MICROCHIP

AN665

Using KEELOQ® to Generate Hopping Passwords

Author: *Lucio Di Jasio*
Arizona Microchip Technology, Italy

INTRODUCTION

The purpose of this application note is to demonstrate how KEELOQ® code hopping technology can be conveniently employed to implement an automatic code hopping password generator/keypad. Using a PIC12C508, the hopping code produced by an HCS300 is converted to a string of 16 hex digits. This string is then transferred to the PC via the keyboard line, thereby emulating the actual pressure of a sequence of keys on a standard PC/AT® keyboard. Since this conversion process is transparent to any application, it appears as if the user is simply typing on a PC/AT-type keyboard.

An ideal situation for implementing this application would be in creating a "super password" for general, access-control secure logins when transmitting information onto the internet (i.e., through a browser) or a Java applet.

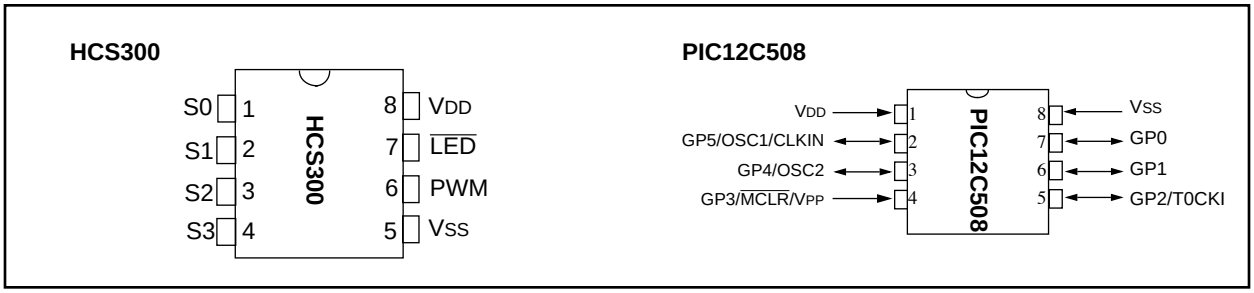
THE "HOPPING" ADVANTAGE

Password-based access control systems are very popular today, but the level of security they provide are often overestimated. Being basically a unidirectional transmission, a password-based system has two very important shortcomings which can lead to unauthorized access: the code is fixed, and the number of possible combinations is relatively low.

The growing speed of communication lines and the computing power of available systems increases the chance of a brute force attack or "code scanning." The use of unsecure means of transmission, where code "grabbing" is possible (i.e., a typical modem connection over phone lines), can make the use of a fixed code highly undesirable. Note that these are the same situation that led to the introduction of the "code hopping" concept in the remote control market.

The basic idea is to have the access code change each time it is used through a sequence where the new codes cannot be predicted even knowing a very large number of previously used ones. Producing such a sequence requires the use of a solid encryption engine. Microchip Technology is currently offering a broad range of encoders based on the proprietary KEELOQ code hopping technology. These encoders make producing a code hopping remote control easy, but as we will see, can also be conveniently used to add the hopping advantage to old password based access control systems in a transparent way.

FIGURE 1: HCS300 AND PIC12C508 PINOUT DIAGRAMS



INTRODUCTION TO KEELOQ ENCODERS

All KEELOQ encoders use the KEELOQ code hopping technology to make each transmission by an encoder unique. The encoder transmissions have two parts. The first part changes each time the encoder is activated and is called the code hopping part and is encrypted. The second part is the unencrypted part of the transmission, principally containing the encoder serial number identifying it to a decoder.

The code hopping contains function information, a discrimination value, and a synchronization counter. This information is encrypted by an encryption algorithm before being transmitted. A 64-bit encryption key is used by the encryption algorithm. If one bit in the data that is encrypted changes, the result is that an average of half the bits in the output will change. As a result, the code hopping changes dramatically for each transmission and can not be predicted.

The synchronization information is used at the decoder to determine whether a transmission is valid or is a repetition of a previous transmission. Previous codes are rejected to safeguard against code grabbers.

The HSC300 and HCS301 encoders transmit two overflow bits which may be used to extend the range of the synchronization counter from 65,536 to 196,608 button operations. The HCS300 and HCS301 encoders include provision for four bits of function information and two status bits in the fixed code portion of its transmission. The two status bits indicate whether a repeated transmission is being sent, and whether the battery voltage is low.

The Microchip HCSXXX encoders all have the ability to transmit a fixed seed. The seed value is programmed into the encoder when the encoder is first initialized along with the counters, key, serial number, and other information. The seed length differs from encoder to encoder, with the HCS300 and HCS301 having a 32-bit seed.

FIGURE 2: KEELOQ ENCODER CODE WORD TRANSMISSION FORMAT

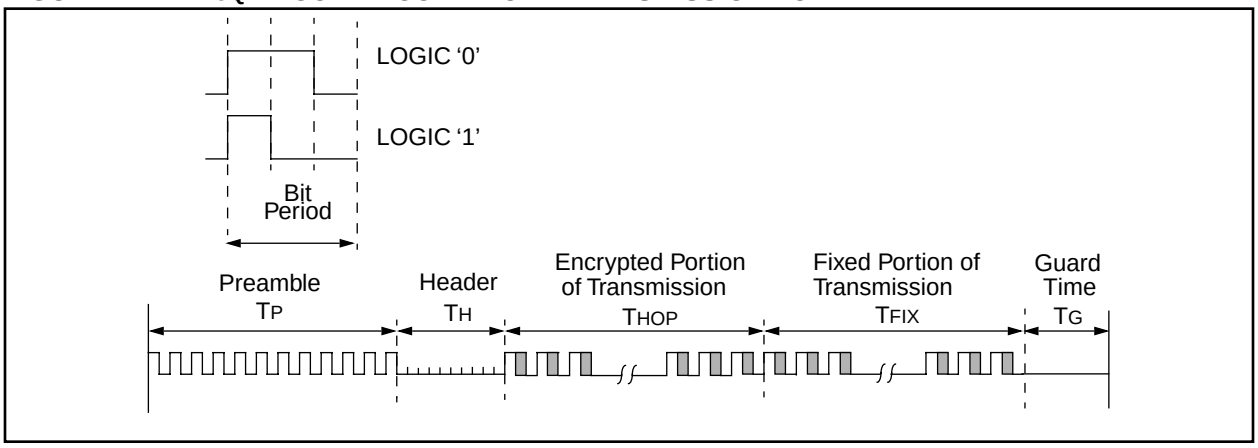


FIGURE 3: KEELOQ ENCODER CODE WORD ORGANIZATION

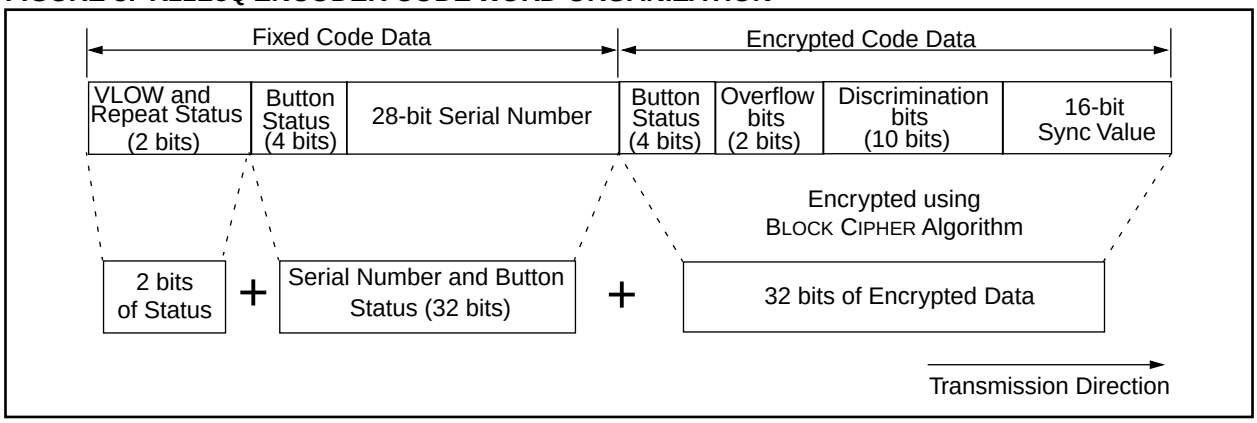
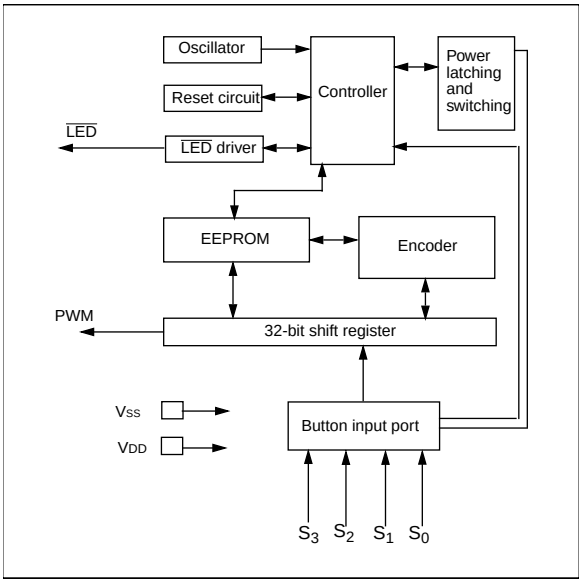


FIGURE 4: HCS300 BLOCK DIAGRAM



The IBM PC-AT® Keyboard Protocol

IBM® was the first to introduce the synchronous serial protocol most of today's PC-ATs use to communicate with a keyboard. This now-standard, 5-pole shielding connector (Figure 5) carries the clock line, data line, ground, and +5V power supply in order to transmit data bidirectionally from the keyboard to the PC.

Typically, data travelling from the keyboard to the PC is accomplished by either key pressure or release information. However, some configuration data (i.e., repeat, delay, and rate) can flow in the opposite direction – for example, during a system boot. The keyboard drives the clock line by using open collector drivers. To disable the keyboard, the PC can keep the clock line low. If the data line is held low by the PC while the clock line is high, the computer transmits a requests to send, and the keyboard goes into receive mode. The keyboard is only allowed to send data when both the clock line and data line are high.

FIGURE 5: STANDARD 5-POLE CONNECTOR

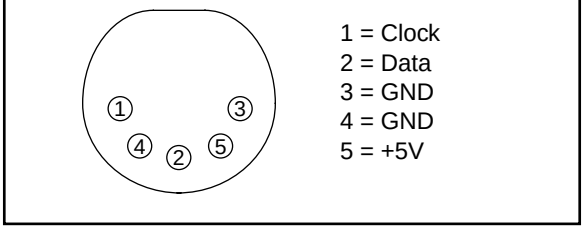
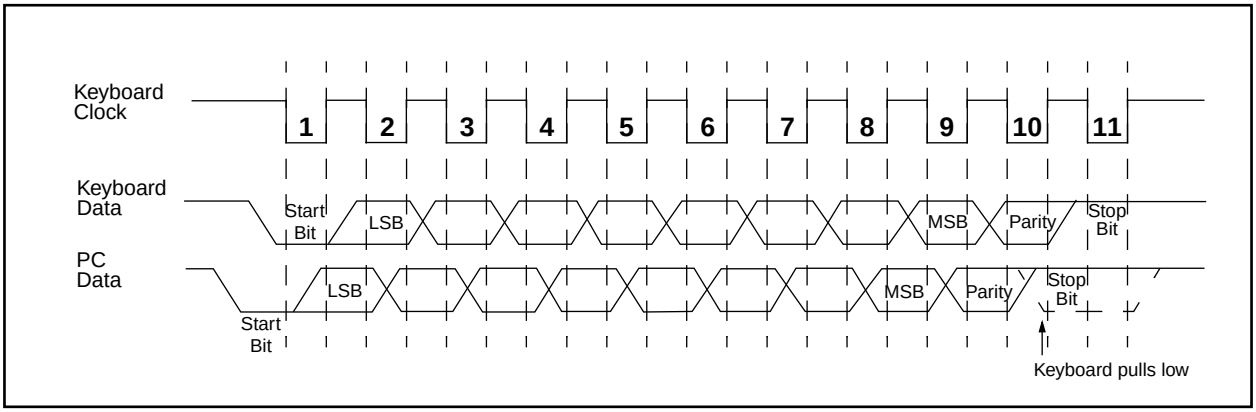


FIGURE 6: AT® KEYBOARD PROTOCOL



Keyboard Transmission

The keyboard pulls the data line low (start bit) and starts the clock. The eight data bits (least significant bit first) are shifted out, followed by the parity (odd), and stop bit (high). Data is valid after the falling edge of the clock and changes after the rising edge of the clock. If no data is transmitted, both the clock line and data line are high. If the computer pulls the clock line low for at least 60 μs before the tenth bit is transmitted, the keyboard stops the transmission and stores the aborted data in a buffer for retransmission at a later time.

Keyboard Receiving

The computer pulls the data line low (start bit), after which the keyboard starts to shift out 11 clock pulses within 15 ms. Transmission has to be completed within 2 ms. Data from the computer changes after the falling edge of the clock line, and is valid before the rising edge of the clock. After the start bit, eight data bits (least significant bit first), followed by the parity bit (odd), and the stop bit (high) are shifted out by the computer with the clock signal provided by the keyboard. The keyboard pulls the stop bit low in order to acknowledge the receipt of the data. If a transmission error occurs (parity error or similar) the keyboard issues a "RESEND" command to the PC.

Key Pressure Release Encoding

Key pressure is communicated to the PC by sending a scan code. Table 1 lists the scan codes corresponding to keys '0'…'F'. Release is communicated by sending the break code (0F0), followed by the previous scan code.

TABLE 1: SCAN CODES

Codes	Key
45	'0'
16	'1'
1E	'2'
26	'3'
25	'4'
2E	'5'
36	'6'
3D	'7'
3E	'8'
46	'9'
1C	'A'
32	'B'
21	'C'
23	'D'
24	'E'
2B	'F'

Proposing a Demo Keypad/Dongle Implementation

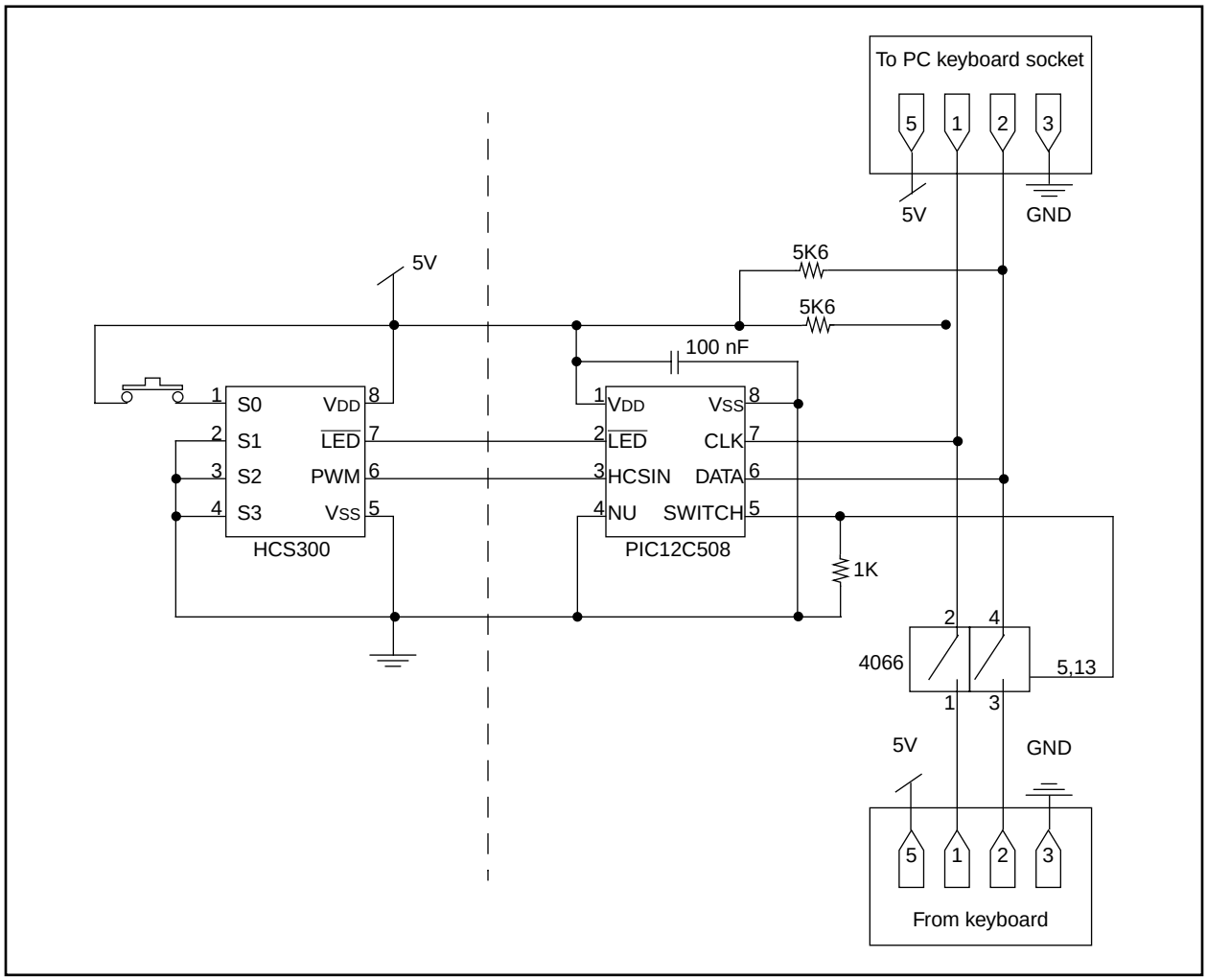
The password generator fits between the keyboard and the PC. A 5-pin plug connects to the PC, supplying power to our device, and the keyboard plugs into the 5-pin socket (Figure 7). The clock and data lines are passed between the PC and keyboard, allowing normal keyboard operations. When S1 is activated, the PIC12C508 receives the new message (16 hex digits) produced by the KEELOQ HCS300 Encoder. The PIC12C508 will then emulate the keyboard, sending the appropriate sequence of key press and key release messages to the PC. To prevent the keyboard from interpreting this transmission as a 'request to send' from the PC, it is necessary to isolate the keyboard from the clock line and data line during the transmission.

The KEELOQ HCS300 Encoder can be part of the dongle or can be removable, like a key, in order to allow different encoders with different encryption keys or serial numbers to be easily exchanged.

Power consumption has to be the lowest possible in order not to excessively load the line. Size and component count should also be kept to the possible minimum in order to allow for a very small package. Ideally, the whole circuit should fit into a small gap between the two connectors.

In the implementation we are proposing, an HCS300 KEELOQ code hopping encoder is used together with a PIC12C508 microcontroller.For simplicity, a standard CMOS quadruple switch (4066) is used to alternatively connect the dongle or the keyboard to the PC line. The HCS300 and the PIC12C508 (both available in 8-pin DIP or SOIC packages), draw extremely low currents as well as internally produce the clock required to operate the dongle. Beside a couple of pull-up resistors required for the clock line and data line, no other components are required to obtain a fully functional hopping password dongle (Figure 7).

FIGURE 7: KEYPAD/DONGLE SCHEMATIC



Software Implementation

The software is composed of three short code segments:

- Receive routine for the KEELOQ HCS300 Encoder
- Keyboard Emulation Routines
- Main loop routine.

Receive Routine for the HCS300 Encoder (RECEIVE)

The RECEIVE routine gathers the first 64 bits transmitted by the HCS300, ignoring the last two bits (repeat and battery status) as they carry no useful information for this application, and packs them into a 8-bytes buffer (Buffer0…Buffer7).

Keyboard Emulation Routines (Sendbit, SendKey)

These implement the transmission of the key scan codes according to the IBM-PC/AT keyboard protocol.

Main Loop

While the CMOS switch connects the PC to the keyboard clock and data lines, the $\overline{\text{LED}}$ output line is continuously polled to detect the activation of the HCS300

Note: Any combination of its four input lines after debouncing activates the encoder.

When the $\overline{\text{LED}}$ line goes low, the CMOS switch is activated to isolate the clock and data lines from the keyboard. The RECEIVE routine is called.

Upon successfully receiving a transmission, a loop is entered where 16 hex digits from the receive buffer are transmitted as a sequence of key press and key release messages, separated by appropriate delays repetitively calling the SENDKEY subroutine.

The software has been developed in the simplest possible form and, therefore, is open to a number of optimizations. For example:

- The PIC12C508 could be put to “sleep” to further reduce power consumption.
- The encoder could be removable and its presence/activation should be properly detected.
- For simplicity, the presented RECEIVE routine requires a 400 μs transmission speed being configured in the encoder, while a more flexible multi-baud rate routine can be used as presented in various other application notes.
- A second code word could be compared with the first code word received to recognize transmission errors (although highly improbable when the encoder is wired to the PIC12508), since there is no decryption, there is no other means to tell that the transmission has not been corrupted.

Decoding Options

A code hopping password can be used to validate access to a wide variety of electronic services providing the recipient application (typically, it will be some software running on a server) is capable of following some simple decryption and verification steps. The fixed unencrypted part of the code (last 8 digits) can be used to identify the user (7 digits) and the function activated on the encoder (1 out of 15, corresponding to the last digit).

The hopping part has to be decrypted using the appropriate 64-bit decryption key. Depending on the desired level of security, many different key generation and management techniques can be adopted. For example, the key could be deduced by the User ID and a Manufacturer’s Key, by the “seed” code of the encoder or could simply be a fixed 64-bit constant.

Learning techniques can also be applied so that the application actually autonomously acquires the required keys and builds a database of users, ID codes, and decryption keys. For a further analysis, consult the following literature:

AN645	*PIC16C57 Based Code Hopping Security System* (DS00645)
AN662	*KEELOQ Code Hopping Decoder Using Secure Learn* (DS00662)
AN663	*KEELOQ Simple Code Hopping Decoder* (DS00663)
TB001	*An Introduction to KeeLoq Code Hopping* (DS91000A)
TB003	*An Introduction to KEELOQ Code Hopping* (DS91002A).

.

Please check the Microchip Worldwide Web at www.microchip.com for the latest version of the source code.

APPENDIX A: HOPPASW.LST

```
MPASM 01.40 Released          HOPPASW.ASM   2-20-1997  16:45:24         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001     LIST        n=0, c=132
                     00002     PROCESSOR   PIC12C508
                     00003     RADIX       HEX
                     00004
                     00005 ;* filename: HOPPASW.ASM
                     00006 ;*****************************************************************
                     00007 ;*      Author:             Lucio Di Jasio
                     00008 ;*      Company:            Microchip Technology
                     00009 ;*      Revision:           RevA0
                     00010 ;*      Date:               4-sept-96
                     00011 ;*      Assembled using:    MPASM rev. 1.40
                     00012 ;*****************************************************************
                     00013 ;*      include files:
                     00014 ;*          p12C508.inc
                     00015 ;*
                     00016 ;*****************************************************************
                     00017 ;*     HCS300 to keyboard interface, for hopping password generation
                     00018 ;*
                     00019 ;*                /                /
                     00020 ;*        HCS300  |5V              |5V
                     00021 ;*      +-------+ |                | +--------+
                     00022 ;* Key1-+S1     +-+               +-+Vdd     +-----GND
                     00023 ;* Key2-+S2  LED+-------------------+LED     +-----PCCLK
                     00024 ;* Key3-+S3  PWM+-------------------+HCSIN   +-----DATA
                     00025 ;* Key4-+S4     +-+                 -+nc      +-----SWITCH
                     00026 ;*      +-------+ |                  +--------+
                     00027 ;*                |
                     00028 ;*               VGnd
                     00029 ;*
                     00030 ;*****************************************************************
                     00031
                     00032     INCLUDE "\pic\include\p12c508.inc"
                     00001         LIST
                     00002 ; P12C508.INC Standard Header File, Vers 1.01 Microchip Technology. Inc.
                     00103         LIST
                     00033
0FFF 0FEA            00034     __CONFIG _IntRC_OSC & _MCLRE_OFF & _CP_OFF & _WDT_OFF
01FF 0000 0003 0001  00035     __IDLOCS H'0312'
     0002
                     00036
                     00037 ;*****************************************************************
                     00038 ;* internal 4MHz clock
                     00039 ;* internal reset
                     00040 ;* no code protect (?)
                     00041 ;* no watchdog
                     00042 ;* ID code is "0312"
                     00043 ;*****************************************************************
                     00044
                     00045 ;
                     00046 ; pin description
                     00047 ;
                     00048 #define LED         GPIO,5      ; Led from HCS300
```

```
                  00049 #define HCSIN       GPIO,4      ; PWM from HCS300
                  00050 #define NC          GPIO,3      ; not used
                  00051 #define SWITCH      GPIO,2      ; sense clock line to/from PC
                  00052 #define DATA        GPIO,1      ; data to PC
                  00053
                  00054 #define PCCLK       0           ; clock to PC
                  00055
                  00056 #define MASKDEF     b'11111011' ; only SWITCH pin in output
                  00057 #define MASKLOW     b'11111001' ; prepare data low
                  00058 #define MASKHIGH    b'11111011' ; prepare data high
                  00059
                  00060 ;
                  00061 ; RAM assignments
                  00062 ;
                  00063     CBLOCK  07
  00000007        00064         BUFFER0         ; receive buffer for hcs300
  00000008        00065         BUFFER1
  00000009        00066         BUFFER2
  0000000A        00067         BUFFER3
  0000000B        00068         BUFFER4
  0000000C        00069         BUFFER5
  0000000D        00070         BUFFER6
  0000000E        00071         BUFFER7
  0000000F        00072         BITCOUNT        ; counters
  00000010        00073         BYTECOUNT       ;
  00000011        00074         TIMEHI          ; timing
  00000012        00075         TIMELO          ;
  00000013        00076         PARITY          ; transmission parity bit
  00000014        00077         AUX             ; g.p.
  00000015        00078         KEY             ; key to encode
  00000016        00079         GPIOTEMP        ; temp copy of tris register
                  00080     ENDC
                  00081 ;-----------------------------------------------------------------------
                  00082
0000              00083             ORG     0
                  00084
0000              00085 Start
0000 0025         00086             movwf   OSCCAL      ; calibrate
0001 0A7A         00087             goto    Main
                  00088
                  00089 ;-----------------------------------------------------------------------
                  00090 ; keyscan table
                  00091 ;
0002              00092 ScanCode
0002 01E2         00093             addwf   PCL,F
0003 0845         00094             retlw   45          ; key '0'
0004 0816         00095             retlw   16          ; key '1'
0005 081E         00096             retlw   1E          ; key '2'
0006 0826         00097             retlw   26          ; key '3'
0007 0825         00098             retlw   25          ; key '4'
0008 082E         00099             retlw   2E          ; key '5'
0009 0836         00100             retlw   36          ; key '6'
000A 083D         00101             retlw   3D          ; key '7'
000B 083E         00102             retlw   3E          ; key '8'
000C 0846         00103             retlw   46          ; key '9'
000D 081C         00104             retlw   1C          ; key 'A'
000E 0832         00105             retlw   32          ; key 'B'
000F 0821         00106             retlw   21          ; key 'C'
0010 0823         00107             retlw   23          ; key 'D'
0011 0824         00108             retlw   24          ; key 'E'
0012 082B         00109             retlw   2B          ; key 'F'
                  00110
                  00111 #define     BREAK   0F0         ; break scan code
                  00112
                  00113 ;*****************************************************************
                  00114 ;*  SubDelay
```

```
                  00115 ;*      short delay functions N us
                  00116 ;*
                  00117 ;*      Input Variables:
                  00118 ;*          none
                  00119 ;*      Output Variables:
                  00120 ;*          none
                  00121 ;************************************************************************
                  00122 ;
0013 0000         00123 SubDelay10    nop
0014 0000         00124 SubDelay9     nop
0015 0000         00125 SubDelay8     nop
0016 0000         00126 SubDelay7     nop
0017 0000         00127 SubDelay6     nop
0018 0000         00128 SubDelay5     nop
0019 0800         00129 SubDelay4     retlw   0           ; 2 call + N nop + 2 retlw
                  00130
                  00131 ;************************************************************************
                  00132 ;*  Wait10ms
                  00133 ;*      waits for approx 10ms
                  00134 ;*
                  00135 ;*      Input Variables:
                  00136 ;*          none
                  00137 ;*      Output Variables:
                  00138 ;*          none
                  00139 ;************************************************************************
                  00140 ;
001A              00141 Wait10ms
001A 0C0F         00142             movlw   .15         ; 15 * .7ms ~= 10ms@4MHz
001B              00143 WaitWx750
001B 0031         00144             movwf   TIMEHI
001C              00145 WaitHi
001C 0072         00146             clrf    TIMELO      ; 256 * 3us ~= 750us@4MHz
001D              00147 WaitLo
001D 02F2         00148             decfsz  TIMELO,F
001E 0A1D         00149             goto    WaitLo
001F 02F1         00150             decfsz  TIMEHI,F
0020 0A1C         00151             goto    WaitHi
0021 0800         00152             retlw   0
                  00153
                  00154 ;************************************************************************
                  00155 ;*  SendBit
                  00156 ;*      sends a bit in AT keyboard protocol
                  00157 ;*
                  00158 ;*      Input Variables:
                  00159 ;*          STATUS,C
                  00160 ;*      Output Variables:
                  00161 ;*          none
                  00162 ;************************************************************************
                  00163 ;
0022              00164 SendBit
0022 0066         00165         clrf    GPIO        ; disable kb and clear out buffers
0023 0CF9         00166         movlw   MASKLOW     ; DATA low prepare
0024 0603         00167         btfsc   STATUS,C
0025 0CFB         00168         movlw   MASKHIGH    ; DATA high prepare
0026 0036         00169         movwf   GPIOTEMP    ; save value
0027 0006         00170         tris    GPIO
                  00171
0028 0C0E         00172         movlw   .14
0029 0032         00173         movwf   TIMELO
002A              00174 SBitT
002A 02F2         00175         decfsz  TIMELO,F        ; 45us loop (data stable)
002B 0A2A         00176         goto    SBitT
                  00177
002C 0416         00178         bcf     GPIOTEMP,PCCLK  ; clk fall
002D 0216         00179         movf    GPIOTEMP,W
002E 0006         00180         tris    GPIO
```

```
                  00181
002F 0C0F         00182         movlw   .15
0030 0032         00183         movwf   TIMELO
0031              00184 SBitT2
0031 02F2         00185         decfsz  TIMELO,F        ; 45us loop (data stable)
0032 0A31         00186         goto    SBitT2
                  00187
0033 0516         00188         bsf     GPIOTEMP,PCCLK  ; clk rise
0034 0216         00189         movf    GPIOTEMP,W
0035 0006         00190         tris    GPIO
0036 0800         00191         retlw   0
                  00192
                  00193 ;*********************************************************************
                  00194 ;*  SendKEY
                  00195 ;*      sends a scan code to the PC
                  00196 ;*
                  00197 ;*      Input Variables:
                  00198 ;*          W
                  00199 ;*      Output Variables:
                  00200 ;*          none
                  00201 ;*********************************************************************
                  00202 ;
0037              00203 SendKEY
0037 0034         00204             movwf   AUX         ; temp storage
                  00205
                  00206 ; wait PC ready
0038              00207 SendW
0038 0706         00208             btfss   GPIO,PCCLK  ; test PCCLK
0039 0A38         00209             goto    SendW       ; loop until HIGH
                  00210
                  00211 ; PC request ?
                  00212 ;           btfss   DATA        ; PC pull down Data?
                  00213 ;           goto    RecKEY      ; go receive first
                  00214
                  00215 ; send start bit
003A 0403         00216             bcf     STATUS,C
003B 0922         00217             call    SendBit
                  00218
                  00219 ; than shift out 8 bit LSB first
003C 0C08         00220             movlw   .8
003D 002F         00221             movwf   BITCOUNT
003E 0073         00222             clrf    PARITY
                  00223
003F              00224 SBitL
003F 0334         00225             rrf     AUX,F       ; next bit
0040 0603         00226             btfsc   STATUS,C
0041 02B3         00227             incf    PARITY,F    ; count parity
0042 0922         00228             call    SendBit
0043 02EF         00229             decfsz  BITCOUNT,F
0044 0A3F         00230             goto    SBitL       ; loop for 8 bit
                  00231
0045 02B3         00232             incf    PARITY,F    ; parity odd
0046 0333         00233             rrf     PARITY,F    ; send parity Bit
0047 0922         00234             call    SendBit
                  00235
0048 0503         00236             bsf     STATUS,C
0049 0A22         00237             goto    SendBit     ; send stop bit (high = released)
                  00238
                  00239
                  00240 ;*********************************************************************
                  00241 ;*  Receive
                  00242 ;*      receives first 64 bit transmitted by an HCS300 encoder
                  00243 ;*      simplified to operate with 400us PWM only
                  00244 ;*
                  00245 ;*      Input Variables:
                  00246 ;*          none
```

```
                  00247 ;*      Output Variables:
                  00248 ;*          BUFFER0..7
                  00249 ;*************************************************************************
                  00250 ;
004A              00251 Receive
004A 0686         00252             btfsc   HCSIN           ; wait for a falling edge
004B 0A4A         00253             goto    Receive
                  00254
                  00255 ; will accept sync pulses from  3.0 to 6.1 ms.
                  00256 ; more than 128 cycles but less than 256
                  00257 ; each cycle is 24 us @4MHz
                  00258 ;
  00000038        00259 PREBIT      EQU     .56
                  00260
004C 006F         00261             clrf    BITCOUNT        ; init counter
004D              00262 Rise
004D 0686         00263             btfsc   HCSIN           ; wait rising edge
004E 0A54         00264             goto    Rise2
                  00265
004F 0913         00266             call    SubDelay10      ; 24us per cycle
0050 0914         00267             call    SubDelay9       ;
                  00268
0051 03EF         00269             incfsz  BITCOUNT,F      ; more than 6,0ms timeout
0052 0A4D         00270             goto    Rise            ; waiting loop
0053 0A4A         00271             goto    Receive         ; timeout restart
                  00272
0054              00273 Rise2
0054 07EF         00274             btfss   BITCOUNT,7      ; if bit7=1 ok
0055 0A4A         00275             goto    Receive         ; else less than  3.0 ms timeout
                  00276
                  00277 ;-----------------------------------------------------------------------
                  00278 ; read folowing 8 bytes (ignore last 2 bit)
                  00279 ;
0056 0C40         00280             movlw   .64             ; 8 bit per byte
0057 002F         00281             movwf   BITCOUNT        ;
                  00282
0058              00283 FirstPreload
0058 0C38         00284             movlw   PREBIT          ; first bit needs no balance
0059 0031         00285             movwf   TIMEHI
005A 0A5D         00286             goto    WHL
                  00287
005B              00288 RNextBit
005B 0C36         00289             movlw   PREBIT-2        ; preload counter
005C 0031         00290             movwf   TIMEHI          ; balance extra rrf time
                  00291
005D              00292 WHL
005D 02B1         00293             incf    TIMEHI,F        ; measure high period
005E 0643         00294             btfsc   STATUS,Z
005F 0A4A         00295             goto    Receive         ; after 1.2ms (200*6) timeout
0060 0686         00296             btfsc   HCSIN           ; loop while High
0061 0A5D         00297             goto    WHL             ;
                  00298
0062 0C38         00299             movlw   PREBIT          ; preload counter
0063 0032         00300             movwf   TIMELO
                  00301
0064              00302 WLL
0064 02B2         00303             incf    TIMELO,F        ; measure low period
0065 0643         00304             btfsc   STATUS,Z
0066 0A4A         00305             goto    Receive         ; after 1.2ms (200*6) timeout
0067 0786         00306             btfss   HCSIN           ; loop while Low
0068 0A64         00307             goto    WLL
                  00308
                  00309 ; shift in the new bit
0069 0211         00310             movf    TIMEHI,W
006A 00B2         00311             subwf   TIMELO,F        ; if TIMEHI > TIMELO Carry = 0
006B 032E         00312             rrf     BUFFER7,F       ; insert bit in buffer
```

```
006C 032D          00313             rrf     BUFFER6,F
006D 032C          00314             rrf     BUFFER5,F
006E 032B          00315             rrf     BUFFER4,F
006F 032A          00316             rrf     BUFFER3,F
0070 0329          00317             rrf     BUFFER2,F
0071 0328          00318             rrf     BUFFER1,F
0072 0327          00319             rrf     BUFFER0,F
                   00320
                   00321 ; compare duty cycle, to skip preamble
0073 0CE0          00322             movlw   0E0             ; test duty cycle
0074 0152          00323             andwf   TIMELO,W        ; delta >200us? (32 cycles)
0075 0643          00324             btfsc   STATUS,Z
0076 0A4A          00325             goto    Receive         ; no! it's a preamble
                   00326
0077 02EF          00327             decfsz  BITCOUNT,F      ; loop to completion
0078 0A5B          00328             goto    RNextBit
0079 0800          00329             retlw   0
                   00330
                   00331 ;-----------------------------------------------------------------------
                   00332 ;***********************************************************************
                   00333 ;*  Main loop
                   00334 ;*      set TRIS and option register
                   00335 ;*      wait for start (LED)
                   00336 ;*      disable keyboard
                   00337 ;*      receive new hopping code
                   00338 ;*      send 16 hex digits
                   00339 ;*      wait transmission end
                   00340 ;*      loop
                   00341 ;***********************************************************************
                   00342 ;
007A               00343 Main
007A 0C04          00344             movlw   b'00000100' ; set switch ON
007B 0026          00345             movwf   GPIO
007C 0CFB          00346             movlw   MASKDEF     ; init port
007D 0006          00347             tris    GPIO
007E 0C00          00348             movlw   0
007F 0002          00349             option
                   00350
0080 06A6          00351             btfsc   LED             ; wait for Led output fall
0081 0A7A          00352             goto    Main
                   00353
0082               00354 Disable
0082 0066          00355             clrf    GPIO            ; send disable kb
                   00356                                     ; SWITCH = LOW
                   00357
0083 094A          00358             call    Receive         ; gets the new hopping code
                   00359
                   00360 ;-----------------------------------------------------------------------
                   00361 ; emulate a keyboard and send data as a sequence of 16 key
                   00362 ; pressed and released, one each hex digit
                   00363 ;
0084 0C08          00364             movlw   .8              ; 8 byte from the buffer
0085 0030          00365             movwf   BYTECOUNT
0086 0C07          00366             movlw   BUFFER0         ; init pointer
0087 0024          00367             movwf   FSR
                   00368
0088               00369 KEYL
0088 0200          00370             movf    INDF,W          ; low nibble
0089 0E0F          00371             andlw   0F
008A 0902          00372             call    ScanCode        ; encode hex nibble
008B 0035          00373             movwf   KEY
008C 0937          00374             call    SendKEY         ; emulate key press
008D 091A          00375             call    Wait10ms
008E 0CF0          00376             movlw   BREAK           ; emulate key release
008F 0937          00377             call    SendKEY
0090 0C01          00378             movlw   1
```

```
0091 091B           00379             call    WaitWx750       ; wait 750us
0092 0215           00380             movf    KEY,W
0093 0937           00381             call    SendKEY
                    00382
0094 091A           00383             call    Wait10ms
                    00384
0095 0380           00385             swapf   INDF,W          ; high nibble
0096 0E0F           00386             andlw   0F
0097 0902           00387             call    ScanCode        ; encode hex nibble
0098 0035           00388             movwf   KEY
0099 0937           00389             call    SendKEY         ; emulate key press
009A 091A           00390             call    Wait10ms
009B 0CF0           00391             movlw   BREAK           ; emulate key release
009C 0937           00392             call    SendKEY
009D 0C01           00393             movlw   1
009E 091B           00394             call    WaitWx750       ; wait 750us
009F 0215           00395             movf    KEY,W
00A0 0937           00396             call    SendKEY
                    00397
00A1 091A           00398             call    Wait10ms
                    00399
00A2 02A4           00400             incf    FSR,F          ; next byte
00A3 02F0           00401             decfsz  BYTECOUNT,F
00A4 0A88           00402             goto    KEYL
                    00403
                    00404 ;-------------------------------------------------------------------
                    00405 ; now wait for the HCS to stop transmisssion (button release)
                    00406 ;
00A5                00407 Release
00A5 07A6           00408             btfss   LED             ; wait Led rise
00A6 0AA5           00409             goto    Release
00A7 0A7A           00410             goto    Main
                    00411
                    00412             END

MPASM 01.40 Released          HOPPASW.ASM   2-20-1997  16:45:24         PAGE  2

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXX-------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X
0200 : XXX------------- ---------------- ---------------- ----------------
0FC0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   168
Program Memory Words Free:   343


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

NOTES:

NOTES:

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

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

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: 972-818-7423 Fax: 972-818-2924

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

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

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

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 Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
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

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, 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'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
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

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
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 Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

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

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

Italy
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

01/18/02