KEY DECODING

OVERVIEW

The routine described in this application note, “key_decoding,” determines the key ID (look-up table index value) each time a key is pressed.

Table 3-1. Arguments and Register Allocations for “key_decoding”

Argument	Addressing	Description
keyinf	Register (bit-wise)	This one-bit flag indicates whether or not a key is valid.
KeyValue	Working register	This working register is used to save the key index value that is used to jump to the key service routine for each input.
Buf	Working register	This working register is used to save the key index code (00H –1FH) that is manipulated by key position.
r0, r1	Working registers r0,r1	These working registers are used to store the start address of each key service handler.
r2, r3	Working registers r2,r3	These working registers are used to read out the start address of the key service handler from the look-up table "keydecidx".

Programming Guidelines

— The key_decoding routine is accessed by the main routine. The main routine has to call key_decoding before key_scan can be executed.

— Because key_decoding determines key identification by manipulating oldkeyin and oldkeyout register values, changing the oldkeyin and oldkeyout value can cause the main key decoding operation to execute. The key_scan subroutine is always located in the key service program at an address *before* the start of the key_decoding subroutine.

Basic Operation

1. First, test if the flag ‘keyinf’ in the main routine is High level. If it is not High, return to the main routine.
2. Otherwise, clear the keyinf flag and calculate the start address of the key service handler, as follows:
   a. Convert the key output data values FEH, FDH, FBH, F7H, EFH, DFH, BFH, and 7FH to the corresponding indexed values — 0, 1, 2, 3, 4, 5, 6, and 7, respectively.
   b. Add the indexed value by some weighted value that is obtained by key input data. The binary input data values 0000 0110B, 0000 0101B, and 0000 0011B have the weighted values 0H, 08H, and 10H, respectively.
   c. The result of steps ‘a’ and ‘b’ becomes the key identification code (00H–1FH).
   d. By manipulating the key ID code and the look-up table for the key service routine’s entry, read out the start address of each key service routine:
      — Because the start address of the key service routine has a 2-byte length, multiply the key ID code by two.
      — Add the start of address from the key decode look-up table.
      — Read the key service address from the preceding action.
3. Jump to the appropriate key service routine.

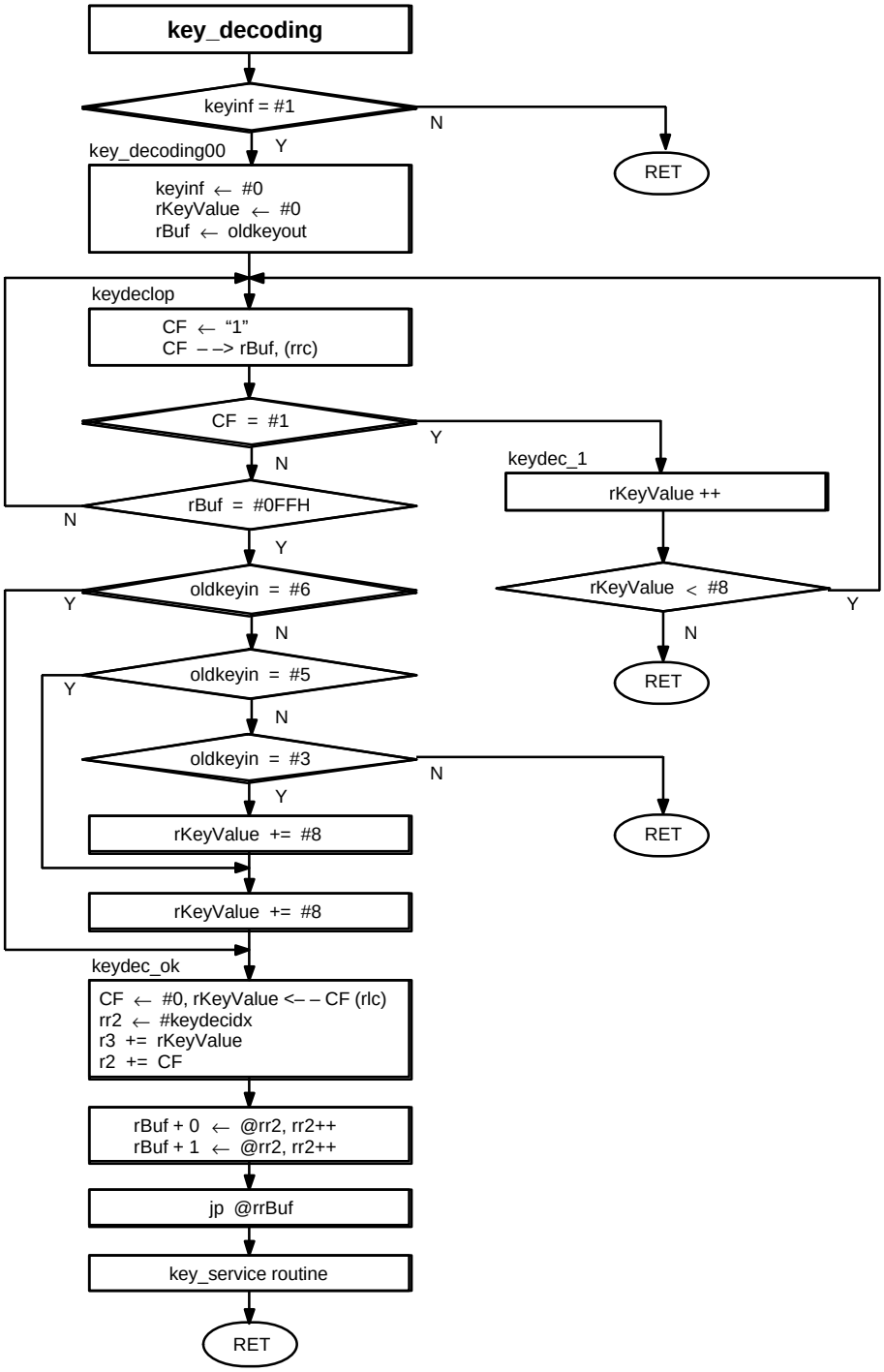

Figure 3-1. Program Flowchart for "key_decoding"

SOURCE CODE FOR KEY DECODING ROUTINE

```
;=============================================
;======          Key Decoding Routine          ======
;=============================================

; Register Definition Table:
Buf             equ     0               ;  r0
KeyValue        equ     1               ;  r1
TIMEFLAG        equ     50H

; Constant Equation:
keyinf          equ     02H

key_decoding:
                tm      TIMEFLAG,#keyinf        ;  Test if keyinf is on
                jr      nz,key_decoding00       ;  Yes
                ret                             ;  If no, return to main procedure

key_decoding00:
                xor     TIMEFLAG,#keyinf        ;  Cear the keyinf flag
                ld      rBuf,oldkeyout          ;  If yes, determine the key ID code
                ld      rKeyValue,#0H           ;  Key strobe  →  rBuf

keydeclop:
                scf                             ;  1111 1110   “0”
                rrc     rBuf                    ;  1111 1101   “1”
                jr      c,keydec_1              ;  1111 1011   "2"
                cp      rBuf,#0FFH              ;
                jr      ne,keydeclop            ;  0111 1111   "7"
                ld      rBuf,oldkeyin           ;  keyin data  →  r1
                cp      rBuf,#6H                ;  xxxx x110  +  0
                jr      eq,keystrb1             ;  xxxx x101  +  8
                cp      rBuf,#5H                ;  xxxx x011  +  16
                jr      eq,keystrb2             ;  where  x  =  “0”, but has no meaning
                ret                             ;  Double key input or only one key output

keydec_1:
                inc     rKeyValue
                cp      rKeyValue,#8H
                jr      ult,keydeclop
                ret                             ;  Two or more keys are pressed simultaneously

keystrb3:
                add     rKeyValue,#8H

keystrb2:
                add     rKeyValue,#8H

keystrb1:
```

SOURCE CODE FOR KEY DECODING SUBROUTINE (Cont.)

```
keydec_ok:
                                                ;  Keyvalue ← key ID code (0–23)
            rcf         cf
            rlc         rKeyValue               ;  Multiply by two because Jump address to key service
                                                ;  routine has a 2-byte length
            ldw         rr2,#keydecidx          ;  Adjust the pointer of key function
            add         r3,rKeyValue
            adc         r2,#0H
            ldci        rBuf,@rr2               ;  Read key function pointer from look-up table in ROM
            ldci        rBuf+1,@rr2
            jp          @rrBuf                  ;  Jump to each function key subroutine


;=======================================
;===  Key Function Service Routine Entry Table   ===
;=======================================

; Key jump address     =   Base address  +  Offset (index value)
;                      =   keydecidx  +  key ID code

; Key service routine for each key              ;  Key ID code

keydecidx:
            dw      keyfnc0, keyfnc1, keyfnc2       ;  00, 01, 02
            dw      keyfnc3, keyfnc4, keyfnc5       ;  03, 04, 05
            dw      keyfnc6, keyfnc7, keyfnc8       ;  06, 07, 08
            dw      keyfnc9, keyfnc10, keyfnc11     ;  09, 0A, 0B
            dw      keyfnc12, keyfnc13, keyfnc14    ;  0C, 0D, 0E
            dw      keyfnc15, keyfnc16, keyfnc17    ;  0F, 10, 11
            dw      keyfnc18, keyfnc19, keyfnc20    ;  12, 13, 14
            dw      keyfnc21, keyfnc22, keyfnc23    ;  15, 16, 17
```