To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

APPLICATION NOTE

RENESAS

Phase-out/Discontinued

RA75X ASSEMBLER PACKAGE

STRUCTURED ASSEMBLER PREPROCESSOR

Document No. EEA-1203
(O. D. No. EEA-603)
Date Published April 1991P
Printed in Japan

APPLICATION NOTE

NEC

Phase-out/Discontinued

RA75X ASSEMBLER PACKAGE

STRUCTURED ASSEMBLER PREPROCESSOR

PREFACE

Target:
This Application Note is intended for the user engineers who understand the uPD75516, ST75X structured assembler preprocessor (RA75X assembler package), and macro processor functions and design application systems using them.

Purpose:
The purpose of the Application Note is for the user to understand the program development method using the ST75X structured assembler preprocessor through uPD75516 application program examples.

Composition:
The Application Note contains the following:

. Gernral description
. Structured assembler outline
. Considerations on use of structured assembler
. Application program examples

Use:
The Application Note assumes that the reader has general knowledge of elctricity, logical circuits, and microcomputers. It describes the uPD75516 as a typical device. The description is also applied to the common parts to the 75X series.

Legend:
Data representation weight: High-order and low-order digits are indicated from left to right.

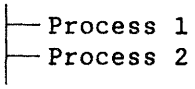
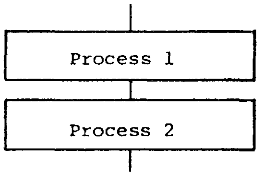
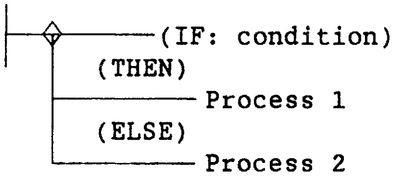
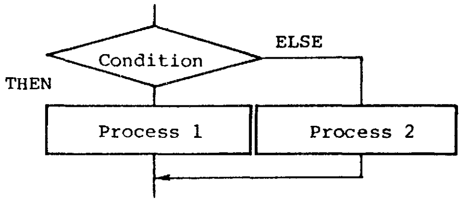
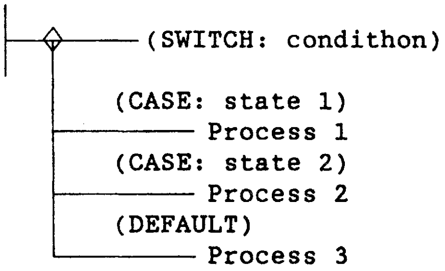
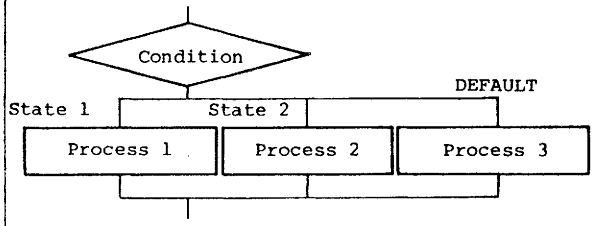
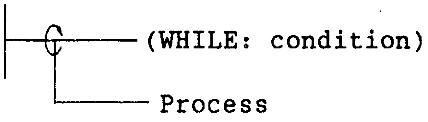
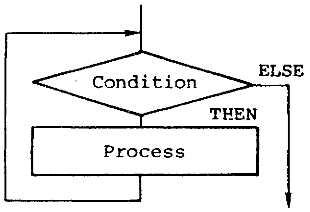
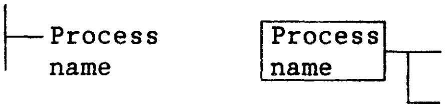
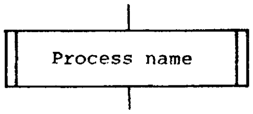
Active low representaition: $\overline{\text{xxx}}$ (pin or signal name is overlined)
Caution: Caution to which you should pay attention
Remarks: Supplementary explanation to the text
Number representation: Binary number xxxx or xxxxB
Decimal number xxxx

Hexadecimal number xxxx or xxxxH

Macro instruction representation: ?xxxxxx (symbol beginning with ?)

SPD: Comparison between SPD representations used in the text and flowcharts is made as shown below:

Caution: This Application Note explains how to develop programs by using the ST75X structured assembler preprocessor and does not gurantee the reliability of the programs contained herein.

	SPD symbols	Flowchart symbols
Serial Processing		
Binary selection	 (IF: condition) (THEN) Process 1 (ELSE) Process 2	
Multiple selection	 (SWITCH: condithon) (CASE: state 1) Process 1 (CASE: state 2) Process 2 (DEFAULT) Process 3	 Condition DEFAULT State 1 State 2 Process 1 Process 2 Process 3
Loop		
Internal call		

Explanation of Package: The items in Explanation of Package are as follows:

<Public declaration symbols>	If external reference declaration of the symbols is made by a user program, the symbols can be referenced within the user program.
<Registers>	Indicates the general purpose registers used by the package.
<RAM area>	Indicates the RAM area used by the package.
<Nesting>	Indicates the nesting level and the maximum size of the stack area used.
<Hardware>	Indicates the peripheral hardware used by the package.
<Interrupts>	Indicates the interrupts used by the package.
<Initialization>	Indicates initialization required to operate the package. Unless otherwise noted, the programs described in the Application Note are assumed to operate with SCC=0 and PCC=3.
<Start method>	Indicates start method of package.

Relevant Documents

Product name	Document name	Document No.	Abbreviation in Application Note
RA75X assembler pakage	User's manual (language)	EEM-747	Language
	User's manual (operation) based on MS_DOS^{TM}	EEM-745	Language
	ST75X user's manual	EEU-642	ST75X UM
Macro processor	User's manual	EEM-722	Macro UM
uPD75516	Pamphet Data sheet User's manual Instruction use table Application Note (I) basic	IB-5051 IC-7580 IEM-5049 IEM-5036 IEM-5104	

CONTENTS

CONTENTS OF FIGURES

CONTENTS OF TABLES

CHAPTER 1 GERNERAL DESCRIPTION

When larger memory space can be used as microprocessor performance improves, various requests are also made for built-in programs.

Consequently, programs become large-scaled and complicated; the number of steps required for program development continues to increase. Program development in the conventional assembly language requires a great number of steps as compared wite the high-level languages. In addition, it is difficult to develop; programs by a group, reuse once prepared programs, etc. For these reasons, built-in program development is oriented for the high-level languages gradually.

However, a compiler is bad in object efficiency as compared with assembler, and is not practical in the 4-bit world. If a high-level language debugger does not exist, it is difficult to debug programs and peripheral function handling instructions that single-chip microcomputers feature cannot be described. If language specifications to enable peripheral function handling instruction description are adopted, easy portability, one of high-level language features must be sacrificed.

Then, "structured assembly language" which lies between the assembly language and high-level language has been proposed.

The Application Note is prepared for you to develop programs using the Structured Assembler Preprocessor (ST75X) for uCOM-75X family development attached to the RA75X assember package. It explains the program development method in the structured assembly language by taking programs using the uPD75516 as examples.

The uPD75516 is a 4-bit single chip microcomputer which has features of internal A/D converter, high-speed processing, and a large number of I/O lines adopting the uCOM-75X family architecture.

The uPD75516 has the following features:

- 16K-byte ROM and 512 x 4-bit RAM
- 8 x 4-bit x 4-bank general purpose registers
- High-speed operation: Minimum instruction execution time 0.95 us (during 4.19-MHz operation)
- 64 I/O lines
- Internal A/D converter
- Internal timer pulse generator
- Four channels of timers
- Nine interrupt sources
- Efficient instruction set which enables 1-,4-, and 8-bit data handling.
- Very low power watch opration in standby mode (subsystem clock operation)

CHAPTER 2 STRUCTURED ASSEMBLER PREPROCESSOR OUTLINE

2.1 What is Structrued Assembler?

The structured assembly language is used for structured programming using control statements such as if and for.

The structured assembly language has the following three features:

(1) Program can be written easily.

  . Label name for a branch need not be considered

  . Transfer instruction which is very descriptive can be described with symbols.

(2) Program can be read easily.

  . Program structure is cleared.

  . Operation or transfer between memory and register can be described in one statement

  . Programs written by other persons are read easlily.

  . Program maintenance (modification) is facilitated.

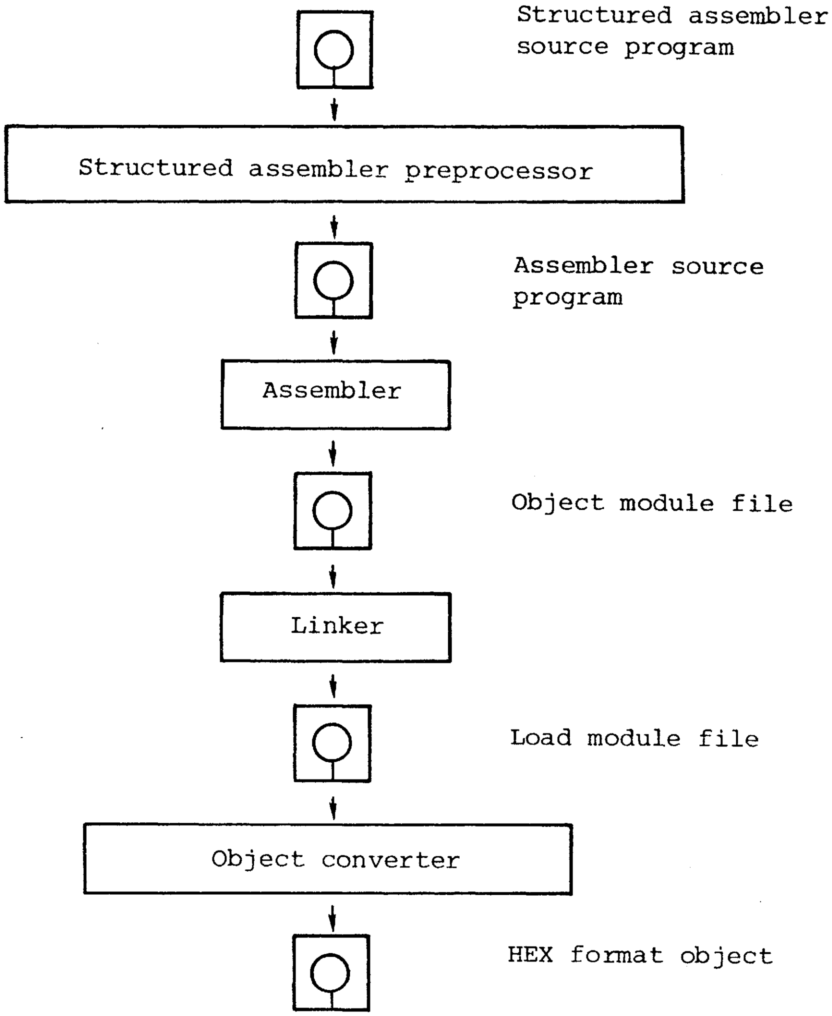
(3) Easy desk debug.

2.2 ST75X Outline

One of the ST75X unique features is the preprocessor format which enables program size optimization processing.

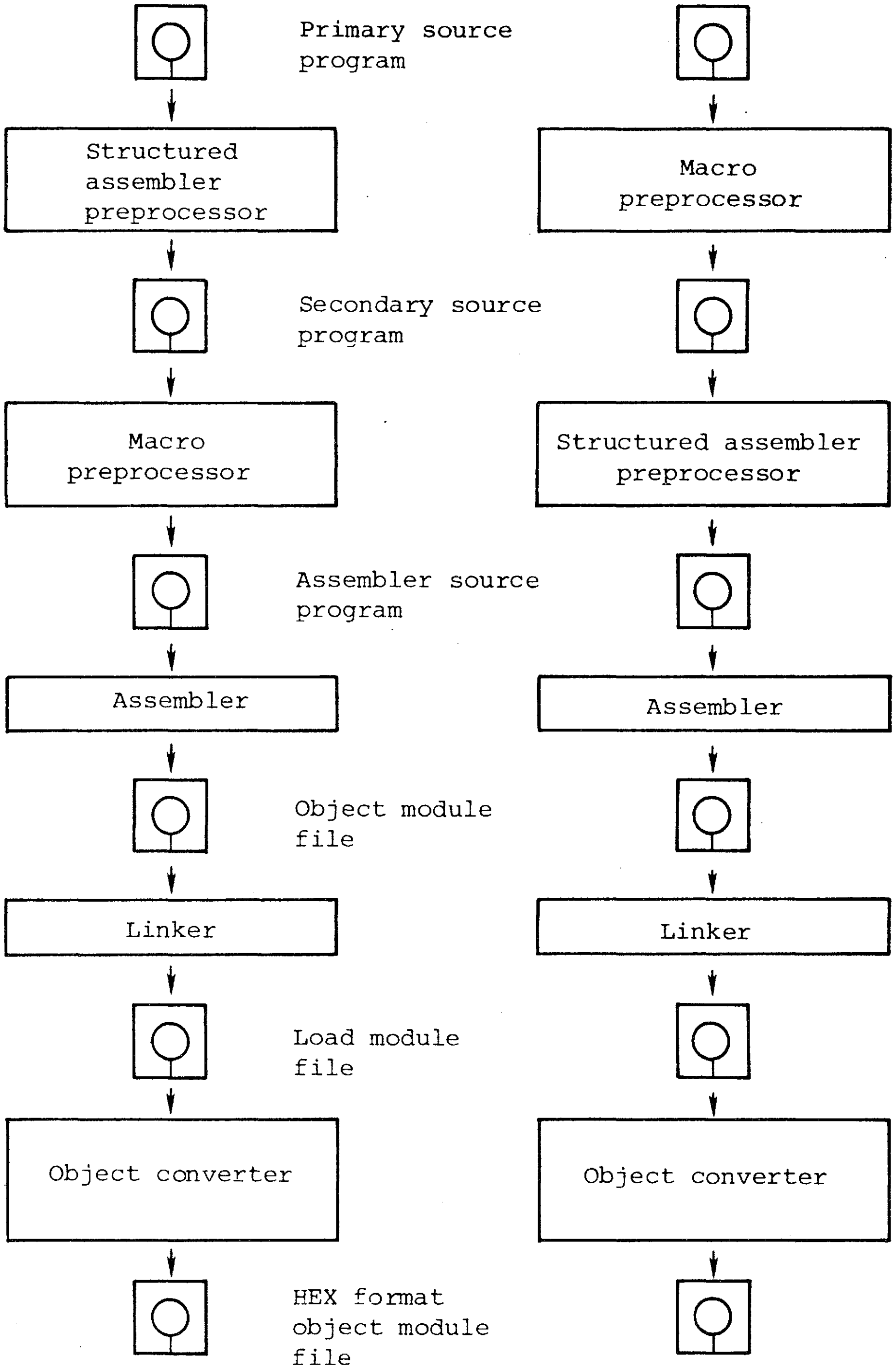
2.2.1 ST75X Processing flow

Structured assembler processing flow is shown below:



The processing flows when the structured assembler and macros are used at the same time are shown below:

① Structured assembler →macro ② Macro →structured assembler



Execute the programs in the Application Note in flow ① .

2.2.2 Control Statements

Table 2-1 lists the control statements that can be used with ST75X. (For details, see ST75X UM Chapter 3.)

Table 2-1 Structured Assembler Control Statements

	Control statement	

if statement	if
	[elseif]
	[else]
	endif
		if_bit
		[elseif_bit]
		[else]
		endif

switch statement	swich
	case
	[default]
	ends	

for statement	for next	

while statement	while
	endw
		while_bit
		endw

until statement	repeat
	until
		repeat
		until_bit

goto statement	goto	
Other statements	break	
	continue	

2.2.3 Operators

Table 2-2 lists the operators that can be used with ST75X. (For details, see ST75X UM 2.4.)

Table 2-2 Structured Assembler Operators

Operator type	Operator
Assignment	= , += , -= , &= . |= , ^=
Increment and decrement	++ , --
Exchange	< - >
Comparison	== , != , < , > , >= , <=
Logical Relational	&& , ||

2.2.4 Pseudo Instructions

Table 2.3 lists the pseudo instructions that can be used with ST75X. (For details, see ST75X UM Chapter 4.)

Table 2-3 Structured Assembler Pseude Instructions

Pseudo instruction type	Pseudo_instruction
Identifier definition	#define
Conditional processing	#ifdef #else #endif
Include	#include
GETI replacement	#defgeti #endgeti

CHAPTER 3 CONSIDERATIONS ON USE OF STRUCTURED ASSEMBLER

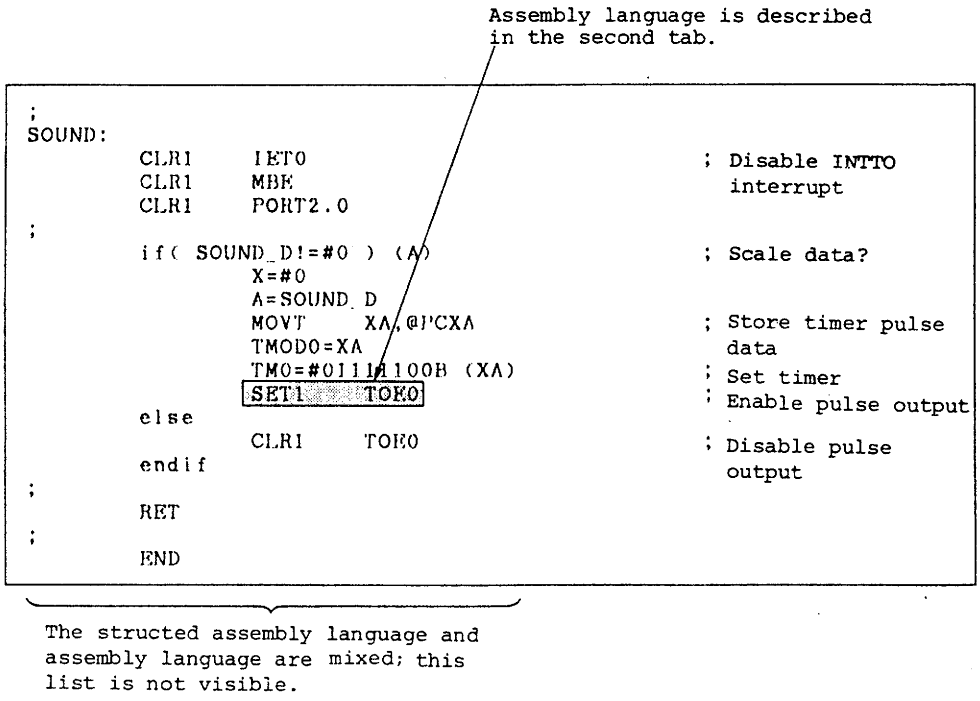
3.1 Considerations on Program Description

If a structured assembler program is described by considering the description positions so that an assembly list used in machine debug is made visible, a visible assembly list is prepared and the debug effeciency is improved.

If a program is described in deep nesting without considering the structured assembler description position, structured statement and assembler mnemonics and comment statements are output out of position on the assembly list, as shown in Example 1.

Example 1: Illegible assembly list example

. Source list



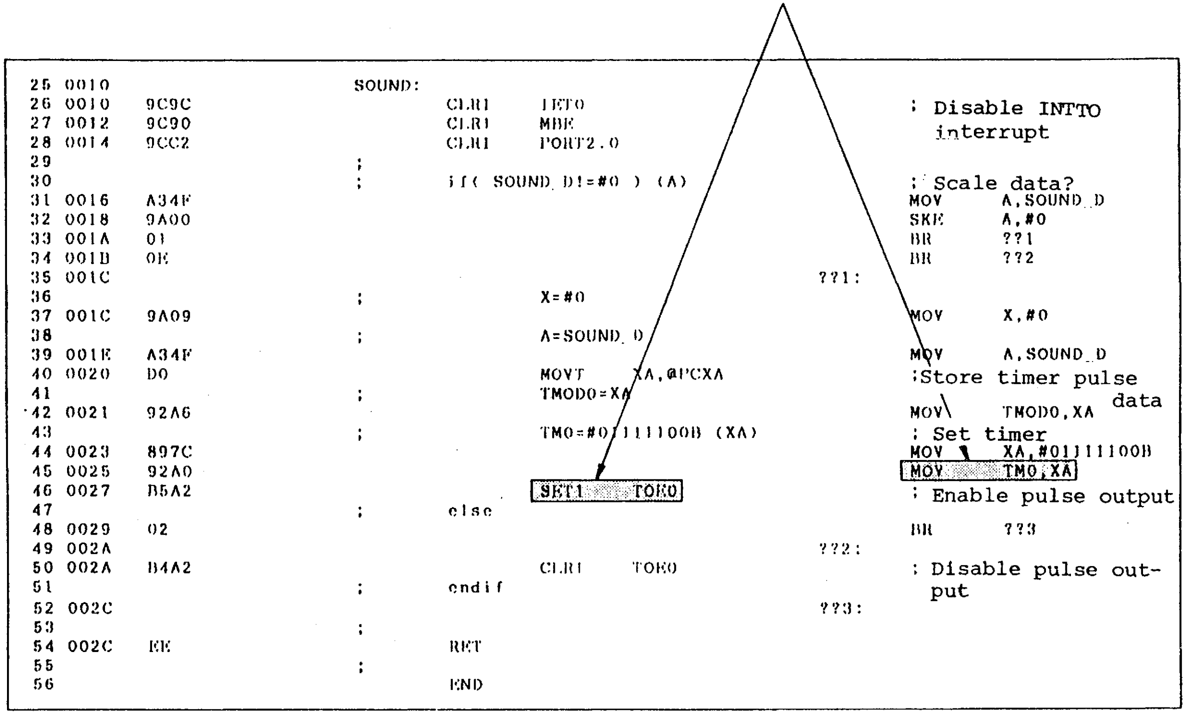
Assembly language is described in the second tab.

```
;
SOUND:
         CLR1      IET0                                     ; Disable INTTO
         CLR1      MBE                                        interrupt
         CLR1      PORT2.0
;
         if( SOUND_D!=#0 ) (A)                              ; Scale data?
                   X=#0
                   A=SOUND_D
                   MOVT      XA,@PCXA                       ; Store timer pulse
                   TMOD0=XA                                   data
                   TM0=#01111100B (XA)                      ; Set timer
                   SET1      TOE0                           ; Enable pulse output
         else
                   CLR1      TOE0                           ; Disable pulse
         endif                                                output
;
         RET
;
         END
```

The structed assembly language and assembly language are mixed; this list is not visible.

. **Assembly list**

Assembler mnemonics are output at different positions (not visible).



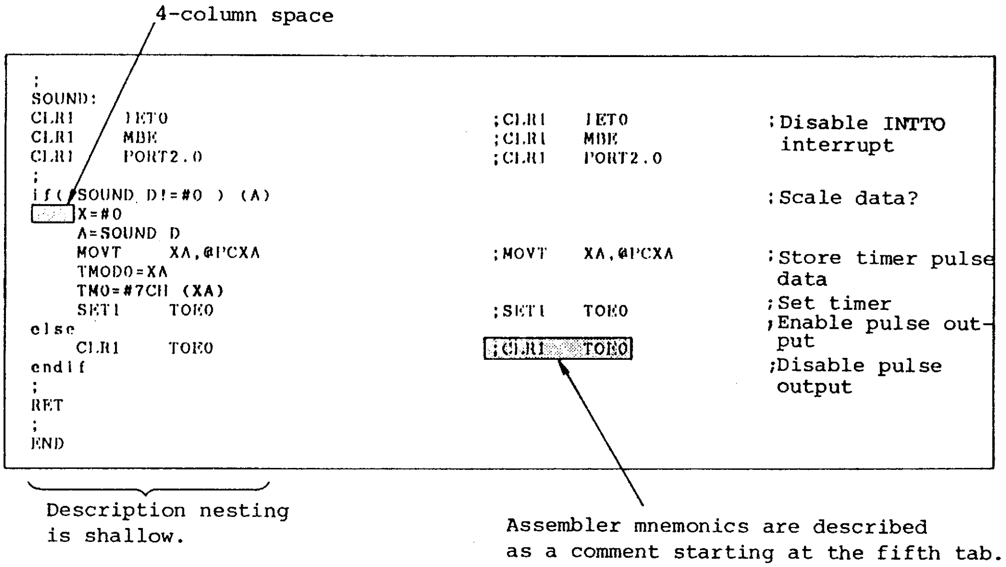
```
25 0010                      SOUND:
26 0010   9C9C                      CLR1      IETO                           ; Disable INTTO
                                                                                interrupt
27 0012   9C90                      CLR1      MBE
28 0014   9CC2                      CLR1      PORT2.0
29                           ;
30                           ;      if( SOUND_D!=#0 ) (A)                    ; Scale data?
31 0016   A34F                                                               MOV     A,SOUND_D
32 0018   9A00                                                               SKE     A,#0
33 001A   01                                                                 BR      ??1
34 001B   0E                                                                 BR      ??2
35 001C                                                               ??1:
36                           ;              X=#0
37 001C   9A09                                                               MOV     X,#0
38                           ;              A=SOUND_D
39 001E   A34F                                                               MOV     A,SOUND_D
40 0020   D0                                MOVT      XA,@PCXA               ;Store timer pulse
41                           ;              TMOD0=XA                                          data
42 0021   92A6                                                               MOV     TMOD0,XA
43                           ;              TM0=#01111100B (XA)              ; Set timer
44 0023   897C                                                               MOV     XA,#01111100B
45 0025   92A0                                                               MOV     TM0,XA
46 0027   B5A2                              SET1      TOE0                   ; Enable pulse output
47                           ;      else
48 0029   02                                                                 BR      ??3
49 002A                                                               ??2:
50 002A   B4A2                              CLR1      TOE0                   ; Disable pulse out-
51                           ;      endif                                      put
52 002C                                                               ??3:
53                           ;
54 002C   EE                        RET
55                           ;
56                                  END
```

The structued assembly language and assembly language are mixed; this list is not visible.

In contrast, the assembly list of a program described in shallow nesting by considering the description position is made visible as shown in Example 2.

Example 2: Visible assembly list example

. Source list



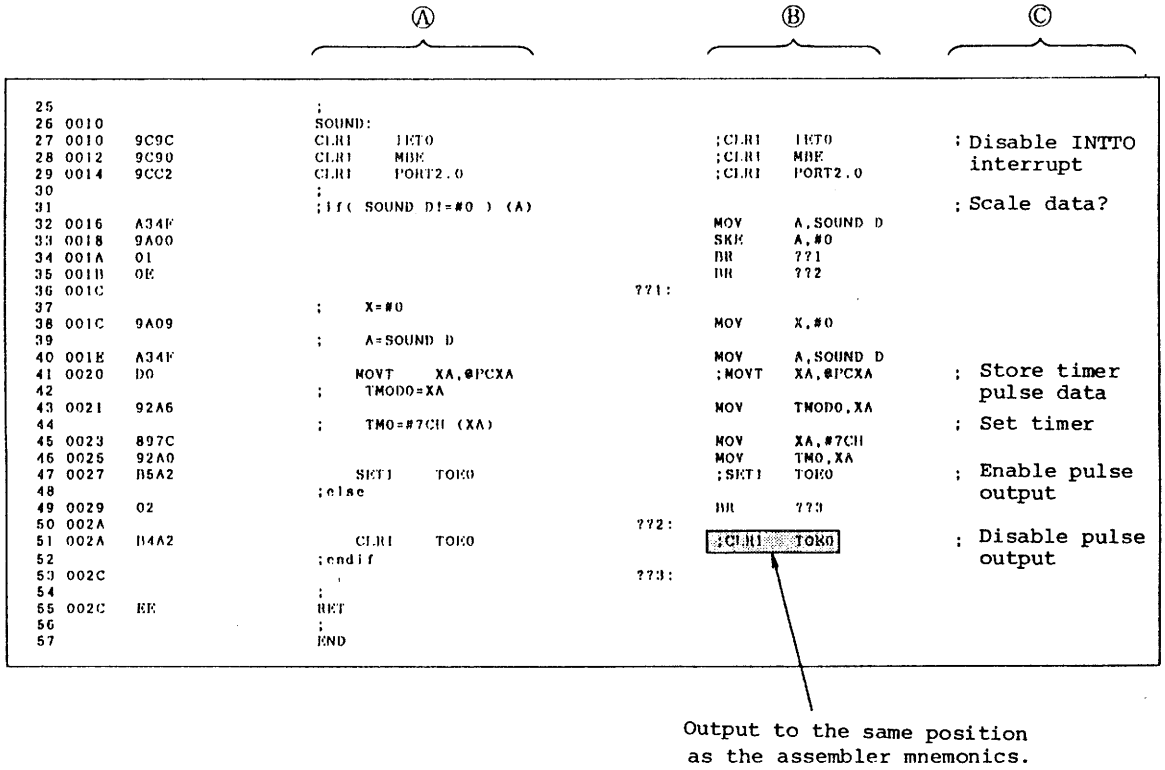
4-column space

```
;
SOUND:
CLR1      IET0                                          ;CLR1     IET0          ;Disable INTTO
CLR1      MBE                                           ;CLR1     MBE            interrupt
CLR1      PORT2.0                                       ;CLR1     PORT2.0
;
if( SOUND D!=#0 ) (A)                                                           ;Scale data?
    X=#0
    A=SOUND D
    MOVT      XA,@PCXA                                  ;MOVT     XA,@PCXA      ;Store timer pulse
    TMOD0=XA                                                                     data
    TMO=#7CH (XA)                                                               ;Set timer
    SET1      TOE0                                      ;SET1     TOE0          ;Enable pulse out-
else                                                                             put
    CLR1      TOE0                                      ;CLR1     TOE0          ;Disable pulse
endif                                                                            output
;
RET
;
END
```

Description nesting is shallow.

Assembler mnemonics are described as a comment starting at the fifth tab.

. Assembly list



```
25                                ;
26 0010                           SOUND:
27 0010    9C9C                   CLR1     IET0                     ;CLR1     IET0          ; Disable INTTO
28 0012    9C90                   CLR1     MBE                      ;CLR1     MBE             interrupt
29 0014    9CC2                   CLR1     PORT2.0                  ;CLR1     PORT2.0
30                                ;
31                                ;if( SOUND D!=#0 ) (A)                                    ; Scale data?
32 0016    A34F                                                      MOV      A,SOUND D
33 0018    9A00                                                      SKE      A,#0
34 001A    01                                                        BR       ??1
35 001B    0E                                                        BR       ??2
36 001C                                                    ??1:
37                                ;     X=#0
38 001C    9A09                                                      MOV      X,#0
39                                ;     A=SOUND D
40 001E    A34F                                                      MOV      A,SOUND D
41 0020    D0                          MOVT     XA,@PCXA             ;MOVT    XA,@PCXA      ;  Store timer
42                                ;     TMOD0=XA                                              pulse data
43 0021    92A6                                                      MOV      TMOD0,XA
44                                ;     TM0=#7CH (XA)                                      ;  Set timer
45 0023    897C                                                      MOV      XA,#7CH
46 0025    92A0                                                      MOV      TM0,XA
47 0027    B5A2                        SET1     TOE0                 ;SET1    TOE0          ;  Enable pulse
48                                ;else                                                       output
49 0029    02                                                        BR       ??3
50 002A                                                    ??2:
51 002A    B4A2                        CLR1     TOE0                 ;CLR1    TOE0          ;  Disable pulse
52                                ;endif                                                      output
53 002C                                                    ??3:
54                                ;
55 002C    EE                     RET
56                                ;
57                                END
```

Output to the same position as the assembler mnemonics.

On the assembly list, tab count is specified as an option in conversion by the structured assembler preprocessor (-WT4, 5, 6).

The ST75X start method is as follows:

<Start method>

```
A > ST75X file name -WT4, 5, 6
```

Structured statements as comment statements, assembler mnemonics (containing mnemonics as comment statements), and comment statements are output to positions A , B , and C of the assembly list respectively. The assembly list becomes visible.

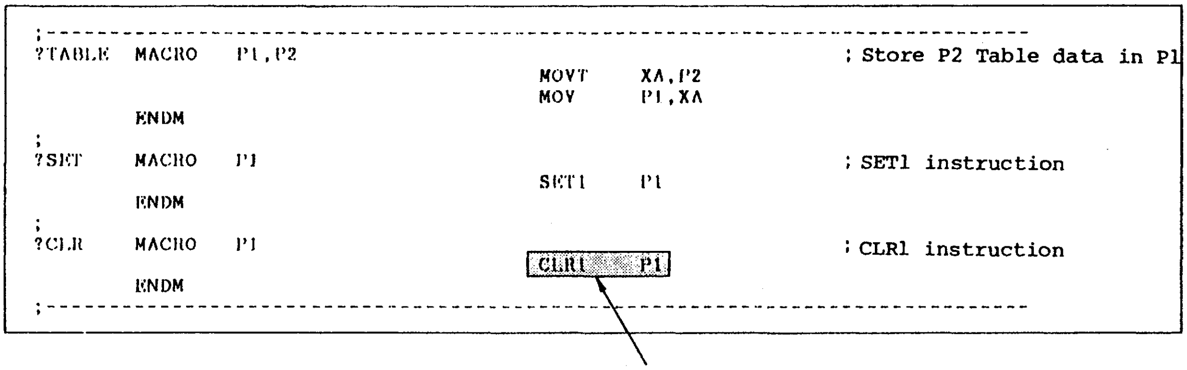
3.2 Structured Assembler and Macro Instructions

As described above, if a program is described by considering the description positions, its assembly list becomes visible. It can be furthermore made visible by using macro instructions.

An example of adding macro instruction to the program shown above is given below:

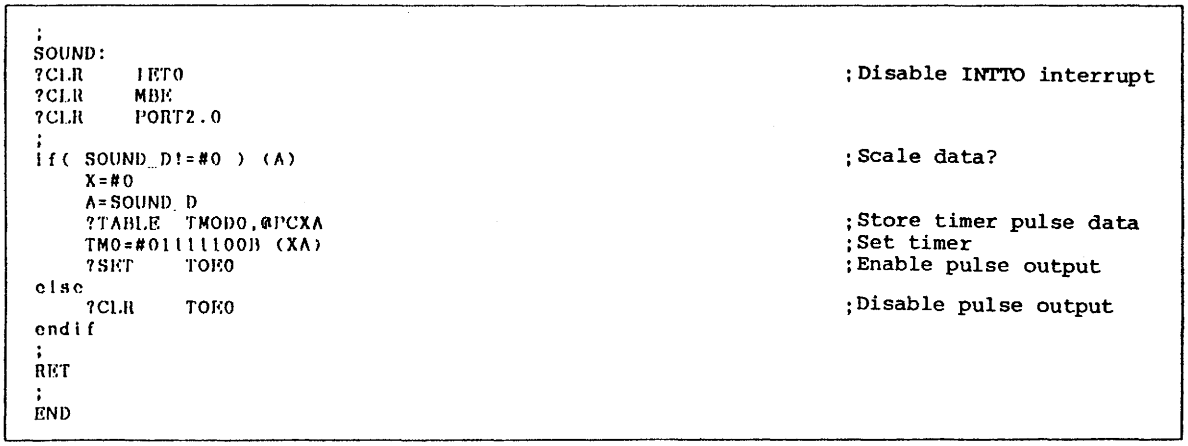
Example 3:

. Source list



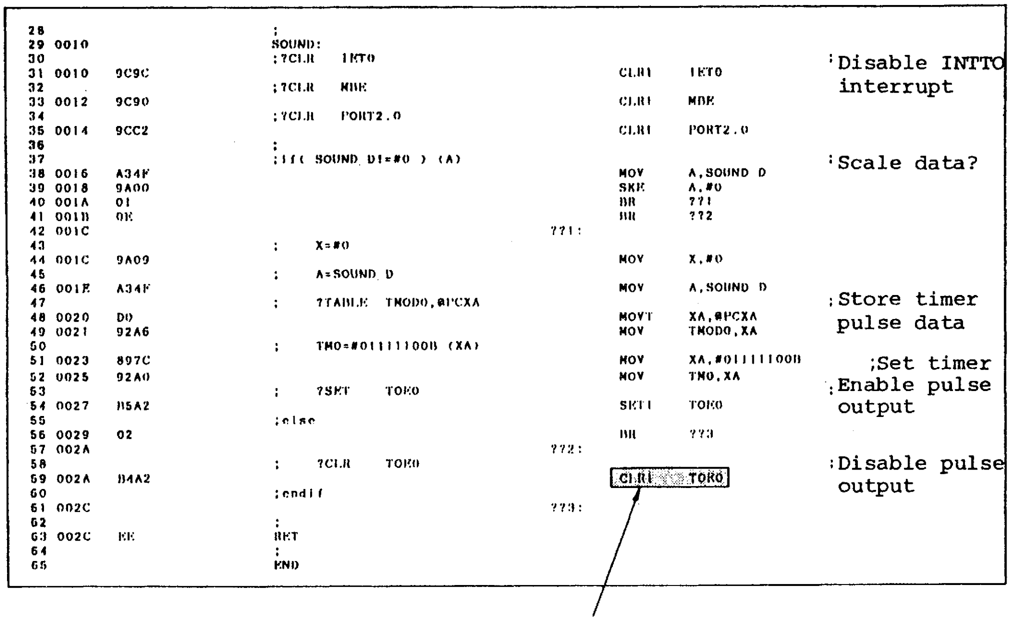
```
;------------------------------------------------------------------------------
?TABLE  MACRO   P1,P2                                         ; Store P2 Table data in P1
                                MOVT    XA,P2
                                MOV     P1,XA
        ENDM
;
?SET    MACRO   P1                                            ; SET1 instruction
                                SET1    P1
        ENDM
;
?CLR    MACRO   P1                                            ; CLR1 instruction
                                CLR1    P1
        ENDM
;------------------------------------------------------------------------------
```

Description with 5-tab space



```
;
SOUND:
?CLR    IET0                                                  ;Disable INTT0 interrupt
?CLR    MBE
?CLR    PORT2.0
;
if( SOUND_D!=#0 ) (A)                                         ;Scale data?
    X=#0
    A=SOUND_D
    ?TABLE  TMOD0,@PCXA                                       ;Store timer pulse data
    TM0=#01111100B (XA)                                       ;Set timer
    ?SET    TOE0                                              ;Enable pulse output
else
    ?CLR    TOE0                                              ;Disable pulse output
endif
;
RET
;
END
```

. Assembly list



```
                    ;
0010                SOUND:
                    ;?CLR    IET0
0010    9C9C                                    CLR1    IET0          ;Disable INTT0
                    ;?CLR    MBE                                       interrupt
0012    9C90                                    CLR1    MBE
                    ;?CLR    PORT2.0
0014    9CC2                                    CLR1    PORT2.0
                    ;
                    ;if( SOUND_D!=#0 ) (A)                            ;Scale data?
0016    A34F                                    MOV     A,SOUND_D
0018    9A00                                    SKE     A,#0
001A    01                                      BR      ??1
001B    0E                                      BR      ??2
001C                                    ??1:
                    ;    X=#0
001C    9A09                                    MOV     X,#0
                    ;    A=SOUND_D
001E    A34F                                    MOV     A,SOUND_D
                    ;    ?TABLE  TMOD0,@PCXA                          ;Store timer
0020    D0                                      MOVT    XA,@PCXA       pulse data
0021    92A6                                    MOV     TMOD0,XA
                    ;    TM0=#01111100B (XA)
0023    897C                                    MOV     XA,#01111100B     ;Set timer
0025    92A0                                    MOV     TM0,XA        ;Enable pulse
                    ;    ?SET    TOE0                                  output
0027    B5A2                                    SET1    TOE0
                    ;else
0029    02                                      BR      ??3
002A                                    ??2:
                    ;    ?CLR    TOE0                                 ;Disable pulse
002A    B4A2                                    CLR1    TOE0           output
                    ;endif
002C                                    ??3:
                    ;
002C    EE          RET
                    ;
                    END
```

The assembler mnemonics expanded by the macro processor are output to this position.

On the assembly list, the tab count is specified as an option in conversion by the structured assembler preprocessor (-WT4, 5, 6).

To macro instructions, assemble in the following order:

1. Structured assembler preprocessor
2. Macro processor
3. RA75X assembler

3.3 Use of Switch Statement

The structured assembler switch statement can be used as described below:

(1) When break statement is used

```
switch (condition)
     case 1:
               Process 1
               break
     case 2:
               Process 2
               break
     case 3:
               Process 3
               break
     case 4:
               Process 4
endsw
```

If the condition is 2, process 2 only is executed and the switch statement is exited.

(2) When break statement is not used

```
switch (condition)
     case 1:
               Process 1
               A = #2
     case 2:
               Process 2
               A = #3
     case 3:
               Process 3
               break
     case 4:
               Process 4
endsw
```

In this example, processing is performed depending on the condition as follows:

. Condition = 1: Process 1 → process 2 → process 3

. Condition = 2: Process 2 → process 3

. Condition = 3: Process 3

. Condition = 4: Process 4

A value is set in the A register becuase the A register value for actual condition decision at the assembler mnemonic level must be changed to the next condition.

Thus, the programming efficiency can be improved by using the switch statement.

3.4 Use of while Statment

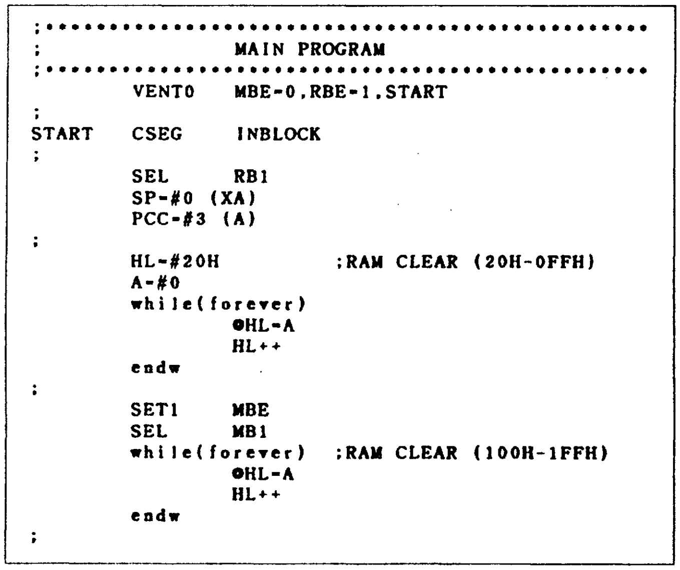
If forever is used for the structured assembler while statement condition and the while statement is existed by a skip instruction, the programming efficiency can be improved.

```
while (forever)
      Process 1
      HL++

endw
```

In this example, when process 1 and HL register increment are repeated and HL register increment generates a carry, the while statement is exited. If the while statement is used in such a manner, the number of branch instructions output by the structured assembler preprocessor is reduced and the programming efficiency is improved.

Example:



```
;**********************************************
;               MAIN PROGRAM
;**********************************************
        VENT0   MBE-0,RBE-1,START
;
START   CSEG    INBLOCK
;
        SEL     RB1
        SP-#0 (XA)
        PCC-#3 (A)
;
        HL-#20H         ;RAM CLEAR (20H-0FFH)
        A-#0
        while(forever)
                @HL-A
                HL++
        endw
;
        SET1    MBE
        SEL     MB1
        while(forever)  ;RAM CLEAR (100H-1FFH)
                @HL-A
                HL++
        endw
;
```

3.5 Program Design Method Using SPD

3.5.1 What is SPD?

SPD is short for Structured Programming Diagram which is a design description technique for structured programming proposed by NEC.

IBM HIPO, HITACHI PAD, NTT HCP, etc., are based on similar concept to SPD.

3.5.2 SPD merits

SPD can be managed as a file with a wordprocessor. Thus, the design can be updated easily.

Next, flowcharts and the SPD program design method are described by taking a 1000-yen note exchange machine as an example.

Now, let's design exchange processing of a 1000-yen note into 10 100-yen coins. Fig. 3-1 shows flowchanrt.

Fig. 3-1 Flowchart 1

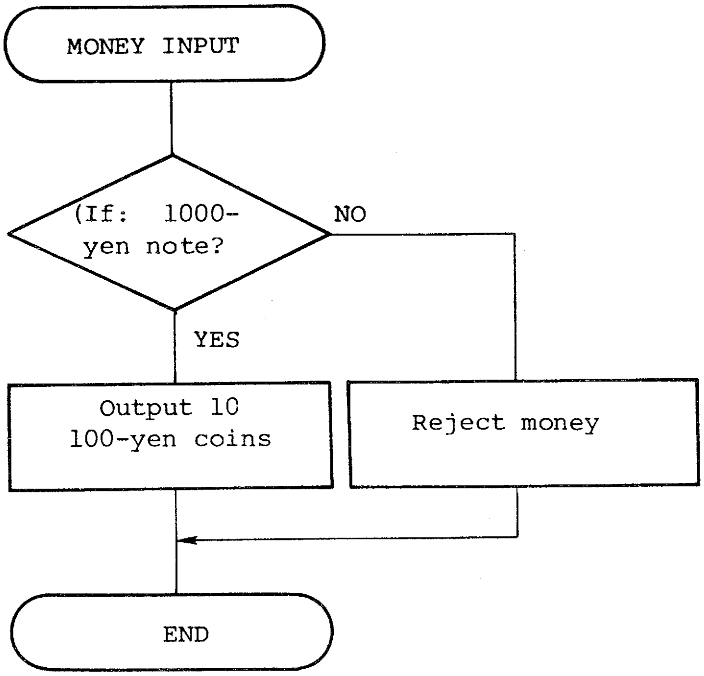


Fig. 3-2 SPD1

```
MONEY INPUT ──◇────── (IF:  1000-yen note?)
              │[THEN]
              ├────── Output 10 100-yen coins
              │[ELSE]
              └────── Reject momey
```

Now, let's add a new function to the exchange machine to enable the user to select by pressing a given button as follows:

- Button 1: 10 100-yen coins
- Button 2: Five 100-yen coins and a 500-yen coin
- Button 3: Two 500-yen coins

Fig. 3-3 and 3-4 show flowchart and SPD of the program design.

Fig. 3-3 Flowchart 2

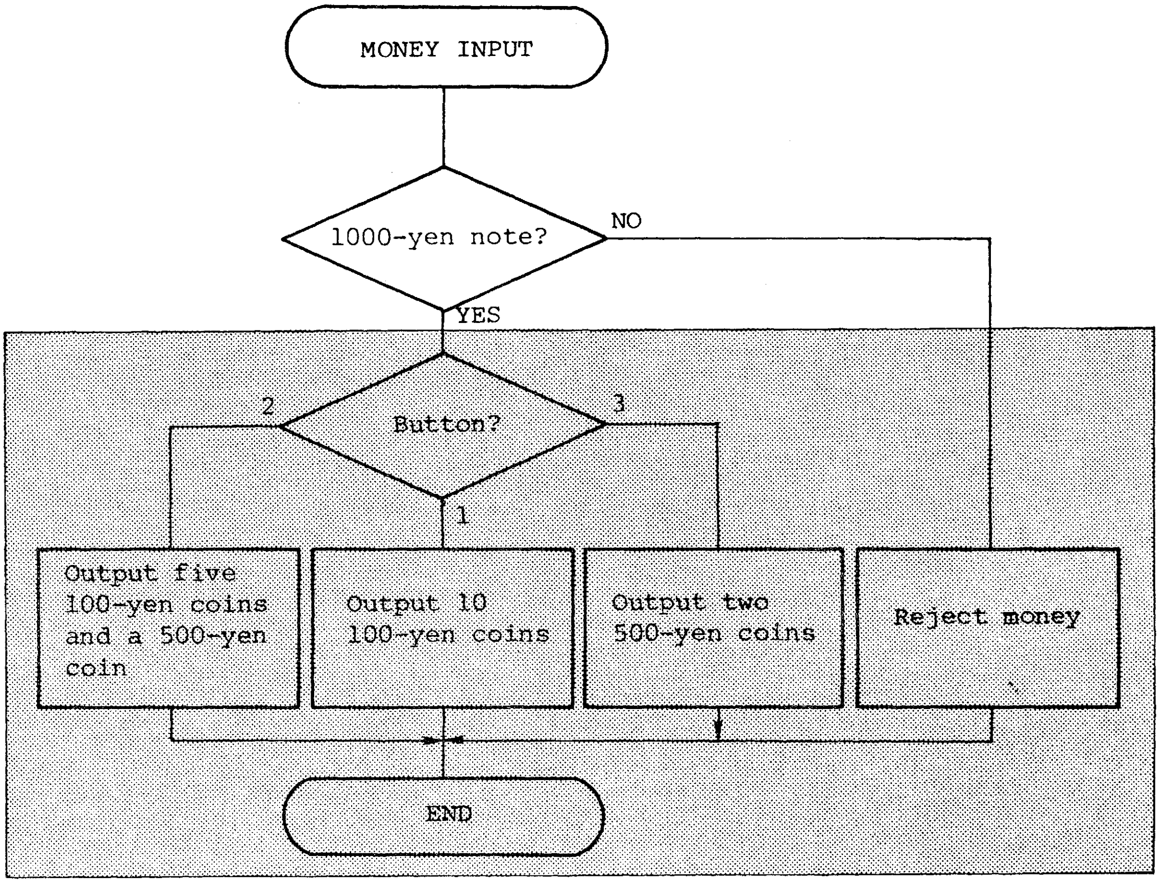
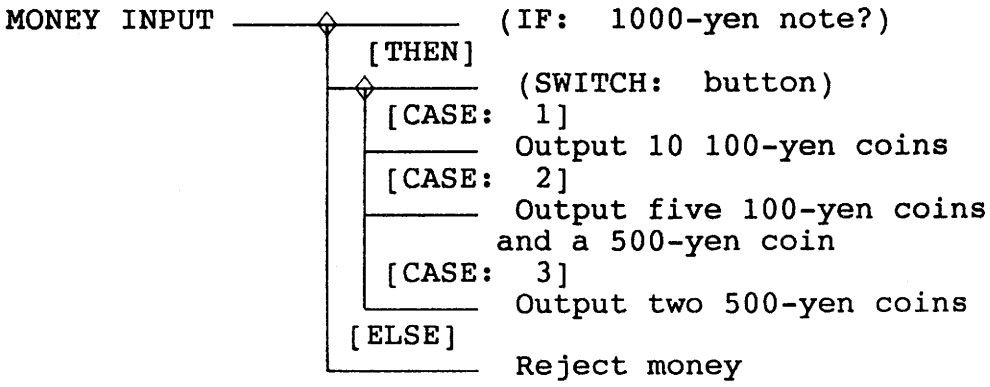


Fig. 3-4 SPD2



```
MONEY INPUT ──◇──────── (IF:  1000-yen note?)
              │ [THEN]
              ├─◇────── (SWITCH:  button)
              │ │ [CASE:  1]
              │ ├────── Output 10 100-yen coins
              │ │ [CASE:  2]
              │ ├────── Output five 100-yen coins
              │ │       and a 500-yen coin
              │ │ [CASE:  3]
              │ └────── Output two 500-yen coins
              │ [ELSE]
              └──────── Reject money
```

In this example, if the program is designed with flowchart, Figs. 3-1 to 3-3 are used. However, since button selection processing is added, the shaded portion of Fig. 3-3 must be all rewritten.

However, when the program is designed with SPD, if Fig. 3-2 is predescribed with a wordprocessor, Fig. 3-4 can be easily obtained by describing only the added function with the wordprocessor insertion function.

Thus, program design which is software design basis can be made easily and precisely by using SPD.

Program design errors and bugs can also be reduced.

3.5.3 SPD and flowchart

Table 3-1 lists the correspondence between SPD description and flowchart description.

Table 3-1 SPD and Flowchart

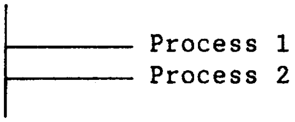
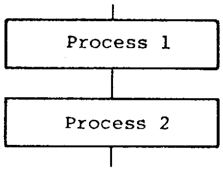
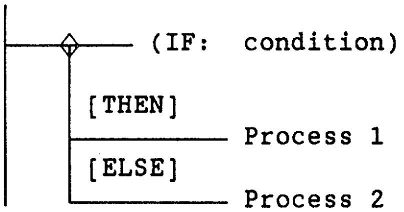
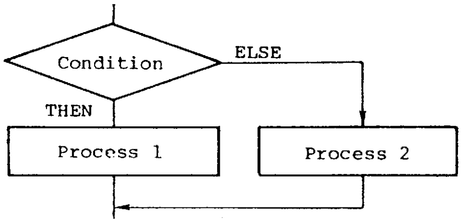
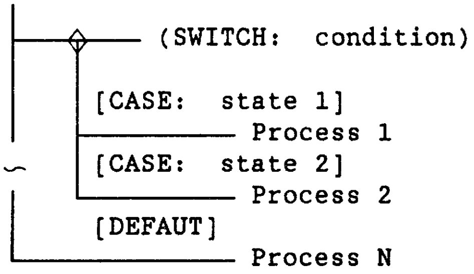
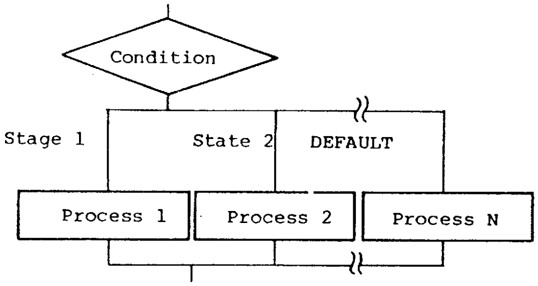
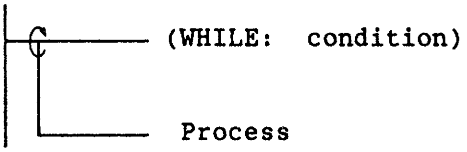
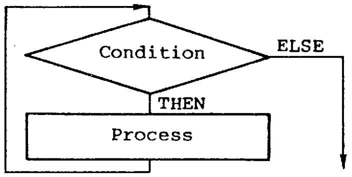
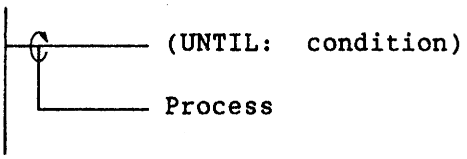
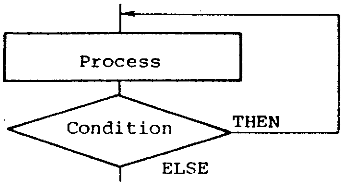
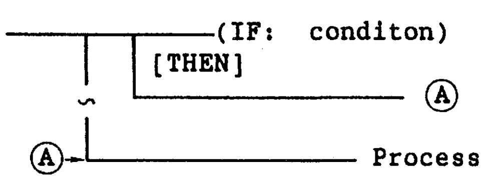
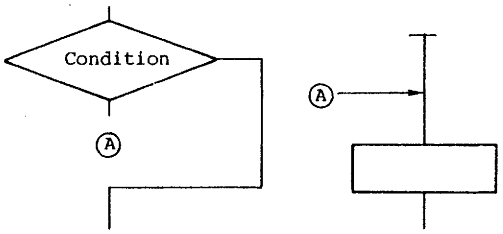
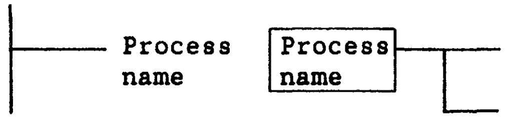
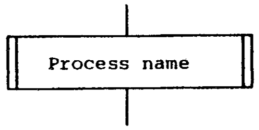
	SPD symbols	Flowchart symbols
Serial processing		
Binary selection	 (IF: condition) [THEN] Process 1 [ELSE] Process 2	
Multiple selection	 (SWITCH: condition) [CASE: state 1] Process 1 [CASE: state 2] Process 2 [DEFAUT] Process N	 Condition Stage 1 State 2 DEFAULT Process 1 Process 2 Process N
Loop		
Loop		

Table 3-1 SPD and Flowchart (Cont'd)

	SPD symbols	Flowchart symbols
Connector		
Internal call		

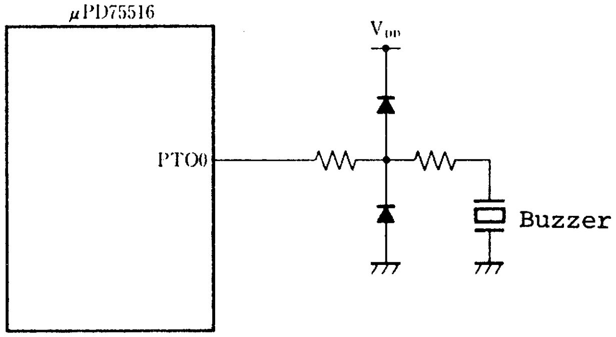
CHAPTER 4 SCALE GENERATION PROGRAM

4.1 Explanation of Program

A program example is given which outputs scale to the external by using the TPO0 pin (T0 (timer/evenet counter 0) output).

4.1.1 Program outline

Fig. 4-1 Scale Generation Diagram



The output frequency from the P20/PTO0 pin is determined by the CP (count pulse) determined by TM0 (timer/event counter 0 mode register) and the value set in the TMOD0 (modulo regiter). Table 4-1 lists the values to be set in TMOD0 for the scale and frequency errors for the scale frequencies when $f_X/2^4$ (f_{CP} = 262 kHz at f_X = 4.194304 MHz) is selected for the CP.

Remarks: f_X: Main system clock frequency
f_{CP}: Count pulse frequency

Table 4-1 Scale Frequencies

Scale		Frequency (Hz)	Value set in modulo register (H)	Output from TPO0 pin	
				Frequency (Hz)	Error (%)
Do	C	523.25	FA	522.20	-0.20
	$C^{\#}$, D^{b}	554.37	EB	555.39	0.18
Re	D	587.33	DE	587.77	0.08
	$D^{\#}$, E^{b}	622.26	D3	618.26	-0.64
Mi	E	659.25	C7	655.36	-0.59
Fa	F	698.46	BC	693.50	-0.71
	$F^{\#}$, G^{b}	739.98	B1	736.36	-0.49
So	G	783.98	A6	780.19	-0.48
	$G^{\#}$, A^{b}	830.61	9E	824.35	-0.75
Ra	A	880	94	879.67	-0.04
	$A^{\#}$, B^{b}	932.33	8C	929.58	-0.29
Si	B	987.77	84	985.50	-0.23
Do	C	1046.5	7C	1048.57	0.20
	$C^{\#}$, D^{b}	1108.74	75	1110.78	0.18
Re	D	1174.66	6F	1170.29	-0.37

When the scale generation program is called as a subroutine, it generates scale corresponding to data in the scale data area (SOUND_D).

When the data is 0, output stops. Table 4-2 lists the scale corresponding to data 1H-FH.

The once output scale continues until the next subroutine call is made.

Table 4-2 Scale Data and Output Scale

Data	0	1	2	3	4	5	6	7
Scale	-	Do	Do$^{\#}$	Re	Re$^{\#}$	Mi	Fa	Fa$^{\#}$
Data	8	9	A	B	C	D	E	F
Scale	So	So$^{\#}$	Ra	Ra$^{\#}$	Si	Do	Do$^{\#}$	Re

4.1.2 Explanation of structured assembler

The program inputs timer pulse data from scale data by using the following data table:

Data table

```
SOUTBL  CSEG  PAGE        ; Timer pulse data table
        DB    0           ; Rest
        DB    0FAH        ; Do
        DB    0EBH        ; Do#
        DB    0DEH        ; Re
        DB    0D3H        ; Re#
        DB    0C7H        ; Mi
        DB    0BCH        ; Fa
        DB    0B1H        ; Fa#
        DB    0A6H        ; So
        DB    9EH         ; So#
        DB    94H         ; La
        DB    8CH         ; La#
        DB    84H         ; Si
        DB    7CH         ; Do
        DB    75H         ; Do#
        DB    6FH         ; Re#
```

Although the switch statement can also be used with the structured assembler, the programming efficiency is not good if the switch statement is used when a large number of data pieces are processed.

Switch statement description example

```
switch (SOUND_D)
        case  0 :                ; Rest
                XA=#0
                break
        case  1 :                ; Do
                XA=#0FAH
                break
              :
        case  E :                ; Do#
                XA=#75H
                break
        case  F :                ; Re#
                XA=#6FH
ends
TMOD0=XA                         ; Store in modulo register
```

4.1.3 Explanation of macro instructions

The following three macro instructions are used:

- . ?TABLE: Table data is read by executing a table look-up instruction and stored in TMOD0.
- . ?SET: Is converted into a SET1 instruction.
- . ?CLR: Is converted into a CLR1 instruction.

Since the assembler mnemonics must be directly described for the SET1 and CLR1 instructions, the ?SET and ?CLR macro instructions are used for description. (See 3.2.)

4.1.4 Explanation of package

<Public declaration symbols>

SOUND_D

<Registers>

. Bank: RBE x RBS . Register: XA

<RAM area>

Address	Name	Use	Initial value
4FH	SOUND_D	Scale data area	——

<Nesting>

One level (4 x 4-bit stack area)

<Hardware>

. Port: Port 2.0 (PTO0 output pin)

. T0 (timer/event counter 0)

<Initialization>

PCC ← 3H

Port 2 ← output mode

<Start method>

Call SOUND.

4.2 SPD

SOUND (subroutine)

```
SOUND
    Disable INTT0 interrupt
    MBE = 0
    Clear PORT2.0
    (IF:  Scale data ≠ 0)
      [THEN]
        Set timer data
        Enable timer pulse output
      [ELSE]
        Disable timer pulse output
    RET
```

4.3 Program Example

SOUND (subroutine)

```
;-------------------------------------------------------------------------
        PUBLIC  SOUND
;-------------------------------------------------------------------------

?TABLE  MACRO   P1,P2                                         ; Store P2 table data in P1
                                        MOVT    XA,P2
                                        MOV     P1,XA
        ENDM

?SET    MACRO   P1                                            ;SET1 instruction
                                        SET1    P1
        ENDM

?CLR    MACRO   P1                                            ;CLR1 instruction
                                        CLR1    P1
        ENDM

;-------------------------------------------------------------------------

SOUND_D DSEG    0 AT 4FH                                      ;Scale data area
        DS      1H

SOUTBL  CSEG    PAGE                                          ;Timer pulse data table
        DB      0                                             ;Rest
        DB      0FAH                                          ;Do
        DB      0EBH                                          ;Do#
        DB      0DEH                                          ;Re
        DB      0D3H                                          ;Re#
        DB      0C7H                                          ;Mi
        DB      0BCH                                          ;Fa
        DB      0B1H                                          ;Fa#
        DB      0A6H                                          ;So
        DB      9EH                                           ;So#
        DB      94H                                           ;La
        DB      8CH                                           ;La#
        DB      84H                                           ;Si
        DB      7CH                                           ;Do
        DB      75H                                           ;Do#
        DB      6FH                                           ;Re

SOUND:
?CLR    IETO                                                  ;Disable INTTO interrupt
?CLR    MBE
?CLR    PORT2.0

if( SOUND_D1=#0 ) (A)                                         ;Scale data?
    X=#0
    A=SOUND_D
    ?TABLE  TMOD0,@PCXA                                       ;Store timer pulse data
    TM0=#01111100B (XA)                                       ;Set timer
    ?SET    TOE0                                              ;Enable pulse output
else
    ?CLR    TOE0                                              ;Disable pulse output
endif

RET

END
```

Assembly list is given in A.1.

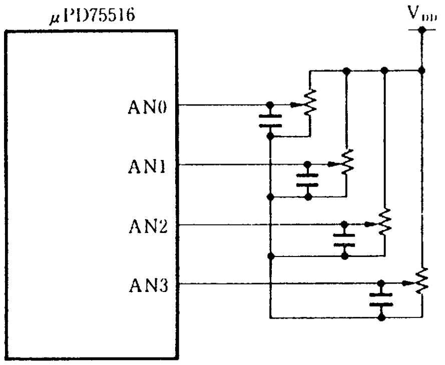
CHAPTER 5 A/D CONVERSION PROGRAM

5.1 Explanation of Program

A program example is given which stores the conversion result in a data area by using the uPD75516 8-bit precision A/D converter which has eight analog input channels.

5.1.1 Program outline

Fig. 5-1 A/D Conversion Diagram



The analog input signal to be converted into digital from is selected by setting ADM (A/D conversion mode register) bits 6 to 4.

The example program uses AN0-AN3 (analog channels 0-3). Analog channel is selected by setting data in the analog channel pointer (ACHN_P).

Table 5-1 Lists the Correspondence between ACHN_P data and the analog channels.

Table 5-1 Correspondence between ACHN_P and Analog Channels

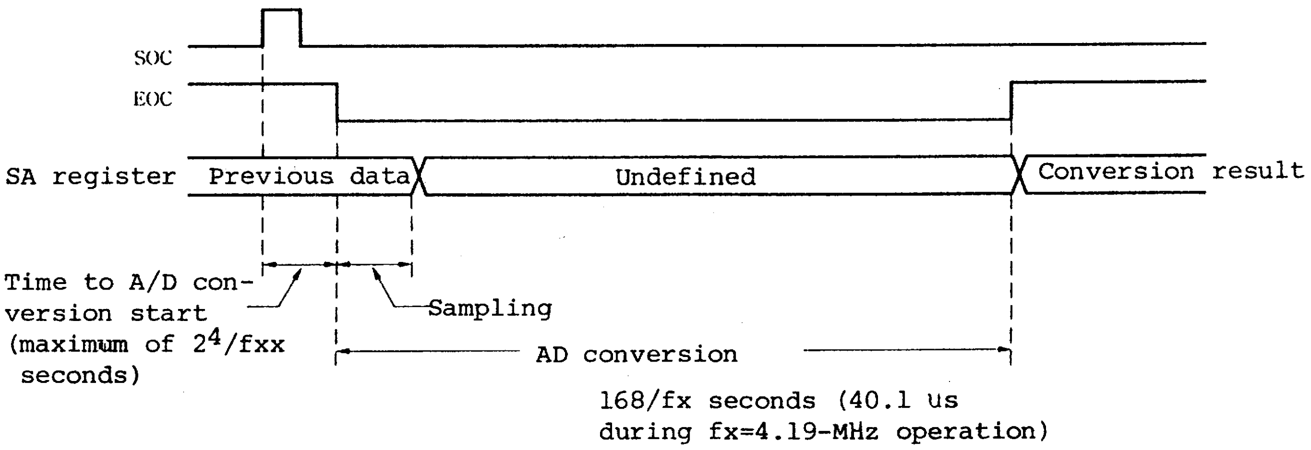
ACHN_P	Analog channel
0	AN0
1	AN1
2	AN2
3	AN3

A/D conversion is started by setting SOC (AMD bit 3) to 1.

After set, the SOC bit is automatically reset to 0. The A/D conversion is executed by the hardware (successive approximation) and the 8-bit data of the conversion result is stored in the SA register. When the A/D conversion terminates, EOC (ADM bit 2) is set to 1.

Fig. 5-2 shows A/D conversion timing chart.

Fig. 5-2 A/D Conversion Timing Chart


SOC
EOC
SA register
Previous data
Undefined
Conversion result
Time to A/D conversion start (maximum of 2^4/fxx seconds)
Sampling
AD conversion
168/fx seconds (40.1 us during fx=4.19-MHz operation)

5.1.2 Explanation of structured assembler

The program makes an infinite loop by using forever for the while statement condition and waits until EOC is set to 1 after communication terminates. When EOC is set to 1, the program performs processing and exits the while loop when break appears.

While loop 1

```
while (forever)
        if_bit (EOC)
                ?STORE    ADM, ACHN_P, #8
                break
        endf
endw
```

This can also be described as follows:

While loop 2

```
while_bit ( ! EOC)
endw
 ?STORE    ADM, ACHN_P, #8
```

5.1.3 Explanation of macro instructions

The following three macro instructions are used:

- ?STORE: ANDs the ACHN_P value with 7 and checks to see if errnoeous data is entered in ADM, then sets in the high-order bits of ADM and 8 in the low-order bits.
- ?WAIT: Waits for four machine cycles from A/D conversion start to EOC reset.
- ?CLR: Is converted into a CLR1 instruction.

5.1.4 Explanation of package

<Public declaration symbols>

ADCNV_D

<Registers>

. Bank: RBE x RBS . Register: XA

<RAM area>

Address	Name	Use	Initial value
47H	ACHN_P	Analog channel pointer	——
48H-49H	ACONV_D	A/D conversion data area	——

<Nesting>

One level (4 x 4-bit stack area)

<Hardware>

. Port: AN0, AN1, AN2, AN3

. A/D converter

<Initialization>

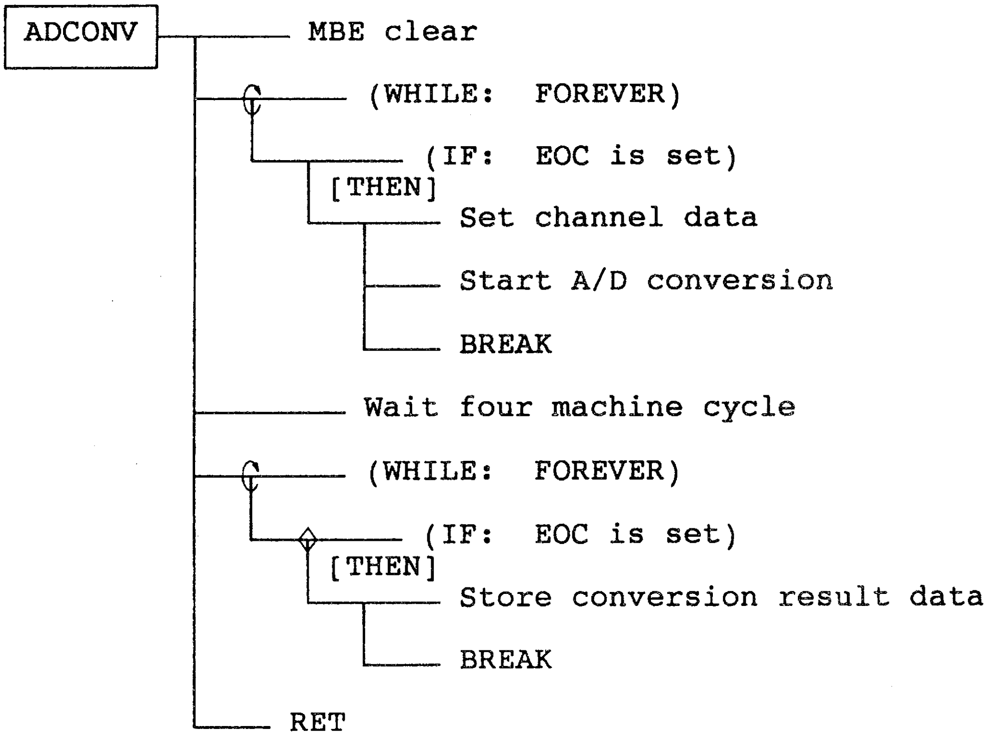
PCC ← 3H

<Start method>

call ADCNV.

5.2 SPD

ADCONV (subroutine)



```
ADCONV
    MBE clear
    (WHILE:  FOREVER)
        (IF:  EOC is set)
        [THEN]
            Set channel data
            Start A/D conversion
            BREAK
    Wait four machine cycle
    (WHILE:  FOREVER)
        (IF:  EOC is set)
        [THEN]
            Store conversion result data
            BREAK
    RET
```

5.3 Program Example

ADCNV (subroutine)

```
;-------------------------------------------------------------------------
        PUBLIC  ACONV_D
;-------------------------------------------------------------------------

?STORE  MACRO   P1,P2,P3                                ; Store P3 and (P2 and 7) in P1
                                        MOV     A,P2
                                        AND     A,#7
                                        MOV     X,P3
                                        XCH     A,X
                                        MOV     P1,XA
        ENDM

?WAIT   MACRO                                           ; Wait for four machine cycles
                                        NOP
                                        NOP
                                        NOP
                                        NOP
        ENDM

?CLR    MACRO   P1                                      ; CLR1 instruction
                                        CLR1    P1
        ENDM
;-------------------------------------------------------------------------

ACHN_P  DSEG    0 AT 47H                                ;Analog channel pointer
        DS      1
ACONV_D:DS      2                                       ;A/D conversion data area

ADCNV   CSEG    INBLOCK

?CLR    MBE

while(forever)
    if_bit(EOC)                                         ;Does conversion terminate?
        ?STORE  ADM,ACHN_P,#8                           ;Set data and start conversion
        break
    endif
endw

?WAIT

while(forever)
    if_bit(EOC)                                         ;Does conversion terminate?
        ACONV_D=SA (XA)                                 ;Store conversion result data
        break
    endif
endw

RET

END
```

Assembly list is given in A.2.

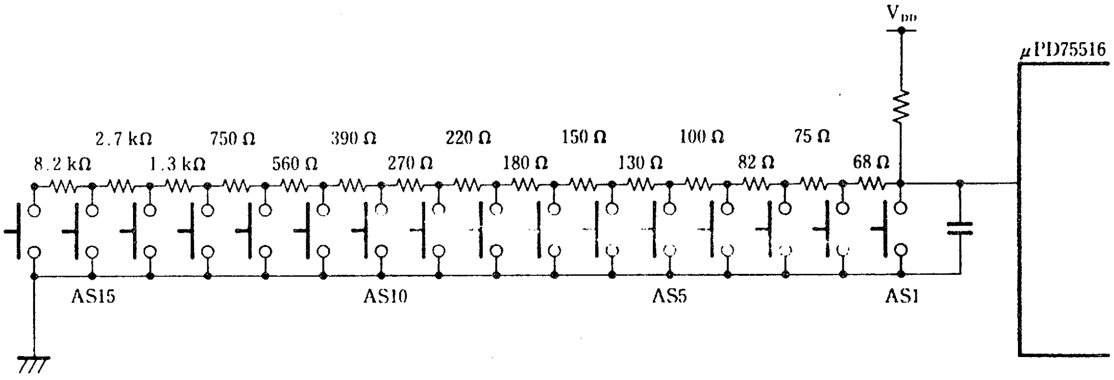
CHAPTER 6 ANALOG KEY INPUT PROGRAM

6.1 Explanation of Program

A program example is given which stores analog key input data in a data area by using A/D converter.

6.1.1 Program outline

Fig. 6-1 Analog Key Diagram



The example program uses AN4 (analog channel 4). There are 16 analog keys. Key input is enabled when a match is found 10 times (about 20 ms) by interrupt for chattering. If key data changes, the key change flag (AKCHA_F) is set.

Table 6-1 lists the correspondence between the analog keys and data stored in the key data area (AKEY_D).

Table 6-1 Correspondence between Analog Keys and Stored Data

Analog key	AS1	AS2	AS3	AS4	AS5	AS6	AS7	AS8
Stored data [H]	0	1	2	3	4	5	6	7
Analog key	AS9	AS10	AS11	AS12	AS13	AS14	AS15	AS16
Stored data [H]	8	9	A	B	C	D	E	F

Table 6-2 lists the A/D conversion values when analog keys AS1-AS16 are pressed although a slight error is contained.

Table 6-2 Correspondence between Analog Keys and A/D Conversion Value

Analog key	AS1	AS2	AS3	AS4	AS5	AS6	AS7	AS8
Conversion value	00H	10H	20H	30H	40H	50H	60H	70H
Analog key	AS9	AS10	AS11	AS12	AS13	AS14	AS15	AS16
Conversion value	80H	90H	A0H	B0H	C0H	D0H	E0H	F0H

6.1.2 Explanation of structured assembler

Here, a given A/D conversion value is converted into key data of 0H-FH as described below:

Conversion of A/D conversion value to key data

```
A<->X          ; The high-order part and low-order part of A/D
                 conversion value in XA register are exchanged.

BC = XA        ; The resultant value is set in BC register.

A<->X          ; The A/D conversion value is restored.

B=#0           ; The high-order part of BC register (low-order
                 part of the A/D conversion value) is restored
                 to 0.

?CMPHL         ; The low-order bits of the A/D conversion value
                 are set in the address addressed by HL register.

if(@HL>#7C) (A) ; If the low-order bit value of the A/D
                  conversion value is greater than 7,
      BC++     ; The BC register value is incremented.

      NOP
endif          ; The BC register value is incremented.
```

6.1.3 Explanation of macro instructions

The following three macro instructions are used:

. ?CMPHL: Sets work area (WORK_W) address in the HL register and stores A register data in WORK_W.

If relational operator ">" is used in a conditional expression, a comparison between the A register and @HL can only be made in 4-bit data comparison. The ?CMPHL instruction is used to set comparison data in @HL.

. ?SET: Is converted into a SET1 instruction.

. ?CLR: Is converted into a CLR1 instruction.

6.1.4 Explanation of package

<Public declaration symbols>

AKCHA_F

<Registers>

. Bank: REB x RBS . Registers: XA, HL, BC

<RAM area>

Address	Name	Use	Initial value
40H-41H	AKEY_D	Key data area	——
42H-43H	AKEY_W	Chattering work area	——
44H.0	AKCHA_F	Key change flag	0
44H.1	ACHCT_F	Chaterring count flag	0
45H	ACH_C	Chaterring counter	0
46H	CMP_W	Work area	——

<Nesting>

One level (6 x 4-bit stack area)

<Hardware>

. Port: AN4

. A/D converter

. BT (basic interval timer)

<Interrupts>

. INTBT

<Initialization>

. PCC ← 3H

. BT (inverval = 1.95 ms)

. ADM (analog channel 4)

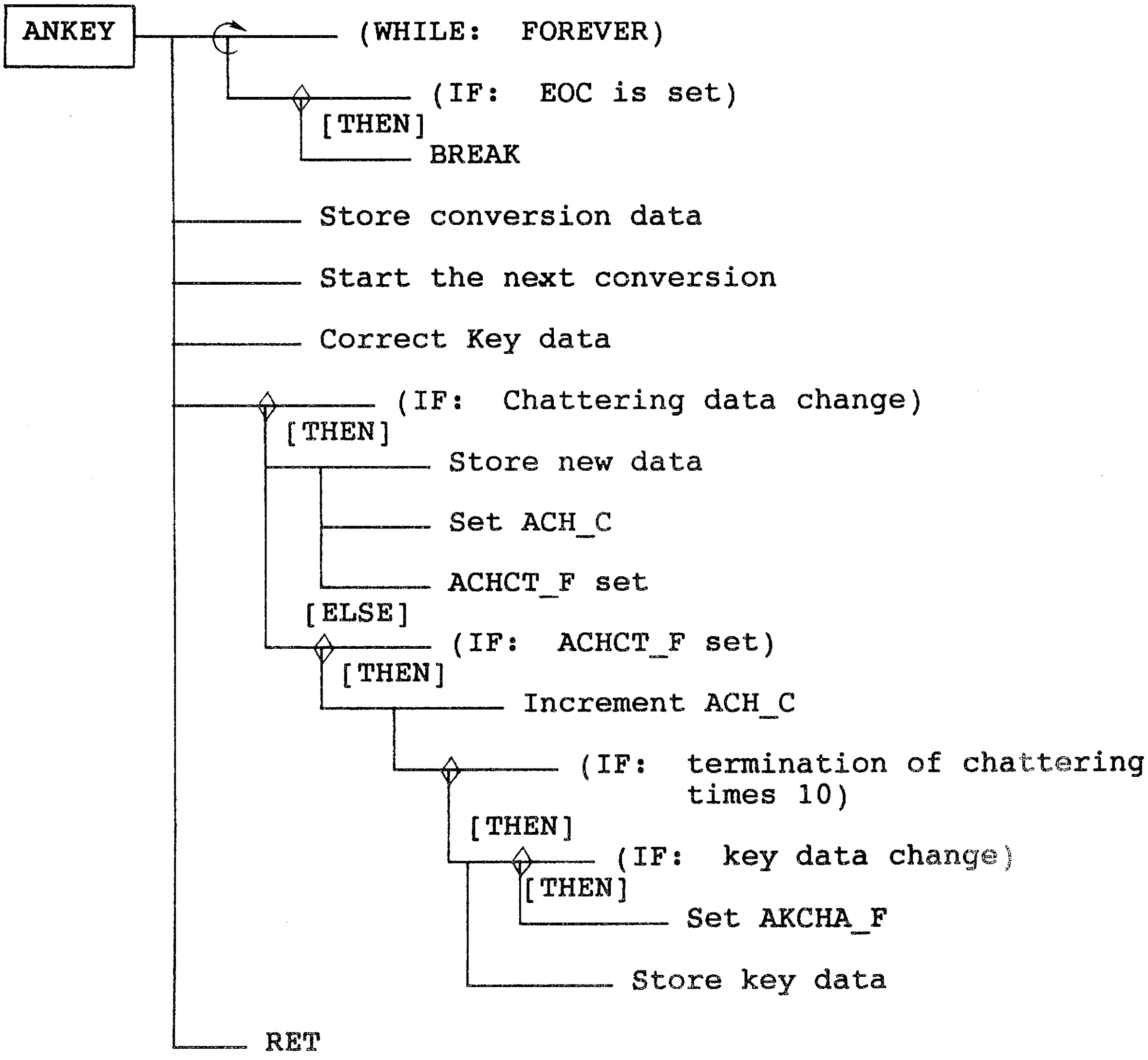
. INTBT enable

<Start method>

. Call ANKEY by making an INTBT interrupt.

6.2 SPD

ANKEY (subroutine)



```
ANKEY
    (WHILE:  FOREVER)
        (IF:  EOC is set)
        [THEN]
            BREAK
    Store conversion data
    Start the next conversion
    Correct Key data
    (IF:  Chattering data change)
    [THEN]
        Store new data
        Set ACH_C
        ACHCT_F set
    [ELSE]
        (IF:  ACHCT_F set)
        [THEN]
            Increment ACH_C
                (IF:  termination of chattering
                      times 10)
                [THEN]
                    (IF:  key data change)
                    [THEN]
                        Set AKCHA_F
                    Store key data
    RET
```

6.3 Program Example

ANKEY (subroutine)

```
;--------------------------------------------------------------------------
        PUBLIC  AKCHA_F
;--------------------------------------------------------------------------

AKEY_D  DSEG    0 AT 40H                                   : Analog key data area
        DS      2
AKEY_W: DS      2                                          ; Analog key work area
AKFLG:  DS      1                                          : Analog flag area
ACH_C:  DS      1                                          : Analog chattering counter
CMP_W:  DS      1                                          ; Work area

AKCHA_F EQU     AKFLG.0                                    : Analog key change flag
ACHCT_F EQU     AKFLG.1                                      Analog chattering count flag
;--------------------------------------------------------------------------

?CMPHL  MACRO                                              ; Store comparison data in HL
                                          MOV     HL,#CMP_W
                                          MOV     @HL,A
        ENDM

?SET    MACRO   P1                                         : SET1 instruction
                                          SET1    P1
        ENDM

?CLR    MACRO   P1                                         : CLR1 instruction
                                          CLR1    P1
        ENDM

;--------------------------------------------------------------------------

ANKEY   CSEG    INBLOCK

while_bit(!EOC)                                            ; Does conversion terminate?
endw

XA=SA                                                      : Store conversion data
BC=XA
?SET    SOC                                                : Start the next conversion
A<->X                                                      : Correct key data
BC=XA
A<->X
B=#0
?CMPHL

if(@HL>#7) (A)
    BC++
    NOP
endif
if(AKEY_W!=BC)  (XA)                                       : Is data changed?

    AKEY_W=BC (XA)                                         ; Set new data
    ACH_C=#(0FH-10) (A)                                    : Set chattering counter
    ?SET    ACHCT_F
else

    if_bit(ACHCT_F)

        if(ACH_C!=#0FH) (A)
            ACH_C++                                        : Increment counter
        else
            ?CLR    ACHCT_F                                : Terminate count
            HL=#AKEY_D

            if(AKEY_W!=@HL) (XA)
                ?SET    AKCHA_F                            ; Set key change flag
            endif

            XA=AKEY_W
            @HL=XA
        endif

    endif

endif

RET

END
```

Assembly list is given in A.3.

Phase-out/Discontinued

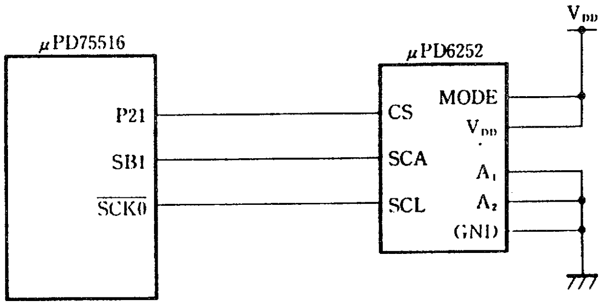
CHAPTER 7 COMMUNICATION PROGRAM WITH EEPROM

7.1 Explanation of Program

A program example is given which communicates with EEPROM by using the serial interface (channel 0).

7.1.1 Program outline

Fig. 7-1 Communication Diagram with EPROM



uPD6252 is used for EPROM. It is 2048-bit (256-word x 8-bit) electrically erasable programmable nonvoltile memory.

Write operation and read operation can be performed on 2-line or 3-line serial bus interface. The example program uses the 3-line serial I/O mode.

The uPD75516 operates in the 2-line serial I/O mode and P21 is used for the CS pin.

The uPD75516 2-line serial I/O mode has the following features:

- The two lines of $\overline{\text{SCK0}}$ (serial clock) and SB0/SB1 (serial data bus) can be used for communication.

- The uPD75516 can communicate with a number of devices by software which handles the output levels to the two lines. It can also deal with any desired communication format.

(1) Start bit issuing

To start data transfer, the low-to-high transition of the CS pin (P21) is made when the SCL pin is high.

(2) Command transmission

After the CS pin is turned on, the uPD75516 transmits an 8-bit command.

SDA: Data input
SCL: Serial clock input

Serial data is read on the serial clock rising edge. Transfer is started when data is written into SIO0. FFH is written for reception.

(a) Write

Write is executed after the RANDOM WRITE command
MSB
[00000000B} is transmitted.

an 8-bit word address is input from the SDA pin, then write data is input in 8-bit units.

The WB (WRITE BUSY) state is entered during the write and SDA pin output goes high. In the WB state, every command input becomes invalid and data transfer is stopped.

(b) Read

MSB
Read is executed after the RANDOM READ command [11000000B] is transmitted.

When a word address is input from the SDA pin, the contents of the memory addressed by the word address are transferred to the read buffer and can be read through the SDA pin.

After the data read terminates, the high-to-low transition of the CS pin is made.

(3) Stop bit issuing

To terminate the data transfer, the high-to-low transition of the CS pin is made when the SCL pin is high. In the WRITE mode, the transferred data is not written into the memory unless stop bit is issued.

The program writes and reads data as described below:

o Write

When the read/write flag (E2RW_F) is set to 1 and the program is called, the program writes the data set in the 8-bit write data area (E2WD_D) into the address set in the 8-bit write address area (E2WA_D).

o Read

When E2RW_F is reset to 1 and the program is called, the program reads the data at the address set in the 8-bit read address area (E2RA_D) and stores it in the 8-bit read data area (E2RD_D).

7.1.2 Explanation of structured assembler

After command transmission, WB signal reception is checked as follows:

WB signal reception check

```
HL=#0                   ; Comparison data 0 is set in HL register.

if (SIO0==HL) (XA)      ; If the SIO0 value is 0, it indicates the
                          WB signal.

     ?SIO2    E2RA_D    ; Read address is transmitted.

     ?SIO2    #0FFH     ; Read data is received.

E2RD_D = SIO0 (XA)      ; Read data is stored.

     ?CLR     CS_0      ; Communication termination.

     break              ; While loop is existed.

endif
```

The value to be compared in the conditional expression, 0 is set in the HL register before the if statement because the statement if (SIO0 = #0) (XA) cannot be described.

SIOSUB communication processing is performed as follows:

SIOSUB communication processing

```
SIOSUB:

    XIO0=XA                ; When send data is set in SIO0,
                             communication starts.
    repeat

    until_bit (IRQCSI0) ; Wait until IRQCSI0 is set.

    ?CLR   IRQCSI0         ; IRQCSI0 is reset.

    RET
```

7.1.3. Explanation of macro instructions

The following three macro instructions are used:

- ?SIO2: Sets send data in XA register and calls SIOSUB.
- ?SET: Is converted into a SET1 instruction.
- ?CLR: Is converted into a CLR1 instruction.

7.1.4 Explanation of package

<Public declaration symbols>

E2WD_D, E2RA_D, E2RD_D, E2RW_F

<Registers>

. Bank: RBE x RBS . Registers: XA. HL

<RAM area>

Address	Name	Use	Initial value
27H.3	E2RW_F	Read/write flag	——
30H-31H	E2WA_D	Write address area	——
32H-33H	E2WD_D	Write data area	——
34H-35H	E2RA_D	Read addrss area	——
36H-37H	E2RD_D	Read data area	——

<Nesting>

Two levels (8 x 4-bit stack area)

<Hardware>

. Port: P01 ($\overline{\text{SCK0}}$ pin)
P03 (SIO/SB1 pin)
P21

. Serial interface (in 2-line serial I/O mode)

<Initialization>

- PCC ← 3H
- CSIM0 (serial operation mode register ← 9FH (2-line serial I/O mode)
- RELT set (SO0 latch set)

<Start method>

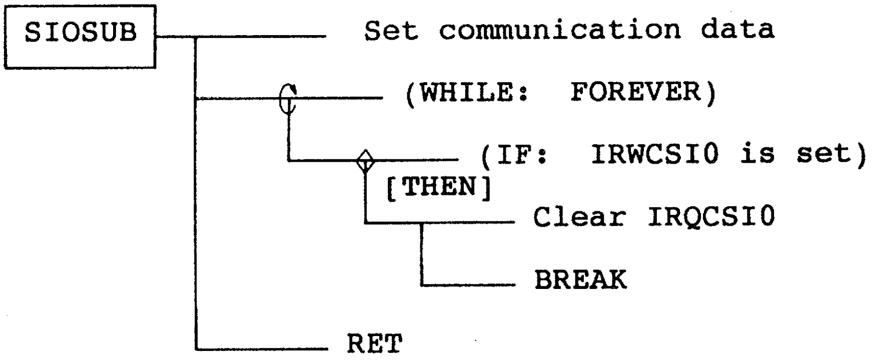
- Call EEPROM.

7.2 SPD

EEPROM (subroutine)

```
EEPROM
  (IF:  E2RW_F is set)
  [THEN]
    (WHILE:  FOREVER)
      MBE = 0
      Set CS pin high
      Set random read command
      Serial communication  SIOSUB
      Prepare for WB signal reception
      Serial communication  SIOSUB
      (IF:  communication OK)
      [THEN]
        Set read address
        Serial communication  SIOSUB
        Prepare for read data reception
        Serial communication  SIOSUB
        Store read data
      Set CS pin low
  [ELSE]
    (WHILE:  FOREVER)
      MBE = 0
      Set CS pin high
      Set random write command
      Serial communication  SIOSUB
      Prepare for WB signal reception
      Serial communication  SIOSUB
      (IF:  communication OK)
      [THEN]
        Set write address
        Serial communication  SIOSUB
        Set write data
        Serial communication  SIOSUB
      Set CS pin low
  RET
```

SIOSUB (subroutine)



7.3 Program Example

(1) EEPROM (subroutine)

```
;------------------------------------------------------------------------
       PUBLIC  E2WD_D,E2RA_D,E2RD_D,E2RW_F
;------------------------------------------------------------------------

E2WA_D  DSEG    0 AT 30H                                    ; Write address area
        DS      2
E2WD_D: DS      2                                           ; Write data area
E2RA_D: DS      2                                           : Read address area
E2RD_D: DS      2                                           : Read data area

E2RW_F  EQU     27H.3                                       : Read/write flag
CS_0    EQU     PORT2.1                                     ;CS for uPD6252

;------------------------------------------------------------------------

?SIO2   MACRO   P1                                          :Transfer data P1
                                        MOV     XA,P1
                                        CALLF   !SIOSUB
        ENDM

?SET    MACRO   P1                                          :SET1 instruction
                                        SET1    P1
        ENDM

?CLR    MACRO   P1                                          :CLR1 instruction
                                        CLR1    P1
        ENDM

;------------------------------------------------------------------------

EEPROM  CSEG    INBLOCK

if_bit(E2RW_F)

    while(forever)                                          : Read mode
        ?CLR    MBE
        ?SET    CS_0                                        : Start communication
        ?SIO2   #0C0H                                       : Transmit random read command
        ?SIO2   #0FFH                                       : Receive WB flag
        HL=#0

        if(SIO0==HL) (XA)                                   : BUSY?
            ?SIO2   E2RA_D                                  : Transmit read address
            ?SIO2   #0FFH                                   : Receive read data
            E2RD_D=SIO0 (XA)                                : Store read data
            ?CLR    CS_0
            break
        endif

        ?CLR    CS_0
    endw

else
    while(forever)                                          :Write mode
        ?CLR    MBE
        ?SET    CS_0                                        :Start communication
        ?SIO2   #0                                          :Transmit random write command
        ?SIO2   #0FFH                                       :Receive WB flag
        HL=#0

        if(SIO0==HL) (XA)
            ?SIO2   E2WA_D                                  :Transmit write address
            ?SIO2   E2WD_D                                  :Transmit write data
            ?CLR    CS_0
            break
        endif

        ?CLR    CS_0
    endw

endif

RET
```

(2) SIOSUB (subroutine)

```
SIOSUB:
SIO0=XA                                          ;Transfer data

repeat
until_bit(IRQCSIO)                               ;Wait for communica-
                                                  tion to terminate
?CLR    IRQCSIO
RET

END
```

Assembly list is given in A.4.

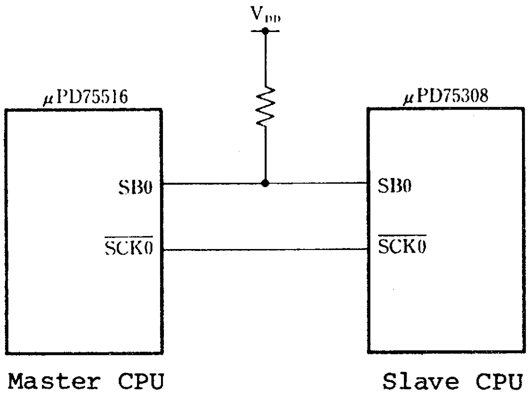
CHAPTER 8   SBI COMMUNICATION PROGRAM

8.1 Explanation of Program

A program example is given for SBI communication with slave CPU by using the serial interface.

8.1.1 Program outline

Fig. 8-1 SBI Communication Diagram



The example program performs SBI communication processing as follows:

(1) Address transmission

Data (0) is set in the communication code pointer (SBI_P). An 8-bigt address is set in the send data area (SBITR_D). The program is called.

(2) Command transmission

Data (1) is set in SBI_P. An 8-bit command is set in SBITR_D. The program is called.

(3) Data transmission

Data (2) is set in SBI_P. 8-bit data is set in SBITR_D. The program is called.

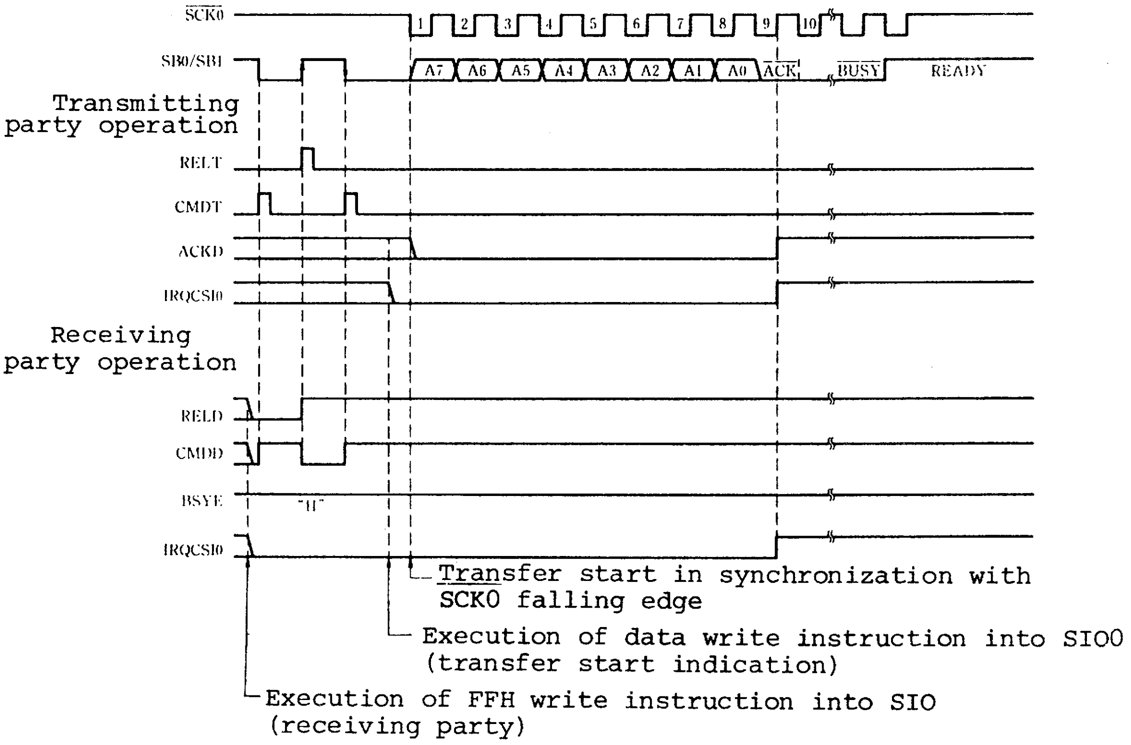
(4) Data reception

Data (3) is set in SBI_P and the program is called.

Fig. 8-2 shows the address, command, and data transfer formats.

Fig. 8-2 Address, Command, and Data Transfer Formats

(a) Address transfer

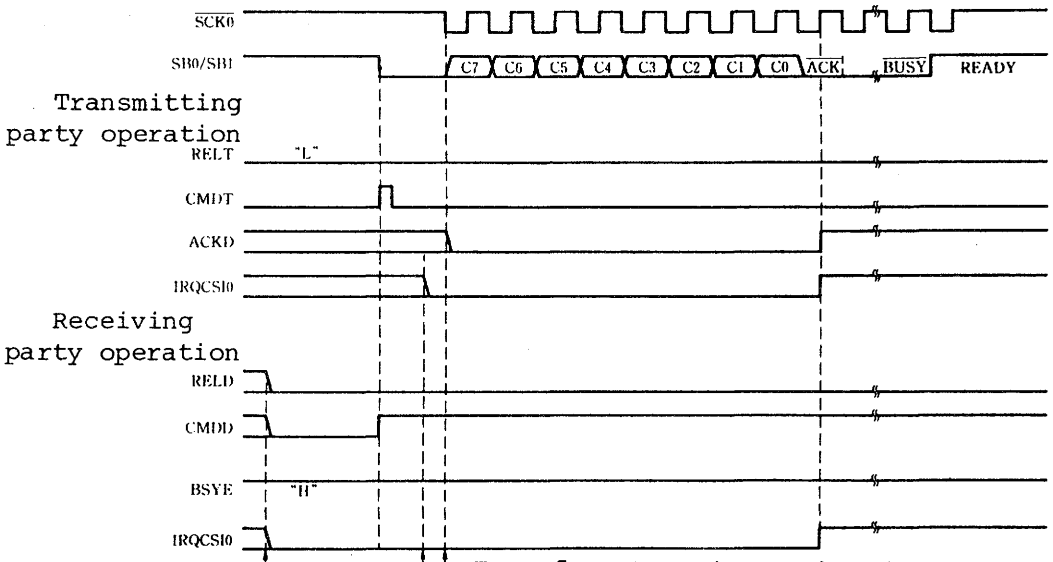


Transfer start in synchronization with SCK0 falling edge

Execution of data write instruction into SIO0 (transfer start indication)

Execution of FFH write instruction into SIO (receiving party)

(b) Command transfer

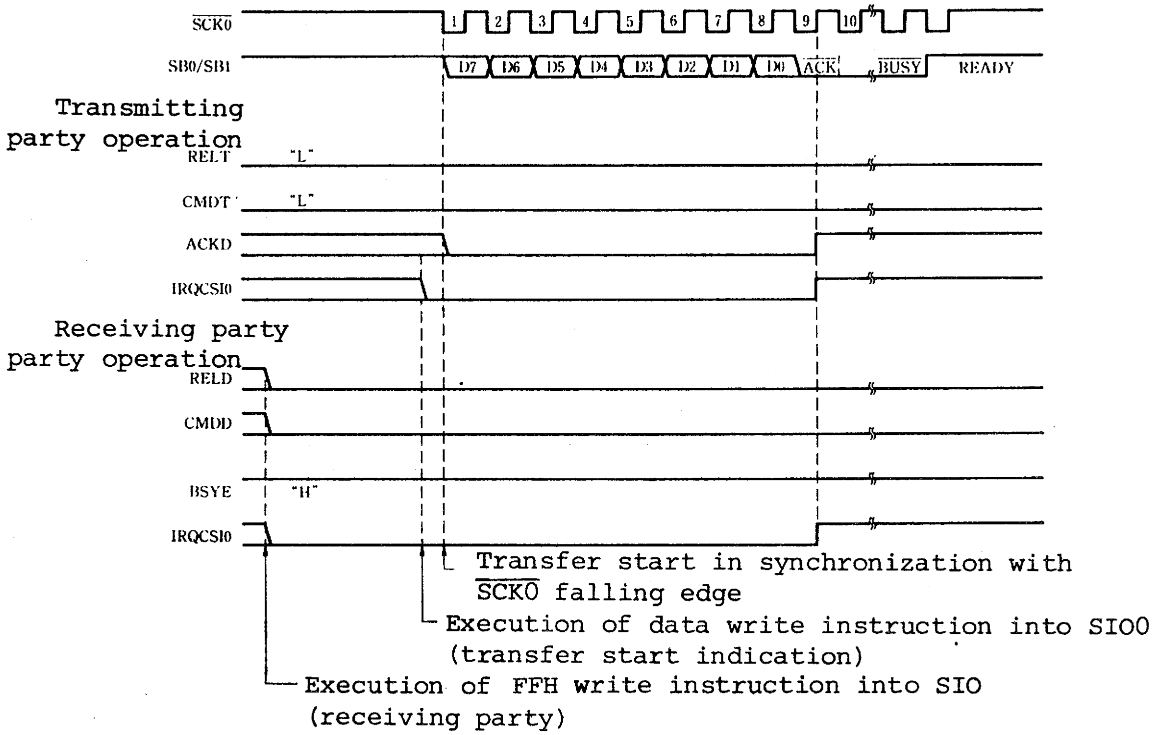


Transfer start in synchronization with SCK0 falling edge

Execution of data write instruction into SIO0 (transfer start indication)

Execution of FFH write instruction into SIO (receiving party)

(c) Data transfer



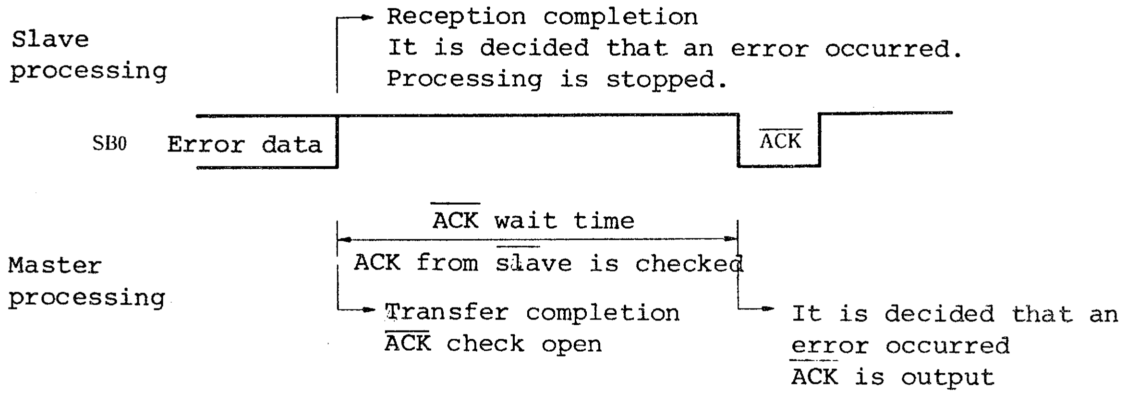
Transfer start in synchronization with $\overline{\text{SCK0}}$ falling edge

Execution of data write instruction into SIO0 (transfer start indication)

Execution of FFH write instruction into SIO (receiving party)

Whether or not an error occurred in the slave CPU is judged depending on the acknowledge signal ($\overline{\text{ACK}}$) returned by the slave CPU.

Fig. 8-3 $\overline{\text{ACK}}$ when an Error Occurred



Reception completion
It is decided that an error occurred.
Processing is stopped.

$\overline{\text{ACK}}$ wait time

ACK from $\overline{\text{slave}}$ is checked

Transfer completion
$\overline{\text{ACK}}$ check open

It is decided that an error occurred
$\overline{\text{ACK}}$ is output

After 8-bit data transfer is complete (INTCSI occurs), the master CPU checks the acknowledge signal ($\overline{\text{ACK}}$) teturned by the slave CPU.

If the slave CPU does not return the $\overline{\text{ACK}}$ signal within a given time after transfer terminates, the master CPU decides that an error occurred in the slave CPU, and transmits dummy $\overline{\text{ACK}}$ signal to the slave CPU. (See Fig. 8-3.)

The program counts the $\overline{\text{ACK}}$ signal wait time based on the timer/event counter interval time. If the $\overline{\text{ACK}}$ signal is not received from the slave CPU by the time the 16th INTT0 occurs after the transfer terminates, it is decided that an error occurred.

When an address, command, or data is transmitted, the same data is written into both SIO0 (shift register) and SVA (slave address register) for an error check when send data changes on the bus line.

8.1.2 Explanation of structured assembler

The program performs signal output and data setting by using the switch statement. Address transmission, command transmission, data transmission, and data reception differ in signal output and data setting.

Signal output and data setting

```
switch(SBI_P)         ; SBI_P value is set in A register as
                        condition value
   case0:             ; If the SBI_P (A register) value is 0,

         ?SET  CMDT   ; Command signal is output.

         ?SET  RELT   ; Bus release signal is output.

         A=#1         ; 1 is set in A register to proceed to
                        case1
   case1:             ; If the SBI_P (A register) value is 1,

         ?SET  CMDT   ; Command signal is output.

         A=#2         ; 2 is set in A register to proceed to
                        case2
   case2:             ; If the SBI_P (A register) value is 2,

         XA=SBTR_D    ; Send data is set.

         break        ; break causes the switch statement to be
                        exited.

   case3:             ; If the SBI_P (A register) value is 3,

         XA=#0FFH     ; FFH is set for preparation for
                        reception.
ends
```

In the switch statement, if break is not described at the end of each case process, the next case decision is made. If the A register value (actual decision value) is changed at the time, another case process can be performed.

The INTT0 interrupt service routine (ACKWOP) performs over wait time error handling for ACK signal reception wait.

Over wait time handing

```
if (ACKW_C==#0FH) (A)  ; If ACKW_F (ACK wait counter) is FH,

      ACKW_C=#0 (A)    ; ACKW_F is reset.

      ?SET  ACKT       ; ACK signal is transmitted.

      ?SET  ERR_F      ; ERR_F (error flag) is set.

      ?CLR  ACKW_F     ; ACKW_F (ACK wait flag) is reset.

      ?CLK  BUSY_F     ; BUSY_F (BUSY flag) is reset.

else                   ; If ACKW_F is not FH,

      ACKW_C++         ; ACKW_F is incremented.
endif
```

This handling causes over wait time to occur when the 15th INTT0 interrupt occurs.

8.1.3 Explanation of macro instructions

The following two macro instructions are used:

- ?SET: Is converted into a SET1 instruction.
- ?CLR: Is converted into a CLR1 instruction.

8.1.4 Explanation of package

<Public declaration symbols>

SBRE_D, SBI_P, and ERR_F

<Registers>

. Bank: MBE x RBE . Register: XA

. Bank: 0 . Register: XA

<RAM area>

Address	Name	Use	Initial value
50H-51H	SBTR_D	Send data area	——
50H-53H	SBRE_D	Receive data area	——
54H	SBI_P	Communication code pointer	——
55H.0	ERR_F	Error flag	0
55H.1	BUSY_F	BUSY flag	0
55H.2	ACKW_F	ACK wait flag	0
56H	ACKW_C	ACK wait counter	0

<Nesting>

Two levels (8 x 4-bit stack area)

<Hardware>

. Port: P01 ($\overline{\text{SCK0}}$ pin)
P02 (SB0/SO0 pin)

. Serial interface (SBI mode)

. T0 (timer/event counter 0)

<Interrupts>

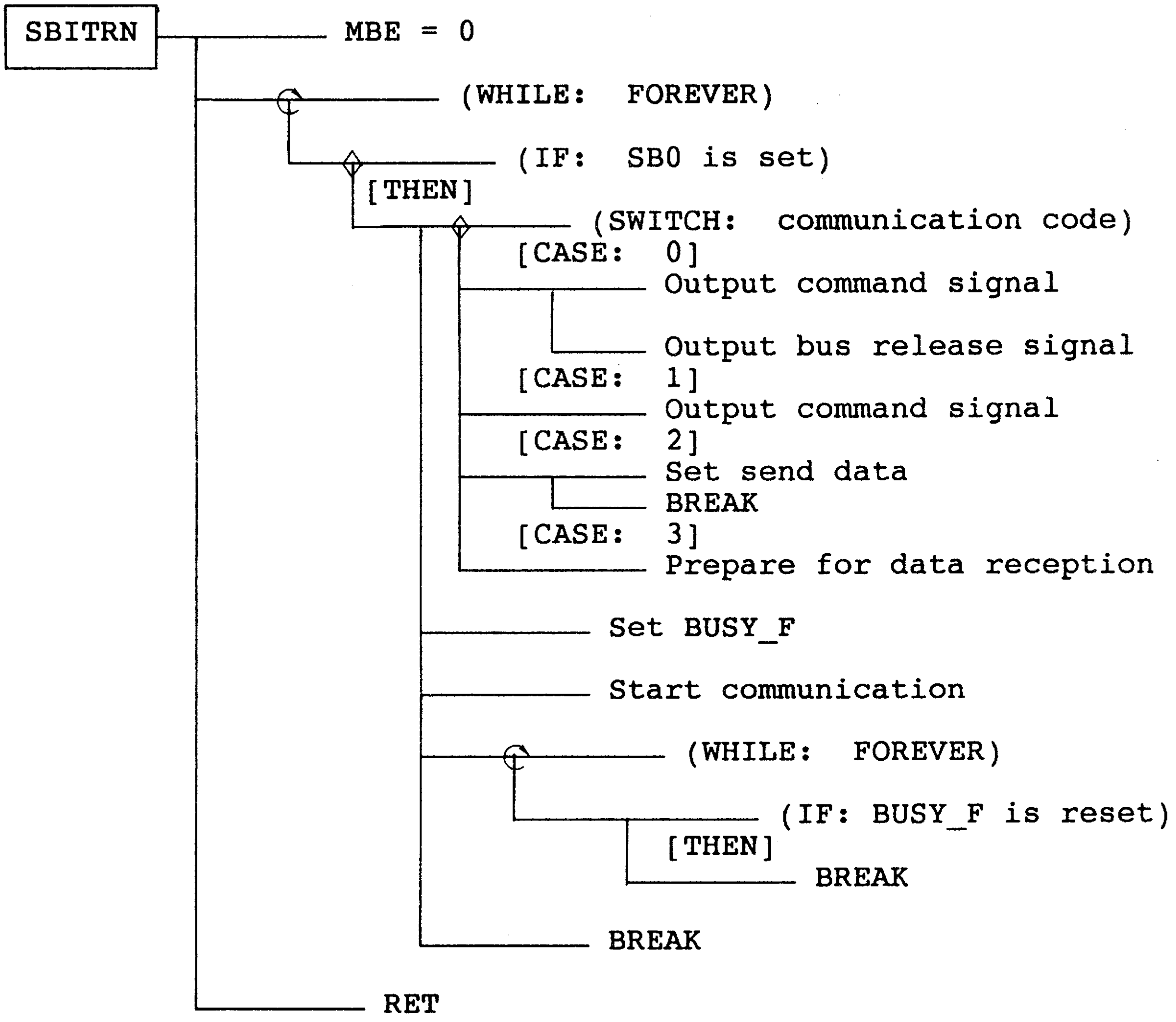
INTCSI0, INTT0

```
<Initialization>
     PCC ← 3H
     POGA (PORT0 pull-up resistor setting)
     CSIM0 ← 8BH (SBI mode, BUS = SB0)
     TMOD0 ← 41H
     TM0 ← 6AH (interval = 1 ms)
     INTCSI0 and INTT0 enable

<Start method>
     Call SBITRN.
```

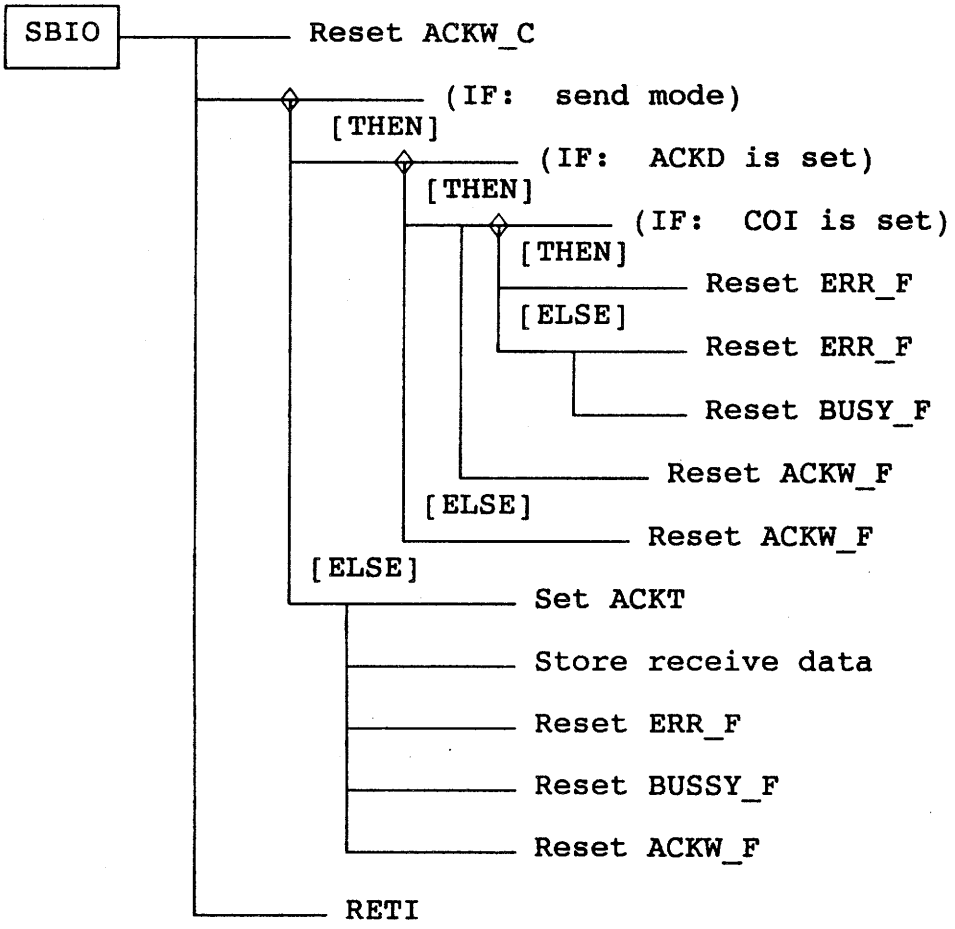
8.2 SPD

SBITRN (subroutine)

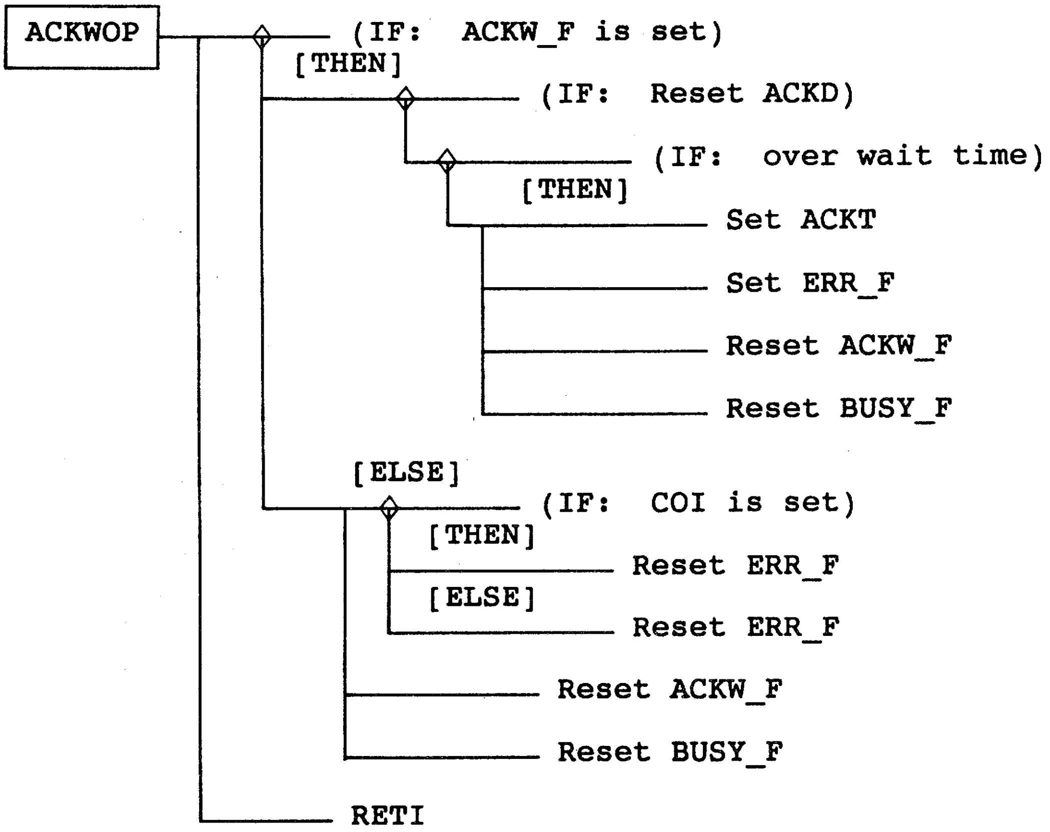


```
SBITRN
    MBE = 0
    (WHILE:  FOREVER)
        (IF:  SB0 is set)
        [THEN]
            (SWITCH:  communication code)
            [CASE:  0]
                Output command signal
                Output bus release signal
            [CASE:  1]
                Output command signal
            [CASE:  2]
                Set send data
                BREAK
            [CASE:  3]
                Prepare for data reception
            Set BUSY_F
            Start communication
            (WHILE:  FOREVER)
                (IF: BUSY_F is reset)
                [THEN]
                    BREAK
            BREAK
    RET
```

SBIO (INTCSI0 interrupt)



ACKWOP (INTT0 interrupt)



```
ACKWOP
    (IF:  ACKW_F is set)
    [THEN]
        (IF:  Reset ACKD)
            (IF:  over wait time)
            [THEN]
                Set ACKT
                Set ERR_F
                Reset ACKW_F
                Reset BUSY_F
    [ELSE]
        (IF:  COI is set)
        [THEN]
            Reset ERR_F
        [ELSE]
            Reset ERR_F
        Reset ACKW_F
        Reset BUSY_F
    RETI
```

8.3 Program Example

(1) SBITRN (subroutine)

```
;------------------------------------------------------------------------

        VENT4   MBE=0,RBE=0,SBIO
        VENT5   MBE=0,RBE=0,ACKWOP

        PUBLIC  SBRE_D,SBI_P,ERR_F
;------------------------------------------------------------------------

SBTR_D  DSEG    0 AT 50H                                  :: Send data area
        DS      2
SBRE_D: DS      2                                         : Receive data area
SBI_P:  DS      1                                         : Communication code pointer
SBIFLG: DS      1                                         : SBI flag area
ACKW_C: DS      1                                         : ACK wait counter

ERR_F   EQU     SBIFLG.0                                  : Error flag
BUSY_F  EQU     SBIFLG.1                                  : Busy flag
ACKW_F  EQU     SBIFLG.2                                  : ACK wait flag

SBI_O   EQU     PORT0.2

SBITRN  CSEG    INBLOCK

;------------------------------------------------------------------------

?CLR    MACRO   P1                                        : CLR1 instruction
                                          CLR1    P1
        ENDM

?SET    MACRO   P1                                        : SET1 instruction
                                          SET1    P1
        ENDM

;------------------------------------------------------------------------

while(forever)

    if_bit(SBI_O)                                         :Is bus busy?

        switch(SBI_P)
            case 0:
                ?SET    CMDT                              :Output command signal
                ?SET    RELT                              :Output bus release signal
                A=#1
            case 1:
                ?SET    CMDT                              :Output command signal
                A=#2
            case 2:
                XA=SBTR_D                                 :Set send data
                break
            case 3:
                XA=#0FFH                                  :Prepare for data reception
        ends

        ?SET    BUSY_F
        SVA=XA
        SIO0=XA                                           :Start communication

        while_bit(BUSY_F)                                 :Wait for communication to terminate
        endw

        break
    endif

 endw

 RET
```

(2) SBIO (INTCS10 interrupt)

```
SBIO    CSEG    INBLOCK

ACKW_C=#0 (A)                                   ; Clear ACK wait counter

if(SBI_P!=#3) (A)                               ; Transmission mode?

    if_bit(ACKD)                                ; ACK signal?

        if_bit(COI)                             ; Is COI set?
            ?CLR    ERR_F
        else
            ?SET    ERR_F
            ?CLR    BUSY_F
        endif

        ?CLR    ACKW_F
    else
        ?SET        ACKW_F
    endif

else
    ?SET    ACKT                                ; Reception mode?
    SBRE_D=SIO0 (XA)                            ; Store receive data
    ?CLR    ERR_F
    ?CLR    BUSY_F
    ?CLR    ACKW_F
endif

RETI
```

(3) ACKWOP (INTT0 interrupt)

```
ACKWOP  CSEG    INBLOCK

if_bit(ACKW_F)                                  ; ACK signal wait?

    if_bit(!ACKD)                               ; Check ACK signal

        if(ACKW_C==#0FH) (A)
            ACKW_C=#0 (A)
            ?SET    ACKT                        ; Over wait time
            ?SET    ERR_F                       ; Error
            ?CLR    ACKW_F
            ?CLR    BUSY_F
        else
            ACKW_C++
        endif

    else

        if_bit(COI)                             ; Check bus error
            ?CLR    ERR_F
        else
            ?SET    ERR_F                       ; Error
        endif

            ?CLR    ACKW_F
            ?CLR    BUSY_F
    endif

endif

RETI

END
```

Assembly list is given in A.5.

CHAPTER 9 NOTATION ADJUSTMENT PROGRAM

9.1 Explanation of Program

A program example is given for conversion of hexadecimal 1-dight data to a four count, six count, octal, decimal, or twelve count number.

9.1.1 Program outline

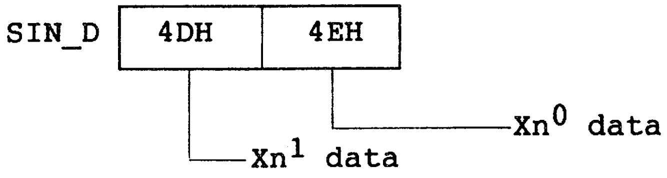
The example program adjusts 4-bit data set in the notation work area (SIN_W) adjusts to the notation indicated by the data set in the notation code pointer (SIN_P) and stores the result in the notation data area (SIN_D).

Table 9-1 lists the correspondence between the SIN_P data and notation.

Table 9-1 SIN_P Data and Notation

SIN_P data	Notation
0	4
1	6
2	8
3	10
4	12

The result is stored in SIN_D as shown below:



n: Notation value

Twelve count digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, and B.

Thus, if hexadecimal number B is adjusted to a twelve count number, B is returned; if hexadeciaml number F is adjusted to a twelve count number, 13 is returend.

9.1.2 Explanation of structured assembler

The while statement is used for six count adjustment processing.

Six count adjustment processing

```
while (#5<@HL) (A)     : While the data in SIN_W (notation work
                          area ) addressed by the HL register
                          is greater than 5,

      A=@HL            : The data in SIN_W is set in the A
                          register.

      A+=# (16-6)      : 10 is added to the SIN_W data in the A
                         register (6 is subtracted).

      NOP              : NOP is inserted to enable a skip.

      X++              : The X register value is incremented.

      @HL=A            : The A register value is stored in SIN_W.

      BC=XA            : The conversion result is stored in the BC
                         register.
 endw
```

The data converted into six count notation is stored in the BC register by this processing.
The ADDC instruction is used for decimal adjustment processing.

Decimal adjuntment processing

```
CIRI    CY        : CY (carry flag) is reset.

A=#6              : 6 is set in the A register.

HL=#SIN_W         : The SIN_W address is set in the HL register.

ADDC    A, @HL    : The A register value is added to the SIN_W
                    value.  If a carry is generated, CY is set
                    and a skip is made.

A+=#10            : 10 is added to the A register.  Even if a
                    carry is generated, no skip is made.

A<->X             : The A register and X register values are
                    exchanged.

if_bit (CY)       : If CY is set,

(SIN_D+1)=#1 (A) : The high-order digit of SIN_D (notation
                    data area) is set to 1.  (The default
                    value is 0.)

endif
                  : The X register value is set in the low-order
                    digit of SIN_D.

SIN_D=X (A)
```

The ADDC instruction adds data with carry, thus CY must have been reset before the instruction is executed. If the ADDC instruction is immediately followed by the ADDS instruction, the ADDS instruction skip function is canceled.

The switch statement is used to separate notation adjustment processing according to the mode.

9.1.3 Explanation of macro instructions

The following macro instruction is used:

. ? SHIFT2: Shifts data right two bits.

9.1.4 Explanation of package

<Public declaration symbols>

SIN_P, SIN_W

<Registers>

. Bank: RBE X RBS   Registers: XA, HL, and BC

<RAM area>

Address	Name	Use	Initail value
4AH-4BH	SIN_D	Notation data area	—
4CH	SIN_P	Notation code pointer	—
4DH	SIN_W	Notation work area	—

<Nesting>

One level (4 x 4-bit stack area)

<Initalization>

PCC ← 3H

<Start method>

Call SINSU.

9.2 SPD

SINSU (subroutine)

```
SINSU
    Reset SIN_D
    (SWITCH:  notation code)
    [CASE:0]
        Adjust the high-order
        Store the adjustment result (high-order data)
        Adjust the low-order two bits of data to a four
        count number.
        Store the adjustment result (low-order data)
    [CASE:1]
        Set data in A register
        Reset X register
        (WHILE:  A register >5)
            Subtract 6 from A register
            Increment X register
        Store XA register value in SIN_D
    [CASE:2]
        Store data
        (IF:  most significant bit of data is 1)
        [THEN]
            Store 1 in SIN_D (high-order digit position)
        AND data and 7
        Store adjustment result (low-order data)
    [CASE:3]
        Store data
            Decimal adjustment
            Store adjustment result
    [CASE:4]
        (IF: data>11)
        [THEN]
            Store 1 in SIN_D (high-order digit position
            Subtract 12 from data
            Store adjustment result (low-order data)
        [ELSE]
            Store adjustment result
RET
```

9.3 Program Example

SINSU (subroutine)

```
;--------------------------------------------------------------------------
        PUBLIC  SIN_P,SIN_W
;--------------------------------------------------------------------------

SIN_D   DSEG    0 AT 4AH                                        ;Notation data area
        DS      2
SIN_P:  DS      1                                               ;Notation code pointer
SIN_W:  DS      1                                               ;Notation work area

;--------------------------------------------------------------------------

?SHIFT2 MACRO   P1                                              ;Shift P1 right two bits
                                                CLR1    CY
                                                RORC    P1
                                                CLR1    CY
                                                RORC    P1
        ENDM

;--------------------------------------------------------------------------

SINSU   CSEG    INBLOCK

SIN_D=#0 (XA)

switch(SIN_P)
    case 0:                                                     ;Four count adjustment result
        A=SIN_W
        A &= #1100B
        ?SHIFT2 A                                               ;Shift data right two bits
        (SIN_D+1)=A                                             ;Store high-order data
        A=SIN_W
        A &= #0011B
        SIN_D=A                                                 ;Store low-order data
        break
    case 1:                                                     ; Six count adjustment result
        HL=#SIN_W
        X=#0
        A=@HL
        BC=XA

        while(#5<@HL) (A)
            A=@HL
            A+=#(16-6)
            NOP
            X++
            @HL=A
            BC=XA
        endw
        XA=BC
        SIN_D=XA                                        ; Store adjustment result
        break

    case 2:                                             ; Eight count adjustment
        A=SIN_W                                           result

        if_bit(SIN_W.3)                                 ; Most significant Bit=1?
            (SIN_D+1)=#1 (A)                            ; Store high-order data
        endif

        A=SIN_W
        A &= #0111B
        SIN_D=A                                         ; Store low-order data
        break
    case 3:                                             ;Decimal adjustment result
        CLR1    CY
        A=#6
        HL=#SIN_W
        ADDC    A,@HL                   ;ADDC   A,@HL
        A+=#10
        A<->X

        if_bit(CY)
            (SIN_D+1)=#1 (A)                            ;Store low-order data
        endif

        SIN_D=X (A)                                     ;Store high-order data
        break
    case 4:                                             ;Twelve adjustment result
        HL=#SIN_W

        if(#11<@HL) (A)
            (SIN_D+1)=#1 (A)                            ;Store high-order data
            A=#16-12                                    (DATA>11)
            A+=@HL
            NOP
            SIN_D=A                                     ;Store low-order data
        else
            (SIN_D+1)=#0 (A)                            ;Store high-order data
            SIN_D=@HL (A)                                (DATA<12)
        endif                                           ;Store low-order data

ends

RET

END
```

Assembly list is given in A.6.

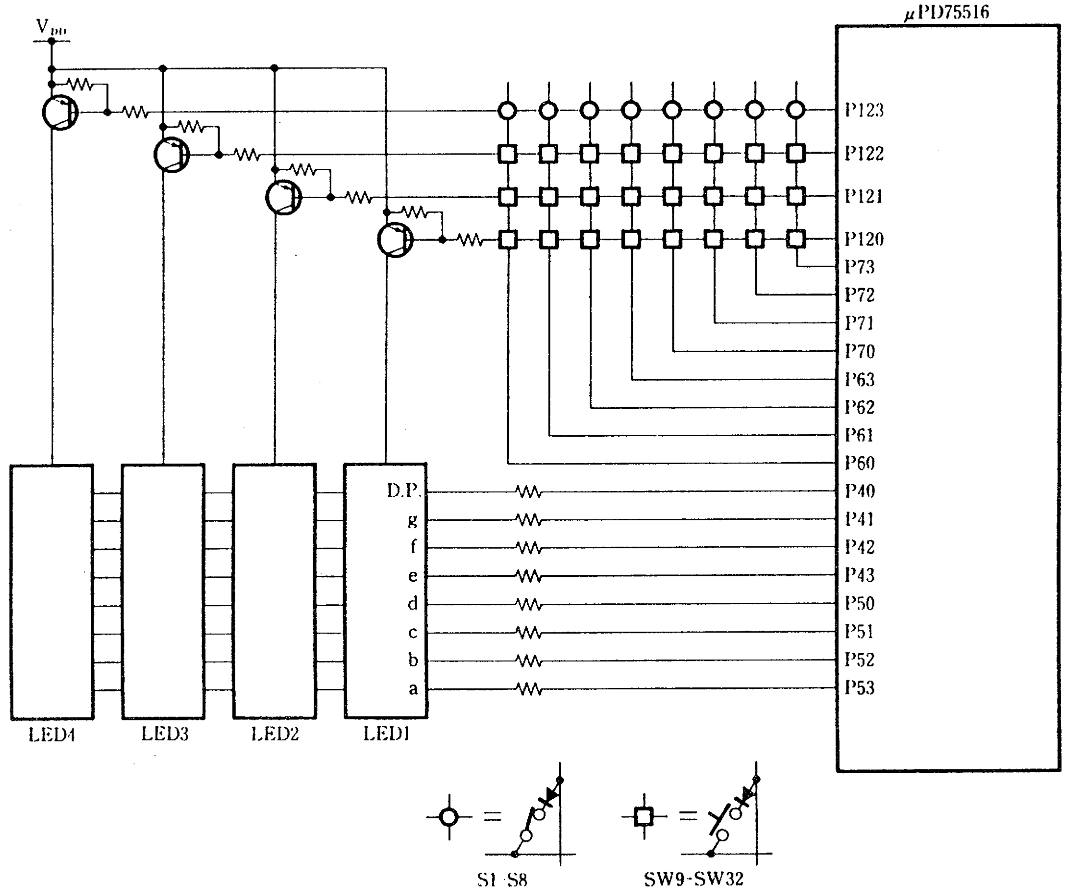
CHAPTER 10 LED DISPLAY AN KEY INPUT PROGRAM

10.1 Explanation of Program

A program example is given which displays setup data on four LEDs and stores key input data.

10.1.1 Program outline

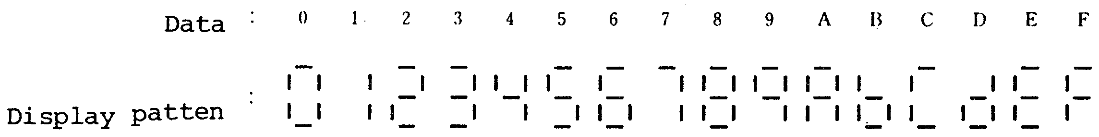
Fig. 10-1 LED and Key Diagram



(1) LED display

4-digit data in the LED display data area (LED_D) is displayed on LED1-LED4. Fig. 10-2 shows the correspondence between written data and display patterns.

Fig. 10-2 LED Display Patterns



(2) Key input

There are 32 keys. The S1-S8 keys are DIP switches. Key input data is set in the key data area (KEY_D). One key corresponds to one memory bit. When a key is pressed, the bit corresponding to the key is set to 1.

If key data changes, the key change flag (KCHA_F) is set. Table 10-1 lists the correspondence between the key and memory bits.

Table 10-1 Correspondence Between Keys and Memory Bits

Key strobe	P120		P121		P122		P123	
Address bit	28H	29H	2AH	2BH	2CH	2DH	2EH	2FH
0	SW25	SW29	SW17	SW21	SW9	SW13	S1	S5
1	SW26	SW30	SW18	SW22	SW10	SW14	S2	S6
2	SW27	SW31	SW19	SW23	SW11	SW15	S3	S7
3	SW28	SW32	SW20	SW24	SW12	SW16	S4	S8

10.1.2 Explanation of structured assembler

The example program stores data after chattering is ended by using the for statement as follows:

Data store processing

```
HL=#KEY_D                   ; KEY_D address is stored in the HL
                              register.

DE=#KEY_W                   ; KEY_W address is stored in the DE
                              register.

for (B=#0;B!=#8;B++)        ; Eight loops are made by using the
                              B register.

    A=@DL                   ; KEY_W address is stored in the A
                              register.

    if (A!=@HL)             ; If the KEY_W data differs from the
                              data stored in KEY_D,

     ?SET KCHA_F            ; KCHA_F (key change flag) is set.

    endif

    A<->@HL+                ; Data is stored in KEY_D and the
                              address set in HL is incremented.

    NOP                     ; NOP  is entered to suppress a skip.

next
```

10.1.3 Explanation of macro instructions

The following six macro instructions are used:

- ?DEC: Decrements 4-bit data.

- ?ROTATE: Rotates 4-bit data with carry.

- ?TABLE: Reads and stores table data.

. ?CALADR: Calculates the cattering data address from the DIG_D (digit data) value and KEY_W (key work area) address for storage.

. ?NOT: Inverts data bit-wise and stores the result.

. ?SET: Is converted into a SET1 instruction.

. ?CLR: Is converted into a CLR1 instruction.

10.1.4 Explanation of package

<Pablic declaration symbols>

DP_D, DIG_D, KEY_D, KCHA_F, LEDST_F

<Registers>

. Bank: 0 . Registers: XA, HL, DE, and BC

<RAM area>

Address	Name	Use	Initial value
20H-23H	LED_D	LED display data area	--
24H	DP_D	DP display data pointer	0
25H	DIG_D	Digit data pointer	0
26H	CH_C	Chattering counter	0
27H.0	KCHA_F	Key change flag	0
27H.1	CHCT_F	Chattering count flag	0
27H.2	LEDST_T	LED display flag	0
28H-2FH	KEY_D	Key data area	--
38H-3FH	KEY_W	Key work area	--

<Nesting>

One level (6 x 4-bit stack area)

<Hardware>

- Ports: PORT4
  PORT5
  PORT6 (KR0-KR3 pins)
  PORT7 (KR4-KR7 pins)
  PORT12
- Basic interval timer

<Interrupt>

INTBT

<Initialization>

- PCC ← 3H
- POGA (PORT6 and PORT7 pull-up resistor setting)
- PMGB (PORT4 and PORT5 output mode)
- PMGC (PORT12 output mode)
- PORT4 ← FH
- PORT5 ← FH
- PORT12 ← FH
- BTM (BT interval=1.95 ms)
- INTBT enable

<Start method>

- Call LEDKEY by making an INTBT interrupt.

10.2 SPD

LEDKEY (subroutine)

```
LEDKEY
    Turn off LED display
    Change DIG_D
    (IF:  LED display flag is set)
    [THEN]
        Output LED display
    [ELSE]
        Display DP
    Set chattering data address
    Input key data
    Change input data
    (IF:  chattering data is changed)
    [THEN]
        Store new data
        Set CH_C
        Set CHCT_F
    (IF:  chattering data end)
    [THEN]
        (IF:  chattering is being counted)
        [THEN]
            (IF:  chattering ends four times)
            [THEN]
                Reset CHCT_F
                Set key data address
                Set key work address
                (WHILE:  key data is stored)
                    (IF:  key data changes)
                    [THEN]
                        Set KCHE_F
                    Store data
                    Increment key data address
            [ELSE]
                Increment CH_C
    RET
```

10.3 Program Example

LEDKEY (subroutine)

```
;-------------------------------------------------------------------------
        PUBLIC  LEDKEY,DIG_D,DP_D,LEDST_F,KEY_D,KCHA_F
;-------------------------------------------------------------------------

LED_D   DSEG    0 AT 20H                                   : LED display data area
        DS      4
DP_D:   DS      1                                          ; DP display data pointer
DIG_D:  DS      1                                          ; Digit data pointer
CH_C:   DS      1                                          : Chattering counter
LKFLG:  DS      1                                          : Key flag area
KEY_D:  DS      8                                          : Key data area

KEY_W   DSEG    0 AT 38H                                   : Key work area
        DS      8

KCHA_F  EQU     LKFLG.0                                    ; Key change flag
CHCT_F  EQU     LKFLG.1                                    : Chattering count flag
LEDST_F EQU     LKFLG.2                                    : LED display flag

;-------------------------------------------------------------------------

?DEC    MACRO   P1                                         : Decrement P1
                                           MOV     A,P1
                                           DECS    A
                                           MOV     P1,A
        ENDM

?ROTATE MACRO   P1                                         : Rotate P1 with carry
                                           MOV     A,P1
                                           RORC    A
                                           MOV     P1,A
        ENDM

?TABLE  MACRO   P1,P2                                      : Store table data addressed by P2 in P1
                                           MOVT    XA,P2
                                           MOV     P1,XA
        ENDM

?CALADR MACRO   P1,P2,P3                                   ; Store data calculated in P2 and P3 in P1
                                           MOV     X,#0
                                           MOV     A,P2
                                           MOV     P1,#P3
                                           ADDS    XA,XA
                                           ADDS    P1,XA
        ENDM

?NOT    MACRO   P1,P2                                      ; Store P2 XOR FFH in P1
                                           MOV     P1,P2
                                           MOV     BC,#0FFH
                                           XOR     P1,BC
        ENDM

?SET    MACRO   P1                                         ; SET1 instruction
                                           SET1    P1
        ENDM

?CLR    MACRO   P1                                         ; CLR1 instruction
                                           CLR1    P1
        ENDM

;-------------------------------------------------------------------------

LEDTBL  CSEG    PAGE
;               abcdefgx                                   ; LED segment data
        DB      00000011B                                  ;"0"
        DB      10011111B                                  ;"1"
        DB      00100101B                                  ;"2"
        DB      00001101B                                  ;"3"
        DB      10011001B                                  ;"4"
        DB      01001001B                                  ;"5"
        DB      01000001B                                  ;"6"
        DB      00011111B                                  ;"7"
        DB      00000001B                                  ;"8"
        DB      00001001B                                  ;"9"
        DB      00010001B                                  ;"A"
        DB      11000001B                                  ;"B"
        DB      01100011B                                  ;"C"
        DB      10000101B                                  ;"D"
        DB      01100001B                                  ;"E"
        DB      01110001B                                  ;"F"
```

Phase-out/Discontinued

```
LEDKEY:
PORT4=#0FFH (XA)                        ;Turn off LED display
?SET    CY

if(DIG_D==#0) (A)
    DIG_D=#3 (A)                        ;Change digit data
    ?CLR    CY
else
    ?DEC    DIG_D
endif

?ROTATE PORT12                          ;Decrement PORT12

if_bit(LEDST_F)
    HL=#LED_D                           ;Set table address
    L=DIG_D (A)
    A=@HL
    X=#0
    ?TABLE  PORT4,@PCXA                 ;Set table Data
    L=DIG_D (A)                         ;Set digit data
    L++

    if(DP_D==L) (A)
        ?CLR    PORT4.0                 ;DP display
    endif

endif

?CALADR HL,DIG_D,KEY_W                  ;Calculate chattering data address
?NOT    XA,PORT6                        ;Correct comparison data

if(XA!=@HL)                             ;Does chattering data change?
    XA<->@HL                            ;Store new data
    CH_C=#0FH-4 (A)                     ;Set chattering counter
    ?SET    CHCT_F                      ;Set chattering count flag
endif

if(#KEY_W+6==HL) (XA)                   ;Chattering end?

    if_bit(CHCT_F)                      ;Is chattering count flag set?

        if(CH_C==#0FH) (A)
            ?CLR    CHCT_F
            HL=#KEY_D                   ;Set key data address
            DE=#KEY_W                   ;Set key work address

            for(B=#0;B!=#8;B++)
                A=@DL                   ;Set preceding data address
                if(A!=@HL)              ;Does data change?
                    ?SET    KCHA_F      ;Set key change flag
                endif
                A<->@HL+                ;Store data and increment address
                NOP
            next

        else
        CH_C++                          ;Increment chattering counter
        endif

    endif

endif

RET

END
```

Assembly list is given in A.7.

APPENDIX A ASSEMBLY LISTS

A.1 Scale Generation Program

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                 89/01/10 21:39:18 PAGE :   1
 **                                                               **

 COMMAND      :                       (COMMAND FILE :                 )
 B:SOUND.ASM MOD=516

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    1                          ;*******************************************************
    2                          ;*                         Scale generation program
    3                          ;*
    4                          ;*******************************************************
    5                          ;
    6                                  PUBLIC  SOUND
    7                          ;
    8                          ;*******************************************************
    9                          ;
   10                          ;
   11                          ;*******************************************************
   12                          ;
   13 ----                     SOUND_D DSEG    0 AT 4FH                                         ; Scale data area
   14 004F                             DS      1H
   15                          ;
   16 ----                     SOUTBL  CSEG    PAGE                                             ; Timer pulse data table
   17 0000    00                       DB      0                                                ; REST
   18 000J    FA                       DB      0FAH                                             ; DO
   19 0002    EB                       DB      0EBH                                             ; DO#
   20 0003    DE                       DB      0DEH                                             ; RE
   21 0004    D3                       DB      0D3H                                             ; RE#
   22 0005    C7                       DB      0C7H                                             ; MI
   23 0006    BC                       DB      0BCH                                             ; FA
   24 0007    B1                       DB      0B1H                                             ; FA#
   25 0008    A6                       DB      0A6H                                             ; SO
   26 0009    9E                       DB      9EH                                              ; SO#
   27 000A    94                       DB      94H                                              ; LA
   28 000B    8C                       DB      8CH                                              ; LA#
   29 000C    84                       DB      84H                                              ; SI
   30 000D    7C                       DB      7CH                                              ; DO
   31 000E    75                       DB      75H                                              ; DO#
   32 000F    6F                       DB      6FH                                              ; RE
   33                          ;
   34 0010                     SOUND:
   35                          ;?CLR   IET0                                                     ; Disable INTTO interrupt
   36 0010    9C9C                                                   CLR1    IET0
   37                          ;?CLR   MBE
   38 0012    9C90                                                   CLR1    MBE
   39                          ;?CLR   PORT2.0
   40 0014    9CC2                                                   CLR1    PORT2.0
   41                          ;
   42                          ;if( SOUND_D!=#0 ) (A)                                           ; Scale data?
   43 0016    A34F                                                   MOV     A,SOUND_D
   44 0018    9A00                                                   SKE     A,#0
   45 001A    01                                                     BR      ??1
   46 001B    0E                                                     BR      ??2
   47 001C                                                       ??1:
   48                          ;    X=#0
   49 001C    9A09                                                   MOV     X,#0
   50                          ;    A=SOUND_D
   51 001E    A34F                                                   MOV     A,SOUND_D
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                        89/01/10 21:39:18 PAGE :   2

    **                                                               **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    52                          ;    ?TABLE  TMOD0,@PCXA                                         ; Store timer pulse data
    53 0020   D0                                                                 MOVT    XA,@PCXA
    54 0021   92A6                                                               MOV     TMOD0,XA
    55                          ;    TM0=#01111100B (XA)                                                  ; Set timer
    56 0023   897C                                                               MOV     XA,#01111100B
    57 0025   92A0                                                               MOV     TM0,XA
    58                          ;    ?SET    TOE0                                                ; Enable pulse output
    59 0027   B5A2                                                               SET1    TOE0
    60                          ;else
    61 0029   02                                                                 BR      ??3
    62 002A                                                              ??2:
    63                          ;    ?CLR    TOE0                                                ; Disable pulse output
    64 002A   B4A2                                                               CLR1    TOE0
    65                          ;endif
    66 002C                                                              ??3:
    67                          ;
    68 002C   EE                RET
    69                          ;
    70                          END

   TARGET CHIP : UPD75516
   STACK SIZE = 0000H

   ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.2 A/D Conversion Program

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                         89/01/10 21:29:13 PAGE :   1

   **                                                                  **

 COMMAND      :                             (COMMAND FILE :                 )
 B:ADCNV.ASM MOD=516

  STNO ADRS R OBJECT   IC MAC     SOURCE STATEMENT

     1                            ;******************************************************************************
     2                            ;*                          A/D conversion program
     3                            ;*
     4                            ;******************************************************************************
     5                            ;
     6                                   PUBLIC  ACONV_D
     7                            ;
     8                            ;******************************************************************************
     9                            ;
    10                            ;
    11                            ;******************************************************************************
    12                            ;
    13 ----                       ACHN_P  DSEG    0 AT 47H                                     ; Analog channel pointer
    14 0047                               DS      1
    15 0048                       ACONV_D:DS      2                                            ; A/D conversion data area
    16                            ;
    17 ----                       ADCNV   CSEG    INBLOCK
    18                            ;
    19                            ;?CLR   MBE
    20 0000   9C90                                                        CLR1    MBE
    21                            ;
    22                            ;while(forever)
    23 0002                                                        ??1:
    24                            ;
    25                            ;    if_bit(EOC)                                                      ; Does conversion end?
    26 0002   A7D8                                                        SKT     EOC
    27 0004   0A                                                          BR      ??2
    28                            ;       ?STORE  ADM,ACHN_P,#8                                  ; Set data and start conversion
    29 0005   A347                                                        MOV     A,ACHN_P
    30 0007   9937                                                        AND     A,#7
    31 0009   9A89                                                        MOV     X,#8
    32 000B   D9                                                          XCH     A,X
    33 000C   92D8                                                        MOV     ADM,XA
    34                            ;       break
    35 000E   01                                                          BR      ??3
    36                            ;    endif
    37 000F                                                        ??2:
    38                            ;
    39                            ;endw
    40 000F   F2                                                          BR      ??1
    41 0010                                                        ??3:
    42                            ;
    43                            ;?WAIT
    44 0010   60                                                          NOP
    45 0011   60                                                          NOP
    46 0012   60                                                          NOP
    47 0013   60                                                          NOP
    48                            ;
    49                            ;while(forever)
    50 0014                                                        ??4:
    51                            ;
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                       89/01/10 21:29:13 PAGE :   2

   **                                             **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

   52                          ;    if_bit(EOC)
   53 0014   A7D8                                           SKT     EOC        ;Does conversion terminate?
   54 0016   05                                             BR      ??5
   55                          ;       ACONV_D=SA (XA)
   56 0017   A2DA                                           MOV     XA,SA      ;Store conversion result data
   57 0019   9248                                           MOV     ACONV_D,XA
   58                          ;       break
   59 001B   01                                             BR      ??6
   60                          ;    endif
   61 001C                                          ??5:
   62                          ;
   63                          ;endw
   64 001C   F7                                             BR      ??4
   65 001D                                          ??6:
   66                          ;
   67 001D   EE                RET
   68                          ;
   69                          END

 TARGET CHIP : UPD75516
 STACK SIZE = 0000H

 ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.3 Analog Key Input Program

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                         89/01/10 21:34:29 PAGE :   1

   **                                                   **

 COMMAND      :                    (COMMAND FILE :              )
 B:ANKEY.ASM MOD=516

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    1                          ;
    2                          ;*******************************************************************************
    3                          ;*                          Analog key input program
    4                          ;*
    5                          ;*******************************************************************************
    6                          ;
    7                                 PUBLIC  AKCHA_F
    8                          ;
    9 ----                     AKEY_D  DSEG    0 AT 40H                                ; Analog key data area
   10 0040                             DS      2
   11 0042                     AKEY_W: DS      2                                       ; Analog key work area
   12 0044                     AKFLG:  DS      1                                       ; Analog flag area
   13 0045                     ACH_C:  DS      1                                       ; Analog chattering counter
   14 0046                     CMP_W:  DS      1                                       ; Work area
   15                          ;
   16        (0110)            AKCHA_F EQU     AKFLG.0                                 ; Analog key change flag
   17        (0111)            ACHCT_F EQU     AKFLG.1                                 ; Analog chattering count flag
   18                          ;
   19                          ;*******************************************************************************
   20                          ;
   21                          ;
   22                          ;*******************************************************************************
   23 ----                     ANKEY   CSEG    INBLOCK
   24                          ;
   25                          ;while_bit(!EOC)                                                   ; Does conversion end?
   26 0000                                                    ??1:
   27 0000   A6D8                                                       SKF     EOC
   28 0002   01                                                         BR      ??2
   29                          ;endw
   30 0003   FC                                                         BR      ??1
   31 0004                                                    ??2:
   32                          ;
   33                          ;XA=SA                                                           ; Store conversion result data
   34 0004   A2DA                                                       MOV     XA,SA
   35                          ;BC=XA
   36 0006   AA56                                                       MOV     BC,XA
   37                          ;?SET   SOC                                                      ; Start the next conversion
   38 0008   B5D8                                                       SET1    SOC
   39                          ;A<->X                                                           ; Correct key data
   40 000A   D9                                                         XCH     A,X
   41                          ;BC=XA
   42 000B   AA56                                                       MOV     BC,XA
   43                          ;A<->X
   44 000D   D9                                                         XCH     A,X
   45                          ;B=#0
   46 000E   9A0F                                                       MOV     B,#0
   47                          ;?CMPHL
   48 0010   8B46                                                       MOV     HL,#CMP_W
   49 0012   E8                                                         MOV     @HL,A
   50                          ;
   51                          ;if(@HL>#7) (A)
```

Phase-out/Discontinued

```
   **                                                                  **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

   52 0013   77                                                                 MOV      A,#7
   53 0014   A8                                                                 SUBS     A,@HL
   54 0015   02                                                                 BR       ??3
   55                              ;    BC++
   56 0016   8E                                                                 INCS     BC
   57 0017   60                        NOP
   58                              ;endif
   59 0018                                                           ??3:
   60                              ;
   61                              ;if(AKEY_W!=BC) (XA)                                           ;Does data change?
   62 0018   A242                                                               MOV      XA,AKEY_W
   63 001A   AA4E                                                               SKE      XA,BC
   64 001C   01                                                                 BR       ??4
   65 001D   0B                                                                 BR       ??5
   66 001E                                                           ??4:
   67                              ;    AKEY_W=BC (XA)                                             ;Set new data
   68 001E   AA5E                                                               MOV      XA,BC
   69 0020   9242                                                               MOV      AKEY_W,XA
   70                              ;    ACH_C=#(0FH-10) (A)                                              ; Set chattering counter
   71 0022   75                                                                 MOV      A,#(0FH-10)
   72 0023   9345                                                               MOV      ACH_C,A
   73                              ;    ?SET    ACHCT_F
   74 0025   9544                                                               SET1     ACHCT_F
   75                              ;else
   76 0027 R 5047                                                               BR       ??6
   77 0029                                                           ??5:
   78                              ;
   79                              ;    if_bit(ACHCT_F)
   80 0029   9744                                                               SKT      ACHCT_F
   81 002B R 5047                                                               BR       ??7
   82                              ;
   83                              ;        if(ACH_C!=#0FH) (A)
   84 002D   A345                                                               MOV      A,ACH_C
   85 002F   9AF0                                                               SKE      A,#0FH
   86 0031   01                                                                 BR       ??8
   87 0032   04                                                                 BR       ??9
   88 0033                                                           ??8:
   89                              ;                ACH_C++                                              ; Increment counter
   90 0033   8245                                                               INCS     ACH_C
   91                              ;        else
   92 0035 R 5047                                                               BR       ??10
   93 0037                                                           ??9:
   94                              ;            ?CLR    ACHCT_F                                       ;Does count end?
   95 0037   9444                                                               CLR1     ACHCT_F
   96                              ;            HL=#AKEY_D
   97 0039   8B40                                                               MOV      HL,#AKEY_D
   98                              ;
   99                              ;            if(AKEY_W!=@HL) (XA)
  100 003B   A242                                                               MOV      XA,AKEY_W
  101 003D   AA19                                                               SKE      XA,@HL
  102 003F   01                                                                 BR       ??11
  103 0040   02                                                                 BR       ??12
  104 0041                                                           ??11:
  105                              ;                ?SET    AKCHA_F                                   ; Set key change flag
  **                                                              **

STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

 106 0041   8544                                                              SET1    AKCHA_F
 107                          ;           endif
 108 0043                                                             ??12:
 109                          ;
 110                          ;           XA=AKEY_W
 111 0043   A242                                                              MOV     XA,AKEY_W
 112                          ;           @HL=XA
 113 0045   AA10                                                              MOV     @HL,XA
 114                          ;       endif
 115 0047                                                             ??10:
 116                          ;
 117                          ;    endif
 118 0047                                                             ??7:
 119                          ;
 120                          ;endif
 121 0047                                                             ??6:
 122                          ;
 123 0047   EE                RET
 124                          ;
 125                          END

TARGET CHIP : UPD75516
STACK SIZE = 0000H

ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.4 Communication Program with EEPROM

```
UCON-75X FAMILY ASSEMBLER V3.5                                                                      89/01/10 21:34:48 PAGE :   1
    **                                                          **

 COMMAND      :                           (COMMAND FILE :             )
 B:EEPROM.ASM MOD=516

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

     1                          ;******************************************************************************
     2                          ;*                    Communication program with  REPROM
     3                          ;*
     4                          ;******************************************************************************
     5                          ;
     6                                  PUBLIC  E2WD_D,E2RA_D,E2RD_D,E2RW_F
     7                          ;
     8 ----                     E2WA_D  DSEG    0 AT 30H                                     ;Write address area
     9 0030                             DS      2
    10 0032                     E2WD_D: DS      2                                            ;Write data area
    11 0034                     E2RA_D: DS      2                                            ;Read address area
    12 0036                     E2RD_D: DS      2                                            ;Read data area
    13                          ;
    14        (009F)            E2RW_F  EQU     27H.3                                        ;Read/write flag
    15        (3FC9)            CS_O    EQU     PORT2.1                                      ;CS for uPD6252
    16                          ;
    17                          ;******************************************************************************
    18                          ;
    19                          ;
    20                          ;******************************************************************************
    21 ----                     EEPROM  CSEG    INBLOCK
    22                          ;
    23                          ;if_bit(E2RW_F)
    24 0000   B727                                                      SKT     E2RW_F
    25 0002 R 502C                                                      BR      ??1
    26                          ;
    27                          ;    while(forever)                                          ;Read mode
    28 0004                                                    ??2:
    29                          ;        ?CLR    MBE
    30 0004   9C90                                                      CLR1    MBE
    31                          ;       ?SET    CS_O                                         ;Start communication
    32 0006   9DD2                                                      SET1    CS_O
    33                          ;       ?SIO2   #0C0H                                        ;Transmit random read command
    34 0008   89C0                                                      MOV     XA,#0C0H
    35 000A R 404F                                                      CALLF   !SIOSUB
    36                          ;       ?SIO2   #0FFH                                        ;Receive WB flag
    37 000C   89FF                                                      MOV     XA,#0FFH
    38 000E R 404F                                                      CALLF   !SIOSUB
    39                          ;       HL=#0
    40 0010   8B00                                                      MOV     HL,#0
    41                          ;
    42                          ;       if(SIO0==HL) (XA)                                    ; BUSY?
    43 0012   A2E4                                                      MOV     XA,SIO0
    44 0014   AA4A                                                      SKE     XA,HL
    45 0016   0F                                                        BR      ??3
    46                          ;           ?SIO2   E2RA_D                                   ;Transmit read address
    47 0017   A234                                                      MOV     XA,E2RA_D
    48 0019 R 404F                                                      CALLF   !SIOSUB
    49                          ;           ?SIO2   #0FFH                                    ;Receive read data
    50 001B   89FF                                                      MOV     XA,#0FFH
    51 001D R 404F                                                      CALLF   !SIOSUB
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                         89/01/10 21:34:48 PAGE :   2

   **                                                                   **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    52                          ;          E2RD_D=SIO0 (XA)                                     ; Store read data
    53 001F   A2E4                                                        MOV     XA,SIO0
    54 0021   9236                                                        MOV     E2RD_D,XA
    55                          ;          ?CLR    CS_O
    56 0023   9CD2                                                        CLR1    CS_O
    57                          ;          break
    58 0025   04                                                          BR      ??4
    59                          ;              endif
    60 0026                                                        ??3:
    61                          ;
    62                          ;      ?CLR    CS_O
    63 0026   9CD2                                                        CLR1    CS_O
    64                          ;   endw
    65 0028 R 5004                                                        BR      ??2
    66 002A                                                        ??4:
    67                          ;
    68                          ;else
    69 002A R 504E                                                        BR      ??5
    70 002C                                                        ??1:
    71                          ;
    72                          ;   while(forever)                                              ; Write mode
    73 002C                                                        ??6:
    74                          ;      ?CLR    MBE
    75 002C   9C90                                                        CLR1    MBE
    76                          ;      ?SET    CS_O                                             ; Start communication
    77 002E   9DD2                                                        SET1    CS_O
    78                          ;      ?SIO2   #0                                               ; Transmit random write command
    79 0030   8900                                                        MOV     XA,#0
    80 0032 R 404F                                                        CALLF   !SIOSUB
    81                          ;      ?SIO2   #0FFH                                            ; Receive WB flag
    82 0034   89FF                                                        MOV     XA,#0FFH
    83 0036 R 404F                                                        CALLF   !SIOSUB
    84                          ;      HL=#0
    85 0038   8B00                                                        MOV     HL,#0
    86                          ;
    87                          ;      if(SIO0==HL) (XA)
    88 003A   A2E4                                                        MOV     XA,SIO0
    89 003C   AA4A                                                        SKE     XA,HL
    90 003E   0B                                                          BR      ??7
    91                          ;                  ?SIO2   E2WA_D                                   ; Transmit write address
    92 003F   A230                                                        MOV     XA,E2WA_D
    93 0041 R 404F                                                        CALLF   !SIOSUB
    94                          ;          ?SIO2   E2WD_D                                       ; Transmit write data
    95 0043   A232                                                        MOV     XA,E2WD_D
    96 0045 R 404F                                                        CALLF   !SIOSUB
    97                          ;          ?CLR    CS_O
    98 0047   9CD2                                                        CLR1    CS_O
    99                          ;          break
   100 0049   04                                                          BR      ??8
   101                          ;      endif
   102 004A                                                        ??7:
   103                          ;
   104                          ;      ?CLR    CS_O
   105 004A   9CD2                                                        CLR1    CS_O
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                              89/01/10 21:34:48 PAGE :   3
    **                                                         **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

  106                          ;    endw
  107 004C R 502C                                                         BR      ??6
  108 004E                                                          ??8:
  109                          ;
  110                          ;endif
  111 004E                                                          ??5:
  112                          ;
  113 004E   EE                RET
  114                          ;
  115                          ;**********************************************************************
  116                          ;*                       SIOSUB
  117                          ;**********************************************************************
  118 004F                     SIOSUB:
  119                          ;SIO0=XA                                                          ; Transfer data
  120 004F   92E4                                                         MOV     SIO0,XA
  121                          ;
  122                          ;repeat
  123 0051                                                          ??9:
  124                          ;until_bit(IRQCSI0)                                        ; Wait for communication to end
  125 0051   BF8D                                                         SKT     IRQCSI0
  126 0053   FD                                                           BR      ??9
  127                          ;
  128                          ;?CLR    IRQCSI0
  129 0054   9C8D                                                         CLR1    IRQCSI0
  130 0056   EE                RET
  131                          ;
  132                          END

 TARGET CHIP : UPD75516
 STACK SIZE = 0000H

 ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.5 SBI Communication Program

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                    89/01/10 21:36:46 PAGE :   1
   **                                                                 **

 COMMAND      :                          (COMMAND FILE :                )
 B:SBITRN.ASM MOD=516

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    1                          ;*****************************************************************************
    2                          ;*                                        SBI communication program
    3                          ;*
    4                          ;*****************************************************************************
    5 0008 R 0000                      VENT4   MBE=0,RBE=0,SBIO
    6 000A R 0000                      VENT5   MBE=0,RBE=0,ACKWOP
    7                          ;
    8                                  PUBLIC  SBRE_D,SBI_P,ERR_F
    9                          ;
   10 ----                     SBTR_D  DSEG    0 AT 50H                                        ;Send data area
   11 0050                             DS      2
   12 0052                     SBRE_D: DS      2                                               ;Receive data area
   13 0054                     SBI_P:  DS      1                                               ;Communication code pointer
   14 0055                     SBIFLG: DS      1                                               ;SBI flag area
   15 0056                     ACKW_C: DS      1                                               ;ACK wait counter
   16                          ;
   17        (0154)            ERR_F   EQU     SBIFLG.0                                        ;Error flag
   18        (0155)            BUSY_F  EQU     SBIFLG.1                                        ;BUSY flag
   19        (0156)            ACKW_F  EQU     SBIFLG.2                                        ;ACK wait flag
   20                          ;
   21        (3FC2)            SBI_O   EQU     PORT0.2
   22                          ;
   23 ----                     SBITRN  CSEG    INBLOCK
   24                          ;
   25                          ;*****************************************************************************
   26                          ;
   27                          ;*****************************************************************************
   28                          ;
   29                          ;while(forever)
   30 0000                                                      ??1:
   31                          ;
   32                          ;    if_bit(SBI_O)                                                      ;Is bus BUSY?
   33 0000   BFE0                                                       SKT     SBI_O
   34 0002 R 502A                                                       BR      ??2
   35                          ;
   36                          ;        switch(SBI_P)
   37 0004   A354                                                       MOV     A,SBI_P
   38                          ;            case 0:
   39 0006   9A00                                                       SKE     A,#0
   40 0008   05                                                         BR      ??3
   41                          ;                ?SET    CMDT                                            ;Output command signal
   42 0009   95E2                                                       SET1    CMDT
   43                          ;                ?SET    RELT                                            ;Output bus release signal
   44 000B   85E2                                                       SET1    RELT
   45                          ;                A=#1
   46 000D   71                                                         MOV     A,#1
   47                          ;            case 1:
   48 000E                                                      ??3:
   49 000E   9A10                                                       SKE     A,#1
   50 0010   03                                                         BR      ??4
   51                          ;                ?SET    CMDT                                            ;Output command signal
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                  89/01/10 21:36:46 PAGE :   2

  **                                                                       **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

  52 0011   95E2                                                     SET1    CMDT
  53                           ;              A=#2
  54 0013   72                                                       MOV     A,#2
  55                           ;          case 2:
  56 0014                                                   ??4:
  57 0014   9A20                                                     SKE     A,#2
  58 0016   03                                                       BR      ??5
  59                           ;              XA=SBTR_D                        ;
  60 0017   A250                                                     MOV     XA,SBTR_D Set send data
  61                           ;              break
  62 0019   05                                                       BR      ??6
  63                           ;          case 3:
  64 001A                                                   ??5:
  65 001A   9A30                                                     SKE     A,#3
  66 001C   02                                                       BR      ??7
  67                           ;              XA=#0FFH                         ;Prepare for data reception
  68 001D   89FF                                                     MOV     XA,#0FFH
  69                           ;      ends
  70 001F                                                   ??7:
  71 001F                                                   ??6:
  72                           ;
  73                           ;      ?SET    BUSY_F
  74 001F   9555                                                     SET1    BUSY_F
  75                           ;      SVA=XA
  76 0021   92E6                                                     MOV     SVA,XA
  77                           ;      SIO0=XA                                 ;Start communication
  78 0023   92E4                                                     MOV     SIO0,XA
  79                           ;
  80                           ;      while_bit(BUSY_F)                       ;Wait for communication to end
  81 0025                                                   ??8:
  82 0025   9755                                                     SKT     BUSY_F
  83 0027   01                                                       BR      ??9
  84                           ;      endw
  85 0028   FC                                                       BR      ??8
  86 0029                                                   ??9:
  87                           ;
  88                           ;       break
  89 0029   02                                                       BR      ??10
  90                           ;   endif
  91 002A                                                   ??2:
  92                           ;
  93                           ;endw
  94 002A R 5000                                                     BR      ??1
  95 002C                                                   ??10:
  96                           ;
  97 002C   EE                 RET
  98                           ;
  99                           ;*****************************************************************************
 100                           ;*                      INTCSIO
 101                           ;*****************************************************************************
 102 ----                      SBIO    CSEG    INBLOCK
 103                           ;
 104                           ;ACKW_C=#0 (A)                                 ;Reset ACK wait counter
 105 0000   70                                                       MOV     A,#0
```

```
   **                                                                            **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

  106 0001   9356                                                              MOV      ACKW_C,A
  107                          ;
  108                          ;if(SBI_P!=#3) (A)                                                 ;Transmit mode?
  109 0003   A354                                                              MOV      A,SBI_P
  110 0005   9A30                                                              SKE      A,#3
  111 0007   02                                                                BR       ??11
  112 0008 R 501D                                                              BR       ??12
  113 000A                                                             ??11:
  114                          ;
  115                          ;    if_bit(ACKD)                                                 ;ACK signal?
  116 000A   A7E3                                                              SKT      ACKD
  117 000C   0D                                                                BR       ??13
  118                          ;
  119                          ;        if_bit(COI)                                              ;Is COI set?
  120 000D   A7E1                                                              SKT      COI
  121 000F   03                                                                BR       ??14
  122                          ;             ?CLR    ERR_F
  123 0010   8455                                                              CLR1     ERR_F
  124                          ;        else
  125 0012   04                                                                BR       ??15
  126 0013                                                             ??14:
  127                          ;             ?SET    ERR_F
  128 0013   8555                                                              SET1     ERR_F
  129                          ;             ?CLR    BUSY_F
  130 0015   9455                                                              CLR1     BUSY_F
  131                          ;        endif
  132 0017                                                             ??15:
  133                          ;
  134                          ;        ?CLR    ACKW_F
  135 0017   A455                                                              CLR1     ACKW_F
  136                          ;    else
  137 0019   02                                                                BR       ??16
  138 001A                                                             ??13:
  139                          ;        ?SET    ACKW_F
  140 001A   A555                                                              SET1     ACKW_F
  141                          ;    endif
  142 001C                                                             ??16:
  143                          ;
  144                          ;else
  145 001C   0C                                                                BR       ??17
  146 001D                                                             ??12:
  147                          ;    ?SET    ACKT                                                 ;Reception mode? Transmission
  148 001D   85E3                                                              SET1     ACKT
  149                          ;    SBRE_D=SIO0 (XA)                                             ;Store receive data
  150 001F   A2E4                                                              MOV      XA,SIO0
  151 0021   9252                                                              MOV      SBRE_D,XA
  152                          ;    ?CLR    ERR_F
  153 0023   8455                                                              CLR1     ERR_F
  154                          ;    ?CLR    BUSY_F
  155 0025   9455                                                              CLR1     BUSY_F
  156                          ;    ?CLR    ACKW_F
  157 0027   A455                                                              CLR1     ACKW_F
  158                          ;endif
  159 0029                                                             ??17:
```

```
  **                                                    **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

  160                          ;
  161 0029    EF               RETI
  162                          ;
  163                          ;******************************************************************************
  164                          ;*                         INTTO
  165                          ;******************************************************************************
  166 ----                     ACKWOP  CSEG    INBLOCK
  167                          ;
  168                          ;if_bit(ACKW_F)                                                 ; ACK signal wait?
  169 0000    A755                                                      SKT     ACKW_F
  170 0002 R  5028                                                      BR      ??18
  171                          ;
  172                          ;    if_bit(!ACKD)                                              ; Check ACK signal
  173 0004    A6E3                                                      SKF     ACKD
  174 0006 R  501C                                                      BR      ??19
  175                          ;
  176                          ;       if(ACKW_C==#0FH) (A)
  177 0008    A356                                                      MOV     A,ACKW_C
  178 000A    9AF0                                                      SKE     A,#0FH
  179 000C    0C                                                        BR      ??20
  180                          ;           ACKW_C=#0 (A)
  181 000D    70                                                        MOV     A,#0
  182 000E    9356                                                      MOV     ACKW_C,A
  183                          ;           ?SET    ACKT                                        ; Over wait time
  184 0010    85E3                                                      SET1    ACKT
  185                          ;           ?SET    ERR_F                                       ; Error
  186 0012    8555                                                      SET1    ERR_F
  187                          ;           ?CLR    ACKW_F
  188 0014    A455                                                      CLR1    ACKW_F
  189                          ;           ?CLR    BUSY_F
  190 0016    9455                                                      CLR1    BUSY_F
  191                          ;       else
  192 0018    02                                                        BR      ??21
  193 0019                                                      ??20:
  194                          ;           ACKW_C++
  195 0019    8256                                                      INCS    ACKW_C
  196                          ;       endif
  197 001B                                                      ??21:
  198                          ;
  199                          ;    else
  200 001B    0C                                                        BR      ??22
  201 001C                                                      ??19:
  202                          ;
  203                          ;        if_bit(COI)                                            ; Check bus error
  204 001C    A7E1                                                      SKT     COI
  205 001E    03                                                        BR      ??23
  206                          ;           ?CLR    ERR_F
  207 001F    8455                                                      CLR1    ERR_F
  208                          ;       else
  209 0021    02                                                        BR      ??24
  210 0022                                                      ??23:
  211                          ;           ?SET    ERR_F                                       ; Error
  212 0022    8555                                                      SET1    ERR_F
  213                          ;       endif
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                       89/01/10 21:36:46 PAGE :   5

    **                                                              **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

   214 0024                                                             ??24:
   215                          ;           ?CLR    ACKW_F
   216 0024   A455                                                              CLR1    ACKW_F
   217                          ;           ?CLR    BUSY_F
   218 0026   9455                                                              CLR1    BUSY_F
   219                          ;
   220                          ;    endif
   221 0028                                                             ??22:
   222                          ;
   223                          ;endif
   224 0028                                                             ??18:
   225                          ;
   226 0028   EF                RETI
   227                          ;
   228                          END

 TARGET CHIP : UPD75516
 STACK SIZE = 0000H

 ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.6 Notation Adjustment Program

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                    89/01/10 21:39:36 PAGE :   1

   **                                                  **

 COMMAND      :                   (COMMAND FILE :              )
 B:SINSU.ASM MOD=516

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

     1                          ;****************************************************************************
     2                          ;*                  Notation adjustment program
     3                          ;*
     4                          ;****************************************************************************
     5                          ;
     6                                  PUBLIC  SIN_P,SIN_W
     7                          ;
     8 ----                     SIN_D   DSEG    0 AT 4AH                                  ;Notation data area
     9 004A                             DS      2
    10 004C                     SIN_P:  DS      1                                         ;Notation code pointer
    11 004D                     SIN_W:  DS      1                                         ;Notation work area
    12                          ;
    13                          ;****************************************************************************
    14                          ;****************************************************************************
    15                          ;
    16 ----                     SINSU   CSEG    INBLOCK
    17                          ;
    18                          ;SIN_D=#0 (XA)
    19 0000     8900                                                    MOV     XA,#0
    20 0002     924A                                                    MOV     SIN_D,XA
    21                          ;
    22                          ;switch(SIN_P)
    23 0004     A34C                                                    MOV     A,SIN_P
    24                          ;     case 0:                                                     ;Four count adjustment processing
    25 0006     9A00                                                    SKE     A,#0
    26 0008 R 501C                                                      BR      ??1
    27                          ;         A=SIN_W
    28 000A     A34D                                                    MOV     A,SIN_W
    29                          ;         A &= #1100B
    30 000C     993C                                                    AND     A,#1100B
    31                          ;         ?SHIFT2 A                                               ;Shift data right two bits
    32 000E     E6                                                      CLR1    CY
    33 000F     98                                                      RORC    A
    34 0010     E6                                                      CLR1    CY
    35 0011     98                                                      RORC    A
    36                          ;         (SIN_D+1)=A                                             ;Store high-order data
    37 0012     934B                                                    MOV     (SIN_D+1),A
    38                          ;         A=SIN_W
    39 0014     A34D                                                    MOV     A,SIN_W
    40                          ;         A &= #0011B
    41 0016     9933                                                    AND     A,#0011B
    42                          ;         SIN_D=A                                                 ;Store low-order data
    43 0018     934A                                                    MOV     SIN_D,A
    44                          ;         break
    45 001A R 507A                                                      BR      ??2
    46                          ;     case 1:                                                     ;Six count adjustment processing
    47 001C                                                     ??1:
    48 001C     9A10                                                    SKE     A,#1
    49 001E R 5038                                                      BR      ??3
    50                          ;         HL=#SIN_W
    51 0020     8B4D                                                    MOV     HL,#SIN_W
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                    89/01/10 21:39:36 PAGE :   2

    **                                                          **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    52                          ;       X=#0
    53 0022   9A09                                                          MOV     X,#0
    54                          ;       A=@HL
    55 0024   E1                                                            MOV     A,@HL
    56                          ;       BC=XA
    57 0025   AA56                                                          MOV     BC,XA
    58                          ;
    59                          ;       while(#5<@HL) (A)
    60 0027                                                          ??4:
    61 0027   75                                                            MOV     A,#5
    62 0028   A8                                                            SUBS    A,@HL
    63 0029   08                                                            BR      ??5
    64                          ;           A=@HL
    65 002A   E1                                                            MOV     A,@HL
    66                          ;           A+=#(16-6)
    67 002B   6A                                                            ADDS    A,#(16-6)
    68 002C   60                            NOP
    69                          ;           X++
    70 002D   C1                                                            INCS    X
    71                          ;           @HL=A
    72 002E   E8                                                            MOV     @HL,A
    73                          ;           BC=XA
    74 002F   AA56                                                          MOV     BC,XA
    75                          ;       endw
    76 0031   F5                                                            BR      ??4
    77 0032                                                          ??5:
    78                          ;
    79                          ;       XA=BC
    80 0032   AA5E                                                          MOV     XA,BC
    81                          ;       SIN_D=XA                                             ;Store adjustment result
    82 0034   924A                                                          MOV     SIN_D,XA
    83                          ;       break
    84 0036 R 507A                                                          BR      ??2
    85                          ;    case 2:                                                 ;Octal adjustment processing
    86 0038                                                          ??3:
    87 0038   9A20                                                          SKE     A,#2
    88 003A R 504C                                                          BR      ??6
    89                          ;       A=SIN_W
    90 003C   A34D                                                          MOV     A,SIN_W
    91                          ;
    92                          ;       if_bit(SIN_W.3)                                      ;Most significant Bit=1?
    93 003E   B74D                                                          SKT     SIN_W.3
    94 0040   03                                                            BR      ??7
    95                          ;           (SIN_D+1)=#1 (A)                                 ;Store high-order data
    96 0041   71                                                            MOV     A,#1
    97 0042   934B                                                          MOV     (SIN_D+1),A
    98                          ;       endif
    99 0044                                                          ??7:
   100                          ;
   101                          ;       A=SIN_W
   102 0044   A34D                                                          MOV     A,SIN_W
   103                          ;       A &= #0111B
   104 0046   9937                                                          AND     A,#0111B
   105                          ;       SIN_D=A                                              ;Store low-order data
```

```
    **                                                         **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

   106 0048   934A                                                   MOV     SIN_D,A
   107                          ;       break
   108 004A R 507A                                                   BR      ??2
   109                          ;    case 3:                                     ; Decimal adjustment processing
   110 004C                                                    ??6:
   111 004C   9A30                                                   SKE     A,#3
   112 004E R 5062                                                   BR      ??8
   113 0050   E6                        CLR1     CY
   114                          ;       A=#6
   115 0051   76                                                     MOV     A,#6
   116                          ;       HL=#SIN_W
   117 0052   8B4D                                                   MOV     HL,#SIN_W
   118 0054   A9                        ADDC     A,@HL                ;ADDC   A,@HL
   119                          ;       A+=#10
   120 0055   6A                                                     ADDS    A,#10
   121                          ;       A<->X
   122 0056   D9                                                     XCH     A,X
   123                          ;
   124                          ;       if_bit(CY)
   125 0057   D7                                                     SKT     CY
   126 0058   03                                                     BR      ??9
   127                          ;           (SIN_D+1)=#1 (A)                     ; Store low-order data
   128 0059   71                                                     MOV     A,#1
   129 005A   934B                                                   MOV     (SIN_D+1),A
   130                          ;       endif
   131 005C                                                    ??9:
   132                          ;
   133                          ;       SIN_D=X (A)                              ; Store high-order data
   134 005C   9979                                                   MOV     A,X
   135 005E   934A                                                   MOV     SIN_D,A
   136                          ;       break
   137 0060 R 507A                                                   BR      ??2
   138                          ;    case 4:                                     ; Twelve count adjustment processing
   139 0062                                                    ??8:
   140 0062   9A40                                                   SKE     A,#4
   141 0064 R 507A                                                   BR      ??10
   142                          ;       HL=#SIN_W
   143 0066   8B4D                                                   MOV     HL,#SIN_W
   144                          ;
   145                          ;       if(#11<@HL) (A)
   146 0068   7B                                                     MOV     A,#11
   147 0069   A8                                                     SUBS    A,@HL
   148 006A   09                                                     BR      ??11
   149                          ;           (SIN_D+1)=#1 (A)                     ; Store high-order data (DATA >11)
   150 006B   71                                                     MOV     A,#1
   151 006C   934B                                                   MOV     (SIN_D+1),A
   152                          ;           A=#16-12
   153 006E   74                                                     MOV     A,#16-12
   154                          ;           A+=@HL
   155 006F   D2                                                     ADDS    A,@HL
   156 0070   60                            NOP
   157                          ;           SIN_D=A                              ; Store low-order data
   158 0071   934A                                                   MOV     SIN_D,A
   159                          ;       else
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                                   89/01/10 21:39:36 PAGE :    4

   **                                                               **

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

  160 0073   06                                                                 BR      ??12
  161 0074                                                              ??11:
  162                          ;            (SIN_D+1)=#0 (A)                                    ; Store high-order data (DATA <12)
  163 0074   70                                                                 MOV     A,#0
  164 0075   934B                                                               MOV     (SIN_D+1),A
  165                          ;            SIN_D=@HL (A)                                       ; Store low-order data
  166 0077   E1                                                                 MOV     A,@HL
  167 0078   934A                                                               MOV     SIN_D,A
  168                          ;        endif
  169 007A                                                              ??12:
  170                          ;
  171                          ;ends
  172 007A                                                              ??10:
  173 007A                                                              ??2:
  174                          ;
  175 007A   EE                RET
  176                          ;
  177                          END

 TARGET CHIP : UPD75516
 STACK SIZE = 0000H

 ASSEMBLY COMPLETE,  NO ERROR FOUND
```

A.7 LED Display and Key Input Program

```
UCON-75X FAMILY ASSEMBLER V3.5                                                                  89/01/10 21:37:11 PAGE :   1

   **                                                            **

 COMMAND                                 (COMMAND FILE :                )
 B:LEDKEY.ASM MOD=516

 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    1                          ;
    2                          ;*****************************************************************************
    3                          ;*                    LED display and key input program
    4                          ;*
    5                          ;*****************************************************************************
    6                          ;
    7                                  PUBLIC  LEDKEY,DIG_D,DP_D,LEDST_F,KEY_D,KCHA_F
    8                          ;
    9 ----                     LED_D   DSEG    0 AT 20H                                    ; LED display data area
   10 0020                             DS      4
   11 0024                     DP_D:   DS      1                                           ; DP display data pointer
   12 0025                     DIG_D:  DS      1                                           ; Digit data pointer
   13 0026                     CH_C:   DS      1                                           ; Chattering counter
   14 0027                     LKFLG:  DS      1                                           ; Key flag area
   15 0028                     KEY_D:  DS      8                                           ; Key data area
   16                          ;
   17 ----                     KEY_W   DSEG    0 AT 38H                                    ; Key work area
   18 0038                             DS      8
   19                          ;
   20        (009C)            KCHA_F  EQU     LKFLG.0                                     ; Key change flag
   21        (009D)            CHCT_F  EQU     LKFLG.1                                     ; Chattering count flag
   22        (009E)            LEDST_F EQU     LKFLG.2                                     ; LED display flag
   23                          ;
   24                          ;*****************************************************************************
   25                          ;
   26                          ;
   27                          ;
   28                          ;
   29                          ;
   30                          ;
   31                          ;
   32                          ;*****************************************************************************
   33                          ;
   34 ----                     LEDTBL  CSEG    PAGE
   35                          ;               abcdefgx                                    ; LED segment data
   36 0000  03                         DB      00000011B                                   ;"0"
   37 0001  9F                         DB      10011111B                                   ;"1"
   38 0002  25                         DB      00100101B                                   ;"2"
   39 0003  0D                         DB      00001101B                                   ;"3"
   40 0004  99                         DB      10011001B                                   ;"4"
   41 0005  49                         DB      01001001B                                   ;"5"
   42 0006  41                         DB      01000001B                                   ;"6"
   43 0007  1F                         DB      00011111B                                   ;"7"
   44 0008  01                         DB      00000001B                                   ;"8"
   45 0009  09                         DB      00001001B                                   ;"9"
   46 000A  11                         DB      00010001B                                   ;"A"
   47 000B  C1                         DB      11000001B                                   ;"B"
   48 000C  63                         DB      01100011B                                   ;"C"
   49 000D  85                         DB      10000101B                                   ;"D"
   50 000E  61                         DB      01100001B                                   ;"E"
   51 000F  71                         DB      01110001B                                   ;"F"
```

Phase-out/Discontinued

```
    **                                                          **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

    52                          ;
    53 0010                     LEDKEY:
    54                          ;PORT4=#0FFH (XA)
    55 0010   89FF                                                      MOV    XA,#0FFH ; Turn off LED display
    56 0012   92F4                                                      MOV    PORT4,XA
    57                          ;?SET   CY
    58 0014   E7                                                        SET1   CY
    59                          ;
    60                          ;if(DIG_D==#0) (A)
    61 0015   A325                                                      MOV    A,DIG_D
    62 0017   9A00                                                      SKE    A,#0
    63 0019   05                                                        BR     ??1
    64                          ;    DIG_D=#3 (A)                                          ; Change digit data
    65 001A   73                                                        MOV    A,#3
    66 001B   9325                                                      MOV    DIG_D,A
    67                          ;    ?CLR    CY
    68 001D   E6                                                        CLR1   CY
    69                          ;else
    70 001E   05                                                        BR     ??2
    71 001F                                                     ??1:
    72                          ;    ?DEC    DIG_D
    73 001F   A325                                                      MOV    A,DIG_D
    74 0021   C8                                                        DECS   A
    75 0022   9325                                                      MOV    DIG_D,A
    76                          ;endif
    77 0024                                                     ??2:
    78                          ;
    79                          ;?ROTATE PORT12                                            ; Decrement PORT12
    80 0024   A3FC                                                      MOV    A,PORT12
    81 0026   98                                                        RORC   A
    82 0027   93FC                                                      MOV    PORT12,A
    83                          ;
    84                          ;if_bit(LEDST_F)
    85 0029   A727                                                      SKT    LEDST_F
    86 002B R 5045                                                      BR     ??3
    87                          ;    HL=#LED_D
    88 002D   8B20                                                      MOV    HL,#LED_D ;Set table address
    89                          ;    L=DIG_D (A)
    90 002F   A325                                                      MOV    A,DIG_D
    91 0031   9972                                                      MOV    L,A
    92                          ;    A=@HL
    93 0033   E1                                                        MOV    A,@HL
    94                          ;    X=#0
    95 0034   9A09                                                      MOV    X,#0
    96                          ;    ?TABLE  PORT4,@PCXA
    97 0036   D0                                                        MOVT   XA,@PCXA        ; Set table data
    98 0037   92F4                                                      MOV    PORT4,XA
    99                          ;    L=DIG_D (A)
   100 0039   A325                                                      MOV    A,DIG_D         ; Set digit data
   101 003B   9972                                                      MOV    L,A
   102                          ;    L++
   103 003D   C2                                                        INCS   L
   104                          ;
   105                          ;    if(DP_D==L) (A)
```

```
     **                                                                  **

  STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT

   106 003E   A324                                                               MOV     A,DP_D
   107 0040   990A                                                               SKE     A,L
   108 0042   02                                                                 BR      ??4
   109                          ;       ?CLR    PORT4.0                                           ; DP display
   110 0043   9CC4                                                               CLR1    PORT4.0
   111                          ;    endif
   112 0045                                                              ??4:
   113                          ;
   114                          ;endif
   115 0045                                                              ??3:
   116                          ;
   117                          ;?CALADR        HL,DIG_D,KEY_W                                                          'Calculate chattering data
   118 0045   9A09                                                               MOV     X,#0                           address
   119 0047   A325                                                               MOV     A,DIG_D
   120 0049   8B38                                                               MOV     HL,#KEY_W
   121 004B   AAC8                                                               ADDS    XA,XA
   122 004D   AAC2                                                               ADDS    HL,XA
   123                          ;?NOT   XA,PORT6                                                  ; Correct comparison data
   124 004F   A2F6                                                               MOV     XA,PORT6
   125 0051   8FFF                                                               MOV     BC,#0FFH
   126 0053   AABE                                                               XOR     XA,BC
   127                          ;
   128                          ;if(XA!=@HL)                                                      ; Does chattering data change?
   129 0055   AA19                                                               SKE     XA,@HL
   130 0057   01                                                                 BR      ??5
   131 0058   07                                                                 BR      ??6
   132 0059                                                              ??5:
   133                          ;    XA<->@HL                                                     ; Store new data
   134 0059   AA11                                                               XCH     XA,@HL
   135                          ;    CH_C=#0FH-4 (A)                                              ; Set chattering counter
   136 005B   7B                                                                 MOV     A,#0FH-4
   137 005C   9326                                                               MOV     CH_C,A
   138                          ;    ?SET    CHCT_F                                               ; Set chattering count flag
   139 005E   9527                                                               SET1    CHCT_F
   140                          ;endif
   141 0060                                                              ??6:
   142                          ;
   143                          ;if(#KEY_W+6==HL) (XA)                                            ; Does chattering end?
   144 0060   893E                                                               MOV     XA,#KEY_W+6
   145 0062   AA4A                                                               SKE     XA,HL
   146 0064 R 508A                                                               BR      ??7
   147                          ;
   148                          ;    if_bit(CHCT_F)                                               ; Is chattering count flag set?
   149 0066   9727                                                               SKT     CHCT_F
   150 0068 R 508A                                                               BR      ??8
   151                          ;
   152                          ;       if(CH_C==#0FH) (A)
   153 006A   A326                                                               MOV     A,CH_C
   154 006C   9AF0                                                               SKE     A,#0FH
   155 006E R 5088                                                               BR      ??9
   156                          ;              ?CLR    CHCT_F
   157 0070   9427                                                               CLR1    CHCT_F
   158                          ;             HL=#KEY_D                                           ; Set key data address
   159 0072   8B28                                                               MOV     HL,#KEY_D
```

```
UCOM-75X FAMILY ASSEMBLER V3.5                                                                     89/01/10 21:37:11 PAGE :   4
   **                                                                 **
 STNO ADRS R OBJECT   IC MAC   SOURCE STATEMENT
  160                          ;           DE=#KEY_W
  161 0074   8D38                                                                 MOV     DE,#KEY_W;Set key work address
  162                          ;
  163                          ;           for(B=#0;B!=#8;B++)
  164 0076   9A0F                                                                 MOV     B,#0
  165 0078   01                                                                   BR      ??10
  166 0079                                                                ??11:
  167 0079   C7                                                                   INCS    B
  168 007A                                                                ??10:
  169 007A   9A87                                                                 SKE     B,#8
  170 007C   01                                                                   BR      ??12
  171 007D   09                                                                   BR      ??13
  172 007E                                                                ??12:
  173                          ;               A=@DL                                                ;Set preceding data address
  174 007E   E5                                                                   MOV     A,@DL
  175                          ;               if(A!=@HL)                                           ;Does data change?
  176 007F   80                                                                   SKE     A,@HL
  177 0080   01                                                                   BR      ??14
  178 0081   02                                                                   BR      ??15
  179 0082                                                                ??14:
  180                          ;                   ?SET    KCHA_F                                   ;Set key change flag
  181 0082   8527                                                                 SET1    KCHA_F
  182                          ;               endif
  183 0084                                                                ??15:
  184                          ;               A<->@HL+                                             ;Store data and increment address
  185 0084   EA                                                                   XCH     A,@HL+
  186 0085   60                                NOP
  187                          ;           next
  188 0086   F2                                                                   BR      ??11
  189 0087                                                                ??13:
  190                          ;
  191                          ;       else
  192 0087   02                                                                   BR      ??16
  193 0088                                                                ??9:
  194                          ;       CH_C++                                                       ;Increment chattering counter
  195 0088   8226                                                                 INCS    CH_C
  196                          ;        endif
  197 008A                                                                ??16:
  198                          ;
  199                          ;    endif
  200 008A                                                                ??8:
  201                          ;
  202                          ;endif
  203 008A                                                                ??7:
  204                          ;
  205 008A   EE                RET
  206                          ;
  207                          END

 TARGET CHIP : UPD75516
 STACK SIZE = 0000H

 ASSEMBLY COMPLETE,  NO ERROR FOUND
```

APPENDIX B MAIN PROGRAM EXAMPLE

The main program is separated into eight modes (S1-S8 are used as mode keys). When any one of S1-S8 is set to ON, the mode corresponding to the mode key is selected; otherwise, no processing is performed.

Table B-1 Correspondence between Keys and Modes

Key	Mode	Processing
S1	1	Scale generation
S2	2	EEPROM read/write
S3	3	A/D conversion
S4	4	SBI communication
S5	5	Notation adjustment
S6	6	Key input
S7	7	Analog key input
S8	8	Scale generation and display

When multiple keys are pressed at a time, the key having the lower key number takes precedence over other keys.

B.1 Scale Generation (Mode 1)

<Packages>

Scale generation program, LED display and key input program, and INTBT program

- When S1 is set to ON, key display and scale generation are performed.
- Keys SW9-SE24 correspond to the scale.
- For key display, SW9-SW24 correspond to display data 0-F.
- Scale generation and key display are performed only when keys are pressed.

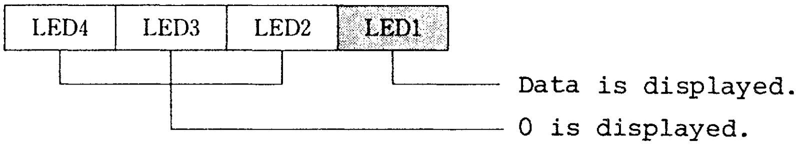


Table B2 lists the correspondence between the keys and generated scale and display data.

Table B-2 Correspondence between Keys and Scale and Display Data

SW	9	10	11	12	13	14	15	16
Scale data	-	Do	Do$^{\#}$	Re	Re$^{\#}$	Mi	Fa	Fa$^{\#}$
Display data	0	1	2	3	4	5	6	7
SW	17	18	19	20	21	22	23	24
Scale data	So	So$^{\#}$	La	La$^{\#}$	Si	Do	Do$^{\#}$	Re
Display data	8	9	A	B	C	D	E	F

<RAM area>

- LED display area (LED_D): 20H-23H
- Key change flag (KCHA_F): 27H.0
- Key data area (KEY_D): 28H-2FH
- Key input conversion work area (KEYIN_W): 4EH
- Scale data area (SOUND_D): 4FH
- No key flag (NOKEY_F): 59H.0
- Mode code pointer (MODE_P): 5AH

<Hard ware>

- Basic interval timer
- Timer/event counter

B.2 EEPROM Read/Write (Mode 2)

<packages>

Communication program with EEPROM, LED display and key input program, and INTBT program

- When the S2 key is set to ON, EEPROM is read/written according to the pressed keys
- Input data and read data are displayed.
- Table B-3 lists the correspondence between the keys and input data.

Table B-3 Correspondence between Keys and Input Data

SW	9	10	11	12	13	14	15	16
Input	0	1	2	3	4	5	6	7
SW	17	18	19	20	21	22	23	24
Input	8	9	A	B	C	D	E	F

SW25 and SW26 are used for:

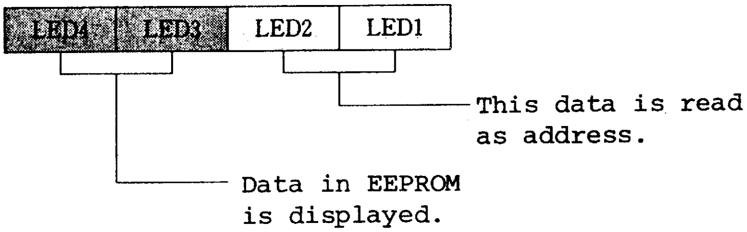
SW25	SW26
Read	Write

EEPROM is read in the following sequence:

(a) Enter read address (two hexadecimal digits) by pressing the keys.

The pressed key data is displayed on LED1. When another key is pressed, the preceding key data is shifted left and the new key data is displayed on LED1.

(b) Press SW25.

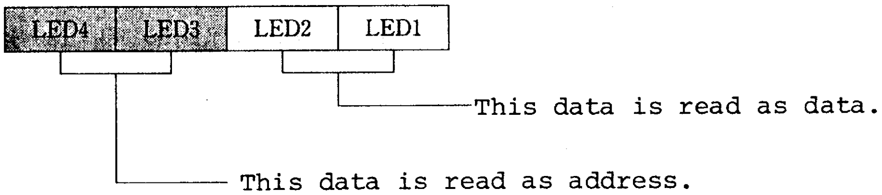


EEPROM is written in the following sequence:

(a) Enter write address (two hexadecimal digits) and data (two hexadecimal digits) in order by pressing the keys.

The pressed key data is displayed on LED1. When another key is pressed, the preceding key data is shifted left and the new key data is displayed on LED1.

(b) Press SW26.



<RAM area>

. Key change flag (KCHA_F)	:	27H.0
. Read/write flag (E2RW_F)	:	27H.3
. Write address area (E2WA_D)	:	30H-31H
. Write data area (E2WD_D)	:	32H-33H
. Read address area (E2RA_D)	:	34H-35H
. Read data area (E2RD_D)	:	36H-37H
. Key input conversion work area (KEYIN_W)	:	4EH
. No key flag (NOKEY_F)	:	59H.0
. Mode code pointer (MODE_P)	:	5AH

<Hardware>

- Serial interface (2-line mode)
- Basic interval timer

B.3 A/D Conversion (Mode 3)

<Programs>

A/D conversion program, LED display and key input program, and INTBT program

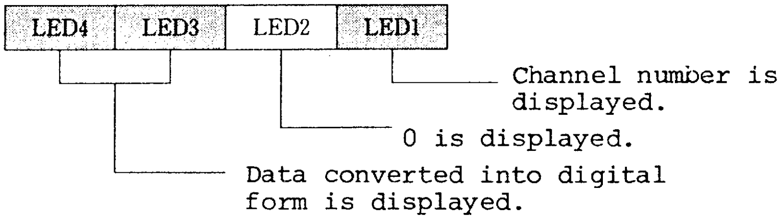
- A/D conversion is executed by pressing SW9-SW12 to enter the channel number.
- Table B-4 lists the correspondence between the keys and channel numbers and input ports.

Table B-4 Correspondence between Keys and Analog Channels

Key	Channel number	Analog channel
SW9	0	AN0
SW10	1	AN1
SW11	2	AN2
SW12	3	AN3

The data converted into digital form is displayed on LEDs.

The display format is as follows:


LED4
LED3
LED2
LED1
Channel number is displayed.
0 is displayed.
Data converted into digital form is displayed.

<RAM area>

. Key change flag (KCHA_F)	:	27H.0
. Analog chattering pointer (ACHN_P)	:	47H
. A/D conversion data area (ACONV_D)	:	48H-49H
. Key input conversion work area (KEYIN_W)	:	4EH

. No key flag (NOKEY_F) : 59H.0

. Mode code pointer (MODE_P) : 5AH

<Hardware>

. A/D converter

. Basic interval timer

B.4 SBI Communication (Mode 4)

<Programs>

SBI communication program, key input program, and INTBT program

- Commands and data are transmitted according to the pressed keys.
- Data is received according to the pressed keys.
- When mode 4 is selected, the program transmits a given address.
- The keys SW9-SW24 are used.
- The low-order four bits of an 8-bit slave address can be set by setting the slave DIP switch. The high-order four bits are set to 0. However, the slave address is read only at reset start.
- uPD75308 is used for the slave CPU.

The slave CPU performs processing according to the command data received from the master CPU, as listed in Table B-5.

Table B-5 Slave CPU Command List

Command	Processing
00H	Character code is written into the LCD display code area at the position pointed to by the current pointer. Then, the pointer is incremented.
01H	Character code is written into the LCD display code area at the position pointed to by the current pointer. Then, the pointer is decremented.
02H	Character data is read from the LCD display code area at the position pointed to by the current pointer. Then, the pointer is incremented. The master CPU transmits the number of output characters immediately following the command.
03H	Character data is read from the LCD display code area at the position pointed to by the current pointer. Then, the pointer is incremented. The master CPU transmits the number of output characters immediately following the command.

Command	Processing
04H	The pointer value indecating the display change position is rewritten. However, since the pointer consists of three bits, only the low-order three bits are valid even if data of 8 or more is written.
05H	The current pointer value is read.
06H	The decimal point at the position pointed to by the current pointer is set.
07H	The decimal point at the position pointed to by the current pointer is reset.
08H	Apostrophe at the position pointed to by the current pointer is set. The pointer value does not change.
09H	Apostrophe at the position pointed to by the current pointer is reset. The pointer value does not change.
FFH	Slave CPU is placed in nonselection state. When this command is acknowledged, neither commands nor data is acknowledged until a new slave address is acknowledged. Table B-6 lists the correspondence between keys and input data

Table B-6 lists the correspondence between the keys and input data.

Table B-6 Correspondence between Keys and Input Data

SW	9	10	11	12	13	14	15	16
Data	0	1	2	3	4	5	6	7
SW	17	18	19	20	21	22	23	24
Data	8	9	A	B	C	D	E	F

SW25, SW26 and SW27 are used for:

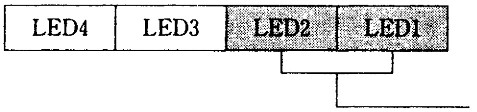
SW25	SW26	SW27
Command transmission	Data transmission	Reception

Commands and data are transmitted in the following sequence:

(a) Enter a command (two hexadecimal digits) or data (two hexadecimal digits) by pressing the keys.

The pressed key data is displayed on LED1. When another key is pressed, the preceding key data is shifted left and the new key data is displayed on LED1.

(b) Press SW25 for command transmission or SW26 for data transmission.



This data is read as command or data.

Commands 00H-0FH can be used. If any other value is entered for a command, an error occurs. When an error still occurs after command transmission is repeated five times, it is displayed on LED1-LED4.

For reception, press SW27.

The current LCD pointer value is received from the slave CPU and displayed on LED3-LED4.

<RAM area>

.	Key change flag (KCHA_F)	:	27H.0
.	Key input conversion work area (KEYIN_W)	:	4EH
.	Send data area (SBTR_D)	:	50H-51H
.	Receive data area (SBRE_D)	:	52H-53H
.	Communication code pointer (SBI_P)	:	54H
.	Error flag (ERR_F)	:	55H.0
.	No Key flag (NOKEY_F)	:	59H.0
.	Mode code pointer (MODE_P)	:	5AH

<Hardware>

- Serial interface (SBI mode)
- Basic interval timer
- Timer/event counter

B.5 Notation Adjustment (Mode 5)

<Programs>

Notation adjustment program, LED display and key input program, and INTBT program.

- The data entered by pressing a key is displayed in a four count, six count, octal, decimal, or twelve count number.
- The entered data must be one hexadecimal digit.

Table B-7 lists the correspondence between the keys and data.

Table B-7 Correspondence between Keys and Data

SW	9	10	11	12	13	14	15	16
Data	0	1	2	3	4	5	6	7
SW	17	18	19	20	21	22	23	24
Data	8	9	A	B	C	D	E	F

SW25-SW29 are used for:

SW25	SW26	SW27	SW28	SW29
adjustment	adjustment	Octal adjustment	Decimal adjustment	adjustment

One hexadecimal digit entered is converted into a decimal number in the following sequence:

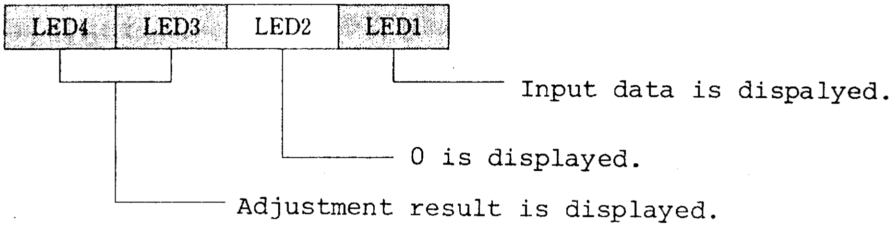
(a) Enter hexadecimal 1-digit data by pressing a key

The entered data is displayed on LED1. When a key is pressed, new data is displayed.

(b) Press SW28.

When SW28 is pressed, the data displayed on LED1 is converted into a decimal number for displayed on LED3-LED4.

The LED display format is as follows:



Input data is dispalyed.

0 is displayed.

Adjustment result is displayed.

Likewise, four count, six count, octal, and twelve count adjustments are performed.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, and B are used for twelve count digits.

<RAM area>

. Key change flag (KCHA_F)	:	27H.0
. Key input conversion work area (KEYIN_W)	:	4EH
. Notation data area (SIN_D)	:	4AH-4BH
. Notation code pointer (SIN_P)	:	4CH
. No key flag (NOKEY_F)	:	59H.0
. Mode code pointer (MODE_P)	:	5AH

<Hardware>

. Basic interval timer

B.6 Key Input (Mode 6)

<Programs>

LED display and key input program and INTBT program.

- Data entered by pressing SW9-SW32 is displayed on LED1-LED2.
- The data is displayed only while the keys are pressed.

Table B-8 lists the correspondence between the keys and display data.

Table B-8 Correspondence between Keys and Display Data

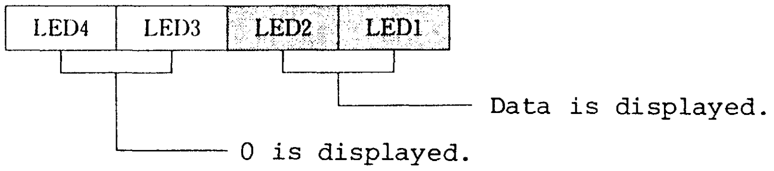
SW	9	10	11	12	13	14	15	16
Data	00	01	02	03	04	05	06	07

SW	17	18	19	20	21	22	23	24
Data	08	09	0A	0B	0C	0D	0E	0F

SW	25	26	27	28	29	30	31	32
Data	10	11	12	13	14	15	16	17

Key chaterring is about 20 ms.

The LED display format is as follows:



Data is displayed.

0 is displayed.

<RAM area>

. LED display data area (LED_D)	:	20H-23H
. Key change flag (KCHA_F)	:	27H.0
. LED display flag (LEDST_F)	:	27H.2
. Key data area (KEY_D)	:	28H-2FH
. Key input conversion work area (KEYIN_W)	:	4EH
. Mode code pointer (MODE_P)	:	5AH

<Hardware>

- Basic interval timer

B.7 Analog Key Input (Mode 7)

<Programs>

Analog key input program

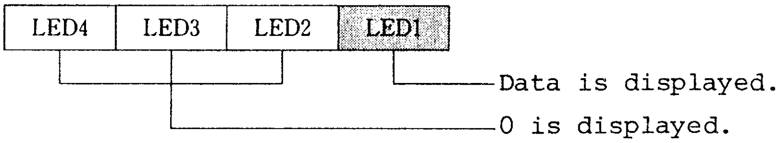
- Data entered by pressing AS1-AS16 is displayed on LED1.
- The data is displayed only while the keys are pressed.

Table B-9 Correspondence between Analog Keys and Display Data

AS	1	2	3	4	5	6	7	8
Data	0	1	2	3	4	5	6	7
AS	9	10	11	12	13	14	15	16
Data	8	9	A	B	C	D	E	F

Key chaterring is about 30 ms.

The LED display format is as follows:



<RAM area>

.	LED display data area (LED_D)	:	20H-23H
.	LED display flag (LEDST_F)	:	27H.2
.	Analog key data area (AKEY_D)	:	40H-41H
.	Analoy key change flag (AKCHA_F)	:	44H.0
.	Key input conversion work area (KEYIN_W)	:	4EH
:	Mode code pointer (MODE_P)	:	5AH

<Hardware>

- A/D converter
- Timer/event counter

B.8 Scale Generation and Display (Mode 8)

<Programs>

Scale generation program, LED display and key input program, and INTBT program

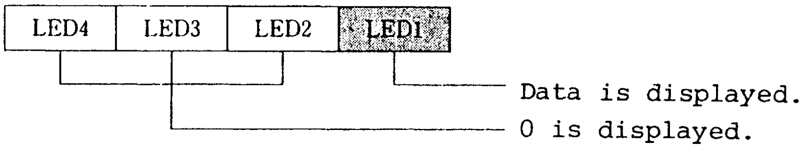
- Scale is generated every second. At the same time, digits are displayed on LED4 in the order of 0 to 8.

Table B-10 lists the correspondence between the display data and generated scale.

Table B-10 Correspondence between Display Data and Scale

Display	0	1	2	3	4	5	6	7	8
Scale	-	Do	Re	Mi	Fa	So	La	Si	Do#

The LED display format is as follows:



<RAM area>

. LED display data area (LED_D)	:	20H-23H
. LED display flag (LEDST_F)	:	27H.2
. Key input conversion work area (KEYIN_W)	:	4EH
. Scale data area (SOUND_D)	:	4FH
. Data counter (DATA_C)	:	57H
. 2-time counter (TIME2_C)	:	58H
. Mode code pointer (MODE_P)	:	5AH

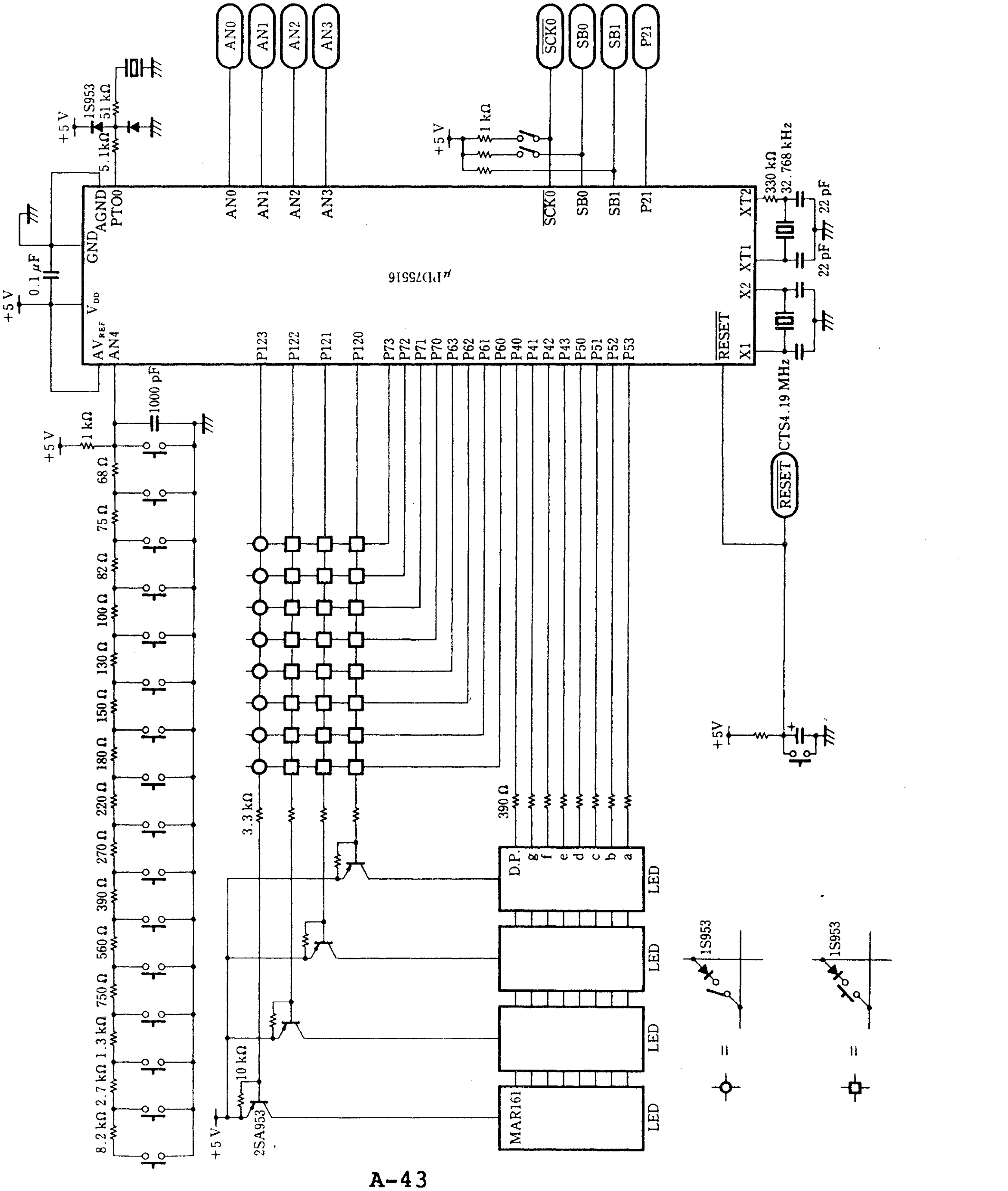
<Hardware>

- Basic interval timer
- Timer/event counter

B.9 Circuit Diagram

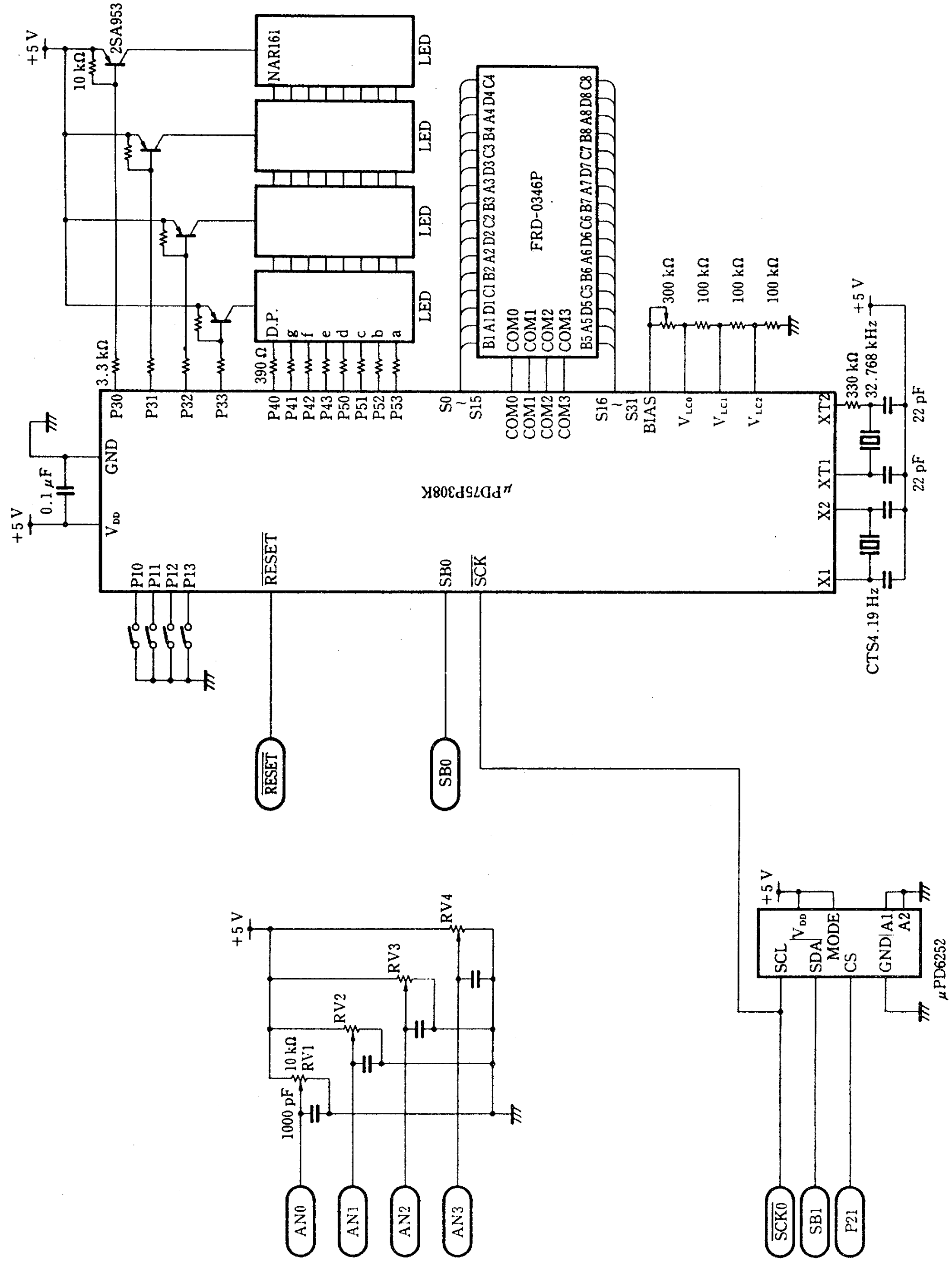
The circuit diagram of the system used with the main program is shown.

Fig. B-1 Circuit Diagram (1/2)



Phase-out/Discontinued

Fig. B-1 Circuit Diagram (2/2)



B.10 Program Example

```
;***************************************************************************
;*
;*                                                 MAIN PROGRAM
;*
;***************************************************************************
        VENT0   MBE=0,RBE=1,APMAIN
        VENT1   MBE=0,RBE=0,INTBT
;
        EXTRN   CODE(SINSU,ADCNV,EEPROM,SBITRN,SOUND,LEDKEY,ANKEY)
        EXTRN   DATA(SIN_P,SIN_D,SIN_W,LED_D,DIG_D,DP_D,KEY_D,ACHN_P,ACONV_D,AKEY_D)
        EXTRN   DATA(E2WA_D,E2WD_D,E2RA_D,E2RD_D,SBTR_D,SBRE_D,SBI_P,SOUND_D)
        EXTRN   BIT(LEDST_F,KCHA_F,AKCHA_F,E2RW_F,ERR_F)
;
DATA_C  DSEG    0 AT 57H                                     ;Data counter
        DS      1
TIME2_C:DS      1                                            ;2-time counter
MAINFLG:DS      1                                            ;Main flag area
MODE_P: DS      1                                            ;Mode code pointer
KEYIN_W DSEG    0 AT 4EH                                     ;Key input conversion work area
        DS      1
;
NOKEY_F EQU     MAINFLG.0                                    ;No key flag
;
;******************************MACRO AREA**********************************
?SET    MACRO   P1                                           ;SET1 instruction
                                              SET1    P1
        ENDM
;
?CLR    MACRO   P1                                           ;CLR1 instruction
                                              CLR1    P1
        ENDM
;
?CALL   MACRO   P1                                           ;CALL instruction
                                              CALLF   !P1
        ENDM
;***************************************************************************
;
;***************************************************************************
;                                           INITIALIZE
;***************************************************************************
APMAIN  CSEG    PAGE
;
SEL     RB2
PCC=#3 (A)                                                   ;Machine cycle 0.95 us
SP=#0 (XA)                                                   ;Set stack pointer
PORT4=#0FFH (XA)                                             ;Set PORT4 and PORT5
PORT12=#0FH (A)                                              ;Set PORT12
POGA=#11000001B (XA)                                         ;Set PORT0, 6, and 7 pull-up resistors
PMGB=#00110100B (XA)                                         ;PORT2, 4, and 5 output mode
PMGC=#00010000B (XA)                                         ;PORT12 output mode
HL=#18H                                                      ;Clear RAM (18H-0FFH)
XA=#0
;
while(XA!=HL)
        @HL=A
        HL++
        NOP
endw
;
BTM=#1111B (A)                                               ;Basic interval timer 1.95 ms
EI      IEBT                                                 ;Enable INTBT interrupt
EI
;***************************************************************************
;                                           MAIN
;***************************************************************************
while(forever)
;
    if((MODE_P+1)==#0) (A)                                   ;MODE_P=1XH?
;
        switch(MODE_P)                                       ; Check mode
            case 1:
                ?CALL   MODE1
                break
            case 2:
                ?CALL   MODE2
                break
            case 4:
                ?CALL   MODE3
                break
            case 8:
                ?CALL   MODE4
        ends
;
    elseif(MODE_P==#0) (A)                                       ;MODE_P=OXH?
;
        switch(MODE_P+1)                                         ;Check mode
            case 1:
                ?CALL   MODE5
                break
            case 2:
                ?CALL   MODE6
                break
            case 4:
                ?CALL   MODE7
                break
            case 8:
                ?CALL   MODE8
        ends
;
    endif
;
endw
;
;********************************************************************************
;                                MODE1
;********************************************************************************
;
MODE1:
LED_D=#0 (XA)                                                    ;Clear LED display data
(LED_D+2)=#0 (XA)
HL=#1
while(MODE_P==HL) (XA)                                           ;Mode=1?
;
    if_bit(KCHA_F)                                               ;Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                            ;Key data change processing
;
        if_bit(NOKEY_F)                                          ;Key input?
            ?CLR    NOKEY_F
            ?CLR    LEDST_F
            SOUND_D=#0 (A)                                       ;Set rest data
        else
            HL=#10H
;
            if(XA<HL)                                            ;Key data <10H?
                LED_D=A                                          ;Change display data
                ?SET    LEDST_F                                  ;Turn on LED display
                SOUND_D=A                                        ;Set scale data
            else
                ?CLR    LEDST_F                                  ;Turn off LED display
                SOUND_D=#0 (A)                                   ;Set rest data
            endif
;
        endif
;
        ?CALL   SOUND                                            ;Scale generation processing
    endif
;
HL=#1
endw
;
?CLR    LEDST_F                                                  ;Turn off LED display
SOUND_D=#0 (A)                                                   ;Set rest data
?CALL   SOUND
RET
;
;********************************************************************************
;                                MODE2
;********************************************************************************
;
MODE2:
LED_D=#0 (XA)                                                    ;Clear LED display data
(LED_D+2)=#0 (XA)
?SET    LEDST_F                                                  ;Turn on LED display
?SET    RELT
CSIM0=#10011111B (XA)                                            ;2-line serial I/O mode
;
HL=#2
while(MODE_P==HL) (XA)                                           ;Mode=2?
;
    if_bit(KCHA_F)                                               ;Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                            ;Change key data
```

```
;
        if_bit(NOKEY_F)                                         ; No key input?
            ?CLR    NOKEY_F
        else
;
            DE=XA
            HL=#10H
            if(XA==HL)                                          ; Key data=10H?
                E2RA_D=LED_D (XA)                               ; Set read address
                ?SET    E2RW_F                                  ; Read mode
                ?CALL   EEPROM                                  ; Communication
                (LED_D+2)=E2RD_D (XA)                           ; Display read data on LED
            endif
;
            XA=DE
            HL=#11H
            if(XA==HL)                                          ; Key data=11H?
                E2WA_D=LED_D+2 (XA)                             ; Set write address
                E2WD_D=LED_D (XA)                               ; Set write data
                ?CLR    E2RW_F                                  ; Write mode
                ?CALL   EEPROM                                  ; Communication
            endif
;
            XA=DE
            HL=#10H
;
            if(XA<HL)                                           ; Key data <10H?
                HL=#LED_D                                       ; Shift display data left
                while(LI=#04H)
                    A<->@HL+
                endwNOP
                LED_D=E (A)                                     ; Set data in display area
            endif
;
        endif
;
    endif
;
HL=#2
endw
;
?CLR    LEDST_F                                                 ;Turn off LED display
RET
;
;*******************************************************************************
;                                    MODE3
;*******************************************************************************
;
MODE3:
LED_D=#0 (XA)                                                   ;Clear LED display data
(LED_D+2)=#0 (XA)
;
HL=#4
while(MODE_P==HL) (XA)                                          ;Mode=3?
;
    if_bit(KCHA_F)                                              ;Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                           ;Key data change processing
;
        if_bit(NOKEY_F)                                         ;No key input?
            ?CLR    NOKEY_F
        else
            HL=#4
;
            DE=XA
            if(XA<HL)                                           ;Key data <4?
                XA=DE
                ACHN_P=A                                        ;Set analog channel data
                LED_D=A                                         ;Set key data in display area
                ?CALL   ADCNV                                   ;A/D conversion processing
                (LED_D+2)=ACONV_D (XA)                          ;Set conversion data in display area
                ?SET    LEDST_F                                 ;Turn on LED dispaly
            endif
;
        endif
;
    endif
;
HL=#4
endw
;
?CLR    LEDST_F                                                 ;Turn off LED display
RET
;
```

```
;****************************************************************************
;                                MODE4
;****************************************************************************
;
;*****************************MACRO AREA*************************************
;
?DATSET MACRO   P1,P2                                          ; Set SBI data
SBI_P=P1 (XA)
SBTR_D=P2 (XA)
        ENDM
;
;*************************************************************
;
MODE4:
SET1    RELT                                                   ; Set SOO latch
CSIM0=#10001011B (XA)                                          ; SBI mode (BUS=SB0)
TMOD0=#41H (XA)                                                ; Set timer/event counter
TM0=#01101100B (XA)
EI      IET0                                  ;EI     IET0     ; Enable INTTO interrupt
EI      IECSI0                                ;EI     IECSI0   ; Enable INTCSIO interrupt
;
LED_D=#0 (XA)                                                  ; Clear LED display data
(LED_D+2)=#0 (XA)
SET1    LEDST_F                                                ; Turn on LED display
?DATSET #0,#0                                                  ; Transmit address
?CLR    ERR_F
?CALL   SBISUB
HL=#8
while(MODE_P==HL) (XA)                                         ; Mode=4?
;
LOOP:
    if_bit(KCHA_F)                                             ; Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                          ; Key data conversion processing
;
        if_bit(NOKEY_F)                                        ; No key input?
            ?CLR    NOKEY_F
        else
            HL=XA
            DP_D=#0 (A)                                        ; Clear DP data
;
            if(H==#1)                                          ; Key data=1XH?
;
                if(L==#0)                                      ; Key data-10H?
                    HL=#10H
                        if(LED_D<HL) (XA)
                            ?DATSET #1,LED_D                   ; Transmit command
                            ?CLR    ERR_F
                            ?CALL   SBISUB
                        else
                            goto    ERROR                      ; Error
                        endif
                elseif(L==#1)                                  ; Key data=11H?
                    ?DATSET #2,LED_D                           ; Transmit data
                    ?CALL   SBISUB
                elseif(L==#2)                                  ; Key data=12H?
                    ?DATSET #1,#5                              ; Transmit command "05H"
                    ?CALL   SBISUB
                    SBI_P=#3 (A)                               ; Receive data
                    ?CALL   SBISUB
                    (LED_D+2)=SBRE_D (XA)                      ; Set receive data in display area
                    LED_D=#0 (XA)
                endif
;
            else
                D=L (A)
                HL=#LED_D
                while(L!=#04)                                  ; Shift display data left
                    A<->@HL+
                    NOP
                endw
                LED_D=D (A)                                    ; Set key data in display area
            endif
;
        endif
    endif
;
HL=#8
endw
;
DP_D=#0 (A)
?CLR    LEDST_F                                                ; Turn off LED display
?DATSET #1,#0FFH                                               ; Transmit command "Logoff"
?CLR    ERR_F
```

```
?CALL   SBISUB
;
DI      IETO                                  ;DI     IETO    ; Disable INTTO interrupt
DI      IECSIO                                ;DI     IECSIO  ; Disable INTCSIO interrupt
RET
;
ERROR:
LED_D=#0EEH (XA)
(LED_D+2)=#0EEH (XA)
DP_D=#1 (A)
goto    LOOP
;
;***************************************************************************
;                                   SBISUB
;***************************************************************************
SBISUB:
for(D=#0;D!=#5;D++)
    ?CALL   SBITRN
    if_bit(!ERR_F)
        break
    endif
next
if_bit(ERR_F)
    goto    ERROR
endif
;
RET
;***************************************************************************
;                                 MODE5
;***************************************************************************
;
MODE5:
LED_D=#0 (XA)                                           ; Clear display data
(LED_D+2)=#0 (XA)
?SET    LEDST_F                                         ; Turn on LED display
HL=#10H
;
while(MODE_P==HL) (XA)                                  ; Mode=5?
;
    if_bit(KCHA_F)                                      ; Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                   ; Key data conversion processing
;
        if_bit(NOKEY_F)                                 ; No key input?
            ?CLR    NOKEY_F
        else
            BC=XA
            HL=#15H
;
            if(XA<HL)                                   ; Key data <14H?
                XA=BC
                HL=#0FH
;
                if(XA>HL)                               ; Key data >0FH?
                    HL=XA
;
                    switch(L)                           ; Check mode
                        case 0:
                            SIN_P=#0 (A)                ; Four count conversion mode
                            break
                        case 1:
                            SIN_P=#1 (A)                ; Six count conversion mode
                            break
                        case 2:
                            SIN_P=#2 (A)                ; Octal conversion mode
                            break
                        case 3:
                            SIN_P=#3 (A)                ; Decimal conversion mode
                            break
                        case 4:
                            SIN_P=#4 (A)                ; Twelve count conversion mode
                    ends
;
                    SIN_W=LED_D (A)                     ; Set data
                    ?CALL   SINSU                       ; Notation adjustment processing
                    LED_D+2=SIN_D (XA)                  ; Set adjustment result in display
                                                          area
                else
                    LED_D=A                             ; Set key data in display area
                    LED_D+2=#0 (XA)
                endif
;
            endif
;
        endif
```

```
;
    endif
;
HL=#10H
endw
?CLR    LEDST_F                                         ;Turn off LED display
RET
;
;****************************************************************************
;                                   MODE6
;****************************************************************************
;
MODE6:
LED_D=#0 (XA)                                           ;Clear display data
(LED_D+2)=#0 (XA)
HL=#20H
;
while(MODE_P==HL) (XA)                                  ;Mode=6?
;
    if_bit(KCHA_F)                                      ;Does key change?
        ?CLR    KCHA_F
        ?CALL   KEYIN                                   ;Key data conversion processing
;
        if_bit(NOKEY_F)                                 ;No key input?
            ?CLR    NOKEY_F
            ?CLR    LEDST_F                             ;Turn off LED display
        else
            LED_D=XA                                    ;Set key data in display area
            ?SET    LEDST_F                             ;Turn on LED display
        endif
;
    endif
;
HL=#20H
endw
?CLR    LEDST_F                                         ;Turn off LED display
RET
;
;****************************************************************************
;                                   MODE7
;****************************************************************************
;
MODE7:
LED_D=#0 (XA)                                           ;Clear display data
(LED_D+2)=#0 (XA)
ADM=#01000100B (XA)                                     ;Set channel 4
HL=#40H
;
while(MODE_P==HL) (XA)                                  ;Mode=7?
;
    if_bit(AKCHA_F)                                     ;Does key change?
        ?CLR    AKCHA_F
        HL=#10H                                         ;No key input?
        if(AKEY_D==HL) (XA)
            ?CLR    LEDST_F                             ;Turn off LED display
        else
            LED_D=AKEY_D (XA)                           ;Set key data in display area
            ?SET    LEDST_F                             ;Turn on LED display
        endif
;
    endif
;
HL=#40H
endw
;
?CLR    LEDST_F                                         ;Turn off LED display
RET
;
;****************************************************************************
;                                   MODE8
;****************************************************************************
;
MODE8:
LED_D=#0 (XA)                                           ;Clear display data
(LED_D+2)=#0 (XA)
?SET    LEDST_F                                         ;Turn on LED display
WM=#00000100B (XA)                                      ;Set watch timer
DATA_C=#0 (A)                                           ;Clear data counter
HL=#80H
;
while(MODE_P==HL) (XA)                                  ;Mode=8?
;
```

```
    if_bit(IRQW)                                                 ;Is IRQW set?
        ?CLR    IRQW
        TIME2_C++                                                ;Increment 2-time counter
;
        if(TIME2_C==#2) (A)                                      ;One second?
            TIME2_C=#0 (A)
;
            switch(DATA_C)                                       ;Change scale data
                case 0:
                    SOUND_D=#1 (A)                               ;DO
                    break
                case 1:
                    SOUND_D=#3 (A)                               ;RE
                    break
                case 2:
                    SOUND_D=#5 (A)                               ;MI
                    break
                case 3:
                    SOUND_D=#6 (A)                               ;FA
                    break
                case 4:
                    SOUND_D=#8 (A)                               ;SO
                    break
                case 5:
                    SOUND_D=#0AH (A)                             ;LA
                    break
                case 6:
                    SOUND_D=#0CH (A)                             ;SI
                    break
                case 7:
                    SOUND_D=#0DH (A)                             ;DO
                    break
                default:
                    SOUND_D=#0 (A)                               ;REST
            ends
;
            if(DATA_C==#8) (A)                                   ;Data counter=8?
                DATA_C=#0 (A)                                    ;Clear data counter
            else
                DATA_C++                                         ;Increment data counter
            endif
;
            LED_D=DATA_C (A)                                     ;Set data counter value in display area
            ?CALL   SOUND                                        ;Scale generation processing
        endif
;
    endif
;
HL=#80H
endw
;
?CLR    LEDST_F                                                  ;Turn off LED display .
SOUND_D=#0 (A)                                                   ;Clear scale data
WM=#0 (XA)                                                       ;Stop watch timer
?CALL   SOUND                                                    ;Stop scale generation
RET
;
;******************************************************************************
;                               KEYIN
;******************************************************************************
;
;****************************MACRO AREA***************************************
?RORXA  MACRO                                                    ;Shift XA register right
                                                CLR1    CY
                                                XCH     A,X
                                                RORC    A
                                                XCH     A,X
                                                RORC    A
        ENDM
;******************************************************************************
KEYIN:
HL=#KEY_D+4
KEYIN_W=#0 (A)
;
while(KEYIN_W!=#3) (A)                                           ;Check counter
    C=#0
    XA=@HL
;
    while(C!=#8)
        ?RORXA                                                   ;Shift XA register right
;
        if_bit(CY)                                               ;Key input?
            break
        endif
```

```
;
        C++
    endw
;
    if_bit(CY)                                              ;Key input?
        break
    endif
;
    KEYIN_W++                                               ;Increment counter
    XA=#2
    HL-=XA
endw
;
switch(KEYIN_W)                                             ;Convert key input data
    case 0:
        HL=#0
        break
    case 1:
        HL=#8
        break
    case 2:
        HL=#10H
        break
    default:
        ?SET    NOKEY_F
ends
;
X=#0                                                        ;Store conversion result in XA
A=C                                                          register
XA+=HL
;
RET
;
;*****************************************************************************
;*                              INTBT
;*****************************************************************************
;
INTBT   CSEG    INBLOCK
;
CALLF   !LEDKEY                                             ;LED display and key input processi
CALLF   !ANKEY                                              ;Analog key input processing
RETI
;
END
```