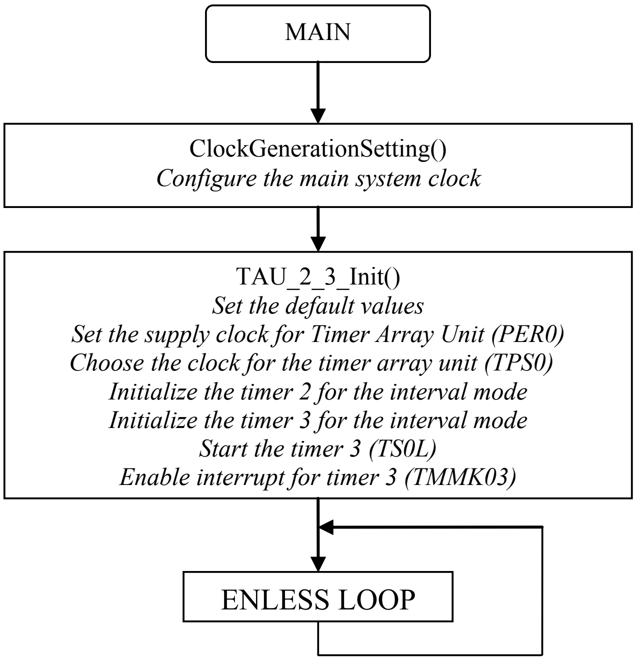


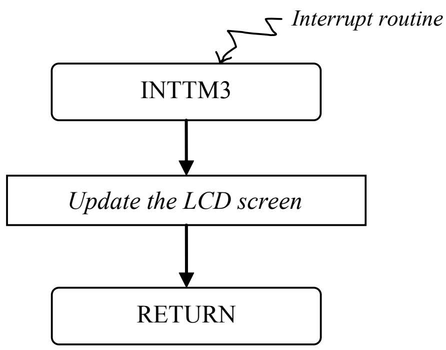
Figure 1-1 Interval mode diagram

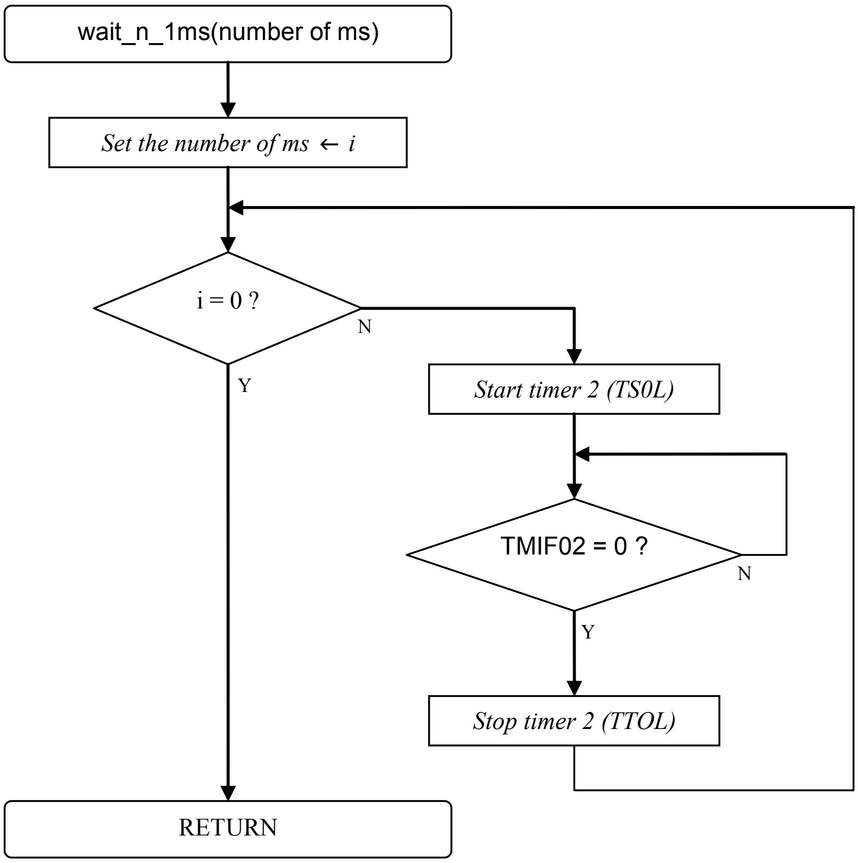
1.1.2 Program specification

```
Channel timer used:          2 without interrupt
                             3 with interrupt
Count clock frequency:      156.2 kHz for Timer 2 (at 20 MHz main system clock)
                            20 MHz for Timer 3 (at 20 MHz main system clock)
Interval measured:           1 ms for the both channel
Compare valueS:              TDR02 = 156d = 0x009C
                             TDR03 = 19 999d = 0x4E1F
Available interrupts:        TM02 interrupt (INTTM02)
                             TM03 interrupt (INTTM03)
```

1.1.3 Software flow chart







1.1.4 Code

```
//--------------------------------------------------------------------------------------------
// Module:   TAU_2_3_Init
// Function: initialization of clock and timer
//--------------------------------------------------------------------------------------------
void TAU_2_3_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0070;             // Timer clock selection register for the timer channel
              // =    0000    0000    0111    0000
              //                      ||||    ++++ --    CKS00 = 20 MHz
              //                      ++++ -----------   CKS01 = 156.2 kHz


    // TIMER 2 INITIALIZATION
    TMR02 = 0x8000;     // Timer 2 mode register
            // =    1000    0000    0000    0000
            //      |   |    ||||    ||      ||| + --    No operation at the start
            //      |   |    ||||    ||      +++ ----    Interval timer mode
            //      |   |    ||||    ++ ------------     Input Timer not used
            //      |   |    |+++ ------------------     Software start selected
            //      |   |    + ---------------------     Only one timer used in this mode
            //      |   + -------------------------      Count on general clock edge
            //      + -----------------------------      Selected clock = CKS1

    TDR02 = 0x009C;          // Timer 2 data register
                             // Interval time = 1 ms = (1/156 200) * (TDR02 + 1)
```

```
    // TIMER 3 INITIALIZATION
    TMR03 = 0x0000;     // Timer 3 mode register
                  // =    0000    0000    0000    0000
                  //     |   |    ||||    ||      ||| + --    No operation at the start
                  //     |   |    ||||    ||      +++ ----    Interval timer mode
                  //     |   |    ||||    ++ --------------   Input Timer not used
                  //     |   |    |+++ --------------------   Software start selected
                  //     |   |    + ----------------------   Only one timer used in this mode
                  //     |   + ---------------------------   Count on general clock edge
                  //     + -------------------------------   Selected clock = CKS0

    TDR03 = 0x4E1F;          // Timer 3 data register
                             // Interval time = 1 ms = (1/20 000 000) * (TDR03 + 1)

    TS0L_bit.no3 = 1;        // Start Timer 3

    TMMK03 = 0;              // Enable Timer channel 3 interrupt -> routine available
    TMPR003 = 1 ;            // Set priority level of this interrupt
    TMPR103 = 1 ;            // -> Level 3 = low priority level
    TMIF03 = 0;              // Clear interrupt flag
    IE = 1;                  // Enable global interrupt

}


//-----------------------------------------------------------------------------------------------
// Module:   wait_n_ms
// Function: waits for n * 1ms, using timer 2
//-----------------------------------------------------------------------------------------------
void wait_n_1ms(unsigned char n)
{
    unsigned char i;

    for (i=0; i < n; i++)
    {
        TS0L_bit.no2 = 1;               // Start Timer 2
        while(!TMIF02);                 // Wait for TM02 Interrupt
        TMIF02 = 0;                     // Clear interrupt flag
        TT0L_bit.no2 = 1;               // Stop Timer 2
    }
}


//-----------------------------------------------------------------------------------------------
// ISR:          isr_INTTM03
// Function:     Timer channel 3 interrupt service routine
//               Timer3 interval time = 1 ms
//-----------------------------------------------------------------------------------------------
#pragma vector = INTTM03_vect
__interrupt void isr_INTTM03(void)
{
    // Clear and set up the LCD screen
}
```

1.2 Square Wave Output

1.2.1 Program description

This program shows the configuration of the 16-bit timer TM07 in order to operate in the square wave output mode.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

Timer Mode Register TMR07 is used to select the chosen clock and to configure for the square wave output mode.

Timer Data Register TDR07 is used to set the period of the square wave output.

$$Period_{square\_signal} = 2 \times Period_{CKS00/CKS01} \times (Value_{TMR07} + 1)$$

The timer has to be configured in order to toggle between high level and low level and consequently to create the square wave signal wanted (TOE0)

When all the registers are configured, the timer can be started with the register TS0 setting the bit TS0L_bit.no7.

To use the interrupt, it is necessary to enable the interrupt for the timer TM07 clearing TMMK07 after having started the timer.

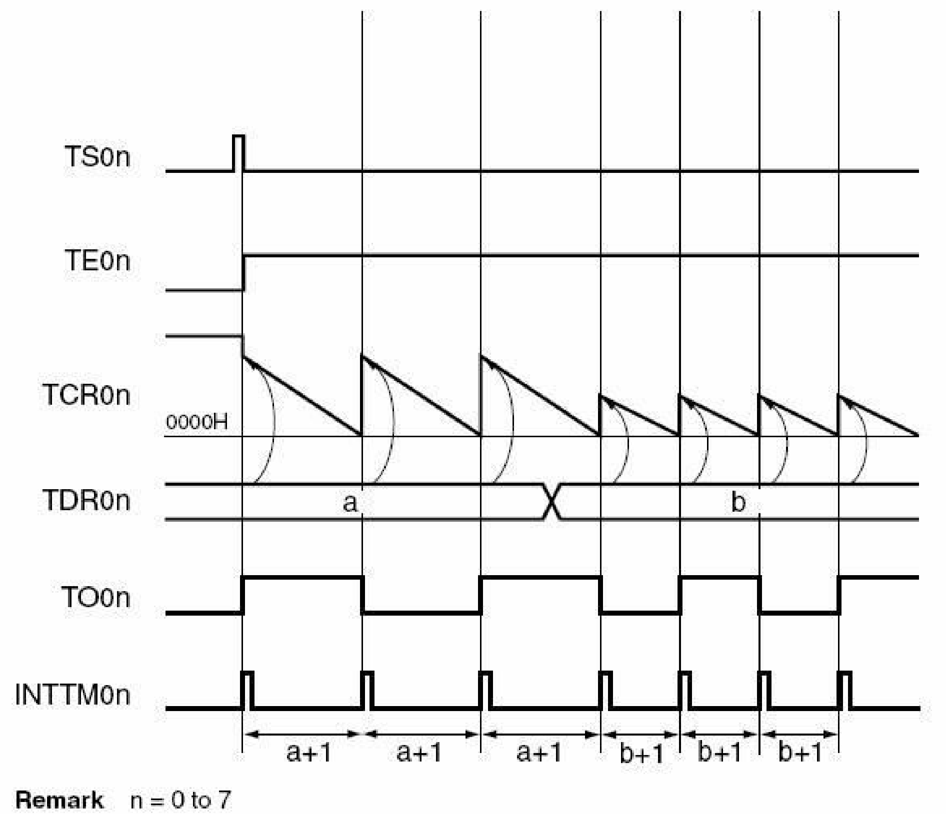


Figure 1-5 Square wave ouput mode diagram

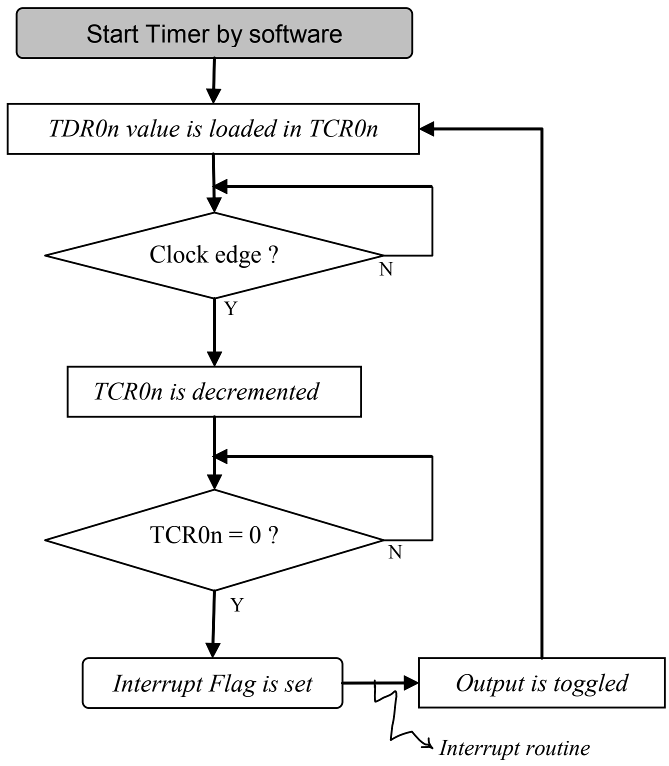
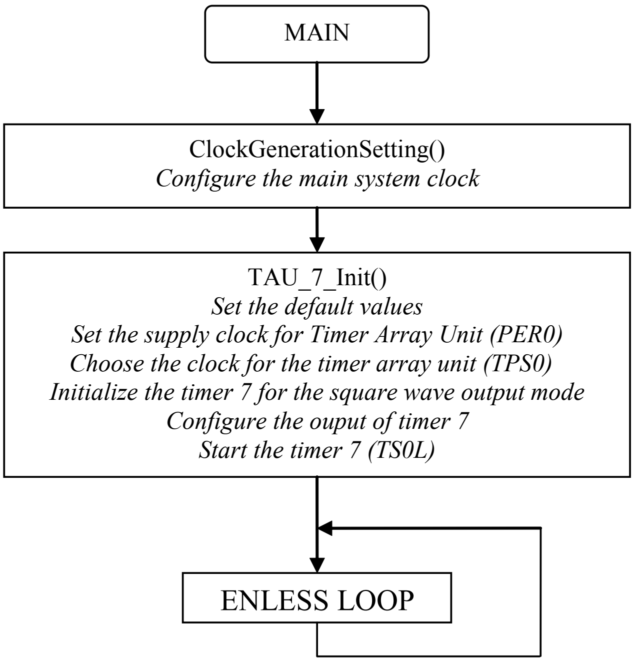


Figure 1-6 Function of the square wave output mode

1.2.2 Program specification

```
Channel timer used:       7
Count clock frequency: 156.2 kHz (at 20 MHz main system clock)
Square wave period:     50 ms
Interval measured:      25 ms
Compare value: TDR07 = 3904d = 0x0F40
Pins used in program:  P14.5/TO07 to output the square wave signal
Available interrupt:   TM07 interrupt (INTTM07)
```

1.2.3 Software flow chart



1.2.4 Code

```
//-------------------------------------------------------------------------------------
// Module:   TAU_7_Init
// Function: initialization of clock and timer
//-------------------------------------------------------------------------------------
void TAU_7_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0070;    // Timer clock selection register for the timer channel
                  // =    0000    0000    0111    0000
                  //                      ||||    ++++ --    CKS00 = 20 MHz
                  //                      ++++ ----------    CKS01 = 156.2kHz

    // TIMER 7 INITIALIZATION
    TMR07 = 0x8001;    // Timer 7 mode register
                  // =    1000    0000    0000    0001
                  //      |  |    ||||    ||      |||+ --    Operation at the start (int+toggle)
                  //      |  |    ||||    ||      +++ ---    Interval timer mode
                  //      |  |    ||||    ++ ------------    Input Timer not used
                  //      |  |    |+++ ------------------    Software start selected
                  //      |  |    + --------------------    Only one timer used in this mode
                  //      |  + -------------------------    Count on general clock edge
                  //      + ----------------------------    Selected clock = CKS1

    TDR07 = 0x0F40;    // Timer 7 data register
                       // Output period = 50 ms = 2 * (1/156 200) * (TDR07 + 1)

    // OUPUT CONFIGURATION
    TOM0 = 0x0000;    // Timer output mode register for output 7
                      // Set toggle mode for this timer
```

```
    TOL0 = 0x0000;     // Timer output level register for output 7

    TO0 = 0x0000;      // Timer output register for output 7

    TOE0 |= 0x0080;    // Timer output enable register for output 7
                       // TO0n operation enabled by count operation: TO0n pin outputs
                       // the square-wave

    // PORT INITIALIZATION linked with output
    PM14_bit.no5 = 0;
    P14_bit.no5 = 0;

    TS0L_bit.no7 = 1;  // Start Timer 7

    //TMMK07 = 0;       // Enable Timer channel 7 interrupt -> routine available
                        // Use only to do actions in the interrupt: it will not affect the
                        // output square wave

}
```

1.3 External Event Counter

1.3.1 Program description

This program shows the configuration of the 16-bit timer TM00 in order to operate in the external event counter mode.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

Timer Mode Register TMR00 is used to select the chosen clock and to configure for the external event counter mode. It also is used to select what type of event will be counted, rising edge, falling edge… and the Timer Data Register TDR00 selects the number of events to be counted.

$$Number_{event\_counted} = Value_{TDRO0} + 1$$

When all the registers are configured, the timer can be started with the register TS0 setting the bit TS0L_bit.no0.

It is necessary to enable the interrupt for the timer TM00 clearing TMMK00 after having started the timer.

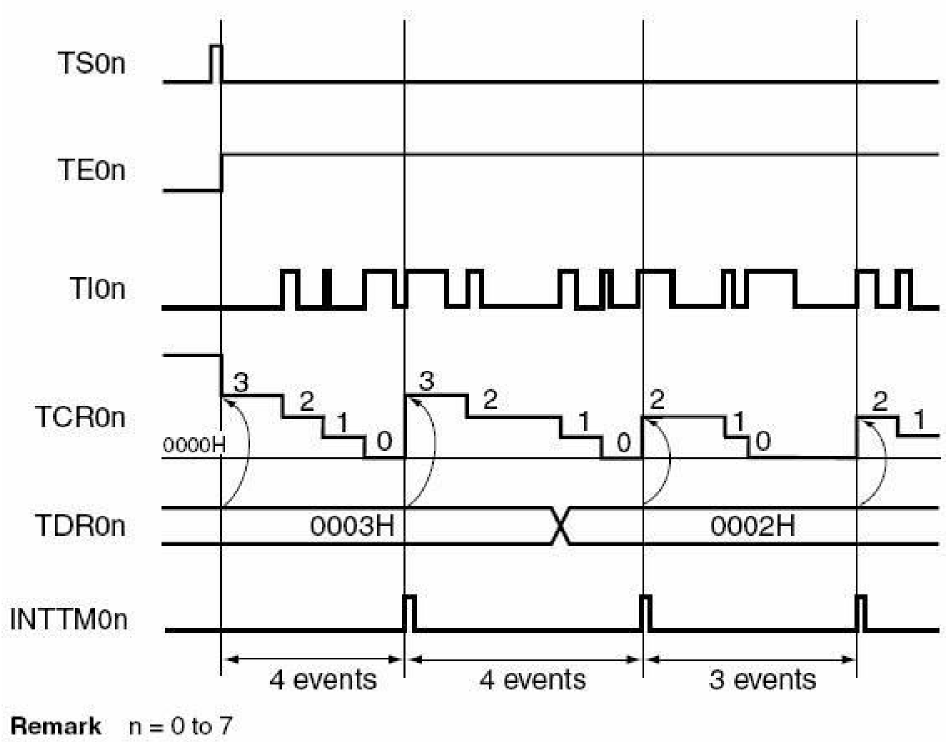


Figure 1-8 External event counter mode diagram

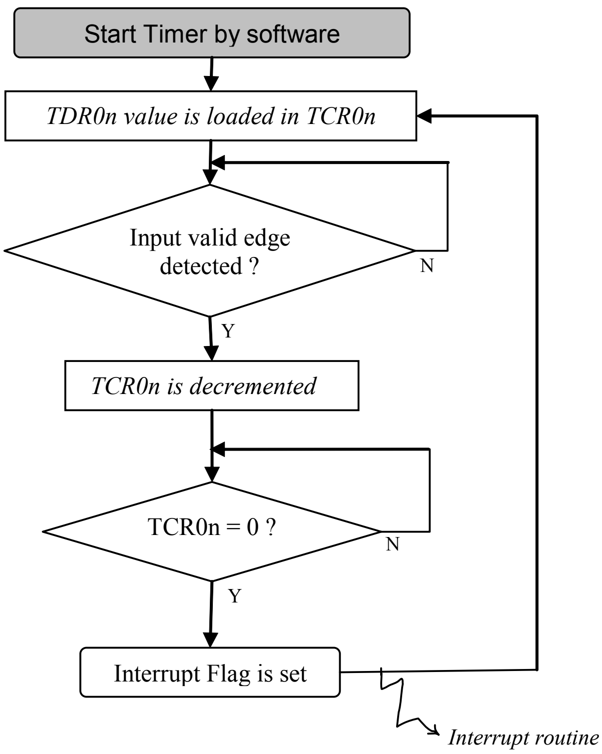


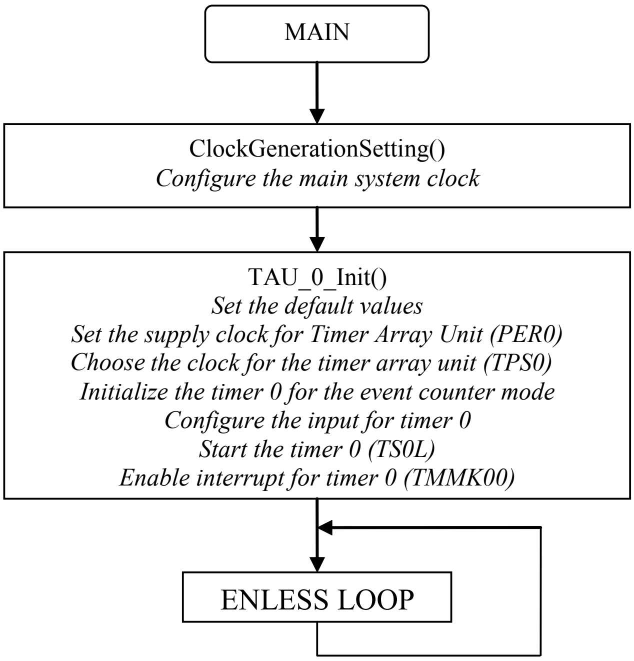
Figure 1-9 Function of the event counter mode

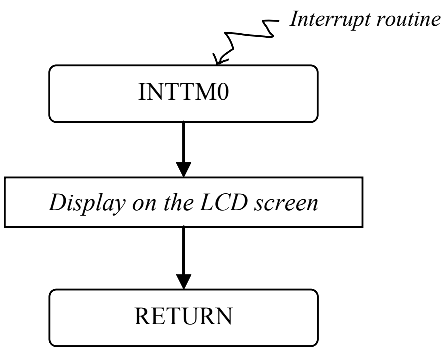
1.3.2 Program specification

```
Channel timer used:        0
```

```
Count clock frequency:     156.2 kHz (at 20 MHz main system clock)
Event counted:             rising edge on the input timer
Number of event counted:   4
Compare value:             TDR00 = 3d = 0x0003
Pins used in program:      P0.0/TI00 to apply input signal with events
Available interrupt:       TM00 interrupt (INTTM00)
```

1.3.3 Software flow chart





1.3.4 Code

```
//-----------------------------------------------------------------------------------------
// Module:   TAU_0_Init
// Function: initialization of clock and timer
//-----------------------------------------------------------------------------------------
void TAU_0_Init(void)
{
```

```
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0070;    // Timer clock selection register for the timer channel
                 // =    0000    0000    0111    0000
                 //            | | | |    ++++ --    CKS00 = 20 MHz
                 //            ++++ ------------    CKS01 = 156.2kHz


    // TIMER 0 INITIALIZATION
    TMR00 = 0x9046;    // Timer 0 mode register
              // =    1001    0000    0100    0110
              //      |  |    ||||    ||      ||| + --    No operation at the start
              //      |  |    ||||    ||      +++ ----    Event count mode
              //      |  |    ||||    ++ -------------    Detection of rising edge on input
              //      |  |    |+++ -------------------    Software start selected
              //      |  |    + ----------------------    Only one timer used in this mode
              //      |  + --------------------------    Count on input valid edge
              //      + -----------------------------    Selected clock = CKS1

    TDR00 = 0x0003;    // Timer 0 data register
                       // Number of event counted = 4 = TDR00 + 1

    TOE0  = 0x0000;    // Timer output enable register for output 0
                       // No use of timer output because timer input is used

    // PORT INITIALIZATION linked with input
    PM0_bit.no0 = 1;
    P0_bit.no0 = 0;

    TS0L_bit.no0 = 1;    // Start Timer 0

    TMMK00 = 0;        // Enable Timer channel 0 interrupt -> routine available
    TMPR000 = 1 ;      // Set priority level of this interrupt
    TMPR100 = 1 ;      // -> Level 3 = low priority level
    TMIF00 = 0;        // Clear interrupt flag
    IE = 1;            // Enable global interrupt

}

//-----------------------------------------------------------------------------------------
// ISR:      isr_INTTM00
// Function: Timer channel 0 interrupt service routine
//-----------------------------------------------------------------------------------------
#pragma vector = INTTM00_vect
__interrupt void isr_INTTM00(void)
{
        LCD_inst(dclear);
        LCD_cursor(0x0);
        LCD_string(&stext02[0]);
}
```

1.4 Divider Function

1.4.1 Program description

This program shows the configuration of the 16-bit timer TM00 in divider function mode. Timer 0 is used this mode because Timer 0 input TI00 and Timer 0 output TO00 are separated: TI00 is on P0.0 and TO00 is on P0.1.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

The Timer Mode Register TMR00 is used to select the chosen clock and to configure the divider function mode. This determines the type of event that will be counted, rising edge, falling edge…

The Timer Data Register TDR00 sets the divider of the input clock .

$$F_{output\_timer} = \frac{F_{input\_timer}}{(Value_{TDR00} + 1) \times 2} \; or \; \frac{F_{input\_timer}}{(Value_{TDR00} + 1)}$$

The timer has to be configured in order to toggle between high level and low level and consequently to create the divided clock wanted (TOE0)

When all the registers are configured, the timer can be started with the register TS0 setting the bit TS0L_bit.no0.

To use the interrupt, it is necessary to enable the interruption for the timer TM00 clearing TMMK00 after having started the timer.

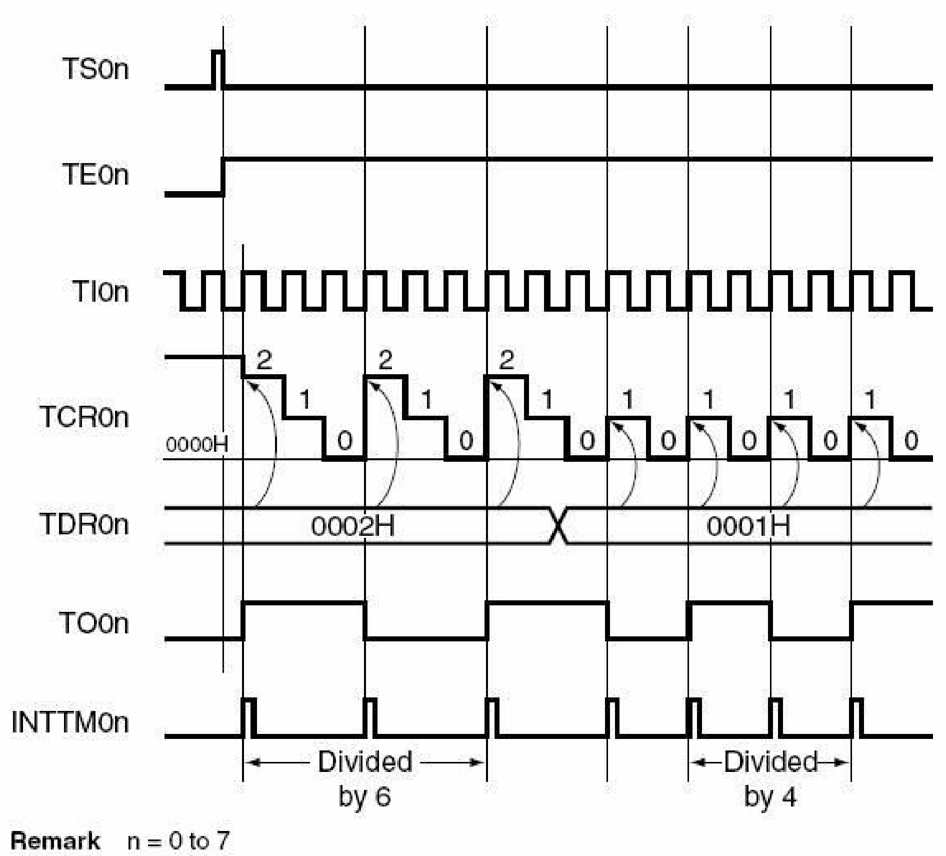


Figure 1-13 Divider function mode diagram

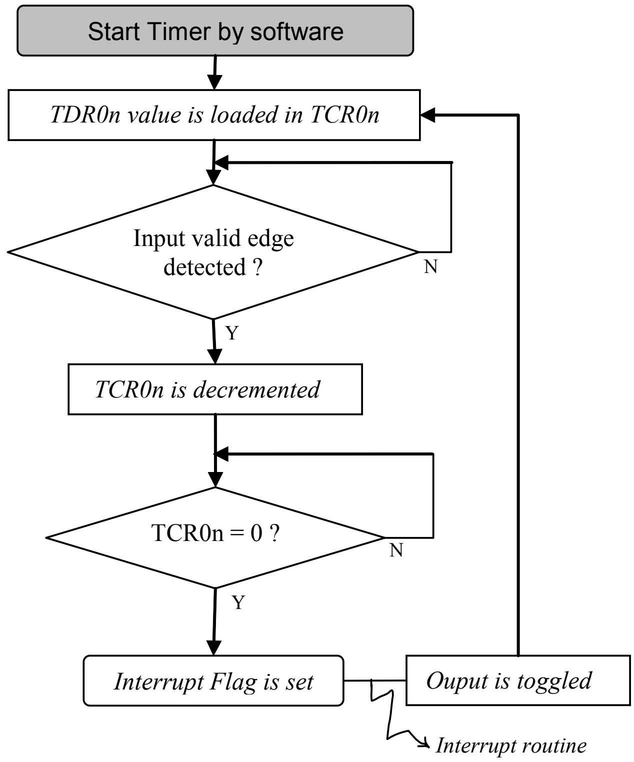
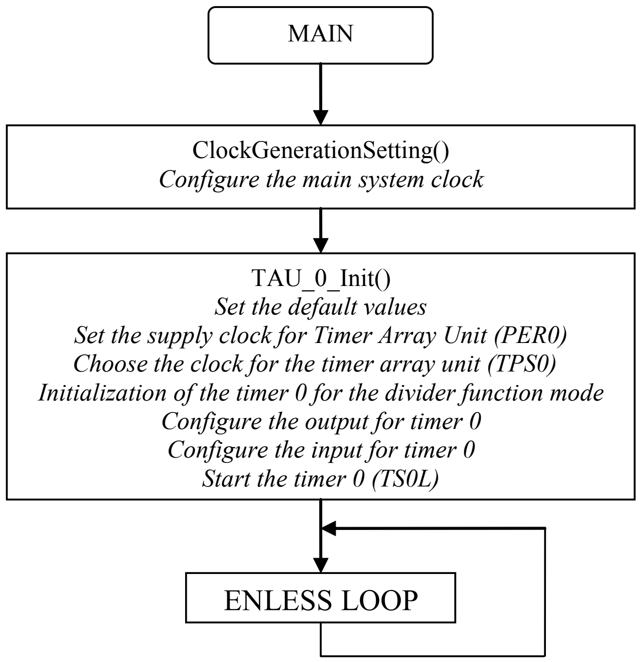


Figure 1-14 Function of the divider function mode

1.4.2 Program specification

```
Channel timer used :     0
Count clock frequency:   156.2 kHz (at 20 MHz main system clock)
Event counted :          rising edge on the input timer
Compare value:           TDR00 = 2
Divider of the input clock obtained: 6
Pins used in program:    P0.0/TI00 to apply input clock signal
                         P0.1/TO00 to ouput the slower clock obtained
Available interrupt:     TM00 interrupt (INTTM00)
```

1.4.3 Software flow chart



1.4.4 Code

```
//-----------------------------------------------------------------------------------------------
// Module:   TAU_0_Init
// Function: initialization of clock and timer
//-----------------------------------------------------------------------------------------------
void TAU_0_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0070;    // Timer clock selection register for the timer channel
                   // =    0000    0000    0111    0000
                   //                      ||||    ++++ --    CKS00 = 20 MHz
                   //                      ++++ ----------    CKS01 = 156.2kHz


    // TIMER 0 INITIALIZATION
    TMR00 = 0x9040;    // Timer 0 mode register
                   // =    1001    0000    0100    0000
                   //      |  |    ||||    ||      |||+ --    No operation at the start
                   //      |  |    ||||    ||      +++ ---    Interval timer mode
                   //      |  |    ||||    ++ -----------    Detection of rising edge on input
                   //      |  |    |+++ -----------------    Software start selected
                   //      |  |    + --------------------    Only one timer used in this mode
                   //      |  + ------------------------    Count on input valid edge
                   //      + ---------------------------    Selected clock = CKS1

    TDR00 = 0x0002;    // Timer 0 data register
                       // Frequency on output = Frequency on input / [2*(TDR00 + 1)]
                       //                     = Frequency on input / 6

    // OUPUT CONFIGURATION
    TOM0 = 0x0000;    // Timer output mode register for output 0
```

```
                    // Set toggle mode for this timer

    TOL0 = 0x0000;    // Timer output level register for output 0

    TO0 = 0x0000;     // Timer output register for output 0

    TOE0 |= 0x0001;   // Timer output enable register for output 0
                      // TO0n operation enabled by count operation: TO0n pin outputs
                      // the low frequency clock created

    // PORT INITIALIZATION linked with input
    PM0_bit.no0 = 1;
    P0_bit.no0 = 0;

    // PORT INITIALIZATION linked with output
    PM0_bit.no1 = 0;
    P0_bit.no1 = 0;

    TS0L_bit.no0 = 1;  // Start Timer 0

    //TMMK00 = 0;       // Enable Timer channel 0 interrupt -> routine available
                        // Use only to do actions in the interrupt: it will not affect the
                        // output clock signal created

}
```

1.5 Input Pulse Interval Measurement

1.5.1 Program description

This program shows the configuration of the 16-bit timer TM00 to operate in the input pulse interval measurement mode.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

The Timer Mode Register TMR00 is used to select the chosen clock and to configure the input pulse interval measurement mode.

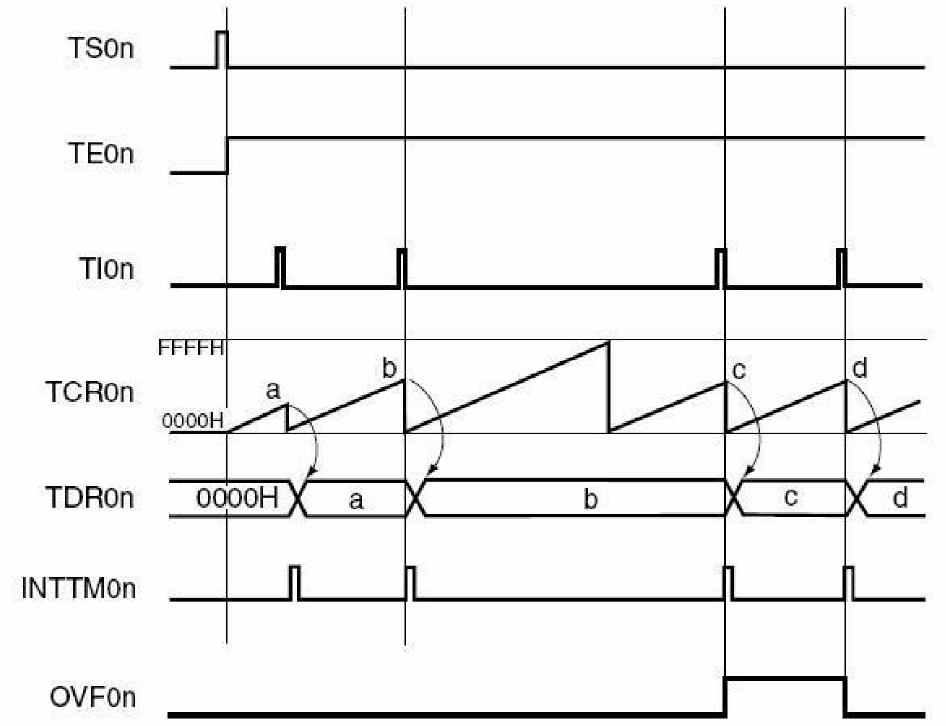
The Timer Data Register TDR00 is used to read the measured value, and consequently to calculate the time interval:

$$Interval_{measured} = Period_{CLK} \times [(10000_H \times TSR0n.OVF) + (Value_{TDR00} + 1)]$$

It is important to be careful about the overflow, which directly influences the value of the register TDR00.

When all the registers are configured, the timer can be started with the register TS0 setting the bit TS0L_bit.no0.

It is necessary to enable the interrupt for the timer TM00 clearing TMMK00 after having started the timer.



Remark n = 0 to 7

Figure 1-16 Input pulse interval measurement mode diagram

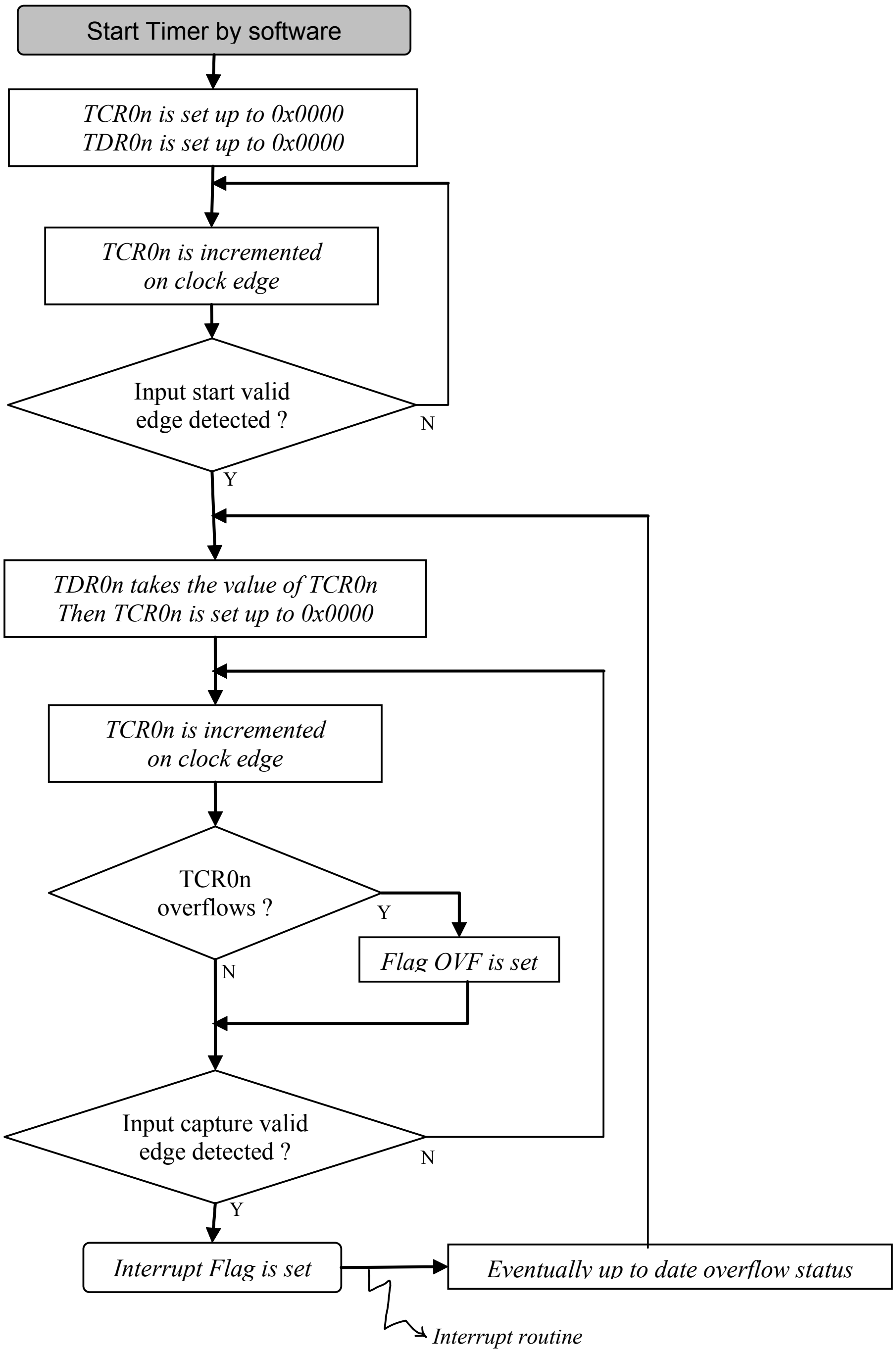
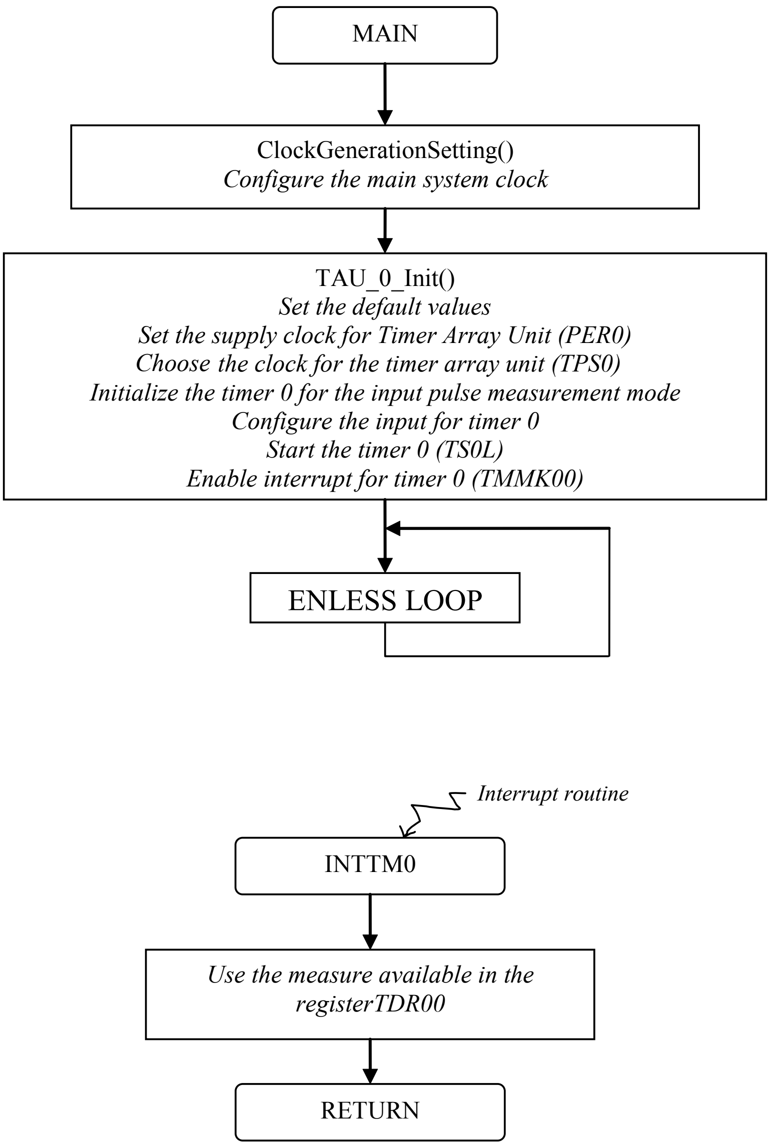


Figure 1-17 Function of the input pulse interval measurement mode

1.5.2 Program specification

```
Channel timer used :     0
Count clock frequency:  1.22 kHz (at 20 MHz main system clock)
Begin/End of counting : rising edge on the input timer
Result available in TDR00
Pins used in program:   P0.0/TI00 to apply the input signal that has to be measured
Available interrupt:     TM00 interrupt (INTTM00)
```

1.5.3 Software flow chart



1.5.4 Code

```
//-----------------------------------------------------------------------------------------------
// Module:   TAU_0_Init
// Function: initialization of clock and timer
//-----------------------------------------------------------------------------------------------
void TAU_0_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x007E;    // Timer clock selection register for the timer channel
                  // =    0000    0000    0111    1110
                  //                      ||||    ++++ --    CKS00 = 1.22 kHz
```

```
              //                      ++++ ----------    CKS01 = 156.2 kHz


    // TIMER 0 INITIALIZATION
    TMR00 = 0x0144;    // Timer 0 mode register
              // =    0000    0001    0100    0100
              //      |  |    ||||    ||      |||+ --    No operation at the start
              //      |  |    ||||    ||      +++ ---    Input measure = capture mode
              //      |  |    ||||    ++ -----------     Detection of rising edge on input
              //      |  |    |+++ -----------------     Start on input valid edge
              //      |  |    + --------------------     Only one timer used in this mode
              //      |  + -------------------------     Count on general clock edge
              //      + ----------------------------     Selected clock = CKS0


    TOE0 = 0x0000;            // Timer output enable register for output 0
                              // No use of timer output because timer input is used

    // PORT INITIALIZATION linked with input
    PM0_bit.no0 = 1;
    P0_bit.no0 = 0;

    TS0L_bit.no0 = 1;         // Start Timer 0

    TMMK00 = 0;              // Enable Timer channel 0 interrupt -> routine available
                             // Permits to use the result of the measure
    TMPR000 = 1;             // Set priority level of this interrupt
    TMPR100 = 1;             // -> Level 3 = low priority level
    TMIF00 = 0;              // Clear interrupt flag
    IE = 1;                  // Enable global interrupt

}

//---------------------------------------------------------------------------------------------
// ISR:      isr_INTTM00
// Function: Timer channel 0 interrupt service routine
//---------------------------------------------------------------------------------------------
#pragma vector = INTTM00_vect
__interrupt void isr_INTTM00(void)
{
        T= TDR00;
}
```

1.6 Measurement of High/Low-Level Width of Input Signal

1.6.1 Program description

This program shows the configuration of the 16-bit timer TM00 in order to operate in the input level width measurement mode.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

The Timer Mode Register TMR00 is used to select the chosen clock and to configure the input level width measurement mode. It is used to select what type of event will be measured: low level or high level width.

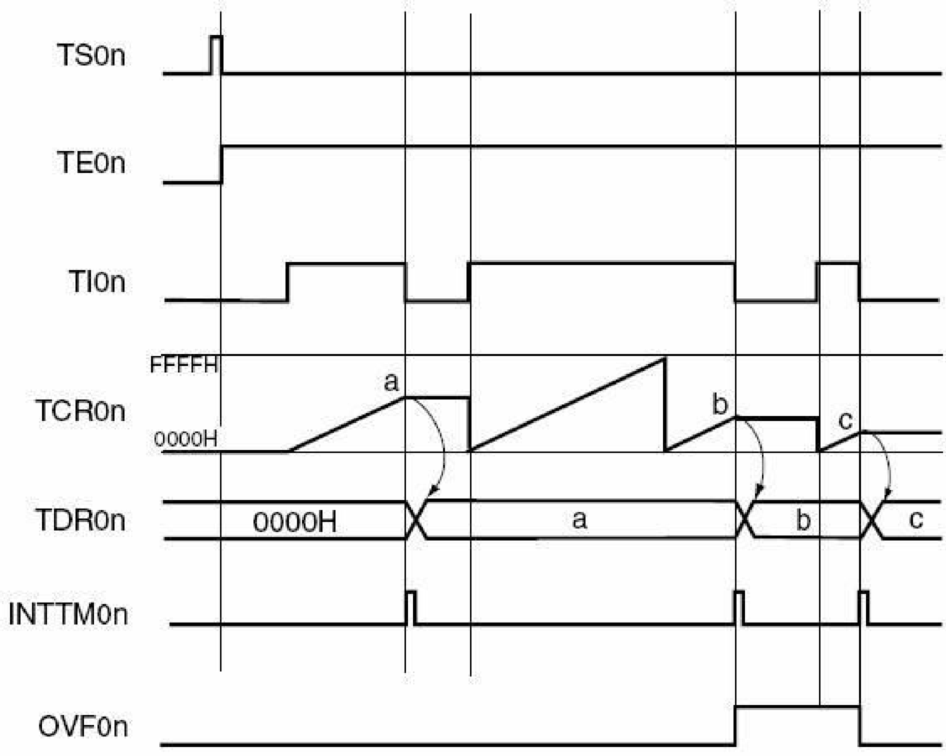
The Timer Data Register TDR00 is read to access the measured value, and consequently to calculate the time width:

$$Width_{measured} = Period_{CLK} \times [(10000_H \times TSR0n.OVF) + (Value_{TDR00} + 1)]$$

It is important to be careful about the overflow, which directly influences the value of the register TDR00.

When all the registers are configured, the timer can be started with the register TS0 setting the bit TS0L_bit.no0.

It is necessary to enable the interrupt for the timer TM00 clearing TMMK00 after having started the timer.



Remark n = 0 to 7

Figure 1-20 Measure of level width mode diagram

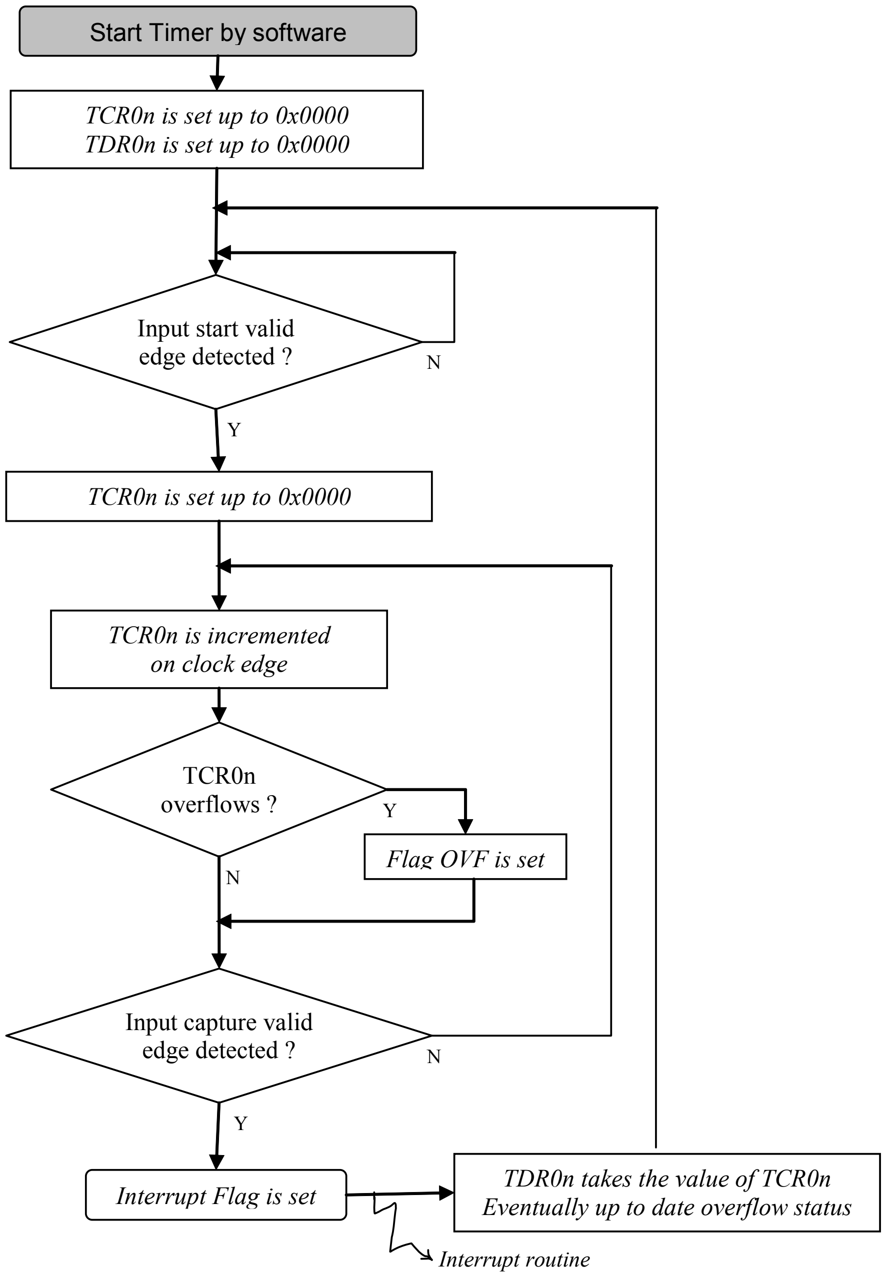
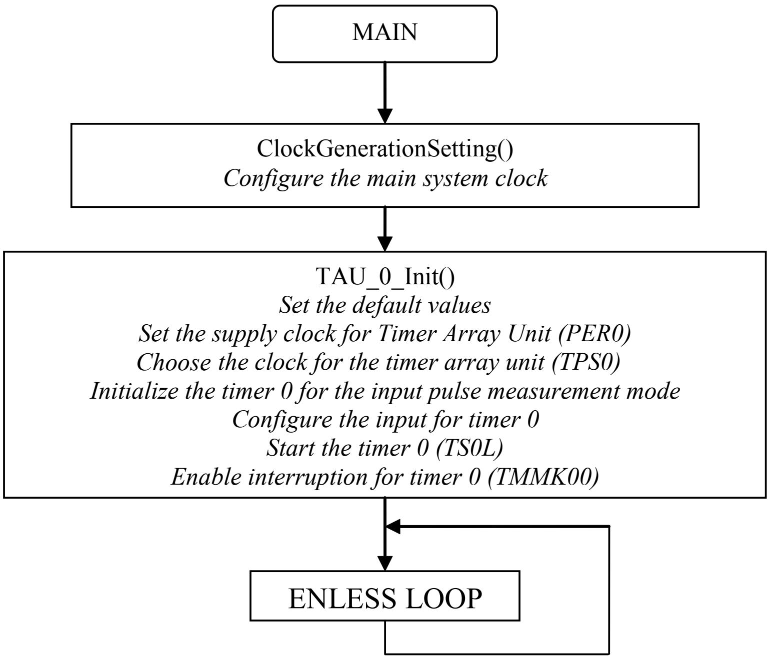


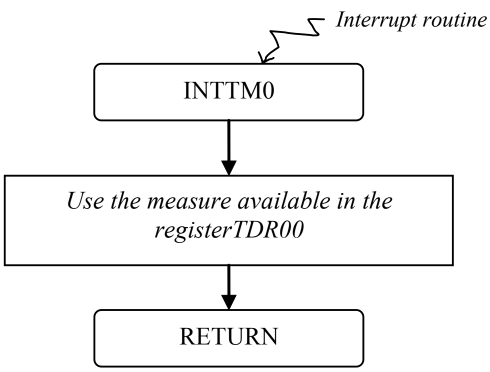
Figure 1-21 Function of the input width measurement mode

1.6.2 Program specification

```
Channel timer used:      0
Count clock frequency: 1.22 kHz (at 20 MHz main system clock)
Begin of counting:     rising edge on the input timer
End of counting:       falling edge on the input timer
Result available in TDR00
Pins used in program:  P0.0/TI00 to apply the input signal
                       that has to be measured
Available interrupt:   TM00 interrupt (INTTM00)
```

1.6.3 Software flow chart





1.6.4 Code

```
//---------------------------------------------------------------------------------------------
// Module:   TAU_0_Init
// Function: initialization of clock and timer
//---------------------------------------------------------------------------------------------
void TAU_0_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    TPS0  = 0x007E;     // Timer clock selection register for the timer channel
                  // =    0000    0000    0111    1110
                  //                      ||||    ++++ --    CKS00 = 1.22 kHz
                  //                      ++++ ----------    CKS01 = 156.2 kHz
```

```
    // TIMER 0 INITIALIZATION
    TMR00 = 0x02CC;    // Timer 0 mode register
                   // =    0000    0010    1100    1100
                   //      |  |    ||||    ||      |||+ --    No operation at the start
                   //      |  |    ||||    ||      +++ ---    Capture & one-count mode
                   //      |  |    ||||    ++ ------------    High-level width measure
                   //      |  |    |+++ ------------------    Start on input valid edge
                   //      |  |    + ---------------------    Only one timer used in this mode
                   //      |  + -------------------------    Count on general clock edge
                   //      + ----------------------------    Selected clock = CKS0

    TOE0 = 0x0000;    // Timer output enable register for output 0
                      // No use of timer output because timer input is used

    // PORT INITIALIZATION linked with input
    PM0_bit.no0 = 1;
    P0_bit.no0 = 0;

    TS0L_bit.no0 = 1;          // Start Timer 0

    TMMK00 = 0;                // Enable Timer channel 0 interrupt -> routine available
                               // Permits to use the result of the measure
    TMPR000 = 1;               // Set priority level of this interrupt
    TMPR100 = 1;               // -> Level 3 = low priority level
    TMIF00 = 0;                // Clear interrupt flag
    IE = 1;                    // Enable global interrupt

}

//-------------------------------------------------------------------------------------------------
// ISR:      isr_INTTM00
// Function: Timer channel 0 interrupt service routine
//-------------------------------------------------------------------------------------------------
#pragma vector = INTTM00_vect
__interrupt void isr_INTTM00(void)
{
        T= TDR00;
}
```

Chapter 2 Operation of the Timer Array Unit Using Multiple Channels

The timer array unit of the 78K0R series of 16 bit microcontrollers can be used in the following modes. For all the modes detailed below, two or more timer channels are required:

- PWM function
- One-shot pulse output function
- Multiple PWM output function

Each mode will be detailed by using an example of time configuration and operation.

For each mode, one channel is used as a master and the other(s) are used as the slave channels. Only channels 0,2,4 and 6 can be used as master. But all channels can be used as slaves. The slave(s) associated to a channel have to follow the associated channel. For example, channel 6 used as master can only have one slave which is channel 7. And channel 4 used as master can have maximum three slave which are channel 5,6 and 7.

2.1 PWM Function

2.1.1 Program description

This program shows the configuration of the 16-bit timers TM04 and TM05 in order to operate in the PWM function mode. For this mode, only two channels are required. So it can be either channels 0&1, channels 2&3, channels 4&5 or channels 6&7.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

The Timer Mode Registers TMR04 and TMR05 are used to select the chosen clock and to configure for the PWM function mode.

The Timer Data Register TDR04 (of the master channel) is used to set the period of the PWM signal.

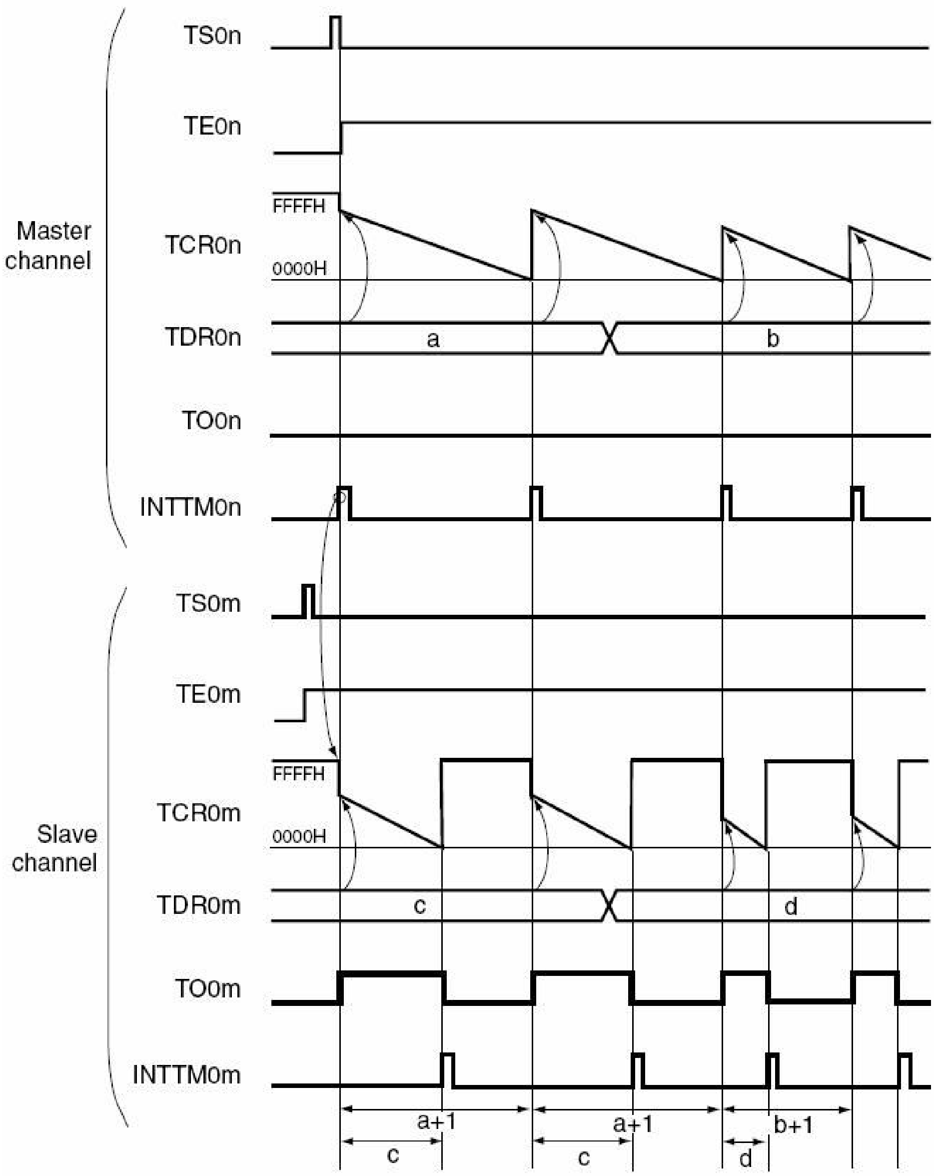
$$Period_{PWMsignal} = Period_{CKS00/CKS01} \times (Value_{TDR04} + 1)$$

The Timer Data Register TDR05 (of the slave channel) is used to set the duty cycle of the PWM signal.

$$DutyCycle_{PWMsignal} = \frac{Value_{TDR05}}{(Value_{TDR04} + 1)} \times 100$$

It is necessary to configure the slave channel in order to output the PWM signal on the associated output pin TO05 with the corresponding bit of the register TEO0.

When all the registers are configured, the timer can be started simultaneously with the register TS0 setting the bits TS0L_bit.no4 and TS0L_bit.no5.



Remark n = 0, 2, 4, 6
m = n + 1

Figure 2-3 PWM mode diagram

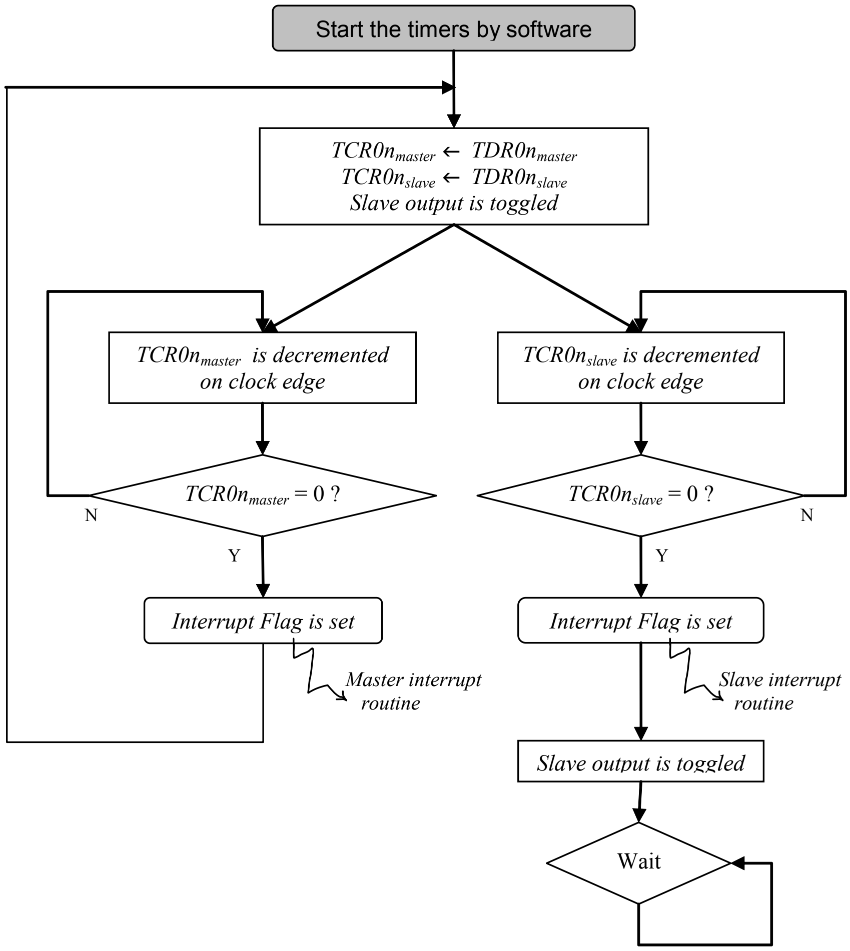


Figure 2-4 Function of the PWM mode

2.1.2 Program specification

```
Channel timer used:       4 as master channel
                          5 as slave channel
Count clock frequency: 156.2 kHz (at 20 MHz main system clock)
PWM signal period:      0.4 s and then 0.2 s after the change in the interrupt
Duty cycle:           30% (then it becomes 60% with the change of PWM period))
Compare value:          TDR04 = 62479d = 0xF40F
                        TDR05 = 18744d = 0x4938
Pins used in program:   P4.6/TO05 to output the PWM signal
Available interrupts:   TM04 interrupt (INTTM04)
                        TM05 interrupt (INTTM05)
```

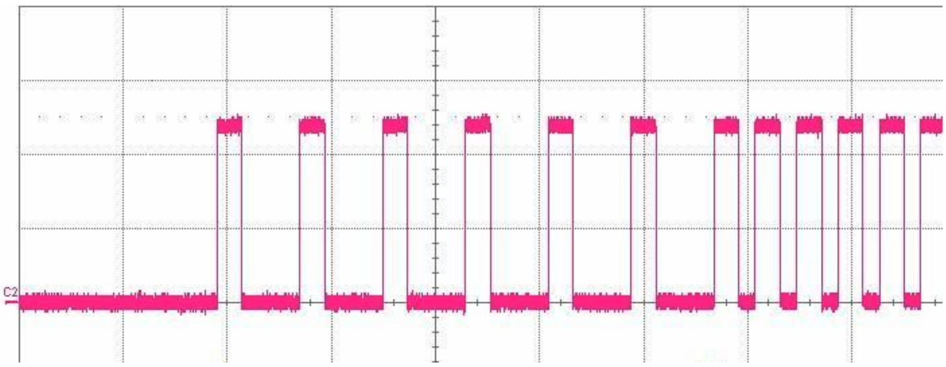
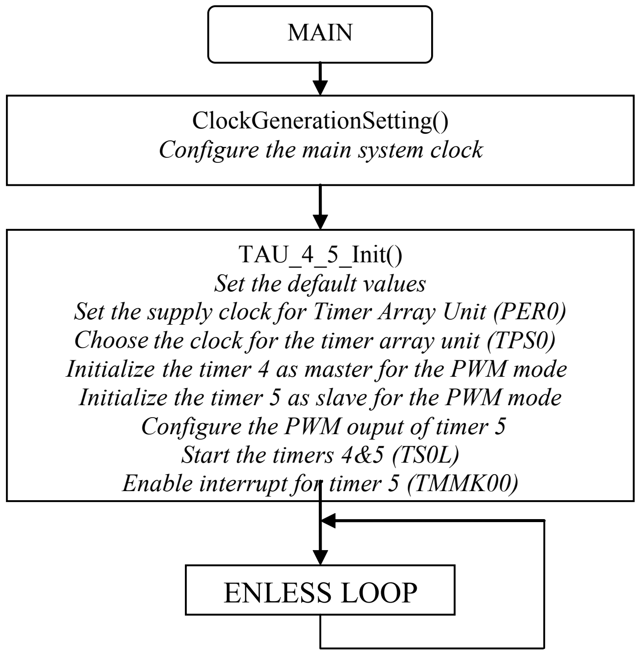
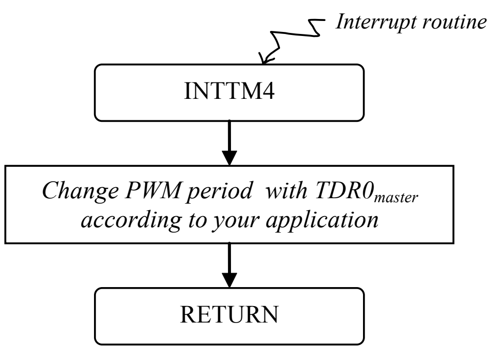


Figure 2-5 Status of the PWM output of timer 5

2.1.3 Software flow chart





2.1.4 Code

```
//-----------------------------------------------------------------------------------------
// Module:   TAU_4_5_Init
// Function: initialization of clock and timer
//-----------------------------------------------------------------------------------------
void TAU_4_5_Init (void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0070;    // Timer clock selection register for the timer channel
```

```
                // =    0000    0000    0111    0000
                //                      ||||    ++++ --    CKS00 = 20 MHz
                //                      ++++ ----------    CKS01 = 156.2kHz


    // TIMER 4 INITIALIZATION = MASTER channel
    TMR04 = 0x8800;     // Timer 4 mode register
                // =    1000    1000    0000    0000
                //      |  |    ||||    ||      |||+ --    No operation at the start
                //      |  |    ||||    ||      +++ ---    Interval timer mode
                //      |  |    ||||    ++ -----------    Input Timer not used- no edge
                //      |  |    |+++ -----------------    Software start selected
                //      |  |    + --------------------    Channel used as Master
                //      |  + -------------------------    Count on general clock edge
                //      + ----------------------------    Selected clock = CKS1

    TDR04 = 0xF40F;    // Timer 4 data register
                       // Set PWM period (16-bit)


    // TIMER 5 INITIALIZATION = SLAVE channel
    TMR05 = 0x8409;     // Timer 5 mode register
                // =    1000    0100    0000    1000
                //      |  |    ||||    ||      |||+ --    Trigger input is valid
                //      |  |    ||||    ||      +++ ---    One count mode
                //      |  |    ||||    ++ -----------    Input Timer not used- no edge
                //      |  |    |+++ -----------------    Starts with master interrupt
                //      |  |    + --------------------    Channel used as Slave
                //      |  + -------------------------    Count on general clock edge
                //      + ----------------------------    Selected clock = CKS1

    TDR05 = 0x4938;    // Timer 5 data register
                       // Set duty cycle

    // OUPUT CONFIGURATION
    TOM0 |= 0x0020;    // Timer output mode register for output 5
                       // Set combination operation mode for this timer

    TOL0 = 0x0000;     // Timer output level register for output 5

    TO0 = 0x0000;      // Timer output register for output 5

    TOE0 |= 0x0020;    // Timer output enable register for output 5

    // PORT INITIALIZATION linked with output
    PM4_bit.no6 = 0;
    P4_bit.no6 = 0;

    TS0 |= 0x0030;      // Start Timer 4 and 5 at the same time

    TMMK04 = 0;         // Enable Timer channel 4 interrupt -> routine available
    TMPR004 = 1;        // Set priority level of this interrupt
    TMPR104 = 1;        // -> Level 3 = low priority level
    TMIF04 = 0;         // Clear interrupt flag
    IE = 1;             // Enable global interrupt

    // TMMK05 = 0;      // Enable Timer channel 5 interrupt -> routine available
}


//-----------------------------------------------------------------------------------
// ISR:      isr_INTTM04
// Function:     Timer channel 4 interrupt service routine
//-----------------------------------------------------------------------------------
#pragma vector = INTTM04_vect
__interrupt void isr_INTTM04(void)
{
    if (i == 5) {
            TDR04 = 0x7A07 ;     // Modify the PWM period
    }
    i++;
}
```

2.2 One-Shot Pulse Output Function

2.2.1 Program description

This program shows the configuration of the 16-bit timers TM00 and TM01 in order to operate in the one-shot pulse output mode. For the pulse output function mode, only two channels are required. The channel selection can be any of the following, channels 0&1, channels 2&3, channels 4&5 or channels 6&7.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

When a signal is applied on the master input the delay count is started. After the delay period the the master channel will generate an interrupt and the pulse width will be measured using the slave channel.

The Timer Mode Register TMR00 and TMR01 is used to select the chosen clock and to configure for the pulse output function mode.

The Timer Data Register TDR00 is used to set delay of the pulse.

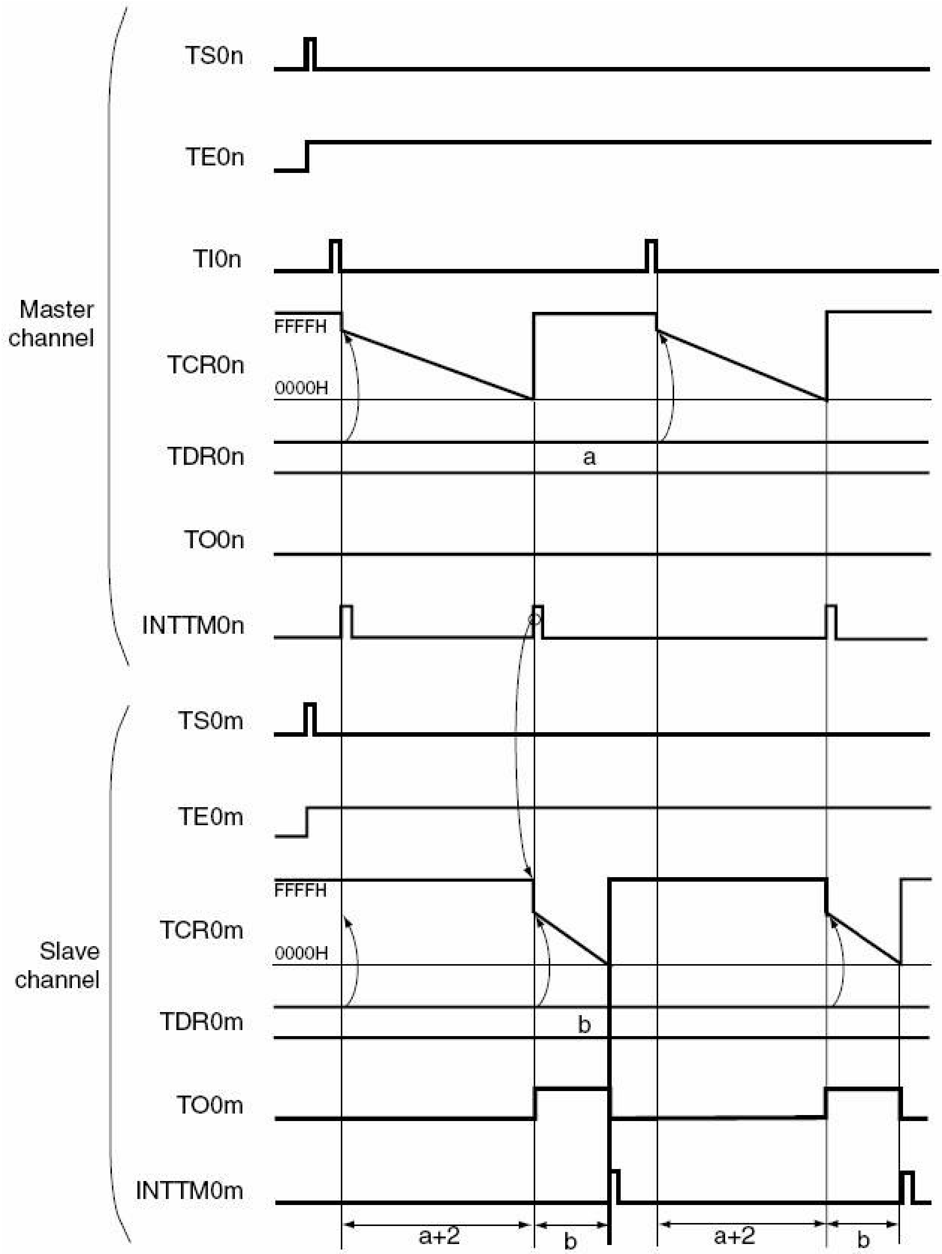
$$Delay\_time = (Value_{TDR00} + 2) \times Period_{CKS00/CKS01}$$

The Timer Data Register TDR01 is used to set the width of the pulse.

$$Pulse\_width = (Value_{TDR01} + 2) \times Period_{CKS00/CKS01}$$

It is necessary to configure the slave channel in order to output the pulse signal on the associated output pin TO01 with the corresponding bit of the register TEO0.

When all the registers are configured, the timers can be started simultaneously with the register TS0 setting the bits TS0L_bit.no0 and TS0L_bit.no1.



Remark n = 0, 2, 4, 6
m = n + 1

Figure 2-8 One shot pulse output mode diagram

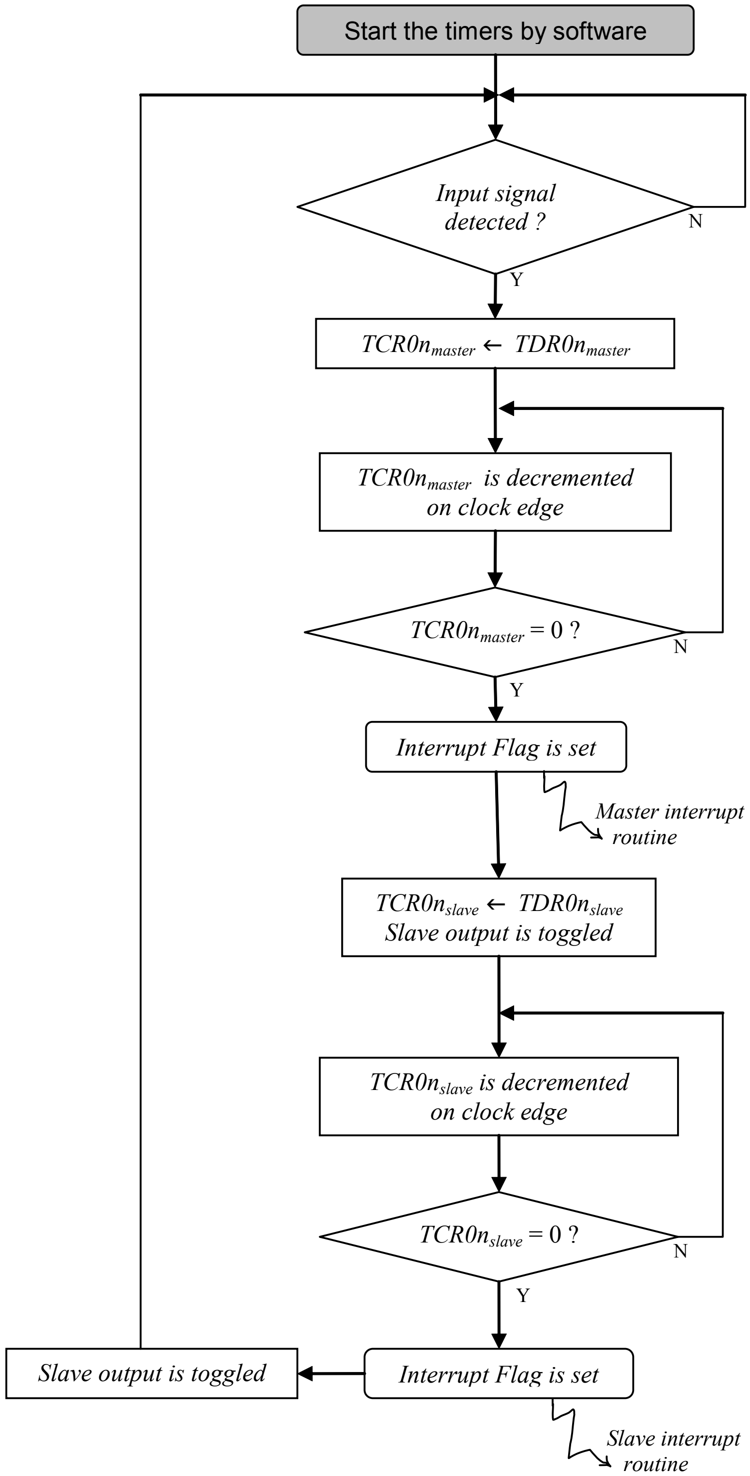


Figure 2-9 Function of the one shot pulse output mode

2.2.2 Program specification

```
Channel timer used:       0 as master channel
                          1 as slave channel
Count clock frequency:    9.76 kHz (at 20 MHz main system clock)
Pulse delay:              0.5 s
Pulse width:              0.1 s
```

```
Compare value:           TDR00 = 4869d = 0x1305
                         TDR01 = 978d = 0x03D2
Pins used in program:    P0.0/TI00 to apply the input signal
                         P1.6/TO01 to output the pulse output signal
Available interruptions: TM00 interruption (INTTM00)
                         TM01 interruption (INTTM01)
```

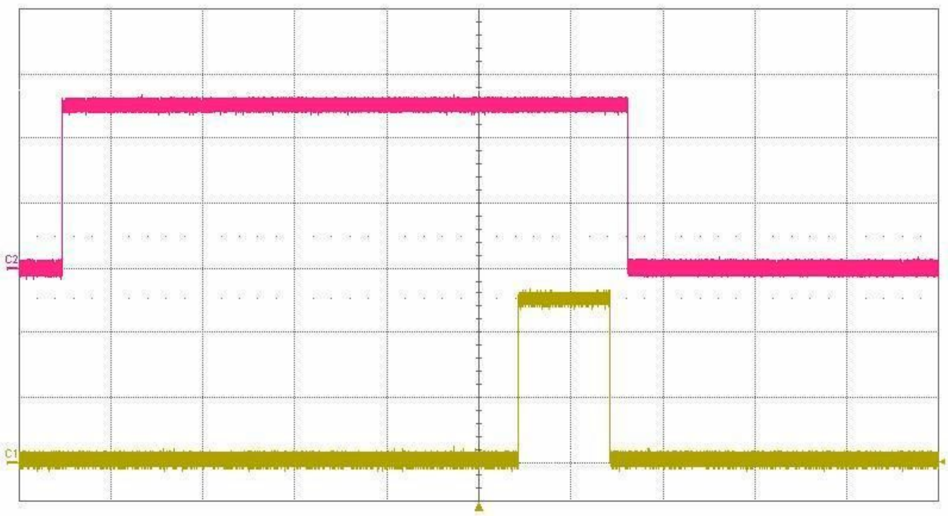
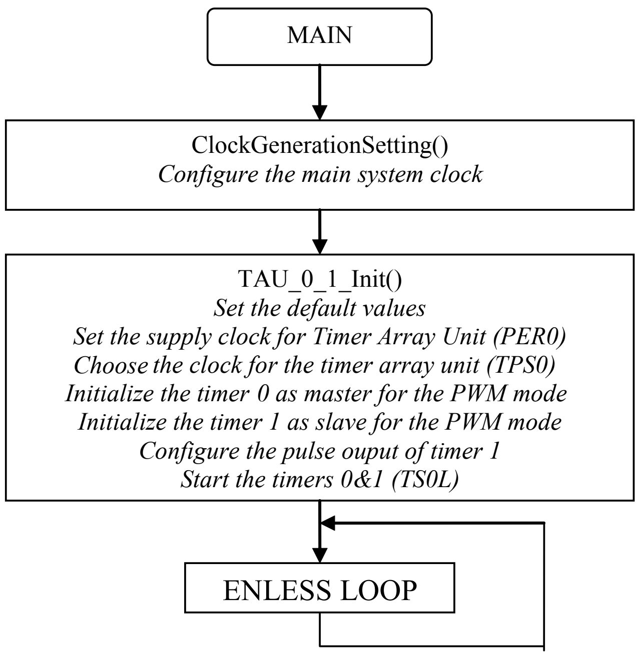


Figure 2-10 Status of the input and output (TOP = input timer 0 and BOTTOM = output timer 1)

2.2.3 Software flow chart



2.2.4 Code

```
//-------------------------------------------------------------------------------------------
// Module:   TAU_0_1_Init
```

```
// Function: initialization of clock and timer
//-------------------------------------------------------------------------------------------
void TAU_0_1_Init (void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x00B0;    // Timer clock selection register for the timer channel
                  // =    0000    0000    1011    0000
                  //      |  |    ||||    ||||    ++++ --    CKS00 = 20 MHz
                  //      |  |    ||||    ++++ ----------    CKS01 = 9.76 kHz


    // TIMER 0 INITIALIZATION = MASTER channel
    TMR00 = 0x8948;     // Timer 0 mode register
                  // =    1000    1001    0100    1000
                  //      |  |    ||||    ||      |||+ --    No operation at the start
                  //      |  |    ||||    ||      +++ ---    One count mode
                  //      |  |    ||||    ++ ------------    Detection of rising edge on input
                  //      |  |    |+++ ------------------    Starts with valid edge on input
                  //      |  |    + ---------------------    Channel used as Master
                  //      |  + -------------------------    Count on general clock edge
                  //      + ----------------------------    Selected clock = CKS1

    TDR00 = 0x1305;    // Timer 0 data register
    // Set pulse delay
    // Pulse delay = 0.5 s =  (TDR00 + 2) / 9760

    // TIMER 1 INITIALIZATION = SLAVE channel
    TMR01 = 0x8409;     // Timer 1 mode register
                  // =    1000    0100    0000    1001
                  //      |  |    ||||    ||      |||+ --    Trigger input is valid
                  //      |  |    ||||    ||      +++ ---    One count mode
                  //      |  |    ||||    ++ ------------    Input Timer not used- no edge
                  //      |  |    |+++ ------------------    Starts with master interrupt
                  //      |  |    + ---------------------    Channel used as Slave
                  //      |  + -------------------------    Count on general clock edge
                  //      + ----------------------------    Selected clock = CKS1

    TDR01 = 0x03D2;    // Timer 1 data register
                       // Set pulse width
                       // Pulse delay = 0.1 s = TDR01 / 9760

    // OUPUT CONFIGURATION
    TOM0 |= 0x0002;    // Timer output mode register for output 1
                       // Set combination operation mode for this timer

    TOL0 = 0x0000;     // Timer output level register for output 1

    TO0 = 0x0000;      // Timer output register for output 1

    TOE0 |= 0x0002;    // Timer output enable register for output 1
                       // TO01 pin outputs the pulse created

    // PORT INITIALIZATION linked with input
    PM0_bit.no0 = 1;
    P0_bit.no0 = 0;

    // PORT INITIALIZATION linked with output
    PM1_bit.no6 = 0;
    P1_bit.no6 = 0;

    TS0 |= 0x0003;        // Start Timer 0 & 1 simultaneously

    //TMMK00 = 0;        // Enable Timer channel 0 interrupt -> routine available
    //TMMK01 = 0;        // Enable Timer channel 1 interrupt -> routine available
                         // Use only to do actions at every interruption

}
```

2.3 Multiple PWM Output Function

2.3.1 Program description

This program shows the configuration of the 16-bit timers TM04, TM05 and TM06 in order to operate in the multiple PWM output mode. For this mode, more than two channels are required, that is to say one master and two slaves minimum. A PWM signal is created on each slave output.

First, the bit for supplying the timer array unit needs to be set (PER0_bit.no0). Then, it is necessary to choose the count clock for the timer array unit (both the timer channels) with the register TPS0. This register allows the selection of two different clocks (CKS01 and CKS00) with different clock frequencies.

The Timer Mode Registers TMR04, TMR05 and TMR06 are used to select the chosen clock and to configure for the multiple PWM function mode.

The Timer Data Register TDR04 (of the master channel) is used to set the period of each PWM signal.

$$Period_{PWMsignal} = Period_{CKS00/CKS01} \times (Value_{TDR04} + 1)$$

The Timer Data Register of each slave channel (TDR05 and TDR06) is used to set the duty cycle of the each PWM output signal.

$$DutyCycle_{PWMsignal} = \frac{Value_{TDR0slave}}{(Value_{TDR04} + 1)} \times 100$$

It is necessary to configure the slave channel in order to output the PWM signals on the associated output pins (TO05 and TO06) with the corresponding bits of the register TEO0.

When all the registers are configured, the timers can be started simultaneously with the register TS0.

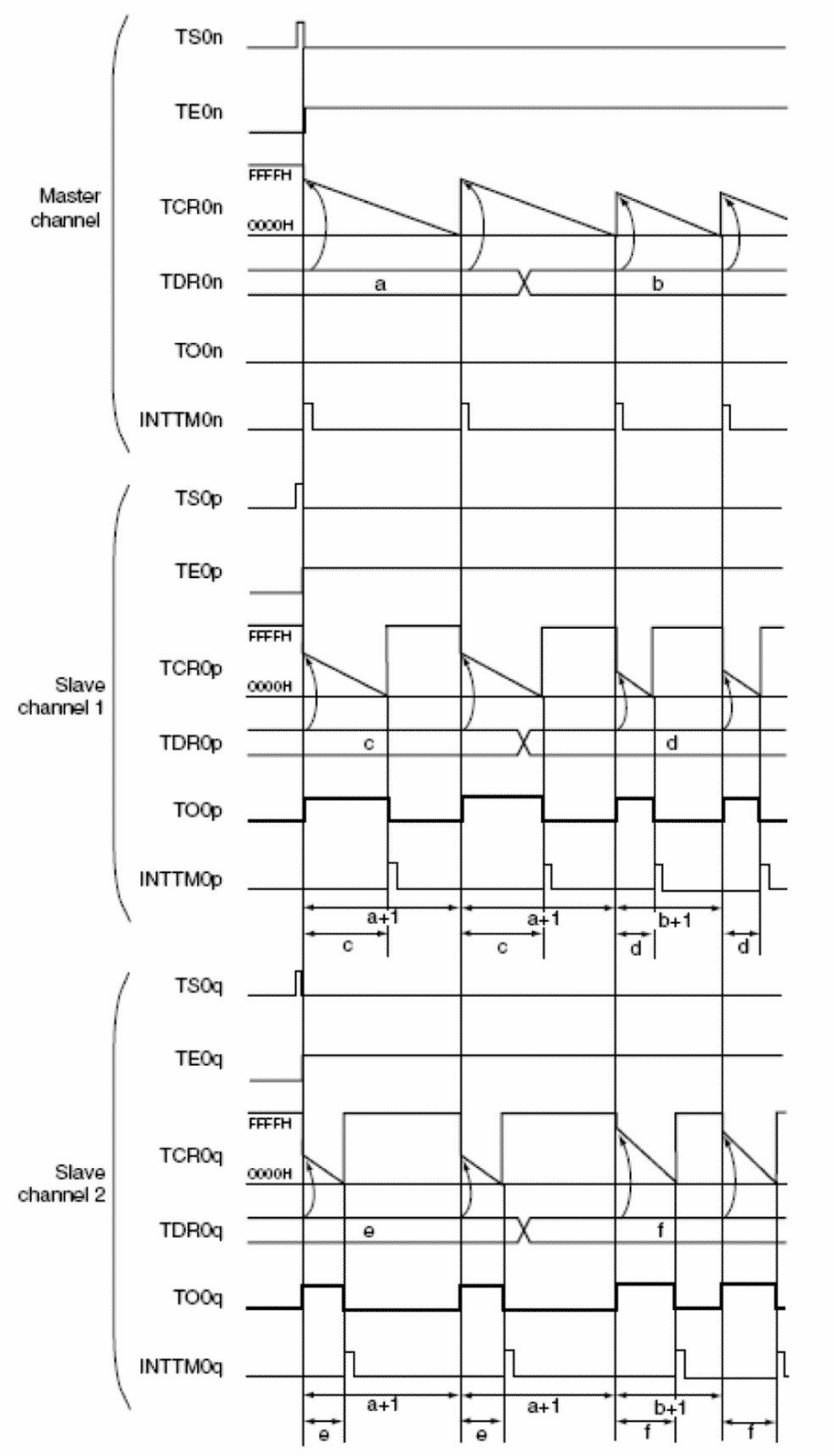


Figure 2-14 Multiple PWM ouput mode diagram

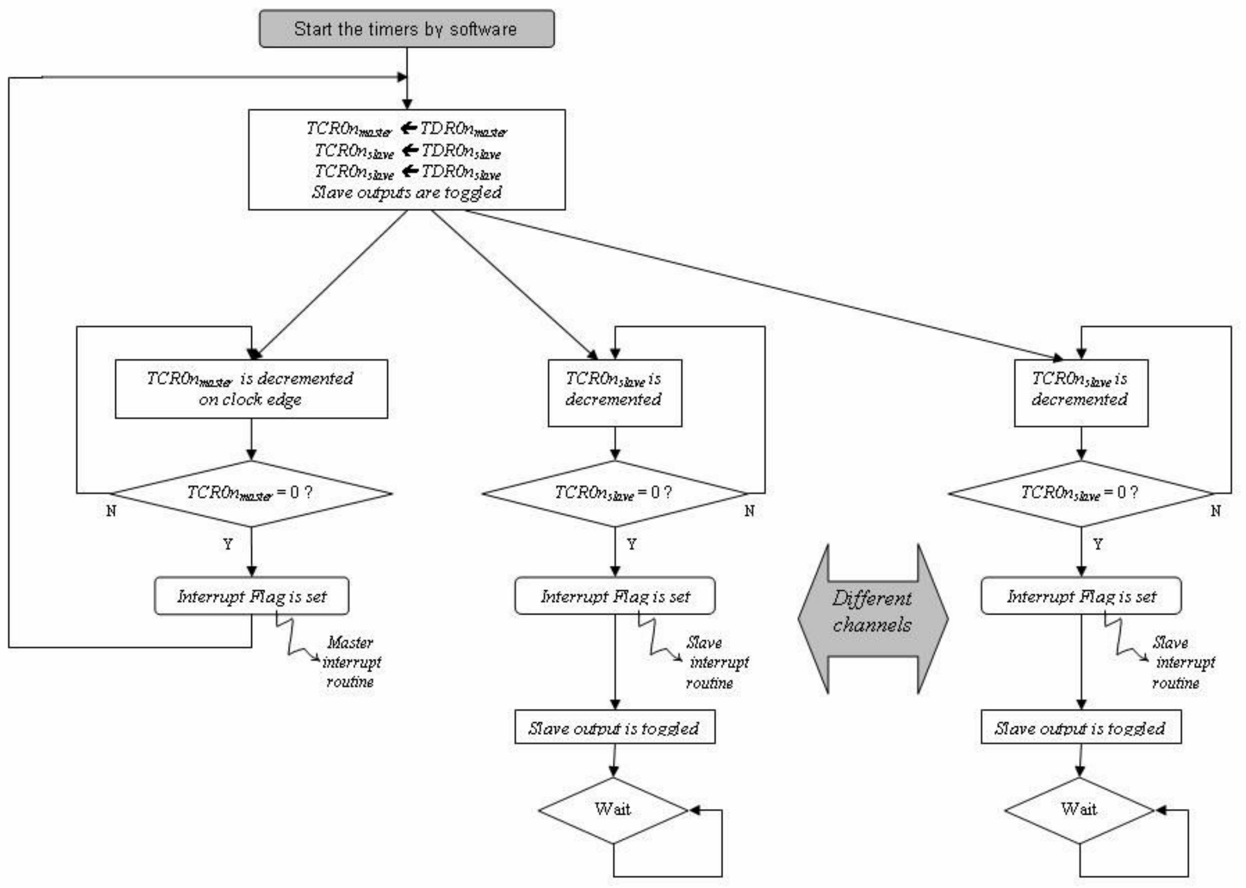


Figure 2-15 Function of the multiple PWM output mode

2.3.2 Program specification

```
Channel timer used:      4 as master channel
                         5 and 6 as slave channels
Count clock frequency: 39.1 kHz (at 20 MHz main system clock)
PWM signal period:       1 s
Duty cycle:             20 % for channel 5 and 80 % after change
                        60 % for channel 6 and 30 % after change
Compare value: TDR04 = 39 099d = 0x98BB
            TDR05 =  7 820d = 0x1E8C -> change to -> TDR05 = 31 279d = 0x7A2F
            TDR06 = 23 460d = 0x5BA4 -> change to -> TDR06 = 11 730d = 0x2DD2
Pins used in program:  P4.6/TO05 to output the PWM signal
                       P13.1/TO06 to output the PWM signal
Available interruptions:  TM04 interruption (INTTM04)
                          TM05 interruption (INTTM05)
                          TM06 interruption (INTTM06)
```

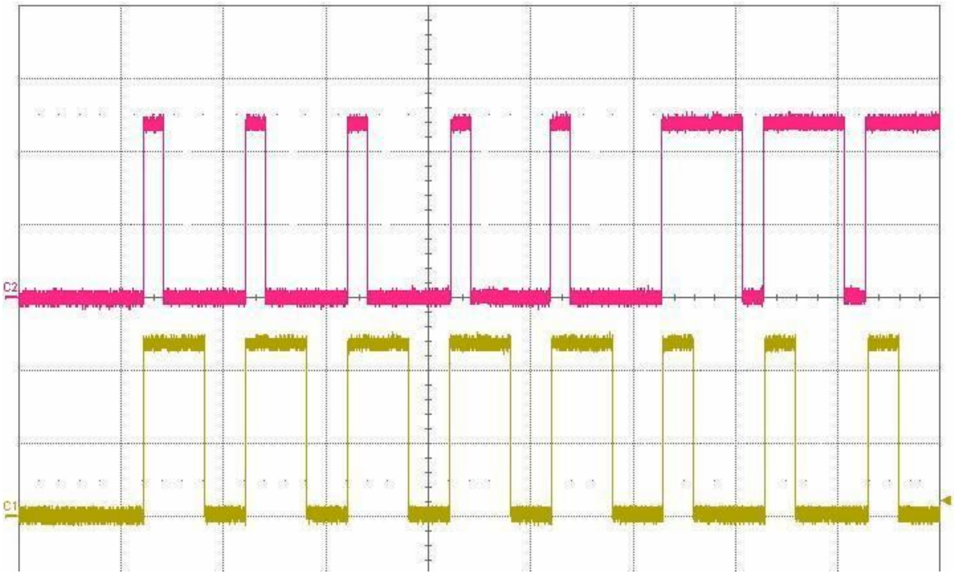
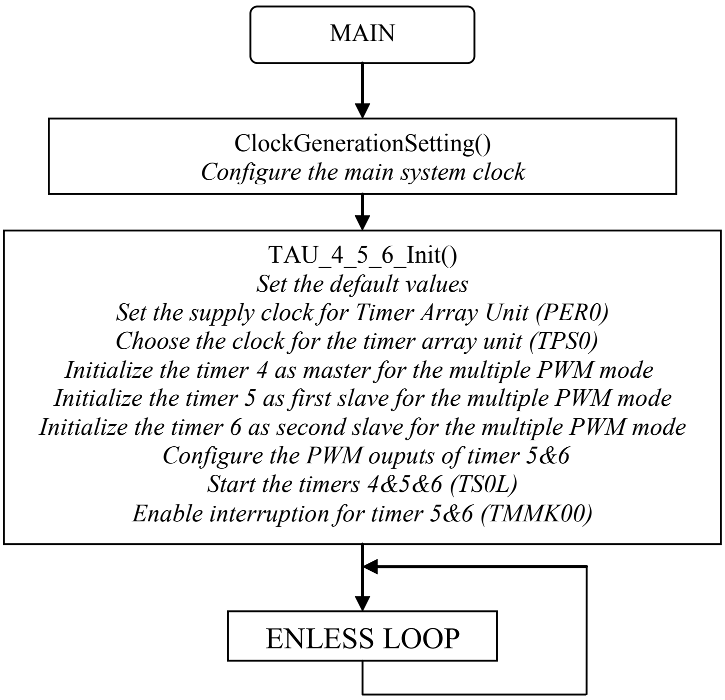
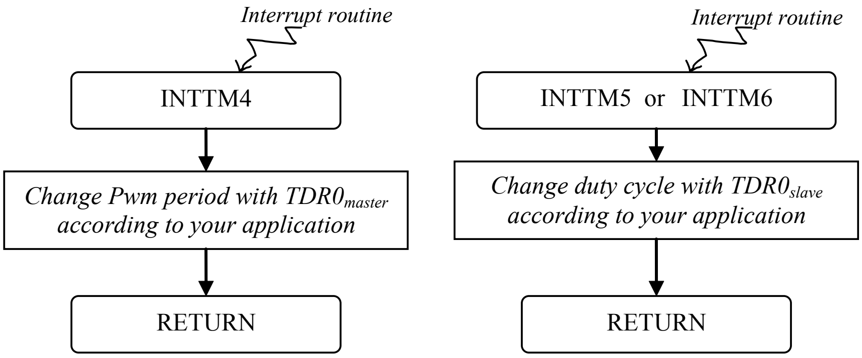


Figure 2-16 Status of the PWM outputs (TOP = output timer 5 and BOTTOM = output timer 6)

2.3.3 Software flow chart





2.3.4 Code

```
//------------------------------------------------------------------------------------------
// Module:   TAU_4_5_6_Init
// Function: initialization of clock and timer
//------------------------------------------------------------------------------------------
void TAU_4_5_6_Init(void)
{
    Default_TAU_Setting ();     // Set all the default values for the TAU

    // CLOCK INITIALIZATION for TIMER ARRAY UNIT
    PER0_bit.no0 = 1;    // Supply input clock to timer array

    TPS0  = 0x0090;    // Timer clock selection register for the timer channel
                    // =    0000    0000    0111    0000
                    //      |  |    ||||    ||||    ++++ --    CKS00 = 20 MHz
                    //      |  |    ||||    ++++ ----------    CKS01 = 39.1 kHz

    // TIMER 4 INITIALIZATION = MASTER channel
    TMR04 = 0x8801;    // Timer 4 mode register
                    // =    1000    1000    0000    0001
                    //      |  |    ||||    ||      |||+ --    No operation at the start
                    //      |  |    ||||    ||      +++ ---    Interval timer mode
                    //      |  |    ||||    ++ ------------    Input Timer not used- no edge
                    //      |  |    |+++ ------------------    Software start selected
                    //      |  |    + ---------------------    Channel used as Master
                    //      |  + --------------------------    Count on general clock edge
                    //      + -----------------------------    Selected clock = CKS1

    TDR04 = 0x98BB;    // Timer 4 data register
                       // Set PWM period (16-bit) = 1 s

        // TIMER 5 INITIALIZATION = SLAVE 1 channel
    TMR05 = 0x8409;     // Timer 5 mode register
                    // =    1000    0100    0000    1001
                    //      |  |    ||||    ||      |||+ --    Trigger input is valid
                    //      |  |    ||||    ||      +++ ---    One count mode
                    //      |  |    ||||    ++ ------------    Input Timer not used- no edge
                    //      |  |    |+++ ------------------    Starts with master interrupt
                    //      |  |    + ---------------------    Channel used as Slave
```

```
                //      |  + --------------------------    Count on general clock edge
                //      + ------------------------------    Selected clock = CKS1

    TDR05 = 0x1E8C;    // Timer 5 data register
                       // Set duty cycle = 20 %

    // TIMER 6 INITIALIZATION = SLAVE 2 channel
    TMR06 = 0x8409;    // Timer 6 mode register
                // =    1000    0100    0000    1001
                //      |  |    ||||    ||      |||+ --    Trigger input is valid
                //      |  |    ||||    ||      +++ ---    One count mode
                //      |  |    ||||    ++ ------------    Input Timer not used- no edge
                //      |  |    |+++ ------------------    Starts with master interrupt
                //      |  |    + ---------------------    Channel used as Slave
                //      |  + --------------------------    Count on general clock edge
                //      + ----------------------------    Selected clock = CKS1

    TDR06 = 0x5BA4;    // Timer 6 data register
                       // Set duty cycle = 60 %

    // OUPUT CONFIGURATION
    TOM0 |= 0x0060;       // Timer output mode register for output 5 & 6
                          // Set combination operation mode for this timer

    TOL0 = 0x0000;        // Timer output level register for output 5 & 6

    TO0 = 0x0000;         // Timer output register for output 5 & 6

    TOE0 |= 0x0060;      // Timer output enable register for output 5 & 6

    // PORT INITIALIZATION linked with output
    PM4_bit.no6 = 0;
    PM13_bit.no1 = 0;
    P4_bit.no6 = 0;
    P13_bit.no1 = 0;

    TS0 |= 0x0070;      // Start Timer 4, 5 & 6 at the same time

    TMMK04 = 0;        // Enable Timer channel 4 interrupt -> routine available
    TMPR004 = 1;       // Set priority level of this interrupt
    TMPR104 = 1;       // -> Level 3 = low priority level
    TMIF04 = 0;        // Clear interrupt flag

    TMMK05 = 0;        // Enable Timer channel 5 interrupt -> routine available
    TMMK06 = 0;        // Enable Timer channel 6 interrupt -> routine available

    PR02L |= 0x03;     // Set priority level of the interrupt of timer 5&6
    PR12L |= 0x03;     // -> Level 3 = low priority level
    TMIF05 = 0;        // Clear interrupt flag
    TMIF06 = 0;        // Clear interrupt flag

    IE = 1;            // Enable global interrupt

}

//---------------------------------------------------------------------------------------------
// ISR:      isr_INTTM04
// Function: Timer channel 4 interrupt service routine
//---------------------------------------------------------------------------------------------
#pragma vector = INTTM04_vect
__interrupt void isr_INTTM04 (void)
{
    i++;
    // possibility of adjusting the period of the PWM output created, using the register TDR04
}

//----------------------------------------------------------------------------------------------
// ISR:      isr_INTTM05
// Function: Timer channel 5 interrupt service routine
//----------------------------------------------------------------------------------------------
#pragma vector = INTTM05_vect
__interrupt void isr_INTTM05 (void)
{
```

```
    // possibility of adjusting the duty cycle of the PWM output of the timer 5
    // this adjustment can follow a closed loop control law for example
    // use of the register TDR05
    if (i == 5) {
            TDR05 = 0x7A2F;    // Timer 5 data register
    }
}


//---------------------------------------------------------------------------------------------
// ISR:      isr_INTTM06
// Function: Timer channel 6 interrupt service routine
//---------------------------------------------------------------------------------------------
#pragma vector = INTTM06_vect
__interrupt void isr_INTTM06 (void)
{
    // possibility of adjusting the duty cycle of the PWM output of the timer 6
    // this adjustment can follow a closed loop control law for example
    // use of the register TDR06
    if (i == 5) {
            TDR06 = 0x2DD2 ;    // Timer 6 data register
    }
}
```