Freescale Semiconductor

Order this document
as AN463/D

APPLICATION NOTE

AN463

68HC05K0 Infra-red Remote Control

Tony Breslin,
MCU Applications Group,
Freescale Ltd**., East Kilbride, Scotland**

The MC68HC05K0 is a low cost, low pin count single chip microcomputer with 504 bytes of user ROM and 32 bytes of RAM. The MC68HC05K0 is a member of the 68HC05K series of devices which are available in 16-pin DIL or SOIC packages. It uses the same CPU as the other devices in the 68HC05 family and has the same instructions and registers. Additionally, the device has a 15-stage multi-function timer and 10 general purpose bi-directional I/0 lines. A mask option is available for software programmable pull-downs on all of the I/O pins and four of the pins are capable of generating interrupts.

The device is ideally suited for remote-control keyboard applications because the pull-downs and the interrupt drivers on the port pins allow keyboards to be built without any external components except the keys themselves. There is no need for external pull-up or pull-down resistors, or diodes for wired-OR interrupts, as these features are already designed into the device.

This application makes use of many of the device features to control an infra-red television remote control. The application could be very easily modified to control any device with a similar transmission protocol. It will run on any of the 'K' devices without modification.

Remote Control Specifications

The basic purpose of a television remote control is to transmit a control instruction to the television. The instruction is generated by a keystroke on the remote control keyboard. The detection and decoding of a key press and the transmission encoding is carried out by the remote control micro controller.

When a key on the remote control keypad is pressed, the micro controller must first determine what key is being pressed and generate an individual code for the key. The key code is then converted to a instruction code that is inserted into the transmission command which, using a defined protocol, is transmitted to the television receiver. The command is continually transmitted as long as the key is being held down.

As the remote control is battery powered it needs to use as little power as possible. This is achieved by entering STOP mode when no keys are being pressed and effectively switches off the device. The micro controller comes out of STOP mode upon receipt of an interrupt request that is generated when a key is pressed.

Remote Control Keyboard

The 68HC05K0 has ten general purpose I/O pins. One of these is used for the transmission signal output leaving nine pins for the keyboard control. Of these, four pins on PortA have internal interrupt request hardware. Using these four pins as inputs allows key presses to be detected without any external interrupt hardware. This leaves the five remaining pins for outputs.

Using the internal pull-down facility and the rising edge interrupt request on the four inputs permits interrupts to be generated. **If the five outputs are set to logic '1', so driving an input from logic '0' to logic '1' when a key is pressed, an interrupt request can be generated**. Using this arrangement a five by four keyboard matrix can be used. An extra four keys can be controlled if the Vdd line is used to drive one row of four keys to logic '1'. Therefore the maximum amount of keys

1	2	3	NORM
4	5	6	MUTE
7	8	9	VOL+
0	PC+	PC-	VOL-
TV/ TEXT	MIX	TIME	CON+
STOP	SUB- PAGE	INDEX	CON-

VDD	31 11	32 12	34 13	38 00
A7	71 14	72 15	74 16	78 01
A6	b1 17	b2 18	b4 19	b8 06
A5	d1 10	d2 2c	d4 2d	d8 07
A4	e1 39	e2 3b	e4 3a	e8 0c
B0	f1 3e	f2 3d	f4 3c	f8 0d
	A0	A1	A2	A3

Figure 1 Keyboard layout with associated scanned and transmitted codes

A depressed key will set one of the input columns to logic ‘1’. By scanning the columns, and setting each row output to logic ‘0’ and then checking if the inputs all become logic ‘0’, the associated row for the key can be determined. If rotating the logic ‘0’ through the five output pins fails to identify a key column, then the key must be connected to the Vdd line. This process gives an individual code for each key which is a combination of the code from the column inputs and the row outputs. This can then be decoded to an instruction that is inserted into the output signal for transmission.

Figure 1 shows the layout of the keyboard on the left and the scanned and transmitted codes on the right. The keyboard layout incorporates the various television controls plus controls for TELETEXT. On the left hand side the codes returned from scanning the keyboard are shown in the upper right-hand corner of each key and the code sent for transmission for that key instruction are shown in the bottom left-hand corner. The I/O pins for each row and column are also shown for each key.

Transmission Protocol

The transmission protocol in this application is that used by the MC144105 IR Remote Control Transmitter. It uses a binary coded 9-bit data word with the LSB being transmitted first. Each bit of the transmitted signal is in the form of a bi-phase pulse code modulated (PCM) signal, whose bit coding is shown in Figure 2. For a transmitted ‘0’ there is a 512µs pause followed by a 32kHz pulse train for 512µs. For a transmitted ‘1’ there is 32kHz pulse train followed by a 512µs pause. This gives a bit time of 1024µs for all bits.This is shown asFigure 2.

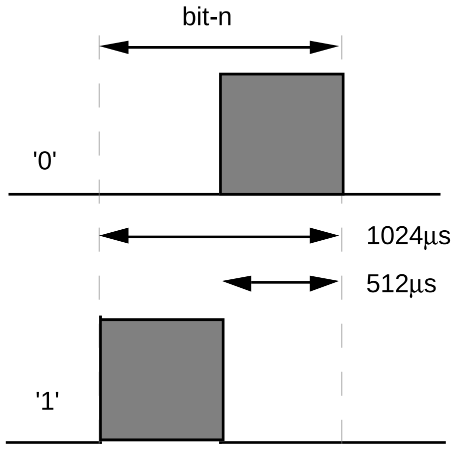


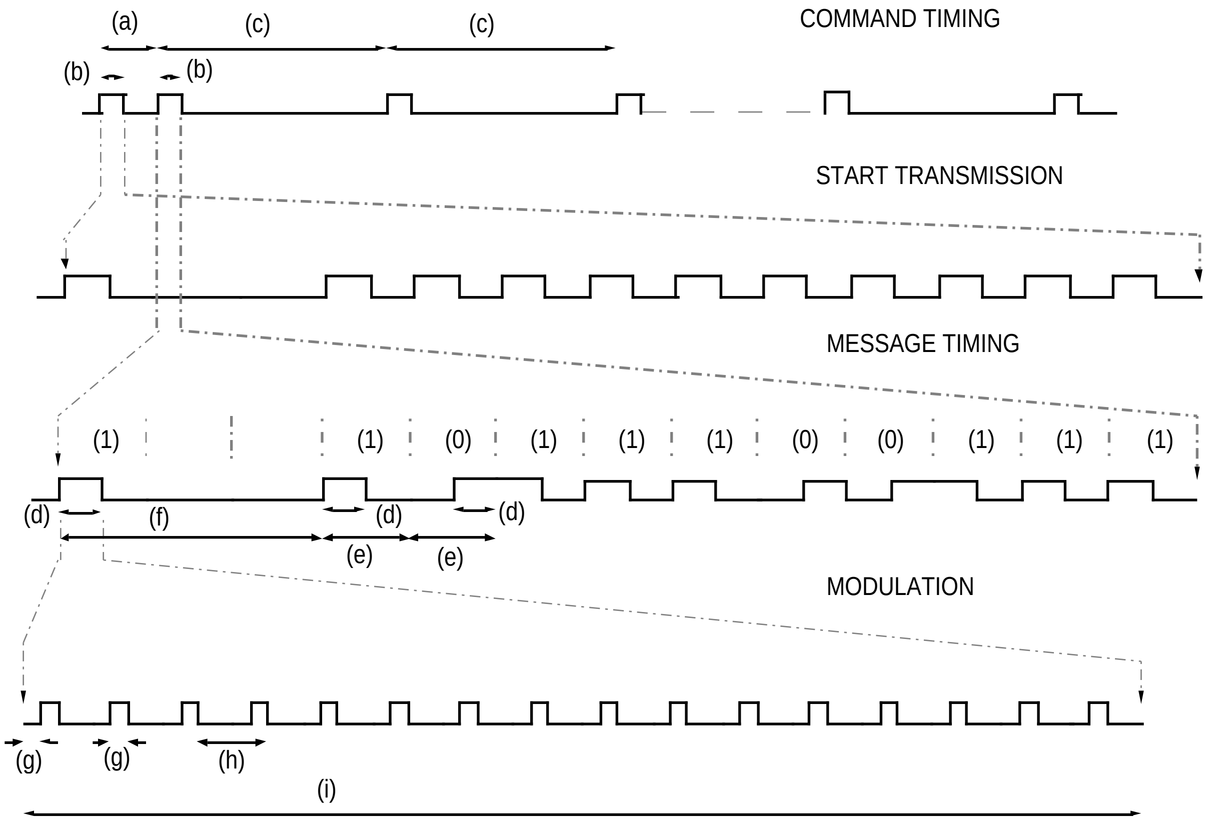
Figure 2 Bit coding of PCM signal

A complete transmission command consists of several messages. Each command begins with a start message of nine 1's followed by the message appropriate to the key pressed. This message is repeatedly transmitted until the key is released. The transmission is terminated after the key is released by a end message of nine 1's.

Every message consists of a pre-bit, a pre-bit pause, a start bit and nine data bits. The pre-bit and the start-bit are always logical '1'. The pre-bit allows for the set up of the automatic gain control in the receiving preamplifier. Figure 3 gives the exact timing relationships for the transmissions.

The command timing in Figure 3 shows that after the start transmission the message is continually re-transmitted at intervals of 131ms (approximately 8Hz) until the key is released. This is shown as time (c). The control timing shows the nine bit instruction 111001110 being transmitted starting with the LSB. The pre-bit pause is equal to two bit periods and is followed by a start-bit of logical '1'. The pulse train is continuous during the transition between transmitting a logic '0' and a logic '1'. The modulating pulse train has a frequency of approximately 32kHz with a mark-to-space ratio of one to three.

The signal for transmission is output through one port pin and is used to drive an IR diode amplifier circuit.



Command timing	(a) = 32.8ms (b) = 13.3ms (c) = 131ms	start transmission start command control transmission	13 * bit time
Message timing	(d) = 512µs (e) = 1.024ms (f) = 3.072ms	(half-bit time) (bit time) (pre-pulse time)	$16/f_{carrier}$
Modulation	(g) = 8µs (h) = 32µs (i) = 512µs	($1/f_{carrier}$) (half-bit time)	$16/f_{carrier}$

Figure 3 Circuit timing

Remote Control Operation

Figure 4 is a flow diagram showing the operation of the remote control on power-up or reset. After the initial set-up of the ports as inputs or outputs the remote control goes into STOP mode. It will remain in STOP mode as long as the device is not reset or a key is not pressed. When a key is pressed an interrupt request is generated. A short time delay makes sure that it is a true key press and not noise and also allows time for any switching effects on the inputs to pass prior to checking the inputs.

The keyboard is then read to find which key has been pressed and the code for the key is decoded into an instruction and transmitted to the television. If the key is held down the instruction is re-transmitted until the key is released. This is useful for the instructions which count through the television channels or adjust the volume, colour or brightness controls.

When the key is released a terminating instruction is sent to the receiver to inform it that the next message received is a separate instruction. This is useful in the case of a one time instruction like sending a channel number. In this example the receiver will tune to a channel only once; to tune to another channel the key must be released and a new instruction sequence received.

After terminating the transmission the ports are reset ready for the next key press and the processor returns to the STOP mode.

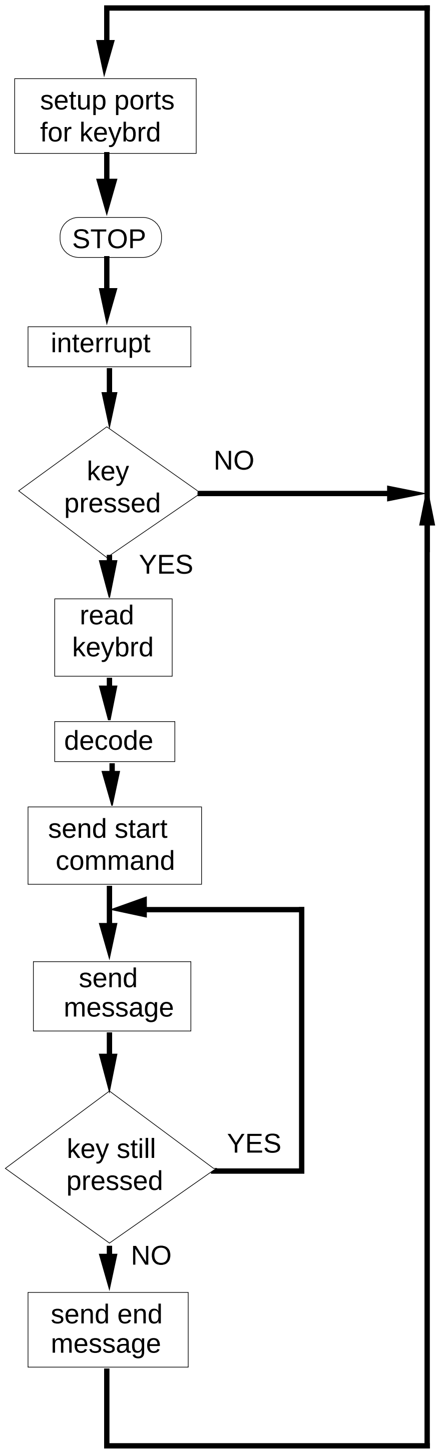


Figure 4 Flow diagram

Hardware

The remote control circuit is shown in Figure 5. The hardware consists of the keyboard, the oscillator and the infra-red amplifier. The oscillator can be a crystal or a ceramic resonator with a frequency of 2MHz. The oscillator frequency is important since the transmission timing is based around a 1MHz internal clock frequency.

The infra-red amplifier uses two transistors and two standard diodes to limit the current through the IR diodes to approximately 1A. There is a need for a large capacitor close to the IR diodes because of the high switching current of the circuit.

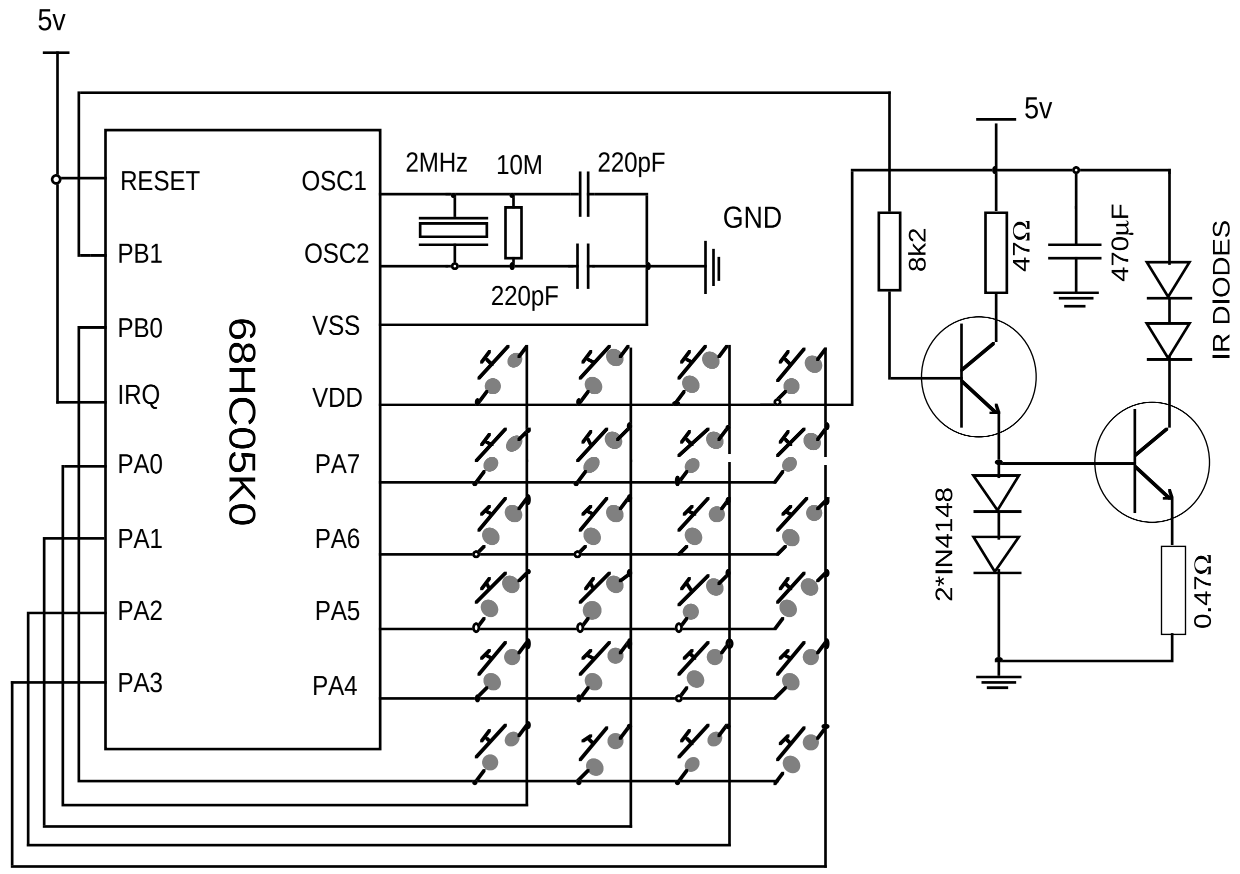


Figure 5 Infra-red remote control

Software

The listing of the remote control assembler code is contained at the end of this application note. The first section of the listing sets up the ports prior to going into STOP mode and waiting for a key to be pressed. PortA bits 0-3 are set up as inputs with the pull-downs enabled. Bits 4-7 are set up as outputs logic '1' as is PortB bit 0. PortB bit 1 is set-up as output logic '0' to switch off the IR amplifier before going into STOP mode.

The next section of code named 'presd' is the routine pointed to by the interrupt vector and is entered when a key is pressed. This routine first calls the keyboard scanning routine to determine which key has been pressed. It then calls the decoding routine to convert the code from the keyboard to a code that will be accepted by the television. The start message is then transmitted and is followed by the instruction message. There is then a check to see if the same key is still being pressed. If it is then the instruction message is re-transmitted until the key is released and the end message is transmitted.

As the transmission protocol requires nine data bits and only one byte instructions are being decoded a flag has to be set for the ninth bit of the transmission routine. For the start and end transmissions this flag is set to 1 to give the nine 1's message. For all instructions the ninth bit is 0 so the flag is cleared.

The decoding routine compares the code from the keyboard scan routine with data array 'keydat'. On a match it takes the corresponding element from the array 'tvdat' as the instruction code for transmission.

The values of the instruction codes shown in the right-hand side of Figure 1 are specific for the receiver application. Each receiver using the same communications protocol will receive the same nine bit instruction but what the instruction does is dependent upon the receiver software. In this example the eight bit instruction '14' changes the channel to number four. In another receiver application the receiver may interpret the instruction code '14' as increase volume.

The transmission routine is entered with the instruction for transmission in 'keyst3'. After the pre-bit and the start-bit are transmitted the instruction byte is rotated (LSB first) into the carry flag. A logic '1' is sent for transmission if the flag is set after rotation and a logic '0' is sent for transmission if the flag is cleared. Each bit is transmitted as shown in Figure 1. The routines 'send0' and 'send1' send a pause of 512µs followed by a 32kHz pulse train for 512µs and a 32kHz pulse train for 512µs followed by a 512µs pause respectively. In the situation when a '1' follows a '0' then a pulse train of 1024µs is required. To avoid breaks in this pulse train the 'send0' routine checks the next bit to be transmitted to see if a double length pulse train must be transmitted. The 'send1' routine then has to check that a double length pulse train has not been sent in the previous one and a half bit periods before sending a pulse train.

The routine 'burst' produces the 32kHz pulse train for a duration set by a count in the accumulator. As the instruction time for setting the PortB bit 1 pin high or low is five clock cycles then the minimum processor clock period is derived by dividing the minimum output state time, which is 8µs when the output is high, by the minimum number of clock cycles to change this state. This gives an internal clock period of 8µs/5 equalling 1.6µs. Adding a three cycle delay will require an internal clock period of 8µs/8 = 1µs, allowing a 2MHz oscillator to be used.

The code size is approximately 300 bytes, leaving memory space for more features to be added to the controller.

Debug

On applying power to the circuit the RESET vector will initialise the program counter at the beginning of the software. When examining the output at PortB bit 1 with an oscilloscope or logic analyser it should be noted that when trying to capture the signal by pressing a key the first signal out will be the start message of nine 1's. To capture the instruction the key should be held down and as the instruction will be continually re-transmitted then the capture can be initiated at this point.

Listing

```
                        ***************************************************************
                        * INFRA RED REMOTE CONTROL FOR K0,K1                          *
                        ***************************************************************
                        * WRITTEN BY A.BRESLIN    13.1.92                             *
                        ***************************************************************
                        * THIS PROGRAM READS AND ENCODES A KEY FROM A 24 KEY KEYBOARD *
                        * TO A FORM OF BIPHASE PULSE CODE MODULATION (PCM) FOR INFRA  *
                        * RED TRANSMISSION. IT USES THE TRANSMISSION PROTOCOL OF THE  *
                        * MC144105 IR REMOTE CONTROL TRANSMITTER                      *
                        ***************************************************************


0000                    porta   equ     00
0001                    portb   equ     01
0004                    ddra    equ     04
0005                    ddrb    equ     05
0008                    tcsr    equ     $08
0010                    papd    equ     $10

00e0                            org     $e0

00e0                    keyst1  rmb     1               ; initial code from keyboard
00e1                    keyst2  rmb     1               ; keycode
00e2                    keyst3  rmb     1               ; code transmitted
00e3                    dflag   rmb     1               ; flag for last and 9th bits


                        **************************************************************
                        * THE PORTS ARE SET UP USING PORTA 0-3 AS INPUTS MAKING USE  *
                        * OF THE INTERNAL INTERUPT GENERATION ON THESE I/0 LINES.    *
                        * STOP MODE IS ENTERED UNTIL A KEY IS PRESSED                *
                        **************************************************************

0200                            org     $200

0200 9a                 start   cli
0201 ad 04              wpres   bsr     setup
0203 9c                         rsp
0204 8e                         stop
0205 20 fa                      bra     wpres

0207 a6 f0              setup   lda     #$f0            ; porta 0-3 inputs
0209 b7 04                      sta     ddra            ; 4-7 as outputs
020b b7 00                      sta     porta           ; set outputs high
020d b7 10                      sta     papd            ; 0-3 pulldown
020f a6 03                      lda     #$03            ; portb 0-1 outputs
0211 b7 05                      sta     ddrb
0213 a6 01                      lda     #$01            ; set portb 0 high
0215 b7 01                      sta     portb
0217 81                         rts


```

```
                        **************************************************************
                        * THE KEY READ IS DECODED FOR TRANSMISSION.                  *
                        * THE TRANSMISSION PROTOCOL REQUIRES A START MESSAGE OF 9    *
                        * ONES FOLLOWED BY THE KEYPRESSED CODE. THIS CODE IS         *
                        * CONTINUALLY RETRANSMITTED IF THE KEY IS HELD DOWN. AN END  *
                        * CODE OF 9 ONES TERMINATES THE TRANSMISSION AND THE DEVICE  *
                        * RETURNS TO STOP MODE.                                      *
                        **************************************************************

0218 ad 34              presd   bsr     keyscn          ; get key pressed
021a b6 e1                      lda     keyst2          ; save key to check
021c b7 e0                      sta     keyst1          ; if key held down
021e ad 67                      bsr     decode          ; decode key pressed
0220 12 e3                      bset    1,dflag         ; set nineth bit to 1
0222 a6 ff                      lda     #$ff            ; send start data
0224 b7 e2                      sta     keyst3          ; to transmission routine
0226 ad 71                      bsr     trnmit          ; nine one's
0228 b6 e1              sndagn  lda     keyst2          ; send key press message
022a b7 e2                      sta     keyst3          ; byte
022c 13 e3                      bclr    1,dflag         ; set nineth bit to 0
022e ad 69                      bsr     trnmit
0230 b6 00                      lda     porta           ; check if key still pressed
0232 a4 0f                      and     #$0f            ; end if no key pressed
0234 26 0f                      bne     endtrn
0236 ad 16                      bsr     keyscn          ; else check if same
0238 b6 e0                      lda     keyst1          ; key pressed
023a b1 e1                      cmp     keyst2
023c 26 07                      bne     endtrn          ; end if not
023e ae c8                      ldx     #$c8            ; delay
0240 5a                 tloop   decx                    ; before next
0241 26 fd                      bne     tloop           ; transmission
0243 20 e3                      bra     sndagn
0245 12 e3              endtrn  bset    1,dflag         ; send end message
0247 a6 ff                      lda     #$ff            ; of nine ones
0249 b7 e2                      sta     keyst3
024b ad 4c                      bsr     trnmit
024d 80                         rti                     ; re-enter stop mode

                        **************************************************************
                        * WHEN A KEY IS PRESSED THE DEVICE COMES OUT OF STOP MODE    *
                        * THE KEYBOARD IS SCANNED TO SEE WHICH KEY IS PRESSED        *
                        **************************************************************

024e cd 02 fc           keyscn  jsr     datwt           ; wait for debounce
0251 b6 00                      lda     porta           ; check if key press
0253 b7 e0                      sta     keyst1          ; store inputs
0255 a4 0f                      and     #$0f            ; mask outputs
0257 27 a7                      beq     start           ; stop if no key pressed
0259 ae ef                      ldx     #$ef            ; set one row low
025b 9f                 nxtrow  txa                     ; read ouput lines
025c b4 e0                      and     keyst1          ; combine with inputs
025e b7 e1                      sta     keyst2          ; store key code
0260 bf 00                      stx     porta           ; to find row which clears inputs
0262 b6 00                      lda     porta           ; check for inputs cleared
0264 a4 0f                      and     #$0f            ; mask outputs
0266 27 1c                      beq     gotit           ; zero in key-press row clears inputs
0268 58                         lslx                    ; check if last row
0269 5c                         incx                    ; set lsb to 1
026a 24 02                      bcc     tryb            ; try portb output if not porta
026c 20 ed                      bra     nxtrow          ; try next porta output row

026e b6 e0              tryb    lda     keyst1
0270 b7 e1                      sta     keyst2
0272 ae f0                      ldx     #$f0
0274 bf 00                      stx     porta           ; set all porta outputs high
0276 11 01                      bclr    0,portb         ; set portb 0 output low
0278 b6 00                      lda     porta           ; check for inputs cleared
027a a4 0f                      and     #$0f            ; mask outputs
027c 27 06                      beq     gotit           ; zero in key-press row clears inputs
027e b6 e1                      lda     keyst2          ;
0280 a4 3f                      and     #$3f            ; set individual code since last row
0282 b7 e1                      sta     keyst2          ; store code
0284 10 01              gotit   bset    0,portb         ; set portb column high again
0286 81                         rts
```

```
                        **************************************************************
                        * THE DECODE ROUTINE USES TWO ARRAYS. IT COMPARES THE KEY    *
                        * VALUE WITH THE ARRAY KEYDAT AND WHEN A MATCH IS FOUND THE  *
                        * CORRESPONDING ELEMENT IN THE ARRAY TVDAT BECOMES THE       *
                        * TRANSMITTED CODE.                                          *
                        **************************************************************

0287 ae 18              decode  ldx     #$18            ; data array offset to zero
0289 d6 03 02           nxtel   lda     keydat,x        ; look at each element of array
028c b1 e1                      cmp     keyst2          ; compare with key read
028e 27 03                      beq     match           ; decode if match
0290 5a                         decx                    ; else try next element
0291 26 f6                      bne     nxtel           ; norm if no match found
0293 d6 03 1a           match   lda     tvdat,x         ; get key code
0296 b7 e1                      sta     keyst2          ; store code to transmit
0298 81                         rts

                        **************************************************************
                        * THE TRANSMISSION PROTOCOL REQUIRES A PRE-BIT, A PRE-BIT    *
                        * PAUSE, A START BIT AND NINE DATA BITS, WHERE THE PRE-BIT   *
                        * AND THE START BIT ARE LOGIC '1'.                          *
                        **************************************************************

0299 10 e3              trnmit  bset    0,dflag         ; initialise for first bit
029b ad 32                      bsr     send1           ; send pre-bit
029d cd 02 fc                   jsr     datwt           ; pre-bit pause
02a0 cd 02 fc                   jsr     datwt           ; equalling four half data periods
02a3 cd 02 fc                   jsr     datwt           ;
02a6 cd 02 fc                   jsr     datwt           ;
02a9 ad 24                      bsr     send1           ; send start bit
02ab ae 08                      ldx     #$08            ; transmit 8 data bits
02ad 34 e2              nxtbit  lsr     keyst3          ; get next bit
02af 25 04                      bcs     data1           ; send 1 if carry set
02b1 ad 28                      bsr     send0           ; send 0 if carry clear
02b3 20 02                      bra     bitsnt
02b5 ad 18              data1   bsr     send1
02b7 5a                 bitsnt  decx                    ; countdown bits sent
02b8 26 f3                      bne     nxtbit          ; send next bit if count not zero
02ba 03 e3 04                   brclr   1,dflag,send00  ; if flag set
02bd ad 10                      bsr     send1           ; send 1 as nineth bit
02bf 20 02                      bra     endend          ;
02c1 ad 18              send00  bsr     send0           ; else send 0
02c3 ae 18              endend  ldx     #$18
02c5 ad 35              loopw   bsr     datwt           ; delay between successive
02c7 ad 33                      bsr     datwt           ; transmissions
02c9 ad 31                      bsr     datwt
02cb 5a                         decx
02cc 26 f7                      bne     loopw
02ce 81                         rts


                        **************************************************************
                        * TO TRANSMIT A LOGIC '1' A 32kHz PULSE TRAIN FOR 512us IS   *
                        * FOLLOWED BY A 512us PAUSE.                                 *
                        **************************************************************

02cf 01 e3 04           send1   brclr   0,dflag,last0   ; check if last bit was zero
02d2 a6 10                      lda     #$10            ; burst if last bit was 1
02d4 ad 15                      bsr     burst           ; 32kHz pulse for 512us
02d6 ad 24              last0   bsr     datwt           ; wait 512us
02d8 10 e3                      bset    0,dflag         ; set flag as 1 sent
02da 81                         rts

```

```
                        *************************************************************
                        * TO TRANSMIT A LOGIC '0' A 512us PAUSE IS FOLLOWED BY A    *
                        * 32kHz PULSE TRAIN FOR 512us. IF A LOGIC '1' FOLLOWS A '0' *
                        * THE 32kHz IS CONTINUED FOR 1024us TO AVOID A PROCESSING   *
                        * DELAY                                                     *
                        *************************************************************

02db ad 1f              send0   bsr     datwt           ; wait 512us
02dd 00 e2 04                   brset   0,keyst3,next1  ; check if next bit is 1
02e0 a6 10                      lda     #$10            ; single burst if 1
02e2 20 02                      bra     datset          ; data set
02e4 a6 20              next1   lda     #$20            ; double burst required
02e6 ad 03              datset  bsr     burst           ; 32kHz pulse for 512us
02e8 11 e3                      bclr    0,dflag         ; clear flag as 0 sent
02ea 81                         rts

                        *************************************************************
                        * THE 32kHz PULSE TRAIN HAS A MARK TO SPACE RATIO OF 1 TO 3 *
                        *************************************************************

02eb 13 01              burst   bclr    1,portb         ; portb 1 low
02ed 21 fe                      brn     *
02ef 12 01                      bset    1,portb         ; portb 1 high
02f1 21 fe                      brn     *
02f3 13 01                      bclr    1,portb         ; portb 1 low
02f5 9d                         nop
02f6 4a                         deca                    ; decrement count
02f7 27 02                      beq     endbur          ; end of burst ?
02f9 20 f0                      bra     burst
02fb 81                 endbur  rts


02fc a6 52              datwt   lda     #$52            ; count
02fe 4a                 loop    deca                    ; to provide 512us delay
02ff 26 fd                      bne     loop            ; after instruction times
0301 81                         rts

0302 31 f1 e1 d1 b1 71  keydat  fcb     $31,$f1,$e1,$d1,$b1,$71
0308 32 f2 e2 d2 b2 72          fcb     $32,$f2,$e2,$d2,$b2,$72
030e 34 f4 e4 d4 b4 74          fcb     $34,$f4,$e4,$d4,$b4,$74
0314 38 f8 e8 d8 b8 78          fcb     $38,$f8,$e8,$d8,$b8,$78

031a 11 3e 39 10 17 14  tvdat   fcb     $11,$3e,$39,$10,$17,$14
0320 12 3d 3b 2c 18 15          fcb     $12,$3d,$3b,$2c,$18,$15
0326 13 3c 3a 2d 19 16          fcb     $13,$3c,$3a,$2d,$19,$16
032c 00 0d 0c 07 06 01          fcb     $00,$0d,$0c,$07,$06,$01


0332 80                 softin  rti

03fa                            org     $3fa

03fa 02 18                      fdb     presd           ; scan keybrd on int
03fc 03 32                      fdb     softin          ; software interrupt
03fe 02 00                      fdb     start           ; resett
```