OVERVIEW

This application note describes the working of a decoder for the HCS101 fixed-code encoder. The decoder is implemented on Microchip’s smallest 8-pin microcontroller with internal EEPROM, the PIC12CE518.

KEY FEATURES

- Stand alone decoder
- Three function outputs
- Capable of learning a single transmitter
- Automatic baud rate detection
- Internal RC oscillator

TABLE 1: FUNCTIONAL INPUTS AND OUTPUTS

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Pin Number</th>
<th>I/O/P Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM</td>
<td>4</td>
<td>I</td>
<td>Demodulated PWM Signal from RF Receiver</td>
</tr>
<tr>
<td>LEARN</td>
<td>3</td>
<td>I</td>
<td>Input to enter LEARN Mode</td>
</tr>
<tr>
<td>LED</td>
<td>2</td>
<td>O</td>
<td>Output to show the status of the LEARN Process</td>
</tr>
<tr>
<td>S0, S1, S2</td>
<td>5,6,7</td>
<td>O</td>
<td>Function Outputs, correspond to Encoder Input pin</td>
</tr>
<tr>
<td>VDD</td>
<td>1</td>
<td>PWR</td>
<td>5V Power Supply</td>
</tr>
<tr>
<td>VSS</td>
<td>8</td>
<td>GND</td>
<td>Common Ground</td>
</tr>
</tbody>
</table>

FIGURE 1: DECODER PIN-OUT

INTRODUCTION TO THE HCS101

The HCS101 is a fixed-code encoder, designed for remote control systems. It was developed to complement Microchip’s KEELOQ® family of encoders. The HCS101 does not contain code hopping technology and is, therefore, intended for applications that don’t involve a high level of security (i.e., remote indoor lighting, remote sprinkler operation, etc.). The HCS101 was designed to be easily upgradable to a Hopping Code KEELOQ encoder, should the need arise for a more secure encoder in the same application. As a result, the HCS101 is pin compatible with the following Microchip KEELOQ encoders:

- HC200
- HC201
- HC300
- HC301
- HC320
- HC360
- HC361
- HC362
**Code Word Transmission Format**

The key to receiving data from the HCS101 encoder is understanding its code word transmission format (Figure 2). There are four distinct parts to every HCS101 code word transmission:

- **Preamble**
- **Header**
- **Data**
- **Guard Time**

The preamble starts the transmission and consists of repeating low and high phases each of length $T_E$, the elemental time period. The header consists of a low phase which has a length $10^*T_E$. Next, come 66 data bits. The data bits are Pulse Width Modulated (PWM). As seen in Figure 2, a logic one is equivalent to a high of length $T_E$, followed by a low of length $2^*T_E$. A logic zero is equivalent to a high of length $2^*T_E$, followed by a low of length $T_E$. The final part of the code word transmission is the guard time. This is the spacing before another code word is transmitted.

**Code Word Organization**

The code word organization of the HCS101 makes it a candidate for most remote needs. Figure 3 shows the code word organization. In very simple applications (like the one detailed in this application note), only the first two bytes of the code word need be received and operated on. Within these two bytes, the 10-bit serial number provides transmitter recognition and the function bits provide functionality. For greater versatility, the 16-bit counter can be received as well. This counter gives the HCS101 added security (a decoder can make sure all transmissions are consecutive) and more functionality (a button pressed consecutively in a certain amount of time can be made to produce a different output from the decoder, than if it was pressed just once). The whole code word can be utilized for the most complex applications.
HCS101 Baud Rates

The HCS101 can be configured for two baud rates, one with $T_E$ equivalent to 400 $\mu$S and the other, equivalent to 200 $\mu$S. When the faster baud rate is used, alternate code words are blanked out. This allows the user to transmit at twice the amplitude of the 400 $\mu$S signal, still within FCC regulations (this may not apply outside the United States).

HARDWARE IMPLEMENTATION

The decoder is implemented on Microchip's PIC12CE518 microcontroller. The controller has an operating frequency of 4 MHz. An internal RC oscillator supplies this frequency. The PIC12CE518 is ideal for use as a decoder because it contains 16 bytes of onboard EEPROM, which is used to store the 10-bit serial number of the transmitter.

The decoder was implemented in the circuit shown in Figure 4. As seen, the decoder drives three outputs, corresponding to S0, S1, and S2 on the encoder. The LEARN button is used to enter LEARN mode and the LEARN LED indicates the decoder status (for an explanation of LEARN, see section SOFTWARE IMPLEMENTATION: LEARN). The RF module receives transmitter data and feeds it into the decoder [pin 4].

Note 1: When first developing or debugging such a system, the encoder can be directly wired to the decoder, in order to isolate software issues from receiver performance issues. RF components can be substituted in later, when the decoder is working in a satisfactory manner.

Note 2: The RF receiver module, specified in Figure 4, is made by Telecontrolli, part number RR6-434 (www.telecontrolli.com).

FIGURE 4: DECODER CIRCUIT
SOFTWARE IMPLEMENTATION

The software for the decoder has the following program segments:

- MAIN loop routine
- RECEIVE routine
- LEARN routine

**Note:** Please refer frequently to the source code, Appendix A, as it will clarify the following descriptions.

MAIN Loop

The MAIN loop is where the decoder program spends most of its time. On every cycle through the MAIN loop, three functions are always called:

- INITIAL routine
- TIMER routine
- CLOCK routine

The INITIAL routine simply initializes the I/O pins of the microcontroller and sets the TMR0 prescaler. The other two functions relate to timing in the decoder. The PIC12CE518 only has one hardware timer, TMR0. Because several timers are needed and only one hardware timer exists, several software timers are created. These timers are based on the principle of a person checking his or her watch. TMR0 is allowed to run freely without ever being reset. The functions TIMER and CLOCK refer to TMR0 every time the MAIN loop is run though, thereby constantly updating their respective timers, based on the change in TMR0 since they were last called. The TIMER function updates the lower and higher bytes (SX1TMR, SX2TMR) of the timer that determines the length of time the LEDS are turned on. The CLOCK function updates the lower and higher order bytes (TMRLOW, TMRHIGH) of the timer that measures data pulse widths.

The MAIN loop plays an important role in the RECEIVE routine. Only one part of the RECEIVE routine need be run through at any point in time. Therefore, MAIN directs a state machine for the RECEIVE routine, based on the program state, STATECNTR. As the program advances through the RECEIVE subroutines, STATECNTR is altered.

RECEIVE Routine

The RECEIVE routine gathers the first 32 bits of incoming encoder transmissions. It starts by essentially waiting for the data bus to go high. Once this occurs, it waits for a valid header. As mentioned before, the header is ten times the pulse element length, TE. Depending on the encoder’s baud rate, TE is either 200 µS or 400 µS. Assuming uncalibrated encoders, TE could vary from 150 µS to 500 µS. This gives the header a length, ranging from 1.5 mS to 5 mS. Therefore, the RECEIVE routine’s first task is to look for a low period which has a length in this range. Once the header is detected, the program advances the RECEIVE routine to begin deciphering the ensuing code word.

Rather than detect what baud rate is being used and then measure pulses accordingly, a simpler approach is used. Because the data bits in the code word are pulse width modulated, a data bit equivalent to a one has a 1:2 high to low ratio. Inversely, a data bit equivalent to a zero has a 2:1 high to low ratio (refer to Figure 3). Therefore, the RECEIVE routine simply measures the length of the high phase and compares it to the low phase, in order to determine if the data bit is logic 1 or 0.

After receiving the first 32-bits of the 64-bit code word, the RECEIVE routine waits for the guard time. This is done so that the routine will not begin detecting another code word before the completion of the immediate one. The serial number within the received data is then validated against the serial number stored in EEPROM. Should the serial number be valid, the function bits are implemented. This results in the corresponding LED being turned on.

LEARN Routine

LEARN is the method in which the decoder gets associated with a specific transmitter. During the LEARN routine, a decoder waits for a transmission from an encoder and then memorizes the serial number in the transmission. Once this process is completed, the decoder will only perform commands that it receives from that specific encoder.

LEARN mode is initialized by pushing the LEARN button. At this point, the LEARN routine turns the LEARN LED on. The decoder then waits for the reception of a transmission, or until LEARN mode times out (after 8 seconds). If the decoder receives a transmission while in LEARN mode, the serial number from the transmitter is stored in EEPROM and LEARN mode is exited. Refer to section PROGRAM DEVELOPMENT: Helpful Files, for information on the EEPROM read and write functions.

PROGRAM DEVELOPMENT

Experienced programmers, familiar with Microchip products, might skip this section. However, a programmer just introduced to the Microchip product line may find this section saves them time and headaches, while developing software for a decoder.
Helpful Files

Microchip provides an abundance of files to aid in timely code development. Template files for all Microchip microcontrollers are available in MPLAB® Simulator and at Microchip’s website, (http://www.microchip.com). These files make it necessary for a software developer to enter only the body of the code. All microcontroller specific calibration and configuration is at the head of each template. Each template also has an #include statement for including the file containing the processor specific variable definitions. The template file for the PIC12CE518 is e518temp.asm. The variable definitions are in a file named p12ce518.inc.

Source code for the functions that read and write data to the internal EEPROM is available on Microchip's website as well. The files named fl51xinc.asm and flash51x.asm contain the code for these functions. These functions are made available in the decoder program by either including fl51xinc.asm, or by linking flash51x.asm.

Indirect Referencing

Understanding indirect referencing is essential to writing more efficient software. Indirect referencing is used extensively in the software for this decoder. Two special function registers in all Microchip microcontrollers were created in hardware primarily for this purpose. These registers are the FSR and the INDF registers. The FSR register is an indirect address pointer. In other words, the address of the register whose contents is desired for operation on, is moved into the FSR register. The INDF register essentially refers to the contents of the register pointed to by FSR. Indirect referencing is very useful in the LEARN and VALIDATE portions of the decoder software, because of the ease with which it allows consecutive registers to be operated on.

Simulating a Code Word

Within MPLAB’s simulation environment (MPLAB SIM), a stimulus file (.sti) can be created that exactly models a code word being sent from the encoder. This modeled code word can be used to test the robustness of a decoder's RECEIVE routine. Although a stimulus file is just a simple text file, it is recommended that the stimulus file be created in a spreadsheet. This way, files that model both HCS101 baud rates can be created with minimal effort. See Figure 5 for an example of a pin stimulus file.

Note: In the case of the decoder, 1 cycle = 1 μS
(1 cycle = (1/4 MHz)/4).

FIGURE 5: HCS101 PREAMBLE WITH 400 μS ELEMENT LENGTH

<table>
<thead>
<tr>
<th>CYCLE</th>
<th>GP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>1</td>
</tr>
<tr>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>1200</td>
<td>1</td>
</tr>
<tr>
<td>1600</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>2400</td>
<td>0</td>
</tr>
<tr>
<td>2800</td>
<td>1</td>
</tr>
<tr>
<td>3200</td>
<td>0</td>
</tr>
<tr>
<td>3600</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: GP3 is the data input pin for the decoder.

CONCLUSION

As seen in this application note, implementing a decoder on the PIC12CE518 for Microchip's HCS101 encoder can be done in a very timely manner. The resulting decoder and transmitter can be used in a wide variety of remote applications and is cost efficient. For remote applications that do not involve a high level of security, Microchip's HCS101 fixed-code encoder is an ideal choice.

MEMORY USAGE

In the PIC12CE518, the following memory was used:

- Data Memory: 14 bytes
- Program Memory: 334 bytes
- EEPROM: 2 bytes

REFERENCES

AN659, Simple Code Hopping Decoder (DS00659)
AN665, Using KeeLoq to Generate Hopping Passwords (DS00665)
PIC12C5XX Data Sheet (DS40139)
HCS101 Data Sheet (DS41115)
APPENDIX A: SOURCE CODE

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.

;*********************************************************************
; Filename:          decode02.asm
; Date:              10/6/00
; File Version:      Rev --
; Assembled using:   MPASM v2.50.02
;
; Author:            Reston A. Condit
; Company:           Microchip Technologies Inc.
;
;*********************************************************************

;*******************************************************************************
;
;    Files required:             
;         p12ce518.inc          ; standard header file
;         fl51xinc.asm          ; EEPROM function file (available on
;                                   Microchip’s website)
;*******************************************************************************
;
; Notes:
;
;
;*******************************************************************************

list p=12ce518,r=dec ; list directive to define processor
#include <p12ce518.inc> ; processor specific variable
                      ; definitions
__CONFIG   _CP_OFF & _WDT_ON & _MCLRE_OFF & _IntRC_OSC

;*******************************************************************************

cblock 0x07
DATA0 ; 1st byte of received data
DATA1 ; 2nd byte of received data
DATA2 ; 3rd byte of received data
DATA3 ; 4th byte of received data
ORIGIN ; a reference used to increment TMRLOW
SX1TMR ; LED timer (low order)
SX2TMR ; LED timer (high order)
TMRLOW ; pulse width timer (low order)
TMRHIGH ; pulse width timer (high order)

;*******************************************************************************
HIGHDWT ; high pulse width
LOWDWT ; low pulse width
STATECNTR ; program state counter
BITCNTR ; data stream bit counter
FLAGS ; flags
endc

COUNTR equ BITCNTR ; misc. counter

;****************************************************************************
** DEFINE STATEMENTS *********************************************************
****************************************************************************

; PIC12CE518 setup parameters
#define GP_INITIAL B'011000' ; inputs: GP3, GP4
 ; outputs: GP0, GP1, GP2, GP5
#define PRESCL B'10000001' ; 1 TMR0 per 4 instruction cycles
 ; Cycle Frequency = 4 MHz/4 = 1 MHz
 ; TMR0 increment = 1us * 4 = 4us

; input and output definitions
#define S2 GPIO,0
#define S1 GPIO,1
#define S0 GPIO,2
#define STREAM GPIO,3 ; DATA stream
#define LRN GPIO,4 ; learn button
#define LED GPIO,5 ; learn LED

; Labels for the status counter
#define BEGN 0x00
#define BEGN1 0x01
#define HEADR 0x02
#define HEADR1 0x03
#define HIGHP 0x04
#define LOWP 0x05
#define RECRD 0x06
#define WAIT 0x07
#define VALID 0x08
#define IMPLMNT 0x09

; FLAGS is parsed as follows
#define LERN FLAGS, 0 ; this flag is set when in learn mode
#define TOGGLE FLAGS, 1
#define HIGHLow FLAGS, 2

;****************************************************************************
** Start of Program *********************************************************
****************************************************************************

org 0x1FF ; processor reset vector

; Internal RC calibration value is placed at location 0x1FF by
; Microchip as a movlw k, where the k is a literal value.

org 0x000 ; coding begins here
movwf OSCCAL ; update register with factory cal val
goto RESET ; initialize the program
NOTE: The following include file is available on Microchip’s webpage.
Fl51xInc.asm includes the necessary functions for reading and
writing to the internal EEPROM of the PIC12CE518.

#include <fl51xinc.asm>; EEPROM functions

RESET
    clrf    FLAGS ; clear flags
    clrf    GPIO ; initialize inputs and outputs
    movlw   BEGN ; setup the state counter to call BEGIN
    movwf   STATECNTR
    goto    MAIN ; goto MAIN

MAIN
    call    INITIAL
    call    TIMER
    call    CLOCK
    movlw   B'000111' ; check if S0, S1, or S2 is set
    andwf   GPIO, W ; if set call SXON
    btfss   STATUS, Z
    call    SXON
    btfsc   LRN ; if learn button is pushed call
    call    LRNDTCT ;   LRNDTCT
    btfsc   LERN ; if in learn mode call LEARN
    movf    STATECNTR, W ; Mask out the high order bits of
    andlw   B'00001111' ; STATECNTR (a noise guard)
    addwf   PCL, F ; The program clock (PCL) is incre-
    goto    BEGIN ;   mented by STATECNTR in order
    goto    HEADER
    goto    HEADER1
    goto    HIGHPLSE
    goto    LOWPULSE
goto RECORD

; These RESET commands correct
goto RESET
; erronious values of STATECNTR
goto RESET
; not caught by the mask above.
goto RESET

goto RESET

goto RESET

goto RESET

;*********************************************************************
;  INITIAL
;  This routine is continually called, initializing the OPTION
;  and GPIO registers in addition to clearing the watchdog timer.
;  This is done to insure that over the lifetime up the chip,
;  these vital registers will never change due to noise.
;
; Output Variables: none
; Input Variables: none
;*********************************************************************

INITIAL

clrwdt ; clear the watchdog timer

movlw GP_INITIAL ; setup the input and output pins

tris GPIO

movlw PRESCL ; setup TMR0 prescaler

option

retlw 0

;*********************************************************************
;  SETWATCH
; Initialize the pulse width timer registers.
;
; Input Variables:
; none
; Output Variables:
; ORIGIN
;*********************************************************************

SETWATCH

movf TMR0, W ; record TMR0’s value in ORIGIN

movwf ORIGIN

clrwf TMRLOW ; clear the low and high order timers

clrwf TMRHIGH

retlw 0

;*********************************************************************
;  CLOCK
; Continually updates TMRLow and TMRHIGH.
;
; Input Variables:
; ORIGIN
; Output Variables:
; TMRLow
; TMRHIGH
;*********************************************************************
**CLOCK**
```
movf ORIGIN, W ; TMRLow is updated based on time
subwf TMRO, W ; passed since ORIGIN was set.
addwf TMRLow, F ; TMRLow resolution = 4us (like TMRO)
btfsc STATUS, C ; TMRLow overflow = 1ms (2^8*4ms)
icnf TMRHIGH, F ; TMRHIGH resolution = 1ms
nop ; TMRHIGH overflow = 0.24sec (2^8*1ms)

nop ; Nop and subtraction commands ensure
movlw 2 ; ORIGIN equals TMRO as called upon
subwf TMRO, W ; in line 2 of CLOCK. (ORIGIN must
movwf ORIGIN ; be updated to equal the value
retlw 0 ; of TMRO at time of operation with
\(\text{ORIGIN}.\))
```

**TIMER**
Continually updates two higher order timers (SX1TMR and SX2TMR) for use in LED timing.
```
; Input Variables:
; none
; Output Variables:
; SX1TMR
; SX2TMR
```

**SXON**
```
btfs-toggle ; TOGGLE forces this routine to spend
goto TIMER1 ; 1/2 of TMRO in TIMER and 1/2 in
movlw B'01111111'; TIMER1.
addwf TMRO, W ; TOGGLE toggles back and forth to a
btfs STATUS, C ; one the rate TMRO overflows.
retlw 0 ; TMRO overflow = 1ms (2^8*4us)
bcf TOGGLE ;
icnf Fsz SX1TMR, F ; SX1TMR resolution = 1ms
retlw 0 ; SX1TMR overflow = 0.25sec (2^8*1ms)
icnf SX2TMR, F ; SX2TMR resolution = 0.25sec
retlw 0 ; SX2TMR overflw = 1min (2^8*0.23sec)
```

**TIMER1**
```
movlw B'01111111'; ; Timer routine spends half its time
addwf TMRO, W ; in TIMER1 waiting to set TOGGLE
btfs STATUS, C ; to one again
retlw 0 ;
bsf TOGGLE
```

**SXON**
```
; Turns all outputs (S0, S1, S2) off when they timeout.
```
```
; Input Variables:
; SX2TMR
; Output Variables:
; S0
; S1
; S2
```
SXON
btfss SX2TMR, 0 ; When SX1TMR overflows, SX2TMR
retlw 0 ; will increment to 1. Recall this
bcf S0 ; will occur at 0.25 seconds
bcf S1 ; (2^8*1ms) after SX1TMR is
bcf S2 ; initiated.
retlw 0

;*****************************************************************************

; LRNDTCT
; When the LEARN button is pushed this function places the
; program in LEARN mode by setting the LERN flag.
;
; Input Variables:
; none
;
; Output Variables:
; LED
; LERN
;*****************************************************************************

LRNDTCT
btfsc LERN ; LEARN mode is initiated by setting
retlw 0 ; the LERN flag high, setting the
movlw BEGN ; State Counter to BEGN, turning the
movwf STATECNTR ; learn LED on and clearing the
bsf LERN ; higher order timers, SX1TMR and
bsf LED ; SX2TMR.
clrfr SX1TMR
clrfr SX2TMR
retlw 0

;*****************************************************************************

; LEARN
; This routine learns the first two bytes of data received from
; the transmitter by storing these bytes in its internal EEPROM.
;
; Input Variables:
; none
;
; Output Variables:
; none
;*****************************************************************************

LEARN
btfss SX2TMR, 5 ; If no valid reception is completed
goto LEARN1 ; within 8 seconds (2^5*0.25sec)
bcf LERN ; then exit LEARN mode, else goto
bcf LED ; LEARN 1.
goto MAIN

LEARN1
movlw VALID ; If the State Counter currently holds
xorwf STATECNTR, W ; the value for executing the
btfss STATUS, Z ; VALIDATE function, then a successful
retlw 0 ; reception has occurred.
movlw 0x00 ; Setup the EEADDR register to write
movwf EEADDR ; to the first EEPROM byte.
movlw DATA0 ; Move DATA0’s address into the FSR
movwf FSR ; register. (See 12CE518 data sheet
movlw 2 ; for indirect referencing.
movwf BITCNTR

LEARN2 movf INDF, W ; Move contents of address specified
        movwf EEDATA ; by FSR into EEDATA.

LEARN3 call WRITE_BYTE ; write to EEPROM
        btfss PC_OFFSET, 7 ; If an error occurred while writing,
goto LEARN3 ; try again.
        incf EEADDR, F
        incf FSR, F
        decfsz BITCNTR, F ; perform write sequence for two bytes
        goto LEARN2
        bcf LED ; exit learn mode
        bcf LERN
        movlw BEGN
        movwf STATECNTR
        retlw 0

;*********************************************************************
; BEGIN
; This function looks for a possible start to the data stream.
;
; Input Variables:
; STREAM
; Output Variables:
; none
;*********************************************************************
BEGIN
        btfsb STREAM
        incf STATECNTR, F ; Make state BEGIN1
        goto MAIN
BEGIN1
        btfsb STREAM
        goto MAIN
        call SETWATCH
        incf STATECNTR, F ; Make state HEADER
        goto MAIN

;*********************************************************************
; HEADER
; Detects a valid header.
;
; Input Variables:
; STREAM
; Output Variables:
; none
;*********************************************************************
HEADER
        btfsb STREAM ; The program loops here until 1.25ms
        goto RESTART ; passes and if the data is still
        btfsb TMRHIGH, 0 ; low. If both hold true -> HEADER1.
        goto MAIN ; 1.25ms occurs when:
        movlw D'64' ; TMRHIGH = 1 <= 2^8*4us = 1ms
        andwf TMRLow, w ; TMRLow = 64 <= 64*4us = 0.25ms
        btfsb STATUS, Z
goto MAIN
incf STATECNTR, F ; Make state HEADER1
goto MAIN

HEADER1
movlw D'6' ; If the data goes high before 6ms
subwf TMRHIGH, W ; then the header is valid, else
btfss STATUS, C ; restart.
goto HEADER2 ; TMRHIGH = 6 = 6*1ms = 6ms
goto RESTART

HEADER2
btfss STREAM
goto MAIN
call SETWATCH
movlw D'32' ; Initiate BITCNTR to 32 in order to
movwf BITCNTR ; receive 32 bits of the data stream.
icnf STATECNTR, F ; Make state HIGHPLSE
goto MAIN

;*********************************************************************
;  HIGHPLSE
; Times the width of high pulses.
;
; Input Variables:
; STREAM
;
; Output Variables:
; none
;*********************************************************************

HIGHPLSE
btfsc TMRHIGH, 0 ; If TMRLOW overflows then RESTART
goto RESTART
btfsc STREAM
goto MAIN
movf TMRLOW, W ; Move the pulse width value to
movwf HIGHWDTH ; HIGHWDTH for later calculations.
call SETWATCH
incf STATECNTR, F ; Make state LOWPULSE
goto MAIN

;*********************************************************************
;  LOWPULSE
; Times the width of low pulses.
;
; Input Variables:
; none
;
; Output Variables:
; none
;*********************************************************************

LOWPULSE
btfsc TMRHIGH, 0 ; If TMRLOW overflows then make
goto LOW2 ; state HEADER.
btfss STREAM
goto MAIN
movf TMRLOW, W ; Move the pulse width value to
movwf LOWWDTH ; LOWWDTH for later calculations.
call SETWATCH
incf STATECNTR, F ; Make state RECORD
LOW2

goto MAIN

movlw HEADR
movwf STATECNTR ; Make state HEADER if lowpulse is too

goto MAIN ;   long.

;*********************************************************************
; RECORD
;   Records each bit as it comes in from the data stream.
;
; Input Variables:
;   STREAM
; Output Variables:
;   DATA0
;   DATA1
;   DATA2
;   DATA3
;*********************************************************************

RECORD

movf HIGHWDTH, W
subwf LOWWDTH, W ; The state of the carry bit after
rrf DATA3, F ;   this operation reflects the data
rrf DATA2, F ;   logic. This is then rotated
rrf DATA1, F ;   into the storage bytes.
rrf DATA0, F

movlw HIGHP
movwf STATECNTR

decfsz BITCNTR, F

goto MAIN

movlw D'4' ; Starting here and including RECORD1
movwf COUNTR ;   a check is made to make sure that
movlw DATA0  ;   the data is not composed entirely
movwf FSR ;   of 1s.

RECORD1

movlw 0xFF
xorwf INDF, W ; Use indirect referencing (see the
btfss STATUS, Z ;   12CE518 data sheet) to point to
goto RECORD2 ;   DATA0 -- DATA3 on subsequent loops
incf FSR, F ;   in RECORD1.

decfsz COUNTR, F

goto RECORD1

goto RESTART

RECORD2

movlw WAIT
movwf STATECNTR ; Make state WAIT4END

goto MAIN

;*********************************************************************
; WAIT4END
;   Wait for the guard time at the end of the code word before
; attempting to receive another code word.
;
; Input Variables:
;   STREAM
; Output Variables:
;   none
;*********************************************************************
WAITH4END

```
btfsc HIGHLOW
    goto WAIT1
    btfsc STREAM
        goto MAIN
        call SETWATCH
        bsf HIGHLOW

WAIT1

```

```
btfss STREAM
    goto WAIT2
    bcf HIGHLOW
    goto MAIN

WAIT2

```

```
btfss TMRHIGH, 3
    goto MAIN
    bcf HIGHLOW
    incf STATECNTR, F
    goto MAIN
```

```
;*****************************************************************************
; VALIDATE
;  Checks that the transmission received is from the valid
;  transmitter.
;
; Input Variables:
;    DATA0
;    DATA1 (only the first two bits)
;
; Output Variables:
;    none

;*****************************************************************************

VALIDATE

```
movlw 0x00
    movwf EEADDR
movlw DATA0
    movwf DATA0
    movlw DATA1
    movwf DATA1
    movlw 0x00
    movwf FSR
    movlw 0x00
    movwf FSR
```

```
call READ_RANDOM
```

```
btfss PC_OFFSET, 7
    goto VAL1
    movf INDF, W
    xorwf EEDATA, W
    btfss STATUS, Z
    goto RESTART
    incf FSR, F
    incf EEADDR, F
```

```
VAL1
```

```
call READ_RANDOM
```

```
btfss PC_OFFSET, 7
    goto VAL1
    movf INDF, W
    xorwf EEDATA, W
    btfss STATUS, Z
    goto RESTART
    incf FSR, F
       incf EEADDR, F
```

```
VAL2
```

```
call READ_RANDOM
```

```
btfss PC_OFFSET, 7
    goto VAL2
    movf INDF, W
    xorwf EEDATA, W
    btfsc EEDATA, 0
    goto RESTART
    btfsc EEDATA, 1
    goto RESTART
    incf STATECNTR, F
    goto MAIN
```

```
```
;*********************************************************************
; IMPLEMENT
; Implements the outputs specified by the received code word.
;
; Input Variables:
; DATA1
;
; Output Variables:
; S0
; S1
; S2
;*********************************************************************

IMPLEMENT
    btfsc DATA1, 7 ; set outputs in accordance with code
    bsf S2 ; word
    btfss DATA1, 7
    bcf S2
    btfsc DATA1, 6
    bsf S1
    btfss DATA1, 6
    bcf S1
    btfsc DATA1, 5
    bsf S0
    btfss DATA1, 5
    bcf S0
    clrf SX1TMR ; initialize the timers for the
    clrf SX2TMR ; outputs
    goto RESTART

;*********************************************************************
; RESTART
; Sets the State Counter to BEGIN so that the receive sequence
; is restarted.
;
; Input Variables:
; none
;
; Output Variables:
; none
;*********************************************************************

RESTART
    movlw BEGN ; restart receive sequence and return
    movwf STATECNTR ; to MAIN
    goto MAIN

;********************************************************************************

end ; directive ‘end of program’
“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, SelectMode 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

**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: 972-816-7423 Fax: 972-816-2924

**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: 248-539-2250 Fax: 248-539-2260

**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: 408-436-7995

**Toronto**
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0999 Fax: 905-673-6509

### ASIA/PACIFIC

**Australia**
Microchip Technology Australia Pty Ltd
Suite 22, 41 Lawson Street
Epping, 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’Shaunessy Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

**Japan**
Microchip Technology Int'l. 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
661-1, 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, 189980
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 høj 1-3
Ballup ØK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

**France**
Arizona Microchip Technology SARL
Parc d’Activité du Moulin de Massy
43 Rue du Saule Trapu
Bâtiment 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

**Greece**
Analog Product Sales
Lochhammer 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

**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

**United Kingdom**
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 3TU
Tel: 44 118 921 5869 Fax: 44 118 921-5820

---

All rights reserved. © 2001 Microchip Technology Incorporated. Printed in the USA. 01/30/01

01/30/01

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.