AN642

Code Hopping Decoder using a PIC16C56

Author: Steven Dawson
Microchip Technology Inc.

OVERVIEW

This application note fully describes the working of a code hopping decoder implemented on a Microchip PIC16C56 microcontroller. Background is given on the various KEELOQ® code hopping encoders that can be used with the decoder, the decoder hardware described, and descriptions of the various software modules comprising the system. The software can be used to implement a stand alone decoder or integrated with full function security systems. The decoder supports the Microchip HCS200, HCS201, HCS300, HCS301, HCS360, HCS361 and HCS410 KEELOQ code hopping encoders.

KEY FEATURES

- Stand alone decoder
- Compatible with Microchip HCS200, HCS201, HCS300, HCS301, HCS360, HCS361 and HCS410 encoders
- Automatic baud rate detection
- Automatic encoder type detection
- Four function outputs
- Six learnable transmitters
- RC Oscillator

Validation

Once a complete transmission has been received from an encoder, the transmission needs to be validated before any further action is taken. Validation consists of the following steps:

1. Check the serial number (24, 28 bits) against the stored encoder serial numbers (M_SERIAL).
2. Decrypt the transmission received (M_HOP).
3. Compare the discrimination value in the decrypted hopping portion of the transmission against the stored discrimination value (M_DIS).
4. Check if the synchronization counter falls within the resynchronization window (M_CHECK1).
5. Check if the synchronization counter falls within the open window. If not, then decoder resynchronization is necessary (M_CHECK2).
6. If resynchronization is necessary wait for a second transmission from the encoder with a consecutive synchronization counter.
7. Update the synchronization counter in EEPROM (M_UPDATE).
8. Set the appropriate outputs (M_BUT).
9. Return to MAIN routine and continue normal housekeeping chores.

Discrimination Values

After decryption, the Code Shift Register (CSR) used by the KEELOQ decryption algorithm contains the same 32 bits of information originally encrypted in the encoder before transmission. 12 of these bits are discrimination bits.

The decryption operation can be checked by comparing parts of the decrypted 32-bit word (the discrimination values) with known values.

For the HCS300/301 the user can program the discrimination bits to contain any value. In the HCS360/361 the discrimination bits are the least significant 8 bits of the serial number. The discrimination bits are stored in the external EEPROM during learn. By comparing the discrimination bits to the bits expected, the integrity of the decryption can be easily verified.

Note: The overflow bits (when available in an encoder) are treated as part of the discrimination value. For example, these bits can be set when the HCS300/301 encoders are programmed. When the encoder's counter overflows the overflow bits are individually cleared extending the counter range from 65,536 to 196,608. The clearing of the overflow bits will result in an 'erroneous' discrimination value being decrypted because the overflow bits are stored as part of the discrimination value. The transmission will be treated as an invalid transmission by this decoder and the transmission discarded. In order to avoid this, the HCS300/301's overflow bits should both be programmed as '0' and the synchronization counter started at '0'.

TABLE 9: HCS200/201, HCS300/301 DECRYPTED HOPPING CODE TRANSMISSION FORMAT

Function* (4 bits)	MSB Encoder disc. bits LS (12 bits including overflow bits)	MSB Synchronization counter LSB (16 bits)

* The HCS200/201 has padding in S3 button position since no S3 button is present. In addition the HCS200/201 does not have overflow bits present and these are also padded.

Synchronization Checking

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

The transmitting encoder has a 16-bit synchronization counter, stored in EEPROM, which is incremented every time the encoder is activated. The synchronization counter value is stored in the decoder's EEPROM every time a valid transmission is received from a particular encoder. When a following transmission is received from the same transmitter it is possible to quickly verify whether the transmission is valid. For example, a grabbed code from the legitimate user's previous transmission will result in a synchronization counter value, that has already been received, being decrypted.

Note:	Two copies of the synchronization counter are stored. The reason for this is should the power go down during an EEPROM write, a corrupted counter value would be read when the device is later powered up, resulting in encoder transmissions erroneously being discarded as invalid.

Provision must be made for the transmitter being pressed while out of range of the decoder. The Microchip decoder does this by allowing two 'synchronization windows'. The **open window** is a reception of a transmission where the synchronization counter is 1 to 16 higher than the previous counter value received. The reception of such a signal will result in an immediate counter update by the decoder and the appropriate outputs being activated.

If the transmitter is pressed more than 16 times out of range of the receiver, resynchronization needs to take place. The **resynchronization window** is a half of the total counter range, 32K big. During resynchronization the decoder waits for two consecutive transmissions from the encoder before resynchronization takes place and the resynchronized counters updated in the decoder's EEPROM. When the decoder receives a transmission with a synchronization counter value more than 16 above the stored counter value and less than 32,768 counts above the stored value, the decoder temporarily stores the value of the synchronization counter received. If the next transmission received has a sequential synchronization counter value the decoder resynchronizes on the last transmission received and activates the appropriate outputs.

If any of the above tests fail the transmission received is discarded. It is easy to change the size of the various windows in the source code. Modifications to the synchronization windows can be made in the M_CHECK routine.

FIGURE 8: DECODER WINDOW OPERATION

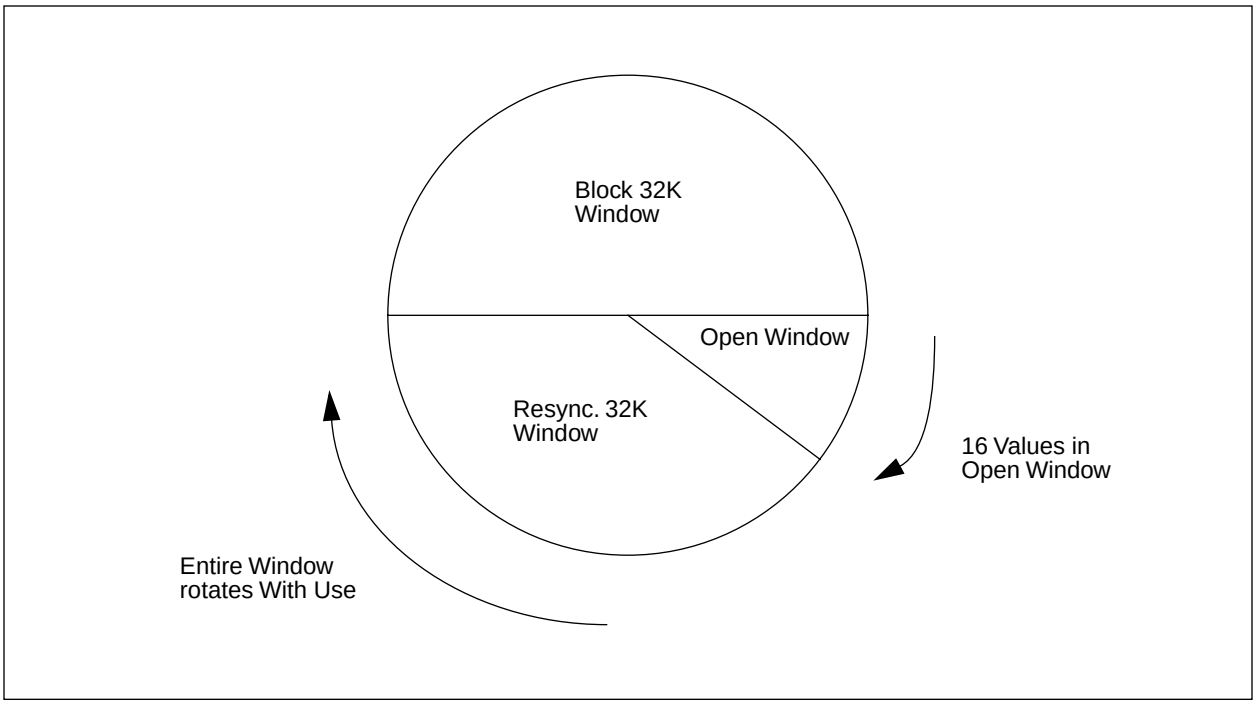
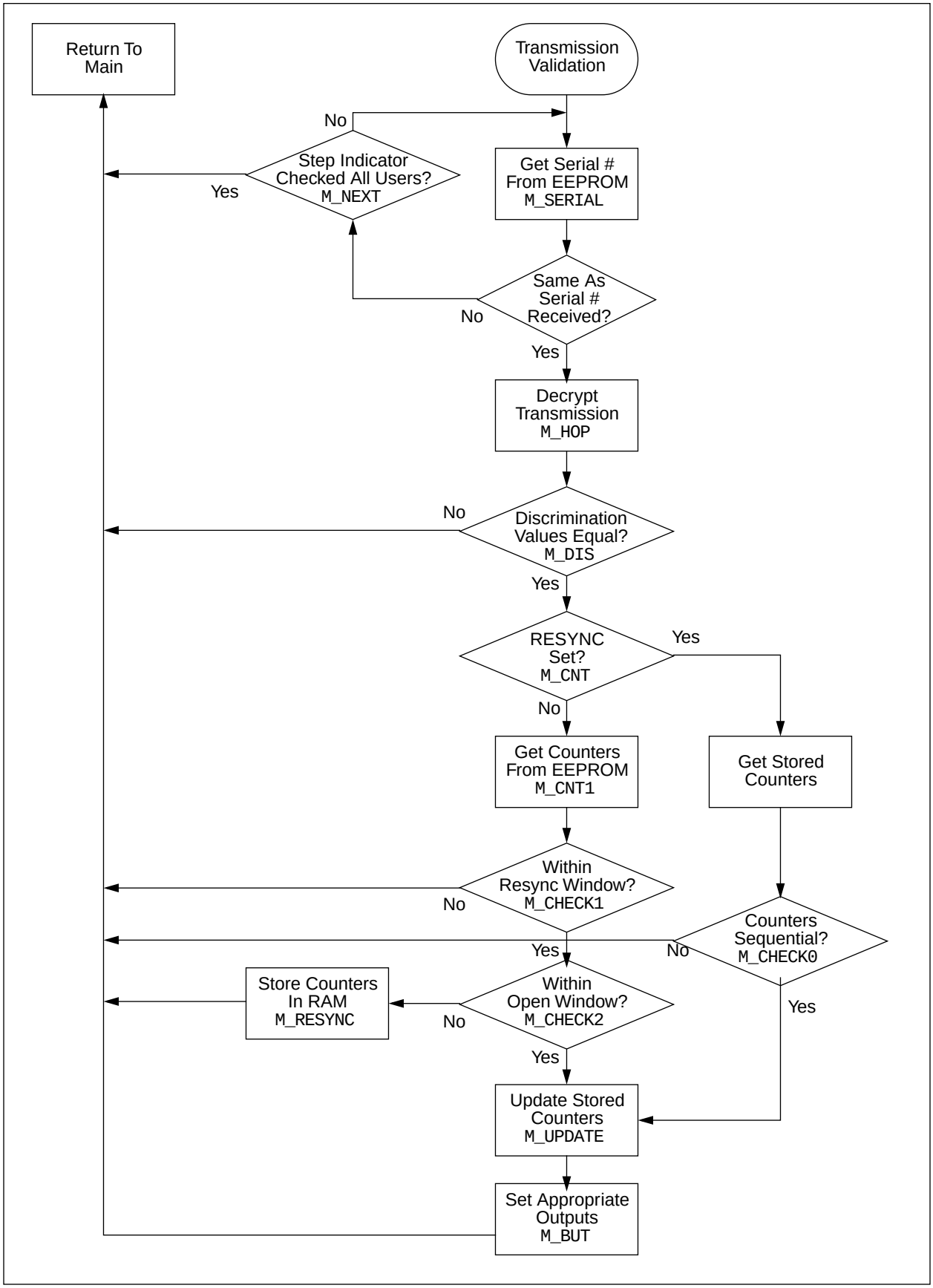


FIGURE 9: VALIDATION FLOW CHART TRANSMISSION VALIDATION FLOWCHART



Function Interpretation

In a single-chip system, where the code hopping decoder and the control program are combined into one device, the function code is interpreted to determine what the system must do. One function can be used to arm the system and lock the vehicle, a second to disarm the system and unlock the vehicle, and a third to open the trunk.

The four function bits in the encrypted portion of a transmission can be used to determine the button(s) pressed on the transmitter. Up to 15 functions can be implemented in this way, 0000 being related to a reset state.

The four function bits transmitted by the KEELOQ encoders are labeled F2, F1, F0 and F3. These correspond to S2, S1, S0, and S3 in the HCS300/301, S2, S1, S0, and S2 in the HCS200/201. The Sn bits in turn correspond with the values of the control inputs of the encoders.

In the Microchip decoder the function code received from the encoder is put onto the function outputs (S0 to S3) if a valid transmission is received (M_BUT). In addition, if the button code that was learned into the system is received, the FUNC OK output is activated.

Output Activation

The Microchip decoder has five momentary outputs namely S0, S1, S2, S3, and FUNC OK. As described in the section on Function Interpretation these outputs are a function of the inputs activated on the encoder. The momentary outputs are activated for 524 ms and extended for 524 ms if a repeated transmission is received. If a new valid transmission with a different function code is received during output activation, the outputs are switched off for 131 ms and the new function output activated.

FUNC OK is set if the function code received by the decoder is the same as the function code received during the learn cycle. The routine displaying the function code information and checking whether the function code received is the same as the one used during learn is called M_BUT.

Key Generation

Before transmitting the hopping portion of the code an encoder uses a 64-bit read protected encryption key to encrypt the information. In order to read the information contained in the hopping portion of the code it is necessary for the decoder to decrypt the data.

The use of a manufacturer's code in key generation allows a unique relationship between encoder serial number and encryption/decryption key pair. This enables each manufacturer to produce encoders that cannot be cloned by competitors. Security of the manufacturer's code is critical to product security and as a result the manufacturer's code (MKEY) is stored in ROM in the PIC16C56 microcontroller.

Two key generation techniques are used in the KEELOQ decoders. The first is a normal key generation algorithm which is used in the Microchip decoder described. The second key generation technique uses a feature on the Microchip HCS encoders called SEED transmission. In this mode the transmitter transmits a SEED programmed during learn instead of the code hopping.

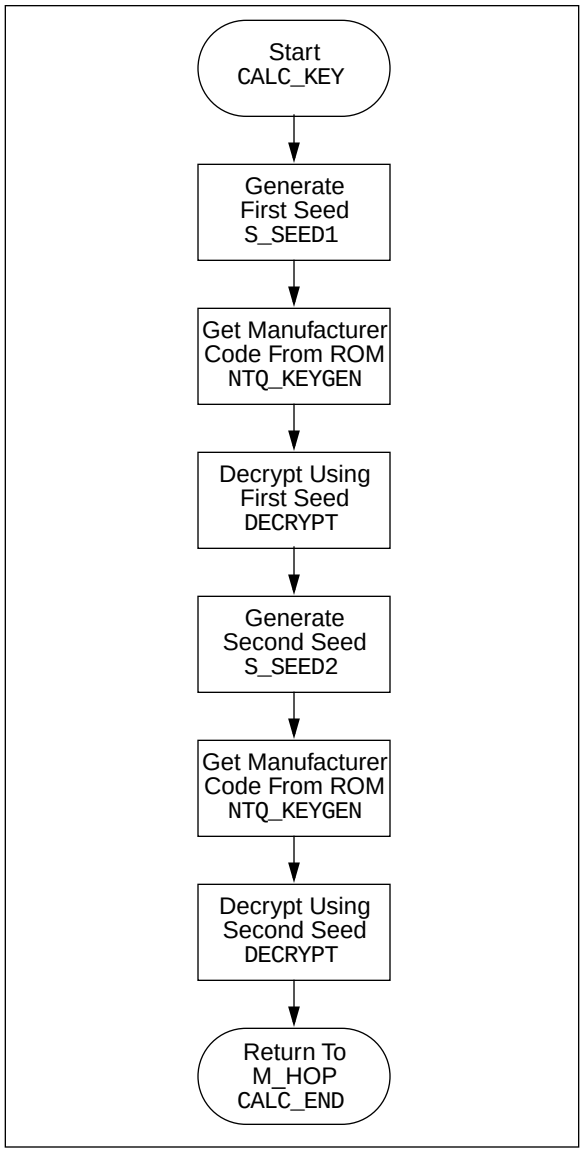
Normal Key Generation

The Microchip decoder described uses the normal key generation technique to generate decryption keys for the encoders. Normal key generation is performed in two steps as shown in the flow chart in Figure 10. The first step generates the least significant 32 bits of the decryption key, the second the most significant 32 bits of the decryption key. The relationship between decryption key and serial number during key generation is achieved by using the serial number of the encoder as a seed for the decryption routine. The length of the serial number is 28 bits for the HCS200/201 and the HCS300/301. The serial number of the encoder is padded to make the key generation seed 32 bits. To allow the most significant 32 bits of the decryption key to differ from the least significant 32 bits the serial number is padded to 32 bits differently on each step. The manufacturer's code is used as the key for the decryption operation during key operation. Table 10 shows the different padding techniques used.

TABLE 10: SEED GENERATION PADDING FOR VARIOUS ENCODERS

	HCS encoders
Seed1 padding	2*******16
Seed2 padding	6*******16

FIGURE 10: NORMAL KEY GENERATION FLOW CHART

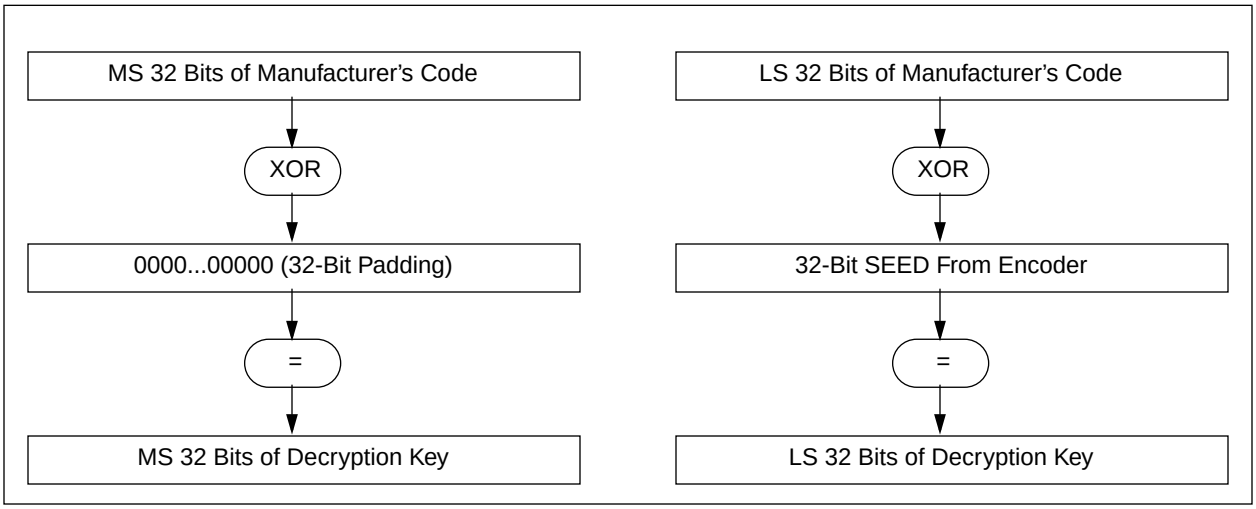


Secure Key Generation

Secure key generation relies on the encoder to supply the seed used to generate a key. The Microchip HCS encoders are able to transmit a fixed 32-bit value (SEED) in place of the code hopping during the learn process. The SEED is padded with zeros to make up 64 bits and then XOR'ed with the manufacturer's code. The value that results is the 64-bit decryption key that is used to decrypt the hopping portion of a transmission.

Since the random seed is only transmitted during the learning process, and is required to generate the key a normal hop transmission cannot be intercepted, a key generated and the hop code decrypted to predict the next hopping code. Figure 11 shows how the secure key generation technique described can be implemented.

FIGURE 11: SECURE KEY GENERATION



Decryption

After receiving a valid transmission the decoder decrypts the code hopping portion of the transmitted code. This is done with the help of the KEELOQ decryption algorithm. Each encoder has its own encryption key which is related to the serial number of the encoder and is calculated when the encoder is programmed. In order to decrypt the transmission a decryption key is calculated by the decoder during the learn cycle and stored in the EEPROM. The decryption routine is called DECRYPT.

The KEELOQ decryption algorithm is used to decrypt the 32-bit code hopping portion of KEELOQ transmissions. A 32-bit Code Shift Register (CSR) contains the received code, and a 64-bit register contains the decryption key

The 64-bit decryption key is retrieved from EEPROM into RAM for every decryption operation. Several 64-bit keys are stored in memory, one for each valid transmitter. The key particular to a given encoder is retrieved to decrypt the code hopping portion of a particular encoder. The key to be retrieved is identified by the serial number of the encoder that is transmitted.

The block diagram (Figure 12) explains the operation during each iteration of the decryption algorithm. A non-linear function (NLF) is used to produce a single bit from five bits in the CSR. This output is combined, via an exclusive-OR function, with two CSR bits and a single-bit from the key register to form an output. At the end of each cycle, the key register is rotated left, and the CSR is rotated left. The MSB (bit 3,7) of the CSR is discarded, and the output from the exclusive-OR function is inserted into the LSB (bit 0,0) of the CSR.

The decryption operation requires 528 iterations. In other words, the operation in the block diagram should be executed 528 times before the decrypted data will appear in the CSR.

The non-linear function (NLF, Table 11) is intended to obscure any linear relationships that might otherwise exist in the encrypted output. The NLF is listed in the form of a 5-bit lookup table, in which the five input bits are $I_4 = CSR_{3,6}$, $I_3 = CSR_{3,1}$, $I_2 = CSR_{2,3}$, $I_1 = CSR_{1,0}$, and $I_0 = CSR_{0,0}$.

FIGURE 12: THE KEELOQ DECRYPTION ALGORITHM

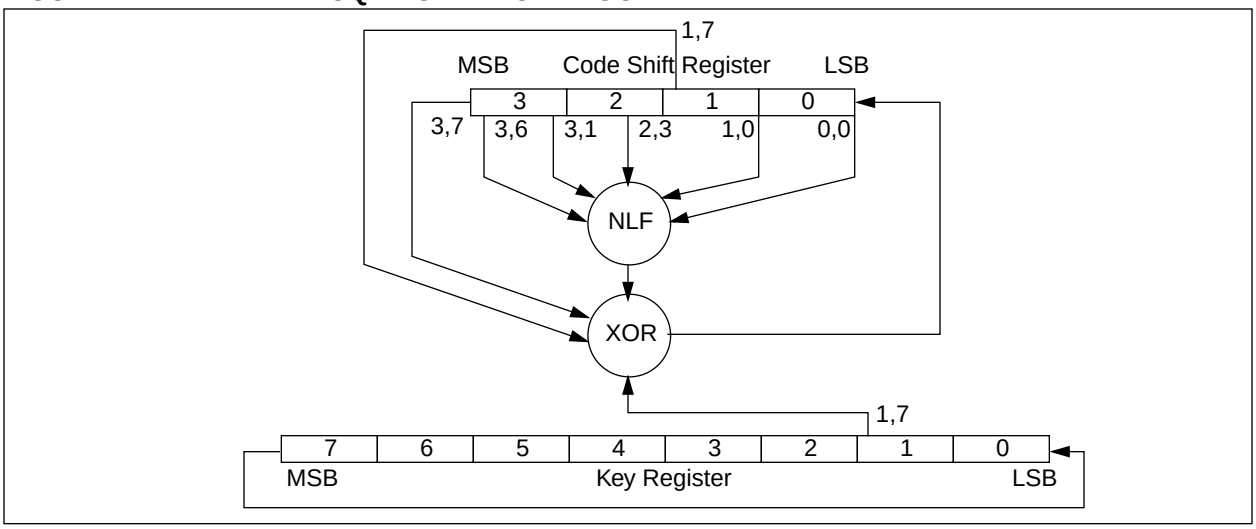


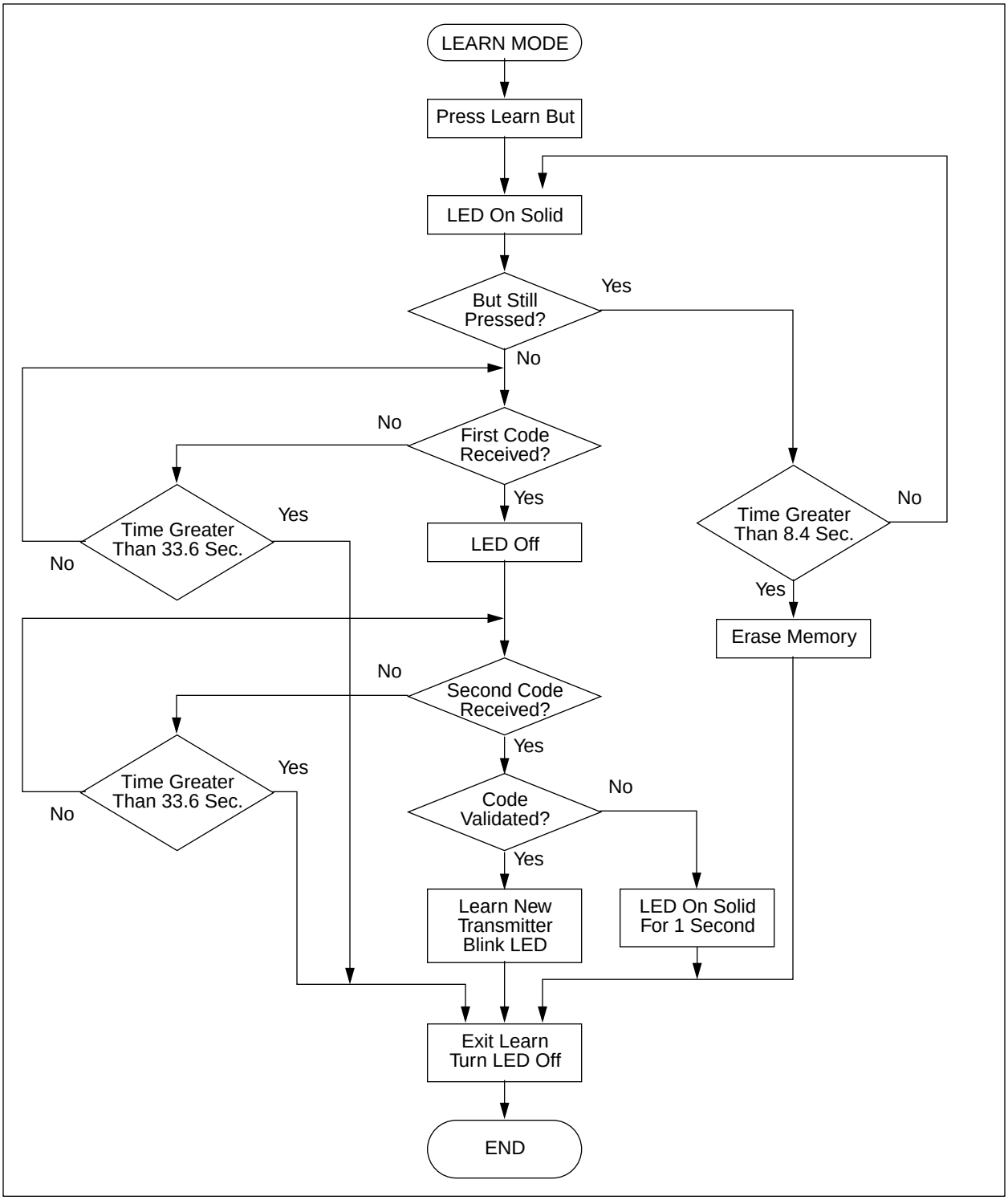
TABLE 11: NON-LINEAR FUNCTION OUTPUT

I4	I3	I2	I1	I0	NLF	I4	I3	I2	I1	I0	NLF
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	1
0	0	0	1	1	1	1	0	0	1	1	1
0	0	1	0	0	0	1	0	1	0	0	1
0	0	1	0	1	1	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1	0	0	1	1
0	1	0	1	0	1	1	1	0	1	0	0
0	1	0	1	1	0	1	1	0	1	1	1
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	0	1	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	0
0	1	1	1	1	0	1	1	1	1	1	0

Learn

A learn indicator is used by the Microchip decoder to keep track of the next position a learn is to take place. The encoder positions in EEPROM form a rotating buffer where the next transmitter to be written over is the transmitter at the tail of the buffer. The LEARN INIT input is active low and the LEARN IND output active high. Learn is initiated by momentarily pressing the LEARN button. The decoder uses the current learn position as a scratch pad area. This means that an unsuccessful learn deletes the information stored at that learn position. The learn indicator is not incremented if the learn was unsuccessful. The following flow diagram shows the learning operation.

FIGURE 13: LEARN OPERATION



Learn (cont'd)

The following checks are performed on the received codes to determine if the transmitter is valid:

1. The first code that is received is checked for bit integrity (RECEIVE).
2. The stored serial numbers are then searched to check if a transmitter being is re-learned. If a re-learn is taking place that position is used or else the position pointed to by the learn indicator is used (M_SERIAL).
3. The serial number is stored in EEPROM and used to generate a decryption key (CALC_KEY).
4. The hop code is decrypted (M_HOP) and the result stored temporarily (M_SL_UPDATE). The counter and serial number are stored (M_UPDATE).
5. The serial number of the second code that is received is compared to the first received serial number (M_SERIAL).
6. The second hop code is decrypted (M_HOP) and the discrimination values compared (M_DIS).
7. The synchronization counters of the decrypted codes are compared to check that they are sequential codes (M_UPDATE).
8. If all the checks pass the learn was successful and the learn indicator is incremented else the position is erased.

Operation of Learn

1. Press and release the LEARN button. Indicator LED will turn on to indicate learn mode.
2. Press transmitter button. The LED will turn off.
3. Press transmitter a second time. The LED will blink to indicate that the transmitter was learned successfully.
4. Repeat steps 1-3 to learn up to six transmitters. The seventh transmitter will overwrite the first transmitter that was learned.
5. Learn will be terminated if two non-sequential codes were received or if two acceptable codes were not decrypted within 33.6 seconds. An invalid learn will be indicated by the LED turning on solid for one second.
6. Erasing all the transmitters is accomplished by pressing and holding the LEARN button for 8.4 seconds. The LED will turn off at the end of the 8.4 seconds to indicate that all the transmitters were erased. The learn indicator is reset to the first position.

TIMER0 (RTCC) Multiplexing

A time keeping scheme is needed to ensure that the system timing is not abandoned while receiving an incoming signal, during learn cycles, key generation and decryption. The system timing is used to allow periodic monitoring of sensors and pulsing outputs with a specific period.

TIMER0 is used to keep track of system time. TIMER0 is an 8-bit timer on the PIC16C56. On the Microchip decoder described, TIMER0 is prescaled to increment every 256 instruction cycles. This makes TIMER0 very useful for keeping track of real time. While various routines are being run, including receive routines and decryption, TIMER0 is periodically checked for a time-out value calculated at the beginning of a certain period (i.e., switch off time of a LED).

The routine checking TIMER0 is called TST_RTCC. The most significant bit (MSB) of TIMER0 changes every 32 ms. In order to extend the range of TIMER0 2 additional 8-bit counters are used, CNT_LW and CNT_HI, which extends the range TIMER0 to 134 seconds. The MSB of TIMER0 is mirrored in the MSB of the STATUS register during startup. During TST_RTCC the 2 bits are compared. If the bits differ, the MSB of TIMER0 has changed indicating that 32 ms has passed. The MSB of STATUS is changed to match the MSB of TIMER0 and the extended counter(CNT_LW and CNT_HI) incremented.

The second portion of the TST_RTCC routine checks appropriate time-out values based on the system status bits in SREG (i.e. To check for the 30s time-out in the learn routine TST_RTCC checks to see if bit three of CNT_HI is set).

ROM MEMORY MAP (8-BIT BYTES)

TABLE 12: ROM MEMORY MAP (8-BIT BYTES)

Word Address	Mnemonic	Description
40	MKEY_0	64-bit Manufacturer's Code (Used to generate decryption keys)
41	MKEY_1	
42	MKEY_2	
43	MKEY_3	
44	MKEY_4	
45	MKEY_5	
46	MKEY_6	
47	MKEY_7	
48	Unused	
49	Unused	
4A	EKEY_0	64-bit EEPROM Key (Used to encrypt EEPROM data)
4B	EKEY_1	
4C	EKEY_2	
4D	EKEY_3	
4E	EKEY_4	
4F	EKEY_5	
50	EKEY_6	
51	EKEY_7	

EEPROM MEMORY MAP (16-BIT WORDS)

TABLE 13: EEPROM MEMORY MAP (16-BIT WORDS)

Address	Mnemonic	Address	Mnemonic
00	USER0	20	CNT20
01	Learn Ind.	21	CNT21
02	DIS0	22	SER20
03	DIS1	23	SER21
04	USER2	24	KEY20
05	USER3	25	KEY21
06	USER4	26	KEY22
07	USER5	27	KEY23
08	DIS2	28	CNT30
09	DIS3	29	CNT31
0A	DIS4	2A	SER30
0B	DIS5	2B	SER31
0C	USER6	2C	KEY30
0D	USER7	2D	KEY31
0E	USER8	2E	KEY32
0F	USER9	2F	KEY33
10	CNT00	30	CNT40
11	CNT01	31	CNT41
12	SER00	32	SER40
13	SER01	33	SER41
14	KEY00	34	KEY40
15	KEY01	35	KEY41
16	KEY02	36	KEY42
17	KEY03	37	KEY43
18	CNT10	38	CNT50
19	CNT11	39	CNT51
1A	SER10	3A	SER50
1B	SER11	3B	SER51
1C	KEY10	3C	KEY50
1D	KEY11	3D	KEY51
1E	KEY12	3E	KEY52
1F	KEY13	3F	KEY53

USER	These words are reserved for user storage.
SER	The encoder serial number storage
KEY	These words contain the decryption key for each encoder.
DIS	Discrimination values and function code storage.
CNT	Two copies of the synchronization counter are stored for each encoder to prevent loss of synchronization information due to EEPROM write failure.

RAM MEMORY MAP (8 BIT BYTES)

TABLE 14: RAM MEMORY MAP (8 BIT BYTES)

Address	Mnemonic	Description
07	FLAGS	Decoder flags
08	ADDRESS	Address register - points to address in EEPROM
09	TXNUM	Current transmitter
0A	OUTBYT	General data register, mask register used in decryption
0B	CNT0	Loop counters
0C	CNT1	
OD	CNT2	
OE	CNT_HI	16-bit clock counter
OF	CNT_LO	
10	TMP1	Temporary registers
11	TMP2	
12	TMP3	
13	TMP4	
14	CSR4	64-bit shift register Used in reception, decryption and key generation
15	CSR5	
16	CSR6	
17	CSR7	
18	CSR0	
19	CSR1	
1A	CSR2	
1B	CSR3	
1C	OLD_BUT	Store previous button code
1D	RAM_HI	16-bit RAM counter (used in resynchronization)
1E	RAM_LW	
1F	SREG	Program state register