SAMSUNG ELECTRONICS

Sample Analog Input and Key Input Routines Using a 4-channel Comparator

Application Note: KS57APN7

Application Engineering Department
LSI 2 Division, Micom Sector

COMPARATOR APPLICATIONS

OVERVIEW

The KS57C0002 microcontroller has a 4-channel comparator. The reference voltage can be supplied either internally or externally at P2.3. The internal reference voltage is stepped in 16 levels. When a conversion is completed, the result is saved in the comparison result register, CMPREG.

CMOD register settings are used to control comparator operation. P2MOD register settings configure port 2 for analog or digital input. The comparator logic is as follows:

If “1”:	Analog input voltage	V_{REF} + 150 mV
If “0”:	Analog input voltage	V_{REF} – 150 mV

NOTE

Voltage (an analog quantity) that is applied to the port pins can be converted into digital data with 4-bit precision (see Figure 7-1).

ANALOG INPUT APPLICATION

Function Description

The program example presented in this section converts a voltage applied to any analog input port pin (CINn) into digital data. The program provides 4-bit resolution and changes the reference voltage in four steps.

Each time the reference voltage is changed, it is compared with the voltage that is currently being applied to a given analog input port pin (CINn). The result of the comparison is then stored in bit sequential carrier 0 (BSC0), starting from the most significant bit. The threshold voltage is determined by the BSC0 value (see Figure 7-2).

More specifically, the program compares the values of the lower four bits of CMOD with the input pin voltage (CINn) in four steps, starting from a value of 0, and increasing to 0EH (reference voltage of 1/32 V_{DD} to 29/32 V_{DD}). This operation is accomplished by means of a binary search.

The result of each comparison is stored in BSC0. By modifying the BSC0 value in CMOD, you can also compare the lower four bits of CMOD starting from 1 and increasing to 0FH (reference voltage of 3/32 V_{DD} to 31/32 V_{DD}).

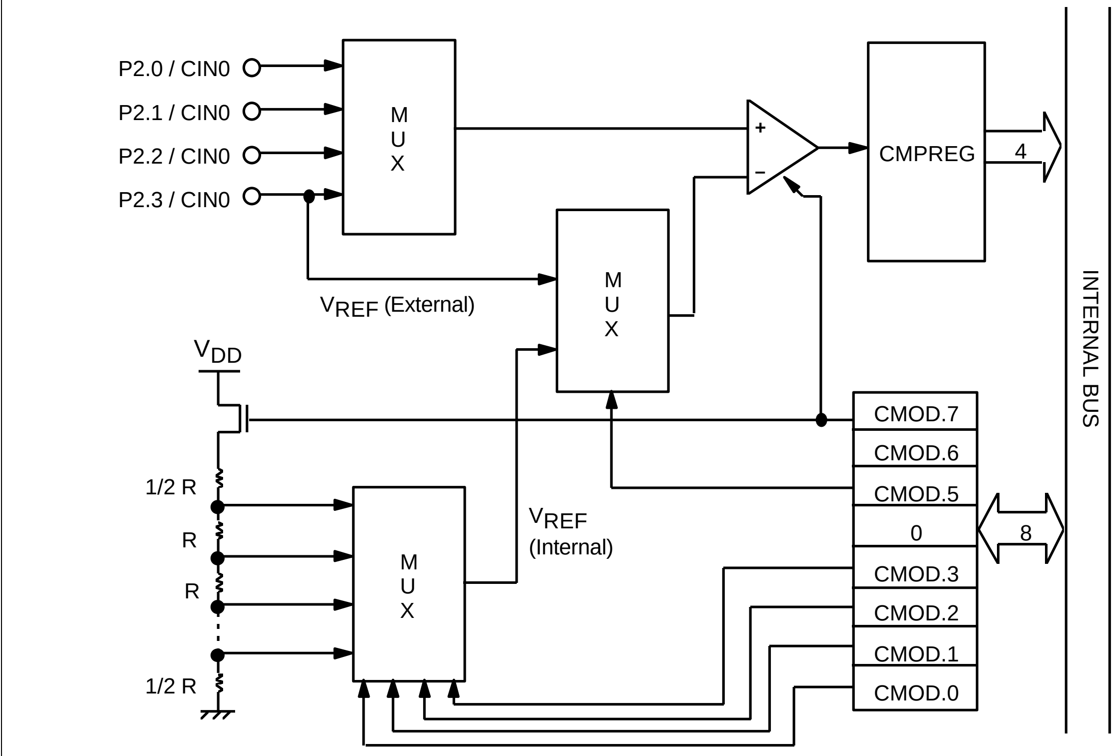

Figure 7-1. Comparator Circuit Diagram for Analog Input Routine

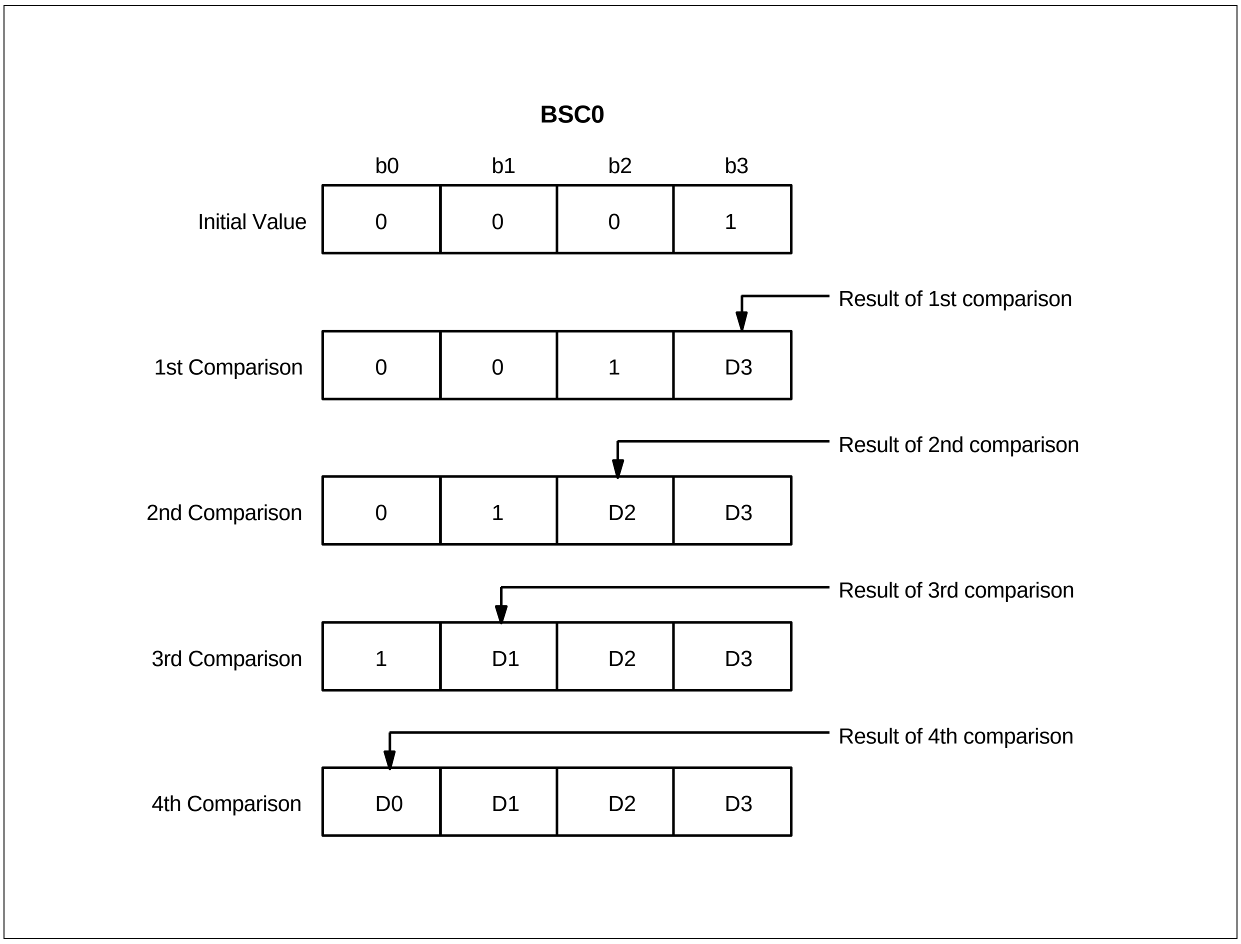

Figure 7-2. Storage of Comparison Results in BSC0

Programming Guidelines

Hardware settings:

P2MOD ← #0H	; Set P2.0/CIN0 – P2.3/CIN3 to analog input
High 4 bits of CMOD ← #0CH	; Internal reference: conversion time = 15.2 µs @4.19 MHz; disable comparator

Nesting :	One level
Hardware assignment:	Comparator input port 0 (CIN0)
Bit sequential carrier:	BSC0

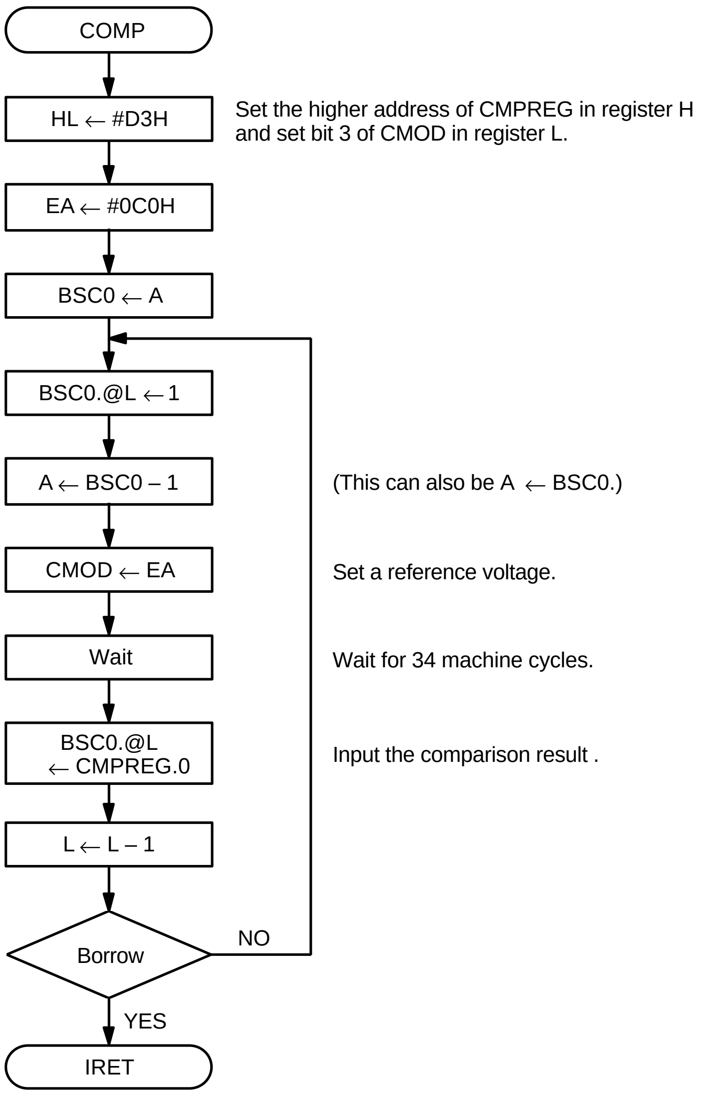
COMP
HL ← #D3H
Set the higher address of CMPREG in register H and set bit 3 of CMOD in register L.
EA ← #0C0H
BSC0 ← A
BSC0.@L ← 1
A ← BSC0 – 1
(This can also be A ← BSC0.)
CMOD ← EA
Set a reference voltage.
Wait
Wait for 34 machine cycles.
BSC0.@L ← CMPREG.0
Input the comparison result .
L ← L – 1
Borrow
NO
YES
IRET

Figure 7-3. Program Flowchart for Analog Input Routine

Source Code for Analog Input Routine

```
          CHIP      C:\SMDSII\DATA\57C0002.DEF

; ==========================================================

; Comparator service routine:

COMP
          BITS      EMB
          SMB       15
          LD        HL,#0D3H            ;  Set the highest 4 bits of CMPREG in register H
          LD        EA,#0C0H            ;  Set bit 3 of CMOD in register L
          LD        BSC0,A              ;  BSC ← 0

LOOP
          BITS      BSC0.@L
          LD        A,BSC0
          DECS      A
          LD        CMOD,EA             ;  Start comparison
          LD        A,#0BH              ;  Wait 36 machine cycles

WAIT
          DECS      A
          JR        WAIT
          RCF
          LD        A,CMPREG
          AND       A,#0001B
          CPSE      A,#0
          CCF
          LDB       BSC0.@L,C           ;  Store the comparison result
          DECS      L
          JP        LOOP
          RET

; ==========================================================
```

KEY INPUT ROUTINE

Function Description

You can use comparator inputs CIN0–CIN3 to process key input signals. In the following sample program, the comparator is used to process input signals from up to 178 keys. When a key is pressed, a reference voltage is input to a comparator input pin. The input voltage is measured by software to determine which key has been pressed.

Figure 7-4 shows how the reference voltage is measured when one of the eight keys is pressed. The circuit diagram in Figure 7-5 shows how the reference voltage is applied to the specific input pin when one of the eight keys is pressed.

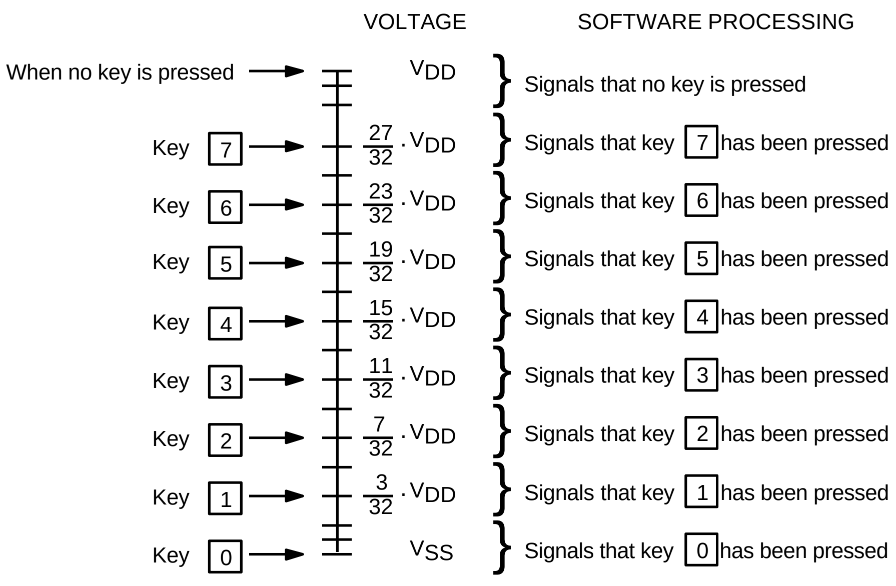
VOLTAGE
SOFTWARE PROCESSING
When no key is pressed
VDD
Signals that no key is pressed
Key 7
27/32 · VDD
Signals that key 7 has been pressed
Key 6
23/32 · VDD
Signals that key 6 has been pressed
Key 5
19/32 · VDD
Signals that key 5 has been pressed
Key 4
15/32 · VDD
Signals that key 4 has been pressed
Key 3
11/32 · VDD
Signals that key 3 has been pressed
Key 2
7/32 · VDD
Signals that key 2 has been pressed
Key 1
3/32 · VDD
Signals that key 1 has been pressed
Key 0
VSS
Signals that key 0 has been pressed

Figure 7-4. Eight-Key Example for Key Input Routine

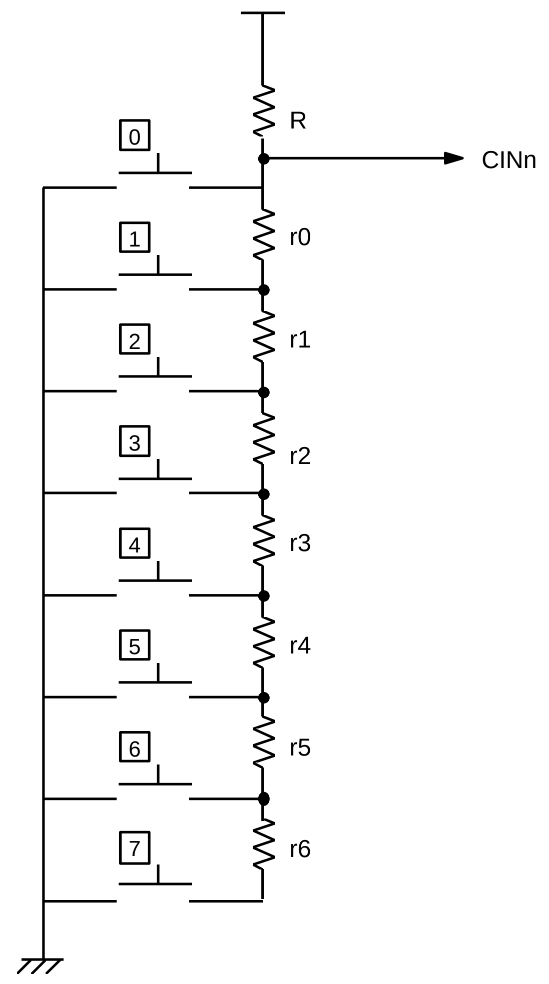

Figure 7-5. Key Input Circuit

Determining the Resistance in a Key Input Circuit

To calculate the resistance of r0 to r6 (shown in Figure 7-5), use the following equation:

$$\sum_{k=0}^{n} rk = \frac{3 + 4n}{29 - 4n} R$$

To obtain the value of r0 to r6, substitute 0 to 6 for ‘n’ in the equation above. For example, if ‘R’ is 2.2 KΩ, then:

r0 = 240 Ω, r1 = 390 Ω, r2 = 510 Ω, r3 = 820 Ω, r4 = 1.3 kΩ, r5 = 2.4 kΩ, and r6 = 6.2 kΩ

Using the Binary Search Method

In the key input circuit, the key with the lower number takes precedence when two or more keys are pressed simultaneously. The sample program uses a binary search method to measure comparator input voltage. It performs four searches of the reference voltage, regardless of the pin voltage value. Figure 7-6 illustrates the binary search method and the corresponding key codes.

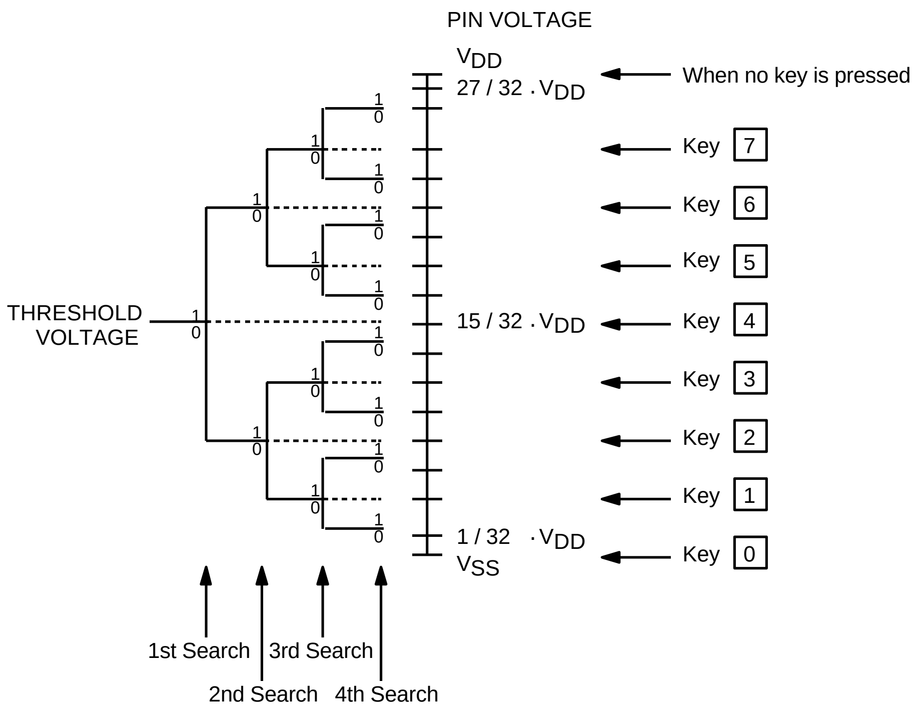

Figure 7-6. Binary Search Method for Key Input Routine

Procedure for Storing Key Code Reference Values

The vertical axis in Figure 7-6 indicates the value of the reference voltage applied to the pin when each of the keys shown on the right has been pressed.

The voltage value obtained during each search operation is stored in a register, starting from the highest bit toward the lowest. The value obtained during the first search is stored in bit 3 of the register, the value obtained during the second search is stored in bit 2, and so on. A decimal 1 is added to the voltage values stored in this manner. Then the contents of register are shifted right one bit to create codes that correspond to the respective keys. If no key is pressed, key code 8 is assumed.

For example, suppose key "3" is pressed. The value obtained by a binary search ("0101B" or "0110B") is stored in a register. In either case, add decimal 1 to the register value and shift the value one bit to the right. The result will be a 3 in decimal format, as illustrated below. Key code 3 is therefore assigned to key "3".

$$0101 + 1 = 0110\,B \xrightarrow{\text{shift}} 0011\,B$$

$$0110 + 1 = 0110\,B \xrightarrow{\text{shift}} 0011\,B$$

In the program example, key inputs are processed using the basic timer. It is assumed that the ON and OFF chattering times are 10 ms. Each time a key code changes, the change flag, CHNGFG, is set.

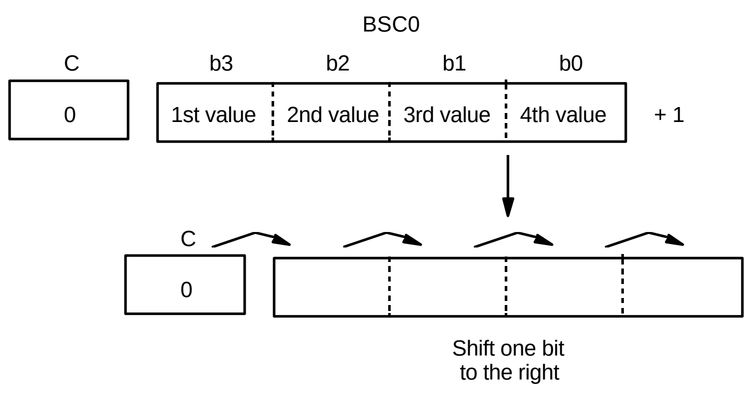

Figure 7-7. Key Code Store Operation

RAM Allocation

Address	50H	51H	52H	53H
	NEW	OLD	DOUCNT	FLAG

NEW:	Stores the codes when chattering occurs
OLD:	Stores the key codes of valid keys
DOUCNT:	Chattering counter
CHNGFG (53H.0):	Change flag

Programming Guidelines

Hardware settings:

BMOD ← #0FH	; Interval time = 1.95 ms
P2MOD ← #0H	; Set CIN0–CIN3 to analog input

Pin assignment:	Comparator input pin (CIN0)
Bit sequential carrier:	BSC0
Nesting:	One level
Interrupt:	INTB

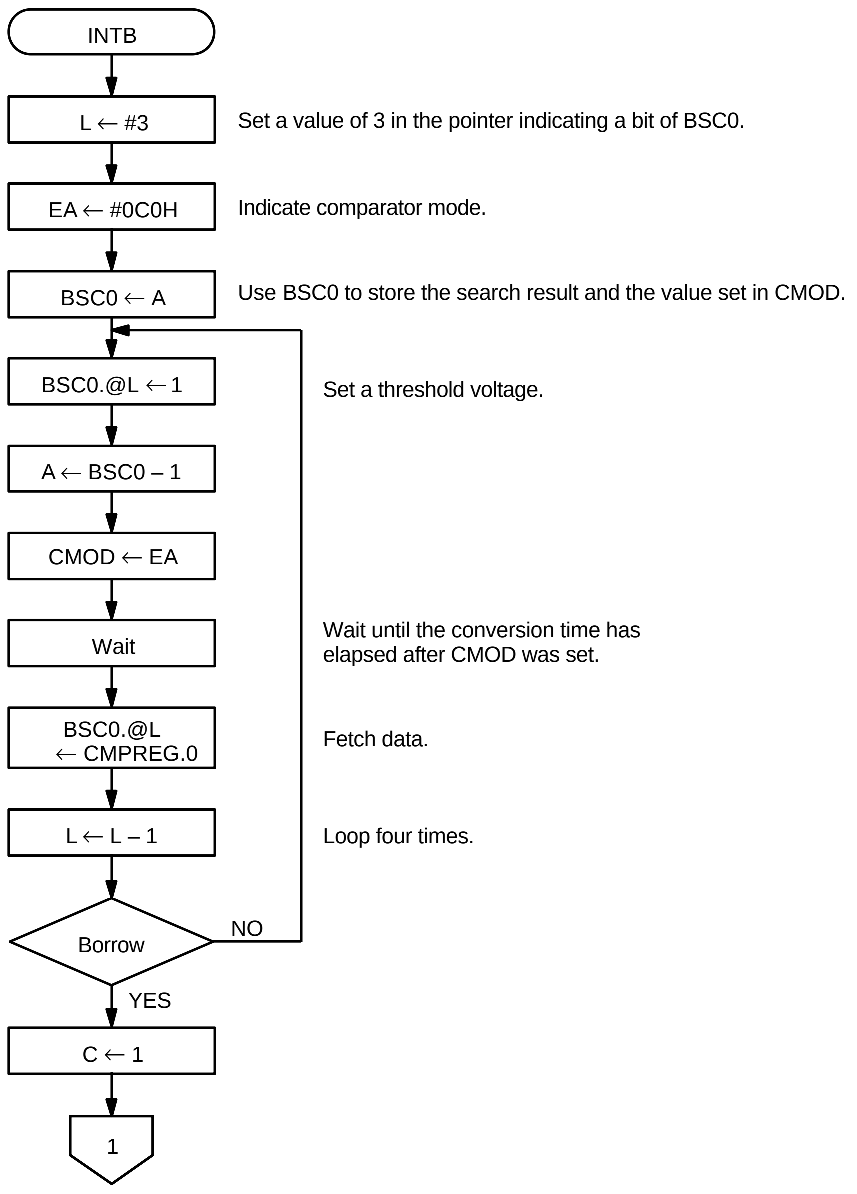
INTB
L ← #3
Set a value of 3 in the pointer indicating a bit of BSC0.
EA ← #0C0H
Indicate comparator mode.
BSC0 ← A
Use BSC0 to store the search result and the value set in CMOD.
BSC0.@L ← 1
Set a threshold voltage.
A ← BSC0 – 1
CMOD ← EA
Wait
Wait until the conversion time has elapsed after CMOD was set.
BSC0.@L ← CMPREG.0
Fetch data.
L ← L – 1
Loop four times.
Borrow
NO
YES
C ← 1
1

Figure 7-8. Program Flowchart for INTB Processing (Key Input Routine)

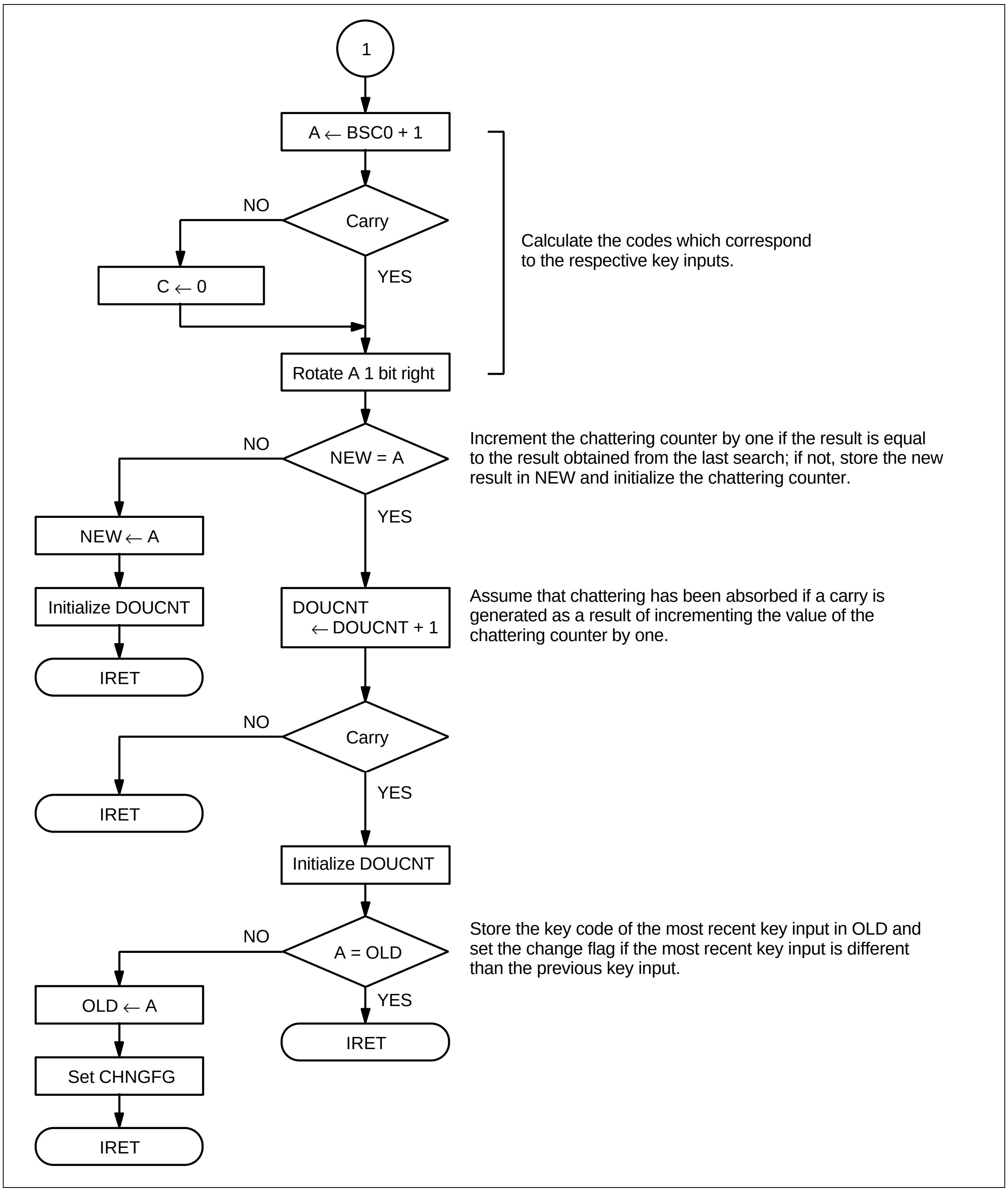
1
A ← BSC0 + 1
Carry
NO
YES
C ← 0
Rotate A 1 bit right
Calculate the codes which correspond to the respective key inputs.
NEW = A
NO
YES
NEW ← A
Initialize DOUCNT
IRET
DOUCNT ← DOUCNT + 1
Increment the chattering counter by one if the result is equal to the result obtained from the last search; if not, store the new result in NEW and initialize the chattering counter.
Assume that chattering has been absorbed if a carry is generated as a result of incrementing the value of the chattering counter by one.
Carry
NO
YES
IRET
Initialize DOUCNT
A = OLD
NO
YES
OLD ← A
Set CHNGFG
IRET
IRET
Store the key code of the most recent key input in OLD and set the change flag if the most recent key input is different than the previous key input.

Figure 7-9. Program Flowchart for Key Input Routine

Source Code for Key Input Routine

```
          CHIP     C:\SMDSII\DATA\57C0002.DEF

NEW       EQU      50H                 ;  Key code when chattering
OLD       EQU      51H                 ;  Valid key code
DOUCNT    EQU      52H                 ;  Debouncing counter
CHNGFG    EQU      53H.0               ;  Change flag

          ORG      0002H
          VENT1    0,0,INTB

; ============================================================

; Initial settings:

INITIAL
          BITR     EMB
          LD       A,#0
          LD       P2MOD,A             ;  Set CIN0–CIN3 to analog input
          LD       A,#0BH
          LD       DOUCNT,A
          LD       A,#8H
          LD       NEW,A
          LD       OLD,A
          BITR     CHNGFG
          LD       A,#0FH
          LD       BMOD,A
          BITS     IEB
          EI
          RET

; INTB processing:

INTB
          PUSH     SB
          SRB      1
          LD       L,#3H               ;  EMB = 0, ERB = 0; specify bit 3
          LD       EA,#0C0H
          LD       BSC0,A

LOOP
          BITS     BSC0.@L
          LD       A,BSC0
          DECS     A
          LD       CMOD,EA             ;  Start comparison
          LD       A,#0BH
```

Source Code for Key Input Routine (Cont.)

```
WAIT
            DECS        A                       ;  Wait for conversion time
            JR          WAIT
            RCF
            LD          A,CMPREG
            AND         A,#0001B
            CPSE        A,#0H
            CCF
            LDB         BSC0.@L,C               ;  Store the conversion result
            DECS        L
            JP          LOOP

            LD          A,BSC0                  ;  Calculate key code
            CCF
            ADS         A,#1
            RCF
            RRC         A

            LD          HL,#NEW
            CPSE        A,@HL
            JP          FKEY2
            INCS        DOUCNT
            POP         SB
            IRET

            LD          A,#0BH
            LD          DOUCNT,A
            LD          A,OLD
            CPSE        A,@HL
            JR          FKEY1
            POP         SB
            IRET

FKEY1
            LD          A, @HL
            LD          OLD,A
            BITS        CHNGFG                  ;  Set the conversion flag
            POP         SB
            IRET

FKEY2
            LD          NEW,A                   ;  Store key code in OLD
            LD          A,#0BH
            LD          DOUCNT,A                ;  Set the conversion flag
            POP         SB
            IRET
            END

; ==========================================================
```