

## KEELOQ<sup>®</sup> Manchester Encoding Receive Routines

Authors: *Lucio Di Jasio*  
Microchip Technology Inc.

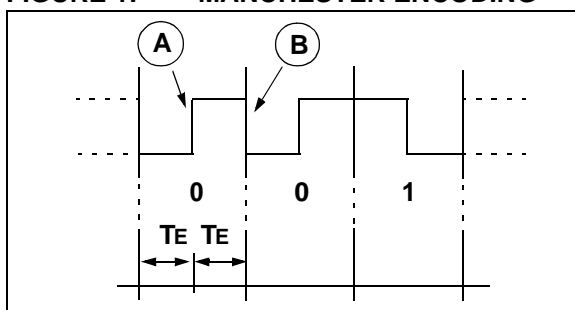
### OVERVIEW

All KEELOQ Encoders use a common code word format. They all pack the bits of information in the same number and position in the transmission sequence. While all the encoders offer a basic Pulse Width Modulation (PWM) method to produce a signal that is suitable for radio transmission, many Advanced KEELOQ Encoders (HCS360 and higher) offer alternative methods. One such alternative method is Manchester encoding. This Technical Brief shows PICmicro<sup>®</sup> microcontrollers (MCUs) assembly routines to receive Manchester encoded transmissions.

### MANCHESTER ENCODING

In Manchester Encoding, as in PWM, clock and data are encoded in a single synchronous bit stream. In this stream, each bit is represented by a transition. If the bit is a '0', the transition is from low to high. If the bit is a '1', the transition is from high to low (see Figure 1).

**FIGURE 1: MANCHESTER ENCODING**



In a typical data stream, there will always be a transition at the center of a bit (A), while at the beginning of a bit there will be a transition depending only on the value of the previous bit (B). The encoding may be alternatively viewed as a phase encoding where each bit is encoded by a positive 90 degree phase transition, or a negative 90 degree phase transition. Manchester code is therefore sometimes known as a bi-phase code.

A Manchester encoded signal contains frequent level transitions which allow the receiver to extract the clock signal easily and reliably. The penalty for introducing frequent transitions, is that the Manchester coded signal consumes more bandwidth than the original signal (sequence of logic ones and zeros or NRZ) but it still compares well with the bandwidth requirements of other encoding systems, such as PWM.

### A MANCHESTER RECEIVER

The role of a Manchester receiver is that of separating the clock information from the transmission stream in order to properly extract the data. This can be done in hardware by means of a Digital Phase Locked Loop (DPLL) circuit or by means of software techniques, as we will show in the following sections.

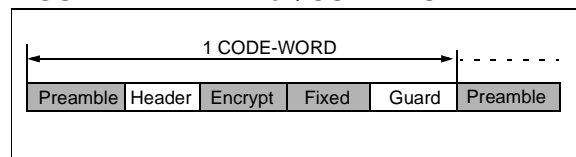
Many different approaches to the problem are possible. This Technical Brief has the sole purpose of offering the reader a starting point for the creation of an interrupt-based Manchester receiver/decoder suitable for use with KEELOQ Encoders.

### KEELOQ MANCHESTER FORMAT

When operating in the Manchester mode, Advanced KEELOQ Encoders build the code-word with a common specific sequence. The elementary period (TE) will be used in the following as the measurement unit. TE will vary from 100  $\mu$ s to 800  $\mu$ s according to the selected baud-rate.

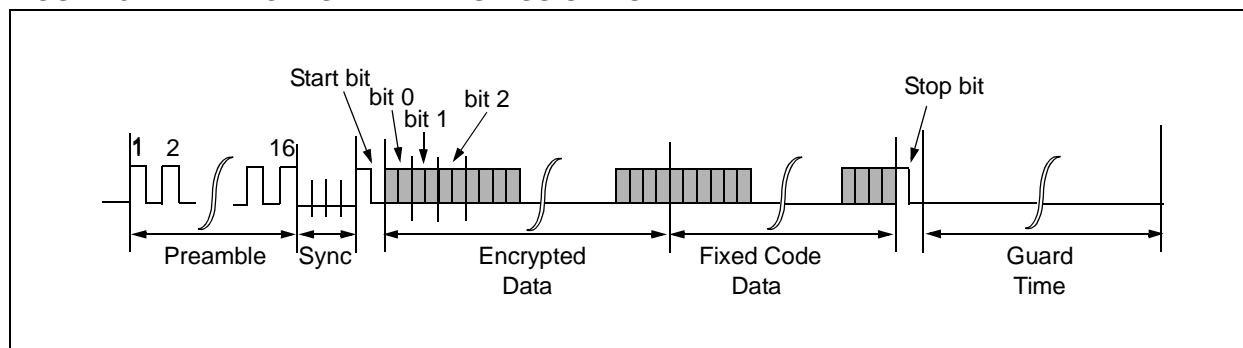
As shown in Figure 2, the code-word is composed of:

**FIGURE 2: KEELOQ CODE-WORD**



- Preamble, consisting of 32 transitions (32xTE)
- Synchronization Header, a pause low of 4xTE duration
- Start bit, consisting of logic 1-bit encoding (2xTE)
- Packet of 65+ bits, the actual data bits (N x 2xTE)
- Stop bit, consisting of a logic 1-bit (2xTE)
- Guard Time, a pause before the whole code word is repeated again (length can vary with encoder models).

**FIGURE 3: MANCHESTER TRANSMISSION FORMAT**



## INTERRUPT RECEIVE TECHNIQUES

Interrupts are available on all Microchip 14-bit and 16-bit core PICmicro microcontrollers. Interrupts can be used to implement an efficient receiver for KEELOQ® Manchester Encoded transmissions.

The example routine presented in Appendix A is based on a simple principle. Timer0 is used to generate a constant period interrupt multiple of the desired baud rate  $T_E$ . The multiplying (oversampling) factor  $RF\_OVERS$  is a user configurable parameter. The value of the parameter (typically in the range of 4 to 8) should be selected as high as possible with the understanding that the higher values will provide more accuracy and flexibility, while causing the interrupt routine to use a larger portion of the CPU processing power.

While the sampling techniques employed in polling and interrupt receivers do not differ much, the whole architecture of a KEELOQ receiver/decoder contains certain advantages and disadvantages. Interrupt-driven receive routines work in the background, while the user main code executes, just as a hardware peripheral would. The only interface between the interrupt service routine and the main program are the buffer that holds the received data (*Buffer*) and the flag that signals the completion of the receiving process (*RF\_FULL*). A significant advantage is that the code is cleaner and is theoretically easier to maintain.

The main disadvantage of interrupt receiving routines is reduced flexibility. The interrupt mechanism is actually stealing processing power from the CPU, and resources from the microcontroller (Timer0). Therefore, the routine length and complexity must be reduced to a minimum.

Sharing resources with the interrupt routine is possible, but requires some attention. For example, the main program is not allowed to write/reset Timer0. However, it is possible to make an effective use of it and the interrupt timing itself to derive multiple software timers for use in the main loop. Enabling other interrupt sources, adding latency to the receiver interrupt servicing, might cause the receiver sampling point to be misplaced and is therefore to be considered very carefully. No timing

can be achieved in the Main Loop by means of simple counters (loops) since the interrupt mechanism will interfere slightly randomizing the loops duration.

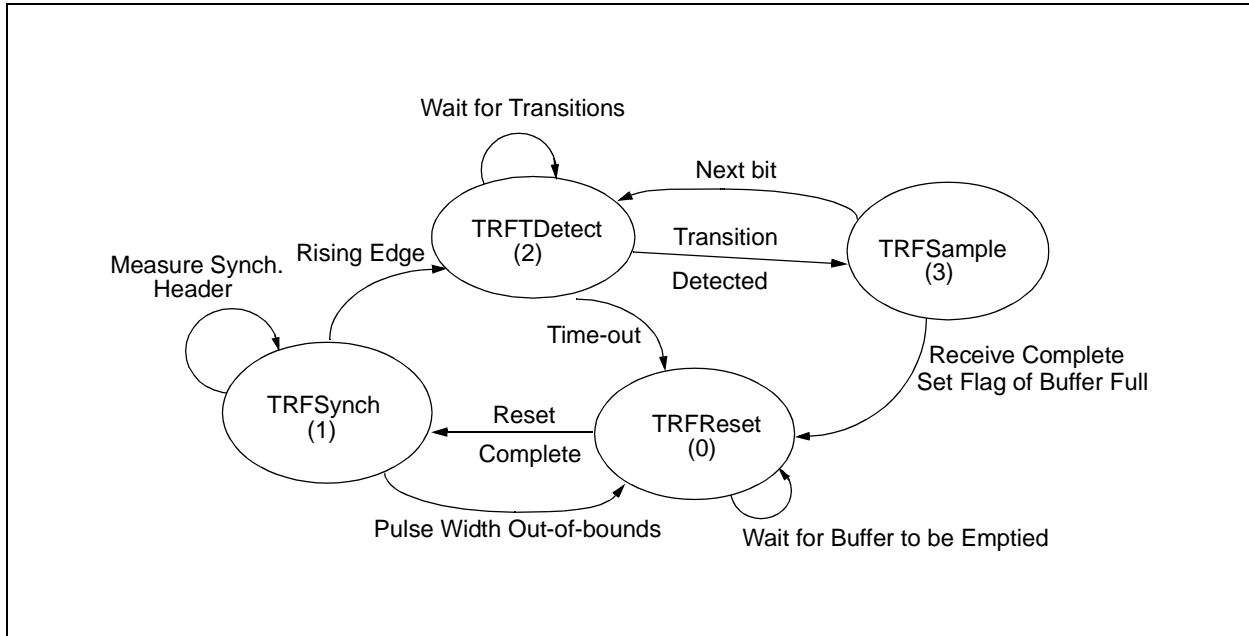
These recommendations are common to all the applications that make use of interrupts.

## CODE DESCRIPTION

The Interrupt Service Routine (ISR) samples the radio receiver digital output and looks for transitions. The ISR operates as a simple state machine capable of four states (see Figure 4).

The sampling rate is determined by the continuous reloading of Timer0 (with *TIMER\_VALUE*). It provides a fixed time base to schedule the execution of one of the state routines according to the value of the variable *RF State*. Each of these state routines is responsible for the advancement of the process to a previous or subsequent state according to a set of rules that determine the actual receiver working mechanism. Each state routine is responsible for loading a skip counter (*RF\_Skip*) with a value of one, to make the execution flow continuous. The skip counter may be loaded with a value larger than one to add a delay and postpone the execution of the next (same) state routine by a multiple of the interrupt timer period, when required.

FIGURE 4: STATE MACHINE TRANSITION GRAPH



The following is a brief description of the workings of the four state routines.

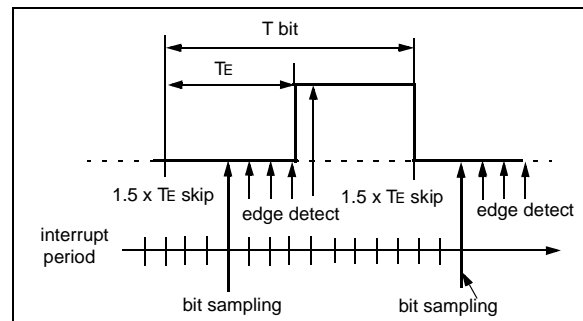
**State 0** (label `TRFReset`) is used at start up and whenever an error or time-out occurs in any of the previous states. Its role is simply that of repositioning the data buffer pointer to the top of the buffer, and clearing the bit counter.

**State 1** (label `RFSYNCH`) is where the receiver waits for the SyncHeader (4 x  $T_E$  pause) after the Preamble and derives from it the value of  $T_E$  (actually  $1.5 \times T_E$ ) expressed in terms of interrupt periods count. This is essentially the way the receiver tries to recalibrate with the transmitter's actual clock frequency, compensating for the ample variations possible due to temperature and battery voltage changes on the transmitters.

**State 2** (label `TRFTDetect`) is where the receiver waits for a transition to happen. This state is critical to the clock extraction process, since the synchronization of the data stream depends on the constant realignment of the receiving process with the input signal transitions. When an edge is detected, a delay of  $1.5 \times T_E$  is set ( $RF\_skip = 1.5 \times T_E$ ) before entering state (3) in order to reach an optimal bit sampling point.

**State 3** (label `TRFSample`), is where the actual data bit is extracted. The action of state '1' combined with the delay of  $1.5 \times T_E$  ensures that data sampling occurs at the ideal place, i.e., at the center of the first half of a bit period (see Figure 5).

FIGURE 5: SAMPLING POINTS



The data bit is then rolled in the buffer and the bit counter advanced. When the required number of bits (`RF_NBITS`) has been received, a flag is set (`RF_Full`) and the receiver state machine loop idles, making the receiver data buffer (`Buffer`) available to the main program for decoding.

In order to re-enable the receiver, the main program resets the same flag (`RF_Full`) releasing control of the receive data buffer.

To complete the Interrupt Service Routine there are a few lines of code dealing with the context saving (at the top) and restoring (at the bottom). This code can be processor dependent, meaning that different models of PICmicro MCUs might actually require modifications to these portions. It is recommended that the reader refers to the specific PICmicro microcontroller datasheet for the suggested implementation of such code.

The `InitRX` routine shows an example of the portion of the PICmicro MCU initialization code required for the receiver start up, where the interrupt services are enabled and the timer options are selected.

## PERFORMANCE

The code presented in the appendix shows an implementation for a clock of 20 MHz and a baud rate corresponding to a TE value of 200  $\mu$ s. The following formula is used to determine the interrupt timer period:

$$\text{TIMER\_VAL} = (\text{XTAL} * \text{BRATE} / (4 * \text{RF\_OVERS}))$$

Changing the parameters XTAL, BRATE, RF\_OVERS to fit different application requirements allows the user to adapt the code to different clock speeds and baud rates within the constraints of the available processing power. It is good practice to keep the percentage of time the processor spends in Interrupt Service Routines (ISR) to a minimum:

$$\text{Pint\%} = \text{Time spent in ISR} / \text{second} * 100$$

or

$$\text{Pint\%} = \text{ISR length} / \text{Interrupt timer period} * 100$$

Such a percentage determines the equivalent average clock speed of execution for the main program (running in foreground) as per the following formula:

$$\text{FEQ} = \text{FOSC} * (1 - \text{Pint\%} / 100)$$

The closer Pint% gets to 100% the slower the main program will appear to execute (FEQ approaching zero), affecting the responsiveness of the whole application.

The absolute limit Pint% = 100, is actually reached when the interrupt timer period (determined by TIMER\_VAL) becomes equal or shorter than the Interrupt Service Routine length (about 60 instruction cycles in this case). In this situation, a new call to the interrupt service routine is made as soon as the processor returns from the previous one, and there is no time left to execute the main program.

When selecting values for a specific application, typically the baud rate will be given, as well as a maximum desired clock speed. The oversampling factor can be used then to optimize the Pint%.

Refer to Table 1 for some possible values:

**TABLE 1: SUGGESTED VALUES**

XTAL	20 MHz	10 MHz	8 MHz	4 MHz	4 MHz
BRATE	200 $\mu$ s	400 $\mu$ s	800 $\mu$ s	800 $\mu$ s	400 $\mu$ s
RF_OVERS	8	8	8	6	4
TIMER_VAL	125	125	200	133	100
Pint%	32%	32%	20%	30%	40%

**Note:** It is important to verify that the value derived from the formula for TIMER\_VAL, being a period for an 8-bit timer (Timer0), must be smaller than ( $2^8 = 256$ ). Compromising with the other parameters allows the user to obtain low use of CPU power (ideally Pint% = 40% or less), while achieving a satisfactory receiver performance.

## MEMORY USAGE

Program Memory Words Used: 133

File Registers Used: 22

## REFERENCES

Secure Learning RKE Systems using KEELQ Encoders	TB001	DS91000
An Introduction to KEELQ Code Hopping	TB003	DS91002
KEELQ CRC Verification Routines	TB043	DS91043
Modular PICmicro Mid-Range MCU Code Hopping Decoder	AN742	DS00742
Modular Mid-Range PICmicro KEELQ Decoder in C	AN744	DS00744

## KEYWORDS

KEELQ, Manchester, Mid-Range, Receiver, Interrupt

### Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PICmicro® Microcontroller is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip PICmicro Microcontroller products.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

## APPENDIX A: SOURCE CODE

```
;*****
; Manchester encoding interrupt driven receiver routines
;
; Uses: interrupt on TMR0
; Accepts: Keeloq code words Manchester encoding
;
; VERSION      1.00      1/16/01      Lucio Di Jasio
;
;*****

#define XTAL      .20      ; clock frequency (MHz)
#define BRATE     .200     ; Te = 200 (us)
#define RF_OVERS  .8       ; oversampling factor (4..8)

TIMER_VAL      equ (XTAL * BRATE) / (4 * RF_OVERS)

        IF (TIMER_VAL > .256) || (TIMER_VAL < .60)
            error "ERROR: Timer period exceeds limits"
        ENDIF

; bit timing limits
#define SHORT_HEAD .3*RF_OVERS ; minimum synch header length accepted
#define LONG_HEAD  .6*RF_OVERS ; maximum synch header length accepted
#define HIGH_TO    .3*RF_OVERS ; bit timeout

#define RF_NBITS.69 ; number of bits to capture

        CBLOCK 0x20 ; omit the address (0x20) if this include module is
                    ; not the first to allocate RAM
            W_TEMP ; context saving W reg.
            STATUS_TEMP ; " STATUS reg.
            PCLATH_TEMP ; " PCLATH reg.
            FSR_TEMP ; " FSR reg.

            RFP ; receive buffer pointer
            RFbitc ; bit counter
            RFSkip ; skip sample counter
            RFsamp ; samples counter
            RFState ; state variable
            RXFlags ; various flags
            Te ; time unit = 1/ baud rate

            Buffer0:9 ; receive data buffer
```

```
; add here extra software timers
    XTMRH           ; example 16 bit software timer
    XTMRH
    XTMRH

    ENDC

; radio input
#define RFIN        PORTB,0

; flags
#define RF_Full     RXFlags,0 ; buffer full, receiver idle
#define LastBit     RXFlags,1 ; used for transition detection

;-----
; Async_ISR
;
; NOTE: place this routine at loc. 0x04
;       context saving can be specific to the PICmicro model used
;       (ref. to PICmicro datasheet for suggested implementation)
;
IntVector
    movwf    W_TEMP           ; context saving, save W first
    swapf    STATUS,W
    bcf      STATUS,RPO       ; assuming use of variables in bank0
    movwf    STATUS_TEMP      ; save status
    movf     PCLATH,W
    movwf    PCLATH_TEMP      ; save PCLATH
    clrf     PCLATH           ; assuming Async_ISR located in page0

;-----
Async_ISR
    movlw    TIMER_VAL        ; non disruptive timer reload
    subwf    TMR0,F

    bcf      INTCON,T0IF       ; interrupt served

    btfsc    RF_Full           ; if buffer still full
    goto     AsyncRFE           ; idle (exit immediate)

    decfsz   RFSkip,F          ; count the skips
    goto     AsyncRFE           ; delay (exit immediate)

    movf     RFState,W         ; launch appropriate state routine
    andlw    03                ; limit to [0..3]
    addwf    PCL,F             ; offset in jump Table

RFTable
    goto     TRFReset          ; 0
    goto     TRFSync           ; 1
    goto     TRFTDetect        ; 2
    goto     TRFSample         ; 3
RFTableEnd

    IF HIGH(RFTable) != HIGH(RFTableEnd)
        error "RFTable crosses page border"
    ENDIF

;-----
; State 1
;     Waiting and measuring a Synch Header
```

```

;
TRFSync
    btfsc    RFIN                ; wait for a rising edge
    goto     TRFRise
    incf     RFSamp,F            ; while input low count the samples
    movf     RFSamp,W
    btfsc    STATUS,Z
    goto     TRFReset           ; check overflows
    incf     RFSkip,F            ; skip = 1
    goto     AsyncRFE            ; remain in state 0

TRFRise
    movlw    SHORT_HEAD         ; check minimum header length
    subwf    RFSamp,W
    btfss    STATUS,C
    goto     TRFReset           ; procede to reset if too short

    movlw    LONG_HEAD          ; check maximum header length
    subwf    RFSamp,W
    btfsc    STATUS,C
    goto     TRFReset           ; procede to reset if too long

TRCalibra
    bcf      STATUS,C            ; RFSamp measure THeader
    rrf      RFSamp,F            ; divide by two
    bcf      STATUS,C
    rrf      RFSamp,W            ; divide by four
    movwf    Te                  ; Te = 1/4 THeader
    movwf    RFSamp
    bcf      STATUS,C
    rrf      RFSamp,W            ; RFSamp = 1/8 THeader
    addwf    Te,F                ; Te = 1/4THeader + 1/8THeader (that is 1.5xTE)
    incf     Te,F                ; round it up

TRinit
    bsf      LastBit             ; init for detection of falling edges
    clrf     RFSamp              ; reset sample counter
    incf     RFState,F           ; move on to transition detection(2)
    incf     RFSkip,F            ; skip=1
    goto     AsyncRFE            ; done

;-----
; State 2
;
; Transition Detection
;
TRFTDetect
    btfsc    LastBit             ; depending on last value of input
    goto     TRFUSET

; last bit = 0
    btfsc    RFIN                ; detect a transition
    goto     TRFTransition
    goto     TRFNOTransition

; last bit = 1
TRFUSET
    btfss    RFIN                ; detect a transition
    goto     TRFTransition

```

```
        goto      TRFNOTransition

TRFTransition      ; transition detected
        incf      RFState,F      ; move on to Sampling (State 3)
        clrf      RFsamp
        movf      Te,W           ; skip = 1.5xTE
        movwf     RFSkip
        goto      AsyncRFE       ; done

TRFNOTransition    ; no transition detected
        incf      RFSkip,F       ; skip =1
        incf      RFsamp,F       ; keep counting time between transitions
        movlw     HIGH_TO        ; check against Timeout value
        subwf     RFsamp,W
        btfss     STATUS,C       ; if timeout fall through Reset
        goto      AsyncRFE       ; done

;-----
; State 0,
;
; Reset receiver
;
TRFReset
        movlw     1
        movwf     RFState        ; move on to TRFSync (1)
        clrf      RFsamp         ; reset sample counter
        movlw     Buffer0         ; reset buffer pointer
        movwf     RFP
        clrf      RFbitc         ; reset bit counter
        incf      RFSkip,F       ; skip =1
        goto      AsyncRFE       ; done

;-----
; State 3,
;
; Data Sampling
;
TRFSample
        incf      RFSkip,F       ; skip = 1
        decf      RFState,F      ; next will be TDetect again (2)

        bcf       LastBit        ; update LastBit for next transition
        btfsc     RFIN
        bsf       LastBit

        movf      FSR,W          ; save FSR
        movwf     FSR_TEMP
        movf      RFP,W          ; load current pointer
        movwf     FSR           ;

        bcf       STATUS,C
        btfsc     RFIN           ; copy data bit in CARRY
        bsf       STATUS,C
        rrf       INDF,F         ; roll in the buffer

        movf      FSR_TEMP,W     ; restore FSR
        movwf     FSR
        incf      RFbitc,F       ; count the bit
```



```

        movlw    07                ; every 8 bit
        andwf    RFbitc,W
        btfsc    STATUS,Z
        incf     RFP,F            ; increment the pointer

        movf     RFbitc,W         ; check all bit in already
        xorlw    RF_NBITS
        btfss    STATUS,Z
        goto     AsyncRFE        ; not yet, done

TRFFull
        movlw    TRFReset         ; received them all
        movwf    RFState         ; next state will be Reset
        bsf      RF_Full         ; set the buffer full flag
        goto     AsyncRFE        ; done

AsyncRFE

;-----
; Example: how to use the interrupt period to derive more software timers
;         showing a 16 bit TIMER that ticks at every interrupt period
; 1 tick (us) = 4 x TIMER_VAL / XTAL (MHz)
;
        incf     XTMRH,F         ; update 16 bit timer
        incfsz   XTMLR,F
        decf     XTMRH,F

;-----
; Context Restore
;
ExitInt
        movf     PCLATH_TEMP,W   ; restore PCLATH (page)
        movwf    PCLATH
        swapf    STATUS_TEMP,W   ; restore status and bank
        movwf    STATUS
        swapf    W_TEMP,F        ; restore W reg.
        swapf    W_TEMP,W
        retfie                   ; exit re-enabling interrupts

;-----
; Init Receiver
;
; Example code for proper interrupt and timer set up
;
InitRX
        clrf     RXFlags
        clrf     INTCON          ; disable interrupts
        clrf     RFState        ; init with Reset state
        clrf     XTMRH          ; init demo software timer
        clrf     XTMLR
        movlw    1
        movwf    RFSkip         ; init skip

        bsf      STATUS,RP0      ; bank 1
        movlw    b'10001111'    ; init timer prescaler 1:1
        movwf    OPTION_REG     ; prescaler assigned to WDT 1:128
        bcf      STATUS,RP1      ; bank 0

        movlw    -TIMER_VAL     ; init timer
        movwf    TMR0

```

```
bsf      INTCON,T0IE    ; enable interrupts on TMR0 overflow
bsf      INTCON,GIE     ; global interrupt enable
return
```

"All rights reserved. Copyright © 2001, Microchip Technology Incorporated, USA. Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights."

### Trademarks

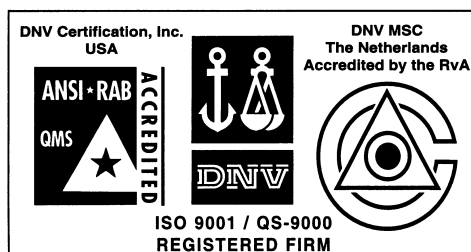
The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELOQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, FlexROM, fuzzyLAB, MPASM, MPLINK, MPLIB, PICDEM, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, Select Mode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.



*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.*



## WORLDWIDE SALES AND SERVICE

### AMERICAS

#### Corporate Office

2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200 Fax: 480-792-7277  
Technical Support: 480-792-7627  
Web Address: <http://www.microchip.com>

#### Rocky Mountain

2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7966 Fax: 480-792-7456

#### Atlanta

500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770-640-0034 Fax: 770-640-0307

#### Austin

Analog Product Sales  
8303 MoPac Expressway North  
Suite A-201  
Austin, TX 78759  
Tel: 512-345-2030 Fax: 512-345-6085

#### Boston

2 Lan Drive, Suite 120  
Westford, MA 01886  
Tel: 978-692-3848 Fax: 978-692-3821

#### Boston

Analog Product Sales  
Unit A-8-1 Millbrook Tarry Condominium  
97 Lowell Road  
Concord, MA 01742  
Tel: 978-371-6400 Fax: 978-371-0050

#### Chicago

333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071 Fax: 630-285-0075

#### Dallas

4570 Westgrove Drive, Suite 160  
Addison, TX 75001  
Tel: 972-818-7423 Fax: 972-818-2924

#### Dayton

Two Prestige Place, Suite 130  
Miamisburg, OH 45342  
Tel: 937-291-1654 Fax: 937-291-9175

#### Detroit

Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250 Fax: 248-538-2260

#### Los Angeles

18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888 Fax: 949-263-1338

#### Mountain View

Analog Product Sales  
1300 Terra Bella Avenue  
Mountain View, CA 94043-1836  
Tel: 650-968-9241 Fax: 650-967-1590

#### New York

150 Motor Parkway, Suite 202  
Hauppauge, NY 11788  
Tel: 631-273-5305 Fax: 631-273-5335

#### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

#### Toronto

6285 Northam Drive, Suite 108  
Mississauga, Ontario L4V 1X5, Canada  
Tel: 905-673-0699 Fax: 905-673-6509

### ASIA/PACIFIC

#### Australia

Microchip Technology Australia Pty Ltd  
Suite 22, 41 Rawson Street  
Epping 2121, NSW  
Australia  
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

#### China - Beijing

Microchip Technology Beijing Office  
Unit 915  
New China Hong Kong Manhattan Bldg.  
No. 6 Chaoyangmen Beidajie  
Beijing, 100027, No. China  
Tel: 86-10-85282100 Fax: 86-10-85282104

#### China - Shanghai

Microchip Technology Shanghai Office  
Room 701, Bldg. B  
Far East International Plaza  
No. 317 Xian Xia Road  
Shanghai, 200051  
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

#### Hong Kong

Microchip Asia Pacific  
RM 2101, Tower 2, Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2401-1200 Fax: 852-2401-3431

#### India

Microchip Technology Inc.  
India Liaison Office  
Divyasree Chambers  
1 Floor, Wing A (A3/A4)  
No. 11, O'Shaughnessey Road  
Bangalore, 560 025, India  
Tel: 91-80-2290061 Fax: 91-80-2290062

#### Japan

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa, 222-0033, Japan  
Tel: 81-45-471-6166 Fax: 81-45-471-6122

### ASIA/PACIFIC (continued)

#### Korea

Microchip Technology Korea  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea  
Tel: 82-2-554-7200 Fax: 82-2-558-5934

#### Singapore

Microchip Technology Singapore Pte Ltd.  
200 Middle Road  
#07-02 Prime Centre  
Singapore, 188980  
Tel: 65-334-8870 Fax: 65-334-8850

#### Taiwan

Microchip Technology Taiwan  
11F-3, No. 207  
Tung Hua North Road  
Taipei, 105, Taiwan  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### Denmark

Microchip Technology Denmark ApS  
Regus Business Centre  
Lautrup hof 1-3  
Ballerup DK-2750 Denmark  
Tel: 45 4420 9895 Fax: 45 4420 9910

#### France

Arizona Microchip Technology SARL  
Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

#### Germany

Arizona Microchip Technology GmbH  
Gustav-Heinemann Ring 125  
D-81739 Munich, Germany  
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

#### Germany

Analog Product Sales  
Lochhamer Strasse 13  
D-82152 Martinsried, Germany  
Tel: 49-89-895650-0 Fax: 49-89-895650-22

#### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-039-65791-1 Fax: 39-039-6899883

#### United Kingdom

Arizona Microchip Technology Ltd.  
505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/30/01

All rights reserved. © 2001 Microchip Technology Incorporated. Printed in the USA. 5/01  Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, except as maybe explicitly expressed herein, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.