SAMSUNG ELECTRONICS

Four Sample Applications for the KS57-Series Basic Timer Module

Application Note: KS57APN1

Application Engineering Department
LSI 2 Division, Micom Sector

BASIC TIMER APPLICATIONS

USING THE BASIC TIMER AS AN INTERVAL TIMER

The primary function of the basic timer (BT) is to measure elapsed time intervals. You can program the KS57-series basic timer module to measure four different time intervals, based on the selected CPU clock. The basic timer module includes a BT mode register, BMOD, and an 8-bit counter, BCNT.

The BCNT value is incremented each time a clock signal is detected which corresponds to the frequency you select using BMOD register settings. When a counter overflow occurs, the basic timer interrupt request flag, IRQB (location FB8H.0), is set to “1” to signal that the designated time interval has elapsed. Next, the basic timer interrupt is generated, BCNT is cleared to zero, and counting resumes from 00H.

You can restart the basic timer (and clear the BCNT value) at any time by setting BMOD.3 to “1”.

Setting the Basic Timer Interval

The following program example shows how to set the basic timer interval:

```
        BITS    EMB
        SMB     15
        LD      A,#1011B
        LD      BMOD,A              ;   Set the IRQB flag every 31.3 ms (fxx = 4.19 MHz)
```

Reading the BCNT Value

To eliminate the possibility of reading unstable data while the counter is incrementing, always execute a BCNT read operation twice. If, after two consecutive read operations, the BCNT values match, you can select the latter value as valid data. Continue to read the BCNT value, however, until this validation condition is met.

The following program code illustrates the looping read operation for BCNT:

```
        BITS    EMB
        SMB     15
        LD      HL,#BCNT

LOOP
        LD      EA,@HL              ;   First read
        LD      YZ,EA
        LD      EA,@HL              ;   Second read
        CPSE    EA,YZ
        JR      LOOP
```

USING THE BASIC TIMER AS A WATCHDOG TIMER

You can use the basic timer as a watchdog timer to prevent program overruns or to escape from an infinite loop. To implement this function in an application program, follow these guidelines:

1. Divide a program into several modules and estimate the time it takes the MCU to process each module under normal operating conditions.
2. Set the basic timer interval for each module to be longer than the module’s normal processing time. This basic timer interval setting should be performed at the start of each program module.
3. When the pre-set basic timer interval of a program module has elapsed, the BT counter (BCNT) should be reset and the BT restarted.
4. If the BT cannot be restarted within the program module’s normal execution time, a basic timer interrupt is generated to signal a possible system malfunction.

Source Code for Watchdog Timer Routine

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

; Reset routine:

RESET
            •
            •
            •
            BITS      EMB
            SMB       15
            LD        A,#1101B
            LD        BMOD,A                ;   Make BT settings and start
            BITS      IEB
            EI

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

; Main routine:

MAIN
            CALL      MODULE1               ;   After 31.3 ms (BMOD = #0BH, fxx = 4.19 MHz)
            CALL      MODULE2               ;   After 7.8 ms (BMOD = #0DH, fxx = 4.19 MHz)
            JP        MAIN

MODULE1
            BITS      EMB
            SMB       15
            LD        A,#1011B
            LD        BMOD,A
            •
            •
            •
            BITS      EMB
            SMB       15
            BITS      BMOD.3
            RET
```

Source Code for Watchdog Timer Routine (Cont.)

```
MODULE2
```

```
        BITS    EMB
        SMB     15
        LD      A,#1011B
        LD      BMOD,A
        •
        •
        •
        BITS    EMB
        SMB     15
        BITS    BMOD.3
        RET

INTB
        BITR    IS0
        BITR    IS1
        JP      RESET

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

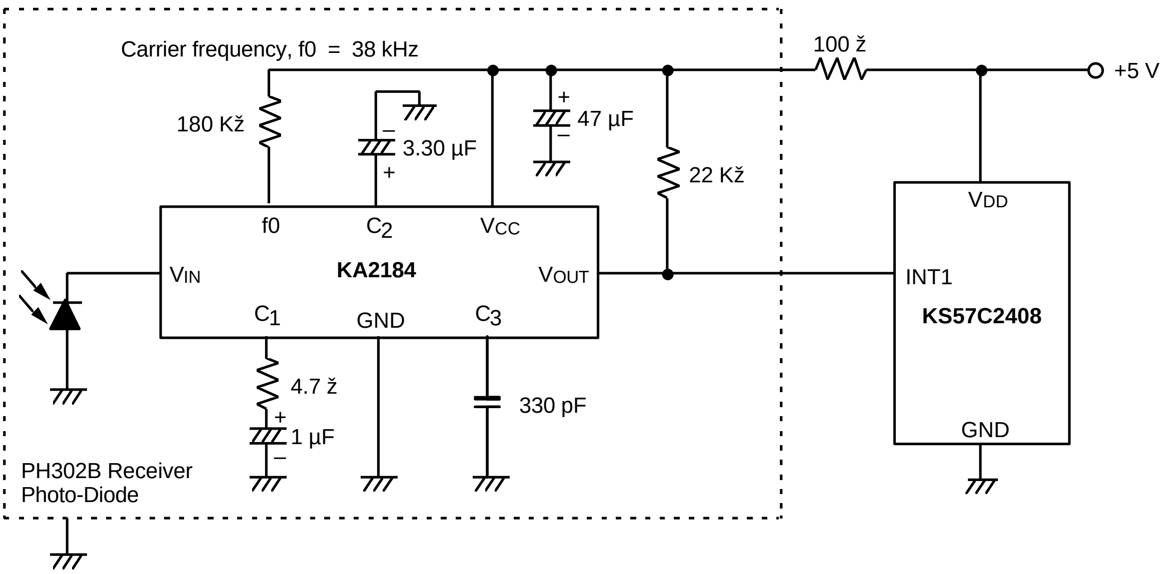
USING THE BASIC TIMER TO RECEIVE SIGNALS FROM A REMOTE CONTROLLER

Function Description

The application program described in this section uses the basic timer to receive data transmitted from a remote controller. The remote controller signal is input at the INT1 pin of a KS57-series microcontroller through a preamplifier circuit, as shown in Figure 1-1.

To encode the data, the time intervals between remote control signals are measured. A typical remote controller signal consists of a leader pulse, custom code, and data code (see KS57APN10). A remote controller receiver circuit inverts the received signal and removes the carrier frequency.

In an actual application, considerable circuit noise may occur before the leader code is received. To protect against signal disruption by this noise, this sample program uses the basic timer to detect the falling edge of the leader code at the external interrupt input pin (INT1).



NOTE: The KA2184 remote controller integrated circuit is manufactured by Samsung Electronics.

Figure 1-1. Remote Controller Carrier Signal (f0) Receiver Circuit

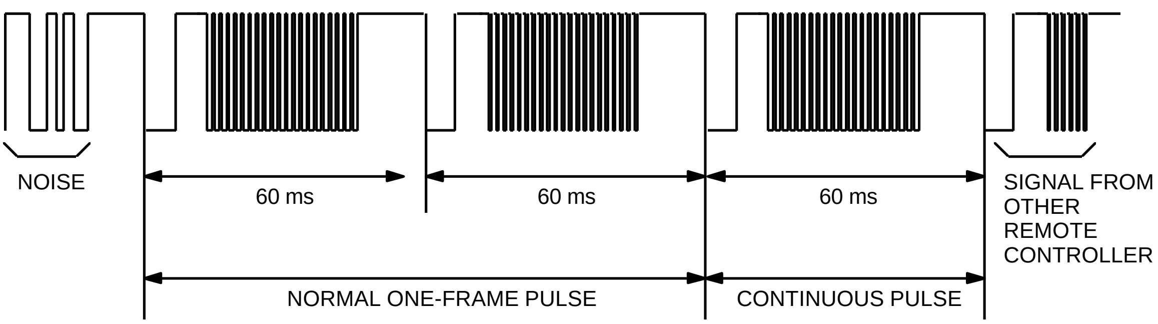


Figure 1-2. Waveform Received From Remote Controller

RAM Allocation

Address	40H	41H	42H	43H	44H	45H	46H	47H	48H	49H	4AH	4BH	4CH	4DH	4EH
Data Value (see NOTE)	1		2	3	4					5	6		7		8

NOTE:
1 = CPCNT (2 nibbles)
2 = STATE (1 nibble)
3 = LECNT (1 nibble)
4 = REDATA (5 nibbles)
5 = Not used
6 = CODEBUF (2 nibbles)
7 = VALID (2 nibbles)
8 = Flags:
VALFG (4EH.0)
CPFG (4EH.1)
FIRFG (4EH.2)

Figure 1-3. RAM Allocation for Basic Timer Remocon Application

For this application, data RAM is allocated to addresses 40H–4EH of memory bank 0, where

CPCNT: Counts the elapsed time since a valid code was input
STATE: Mode status, indicating the last signal edge which was detected
LECNT: Measures the time during which the leader code is Low level
REDATA: This area is used to store receive data
CODEBUF: Buffer area for last code input
VALID: Test data for code validity check
FLAGS:
- VALFG (4EH.0): Flag is set when a valid code is input
- CPFG (4EH.1): Flag is set when a valid constant pulse is input
- FIRFG (4EH.2): Flag is set when the first 60-ms pulse is input

Programming Guidelines

To properly initialize the basic timer, write the following values to the basic timer mode register and the IPR register:

```
BMOD ← #0FH        ; Select the basic timer interrupt with a 1.95-ms interval
IPR ← #0BH         ; Select INT1 as the highest-priority interrupt
```

Interrupts: INT1 and INTB
Nesting: Two levels
Port assignment: P1.1 (shared with INT1)

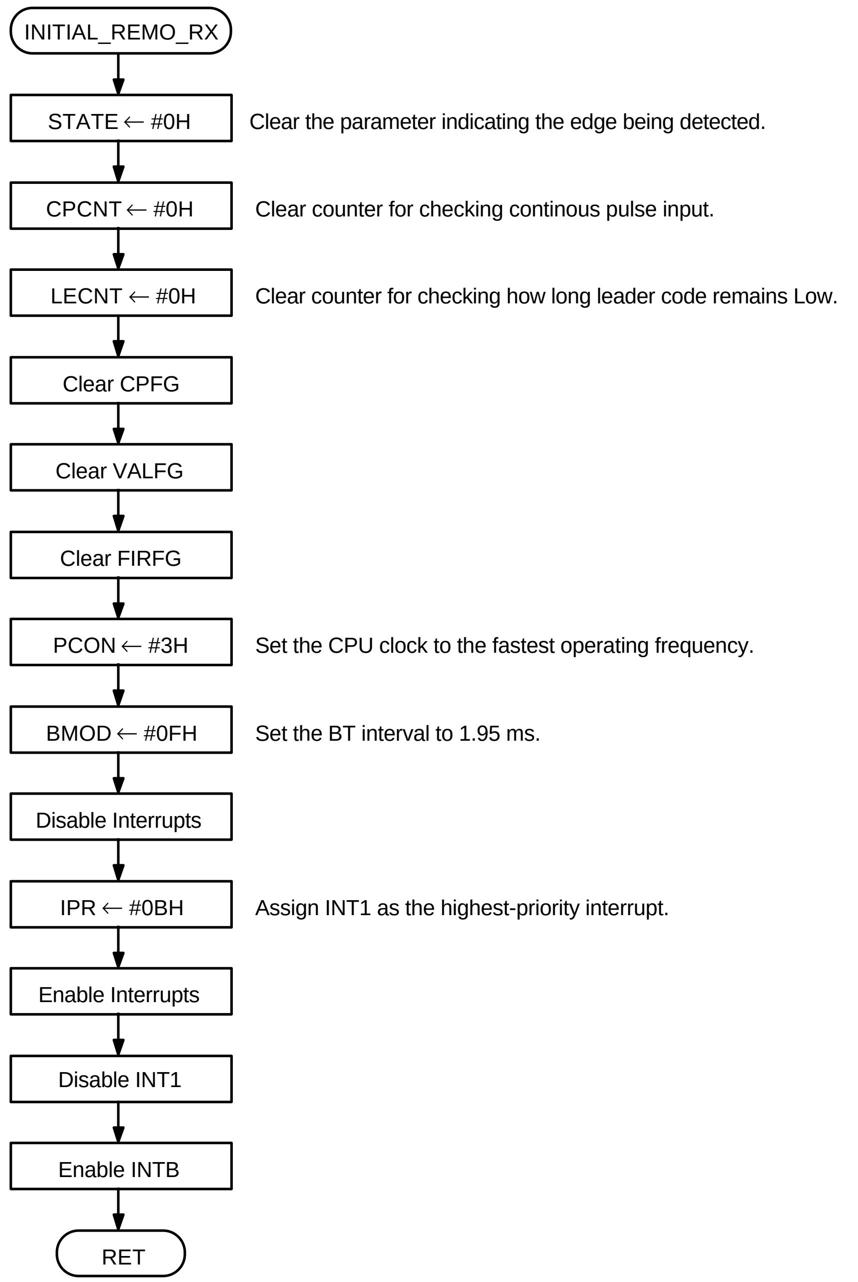


Figure 1-4. Program Flowchart for Remote Controller Rx Initialization

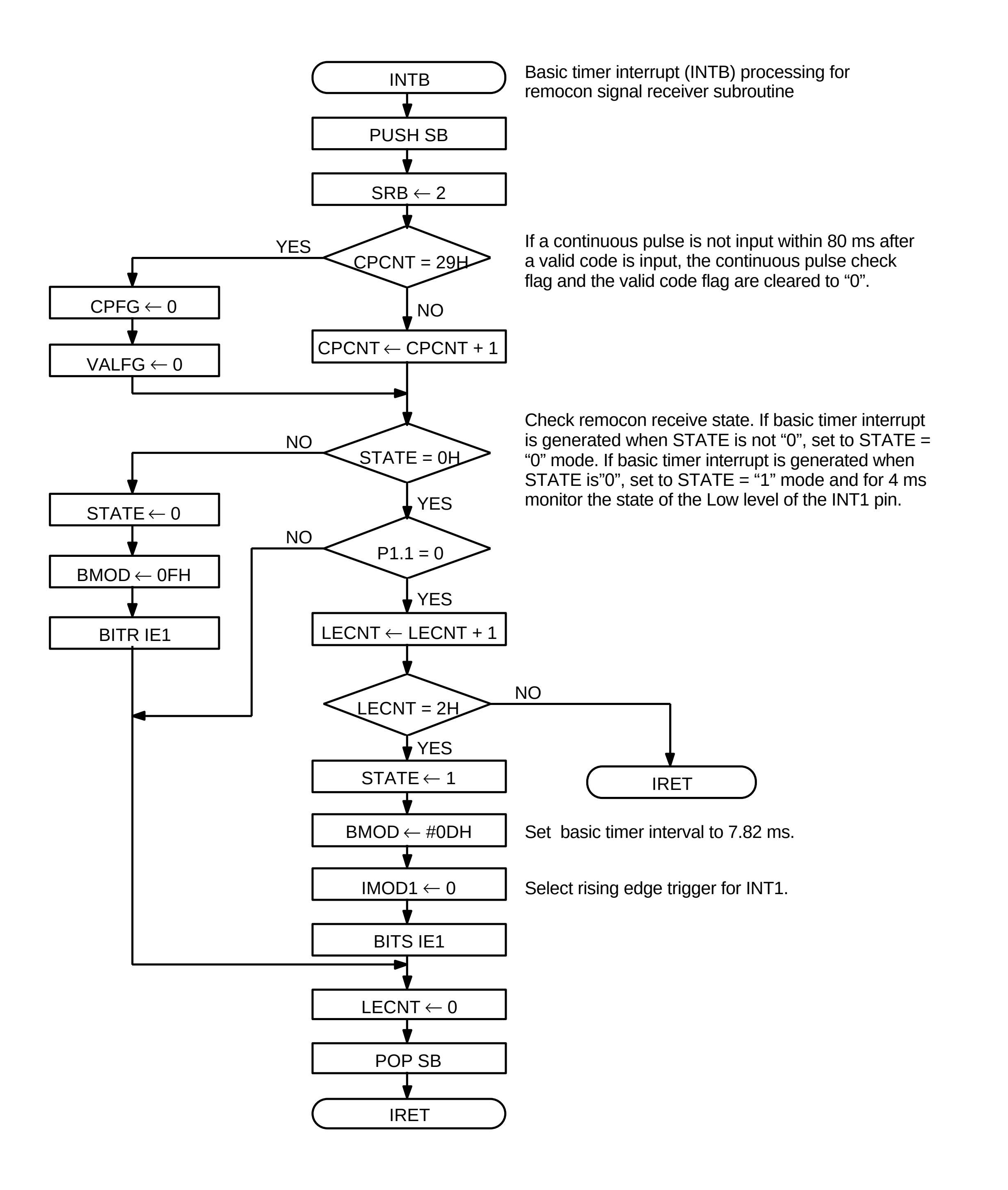


Figure 1-5. Program Flowchart for Basic Timer Interrupt (INTB) Processing

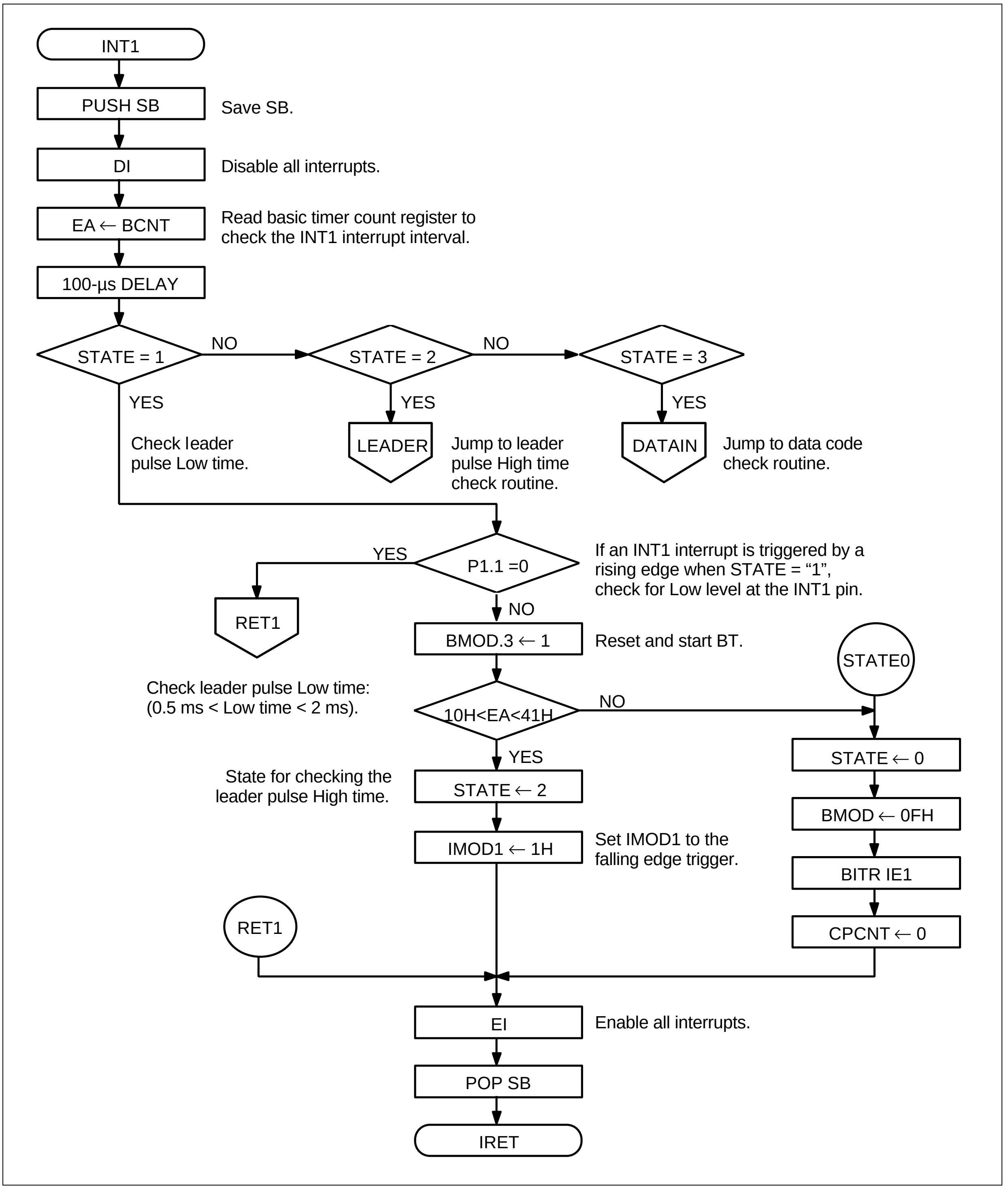


Figure 1-6. Program Flowchart for INT1 Processing

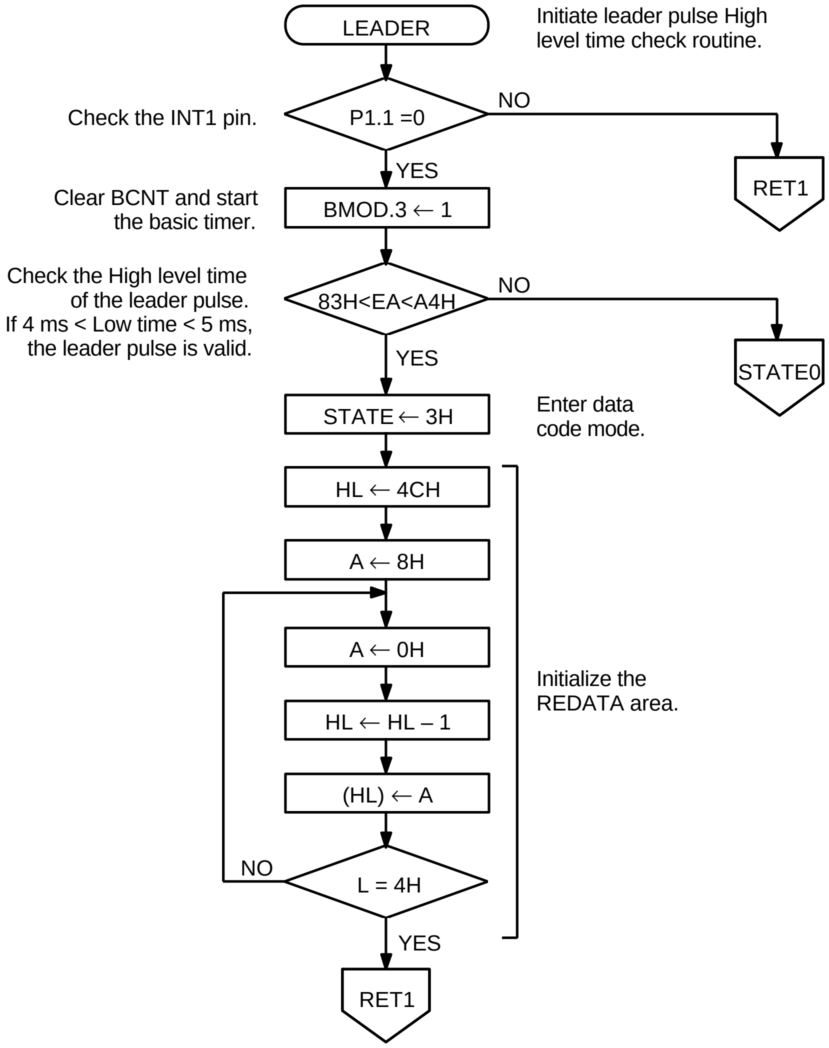


Figure 1-7. Program Flowchart for Leader Pulse High Level Check Routine

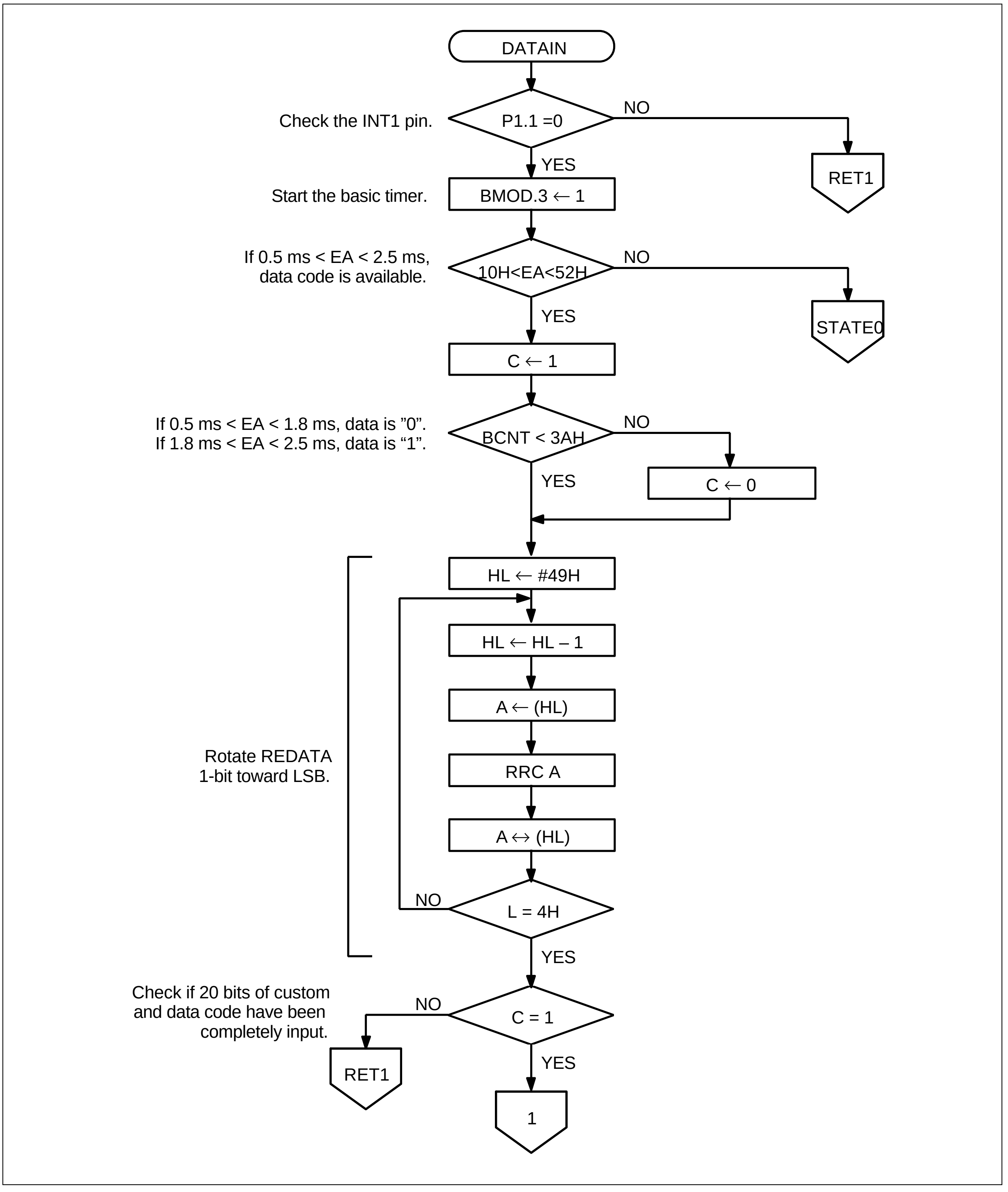


Figure 1-8. Program Flowchart for Data Code Check Routine

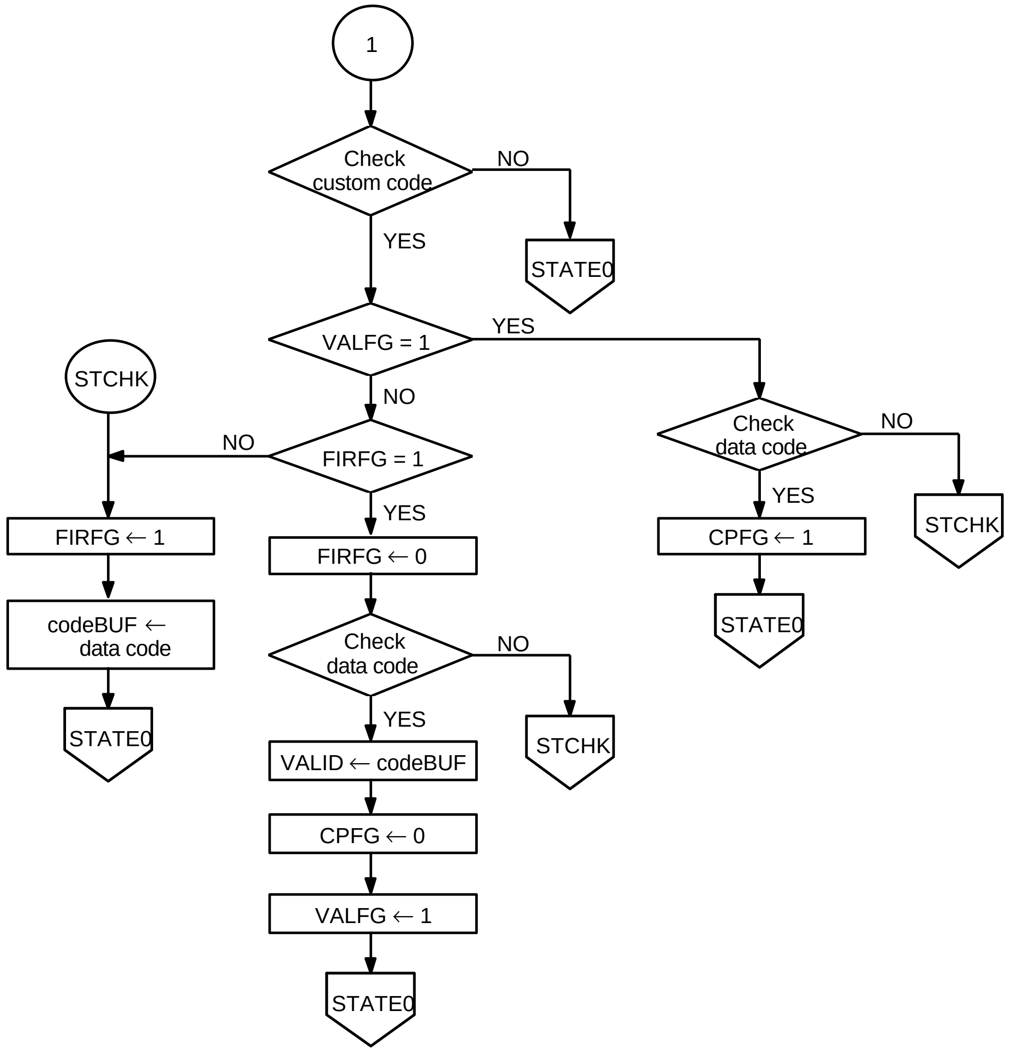


Figure 1-9. Program Flowchart for Custom Code Check

Source Code for Remote Controller Application

```
; CHIP        C:\SMDSII\DATA\57C2408.DEF

CPCNT                   EQU    40H
STATE                   EQU    42H
LECNT                   EQU    43H
REDATA                  EQU    44H
VALID                   EQU    4CH
FLAG                    EQU    4EH
VALFG                   EQU    FLAG.0
CPFG                    EQU    FLAG.1
FIRFG                   EQU    FLAG.2
CUSTOMA                 EQU    55H
CUSTOMB                 EQU    56H
            ORG           0000H
            VENT1         0,1,INTB
            ORG           0006H
            VENT3         0,0,INT1

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

; Initialize remocon receive subroutine:

INITIAL_REMO_RX
            BITR          EMB
            LD            EA,#00H
            LD            STATE,A
            LD            CPCNT,EA
            LD            LECNT,A
            BITR          CPFG
            BITR          VALFG
            BITR          FIRFG
            LD            A,#3H
            LD            PCON,A                ;  Set to high-speed mode
            LD            A,#0FH
            LD            BMOD,A                ;  Set timer interval to 1.95 ms
            DI
            LD            A,#0BH
            LD            IPR,A                 ;  Assign INT1 highest interrupt priority level
            EI
            BITR          IE1
            BITS          IEB
            RET
```

Source Code for Remote Controller Application (Cont.)

```
; INTB processing:

INTB
        PUSH    SB
        SRB     2               ;  EMB = 0, SRB = 2
        LD      EA,CPCNT        ;  CPCNT = 29H check
        LD      YZ,#0D7H
        ADS     EA,YZ           ;  Repeat code check in 80-ms intervals
        JPS     INCRIT

        LD      EA,#00H
        LD      VALID,EA
        BITR    CPFG            ;  Repeat input flag ← "0"
        BITR    VALFG           ;  Valid code input flag ← "0"
; Present remocon input time check
MODCHK
        LD      A,STATE
        CPSE    A,#0H
        JPS     DISINT          ;  Not a valid state for leader pulse check
        BTSF    P1.1            ;  Check the INT1 pin
        JPS     BTCLR
        LD      HL,#LECNT       ;  Increment LECNT
        INCS    @HL
        NOP
        CPSE    @HL,#02H        ;  Check in 4-ms intervals
        JPS     IBF
        LD      A,#1H
        LD      STATE,A         ;  STATE ← 1H
        LD      A,#0H
        LD      IMOD1,A         ;  Select INT1 rising edge trigger
        LD      A,#0DH          ;  Set basic interval time to 7.82 ms
        LD      BMOD,A
        BITS    IE1

BTCLR
        LD      A,#0H
        LD      LECNT,A         ;  LECNT ← 0H
        JPS     IBF

DISINT
        BITR    IE0
        LD      A,#0H
        LD      STATE,A         ;  STATE ← 0H
        LD      A,#0FH
        LD      BMOD,A          ;  Set basic interval time to 1.95 ms
        JR      BTCLR
```

Source Code for Remote Controller Application (Cont.)

```
INCRIT
        LD      EA,CPCNT
        ADS     EA,#01H         ; CPCNT ← CPCNT + 1
        LD      CPCNT,EA
        JPS     MODCHK
IBF
        POP     SB
        IRET

        ORG     0300H

        JPS     LEADCK          ; Jump to routine that checks Low time of leader pulse
        JPS     LEADER          ; Jump to routine that checks High time of leader pulse
        JPS     DATAIN          ; Jump to the data code check routine

; INT1 processing:

INT1
        DI
        PUSH    EA              ; Push EA, HL, WX, YZ, and SB values onto stack
        PUSH    HL
        PUSH    WX
        PUSH    YZ
        PUSH    SB

; 100-µs DELAY:

        LD      EA,#15H

LOP
        DECS    EA
        JR      LOP

; Multiple branch processing:

        LD      A,STATE
        LD      E,#0H
        ADS     EA,#0FFH
        NOP
        LD      WX,EA
        ADS     WX,EA           ; WX ← (STATE – 1) × 2

; Read BCNT:

LOOP1
        LD      EA,BCNT
        LD      YZ,EA
        LD      EA,BCNT
        CPSE    EA,YZ
        JR      LOOP1
        JR      @WX             ; Multi-level branch
```


SAMSUNG ELECTRONICS

Source Code for Remote Controller Application (Cont.)

```
LEADCK
          BTST      P1.1                  ;  Check the Low level time of the leader pulse
          JPS       RET1
          BITS      BMOD.3                ;  Restart the basic timer
          ADS       EA,#0F0H              ;  EA > #10H
          JPS       STATE0
          ADS       EA,#0DEH              ;  EA < #0100 – #0DE + #1F = #41H
                                          ;  (0.5 ms < BCNT < 2 ms)
          JPS       STATE1                ;  Yes

STATE0
          LD        A,#0H                 ;  STATE ← 0
          LD        STATE,A
          LD        A,#0FH
          LD        BMOD,A                ;  Set basic timer interval to 1.95 ms
          BITR      IE1
          LD        EA,#0
          LD        CPCNT,EA              ;  Clear CPCNT

RET1
          POP       SB
          POP       YZ
          POP       WX
          POP       HL
          POP       EA
          EI
          IRET

STATE1
          LD        A,#1H                 ;  Select INT1 falling edge trigger
          LD        IMOD1,A
          INCS      STATE                 ;  STATE ← 2
          JPS       RET1

; Leader pulse check routine:

LEADER
          BTSF      P1.1                  ;  Check INT1 pin Low level
          JPS       RET1
          BITS      BMOD.3                ;  Restart the basic timer
          ADS       EA,#7EH               ;  EA > 83H
          JPS       STATE0
          ADS       EA,#9DH               ;  EA < #100 – #9D + 41 = 0A4H
          JPS       STATE3                ;  Yes
          JPS       STATE0

STATE3
          INCS      STATE                 ;  STATE ← 3
          LD        HL,#49H
          LD        A,#8H                 ;  # (VALID – 1) address MSB = "1"
```

Source Code for Remote Controller Application (Cont.)

```
REMO1
```

```
            LD          A,#0H           ;   Clear all other bits
            DECS        HL
            LD          @HL,A
            CPSE        L,#4H
            JR          REMO1
            JPS         RET1

; Data input subroutine

DATAIN
            BTSF        P1.1            ;   Check P1.1
            JPS         RET1
            BITS        BMOD.3          ;   Start the basic timer
            ADS         EA,#0F0H        ;   EA > 10
            JPS         STATE0
            ADS         EA,#0BEH        ;   EA < 100 – 0BE + 10  =  2
            JR          DAT0            ;   Yes, 0.5 ms < EA < 2.5 ms
            JPS         STATE0

DAT0
            SCF
            ADS         EA,#18H         ;   EA > 0FF – 18 – 0BE – 0F0 = 3AH
            RCF                         ;   0.5 ms < EA < 1.8 ms data  = “0”
            LD          HL,#49H         ;   1.8 ms < EA < 2.5 ms data  =  “1”

DATABR
            DECS        HL              ;   Rotate remocon receive data one bit toward LSB
            LD          A,@HL
            RRC         A
            XCH         A,@HL
            CPSE        L,#4H
            JR          DATABR
            BTST        C               ;   Check for 20-bit data input completion
            JPS         RET1            ;   No
```

Source Code for Remote Controller Application (Cont.)

```
; Custom code check routine:

        LD      EA,REDATA           ; Yes, 20-bit data input complete
        LD      HL,#CUSTOMA         ; Custom code = read custom data?
        CPSE    EA,HL
        JPS     STATE0
        LD      A,REDATA + 2
        LD      E,#CUSTOMB
        CPSE    A,E
        JPS     STATE0
        BTST    VALFG               ; Check VALFG
        JR      FIRCHK
        LD      HL,#(REDATA+3)      ; Read data code = valid data code
        LD      EA,VALID
        CPSE    EA,@HL
        JPS     STCHK
        BITS    CPFG                ; Repeat key is pressed
        JPS     STATE0

FIRCHK
        BTST    FIRFG               ; Check FIRFG
        JPS     STCHK
        BITR    FIRFG
        LD      HL,#(REDATA+3)      ; Read data code = codeBUF data
        LD      EA,codeBUF
        CPSE    EA,@HL
        JPS     STCHK
        LD      EA,codeBUF          ; VALID ← codeBUF
        LD      VALID,EA
        BITR    CPFG
        BITS    VALFG
        JPS     STATE0

STCHK
        BITS    FIRFG               ; codeBUF ← read data code
        LD      EA,REDATA+3
        LD      codeBUF,EA
        JPS     STATE0

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

USING THE BASIC TIMER TO MEASURE PULSE WIDTH

Function Description

The following routine uses the basic timer to measure the width of the High-level pulse that is input at the INT4 interrupt pin (with both rising and falling edge detection). In this case, the pulse width does not exceed the basic timer counter (BCNT) value, which is at least 7.8 ms.

Source Code for Pulse Width Measurement Routine

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

          BUFF      EQU                 30H
          BUFF      EQU                 32.OH

; INT4 processing:

LOOP
          LD        EA,BCNT             ;  EMB = “0”
          LD        YZ,EA
          LD        EA,BCNT
          CPSE      EA,YZ
          JR        LOOP
          BTST      P1.3                ;  P1.3 = “1”?
          JR        AA                  ;  No
          LD        BUFF,EA             ;  Store the BCNT value
          BITR      FLAG                ;  Clear the data present flag
          IRET

AA
          LD        EA,BUFF
          SBS       YZ,EA
          NOP
          LD        EA,YZ
          LD        BUFF,EA             ;  Store data
          BITS      FLAG                ;  Set the data present flag
          IRET

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