Application Note

Programming Examples

For K0 and K0S Microcontrollers

Document No. U13218EU1V0AN00
April 1998

NEC

NEC

Introduction 1

This application note contains program examples illustrating basic functions of the K0 and K0S 8-bit microcontroller families in ways that can shorten development time and improve time to market. Each chapter in this application note includes the following:

- Brief description of the software routine
- Specification of the application software program with supporting diagrams
- Program flowchart
- Listings of code in Assembly language and C language

Program Examples

In the following table, chapter numbers and titles are cross-referenced with the applicable K0 and K0S devices. With minor modifications, the programs also could be used with other microcontrollers in the K Series® product line.

Chapter →	2	3	4	5	6	7	8	9	10	11	12	13	14
Devices ↓	**Sine Wave**	**A/D**	**3-Wire I/O**	**UART**	**Software UART**	**8-Bit Timer**	**DTMF**	**I^2C Interface**	**LCD Control**	**Keyboard Input**	**Watch Timer**	**Standby**	**LCD Demo**
µPD7808x		Y		Y	Y		Y						
µPD7801xF µPD7801xH		Y	Y		Y	Y		Y		Y	Y	Y	
µPD78002x µPD78003x					Y								
µPD7805x µPD7805xF µPD78005x	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y	
µPD7807x µPD78070A	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	
µPD7806x		Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y
µPD78030x		Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y
µPD7804xH µPD7804xF		Y	Y		Y	Y					Y	Y	
µPD78020x		Y	Y		Y	Y					Y	Y	
µPD78092x µPD78096x					Y								
µPD78901x					Y								
µPD78902x					Y								

Using the Programs

The best way to use the programs is to run the appropriate code either with the SM78K0 software simulator or the DDB-K0070A evaluation board, both of which can be obtained by contacting your local NEC representative. The software simulator with virtual hardware support provides the best low-cost option, because it is common to other development tools and allows you to become familiar with a standard development environment. The program examples were developed by NEC Electronics Inc. for demonstration purposes only. Modifications may be necessary for use in real applications.

Associated Documentation

This document should be used in conjunction with the appropriate hardware and software manuals and data sheets, which can be obtained by calling 1-800-366-9782 or faxing your request to 1-800-729-9288.

Document Name	Document Number
K Series® Selection Guide	U10930EJAV0SG00
K0 Family Selection Guide	U11126EJ6V0SG00
K0 Family (Instructions)	U12326EJ3V0UM00
K0 Family Basics (II)	U10121EJ2V0AN0
K0 Family Fundamentals (III)	U10182EJ1V1AN0
K0 Integrated Design Microcontroller	U11987EU1V0BR00
K0S Instruction Set	U11047EJ2V0UM00

“Include” File for C Language Programs

Be sure to use the following code in your “include file” when using any of the program examples in chapters 2 - 13.

```
/*****************************************************************
*        File name: define.h
*        Date:                    7/18/97
*Include file for definition
*****************************************************************/
#define TRUE1
#define FALSE0
//
//extended functions
//
#pragma sfr// SFR
#pragma asm// ASM
#pragma opc// DATA INSERTION

#ifdef K0S
#pragma realregister // using in K0S compiler
#endif

#pragma halt
#pragma stop
#pragma nop
#pragma di
#pragma ei

#pragma access// PEEK, POKE

#define CR13
#define LF10
//
//for only DDB-K0070A
//
#define DDB
*****************************************************************
```

NEC

Sine Wave Generation 2

Applicable K0 Devices

- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A

Description

Sine wave output applications using a D/A converter are widely used in the industry. The D/A converter converts a digital input into an analog value. The D/A converter in the K0 family consists of two channels with 8-bit resolution and uses the R-2R resistor ladder method of conversion.

There are two modes of operation: normal and real-time. Normal operation generates the analog voltage output signal immediately after the D/A conversion. Real-time operation generates the analog voltage output signal, which is gated by an output trigger, after the D/A conversion is complete. This mode is common in modem applications such as cordless telephone sets.

Because the converter voltage changes in steps, the output is usually connected to a low-pass filter. Figure 2-1 shows the digital and analog output waveforms. Figure 2-2 is the flowchart for the conversion program.

Figure 2-1. Analog Output in Real-TIme Output Mode

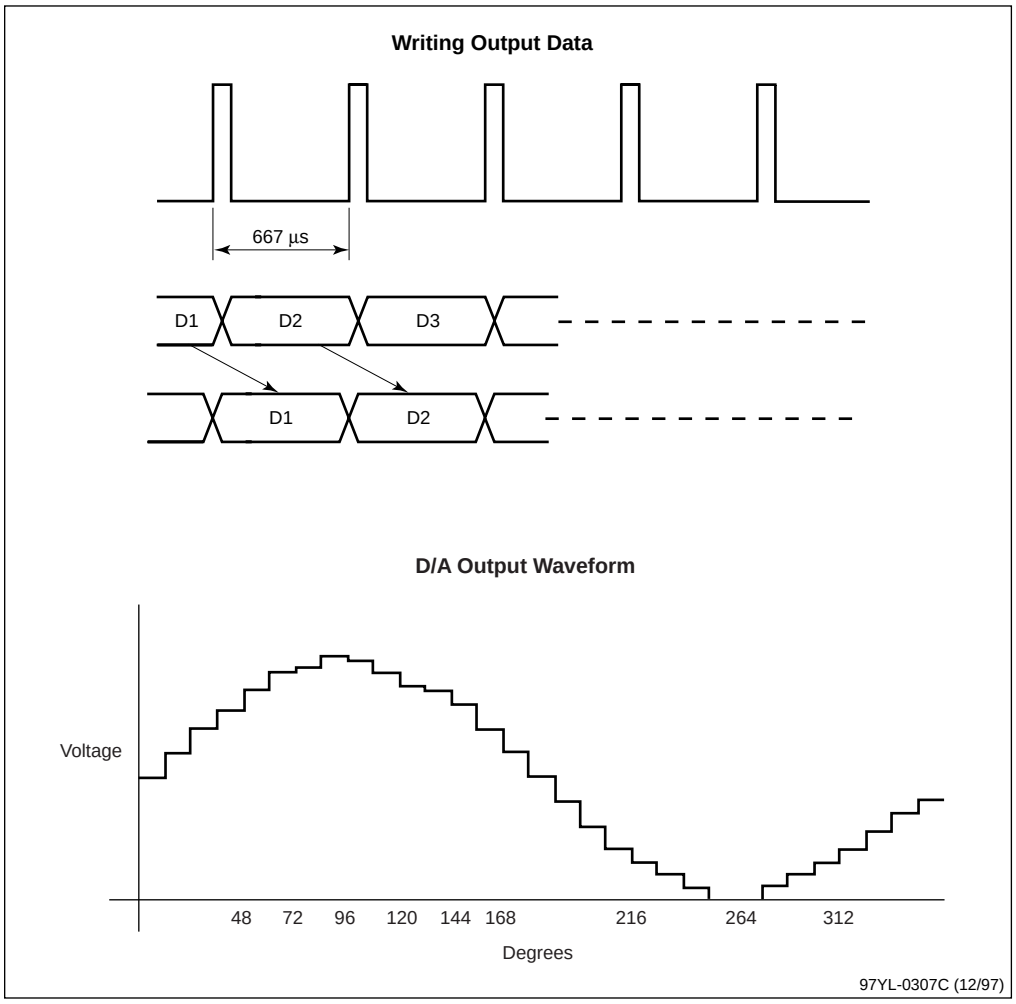
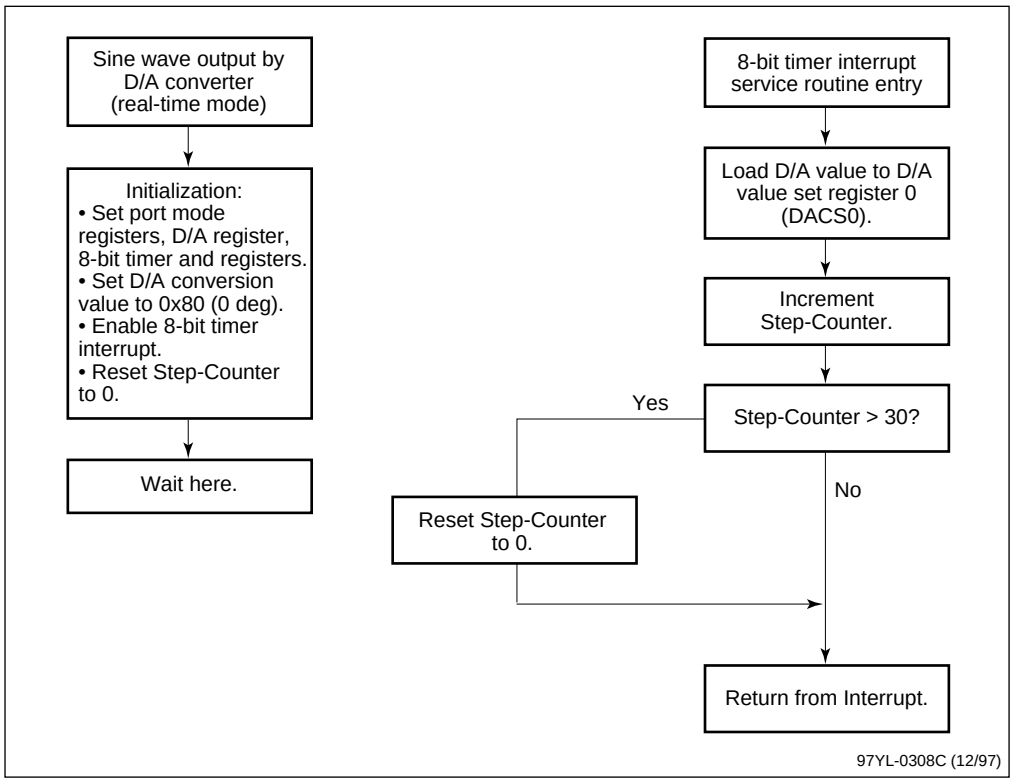


Figure 2-2. Flowchart of a Sine Wave Program



Registers to Output a Sine Wave

- D/A converter mode (DAM) register (Figure 2-3)
- D/A conversion value set registers (DASC0, DASC1)

Figure 2-3. D/A Converter Mode Register (DAM)

7	6	5	4	3	2	1	0
0	0	DAM5	DAM4	0	0	DACC1	DACE0

Bit	Name	Description
0	DACE0	D/A conversion channel 0 0 = Stop 1 = Enable
1	DACE1	D/A conversion channel 1 0 = Stop 1 = Enable
4	DAM4	Operation mode channel 0 0 = Normal 1 = Real-time output
5	DAM5	Operation mode channel 1 0 = Normal 1 = Real-time output

Output voltage = $AV_{REF}1$ x DACSn / 256

Where:

– $AV_{REF}1$ = reference voltage
– DACSn (n = 0 or 1) = D/A counter register

Software Specifications

Specification	Description
Output frequency	50 Hz
D/A channel number	0
Voltage output type	Real-time output
Number of steps	30 steps in the output voltage cycle
Interval time of timer	1/(50 Hz x 30 steps) = 667 µs
AVref1	5 volts

Assembly Language Program: Example of D/A Conversion

```
;*********************************************************************
;                                          File name: sine.asm
;                                          Date: 12/8/96
;*********************************************************************
; This sample program demonstrates the use of a D/A converter channel with an
; 8-bit resolution. The D/A can be used in two modes: normal mode and
; real-time mode. In the normal mode, the output trigger writes
; data to the D/A conversion value setting registers 0 and 1 (DACS0 and DACS1).
; In the real-time mode, the output is triggered by interrupt
; requests  (INTTM1 and INTTM2), which are asserted by 8-bit timer/event counters 1 and 2.
; In this mode, data is set into DACS0 and DACS1 after an output trigger has
; been generated or until the next output trigger is generated.
;
;         Sine wave output:
;         =================
;
;         The sine wave output example has a frequency of 50 Hz and is implemented by using
;         the 8-bit real-time output mode of D/A converter channel 0. Each
;         8-bit real-time interrupt, DACS0 is output and the next output
;         data is set into DACS0.
;         The interval time of the 8-bit timer/event counter 1 is set to
;         667 µs (30 steps if a 50-Hz frequency is chosen).
;
;                                    1 MHz
;         Step Timer value = ------------------ = 667 µs
;                            50 Hz x 30 (steps)
;
;         D/A Output:
;                                     AVref1 x DACS0
;         ANO0 pin output voltage = -----------------
;                                           256
;
;*********************************************************************
;         Sample program specification
;=====================================================================
;*********************************************************************
;         Restrictions if using the DDB-K0070A Development Board:
;         . Always leave bit 1 of register MK0L equal to 0 (INTP0).
;         . Do not use the software break (BRK). The monitor uses it.
;         . Do not use the 32-kHz subsystem clock because of the UART.
;         . Do not use the STANDBY mode either because of the UART.
;*********************************************************************
;
;         for only DDB-K0070A
;
$set (DDB)
;
;=====================================================================
;
;         Internal high RAM allocation
;
          dseg      IHRAM
          ds        20h
Stack:
;
;         Short address space
;
          dseg      saddr
count:    ds         (1)

          $ej
;=====================================================================
```

```
;         Reset vector
;=====================================================================
Vector    cseg      at 0000h
          dw        ResetStart
;
          org       24h
          dw        INTTM1                ;Generation of 8-bit timer/event counter 1 match

          $ej
;=====================================================================
;         Main
;=====================================================================
Main      cseg      at 1000h

ResetStart:
          di                              ;Disable interrupt
;
;         Select register bank 0
;
          sel       RB0                   ;Select register bank 0 for background
;
;         Select Master clock  (5 MHz)
;          . MCS = 1, no divided clock
;
          mov       OSMS,#00000001b
;
;         Init. I/O registers for sine wave output
;
          call      !InitSineWaveOutput
;
;         Set stack pointer (IHMEM + 20H)
;
          movw      sp,#Stack             ;Stack ptr
;
;         Enable INTP0 for DDB-K0070A
;
$if (DDB)
          clr1      PMK0
$endif
;
;         Enable interrupt
;

          EI
;
;         Wait here
;
main_loop:
          br        main_loop

;
          $ej
;=====================================================================
;         Output each step of the sine wave
;         Switch to register bank 1
;         Input:
;                   count = step count
;         Output:
;                   DAC output
;=====================================================================
INTTM1:
          sel       RB1
; Get the data
          movw      hl,#SineDataTable
```

```
          mov       a,count
          mov       b,a
          mov       a,[hl+b]
; output
          mov       DACS0,a
; next step
          inc       count
          cmp       count,#TableSizeSineData+1
          bc        $inttm1_10
; reset counter
          mov       count,#0
inttm1_10:
          reti

SineDataTable:                                          ;Degree
          db        09bh                  ;3.02 v       12
          db        0b4h                  ;3.5168 v     24
          db        0cbh                  ;3.9695 v     36
          db        0dfh                  ;4.3579 v     48
          db        0efh                  ;4.6651 v     60
          db        0fah                  ;4.8776 v     72
          db        0ffh                  ;4.9863 v     84

          db        0ffh                  ;4.9863 v     96
          db        0fah                  ;4.8776 v     108
          db        0efh                  ;4.6651 v     120
          db        0dfh                  ;4.3579 v     132
          db        0cbh                  ;3.9695 v     144
          db        0b4h                  ;3.5168 v     156
          db        09bh                  ;3.02 v       168

          db        080h                  ;2.5 v        180
          db        065h                  ;1.9802 v     192
          db        04ch                  ;1.4832 v     204
          db        035h                  ;1.0305 v     216
          db        021h                  ;0.6421 v     228
          db        011h                  ;0.3349 v     240
          db        006h                  ;0.1224 v     252
          db        001h                  ;0.0137 v     264

          db        001h                  ;0.0137 v     276
          db        006h                  ;0.1224 v     288
          db        011h                  ;0.3349 v     300
          db        021h                  ;0.6421 v     312
          db        035h                  ;1.0305 v     324
          db        04ch                  ;1.4832 v     336
          db        065h                  ;1.9802 v     348
          db        080h                  ;2.5 v        360
SineDataEnd:

TableSizeSineData           equ             (SineDataEnd - SineDataTable)

;*********************************************************************
;         Init. I/O registers for sine wave output
;
;         Sine wave output specification:
;                   . Sine wave output 50 Hz
;                   . Real-time mode
;                   . Number of steps for output = 30 steps
;                   . 8-bit timer mode = interval
;*********************************************************************
InitSineWaveOutput:
;
;         Set port13 in input mode (PM13)
```

```
;          p130 must be set to input mode for D/A conversion
;
           mov       PM13,#11111111b
;
;          8-bit timer register 1 count clock selection (TCL1)
;           . tcl1_0/3 = 1001b, MCS= 1, fx/2^4 (313 kHz = 3.2 µs)
;
           mov       TCL1,#11011001b
;
;          Set 8-bit timer compare register to 668 µs (50 Hz, 30 steps)
;
           mov       CR10,#208              ; t = 667 µs/3.2 = 208.44
;
;          Set 8-bit timer/event control register
;           . tce1 (bit0) = 0, Disable 8-bit timer register 1
;           . tce2 (bit1) = 0, Disable 8-bit timer register 2
;           . tmc12 (bit2) = 0, 8-bit timer register * 2 channel mode (tm1,tm2)
;
           mov       TMC1,#00000000b
;
;          8-bit timer operation enable
;
           set1      TCE1
;
;          Set D/A converter - 0 degree (2.5 v)
;
           mov       DACS0,#80h
;
;          D/A converter mode register (DAM)
;          DACE0 (bit 0) = 1 conversion enable for channel 0
;          DAM4 (bit 4 ) = 1 real-time output mode for channel 0
;
           mov       DAM,#00010001b
;
;          . Clear 8-bit timer 1 interrupt request flag
;          . Enable 8-bit timer 1 interrupt
;

           clr1      TMIF1
           clr1      TMMK1
;
;          Reset Step-Counter to 0
;
           mov       count,#0
           ret

;*************************************************************************
           end
_
```

C Language Program: Example of D/A Conversion

```
#define DEBUG 0

#pragma interrupt INTTM1 IRQ_8bit_Timer RB1

/*********************************************************************
;                                          File name: sine.c
;                                          Date: 9/2/97
;*********************************************************************
; This sample program demonstrates the use of a D/A converter channel with an
; 8-bit resolution. The D/A can be used in two modes: normal mode and
; real-time mode. In the normal mode, the output trigger writes
; data to the D/A conversion value setting registers 0 and 1 (DACS0 and DACS1).
; In the real-time mode, the output is triggered by interrupt
; requests  (INTTM1 and INTTM2), which are asserted by 8-bit timer/event counters 1 and 2.
; In this mode, data is set into DACS0 and DACS1 after an output trigger has
; been generated or until the next output trigger is generated.
;
;         Sine wave output:
;         ================
;
;         The sine wave output example has a frequency of 50 Hz and is implemented by using
;         the 8-bit real-time output mode of D/A converter channel 0. Each
;         8-bit real-time interrupt, DACS0 is output and the next output
;         data is set into DACS0.
;         The interval time of the 8-bit timer/event counter 1 is set to
;         667 µs (30 steps if a 50-Hz frequency is chosen).
;
;                                    1 MHz
;         Step Timer value = ------------------ = 667 µs
;                            50 Hz x 30 (steps)
;
;         D/A Output:
;                                     AVref1 x DACS0
;         ANO0 pin output voltage = -----------------
;                                          256
;*********************************************************************
;         Sample program specification
;=====================================================================
;*********************************************************************
;         Restrictions if using the DDB-K0070A Development Board:
;         . Always leave bit 1 of register MK0L equal to 0 (INTP0).
;         . Do not use the software break (BRK). The monitor uses it.
;         . Do not use the 32-kHz subsystem clock because of the UART.
;         . Do not use the STANDBY mode either because of the UART.
;
=====================================================================*/
#include "define.h"
/*=====================================================================
;         Delta value for sine wave output
=====================================================================*/
unsigned char const SineDataTable[] ={
          0x9b,      //;3.02 v               12
          0xb4,      //;3.5168 v             24
          0xcb,      //;3.9695 v             36
          0xdf,      //;4.3579 v             48
          0xef,      //;4.6651 v             60
          0xfa,      //;4.8776 v             72
          0xff,      //;4.9863 v             84

          0xff,      //;4.9863 v             96
          0xfa,      //;4.8776 v             108
```

```
          0xef,     //;4.6651 v             120
          0xdf,     //;4.3579 v             132
          0xcb,     //;3.9695 v             144
          0xb4,     //;3.5168 v             156
          0x9b,     //;3.02 v               168

          0x80,     //;2.5 v                180
          0x65,     //;1.9802 v             192
          0x4c,     //;1.4832 v             204
          0x35,     //;1.0305 v             216
          0x21,     //;0.6421 v             228
          0x11,     //;0.3349 v             240
          0x06,     //;0.1224 v             252
          0x01,     //;0.0137 v             264

          0x01,     //;0.0137 v             276
          0x06,     //;0.1224 v             288
          0x11,     //;0.3349 v             300
          0x21,     //;0.6421 v             312
          0x35,     //;1.0305 v             324
          0x4c,     //;1.4832 v             336
          0x65,     //;1.9802 v             348
          0x80      //;2.5 v                360
};
//
//
//
__sreg unsigned char count;
void InitSineWaveOutput ();
/*====================================================================
;         Main
;====================================================================*/
void main (void)
{
          DI ();
//
//        Select Master clock  (5 MHz)
//        MCS = 1, no divided clock
//
          OSMS = 0b00000001;
//
//        Init. I/O registers for sine wave output
//
          InitSineWaveOutput ();

#ifdef DDB
//
//        Enable INTP0 for DDB-K0070A
//
          PMK0 = 0;
#endif
          EI ();
//
//        Wait here
//
          while (TRUE);
}
/*====================================================================
;         Output a step for sine wave
;         Switch to register bank 1
;         Input:
;                   count = step number
;         Output:
;                   Output DAC count
```

```
;=====================================================================*/
void IRQ_8bit_Timer ()
{
          DACS0 = SineDataTable[count++];
          if ( count > sizeof SineDataTable) count = 0;
}
/*********************************************************************
;         Init. I/O registers for sine wave output
;
;         Sine wave output specification:
;                   . Sine wave output 50 Hz
;                   . Real-time mode
;                   . Number of steps for output = 30 steps
;                   . 8-bit timer mode = interval
;*********************************************************************/
void InitSineWaveOutput ()
{
//
//        Set port 13 in input mode (PM13)
//        p130 must be set to input mode for D/A conversion
//
          PM13= 0b11111111;
//
//        8-bit timer register 1 count clock selection (TCL1)
//         . tcl1_0/3 = 1001b, MCS= 1, fx/2^4 (313 kHz = 3.2 µs)
//
          TCL1=0b11011001;
//
//        Set 8-bit timer compare register to 668 µs (50 Hz, 30 steps)
//
          CR10=208; //t = 667 µs/3.2 = 208.44
//
//        Set 8-bit timer/event control register
//         . tce1 (bit0) = 0, Disable 8-bit timer register 1
//         . tce2 (bit1) = 0, Disable 8-bit timer register 2
//         . tmc12 (bit2) = 0, 8-bit timer register * 2 channel mode (tm1,tm2)
//
          TMC1=0b00000000;
//
//        8-bit timer operation enable
//
          TCE1 = 1;
//
//        Set D/A converter - 0 degree (2.5 v)
//
          DACS0=0x80;
//
//        D/A converter mode register (DAM)
//        DACE0 (bit 0) = 1 conversion enable for channel 0
//        DAM4 (bit 4) = 1 real-time output mode for channel 0
//
          DAM=0b00010001;
//
//        . Clear 8-bit timer 1 interrupt request flag
//        . Enable 8-bit timer 1 interrupt
//
          TMIF1 = 0;
          TMMK1 = 0;
//
//        Reset Step-Counter to 0
//
          count = 0;
}
_
```

NEC

A/D Converter 3

Applicable K0 Devices

- µPD7808x
- µPD7801xF
- µPD7801xH
- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A
- µPD7806x
- µPD78030x
- µPD7804xH
- µPD7804xF
- µPD78020x

Description

The K Series A/D converter consists of eight channels (ANI0 to ANI7) with 8-bit resolution. The conversion is based on the successive approximation method and the conversion result is held in the 8-bit A/D conversion result register (ADCR). There are two ways to start A/D conversion.

1. *Hardware*: conversion is triggered by the interrupt input (INTP3).
2. *Software*: conversion starts by setting the A/D converter mode register

The input channel is selected from ANI0 through ANI7, and A/D conversion is carried out. In a hardware start,operation stops after a single conversion. In a software start, the operation is repeated and an interrupt request (INTAD) generated each time an A/D conversion is completed.

A/D conversion is performed continuously until bit 7 (CS) of the ADM register is reset to 0 by software. If a write access to the ADM register is performed during an A/D conversion, the conversion is initialized, and if the CS bit is set to 1, conversion starts again from the beginning. Upon termination of the A/D conversion, the result is stored in the ADCR and an interrupt request signal (INTAD) generated. If data sets bit 7 (CS) of the ADM during an A/D conversion, the operation stops immediately.

A/D Converter Registers

Function	Register
Analog input	8 channels (ANI0 to ANI7)
Control register	A/D converter mode (ADM) register (Figure 3-1)
	A/D converter input select (ADIS) register (Figure 3-2)
	External interrupt mode register 1 (INTM1)
	A/D conversion result (ADCR) register

A/D Converter Program

The example program (Figure 3-3) reads all eight channels sequentially in a loop.

Software Specification

- Number of A/D input channels to be selected: 8
- Conversion time selection: 20 µs (100/fx, MCS = 1 with fx = 5.0 MHz)
- Conversion starts: software

Figure 3-1. A/D Converter Mode Register (ADM) Setup

7	6	5	4	3	2	1	0
CS	TRG	FR1	FR0	ADM3	ADM2	ADM1	HSC

Bit(s)	Name	Description
5, 4, 0	FR1, FR0, HSC	A/D conversion time 1 0 1 = 20 µs
3–1	ADM3–ADM1	Analog input channel selection 0 0 0 = channel 0
6	TRG	External trigger selection 0 = Software start
7	CS	A/D conversion operation control 0 = Stop

Figure 3-2. A/D Converter Input Select Register (ADIS) Setup

7	6	5	4	3	2	1	0
0	0	0	0	ADIS3	ADIS2	ADIS1	ADIS0

Bits	Name	Description
3–0	ADIS3–ADIS0	Number of analog input channels 1 0 0 0 = 8 channels

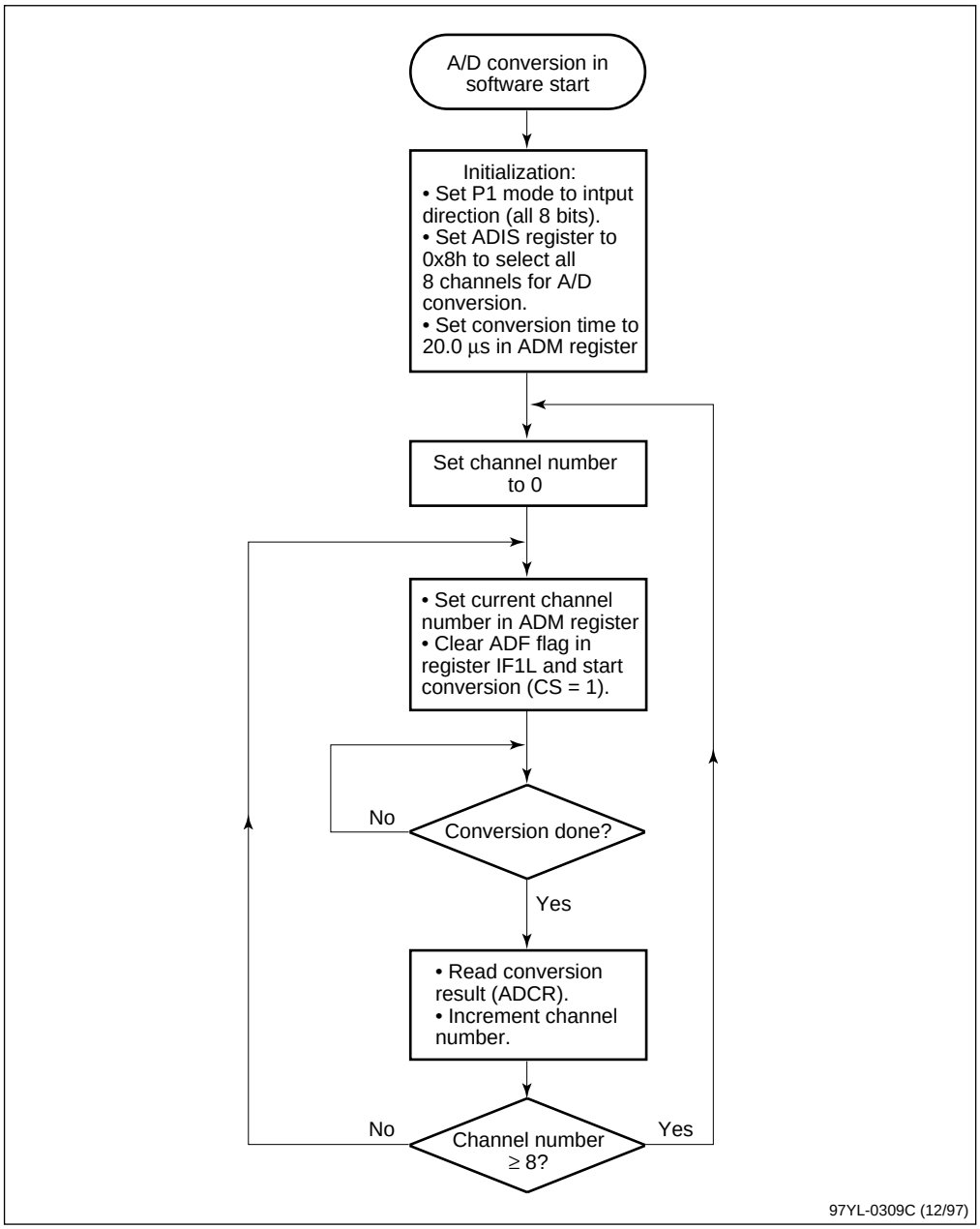
Input Voltage and Conversion Results

The relation between the voltage at the analog input pins and the A/D conversion result (the value stored in ADCR) = INT [V_{IN}/V_{IN} x 256 + 0.5].

Where:

– INT[ ] = function that returns integer parts of value in brackets
– V_{IN} = analog input voltage
– V_{REF} = $AV_{REF}0$ pin voltage

Figure 3-3. Flowchart of A/D Conversion Program



Assembly Language Program: Example of A/D Conversion

```
;*********************************************************************
;                                          File name: ADC.ASM
;                                          Date: 9/5/97
;*********************************************************************
; This module performs the A/D converter function in the K0 family.
; The conversion method is based on successive approximation and the
; conversion result is held in the 8-bit A/D conversion result register (ADCR).
;=====================================================================
; - A/D CONVERSION OPERATION IN SOFTWARE START
;
;         . Set an A/D channel for A/D conversion.
;         . Set a conversion time.
;         . Start conversion.
;         . Wait for the A/D channel conversion completed.
;         . Read the conversion result.
;         . Reset the interrupt request flag.
;         . Return with the result in the A register.
;
;=====================================================================
;         Input voltage and result:
;
;         ADCR (result register) = INT [(Vin/Vref) x 256 + 0.5]
;
;         Where,
;                   INT   = integer parts of the result
;                   Vin   = analog input voltage
;                   Vref  = AVref0 pin voltage (reference voltage)
;                   ADCR  = the conversion result register
;=====================================================================
;
;         Module name: ReadADdata
;
;         Input:
;                   A = channel # (0 to 7)
;
;         Return:
;                   A = The A/D result of the given channel
;
;*********************************************************************
;
;         for only DDB-K0070A
;
$set (DDB)
;=====================================================================
;         Port assignment for A/D
;=====================================================================

ADDirPort               equ                 PM1

;=====================================================================
;
; Select 100/fx (20 µs) if fx = 5.00 MHz
;         fr1/fr0/hsc = 101b and Start conversion bit on
;
ConversionTime          equ                 00100001b
;=====================================================================
;
;         Internal high RAM allocation
;         FWA = 0fb00h
;=====================================================================
          dseg      IHRAM
          ds        20h
```

```
Stack:
          $ej
;=========================================================================
;         Reset vector
;=========================================================================
Vector    cseg      at 0000h
          dw        ResetStart
; interrupt vector if any
          org       20h
;         dw
          $ej
;=========================================================================
;         Main
;=========================================================================
Main      cseg      at 80h

ResetStart:
          di                              ;Disable interrupt
;
;         Select register bank 0
;
          sel       RB0                   ;Select register bank 0
;
;         Set stack pointer (IHMEM + 20H)
;
          movw      SP,#Stack             ;Stack ptr
;
;         Select Master clock  (4.19 MHz)
;          . MCS = 1, no divided clock
;
          mov       OSMS,#00000001b
;
;         Enable INTP0 for DDB-K0070A
;
$if (DDB)
          clr1      PMK0
$endif
;
;         Init. ADC register
;
          call      !InitADC
;
;         16 step pwm output
;
loop:
          mov       a,#0                  ;Channel # 0
          call      !ReadADdata
          br        loop

;=========================================================================
;
;         Module name: ReadADdata
;
;         Input:
;                   A = channel # (0 to 7)
;         Return:
;                   A = The A/D result of the given channel
;
;=========================================================================
ReadADdata:
;         Set A/D channel
          add       a,a                   ;position to bit 1
;         merge with conversion time and start A/D conversion
          or        a,#ConversionTime
```

```
;          Clear interrupt flag
;          Start conversion
           mov       ADM,a

           clr1      ADIF
           set1      ADM.7
;
;          Wait for conversion done
;
wait:      bf        ADIF,$wait
;
;          Read the conversion result
;
           mov       a,ADCR
;          Stop conversion
           clr1      ADM.7
           ret
;*************************************************************************
;          ADC init
;          . Set conversion time
;          . Set # of A/D channels
;*************************************************************************
InitADC:
;          Set memory expansion mode register
           mov       ADDirPort,#11111111b
;
;          A/D conversion mode register
;
           mov       ADM,#ConversionTime
;
;          A/D input selection (select all 8 channels)
;
           mov       ADIS,#00001000b
           ret
;*************************************************************************
           end
```

C Language Program: Example of A/D Conversion

```
/*********************************************************************
;                                          File name: ADC.C
;                                          Date: 9/5/97
;*********************************************************************
; This module performs the A/D converter function in the K0 family.
; The conversion method is based on successive approximation and the
; conversion result is held in the 8-bit A/D conversion result register (ADCR).
;=====================================================================
; - A/D Conversion Operation in Software Start
;
;         . Set an A/D channel for A/D coversion.
;         . Set a conversion time.
;         . Start conversion.
;         . Wait for the A/D conversion to be completed.
;         . Read the conversion result.
;         . Reset the interrupt request flag.
;         . Return with the result in the A register.
;
;=====================================================================
;         Input voltage and result:
;
;         ADCR (result register) = INT [(Vin/Vref) x 256 + 0.5]
;
;         Where,
;                   INT = integer parts of the result
;                   Vin = analog input voltage
;                   Vref= AVref0 pin voltage (reference voltage)
;                   ADCR= the conversion result register
;=====================================================================
;
;         Module name: ReadADdata
;
;         Input:
;                   A = channel # (0 to 7)
;
;         Return:
;                   A = The A/D result of the given channel
;
;*********************************************************************
;
;         for only DDB-K0070A
;
;*********************************************************************/

#include "define.h"

//;=====================================================================
//;         Port assignment for A/D
//;=====================================================================

#define ADDirPortPM1

//;=====================================================================
//;
//; Select 100/fx (20 µs) if fx = 5.00 MHz
//;         fr1/fr0/hsc = 101b and start conversion bit on
//;
#define ConversionTime        0b00100001
//
//        Variables
//
__sreg unsigned char ChannelNumber;
```

```
__sreg unsigned char AD_Data;
//
//        functions
//
void InitADC ();
void ReadADdata ();

//;=====================================================================
//;         Main
//;=====================================================================
void Main (void)
{
//;
//;        Select Master clock  ( 5.00 MHz)
//;        . MCS = 1, no divided clock
//;
          OSMS = 0b00000001;
//;
//;         Enable INTP0 for DDB-K0070A
//;
#ifdef DDB
          PMK0 = 0;
#endif
//;
//;         Init. ADC register
//;
          InitADC ();
//;
//;         16-step pwm output
//;
          ChannelNumber = 0;
          while (TRUE)
          {
                    ReadADdata ();
          }
}
/*;=====================================================================
;         Module name: ReadADdata
;
;         Input:
;                   ChannelNumber = channel # (0 to 7)
;         Return:
;                   Unsigned char = The A/D result of the given channel
;=====================================================================*/
void ReadADdata ()
{
//;         Set A/D channel
//        ADM = ChannelNumber<< 1 | ConversionTime;
          ADM = ConversionTime;
          ADM |= ChannelNumber<< 1;

          ADIF = 0;
          ADM.7 = 1;
//;
//;         Wait for conversion done
//;
          while (TRUE)
          {
                    if ( ADIF == 1 ) break;
          }
//;
//;         Read the conversion result
//;
          AD_Data = ADCR;
```

```
//;         Stop conversion
          ADM.7 = 0;
}

/************************************************************************
;         ADC init
;         . Set conversion time
;         . Set # of A/D channels to all 8
;************************************************************************/
void InitADC ()
{
//          Set memory expansion mode register
          ADDirPort = 0b11111111l;
//
//        Starts by software, conversion time = 20 µs
//
          ADM = ConversionTime;
//
//          A/D input selection (select all 8 channels)
//

          ADIS = 0B00001000;
}
```

NEC

3-Wire Serial I/O (SPI) 4

Applicable K0 Devices

- µPD7801xF/7801xH
- µPD7805x/78005x/7805xF
- µPD7807x/78070A
- µPD7806x
- µPD78030x
- µPD7804xH/7804xF
- µPD78020x

Description

Most K0 and K0S microcontrollers support 3-wire serial I/O mode, which includes an emulation of the serial peripheral interface (SPI) and is used for transfer of 8-bit data using three lines: serial clock (SCKn), serial output (SOn), and serial input (SIn). This technique enables simultaneous transmission and reception and reduces data transfer processing time.

The start bit of transferred 8-bit data is switchable between the most MSB and LSB, so any serial device can be connected regardless of its start-bit recognition. The 3-wire mode should be used when connecting with peripheral I/O devices or display controllers that incorporate a conventional synchronous clocked serial interface. The clock source can be selectable either internally or externally, and the maximum clock speed can be selected up to 1.25 MHz (fxx/4).

3-Wire I/O Registers

Register	Description
Register	Serial data output register (SOn)
Control register	Timer clock select register (TCL3)
	Serial operating mode register 0 (CSIMn)
	Serial bus interface control register (SBIC)
	Port mode register 2 (PM2)

Operation

The 3-wire serial I/O mode is used for data transmission/reception in 8-bit data units. A bit-wise data transmission/reception is carried out in synchronization with the serial clock.

- **Transmission**: When data is written to SOn, transmission occurs on a bit-by-bit basis at the falling edge of the serial clock (SCKn). The transmitted data is held in the SOn latch and output on the SOn pin. When the 8-bit data is completely transmitted, SOn operation terminates and an interrupt request flag (CSIIF0) is set.

- **Reception**: To read 8-bit data with the internal timer (8-bit timer register 2), dummy data has to be transmitted to the SOn in order to start generation of a clock SCKn. The received data is latched in SIn at the rising edge of SCKn. Upon completion of the 8-bit data transfer, SIn operation stops automatically and the interrupt request flag (CSIIFn) is set.

Example Program

The example program transmits a test string ‘ABCDEFGHIJKLMN’ and receives 256 bytes of data from channel 0. See Figures 4-1 through 4-3 for the timing diagram, mode register format (CSIM0), and program flowchart.

Software Specification

Specification	Description
Interface setup	Master and a slave
Serial interface	Channel 0
Output string	ABCDEFGHIJKLMN
I/O timer clock selection	19.5 kHz with MCS = 1; fx = 5.0 MHz
Input buffer size	256-byte ring buffer

Figure 4-1. 3-Wire Serial I/O Timing

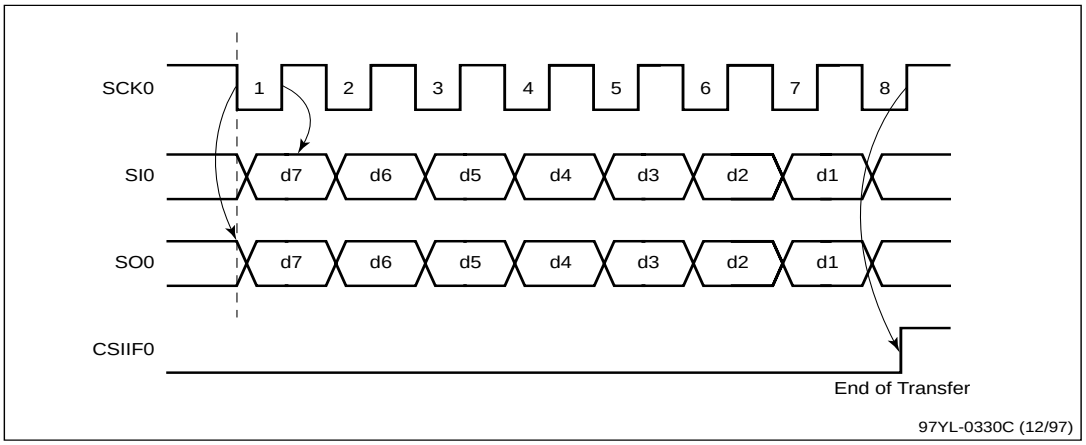
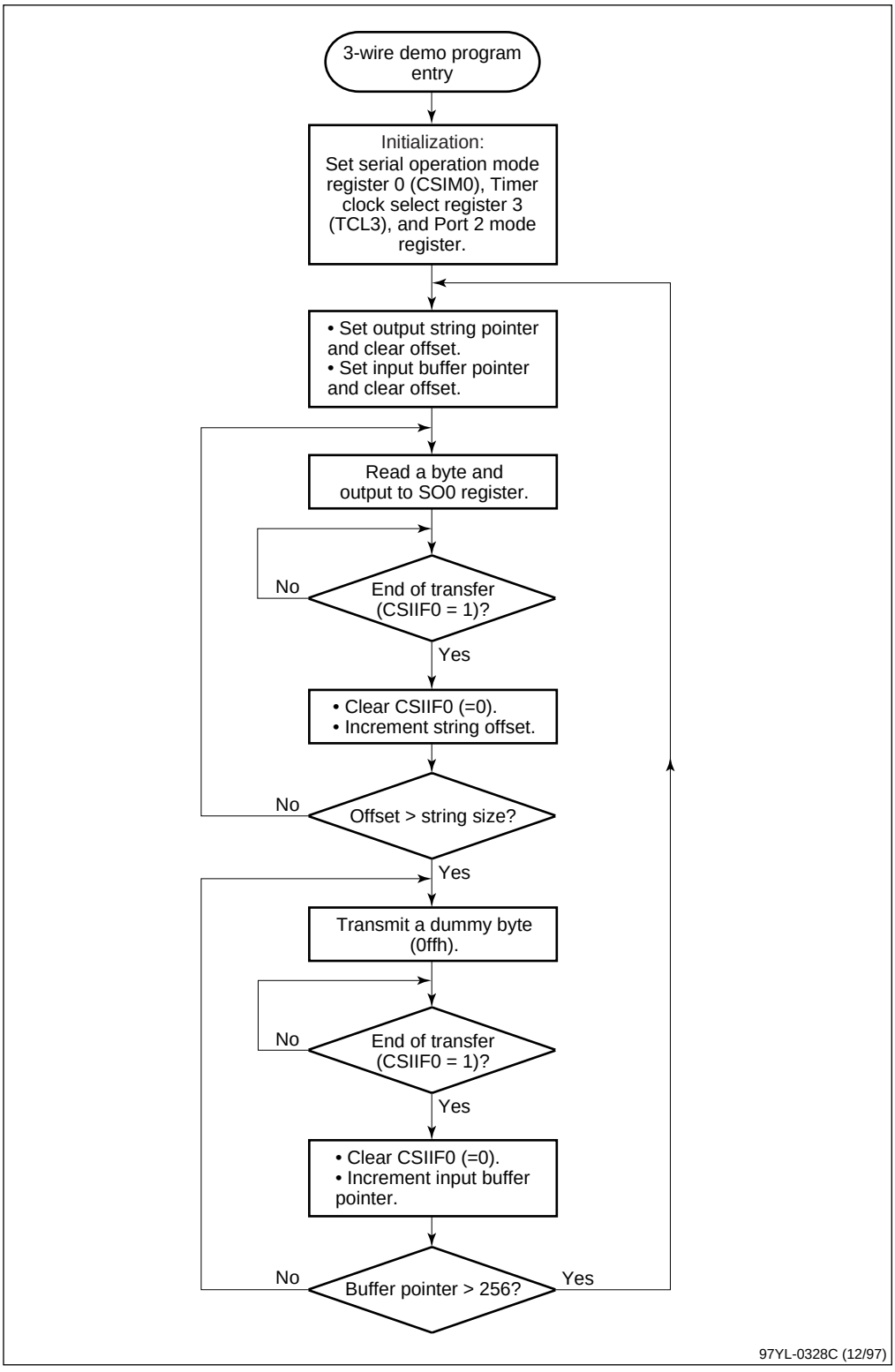


Figure 4-2. Serial Operation Mode Register 0 (CSIM0)

7	6	5	4	3	2	1	0
CSIE0	COI	WUP	CSIM04	CSIM03	CSIM02	CSIM01	CSIM00

Bit(s)	Name	Description
1–0	CSIM01, CSIM00	Clock selection 110 = clock specified with bits 0 to 3 of register TCL3
2	CSIM02	Start bit 0 = MSB
4–3	CSIM04, CSIM03	Serial I/O mode 0 0 = 3-wire
5	WUP	Wake-up function 0 = Interrupt request signal with each transfer in any mode
6	COI	Slave address comparison result 0 = Slave address register not equal to serial I/O register 0 data
7	CSIE0	Serial interface channel 0 operation 0 = stop and 1 = enable

Figure 4-3. Flowchart of 3-Wire I/O Program



Assembly Language Program: Example of 3-Wire Serial Interface

```
;*********************************************************************
;                                          File name: SPI.ASM
;                                          Date: 9/6/97
;
;*********************************************************************
; This sample program demonstrates the SPI (serial peripheral interface)
; mode (3-wire serial I/O mode) of the K0 family (Master operation).
;
;;         Algorithm:
;           - Transmission:
;                     .Set Rx/Tx to regular 3-wire serial mode.
;                     .Transmit a character.
;                     .Wait for Tx to be done.
;                     .Clear the interrupt flag and wait for next data.

;           - Receive:
;                     .Set Rx/Tx to regular 3-wire serial mode.
;                     .Transmit a dummy character (000h) to be output to SCK.
;                     .Wait for Rx to be done.
;                     .Read a serial data sting.
;                     .Clear the interrupt flag and wait for next data.

;          This example code outputs a data string:
;
;                     "ABCDEFGHIJKLMN"
;
;          And reads 256 bytes of data.
;
;*********************************************************************
;
;          for only DDB-K0070A
;
$set (DDB)
;=====================================================================
;          Equates
;=====================================================================
SioInDir                equ                 pm2.5          ;Input port direction
SioInPort               equ                 p2.5           ;Input port
SioOutDir               equ                 pm2.6          ;Output port direction
SioOutPort              equ                 p2.6           ;Output port
SClockDir               equ                 pm2.7          ;Clock
SClockPort              equ                 p2.7           ;Clock

;=====================================================================
;          Saddr area
;          FWA = 0fe20h
;=====================================================================
           dseg       saddr
;=====================================================================
;          Even boundary
;=====================================================================
           dseg       saddrp
;=====================================================================
;          Bit segment
;=====================================================================
           bseg

;=====================================================================
;
;          Internal high RAM allocation
;
           dseg       IHRAM
```

```
;
;          Input buffer
;
InputBuffer:
           ds         100h
InputBufferEnd:

InputBufferSize equ (InputBufferEnd - InputBuffer) and 0ffh
;
;          Stack ptr

Fwa_Stack:
           ds         20h
Stack:
;
           $ej
;=========================================================================
;          Reset vector
;=========================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           $ej
;=========================================================================
;          Main
;=========================================================================
Main       cseg       at 80h

ResetStart:
           di                                ;Disable interrupt
;
;          Oscillation mode select register
;           . Does not use divider circuit
;
           mov        osms,#00000001b
;
;          Set stack pointer (IHMEM + 20H)
;
           movw       sp,#Stack              ;Stack ptr
;
;          Init. UART register
;
           call       !Init_SPI1
;
;          Enable INTP0 for DDB-K0070A
;
$if (DDB)
           clr1       pmk0
$endif
;
;          Output test string
;          'ABCDEFGHIJKLMN'
;
OutputData:
           set1       csiif0

           movw       hl,#TestString
           mov        c,#TestStringSize
           mov        b,#0
Write_loop:
           mov        a,[hl+b]
;          Wait for transfer completion
           bf         csiif0,$$
;          Output data
```

```
          clr1      csiif0
          mov       sio0,a                  ;Output data
          inc       b
          dbnz      c,$Write_loop
;
;         Jump to input data from serial I/O

          br        InputData
;
;         Test output string
;
TestString:
          db        'ABCDEFGHIJKLMN',0
TestStringEnd:

TestStringSize equ TestStringEnd - TestString

;=====================================================================
;         Read data and store to an input buffer (256 bytes)
;=====================================================================
InputData:
;
;         Read data from serial input
;         ==========
;         Clear buffer
;
clear:
          movw      hl,#InputBuffer
          mov       c,#InputBufferSize
          mov       b,#0
          mov       a,#0
clear_loop:
          mov       [hl+b],a
          inc       b
          dbnz      c,$clear_loop
;
          clr1      csiif0
;
;         Read data
;
          movw      hl,#InputBuffer
          mov       c,#InputBufferSize
          mov       b,#0
Read_loop:
          mov       sio0,#00 ;Output dummy data for clock generation
;         Wait for transfer completion
          bf        csiif0,$$
          clr1      csiif0
;         Input data
          mov       a,sio0
          mov       [hl+b],a                ;store data
          inc       b
          dbnz      c,$Read_loop

          br        OutputData              ;output data


;=====================================================================
;         Init. for SPI case 3
;         . Set 3-wire serial I/O
;         . Clock speed = 51.28 µs (19.5 kHz)
;=====================================================================
Init_SPI1:
;
```

```
;         Clock specified with bits 0 to 3 of timer clock selection register 3 (TCL3)
;         Disable operation (port mode), bit7 = 0
;
;*************************************************************
;*         Serial interface channel 0 control register (CSIM0) *
;*                                                             *
;*         1 0 0 0 0 0 1 1                                     *
;*         ^ ^ ^ ^ ^ ^ ^ ^                                     *
;*         | | | | | | | |__ 1 Clock specified with bits 0 to 3 of TCL3 *
;*         | | | | | | | |                                     *
;*         | | | | | | |____ 1 Clock specified with bits 0 to 3 of TCL3 *
;*         | | | | | | |                                       *
;*         | | | | | |______ DIR0: Start bit 0 = MSB, 1 = LSB*
;*         | | | | |                                           *
;*         | | | | |________ 0 3-wire mode                     *
;*         | | | |                                           *
;*         | | | |__________ 0 3-wire mode                   *
;*         | | |                                             *
;*         | | |____________ 0 Interrupt request signal generation *
;*         |              with each serial transfer in any mode*
;*         | |                                               *
;*         | |_______________ 0 Slave address register not equal to *
;*         |                   serial I/O shift register 0 data*
;`         |________________ 1 Enable operation              *
;*                                                           *
;*************************************************************/
          mov       csim0,#10000011b
;
;         Clock selection (chn0) - mcs = 1, fxx/2^8 = 51.28 µs (19.5 kHz)
;
          mov       tcl3,#00001101b
;
;         Serial data input,output, clock
;
          set1      SioInDir                ;Input data direction
;
          clr1      SioOutDir               ;Output data direction
          clr1      SioOutPort              ;Output data low
;
          clr1      SClockDir               ;Clock output
          set1      SClockPort              ;Output clock High

          ret
;*********************************************************************
          end
_
```

C Language Program: Example of 3-Wire Serial Interface

```
/*********************************************************************
;                                          File name: SPI.C
;                                          Date: 9/8/97
;
;*********************************************************************
; This sample program demonstrates the SPI (serial peripheral interface)
; mode (3-wire serial IO mode) of the K0 family (Master operation).
;
;
;         Algorithm:
;          - Transmission:
;                   .Set Rx/Tx to regular 3-wire serial mode.
;                   .Transmit a character.
;                   .Wait for Tx to be done.
;                   .Clear the interrupt flag and wait for next data.

;          - Receive:
;                   .Set Rx/Tx to regular 3-wire serial mode.
;                   .Transmit a dummy character (0ffh) to be output to SCK.
;                   .Wait for Rx to be done.
;                   .Read a serial data string.
;                   .Clear the interrupt flag and wait for next data.

;         This example code outputs a data string:
;
;                   "ABCDEFGHIJKLMN"
;
;         And reads 256 bytes of data.
;
;*********************************************************************/

#include "define.h"

/*;====================================================================
;         Equates
;====================================================================*/
#define SioInDir              PM2.5 // ;Input port direction
#define SioInPort             P2.5 // ;Input port
#define SioOutDir             PM2.6 // ;Output port direction
#define SioOutPort            P2.6 // ;Output port
#define SClockDir             PM2.7 // ;Clock
#define SClockPort            P2.7 // ;Clock
//
//;
//;         Test Output String
//;
const unsigned char TestString[] = "ABCDEFGHIJKLMN";
//;
//;         Input buffer
//;
unsigned char InputBuffer[0x100];
__sreg    unsigned char i,j;
//
//
//
void      Init_spi ();
/*;====================================================================
*         Main
*;====================================================================*/
void Main (void)
{
          DI ();                          //;Disable interrupt
```

```
//;
//;       Oscillation mode select register
//;       Does not use divider circuit
//;
          OSMS =0b00000001;
//;
//;       Init UART Register
//;
          Init_spi ();
//;
//;       Enable INTP0 for DDB-K0070A
//;
#ifdef    DDB
          PMK0 = 0;
#endif
          {
                    //;=====================================================
                    //;                     Output a test string
                    //;                     'ABCDEFGHIJKLMN'
                    //;=====================================================
                    CSIIF0 = 1;
                    for (i=0;i< sizeof TestString;i++)
                    {
                                        while (TRUE) { if ( CSIIF0 ) break; }
                                        CSIIF0 = 0;
                                        SIO0 = TestString[i];
                    }
                    //;=====================================================
                    //;                     Read data and store to an input buffer (256 bytes)
                    //;=====================================================
                    //;                     Clear buffer
                    //;
                    for (i=0;i< sizeof InputBuffer;i++) InputBuffer[i] = 0;
                    //
                    //                      Read data
                    //
                    CSIIF0 = 0;
                    for (i=0;i< sizeof InputBuffer;i++)
                    {
                                        SIO0 = 0x00;    // output a dummy byte
                                        while (TRUE) { if ( CSIIF0 ) break; }
                                        CSIIF0 = 0;
                                        InputBuffer[i] = SIO0;
                    }
          }
}
/*;=====================================================================
;         Init. for SPI case 3
;         . Set 3-wire serial I/O
;         . Clock speed = 51.28 µs (19.5 kHz)
;=====================================================================*/
void      Init_spi ()
{
//;
//;        Clock specified with bits 0 to 3 of timer clock selection register (TCL3)
//;        Disable operation (port mode), bit7 = 0
//;
```

```
/*;************************************************************
;*          Serial interface channel 0 control register, (CSIM0) *
;*                                                               *
;*          1 0 0 0 0 0 1 1                                      *
;*          ^ ^ ^ ^ ^ ^ ^ ^                                      *
;*          | | | | | | | |__ 1 Clock specified with bits 0 to 3 of TCL3 *
;*          | | | | | | |                                        *
;*          | | | | | | |____ 1 Clock specified with bits 0 to 3 of TCL3 *
;*          | | | | | |                                          *
;*          | | | | | |______ DIR0:Start bit 0 = MSB, 1 = LSB *
;*          | | | | |                                            *
;*          | | | | |________ 0 3-wire mode                      *
;*          | | | |                                            *
;*          | | | |__________ 0 3-wire mode                    *
;*          | | |                                              *
;*          | | |____________ 0 Interrupt request signal generation *
;*          | |                 with each serial transfer in any mode* *
;*          | |_______________ 0 Slave address register not equal to*
;*          |                    serial I/O shift register 0 data*
;*          |________________ 1 Enable operation                *
;*                                                               *
;************************************************************/
          CSIM0 =0b10000011;
//;
//;         Clock selection (chn0) - MCS= 1, fxx/2^8 = 51.28 µs (19.5 kHz)
//;
          TCL3=0b00001101;
//;
//;         Serial data input,output, clock
//;
          SioInDir = 1;                    //;Input data direction
//;
          SioOutDir = 0;                   //;Output data direction
          SioOutPort = 0; //;Output data low
//;
          SClockDir = 0;                   //;Clock output
          SClockPort = 1; //;Output clock High
}
```

NEC

Asynchronous Serial Interface (UART) 5

Applicable K0 Devices

- µPD7808x
- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A
- µPD7806x
- µPD78030x

Description

Asynchronous serial I/O is widely used in the industry, most commonly for RS-232 communication. The data transmission/reception is provided at programmable rates, generated either by a dedicated on-chip baud rate generator or by scaling the input clock. A data frame of transmitted/received data consists of a start bit, character bits, a parity bit, and stop bits. Prior to a data transmission or reception, the asynchronous control registers have to be set according to the required data format.

Baud Rate

When using the on-chip baud-rate generator, a baud rate must be selected. A programmable divide factor of the main system clock generates the transmit/receive clock for the baud rate, calculated as baud rate = fxx/ 2^n x k.

Where:

– fxx = main system clock frequency (5 MHz typical)
– n = value set by TPS0–TPS3 in baud rate generator control register (BRGC)
– k = value set in MDL0–MDL3 in BRGC

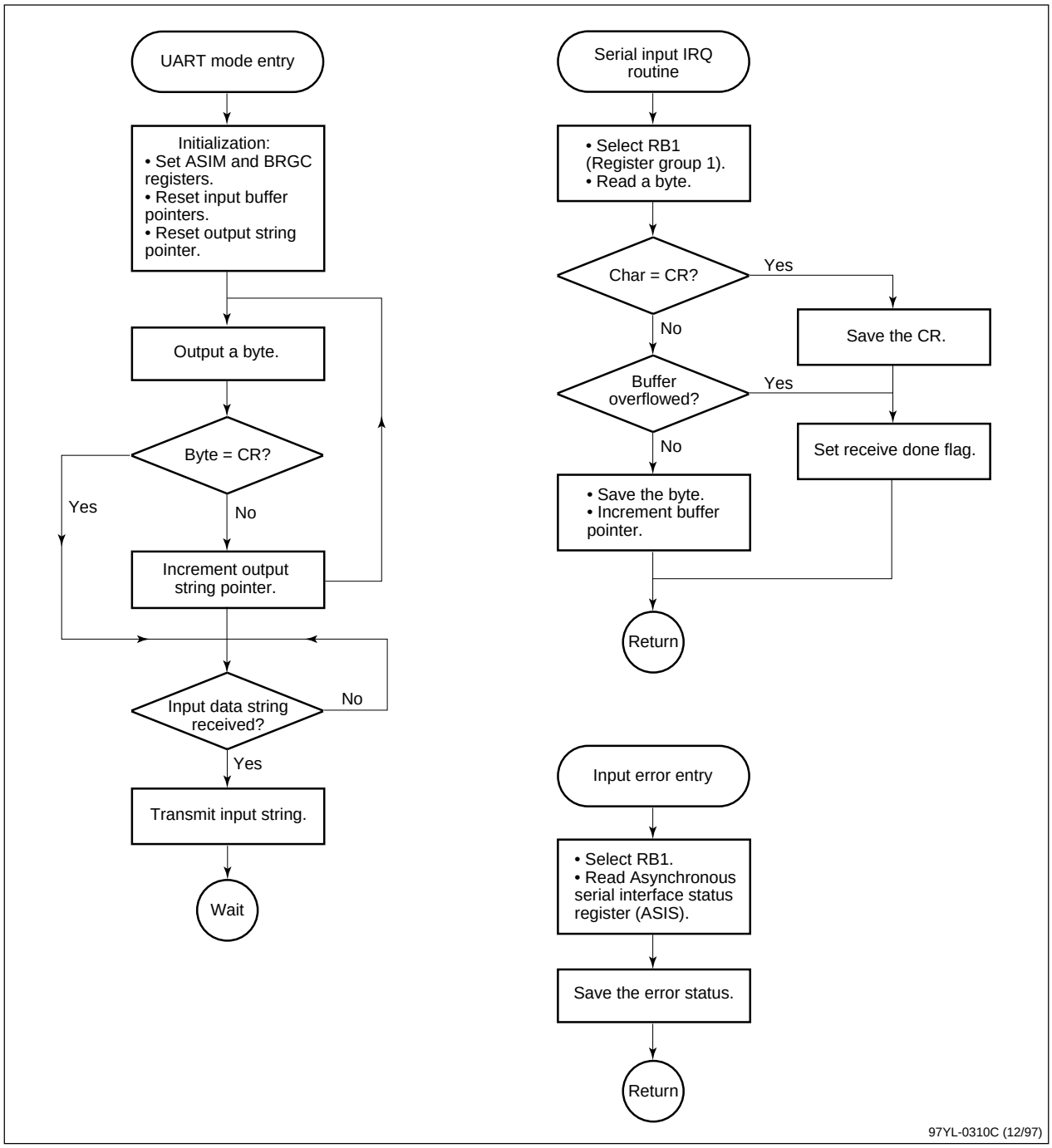
UART Registers

Register Name	Description
_____ register	Transmit shift register (TXS)
	Receive shift register (RXS)
	Receive buffer register (RXB)
Control register	Serial operating mode register (CSIMn)
	Asynchronous serial interface mode register (ASIM)
	Asynchronous serial interface status register (ASIS)
	Baud rate generator control register (BRGC)
	Port mode register 7 (PM7)

Example Program

The example program initializes the internal UART control registers for asynchronous serial interface mode and transmits a UART test string. The serial input data and errors on input data routines are interrupt driven (Figure 5-1).

Figure 5-1. Flowchart of Example Program



Software Specifications

Specification	Description
Serial data format	1 start bit, 7 data bits, even parity bit, 2 stop bits
Baud rate	9600 baud
Output string	UART TEST STARTS
Input module	Interrupt driven; received data is stored in an input buffer
Error detection	Errors in the receiving data are handled by a serial error interrupt routine

Control Register Settings

Figure 5-2. Asynchronous Serial Interface Mode Register (ASIM)

7	6	5	4	3	2	1	0
TXE	RXE	PS1	PS0	CL	SL	ISRM	SCK

Bit	Name	Description
0	SCK	Clock selection 1 = Dedicated baud rate generator output
1	ISRM	Reception completion interrupt 0 = Interrupt generated in case of error generation
2	SL	Transmit data stop bits 1 = 2 bits
3	CL	Character length 0 = 7 bits
5–4	PS1–PS0	Parity bits 11 = Even parity
6	RXE	Receive operation 1 = Enabled
7	TXE	Transmit operation 1 = Enabled

Note: In UART mode, register CSIM2 should be set to 00h.

Figure 5-3. Asynchronous Serial Interface Status Register (ASIS)

7	6	5	4	3	2	1	0
0	0	0	0	0	PE	FE	OVE

Bit	Name	Description
0	OVE	Overrun error flag 0 = Error not generated 1 = Error generated
1	FE	Framing error flag 0 = Error not generated 1 = Error generated
2	PE	Parity error flag 0 = Error not generated 1 = Error generated

Assembly Language Program: Example of UART

```
;*************************************************************************
;                                          File name: UART.ASM
;                                          Date: 9/9/97
;*************************************************************************
;         Description:
; This example program demonstrates the asynchronous serial
; interface (UART). Serial interface channel 2 is set by using serial operation
; mode register 2 (CSIM2), asynchronous serial interface mode register (ASIM),
; asynchronous serial interface status register (ASIS), baud-rate generator
; control register (BRGC), and oscillation mode select register (OSMS).
; The UART mode of serial interface channel 2 transmits or receives 1-byte
; data following a start bit and can perform full-duplex operation.
;
; The example program will output a start-up message 'UART TEST STARTS' and then wait
; for string data up to 20 bytes or a carriage return, and output the input buffer to verify data.
;
;         The main program uses registers in RB0 (register group 0) and the receive interrupt
;         routine uses registers in RB1 (register group 1).
;
;         One frame of transmit/receive data consists of a start bit,
;         8 character bits, no parity bit, and 2 stop bits.
;
;          The baud-rate setting:
;
;                         fxx
;          Baud rate = ------------
;                      2^n x k
;
;                   fxx = Main system clock frequency
;                   n = Value set by TPS0-TPS3 (1 <= n <= 11)
;                   k = Value set by MDL0-MDL3 (0 <= k <= 14)
;
;         Example:
;                   Baud rate: 9600 bauds
;                   Even parity
;                   Start bit: 1
;                   Stop bits: 2
;                   8-bit: LSB first
;                   CTS input pin: p31
;                   RTS output pin: p32
;
;=======================================================================
;         Output test string:'UART TEST STARTS'
;         Non-interrupt-driven TX module
;=======================================================================
;         Input: Serial data stored in RxDataBuffer (10 bytes)
;         Interrupt-driven RX module
;=======================================================================
;         Input error: Stored status of error bits
;=======================================================================
;         for only DDB-K0070A
;
$set (DDB)
;
;
;=======================================================================
;
CR             equ            13             ;Carriage return
LF             equu           10             ;Line feed

;=======================================================================
;         Equates
```

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

RTS_O          equ            P3.2           ;Request-to-send
RTSDirPort     equ            P3.2           ;Request-to-send direction port
CTS_I          equ            P3.1           ;Clear-to-send

;=====================================================================
;          Device selection flag
;=====================================================================
;
;          K0S family
;
K789026 set     0
;
;          K780024, K780034
;
K780024 set     1

;=====================================================================
;
$_if K789026 = 1
RXDirPort      equ            PM2.2          ;RX direction port
TXDirPort      equ            PM2.1          ;TX direction port
TXDataPort     equ            P2.1           ;TX data port

$else

$_if K780024 = 1

RXDirPort      equ            PM2.3          ;RX direction port
TXDirPort      equ            PM2.4          ;TX direction port
TXDataPort     equ            P2.4           ;TX data port

$else

RXDirPort      equ            PM7.0          ;RX direction port
TXDirPort      equ            PM7.1          ;TX direction port
TXDataPort     equ            P7.1           ;TX data port

$endif     ; K7800024

$endif     ; K780026

;=====================================================================
;          saddr area
;          FWA = 0fe20h
;=====================================================================
           dseg       saddr

RXData:        ds                                  ;RX data
RXDataBufferSize equ10                             ;RX data buffer size
RXDataBuffer:  ds             RXDataBufferSize+1   ;Add 1 more byte for CR

ErrorFlags:    ds             1 ;RX Error

OverrunErr     equ            0
FramingErr     equ            1
ParityErr      equ            2

;=====================================================================
;          Even boundary pair
;=====================================================================
           dseg       saddrp
```

```
;=====================================================================
;          Bit segment
;=====================================================================
           bseg
F_RX_end       dbit           ;Reception end flag
F_Err          dbit           ;Reception error flag

RecDone        dbit           ;a string received done

;=====================================================================
;
;          Internal High RAM allocation
;
           dseg       IHRAM
           ds         20h
Stack:

           $ej
;=====================================================================
;          Reset vector
;=====================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           org        18h
           dw         INTSER      ;Serial interface channel 2 UART reception error
           org        1ah
           dw         INTSR       ;End of serial interface channel 2 UART reception
;          org        1ch
;          dw         INTST       ;End of serial interface channel 2 UART transfer

           $ej
;=====================================================================
;          Main
;=====================================================================
Main       cseg       at 80h

ResetStart:
           di                                ;Disable interrupt
;
;          Oscillation mode select register
;           . Does not use divider circuit
;
           mov        OSMS,#00000001b
;
;          Set stack pointer (IHMEM + 20H)
;
           movw       SP,#Stack              ;Stack ptr
;
;          Init. UART register
;
           call       !Init_UART
;
;          Enable INTP0 for DDB-K0070A
;
$if (DDB)
           clr1       PMK0
$endif
           clr1       RecDone                ;string received flag = 0
           ei
;
;          Output start message
;
           mov        c,#TestStringSize
```

```
          movw      hl,#TestString
          call      !OutputAString
;
;         Wait for a string to be received
;

          bf        RecDone,$$              ;string received flag = 0
;
;         Output the input buffer data
;
          mov       c,#RxDataBufferSize
          movw      hl,#RxDataBuffer
          call      !OutputAString
;
;         Reset flag to 0

          clr1      RecDone
;
          br        $$


TestString:
          db        'UART TEST STARTS',CR,LF
TestStringEnd:

TestStringSizeequ   TestStringEnd - TestString

;=====================================================================
;         Module name: Init_UART
;
;         DESCRIPTION:
;                   .This module is to initialize UART control and mode
;                    registers.
;
;         OPERATION:
;                   .Set dual ports to input for RX and output for TX
;                    and latch the TX port to high.
;                   . Data format: 1 start, 8-bit data, 1 stop, no parity.
;                   . Set baud rate to 9600.
;                   . Enable RX and TX operation.
;                   . Clear all variables and flags for serial data transfer.
;=====================================================================
Init_UART:
;
;         p3.1 = CTS (input), P3.2 = RTS (output)
;
          set1      RTS_O
          clr1      RTSDirPort
;
;         Set TX/RX port direction and output latch
;
          set1      RXDirPort               ;RX set to input direction
          clr1      TXDirPort               ;TX set to output direction
          set1      TXDataPort              ;Latch output high
;
;         Baud rate (9600 bauds/err=1.73 %)
;
          mov       BRGC,#10010000b ;9600 bauds err = 1.73%/MCS = 1, fx = 5 MHz
```

```
;*************************************************************
;*         Asynchronous serial interface mode register (ASIM) *
;*                                                           *
;*          1 1 1 1 0 1 0 1                                  *
;*          ^ ^ ^ ^ ^ ^ ^ ^                                  *
;*          | | | | | | | |__ 1 Dedicated baud rate generator output*
;*          | | | | | | |                                    *
;*          | | | | | | |____ 0 Reception error interrupt *
;*          | | | | | |                                      *
;*          | | | | | |______ 1 2 stop bits                   *
;*          | | | | |                                        *
;*          | | | | |________ 0 7-bit data                    *
;*          | | | |                                          *
;*          | | | |___________ 1 Even parity                 *
;*          | | |                                          *
;*          | | |_____________ 1 Even parity                *
;*          | |                                            *
;*          | |_______________ 1 RX operation enabled *
;*          |                                              *
;*          |________________ 1 TX operation enabled *
;*                                                           *
;*************************************************************/
          mov       ASIM,#11110101b

;
;         Set TX done interrupt flag to 1
;
          set1      STIF
;
;         Clear reception end receive error interrupt request flags
;
          clr1      SERIF
          clr1      SRIF
;
;         Clear software flags
;
          clr1      F_Rx_end              ;Reception end
          clr1      F_Err                 ;Error in data reception
;
;         Set receive buffer ptr
;
          sel       RB1
          movw      hl,#RxDataBuffer
          mov       b,#0                  ;offset to 0
;
;         Set to background register bank
;
          sel       RB0
;
;         Enable reception end and receive error interrupts
;
          clr1      SERMK
          clr1      SRMK
;         No TX interrupt driven
          set1      STMK
          ret
;
          $ej

;=====================================================================
;         Module name: OutputAString
;
;         DESCRIPTION:
```

```
;                   .This module outputs data in the buffer pointed by the HL register
;                    up to carriage return, a null, or maximum buffer size.
;
;         OPERATION:
;                   . Read a byte from the buffer and incr. offset in B register.
;                   . Check if the byte is the terminator.
;                   . If no output, use the byte else return.
;
;         Input registers:
;          HL = output buffer ptr
;          c = size of buffer
;=======================================================================
OutputAString:
          mov       b,#0
OutLoop:
          mov       a,[hl+b]
          cmp       a,#CR
          bz        $tx10                   ;no more
          cmp       a,#0
          bz        $tx10                   ;no more
          xch       a,b                     ;save a register
          cmp       a,c
          xch       a,b
          bnc       $tx10                   ;full, exit

          call      !OutAByte
;         Incr. data ptr
          inc       b
          br        OutLoop
; output a CR
tx10:
          mov       a,#CR
          call      !OutAByte
          mov       a,#LF
          call      !OutAByte
          ret

;=======================================================================
;         Module name: OutAByte
;
;         DESCRIPTION:
;                   .This module outputs a byte in the A register.
;
;         OPERATION:
;                   . Wait for TX to be done.
;                   . Clear TX done flag.
;                   . Output a character to the TX buffer.
;
;         Input registers:
;                   A register
;=======================================================================
OutAByte:
;         Wait for Tx done
          bf        STIF,$$
          clr1      STIF
;         Output a data
          mov       TXS,a
          ret
```

```
;=====================================================================
;          Module name: INTSR
;
;          DESCRIPTION:
;                    .This module is the interrupt service routine for a
;                     receive operation.
;
;          OPERATION:
;                    . Switch bank to 1.
;                    . Disable request to send (RTS).
;                    . Read a byte in the RX buffer.
;                    . Save the byte to the input buffer.
;                    . Incr. offset pointer.
;                    . If number of bytes received exceeds 10 bytes or
;                      character received is CR, then set buffer full flag
;                      and reset offset pointer to 0.
;                    . Enable RTS.
;
;          Input registers:
;           RXB = data
;           HL = buffer ptr
;           B reg = offset
;
;=====================================================================
INTSR:
           sel       RB1
;          Set RTS - Busy
           set1      RTS_O
;
;          Receive data
;
           mov       a,RXB
           set1      F_RX_end               ;Set a character received
;          Save to the buffer
           mov       [hl+b],a
           inc       b
; check end
           cmp       a,#CR                  ;CR?
           bz        $yesCr                 ;Yes
;
           mov       a,b
           cmp       a,#RXDataBufferSize
           bc        $not full
yesCr:
; Rec buffer is full
           set1      RecDone                ;Set buffer full flag
           mov       b,#0                   ;reset offset
;
notfull:
;          Clear RTS
           clr1      RTS_O
;          Clear error flag
           clr1      F_Err
;
           reti

           $ej
```

```
;=========================================================================
;          Module name: INTSER
;
;          DESCRIPTION:
;                     .This module is the interrupt service routine for a
;                      reception error.
;
;          OPERATION:
;                     . Switch bank to 1.
;                     . Read error status and store in the error flags variable (ErrorFlags).
;=========================================================================
INTSER: ;Serial interface channel 2 UART reception error

          sel       RB1
;
;         Check reception error
;
          mov       a,ASIS
          mov       ErrorFlags,a
;
;         Set an error flag
;
          set1      F_Err
;
          reti

;*************************************************************************
          end
```

C Language Program: Example of UART

```
/*;*****************************************************************************
;                                          File name: UART.C
;                                          Date: 9/9/97
;*****************************************************************************
;         Description:
; This example program demonstrates the asynchronous serial
; interface (UART). Serial interface channel 2 is set by using serial operation
; mode register 2 (CSIM2), asynchronous serial interface mode register (ASIM),
; asynchronous serial interface status register (ASIS), baud-rate generator
; control register (BRGC), and oscillation mode select register (OSMS).
; The UART mode of serial interface channel 2 transmits or receives 1-byte
; data following a start bit and can perform full-duplex operation.
;
; The example program will output a start-up message 'UART TEST STARTS' and then wait
; for string data up to 20 bytes or a carriage return, and output the input buffer to verify data.
;
;         The main program uses registers in RB0 (register group 0) and the Receive interrupt
;         routine uses registers in RB1 (register group 1).
;
;         One frame of transmit/receive data consists of a start bit,
;         8 character bits, no parity bit, and 2 stop bits.
;
;          The baud rate setting:
;
;                         fxx
;          Baud rate = ------------
;                      2^n x k
;
;                   fxx = Main system clock frequency
;                   n = Value set by TPS0-TPS3 ( 1 <= n <= 11)
;                   k = Value set by MDL0-MDL3 ( 0 <= k <= 14)
;
;         Example:
;                   Baud rate: 9600 bauds
;                   Even parity
;                   Start bit: 1
;                   Stop bits: 2
;                   8-bit: LSB first
;                   CTS input pin: p31
;                   RTS output pin: p32

;
;=====================================================================
;         Output test string:'UART TEST STARTS'
;         Non-interrupt-driven TX module
;=====================================================================
;         Input: Serial data stored in RxDataBuffer (10 bytes)
;         Interrupt-driven RX module
;=====================================================================
;         Input error: Stored status of error bits
;=====================================================================
;=====================================================================*/
//
//
//
#include "define.h"
//
//        interrupt vectors
//
#pragma interrupt INTSER IntSerialError RB1
#pragma interrupt INTSR IntSerialInRB1
```

```
/*;============================================================================
;         Equates
;============================================================================*/

#define RTS_O                 P3.2          // ;Request-to-send
#define RTSDirPort            P3.2          // ;Request-to-send direction port
#define CTS_I                 P3.1          // ;Clear-to-send

/*;============================================================================
;         Device selection flag
;============================================================================*/
//;
//;         K0S family
//;
// #define K789026            0
//;
//;         K780024, K780034
//;
// #define K780024            1

//;============================================================================
//;
#ifdef K789026

#define RXDirPort             PM2.2         //;RX direction port
#define TXDirPort             PM2.1         //;TX direction port
#define TXDataPort            P2.1          //;TX Data port

#else

#ifdef K780024

#define RXDirPort             PM2.3         //;RX direction port
#define TXDirPort             PM2.4         //;TX direction port
#define TXDataPort            P2.4          //;TX data port

#else

#define RXDirPort             PM7.0         //;RX direction port
#define TXDirPort             PM7.1         //;TX direction port
#define TXDataPort            P7.1          //;TX data port


#endif    //; K780024

#endif    //; K789026

/*;============================================================================
;         saddr area
;         FWA = 0fe20h
;============================================================================*/
__sreg unsigned char RXData;
__sreg unsigned char bptr;
__sreg unsigned char RXDataBuffer[10];
__sreg unsigned char TXData;
__sreg unsigned char i;

__sreg unsigned ErrorFlags;

#define OverrunErr            0
#define FramingErr            1
#define ParityErr             2
```

```
unsigned char const TestString[] = {"UART TEST STARTS\n"};

/*;=========================================================================
;          Bit Segment
;=========================================================================*/

bit F_Err;                    //;Reception error flag
bit RecDone    ;              //;a string received Done


void OutAByte ();
void Init_UART ();

/*;=========================================================================
;          Main
;=========================================================================*/
void main ()
{

//;
//;         Oscillation mode select register
//;         . Does not use divider circuit
//;
          OSMS = 0b00000001;
//;
//;         Init. UART register
//;
          Init_UART ();
//;
//;         Enable INTP0 for DDB-K0070A
//;
#ifdef DDB
          PMK0 = 0;
#endif
          RecDone = 0;//;string received flag = 0
          EI ();
//;
//;         Output start message
//;
          for (i=0;i < sizeof TestString ; i++)
          {
                    TxData = TestString[i];
                    OutAByte ();
                    if (TxData == '\n') break; // == LF
          }
          TxData = CR;
          OutAByte ();
//;
//;         Wait for a string to be received
//;


          while (TRUE) { if ( RecDone ) break;}
//;
//;         Output the input buffer data
//;
          for (i=0;i < sizeof RxDataBuffer ; i++)
          {
                    TxData = RxDataBuffer[i];
                    OutAByte ();
                    if (TxData == CR ) break;
          }
          TxData = LF;
          OutAByte ();
//;
```

```
//;         Reset flag to 0

          RecDone = 0;
//;
          while (TRUE);
}

/*;=====================================================================
;         Module name: Init_UART
;
;         DESCRIPTION:
;                   .This module is to initialize UART control and mode
;                    registers.
;
;         OPERATION:
;                   .Set dual ports to input for RX and output for TX
;                    and latch the TX port to high
;                   . Data format:1 start, 8-bit data, 1 stop, no parity
;                   . Set Baud rate to 9600.
;                   . Enable RX and TX operation.
;                   . Clear all variables and flags for serial data transfer
;=====================================================================*/
void Init_UART ()
{
//;
//;         p3.1 = CTS (input), P3.2 = RTS (output)
//;
          RTS_O     = 1;
          RTSDirPort = 0;
//;
//;         Set TX/RX port direction and output latch
//;
          RXDirPort =1;                     //;RX set to input direction
          TXDirPort =0;                     //;TX set to output direction
          TXDataPort=1;                     //;Latch output high
//;
//;         Baud rate (9600 baud/err=1.73%)
//;
          BRGC = 0b10010000; //;9600 baud err = 1.73%/MCS = 1, fx = 5 MHz


/*************************************************************
;*         Asynchronous serial interface mode register (ASIM) *
;*                                                            *
;*         1 1 1 1 0 1 0 1                                    *
;*         ^ ^ ^ ^ ^ ^ ^ ^                                    *
;*         | | | | | | | |__ 1 Dedicated baud rate generator output*
;*         | | | | | | |                                      *
;*         | | | | | | |____ 0 Reception error interrupt *
;*         | | | | | |                                        *
;*         | | | | | |______ 1 2 stop bits                     *
;*         | | | | |                                          *
;*         | | | | |________ 0 7-bit data                     *
;*         | | | |                                            *
;*         | | | |___________ 1 Even parity                  *
;*         | | |                                            *
;*         | | |_____________ 1 Even parity                 *
;*         | |                                              *
;*         | |_________________ 1 RX operation enabled *
;*         |                                                *
;*         |___________________ 1 TX operation enabled *
;*                                                            *
;*************************************************************/
          ASIM = 0b11110101;
```

```
//;
//;         Set TX done interrupt flag to 1
//;
          STIF = 1;
//;
//;         Clear reception end receive error interrupt request flags
//;
          SERIF = 0;
          SRIF = 0;
//;
//;         Clear software flags
//;
          F_Err = 0; // ;Error in data reception
//;
//;         Enable reception end and receive error interrupts
//;
          SERMK = 0;
          SRMK = 0;
//;         No TX interrupt driven
          STMK = 1;
}

/*;=====================================================================
;         Module name: OutAByte
;
;         DESCRIPTION:
;                   .This module outputs a byte in the A register.
;
;         OPERATION:
;                   . Wait for TX to be done.
;                   . Clear TX done flag.
;                   . Output a character to the TX buffer.
;
;         Input registers:
;                   TXData = data byte
;=====================================================================*/
void OutAByte ()
{
//;         Wait for TX done
          while (TRUE) { if ( STIF ) break; }
          STIF = 0;
//;         Output a data
          TXS = TXData;
}

/*;=====================================================================
;         Module Name: INTSR
;
;         DESCRIPTION:
;                   .This module is the interrupt service routine for a
;                    receive operation.
;
;         OPERATION:
;                   . Switch bank to 1.
;                   . Disable request to send (RTS).
;                   . Read a byte in the RX buffer.
;                   . Save the byte to the input buffer.
;                   . Incr. offset pointer.
;                   . If number of bytes received exceeds 10 bytes or
;                     character received is CR, then set buffer full flag
;                     and reset offset pointer to 0.
;                   . Enable RTS.
;
```

```
;          Input registers:
;           RXB = data
;           HL = buffer ptr
;           B reg = offset
;
;=========================================================================*/
void IntSerialIn ()
{
          RTS_O = 1;
//;
//;         Receive data
//;
          RxData = RXB;
          RxDataBuffer[bptr++] = RxData;
          if (RxData == CR || bptr >= size of RxDataBuffer)
          {
                    bptr = 0;
                    RecDone = 1;
          }
//;         Clear RTS
          RTS_O = 0;
//;         Clear error flag
          F_Err = 0;
;
}

/*;=========================================================================
;          Module name: INTSER
;
;          DESCRIPTION:
;                    .This module is the interrupt service routine for a
;                     reception error.
;
;          OPERATION:
;                    . Switch bank to 1.
;                    . Read error status and store in the error flags variable (ErrorFlags).
;=========================================================================*/
void IntSerialError ()
{
//;
//;         Check reception error
//;
          ErrorFlags = ASIS;
//;
//;         Set an error flag
//;
          F_Err     = 1;
//;

}
```

NEC

Software UART 6

Applicable Devices

- All K0 microcontrollers
- All K0S microcontrollers

Description

A software UART is a software-driven, asynchronous serial interface I/O. The software UART needs a standard output port and a standard input port with an interrupt enable on the input port. The data transmission timing is carried out with an 8-bit timer that generates a bit or a half-bit time for a given baud rate. A frame of transmit/receive data consists of a start bit, character bits, stop bits, and a parity bit. Unlike the hardware UART, the frame size of the software UART can be easily changed, and no other control registers are required except an 8-bit timer control register. The software UART contains the following registers:

- Output port for Tx
- Input port for Rx (This port should be interrupted when a falling edge is detected on the pin.)
- 8-bit timer control registers

Setting Baud Rate

A software UART uses an 8-bit timer to set the baud rate. The bit time of a baud rate is calculated as bit time in μs = 10^6/baud rate. In other words, for 9600 baud rate, 1,000,000/9600 = 104.1667 μs.

Example Program

This program demonstrates the functionality of a software UART by using an output port and an input port with an interrupt on a falling edge of a serial data. When the program starts, it calls a software UART initialization routine to set the ports and 8-bit timer. Once the initialization is complete, it outputs a test string “Send a test string” and waits for a string of data from the host. Each received data is echoed back to the host as well as stored in the buffer. If the string is terminated by a carriage return, the entire data string received is sent to the host. A parity check can be easily implemented by adding 1s in the character and 1s in the parity bit. If odd parity is being checked, the 1s in the character and the parity bit must result in an odd number.

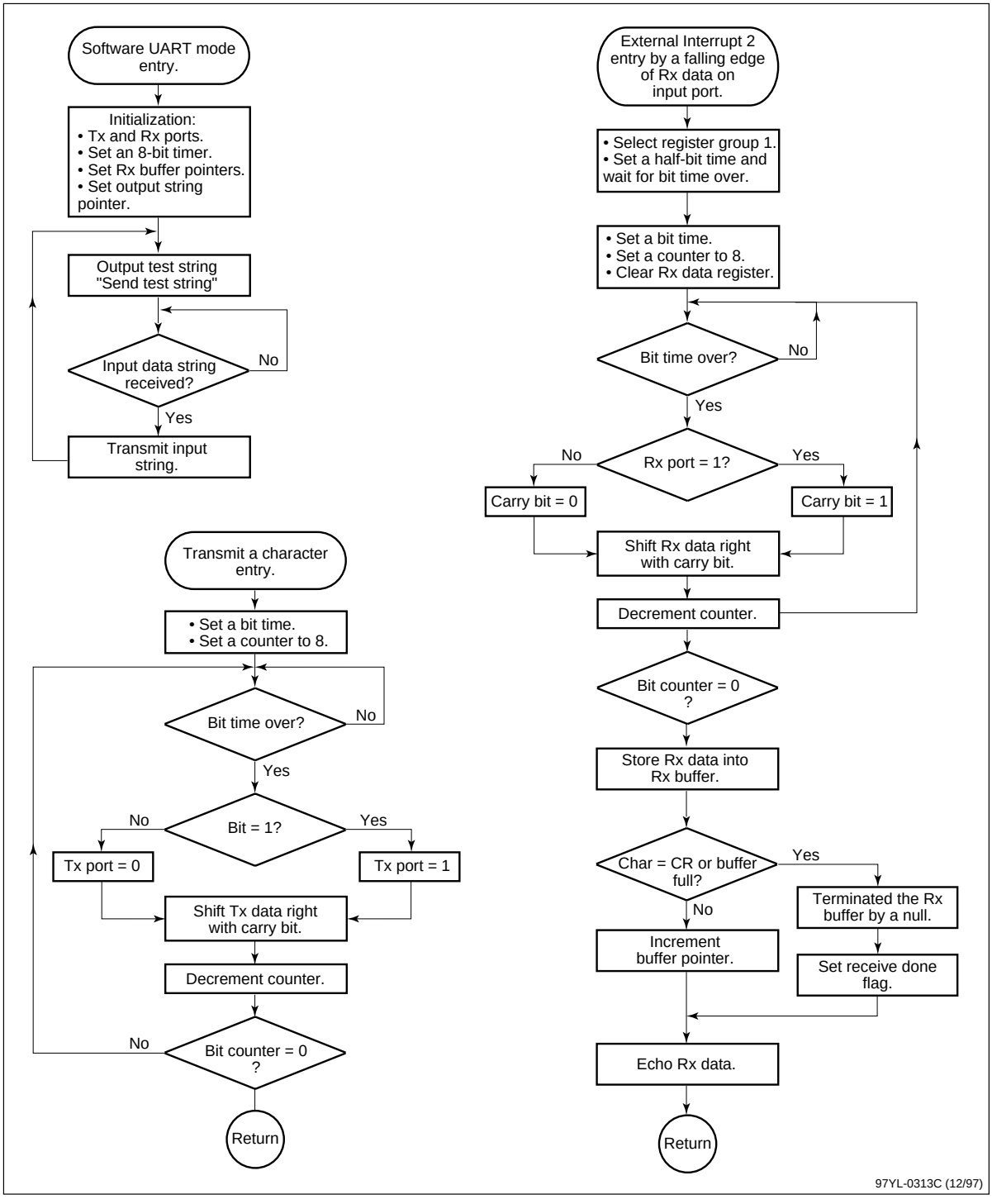
Specifications

Specification	Description
Modules	Main loop
	Output a data string
	Output a character
	Input a character
	Initialization
Baud rate	9600 baud
Data format	1 start bit, 8 data bits, 2 stop bits, and no parity bit

Usage Restrictions

- Receiving cannot be faster than 19.2K because of the relatively slow input data service time.
- When a frame is transmitted, all other interrupts must be disabled because an interrupt will cause a bit-time error.
- The software UART will not work if Tx and Rx data is simultaneously transmitted and received.

Figure 6-1. Flowchart of a Software UART Program



Assembly Language Program: Example of Software UART Implementation

```
;*************************************************************************
;                                        File name: SOFTUART.ASM
;                                        Date: 9/16/97
;*************************************************************************
;         Description:
;
; This program implements a software UART.
;
; The serial data pin P02 (INTP2) is used for Rx data and P01 for Tx data.
;
;         Tx data: Output a test string.
;         Rx data: Data from RS-232 is stored in the RxBuffer when INTP2 is interrupted
;                  by the start bit of a character.
;
;         The baud rate for this demo module is set to 9600 baud.
;
;
; Description:
; -------------
;
; Serial data is transmitted or received with a fixed baud rate. The
; standard baud rates are:
;
;         300, 600, 1200, 2400, 4800, 9600, 14.4k, 19.2k, ...
;
;         To calculate the bit time of a baud rate, an 8-bit timer (tc0)
;         is used to set a bit time or half-bit (HBt) time.
;
;         Formula of a bit time:
;
;                  Bit time = fxx/baud_rate
;
;                  fxx = fx/2^n (selected by timer clock select register)
;
;         For example, 9600 baud rate (fxx = 1.25 MHz):
;
;                  1,250,000/9600 = 130.2083 µs
;
;         Serial data format for this example program:
;
;                  1 start bit, 8 data bits, 2 stop bits, no-parity bit.
;
;         LIMITS:
;         -------
;
;         . Software UART may not be usable if the baud rate is faster than
;           19.2k because the input data service time is not fast enough
;           to service the data.
;
;         . When a character is transmitted, all other interrupts must be disabled
;           because an interrupt will cause a bit-time error.
;
;         . It will not work if Tx and Rx data are simultaneously transmitted
;           and received.
;
;         This example uses 9600 baud rate.
;
;*****************************************************************************
;=====================================================================
;         Output test string:'Send a test data string',0
;         Non-interrupt-driven TX module
;=====================================================================
```

```
;         Input: Serial data stored in RxDataBuffer (80 bytes)
;         Interrupt-driven RX module
;
;=====================================================================
;         for only DDB-K0070A
;
$set (DDB)
;
;
;=====================================================================
;
CR        equ       13                    ;Carriage return
LF        equ       10                    ;Line feed

;=====================================================================
;         Equates
;=====================================================================
;//
;//         Bit time of some baud rates based on fx = 5 MHz
;//         Counter resolution
;//         fxx = 1.25 MHz (0.8 µs)
;//
BitTime_BaudRate384k          equ 33          ;// Theoretical count 32.5521
BitTime_BaudRate192k          equ 65          ;// 65.1042
BitTime_BaudRate144k          equ 87          ;// 86.8056

BitTime_BaudRate9600          equ 130         ;// 130.2083
HBtTime_BaudRate9600          equ 65          ;// 65.1041
;/;
;/;         K0S family
;/;
;/ #define K0S
;
$set (K0)

$if (K0)

TxData                        equ P0.1       ;// TX
TxDataPDir                    equ PM0.1      ;// TX data dir

RxData                        equ P0.2       ;// RX
RxDataPDir                    equ PM0.2      ;// RX data dir

$else

TxData                        equ P3.1       ;// TX
TxDataPDir                    equ PM3.1      ;// TX data dir

RxData                        equ P3.2       ;// RX
RxDataPDir                    equ PM3.2      ;// RX data dir

$endif

;=====================================================================
;         Saddr area
;         FWA = 0fe20h
;=====================================================================
          dseg      saddr

;=====================================================================
;         Even-boundary pair
;=====================================================================
          dseg      saddrp
```

```
;=====================================================================
;          Bit segment
;=====================================================================
           bseg
Full                dbit                    ;buffer full

;=====================================================================
;
;          Internal High RAM allocation
;
;=====================================================================
           dseg       IHRAM
           ds         20h
Stack:

RxBufferSize                   equ 80
RxBuffer:                      ds  (RxBufferSize)


;=====================================================================
;
;          macro
;
;;WaitT1Int while (TRUE){if (TMIF1) break;} TMIF1 = 0
;=====================================================================
WaitT1Int  macro
           bf         TMIF1,$$
           clr1       TMIF1
           endm

           $ej
;=====================================================================
;          Reset vector
;=====================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           org        0Ah
           dw         INTERRUPT2

           $ej
;=====================================================================
;          Main
;=====================================================================
Main       cseg       at 80h

ResetStart:
           di                                 ;Disable interrupt
;
;          Oscillation mode select register
;           . Does not use divider circuit
;
           mov        OSMS,#00000001b
;
;          Set stack pointer (IHMEM + 20H)
;
           movw       SP,#Stack               ;Stack ptr
;.
;          Init UART Register
;
           call       !Init_UART
;
;          Enable INTP0 for DDB-K0070A
;
```

```
$if (DDB)
          clr1      PMK0
$endif

Main_Loop:
          clr1      Full
          ei
;
;         Output start message
;
          mov       c,#TestStringSize
          movw      hl,#TestString
          call      !OutputAString
;
;         Wait for a string to be received
;
          bf        Full,$$ ;string received flag = 0
          clr1      Full
;
;         Output the input buffer Data
;
          mov       a,#LF
          call      !OutAByte
          mov       a,#CR
          call      !OutAByte

          mov       c,#RxBufferSize
          movw      hl,#RxBuffer
          call      !OutputAString

          mov       a,#LF
          call      !OutAByte
          mov       a,#CR
          call      !OutAByte

          br        $Main_Loop
;
;
TestString:
          db 'Send a test data string',0
TestStringEnd:

TestStringSizeequ    TestStringEnd - TestString

;=====================================================================
;         Module name: Init_UART
;
;         DESCRIPTION:
;                   .This module is to initialize UART control and mode
;                    registers.
;
;         OPERATION:
;                   .Set dual ports to input for Rx and output for Tx
;                    and latch the Tx port to high.
;                   . Data format:
;                   . 1 start bit, 8-bit data, 2stop bits, no parity bit.
;                   . Set baud rate to 9600.
;                   . Enable Rx and Tx operation.
;                   . Clear all variables and flags for serial data transfer.
;=====================================================================
Init_UART:
;
;         Set Port
;
```

```
          set1      RxDataPDir              ; Input direction
          clr1      TxDataPDir              ; Output direction
          set1      TxData                  ; Set to high
;
;         Set interrupt for INTP2
;
;         PMK2 = 0                          ;// Unmask INTP2
;         INTM1 = 0                         ;// all falling edge interrupt
          clr1      PMK2
          mov       INTM1,#0
;
;         Baud rate = 9600 baud
;         Set timer 0 Register
;         Tick = 1.25 MHz (800 ns)
;         Disable 8-bit timer
;
;          TCL1 = 0b00000111                ;// Select 1.25 MHz as count clock
;          TMC1 = 0B00000000                ;// stop operation
;          TOC1 = 0B00000000                ;// No output compare
;
          mov       TCL1,#00000111b         ;// Select 1.25 MHz as count clock
          mov       TMC1,#00000000b         ;// stop operation
          mov       TOC1,#00000000b         ;// No output compare
;
;         Clear software flags
;         Set receive buffer ptr
;
          sel       RB1
          movw      hl,#RxBuffer
          mov       b,#0                    ;offset to 0
;
;         Set to background register bank
;
          sel       RB0
          ret
;
          $ej

;=========================================================================
;         Module name: OutputAString
;
;         DESCRIPTION:
;                   .This module outputs data in the buffer pointed by the HL register
;                    up to carriage return, a null, or maximum buffer size.
;
;         OPERATION:
;                   . Read a byte from the buffer and increment offset in B register.
;                   . Check if the byte is the terminator.
;                   . If no output,use the byte else return.
;
;         Input registers:
;          HL = output buffer ptr
;          c = size of buffer
;=========================================================================
OutputAString:
          mov       e,#0
OutLoop:
          mov       a,e
          mov       b,a
          mov       a,[hl+b]
          cmp       a,#CR
          bz        $tx10                   ;no more
          cmp       a,#0
          bz        $tx10                   ;no more
```

```
          xch       a,e                     ;Save a register
          cmp       a,c
          xch       a,e
          bnc       $Tx10                   ;full, exit

          call      !OutAByte
;         Incr. data ptr
          inc       e
          br        OutLoop
; output a CR
Tx10:
          mov       a,#CR
          call      !OutAByte
          mov       a,#LF
          call      !OutAByte
          ret

;=====================================================================
;         Module name: OutAByte
;
;         DESCRIPTION:
;                   .This module outputs a byte in the A register.
;
;         OPERATION:
;                   . Wait for Tx to be done.
;                   . Clear Tx done flag.
;                   . Output a character to the Tx buffer.
;
;         Input registers:
;                   A register
;         used B reg.
;=====================================================================
OutAByte:
;         Set a bit time
          mov       CR10,#082H              ; 130
;
;         Start bit
;
          set1      TMC1.0
          clr1      TxData
;
;         Read 8 Bits
;
          mov       b,#8H ; 0
outlp:
; wait for a bit time
          WaitT1Int
          bf        a.0,$?L0028
          set1      TxData
          br        $?L0029
?L0028:
          clr1      TxData
?L0029:
          clr1      CY
          rorc      a,1
          dbnz      b,$outlp
          WaitT1Int                         ; // wait for a bit time
;
;         two stop bits
;
          set1      TxData
          WaitT1Int                         ; // wait for a bit time
          WaitT1Int                         ; // wait for a bit time
          clr1      TMC1.0
```

```
          ei
          ret

;=====================================================================
;         Module name: INTSR
;
;         DESCRIPTION:
;                   .This module is the interrupt service routine for
;                    receive operation.
;
;         OPERATION:
;                   . Switch bank to 1.
;                   . Disable request to send (RTS).
;                   . Read a byte in the Rx buffer
;                   . Save the byte to the input buffer.
;                   . Incr. offset pointer.
;                   . If number of bytes received exceeds 10 bytes or
;                     character received is CR, then set buffer full flag
;                     and reset offset pointer to 0.
;                   . Enable RTS
;
;         Input registers:
;          HL = buffer ptr
;          B reg = offset
;
;=====================================================================
INTERRUPT2:
          sel       RB1
;
;         Half-bit time
;
          mov       CR10,#HBtTime_BaudRate9600 - 8 ; 8*4 Cycles Compensation
          set1      TMC1.0
          WaitT1Int                          ; // wait for a bit time
;
          mov       CR10,#BitTime_BaudRate9600;
          mov       a,#0
          mov       c,#8
Rxlp:
          WaitT1Int                         ;// wait for a bit time
          clr1      cy
          bf        RxData,$rx10
          set1      cy
Rx10:
          rorc      a,1
          dbnz      c,$rxlp
;
;         Save data
;
;         Save to the buffer
          mov       [hl+b],a
          mov       d,a
          inc       b
; check end
          cmp       a,#CR                                   ;CR?
          bz        $yesCr                                  ;Yes
;
          mov       a,b
          cmp       a,#RxBufferSize
          bc        $notfull
yesCr:
          dec       b
;         Store a terminator
          mov       a,#0
```

```
            mov         [hl+b],a                ;terminator
;
;           Rec buffer is full
;
            set1        Full                    ;Set buffer full flag
            mov         b,#0                    ;reset offset
;
;           Output a char.
;
notfull:
;
;           Reset int2 flag
;
            clr1        PIF2
;
;           Echo the char.
;
            push        bc
            mov         a,d
            call        ! OutAByte
            pop         bc
;
            reti

;*******************************************************************
            end
```

C Language Program: Example of Software UART Implementation

```
#define DEBUG 0

#pragma interrupt INTP2 Interrupt2 RB1

/*****************************************************************************
;                                                          File Name: SOFTUART.C
;                                                          Date: 9/16/97
;
; This program is an application for a software UART.
;
; The serial data pins are P02 (INTP2) for Rx data and P01 for Tx data (K0 family).
;
;         Tx data: Output a test string.
;         Rx data: Data from RS-232 is stored in the RxBuffer when INTP2 is interrupted
;                  by the start bit of a character.
;
;         The baud rate for this demo module is set to 9600.
;
;
; Description:
; -------------
; Serial data is transmitted or received with a fixed baud rate. The
; standard baud rates are:
;
;         300, 600, 1200, 2400, 4800, 9600, 14.4k 19.2k, ...
;
;         To calculate the bit time of a baud rate, an 8-bit timer (tc0)
;         is used to set a bit time or half-bit (HBt) time.
;
;         Formula of a bit time:
;
;                  Bit time = fxx/baud_rate
;
;                  fxx = fx/2^n (selected by timer clock select register)
;
;         For example, 9600 baud rate (fxx = 1.25 MHz):
;
;                  1,250,000 / 9600 = 130.2083 µs
;
;         Serial data format for this application example program:
;
;                  1 start bit, 8 data bits, 2 stop bits, no parity bit.
;
;         LIMITS:
;         -------
;
;         Software UART may not be usable if the baud rate is faster than
;         19.2k because the input data service time is not fast enough
;         to service the data.
;
;         When a character is transmitted, all other interrupts must be disabled
;         because an interrupt will cause bit-time error.
;
;         It will not work if Tx and Rx data are simultaneously transmitted
;         and received.
;
;         This example uses 9600 baud rate.
;
*****************************************************************************/
#include "define.h"
//
```

```
//        Bit time of some baud rates based on fx = 5 MHz
//        Counter resolution
//        fxx = 1.25 MHz (0.8 µs)
//
#define BitTime_BaudRate384k                 33           ;// Theoretical count 32.5521
#define BitTime_BaudRate192k                 65           ;// 65.1042
#define BitTime_BaudRate144k                 87           ;// 86.8056

#define BitTime_BaudRate9600                 130          ;// 130.2083
#define HBtTime_BaudRate9600                 65           ;// 65.1041
//
//

//#define K0S

#ifndef K0S

#define TxData                               P0.1          //TX
#define TxDataPDir   PM0.1                                 //TX data dir

#define RxData                               P0.2          //RX
#define RxDataPDir   PM0.2                                 //RX data dir

#else

#define TxData                               P3.1          //TX
#define TxDataPDir   PM3.1                                 //TX data dir

#define RxData                               P3.2          //RX
#define RxDataPDir   PM3.2                                 //RX data dir

#endif


#define WaitT1Intwhile (TRUE){if (TMIF1) break;} TMIF1 = 0

#if 1
unsigned char const TestString[] = "Send a test data string";
#else
unsigned char const TestString[] = {0x55,0x55,0};
#endif

#define RxBufferSize80
unsigned char RxBuffer[RxBufferSize];

__sreg unsigned char i,j,k,c;

__sreg unsigned char ir,ichar, bptr;

bit full;

void SerialTx (unsigned char c);

void Main ()
{
          DI ();

          OSMS = 0b00000001;                 // select 5 MHz
          PMK0 = 0;                          // for DDB BOARD - INTP0 has to be set

                                             // Set timer 0 Register

          TCL1 = 0b00000111;                 // Select 1.25 MHz as count clock
          TMC1 = 0B00000000;                 // stop operation
```

```
          TOC1 = 0B00000000;                // No output compare

          RxDataPDir = 1;                   // set to input dir. for Rx
          TxDataPDir = 0;                   // Set to output dir. for Tx
          TxData     = 1;                   // Set to high

          PMK2 = 0;                         // Unmask INTP2
          INTM1 = 0;                        // all falling edge interrupt

          bptr = 0;

          EI ();

          while (TRUE)
          {
                    Full = 0;
                    //
                    // clear Rx buffer
                    //
                    for (i = 0; i < sizeof RxBuffer; i++) RxBuffer[i++] = 0;

                    i = 0;
                    while (TRUE)
                    {
                                        c = TestString[i++];
                                        if ( !c ) break;
                                        SerialTx (c);
                    }
                    // Output CR/LF
                    SerialTx (0x0D);
                    SerialTx (0x0A);

                    // CHECK input buffer full

                    while (!Full);

                    SerialTx (0x0A);        //LF
                    for (i=0;i< strlen (RxBuffer) ; i++)
                    {
                                        c = RxBuffer[i];
                                        if ( !c ) break;
                                        SerialTx (c);
                    }
                    // Output CR/LF
                    SerialTx (0x0D);        //CR
                    SerialTx (0x0A);        //LF
          }

}

/*************************************************************************=*****
*         Output a character to serial port
*         Send from LSB to MSB
*************************************************************************=*****/
void SerialTx (unsigned char c)
{

register unsigned char i;

          DI ();

          CR10 = BitTime_BaudRate9600;
          TCE1 = 1;  // Enable tc1 only
          //
```

```
          // Start Bit
          //
          TxData = 0;
          //
          // Send Data
          //
          for (i = 0; i < 8; i ++)
          {
                    WaitT1Int;              // wait for a bit time
                    if ( c & 1) TxData = 1; else TxData = 0;
                    c >>= 1;

          }
          WaitT1Int; // wait for a bit time
          //
          // two stop bit
          //
          TxData = 1;
          WaitT1Int; // wait for a bit time
          WaitT1Int; // wait for a bit time

          TCE1 = 0;   // Disable tc1 only
          EI ();

}
/*************************************************************************=*****
*         Read a character from the serial port
*         Read a character and store to the input buffer
*         Note:
*         Interrupt acknowledge time is min. 7 clks to 33 clks.
*         This time must be compensated when data is formed.
*************************************************************************=*****/
void Interrupt2 (void)
{
          NOP ();
          CR10 = HBtTime_BaudRate9600 - 8; // 8*4 Cycles Compensation
          TCE1 = 1;   // Enable tc1 only
          WaitT1Int; // wait for a bit time
          // Position at the half of start bit
          //
          // Rec Data
          //
          CR10 = BitTime_BaudRate9600;
          ichar = 0;

          for (ir= 0; ir< 8; ir++)
          {
                    ichar >>= 1;
                    WaitT1Int;
                    if ( RxData == 1) ichar |= 0x80;
          }
          //
          //         Don't read stop bit to speed up for next character reading
          //         Save to buffer
          //
          RxBuffer[bptr++] = ichar;
          if (bptr >= RxBufferSize || ichar == 0X0D)
          {
                    bptr--;
                    RxBuffer[bptr] = 0;
                    bptr = 0;
                    Full = 1;
          }
          TCE1 = 0;   // Disable TC1 only
```

```
//
//        Reset the interrupt flag
          PIF2 = 0;                          // Reset INT2 interrupt flag
//        Echo the char.
          SerialTx (ichar);
}
```

NEC

Interval Timer Operation with 8-Bit Timer 2 7

Applicable Devices

- µPD7801xF
- µPD7801xH
- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A
- µPD7806x
- µPD78030x
- µPD7804xH
- µPD7804xF
- µPD78020x

Description

The purpose of this program is to demonstrate interval timer operation in the K series environment. The 8-bit timer/event counter channels 1 and 2 operate as interval timers by generating interrupt requests repeatedly at intervals preset to 8-bit compare registers CR10 and CR20. When the current counter values of timer registers 1 and 2 match the values set to the compare registers, the timer registers are reset to 0 and interrupt request signals INTTM1 and INTTM2 are generated.

The counting clock of timer TM1 can be selected with bits 0 to 3 (TCL10 to TCL13) of timer clock select register TCL1. The counting clock of timer TM2 can be selected with bits 4 to 7 (TCL14 to TCL17) of TCL1.

The interval timer operation is widely used in such areas as a sampling rate with an ADC operation, an elapsed time setting, various delay functions, and time-of-day setting.

The 8-bit timer 2 includes the following registers:

- Timer register 2 (TM2)
- Output compare register 2 (CR20)
- Timer clock select register TCL1)
- 8-bit timer mode control register 1 (TMC1)

Example Program

This sample program sets two channels' elapse time values in the range from 200 µs to 12.8 seconds (see Figures 7-1 and 7-2). When the current value of the count clock matches the value of compare register 2, an interrupt request is generated. With interval timer operation, the user can set longer elapse times by decrementing the repeat counter.

1. SoftCounter_0 is an 8-bit software counter that decrements every 200 µs until it reaches 0. The counter can be used to set elapse time between 200 µs and 51.2 milliseconds.

2. SoftCounter_1 is a 16-bit software counter that decrements every 200 µs until it reaches 0. The counter can be used to set elapse time between 200 µs and 12.8 seconds.

3. The background module will set an elapse time to 10 and 100 milliseconds, respectively, and toggle P1.0 and P1.1 whenever each time reaches 0.

The modules in the sample program contain three parts:

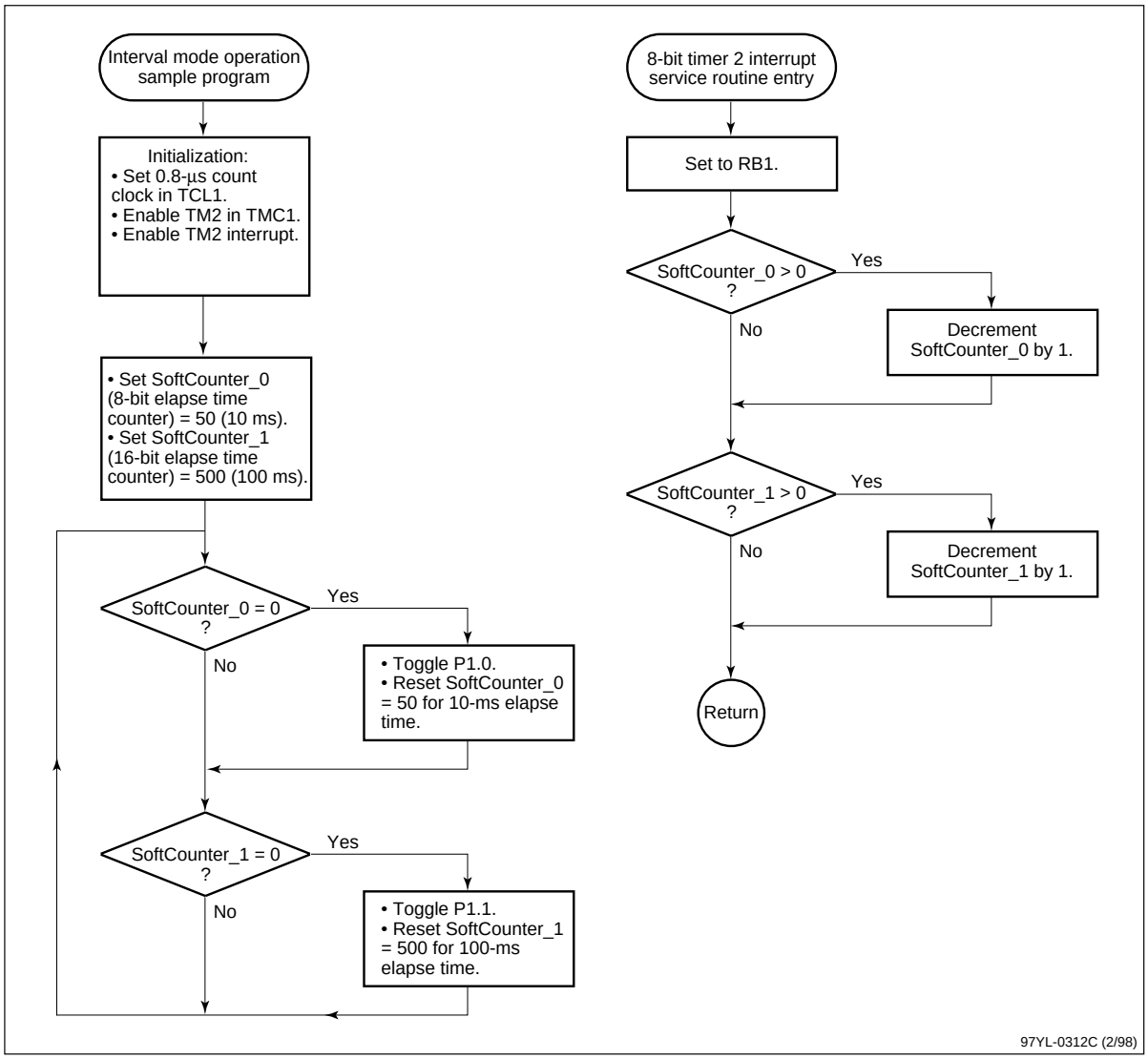
1. Main module
2. 8-bit timer mode 2 initialization
3. 8-bit timer mode 2 interrupt service routine

Figure 7-1. 8-Bit Timer Mode Control Register (TMC1)

7	6	5	4	3	2	1	0
0	0	0	0	0	TMC12	TCE2	TCE1

Bit		
0	TCE1	8-bit timer register 1 operation 0 = Operation stop (TM1 clear to 0)
1	TCE2	8-bit timer register 2 operation 1 = Operation enable
2	TMC12	Operating mode 0 = 8-bit timer register x 2 channel modes (TM1, TM2)

Figure 7-2. Flowchart of Interval Timer Operation



Assembly Language Program: Example of an 8-Bit Interval Timer Implementation

```
;*****************************************************************************
;                                          File name: TIMER2.ASM
;                                          Date: 9/17/97
;*****************************************************************************
;         Description:
;
; This program demonstrates interval timer operations in the K0 and
; K0S family microcontrollers. The 8-bit timer/event counters 1 and 2 operate as
; interval timers. They generate interrupt requests repeatedly at intervals
; preset to 8-bit compare registers CR10 and CR20. When the
; counter values of 8-bit timer registers 1 and 2 match the values set
; to the compare registers, the timer registers are reset to 0 and
; interrupt request signals INTTM1 and INTTM2 are generated.
;
; The counting clock of TM1 can be selected with bits 0 to 3
; of timer clock select register TCL1. The counting clock of TM2 can be selected
; with bits 4 to 7 of TCL1.
;
; The interval timer operation is widely used in the areas of sample rate with ADC
; operation, elapsed time setting, various delay functions, and time-of-day setting.
;
;         Sample program
;         --------------
;
; The sample program sets two delay times from 200 µs to 12.8 seconds.
; When the count clock matches the value of compare register 2, an interrupt
; request is generated. With interval timer operation, the user can set longer elapse
; times by decrementing the repeat counter.
;
;SoftCounter_0 is an 8-bit software counter that decrements
;         every 200 µs until it reaches 0.
;         The counter can be used to set elapse times between 200 µs and
;         51.2 ms.
;
;SoftCounter_1 is a 16-bit software counter that
;         decrements every 200 µs until it reaches 0.
;         The counter can be used to set elapse times between 200 µs and
;         12.8 seconds.
;
;The background module will set an elapse time to 10 and 100 ms and
;toggle P1.0 and P1.1, respectively, whenever each time reaches 0.
;
;=============================================================================
;         for only DDB-K0070A
;
$set (DDB)
$set (K0)
;=============================================================================
;         Equates
;=============================================================================
;=============================================================================
;         saddr area
;         FWA = 0fe20h
;=============================================================================
          dseg      saddr
;
;         Time of day by interval mode
;

SoftCounter_0: ds 1 ; 8-bit elapse counter

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

```
;          Even boundary pair
;=========================================================================
           dseg       saddrp
SoftCounter_1: ds 2 ; 16-bit elapse counter

;=========================================================================
;          Bit segment
;=========================================================================
           bseg

;=========================================================================
;
;          Internal high RAM allocation
;
           dseg       IHRAM
           ds         20h
Stack:

           $ej
;=========================================================================
;          Reset vector
;=========================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           org        26h
           dw         Interval_timer_2 ; 8-bit timer 2 interrupt

           $ej
;=========================================================================
;          Main
;=========================================================================

Main       cseg       at 80h

           public     ResetStart
ResetStart:
;
;          Oscillation mode select register
;           . Does not use divider circuit
;
           $if (K0)
           mov        OSMS,#00000001b
           $endif
;
;          Set stack pointer (IHMEM + 20H)
;
           movw       SP,#Stack               ;Stack ptr
;
;          Init. Watch timer control register
;
           Call       !InitTimer_2

;
;          Enable INTP0 for DDB-K0070A
;
$if (DDB)
           clr1       PMK0
$endif
;
;          Clear time variables
;
;          Set Port 1.0 and Port 1.1 to output direction
;
```

```
          clr1      PM1.0
          clr1      P1.0
          clr1      PM1.1
          clr1      P1.1
;
;         Clear IRQ flags
          clr1      TMIF2
          ei
;
;         Set for 10-ms and 100-ms elapse time
;
          mov       SoftCounter_0,#50       ;10 ms
          movw      SoftCounter_1,#500      ;100 ms
;
main_loop:
          cmp       SoftCounter_0,#0
          bnz       $main15
;         Toggle P1.0
          set1      P1.0
          nop
          clr1      P1.0
          mov       SoftCounter_0,#50
;
;         set for 100-ms elapse time
;
main15:
          movw      ax,SoftCounter_1
          cmpw      ax,#0
          bnz       $main20                 ;Don't decrement if the value is 0
;
          movw      SoftCounter_1,#500

;         Toggle P1.1
          set1      P1.1
          nop
          clr1      P1.1
;
main20:
          br        $main_loop

;=========================================================================
;         Initialization
;
;         . Set 8-bit timer 2 to 0.8-µs count clock.
;         . Set interrupt every 200 µs.
;         . Enable 8-bit timer 2 interrupt request.
;
;=========================================================================
InitTimer_2:
;
;         8-bit timer mode control register (TMC1)
;
;         0 0 0 0 0 0 1 0
;         | | | | | | | |__ timer 1 operation disable
;         | | | | | | |____ timer 2 operation enable
;         | | | | | |_______ 8-bit timer register x 2 channel mode
;         | | | | |_________ 0
;         | | | |___________ 0
;         | | |_____________ 0
;         | |_______________ 0
;         |_________________ 0
;
          mov       TMC1,#00000010b
;
```

```
;         8-bit timer clock selection register (TCL1)
;         Set count clock to 0.8 µs for timer 2
;
          mov       TCL1,#01110000b
;
;         Set interval timer to timer 2 output compare register
;         as 200 interval timer operation
;
          mov       CR20,#250
;
;         Set TM2 interrupt mask register (enable interrupt)
;
          clr1      TMMK2
          ret

;=========================================================================
;         Module name: Interval timer 2 interrupt
;
;         DESCRIPTION:
;                   .Set interval time variables.
;
;         OPERATION:
;                   . Increment tick.
;                   . If tick = 33, then increment seconds.
;                   . If seconds = 60, increment minutes.
;                   . If minutes = 60, increment hours.
;
;=========================================================================

          public    Interval_timer_2
Interval_timer_2:
          sel       RB1
;
;         8-bit soft counter
;
          cmp       SoftCounter_0,#0
          bz        $it10                               ;Don't decrement if the value is 0
          dec       SoftCounter_0
;
;         16-bit soft counter
;
it10:
          movw      ax,SoftCounter_1
          cmpw      ax,#0
          bz        $it20                               ;Don't decrement if the value is 0
; Decrement
          decw      ax
          movw       SoftCounter_1,ax
;
it20:
          reti

;=========================================================================
          end
```

C Language Program: Example of an 8-Bit Interval Timer Implementation

```
/*;*****************************************************************************
;                                          File name: TIMER2.C
;                                          Date: 9/17/97
;*****************************************************************************
;         Description:
; This program demonstrates interval timer operations in the K
; series microcontrollers. The 8-bit timer/event counters 1 and 2 operate as
; interval timers. They generate interrupt requests repeatedly at intervals
; preset to 8-bit compare registers CR10 and CR20. When the
; counter values of 8-bit timer registers 1 and 2 match the values set
; to the compare registers, the timer registers are reset to 0 and
; interrupt request signals INTTM1 and INTTM2 are generated.
;
; The counting clock of TM1 can be selected with bits 0 to 3
; of timer clock select register TCL1. The counting clock of TM2 can be selected
; with bits 4 to 7 of TCL1.
;
; The interval timer operation is widely used in the areas of sample rate with ADC
; operation, elapsed time setting, various delay functions, and time-of-day setting.
;
;         Sample program
;         --------------
;
; The sample program sets two delay times from 200 µs to 12.8 seconds.
; When the count clock matches the value of compare register 2, an interrupt
; request is generated. With interval timer operation, the user can set longer elapse
; times by decrementing the repeat counter.
;
;SoftCounter_0 is an 8-bit software counter that decrements
;         every 200 µs until it reaches 0.
;         The counter can be used to set elapse times between 200 µs and
;         51.2 ms.
;
;SoftCounter_1 is a 16-bit software counter that
;         decrements every 200 µs until it reaches 0.
;         The counter can be used to set elapse times between 200 µs and
;         12.8 seconds.
;
;The background module will set an elapse time to 10 and 100 ms and
;toggle P1.0 and P1.1, respectively, whenever each time reaches 0.
;
;=============================================================================*/
#include "define.h"
#define K0

#pragma interrupt INTTM2 Interval_timer_2 RB1

//;
//;         Time of day by Interval mode
//;

__sreg     unsigned char SoftCounter_0;//; 8-bit elapse counter
__sreg     unsigned int SoftCounter_1;//; 16-bit elapse counter

void InitTimer_2 (void);


/*;=====================================================================
;         Main
;=====================================================================*/
void main (void)
{
```

```
//;
//;         Oscillation mode select register
//;         . Does not use divider circuit
//;
#ifdef K0
          OSMS = 0b00000001;
#endif
//;
//;         Init. Watch timer control register
//;
          InitTimer_2 ();

//;
//;         Enable INTP0 for DDB-K0070A
//;
#ifdef DDB

          PMK0 = 0;

#endif
//;
//;         Clear time variables
//;
//;         Set Port 1.0 and Port 1.1 to output direction
//;
          PM1.0 = 0;
          P1.0 = 0;
          PM1.1 = 0;
          P1.1 = 0;
//;
//;         Clear IRQ flags
//;
          TMIF2 = 0;
          EI ();
//;
//;         Set for 10-ms and 100-ms elapse time
//;
          SoftCounter_0 = 50;// ;10 ms
          SoftCounter_1 = 500;// ;100 ms
//;
          while (TRUE)
          {
                    if ( !SoftCounter_0)
                    {
                    //;
                    //;                      Set for 100-ms elapse time
                    //;
                                        SoftCounter_0 = 50;
                                        //; Toggle P1.0
                                        P1.0 = 1;
                                        NOP ();
                                        P1.0 = 0;
                    }
                    //;
                    //;                      Set for 100-ms elapse time
                    //;
                    if ( !SoftCounter_1)
                    {
                                        SoftCounter_1 = 500;
                                        //; Toggle P1.0
                                        P1.1 = 1;
                                        NOP ();
                                        P1.1 = 0;
                    }
```

```
          }
}

/*;=====================================================================
;         Initialization
;
;         . Set 8-bit timer 2 to 0.8-µs count clock.
;         . Set interrupt every 200 µs.
;         . Enable 8-bit timer 2 interrupt request.
;
;=====================================================================*/
void InitTimer_2 (void)
{
/*
;
;         8-bit timer mode control register (TMC1)
;
;         0 0 0 0 0 0 1 0
;         | | | | | | | |__ timer 1 operation disable
;         | | | | | | |____ timer 2 operation enable
;         | | | | | |______ 8-bit timer register x 2 channel mode
;         | | | | |________ 0
;         | | | |__________ 0
;         | | |____________ 0
;         | |______________ 0
;         |________________ 0
;
*/
          TMC1 = 0b00000010;
//;
//;         8-bit timer clock selection register (TCL1)
//;         Set count clock to 0.8 µs for timer 2
//;
          TCL1 = 0b01110000;
//;
//;         Set interval timer to timer 2 output compare register
//;         as 200 interval timer operation
//;
          CR20 = 250;
//;
//;         Set TM2 interrupt mask register (enable interrupt)
//;
          TMMK2 = 0;
}
/*;=====================================================================
;         Module name: Interval timer 2 interrupt
;
;         DESCRIPTION:
;                   .Set interval time variables.
;
;         OPERATION:
;                   . Increment tick.
;                   . If tick = 33, then increment seconds.
;                   . If seconds = 60, increment minutes.
;                   . If minutes = 60, increment hours.
;
;=====================================================================*/

void Interval_timer_2 (void)
{
//;
//;         8-bit soft counter
//;
          if (SoftCounter_0) SoftCounter_0--;
```

```
//;
//;         16-bit soft counter
//;
          if (SoftCounter_1) SoftCounter_1--;
}
```

NEC

DTMF Tone Generation 8

Applicable Devices

- µPD7808x
- µPD7807x
- µPD780870A

Description

Dual-tone multifrequency (DTMF) signaling is widely used in telephony and telecommunications. With the growing popularity of voice and electronic mail, along with other interactive communication services, the need for DTMF signaling is expanding.

DTMF tones can be generated by pulse width modulation (PWM), eliminating the need for an external DAC. The tones are produced by using a form of delta modulation. When run through a low-pass filter, the result is an accurate DTMF tone.

The actual implementation of a DTMF signal is accomplished by adding two different sinusoids. The following matrix shows the required frequencies for a digit (phone button) selection. The formula “delta” adds these two tones and selects the appropriate cosine value from a table. This cosine value is used to set T1. The total value of T1 + T2 is fixed; therefore (T1 + T2) – T1 = T2. Both values are loaded into the PWM registers and the appropriate PWM timing is output.

Matrix

	Column 1: 1209 Hz	Column 2: 1336 Hz	Column 3: 1477 Hz	Column 4: 1633 Hz
Row 1: 697 Hz	1	2	3	A
Row 2: 770 Hz	4	5	6	B
Row 3: 852 Hz	7	8	9	C
Row 4: 941 Hz	*	0	#	D

Example: If the “1” digit is selected, tones of 1209 Hz and 697 Hz are used.

The “delta” equation shown below calculates the sine wave frequency for a selected digit.

Delta = N*f/fs

Where:

f = DTMF frequency
fs = sampling frequency (typically 8 kHz)
N = 256 table length

Note: A higher sampling rate will generate a more accurate sine wave.

Low-Pass Filter Output with PWM Signal Timing

- t1 + t2 = sample frequency (example 8 kHz = 125 µs)
- The delta value must be adjusted according to the system clock.

Cosine Twiddle Factor Formula for Dual-Tone with 8-Bit Resolution

- Cosine (360/256 * n) * (0x10000/0x100/4) + 0x40
- Cosine (360/256 * n) * 0x40 + 0x40

PWM Output Signal Duty Cycle Computing Formula

T1 = Delta of low-tone frequency + delta of high-tone frequency

PWM Output Operation Registers

- 8-bit timer control registers 5 and 6 (TMCn)
- Output compare registers for 8-bit timers 5 and 6 (CR50 or CR60)
- 8-bit timer clock select registers 5 and 6 (TCLn)

DTMF Tone Generation Demo Program

The DTMF tone program module employs a PWM output function of the 8-bit timer control register (TMC5) in the µPD7808x, µPD7807x, and µPD78070A families (Figure 8-1). The 5-MHz system clock is used. The duty ratio of the PWM signal is determined by the preset values in the compare register (CR50). Pulses are output from pin TO5. The active level of the PWM pulse is selected by setting bit 1 of the TMC register to 1. The program transmits a one-second DTMF tone for each symbol in the 4 x 4 matrix in a loop.

Figure 8-2 is the PWM output timing diagram. Figure 8-3 is the program flowchart.

The PWM output frequency is selected at 9.765 kHz (102.4 µs). The program initializes the first high-frequency and low-frequency delta values from the initial delta table for a DTMF tone when a new digit's DTMF tone is generated.

A new phase value is generated at each interrupt (every 102.4 µs) by adding the delta value to the previous phase value for high and low frequencies. After that, two twiddle factors based on new phase values for high and low frequencies are added to load the value to the output compare register 5 in order to output a new duty cycle.

Figure 8-1. 8-Bit Timer Control Register (TMC5); Settings for PWM Output Operation

7	6	5	4	3	2	1	0
TCE	TMC	0	0	LVS	LVR	TMC	TOE

Bit	Name	Description
0	TOE	TOUT enable = 1
1	TMC	Active level low = 1
2	LVR	x = don't care
3	LVS	x = don't care
6	TMC	PWM mode = 1
7	TCE	TM operation enable = 1

If CRn0 is changed during TMn operation, the value changed is not reflected until TMn overflows. If the CRn0 is set to 00, the Tout line is set to an inactive level. If CRn0 is set to 255, only one cycle out of 256 cycles is set to low.

Figure 8-1. PWM Output Timing (CR50=N)

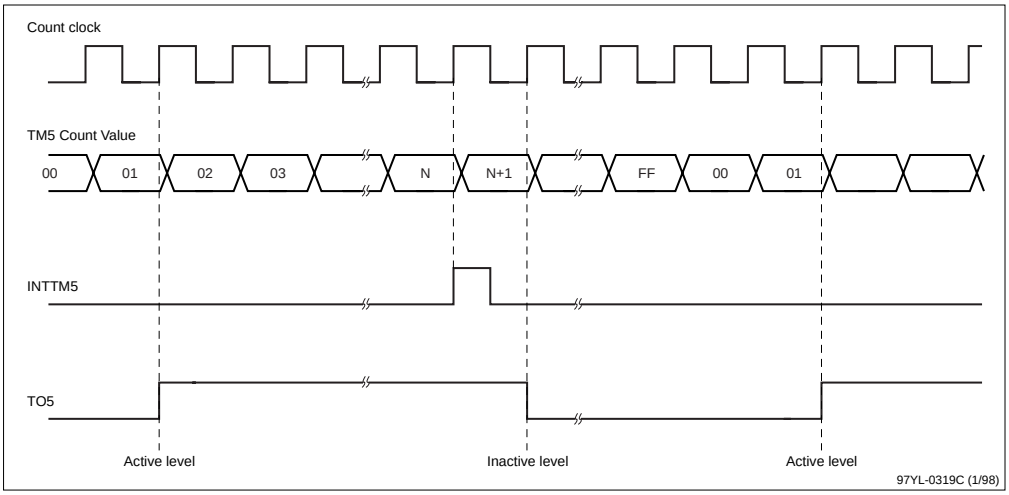
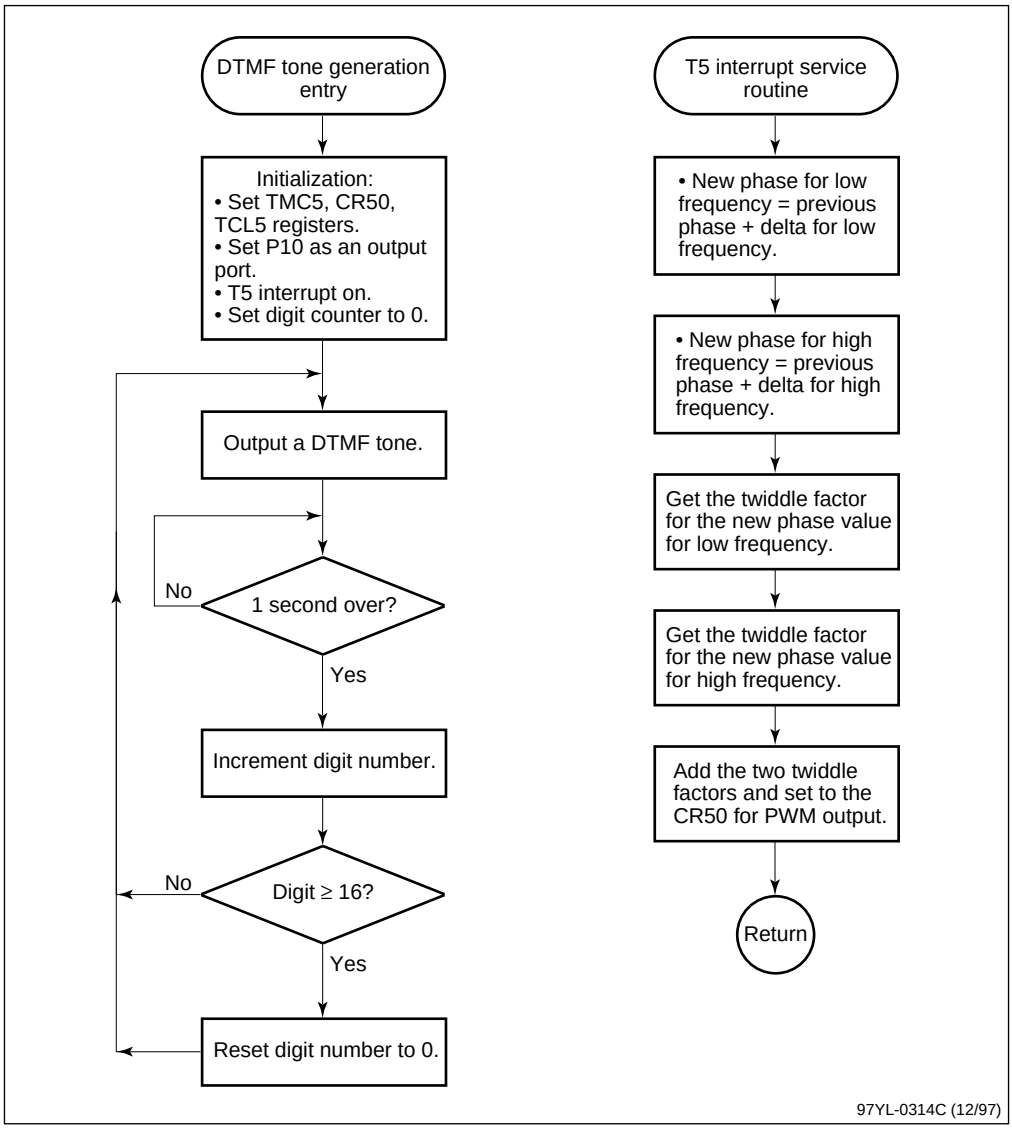


Figure 8-1. Flowchart of a DTMF Tone Generation Demo Program



Assembly Language Program: Example of DTMF Implementation

```
;/*********************************************************************
;                                                          File name: DTMF.ASM
;                                                          Date: 7/26/97
; BHA 7/26/97
;
; Demonstrates DTMF tone generation. Each DTMF is generated in
; sequence from digit 1.
;
; Dual-tone multifrequency signaling is widely used in
; telephone/communication environments. With the popularity of
; voice mail, electronic mail, and other interactive communication services,
; the need for DTMF signaling is expanding. Two sinusoids are added
; together to implement a DTMF signal.
;
;                     Column 1       Column 2       Column 3       Column 4
;                      1209 Hz       1336 Hz        1477 Hz        1633 Hz
; Row 1 697 Hz         1              2              3              A
; Row 2 770 Hz         4              5              6              B
; Row 3 852 Hz         7              8              9              C
; Row 4 941 Hz         *              0              #              D
;
;
; The step-size delta is calculated according to the sine wave frequency
; to be generated (delta = N*f/fs).
; This program uses the following data:
;
;         fs = 8 kHz, the sampling frequency
;         Cosine wave table scaled by 256
;         Sine wave output on PWM output
;         N = 256 table length
;
; This module is designed to produce DTMF tones without an external DAC.
; The tones are produced by a form of delta modulation using a PWM
; output port. When the output is run through a low-pass filter, the result is an
; accurate DTMF tone. Crystal frequency is 8.38 MHz.
;
;         Low-pass filter output with PWM signal timing:

                ;
                ;                       t1              t2

;                   t1 + t2 = sample frequency (ex: 8 kHz = 125 µs)
;                   if t1 = t2, then low-pass filter output = 0
;                   if t1 > t2, then low-pass filter output = positive
;                                       positive peak: t1 = 125 µs and t2 = 0 µs
;                   if t1 < t2, then low-pass filter output = negative
;                                       negative peak: t1 = 0 µs and t2 = 125 µs
;
;         The delta value may need to be adjusted when the actual circuit is designed,
;         depending on the clock.
;
;         Twiddle factor formula:
;                   cosine (360/256 * n) * (0x10000/0x100/4) + 0 x 40
;                   cosine (360/256 * n) * 0 x 40 + 0 x 40
;
;*********************************************************************
;*         Revision history
;*
;*         rev. 1.0 8/14/97
;*         . Digit 0 and '*' delta values are swapped.
;*
```

```
;*          Old:
;*          25,32,     // ;0 941 + 1336
;*          25,35,     // ;* 941 + 1209
;*          New:
;*          25,35,     // ;0 941 + 1336
;*          25,32,     // ;* 941 + 1209
;*********************************************************************/
PWM_DirPort0           equ PM10.0
PWM_Port0              equ P10.0
PWM_DirPort1           equ PM10.1
PWM_Port1              equ P10.1
PWM_Enable             equ TMC5.7 = 1
PWM_Disable            equ TMC5.7 = 0
PWM_Reg0               equ CR50

_1SEC      equ 5000 ; // 1000000 / 200 = 5000

;=====================================================================
;          Bit segment
;=====================================================================
           bseg
_1SecFlag  dbit

;=====================================================================
;          saddr area
;          FWA = 0fe20h
;=====================================================================
               dseg    saddr
Delta1:        ds      1
Delta2:        ds      1
DigitN:        ds      1
DeltaValue:    ds      1
Phase1:        ds      1
Phase2:        ds      1
_i:            ds      1

;=====================================================================
;          Even boundary pair
;=====================================================================
           dseg       saddrp
_200UsecTick:   ds     2

;=====================================================================
;
;          Internal High RAM allocation
;
           dseg       IHRAM
           ds         20h
Stack:

           $ej
;=====================================================================
;          Reset vector
;=====================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           org        2Ah
           dw         DTMF_ToneGen             ;T5 interrupt

           $ej
;=====================================================================
;          Main
;=====================================================================
```

```
Main      cseg      at 80h

          public    ResetStart
ResetStart:

          mov       OSMS,#00000001b         ; // select 5 MHz
          call      !PWM_Init
          clr1      PMK0                    ; // for DDB BOARD - INTP0 has to be set
          EI                                ; // enable interrupt
          mov       DigitN,#0
          clr1      TMMK5                   ; // T5 interrupt on

;
main_loop:
;                   i = DigitN *2;
;                   Delta1 = DeltaTable[i];
;                   Phase1 = Delta1;
;                   Delta2 = DeltaTable[i+1];
;                   Phase2 = Delta2;
;         Low frequency
                    mov                     a,DigitN
                    clr1                    CY
                    rol                     a,1 ;DigitN *= 2;
                    mov                     b,a
                    movw                    hl,#DeltaTable
                    mov                     a,[hl+b]
                    mov                     Phase1,a
;         High frequency
                    inc                     b
                    mov                     a,[hl+b]
                    mov                     Phase2,a
;
;                   // min. output length 50 ms (typically 75 ms)
;                   _1SecFlag = 0;
;                   _200UsecTick = 0;
;                   PWM_Enable;
;                   while (TRUE) if ( _1SecFlag) break;
;                   PWM_Disable;
;
                    clr1                    _1SecFlag
                    movw                    AX,#0
                    movw                    _200UsecTick,ax
                    set1                    TMC5.7
;         Wait for 1 second over
                    bf                      _1SecFlag,$$
                    clr1                    _1SecFlag
                    set1                    TMC5.7
                    inc                     DigitN
                    mov                     a,DigitN
                    cmp                     a,#16
                    bc                      $main_loop
;
                    mov                     DigitN,#0
                    br                      $main_loop

;/*****************************************************************
;*         A tone generation function
;*         Input: digit #
;*;*****************************************************************/
DTMF_ToneGen:
;           Phase1 = Phase1 + Delta1;
;           Phase2 = Phase2 + Delta2;
;           DeltaValue = Cosine [Phase1] + Cosine [Phase2];
```

```
;
;           Output delta value to PWM
;
;           PWM_Reg0 = DeltaValue;
; High frequency
           mov        a,Phase2
           add        a,Delta2
           mov        Phase2,a
; Low frequency
           mov        a,Phase1
           add        a,Delta1
           mov        Phase1,a
; Low frequency
           mov        b,a
           movw       hl,#Cosine
           mov        a,[hl+b]
           mov        c,a
; High frequency
           movw       hl,#Cosine
           mov        a,Phase2
           mov        b,a
           mov        a,[hl+b]
           add        a,c
           mov        PWM_Reg0,a
           ret

;/*********************************************************************
;*          PWM init.
;*********************************************************************/
PWM_Init:
;//         Set PWM output port
;           PWM_DirPort0 = 0;
;           PWM_Port0 = 0;
           clr1       PWM_DirPort0
           clr1       PWM_Port0
;//         Selects CR50 as compare register
;           PWM_Reg0 = 0b00000000;
           mov        PWM_Reg0,#0
;/*====================================================================
;*          Set clock count register
;*
;           TCL53 to TCL50
;
;           0101       = fx/1 (5.0 MHz)
;           0110       = fx/2 (2.5 MHz)
;           0111       = fx/2^2 (1.25 MHz)
;           1000       = fx/2^3 (625 kHz)
;           1001       = fx/2^4 (313 kHz)
;           1010       = fx/2^5 (156 kHz)
;           1011       = fx/2^6 (78.1 kHz)
;           1100       = fx/2^7 (39.1 kHz)
;           1101       = fx/2^8 (19.5 kHz)
;           1110       = fx/2^9 (9.8 kHz)
;           1111       = fx/2^11 (2.4 kHz)
;
;           If clock count = fx/2, then sample rate would be 9,765 Hz == 9.8 kHz.
;           Delta = 256 * fs/fx (9800 Hz).
;====================================================================*/
;           TCL5 = 0b00000110;
           mov        TCL5,#00000110b ;
;/*====================================================================
```

```
;         8-bit timer mode control register
;
; (MSB)   TCE         = 1, TM0 operation enable
;         TMC         = 1, PWM mode
;                     = 0, always
;                     = 0, always
;         LVS         = 0, Don't care
;         LVR         = 0, Don't care
;         TMC         = 1, Active High
; (LSB)   TOE         = 1, TOn output enable
;=====================================================================*/
;          TMC5 = 0b11000011;
          mov       TMC5,#11000011b ;
          ret

;/*******************************************************************
;; Hex codes --> Tone pairs Phone Pad
;;   0001   1 697 + 1209      1
;;   0010   2 697 + 1336      2
;;   0011   3 697 + 1477      3
;;   0100   4 770 + 1209      4
;;   0101   5 770 + 1336      5
;;   0110   6 770 + 1477      6
;    0111   7 852 + 1209      7
;    1000   8 852 + 1336      8
;    1001   9 852 + 1477      9
;    1010   A 941 + 1336      0
;    1011   B 941 + 1209      *
;    1100   C 941 + 1477      #
;    1101   D 697 + 1633      A
;    1110   E 770 + 1633      B
;    1111   F 852 + 1633      C
;    0000   0 941 + 1633      D
;*******************************************************************
;*
;*         Delta values (with 5-MHz system clock)
;*
;*         Delta = Fs * 256 / Fx
;*         where, Fs = Sample frequency
;*                   Fx = Tone frequecy
;*                   (Base counter clock = 0.4 µs)
;*                   (Sample freq = 1/(0.4 * 256) = 9765.625 Hz)
;*
;*******************************************************************/
DeltaTable:

          db 18,32   ; // ;1 697 + 1209
          db 18,35   ; // ;2 697 + 1336
          db 18,39   ; // ;3 697 + 1477

          db 20,32   ; // ;4 770 + 1209
          db 20,35   ; // ;5 770 + 1336
          db 20,39   ; // ;6 770 + 1477

          db 22,32   ; //;7 852 + 1209
          db 22,35   ; //;8 852 + 1336
          db 22,39   ; //;9 852 + 1477;

          db 25,35   ; //;0 941 + 1336
          db 25,32   ; //;* 941 + 1209
          db 25,39   ; //;# 941 + 1477

          db 18,43   ; //;A 697 + 1633
          db 20,43   ; //;B 770 + 1633
```

```
          db 22,43   ; //;C 852 + 1633
          db 25,43   ; //;D 941 + 1633

;//         Twiddle factors

Cosine:
 db 07fh,07fh,07fh,07fh,07fh,07fh,07eh,07eh
 db 07eh,07dh,07dh,07dh,07ch,07ch,07bh,07bh
 db 07ah,07ah,079h,078h,078h,077h,076h,075h
 db 074h,074h,073h,072h,071h,070h,06fh,06eh
 db 06dh,06bh,06ah,069h,068h,067h,066h,064h
 db 063h,062h,060h,05fh,05eh,05ch,05bh,05ah
 db 058h,057h,055h,054h,052h,051h,04fh,04eh
 db 04ch,04bh,049h,048h,046h,045h,043h,042h

 db 040h,03eh,03dh,03bh,03ah,038h,037h,035h
 db 034h,032h,031h,02fh,02eh,02ch,02bh,029h
 db 028h,026h,025h,024h,022h,021h,020h,01eh
 db 01dh,01ch,01ah,019h,018h,017h,016h,015h
 db 013h,012h,011h,010h,00fh,00eh,00dh,00ch
 db 00ch,00bh,00ah,009h,008h,008h,007h,006h
 db 006h,005h,005h,004h,004h,003h,003h,003h
 db 002h,002h,002h,001h,001h,001h,001h,001h

 db 001h,001h,001h,001h,001h,001h,002h,002h
 db 002h,003h,003h,003h,004h,004h,005h,005h
 db 006h,006h,007h,008h,008h,009h,009h,00bh
 db 00ch,00ch,00dh,00eh,00fh,010h,011h,012h
 db 013h,015h,016h,017h,018h,019h,01ah,01ch
 db 01dh,01eh,020h,021h,022h,024h,025h,026h
 db 028h,029h,02bh,02ch,02eh,02fh,031h,032h
 db 034h,035h,037h,038h,03ah,03bh,03dh,03eh

 db 040h,042h,043h,045h,046h,048h,049h,04bh
 db 04ch,04eh,04fh,051h,052h,054h,055h,057h
 db 058h,05ah,05bh,05ch,05eh,05fh,060h,062h
 db 063h,064h,066h,067h,068h,069h,06ah,06bh
 db 06dh,06eh,06fh,070h,071h,072h,073h,074h
 db 074h,075h,076h,077h,078h,078h,079h,07ah
 db 07ah,07bh,07bh,07ch,07ch,07dh,07dh,07dh
 db 07eh,07eh,07eh,07fh,07fh,07fh,07fh,07fh

          end
```

C Language Program: Example of DTMF Implementation

```
#define DEBUG 0

/*************************************************************************
;                                                          File name: DTMF.C
;                                                          Date: 7/26/97
; BHA 7/26/97
;
; Demonstrates DTMF tone generation. Each DTMF is generated in
; sequence from 1.

; Dual-tone multifrequency signaling is widely used in
; telephone/communication environments. With the popularity of
; voice mail, electronic mail, and other interactive communication services,
; the need for DTMF signaling is expanding. Two sinusoids are added
; together to implement a DTMF signal.
;
;                   Column 1       Column 2       Column 3       Column 4
;                    1209 Hz        1336 Hz        1477 Hz        1633 Hz
; Row 1 697 Hz          1              2              3              A
; Row 2 770 Hz          4              5              6              B
; Row 3 852 Hz          7              8              9              C
; Row 4 941 Hz          *              0              #              D
;
; The step-size delta is calculated according to the sine wave frequency
; to be generated (delta = N*f/fs).
; This program uses the following data:
;
;         fs = 8 kHz, the sampling frequency
;         Cosine-wave table scaled by 256
;         Sine-wave output on PWM output
;         N = 256 table length
;
; This module is designed to produce DTMF tones without an external DAC.
; The tones are produced by a form of delta modulation, using a PWM
; output port. When run through a low-pass filter, the result is an
; accurate DTMF tone. Crystal frequency is 8.38 MHz.
;
;         Low-pass filter output with PWM signal timing:

                                  t1              t2


;                   t1 + t2 = sample frequency (ex: 8 kHz = 125 µs)
;                   if t1 == t2, then low-pass filter output = 0
;                   if t1 > t2, then low-pass filter output = positive
;                                       positive peak: t1 = 125 µs and t2 = 0
;                   if t1 < t2, then low-pass filter output = negative
;                                       negative peak: t1 = 0 and t2 = 125 µs
;
;         The delta value may need to be adjusted when the actual circuit is designed,
;         depending on the clock.
;
;         Twiddle factor formula:
;                   cosine (360/256 * n) * (0x10000/0x100/4)+ 0 x 40
;                   cosine (360/256 * n) * 0 x 40 + 0 x 40
;
*************************************************************************
*         Revision history
*
*         rev. 1.0 8/14/97
*         . Digit 0 and '*' delta values are swapped.
```

```
*
*         Old:
*          25,32,     // ;0 941 + 1336
*          25,35,     // ;* 941 + 1209
*         New:
*          25,35,     // ;0 941 + 1336
*          25,32,     // ;* 941 + 1209
*****************************************************************/
#include "define.h"

#pragma interrupt INTTM5 DTMF_ToneGenRB1
#pragma interrupt INTTM1 Timer_8bit_IntervalRB2

/*****************************************************************
; Hex codes --> Tone pairs Phone Pad
; 0001  1 697 + 1209         1
; 0010  2 697 + 1336         2
; 0011  3 697 + 1477         3
; 0100  4 770 + 1209         4
; 0101  5 770 + 1336         5
; 0110  6 770 + 1477         6
; 0111  7 852 + 1209         7
; 1000  8 852 + 1336         8
; 1001  9 852 + 1477         9
; 1010  A 941 + 1336         0
; 1011  B 941 + 1209         *
; 1100  C 941 + 1477         #
; 1101  D 697 + 1633         A
; 1110  E 770 + 1633         B
; 1111  F 852 + 1633         C
; 0000  0 941 + 1633         D
*****************************************************************
*
*         Delta values (with 5-MHz system clock)
*
*         delta =    Fs * 256 / Fx
*                    Where, Fs = Sample frequency
*                    Fx = Tone frequency
*                    (Base counter clock = 0.4 µs)
*                    (Sample freq = 1/(0.4 * 256) = 9765.625 Hz)
*
*****************************************************************/
const unsigned char DeltaTable[] =
{
#ifdef DEBUG
          0,0,
          32,32,
          64,64,
          96,96,
          128,128,
          160,160,
          192,192,
          226,226,
          000,000
#else
          18,32,     // ;1 697 + 1209
          18,35,     // ;2 697 + 1336
          18,39,     // ;3 697 + 1477

          20,32,     // ;4 770 + 1209
          20,35,     // ;5 770 + 1336
          20,39,     // ;6 770 + 1477

          22,32,     // ;7 852 + 1209
```

```
          22,35,     // ;8 852 + 1336
          22,39,     // ;9 852 + 1477

          25,35,     // ;0 941 + 1336
          25,32,     // ;* 941 + 1209
          25,39,     // ;# 941 + 1477

          18,43,     // ;A 697 + 1633
          20,43,     // ;B 770 + 1633
          22,43,     // ;C 852 + 1633
          25,43      // ;D 941 + 1633
#endif
};

//        Twiddle factors

const unsigned char Cosine[] =
{

 0x7f,0x7f,0x7f,0x7f,0x7f,0x7f,0x7e,0x7e,
 0x7e,0x7d,0x7d,0x7d,0x7c,0x7c,0x7b,0x7b,
 0x7a,0x7a,0x79,0x78,0x78,0x77,0x76,0x75,
 0x74,0x74,0x73,0x72,0x71,0x70,0x6f,0x6e,
 0x6d,0x6b,0x6a,0x69,0x68,0x67,0x66,0x64,
 0x63,0x62,0x60,0x5f,0x5e,0x5c,0x5b,0x5a,
 0x58,0x57,0x55,0x54,0x52,0x51,0x4f,0x4e,
 0x4c,0x4b,0x49,0x48,0x46,0x45,0x43,0x42,

 0x40,0x3e,0x3d,0x3b,0x3a,0x38,0x37,0x35,
 0x34,0x32,0x31,0x2f,0x2e,0x2c,0x2b,0x29,
 0x28,0x26,0x25,0x24,0x22,0x21,0x20,0x1e,
 0x1d,0x1c,0x1a,0x19,0x18,0x17,0x16,0x15,
 0x13,0x12,0x11,0x10,0x0f,0x0e,0x0d,0x0c,
 0x0c,0x0b,0x0a,0x09,0x08,0x08,0x07,0x06,
 0x06,0x05,0x05,0x04,0x04,0x03,0x03,0x03,
 0x02,0x02,0x02,0x01,0x01,0x01,0x01,0x01,

 0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x02,
 0x02,0x03,0x03,0x03,0x04,0x04,0x05,0x05,
 0x06,0x06,0x07,0x08,0x08,0x09,0x09,0x0b,
 0x0c,0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,
 0x13,0x15,0x16,0x17,0x18,0x19,0x1a,0x1c,
 0x1d,0x1e,0x20,0x21,0x22,0x24,0x25,0x26,
 0x28,0x29,0x2b,0x2c,0x2e,0x2f,0x31,0x32,
 0x34,0x35,0x37,0x38,0x3a,0x3b,0x3d,0x3e,

 0x40,0x42,0x43,0x45,0x46,0x48,0x49,0x4b,
 0x4c,0x4e,0x4f,0x51,0x52,0x54,0x55,0x57,
 0x58,0x5a,0x5b,0x5c,0x5e,0x5f,0x60,0x62,
 0x63,0x64,0x66,0x67,0x68,0x69,0x6a,0x6b,
 0x6d,0x6e,0x6f,0x70,0x71,0x72,0x73,0x74,
 0x74,0x75,0x76,0x77,0x78,0x78,0x79,0x7a,
 0x7a,0x7b,0x7b,0x7c,0x7c,0x7d,0x7d,0x7d,
 0x7e,0x7e,0x7e,0x7f,0x7f,0x7f,0x7f,0x7f

};

#define PWM_DirPort0          PM10.0
#define PWM_Port0             P10.0
#define PWM_DirPort1          PM10.1
#define PWM_Port1             P10.1
#define PWM_Enable            TMC5.7 = 1
#define PWM_Disable           TMC5.7 = 0
#define PWM_Reg0              CR50
```

```
#define _1SEC  5000           // 1000000 / 200 = 5000

bit                           _1SecFlag;

unsigned char                 Delta1;
unsigned char                 Delta2;
unsigned char                 DigitN;
unsigned char                 DeltaValue;
unsigned char                 Phase1;
unsigned char                 Phase2;
unsigned char                 i;

unsigned int _200UsecTick;


void PWM_Init ();


void main (void)
{

        OSMS = 0b00000001;        // select 5 MHz
        PWM_Init ();
        PMK0 = 0;                 // for DDB BOARD - INTP0 has to be set
        EI ();                    // enable interrupt
        DigitN = 0;
        TMMK5 = 0;                // T5 interrupt on
        TMMK1 = 0;                // T1 interrupt on
        while (TRUE)
        {
                    i = DigitN * 2;
                    Delta1 = DeltaTable[i];
                    Phase1 = Delta1;
                    Delta2 = DeltaTable[i+1];
                    Phase2 = Delta2;

                    // Min. output length 50 ms (typically 75 ms)
                    _1SecFlag = 0;
                    _200UsecTick = 0;
                    PWM_Enable;
                    while (TRUE)
                    {
                                              if ( _1SecFlag) break;
                    }
                    PWM_Disable;
                    DigitN += 1;
                    if (DigitN >= 16 ) DigitN = 0;
          }
}

/*********************************************************************
*          A tone generation function
*          Input: digit #
*;*********************************************************************/
void DTMF_ToneGen (void)
{
          //
          // Output initial value
          //

#ifndef DEBUG

          Phase1 = Phase1 + Delta1;
```

```
          Phase2 = Phase2 + Delta2;
          DeltaValue = Cosine [Phase1] + Cosine [Phase2];
          //
          //        Output delta value to PWM
          //
          PWM_Reg0 = DeltaValue;

#else

          DeltaValue = Cosine [Phase1];
          //
          //        Output delta value to PWM
          //
          PWM_Reg0 = DeltaValue;

#endif
}
/*********************************************************************
*         PWM init.
*********************************************************************/
void PWM_Init ()
{
//        Set PWM output port
          PWM_DirPort0 = 0;
          PWM_Port0 = 0;
//        Selects CR50 as compare register
          PWM_Reg0 = 0b00000000;
/*====================================================================
*         Set clock count register
*
;         TCL53 to TCL50
;
;         0101      = fx/1 (5.0 MHz)
;         0110      = fx/2 (2.5 MHz)
;         0111      = fx/2^2 (1.25 MHz)
;         1000      = fx/2^3 (625 kHz)
;         1001      = fx/2^4 (313 kHz)
;         1010      = fx/2^5 (156 kHz)
;         1011      = fx/2^6 (78.1 kHz)
;         1100      = fx/2^7 (39.1 kHz)
;         1101      = fx/2^8 (19.5 kHz)
;         1110      = fx/2^9 (9.8 kHz)
;         1111      = fx/2^11(2.4 kHz)
;
;         If clock count = fx/2, then sample rate would be 9,765 Hz == 9.8 kHz.
;         Delta = 256 * fs/fx (9800 Hz).
;====================================================================*/
          TCL5 = 0b00000110;
/*====================================================================
;         8-bit timer mode control register
;
; (MSB)   TCE        = 1, TM0 operation enable
;         TMC        = 1, PWM mode
;                    = 0, always
;                    = 0, always
;         LVS        = 0, Don't care
;         LVR        = 0, Don't care
;         TMC        = 1, Active high
; (LSB)   TOE        = 1, TOn output enable
;====================================================================*/
          TMC5 = 0b11000011;
```

```
#ifdef DEBUG
//         Set 8-bit timer interval for 409.6 µs
           TCL1 = 0x07;// SET 800 ns
           TMC1 = 0x03;// enable
           CR10 = 250;// 200 µs

#endif
}


void Timer_8bit_Interval ()
{
           // test point
#ifdef DEBUG
           PM2 = 0; // output direction
           P2.0 = 1;
           P2.0 = 0;
#endif

           _200UsecTick++;
           if (_200UsecTick > _1SEC)
           {
                      _200UsecTick = 0;
#ifndef DEBUG
                      _1SecFlag = 1;
#endif
           }
}
_
```

NEC

I²C Bus Interface 9

Applicable Devices

- μPD7801xF (Y version)
- μPD7801xH (Y version)
- μPD7805x (Y version)
- μPD7805xF (Y version)
- μPD78005x (Y version)
- μPD7807x (Y version)
- μPD78070A (Y version)
- μPD7806x (Y version)
- μPD78030x (Y version)

Description

The I²C™ bus uses a two-wire interface consisting of a serial data line (SDA) and a serial clock line (SCL) to exchange information between devices connected to the bus. Each device on the bus has a unique address and can operate as a transmitter or a receiver depending on its particular function. Devices are further characterized as master or slave. A master is defined as a device that initiates, controls, and terminates a transfer and generates all framing and clock signals. A slave is defined as any device addressed by a master.

Multiple masters are allowed by I²C bus specifications because a bus arbitration and clock synchronization scheme is defined so that messages are not corrupted when more than one device attempts to take master control.

Both the SDA and SCL lines are bidirectional and are pulled up to the positive logic supply rail via resistors. Both lines are high for the bus idle state. In the output mode, the SDA and SCL lines are open-drain or open-collector. Data is exchanged 8 bits at a time and can be transferred at rates up to 100 kbps (standard mode) or 400 kbps (fast mode).

Data transfers on the I²C bus are controlled by two bus states generated by the bus master START and STOP bit conditions. A START condition is defined as a high-to-low transition on the SDA line while the SCL line is high. A STOP condition is defined as a low-to-high transition on the SDA line while the SCL line is high. Refer to Figure 9-1.

Data must always be valid on SDA during SCL high, and that is only allowed to be changed during the SCL low period. Refer to Figure 9-2. Thus, one bit of data is transmitted for each SCL period.

Figure 9-1. START and STOP Conditions

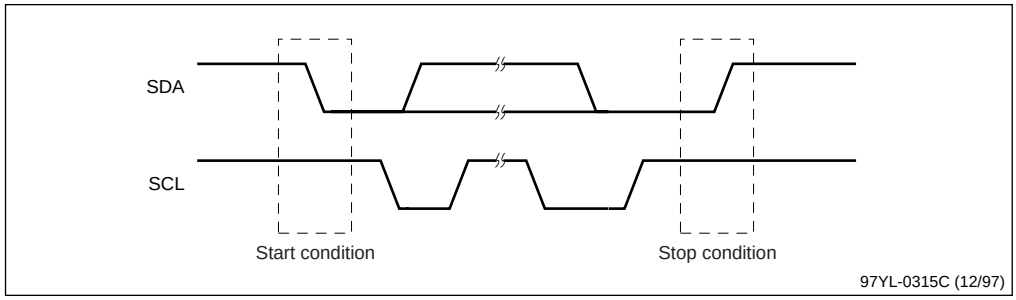
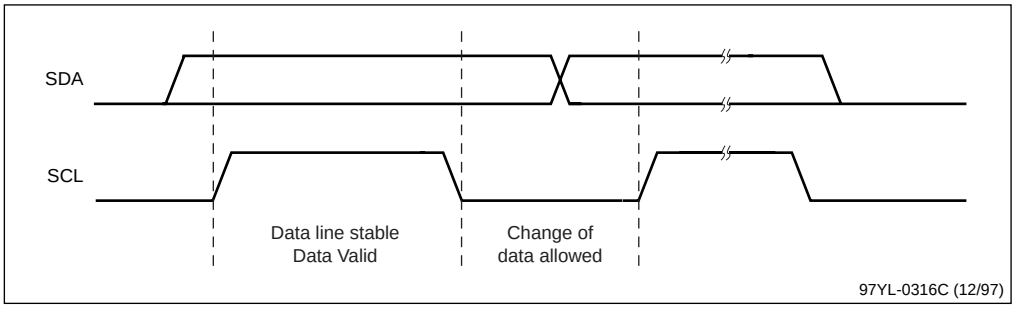
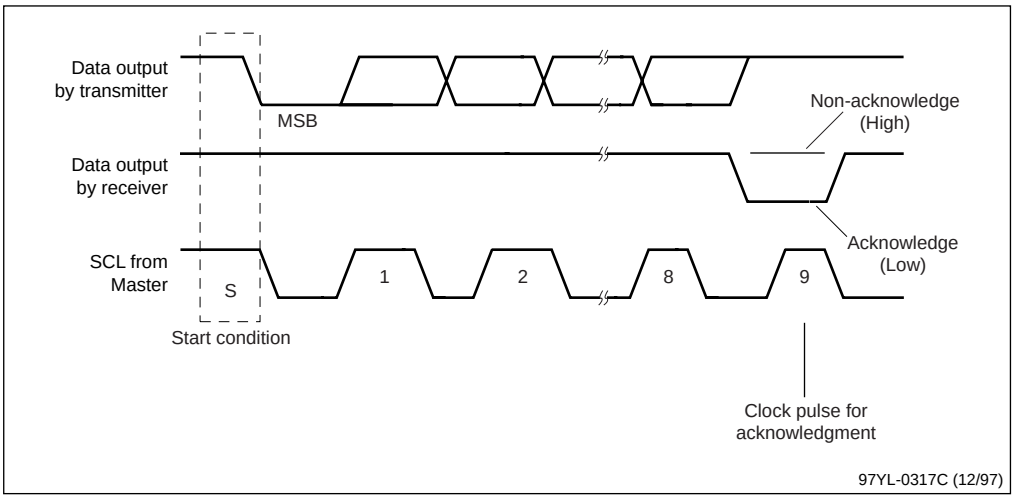


Figure 9-2. Valid Bit Data on the I²C Bus



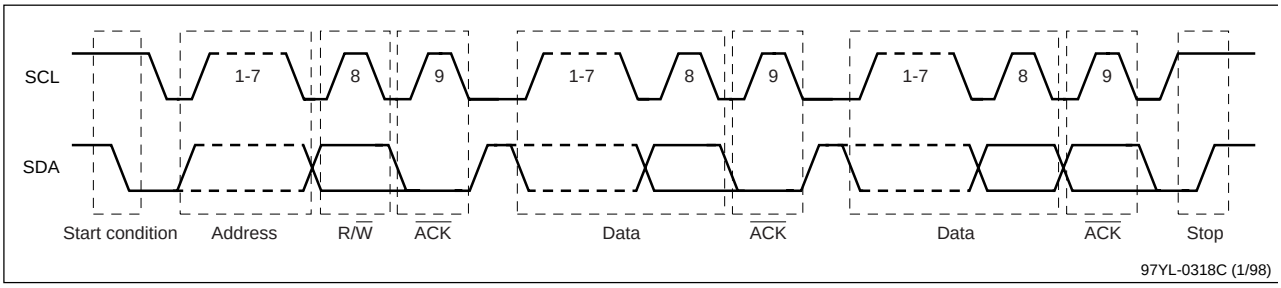
Following the START condition, the first 8-bit byte sent in a bus message is a 7-bit slave address field, along with a data direction or R/W bit. (This discussion is limited to the 7-bit addressing mode.) The data direction bit (LSB) controls whether or not the master will transmit (0 = write) or receive (1 = read) data from the addressed slave. For every byte exchanged, the MSB is always transmitted first. All bus byte transactions are followed by an acknowledge bit (refer to Figure 9-3). The acknowledge bit is a low-level signal placed on the SDA line by the receiving device (master or slave) during the master-transmitted acknowledge clock pulse (ninth high SCL clock pulse).

Figure 9-3. I²C Bus Acknowledge, Non-Acknowledge



If the receiver is unable to receive data (busy slave receiver) or must signal the end-of-data condition (master receiver), a non-acknowledge is sent (SDA high during the ninth high SCL clock pulse time as shown in Figure 9-3). Following the START and slave address transmission, data is exchanged between the master and receiver as required. Upon exchange of the final byte and its acknowledge, the master issues the STOP condition to end bus usage. See Figure 9-4 for a sample I^2C bus data transfer.

Figure 9-4. Sample I²C Bus Message



EEPROM Interface

The preceding general overview of I^2C bus operation provides the foundation necessary to proceed with a serial EEPROM interface in the I^2C 7-bit addressing mode.

Control Registers for I²C Bus Mode Operation

- Serial operating mode register 0 (CSIM0)
- Serial bus interface control register (SBIC)
- Interrupt timing specification register (SINT)

Example Program for the I²C Bus EEPROM Specification

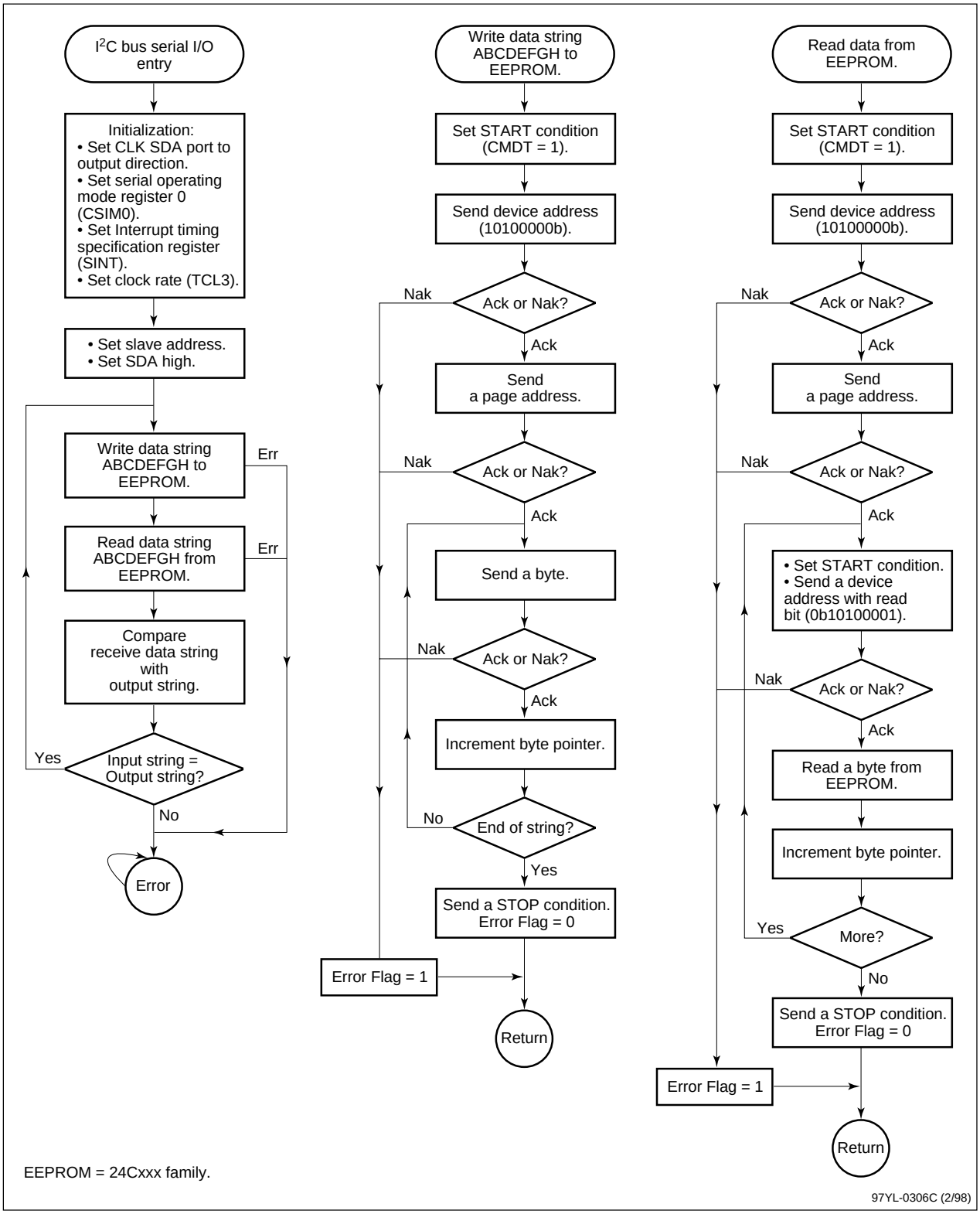
This program provides an overview of the I^2C bus interface and the two-wire 24Cxxx EEPROM family (I^2C slave). See the flowchart in Figure 9-5.

Device	Bytes	Max. Devices	Page Write (Bytes)	Device Address (MSB to LSB)
24C01A	128	8	8	1 0 1 0 a2 a1 a0 R/W

Example Mode Operation for Output

- Write "ABCDEFGH" to the EEPROM.
- Read the same data back.
- Compare for a match.

Figure 9-5. Flowchart for I²C Interface



Assembly Language Program: Example of I²C Bus Interface and 24Cxxx EEPROM

```
;*************************************************************************
;                                          File name: I2CBUS.ASM
;                                          Date: 9/20/97
;*************************************************************************
;         Description:
; The I2C bus uses a two-wire interface consisting of a serial data line (SDA)
; and a serial clock line (SCL) to exchange information between devices connected
; to the bus. Each device on the bus has a unique address and can
; operate as a transmitter or a receiver depending on its particular function.
; Devices are further characterized as master or slave.
; A master is defined as a device that initiates, controls, generates all framing and clock
; signals, and terminates a transfer. A slave is defined as any
; device addressed by a master.
;
; Multiple masters are allowed by I2C bus specifications because a bus arbitration
; and clock synchronization scheme is defined so that messages are not corrupted when
; more than one device attempts to take master control.
;
; Both the SDA and SCL lines are bidirectional and are pulled up to the positive
; logic supply rail.
;
; Data transfers on the I2C bus are controlled by two bus states generated by
; the bus master START and STOP bit conditions. A START condition is defined
; as a high-to-low level transition on the SDA line while the SCL line is high. A STOP condition
; is defined as a low-to transition on the SDA line while the SCL line is high.
; Data must always be valid on SDA during SCL high, and that is only allowed
; to be changed during the SCL low period. Thus, one bit of data is transmitted for
; each SCL period.
;
; Following the START condition, the first 8-bit byte sent in a bus message is
; a 7-bit slave address field ,along with a data direction or R/W bit. (This
; discussion is limited to the 7-bit addressing mode.) The data direction bit (LSB)
; controls whether or not the master will transmit (0 = write) or receive (1 = read) data
; from the addressed slave. For every 8-bit byte exchanged, the MSB is always
; transmitted first. All 8-bit byte bus transactions are followed by an acknowledge
; bit. The acknowledge bit is a low-level signal placed on the SDA line by the
; receiving device (master or slave) during the master-transmitted acknowledgment
; pulse (ninth high SCL clock pulse).
;
; If the receiver is unable to receive data or must signal the end-of-data condition
; (master receiver), a non-acknowledge is sent. Upon exchange of the final byte
; and its acknowlege, the master issues the STOP condition to end bus usage.
;=======================================================================
;         I2C bus EEPROM specification.
;
;         This example program provides an overview of the I2C bus interface
;         and the two-wire 24Cxxx EEPROM family (I2C slave).
;
;                        # of       Max. # of      Page Write        Device Address
;         Device        Bytes        devices      (# of bytes)        (MSB to LSB)
;;        24C01A         128            8              8         1 0 1 0 a2 a1 a0 R/W
;
;=======================================================================
;         Example mode operation for output
;
;         Write "ABCDEFGH' to the EEPROM.
;         Read the same data back.
;         Compare for a match.
;
;=======================================================================
;         for only DDB-K0070A
```

```
;
$set (K0)
$set (DDB)

;=====================================================================
;
DebugFlag set 1
;
CR                equ         13              ;Carriage return
LF                equ         10              ;Line feed
;=====================================================================
;          Equates
;=====================================================================
SDAPort           equ         P2.5            ;Latch output high for SDA0
SDADirBit         equ         PM2.5           ;Serial data direction register
SCLKPort          equ         P2.7            ;Latch output high for SCL
SCLKDirBit        equ         PM2.7           ;Serial clock direction register
;
CLCx        EQU        SINT.3
;
;          Slave address for 24C01A (8 bytes/page)

SlaveAddr         equ         10100000b       ;a2/a1/a0 = 0
;
;                 24C01A page address
;
PageAddr          equ         00000000b       ;page address in the EEPROM

;=====================================================================
;          Wait for IRQ and check acknowledge signal
;=====================================================================
WaitIntAndAck     macro       j1
                  bf          CSIIF0,$$
                  clr1        CSIIF0
                  bf          ACKD,$j1 ;No Ack
                  endm

;=====================================================================
;          Wait for START condition and set clock low
;=====================================================================
SetStartCondition macro
;                 Set START Condition
                  set1        CMDT            ; Start ON
                  bf          CMDD,$$         ; Wait for START condition detected
                  endm

;=====================================================================
;          Wait for STOP condition and set clock low
;=====================================================================
SetStopCondition macro
                  bf          CLD,$$          ;busy
;                 Set STOP condition
                  set1        RELT
;                 Set clock line to high impedence
                  set1        CLCx
                  bf          CLD,$$          ;Low (busy)?
                  endm
;=====================================================================
;          Send device address
;=====================================================================
SendDeviceAddress macro
;                 Set START Condition
                  mov         a,SVA0
                  mov         SIO0,a          ;Send a device address
```

```
                endm
;=============================================================================
;               saddr area
;               FWA = 0fe20h
;=============================================================================
                dseg        saddr

SVA0:           DS          1

;=============================================================================
;               Even boundary pair
;=============================================================================
                dseg        saddrp

;=============================================================================
;               Bit segment
;=============================================================================
                bseg
;=============================================================================
;               Internal High RAM allocation
;
                dseg        IHRAM
                ds          20h
Stack:
InputBuffer:    ds          10              ;10 bytes

                $ej
;=============================================================================
;               Reset Vector
;=============================================================================
Vector          cseg        at 0000h
                dw          ResetStart

;               org         20h
;               dw          I2CInter ;I2C bus interrupt

                $ej
;=============================================================================
;               Main
;=============================================================================
Main            cseg        AT              80h

ResetStart:
;
;               Oscillation mode select register
;                . Does not use divider circuit
;
                $if (K0)
                mov         OSMS,#00000001b
                $endif
;
;               Set stack pointer (IHMEM + 20H)
;
                movw        SP,#Stack       ;Stack ptr
;
;               Init. I2C bus mode/control register
;
                call        !InitI2C
;
;               Set receive buffer ptr
;
                sel         RB1
                mov         b,#0                        ;offset to 0
```

```
                sel         RB0
;
;               Enable INTP0 for DDB-K0070A
;
$if (DDB)
                clr1        PMK0
$endif
main_loop:
;
;               Write to I2C bus I/F
;
                call        !OutputToI2C
                bnc         $Error
;
;               Read data from I2C bus I/F
;
                call        !InputFromI2C
                bnc         $Error
;
;               Check data valid
;
                movw        hl,#TestString
                movw        de,#InputBuffer
                mov         c,#TestStringSize
cmpr:
                mov         a,[de]
                cmp         a,[hl]
                bz          $matched
;               Error - Not Matched
                nop
                nop
matched:
                incw        hl
                incw        de
                dbnz        c,$cmpr
;
                br          main_loop
;
;               Error condition detected
;
Error:
                nop
                br          main_loop

;=====================================================================
;               Module name: OutputToI2C
;
;               DESCRIPTION:
;                           This module is to write a page data (8 bytes) to the
;                            24Cxxx (EEPROM) through I2C bus I/F.
;
;               OPERATION:
;
;                           .Set IRQ on 9th clock (falling edge).
;                           .Set START condition.
;                           .Write a device address and wait for the IRQ.
;                           .Check ACK signal.
;                           .Write EEPROM address to be written.
;                           .Load a byte from the test string.
;                           .Write data and check ACK signal.
;                           .After last byte is transferred, set STOP condition.
;
;=====================================================================
```

```
;                Input:
;                           None
;                Return:
;                           CY = 1 for normal return
;                           CY = 0 for Error return
;=====================================================================
OutputToI2C:
;                Check clock level
                 bf         CLD,$$          ;Low - busy

                 SetStartCondition

                 SendDeviceAddress
                 WaitIntAndAckTxError

;                Send page address in the EEPROM

                 mov        SIO0,#PageAddr
                 WaitIntAndAckTxError
;
;                Output data
;
                 movw       hl,#TestString
                 mov        c,#TestStringSize
Outloop:
                 mov        a,[hl]
                 mov        SIO0,a          ;Send data
                 WaitIntAndAckTxError
                 incw       hl
                 dbnz       c,$Outloop
;
                 SetStopCondition
;
                 set1       CY
                 ret                        ;Return with CY = 1
;
;                Error
;
TxError:
                 SetStopCondition
;
                 clr1       CY              ;Return with CY = 0
                 ret

TestString:
                 db         'ABCDEFGH',CR,LF
TestStringEnd:
TestStringSize   equ        TestStringEnd - TestString

;=====================================================================
;                Module name: InputFromI2C
;
;                DESCRIPTION:
;                           This module is to read a page data (8 bytes) from the
;                            24Cxxx (EEPROM) through I2C bus I/F.
;
;                OPERATION:
;
;                           .Set IRQ on 9th clock.
;                           .Write slave device address to be written.
;                           .Write EEPROM address to be written.
;                           .Send ACK signal if OK.
;
```

```
;                           .Set IRQ on 8th clock.
;                           .Send START condition..
;                           .Read data and check ACK signal
;                           .After last byte is read, set STOP condition.
;
;=====================================================================
;                 Input:
;                           None
;                 Return:
;                           CY = 1 for normal return
;                           CY = 0 for error return
;=====================================================================
InputFromI2C:
                  SetStartCondition

                  SendDeviceAddress
                  WaitIntAndAckTxError
;
;                 Send page address in the EEPROM
;
                  mov         SIO0,#PageAddr
                  WaitIntAndAckTxError
;
                  SetStartCondition

;                 Send device address with read signal

                  mov         a,SVA0
                  add         a,#1                        ;Set read signal
                  mov         SIO0,a                      ;Send a device address
;                 Wait for IRQ and Check ACK
                  WaitIntAndAckTxError
;
;                 Check receive status
;                 When LSB of device address = 1
;                 Read data
;
                  movw        hl,#InputBuffer
                  mov         c,#TestStringSize
Inloop:
;                 Cancel wait and start reception
                  mov         SIO0,#0ffh                  ;Start receiving data
                  WaitIntAndAckRxError
;
;                 Read data
;
                  mov         a,SIO0                      ;Read data
                  mov         [hl],a
                  incw        hl
                  dbnz        c,$Inloop

                  SetStopCondition
;
                  set1        CY
                  ret                                     ;Return with CY = 1
;
;                 Error
;
RxError:
;
                  SetStopCondition
;
                  clr1        CY                          ;Return with CY = 0
```

```
                ret

;=====================================================================
;               Module name: InitI2C
;
;               DESCRIPTION:
;                           This module is to initialize the I2C bus I/F control and mode
;                           registers.
;
;               OPERATION:
;                           .Set SCL to output direction and SDA to output direction.
;                           .Set transfer clock.
;                           .Enable operation.
;                           .Enable IRQ on STOP condition.
;                           .Set slave device address.
;                           .Enable IRQ on 9th clock falling edge.
;
;=====================================================================
;               Input:      none
;               Output: none
;
;=====================================================================
InitI2C:
o
;               Set SCL to output direction
;
                set1        SDAPort         ;Latch output high for SDA
                set1        SCLKPort        ;Latch output high for SCLK
;
                clr1        SCLKDirBit      ;Serial cloock set to output direction
                clr1        SDADirBit       ;Serial data to output direction

;*************************************************************
; Serial operating mode register 0 (CSIM0)
;*                                                           *
;*         1 0 0 1 1 1 1 1                                   *
;*         ^ ^ ^ ^ ^ ^ ^ ^                                   *
;*         | | | | | | |_|__ 11 - Clock specified with bits 0 to 3 of TCL3*
;*         | | | | | | |                                     *
;*         | | | |_|_|_____ 111 I2C bus mode (P25 = SDA0)  *
;*         | | |___________ 0 IRQ with each serial transfer *
;*         | |_____________ 0 Read only - slave addr. not equal to serial I/O shift register data*
;*         |_____________ 1 Enable operation               *
;*************************************************************/
          mov       CSIM0,#10011111b
;*************************************************************
;*        Interrupt timing specification register (SINT) *
;*                                                           *
;*         0 0 0 0 1 0 1 1                                   *
;*         ^ ^ ^ ^ ^ ^ ^ ^                                   *
;*         | | | | | | |_|__ 11 9-clock wait                 *
;*         | | | | | |_____ 0 Indicates that the wait state has *
;*         | | | | |  released                               *
;*         | | | | |________ 1 Other than serial transfer SCL *
;*         | | | |    pin, output is set to high impedance *
;*         | | | |__________ 0 Bits 0 to 7                   *
;*         | | |                                           *
;*         | | |___________ 0 CSIIF0 is set to 1 after end of *
;*         | |                                serial transfer *
;*         | |_____________ 0 SCL line level low (read only) *
;*         |                                               *
;*         |_____________ 0                                *
;*                                                           *
```

```
;*****************************************************************/
                    mov         SINT,#00001011b
;
;                   Clock selection = 1000b (39.1 kHz)
;
                    mov         TCL3,#11001000b
;
;                   Serial bus interface control register (SBIC)
;
                    set1        RELT;SO Latch is set to 1 after setting automatically clear to 0
;
;                   Slave address for 24C1A
;                   1010 A2 A1 A0 R/W = 10100000b
;
                    mov         SVA0,#SlaveAddr
                    ret

;*****************************************************************
          end
_
```

C Language Program: Example of I²C Bus Interface and 24Cxxx EEPROM

```
/*;*************************************************************************
;                                          File name: I2CBUS
;                                          Date: 9/20/97
;*************************************************************************
;         Description:
; The I2C bus uses a two-wire interface consisting of a serial data line (SDA)
; and a serial clock line (SCL) to exchange information between devices connected
; to the bus. Each device on the bus has a unique address and can
; operate as a transmitter or a receiver depending on its particular function.
; Devices are further characterized as master or slave.
; A master is defined as a device that initiates, controls, and terminates a transfer and generates
; all framing and clock signals. A slave is defined as any device addressed by a master.
;
; Multiple masters are allowed by I2C bus specifications because a bus arbitration
; and clock synchronization scheme is defined so that messages are not corrupted when
; more than one device attempts to take master control.
;
; Both the SDA and SCL lines are bidirectional and are pulled up to the positive
; logic supply rail.
;
; Data transfers on the I2C bus are controlled by two bus states generated by
; the bus master START and STOP bit conditions. A START condition is defined
; as a high-to-low transition on the SDA line while the SCL line is high. A STOP condition
; is defined as a low-to-high transition on the SDA line while the SCL line is high.
; Data must always be valid on SDA during SCL high, and that is only allowed
; to be changed during the SCL low period. Thus, one bit of data is transmitted for
; each SCL period.
;
; Following the START condition, the first 8-bit byte sent in a bus message is
; a 7-bit slave address field, along with a data direction or R/W bit. (This
; discussion is limited to the 7-bit addressing mode.) The data direction bit (LSB)
; controls whether or not the master will transmit (0 = write) or receive (1 = read) data
; from the addressed slave. For every 8-bit byte exchanged, the MSB is always
; transmitted first. All 8-bit byte bus transactions are followed by an acknowledge
; bit. The acknowledge bit is a low-level signal placed on the SDA line by the
; receiving device (master or slave) during the master-transmitted acknowledgment
; pulse (ninth high SCL clock pulse).
;
; If the receiver is unable to receive data or must signal the end-of-data condition
; (master receiver), a non-acknowledge is sent. Upon exchange of the final byte
; and its acknowlege, the master issues the STOP condition to end bus usage.
;=========================================================================
;         I2C bus EEPROM specification.
;
;         This example program provides an overview of the I2C bus interface
;         and the two-wire 24Cxxx EEPROM family (I2C slave).
;
;                        # of       Max. # of      Page Write        Device Address
;         Device       Bytes        devices      (# of bytes)        (MSB to LSB)
;;        24C01A        128            8              8         1 0 1 0 a2 a1 a0 R/W
;
;=========================================================================
;         Example mode operation for output
;
;         Write "ABCDEFGH' to the EEPROM.
;         Read the same data back.
;         Compare for a match.
;
;=========================================================================*/
```

```
#include "define.h"

/*;=========================================================================
;          Equates
;=========================================================================*/
#define SDAPort               P2.5           // Latch output high for SDA0
#define SDADirBit             PM2.5          // Serial data direction register
#define SCLKPort              P2.7           // Latch output high for SCL
#define SCLKDirBit            PM2.7          // Serial clock direction register
//
#define CLCx                  SINT.3
//;
//;                            Slave address for 24C01A (8 bytes/page)
//;
#define SlaveAddr             0b10100000     ;//a2/a1/a0 = 0
//;
//;         24C01A page address
//;
#define PageAddr              0b00000000     ;//page address in the EEPROM

/*;=========================================================================
;          Macro definitions
;=========================================================================*/

#define TxWaitIntAndAck while (!CSIIF0); CSIIF0 = 0; if (!ACKD) goto TxError;
#define RxWaitIntAndAck while (!CSIIF0); CSIIF0 = 0; if (!ACKD) goto RxError;
#define SetStartCondition CMDT = 1; while (!CMDD);
#define SetStopCondition while (!CLD); RELT = 1; CLCx = 1; while (!CLD);
#define SendDeviceAddress SIO0 = SVA0;


__sreg unsigned char SVA0;
#define InputBufferSize 10
__sreg unsigned char InputBuffer[InputBufferSize];
__sreg unsigned char i;
__sreg unsigned char c;

bit ErrorFlag;


unsigned char const TestString[] = "ABCDEFGH";

void InputFromI2C ();
void InitI2C ();
void OutputToI2C ();

/*;=========================================================================
;          Main
;=========================================================================*/
void main ()
{
//;
//;         Oscillation mode select register
//;         . Does not use divider circuit
//;
#ifdef K0
          OSMS=0b00000001;
#endif
//;
//;         Init. I2C bus mode/control register
//;
          InitI2C ();
//;
```

```
//;         Set receive buffer ptr
//;
          while (TRUE)
          {
                    ErrorFlag = 0;
                    //;
                    //;                     Write to I2C bus I/F
                    //;
                    OutputToI2C ();
                    if (ErrorFlag) go to ErrorStop;
                    //;
                    //;                     Read data from I2C bus I/F
                    //;
                    InputFromI2C ();
                    if (ErrorFlag) goto ErrorStop;
                    //;
                    //;                     Check data valid
                    //;
                    for (i =0 ;i < InputBufferSize; i++)
                    {
                                        if (TestString[i] != InputBuffer[i]) goto ErrorStop;
                    }
          }

ErrorStop:
          while (TRUE);
}
/*;=====================================================================
;         Module name: OutputToI2C
;
;         DESCRIPTION:
;                   This module is to write a page data (8 bytes) to the
;                    24Cxxx (EEPROM) through I2C bus I/F.
;
;         OPERATION:
;
;                   .Set IRQ on 9th clock (falling edge).
;                   .Set START condition.
;                   .Write a device address and wait for the IRQ.
;                   .Check ACK signal.
;                   .Write EEPROM address to be written.
;                   .Load a byte from the test string.
;                   .Write data and check ACK signal.
;                   .After last byte is transferred, set STOP condition.
;
;=====================================================================
;         Input:
;                   None
;         Return:
;                   CY = 1 for normal return
;                   CY = 0 for error return
;=====================================================================*/
void OutputToI2C ()
{
//;         Check clock level
          while (!CLD);//                   ;Low - busy

          SetStartCondition;

          SendDeviceAddress;
          TxWaitIntAndAck;
```

```
//;         Send page sddress in the EEPROM

          SIO0 = PageAddr;
          TxWaitIntAndAck;
//;
//;         Output Data
//;
          for (i= 0 ;sizeof TestString;i++)
          {
                    SIO0 = TestString[i];
                    TxWaitIntAndAck;
          }
//
          SetStopCondition;
//
          ErrorFlag = 0;
          return;
//;
//;         Error
//;
TxError:
          SetStopCondition
//;
          ErrorFlag = 1;
}
/*;=====================================================================
;         Module name: InputFromI2C
;
;         DESCRIPTION:
;                   This module is to read a page data (8 bytes) from the
;                    24Cxxx (EEPROM) through I2C bus I/F.
;
;         OPERATION:
;
;                   .Set IRQ on 9th clock.
;                   .Write slave device address to be written.
;                   .Write EEPROM address to be written.
;                   .Send ACK signal if OK.
;                   .Set IRQ on 8th clock.
;                   .Send START condition.
;                   .Read data and check ACK signal.
;                   .After last byte is read, set STOP condition.
;
;=====================================================================
;         Input:
;                   None
;         Return:
;                   CY = 1 for normal return
;                   CY = 0 for error return
;=====================================================================*/
void InputFromI2C ()
{
          SetStartCondition;

          SendDeviceAddress;
          RxWaitIntAndAck;
//;
//;         Send page address in the EEPROM
//;
          SIO0 = PageAddr;
          RxWaitIntAndAck;
//;
          SetStartCondition;
//
```

```
//;         Send device address with read signal
//
          SIO0 = SVA0+1;
          RxWaitIntAndAck;
//
//        Read data string
//
          for (i=0;i<sizeof InputBuffer;i++)
          {
                    SIO0 = 0xff;
                    RxWaitIntAndAck;
                    InputBuffer[i] = SIO0;
          }
//
          SetStopCondition;
          ErrorFlag = 0;
          return;
//
//        Error
//
RxError:
          ErrorFlag = 1;
}
/*;=====================================================================
;         Module name: InitI2C
;
;         DESCRIPTION:
;                   .This module is to initialize the I2C bus I/F control and mode
;                   registers.
;
;         OPERATION:
;                   .Set SCL to output direction and SDA to output direction.
;                   .Set transfer clock.
;                   .Enable operation.
;                   .Enable IRQ on STOP condition.
;                   .Set slave device address
;                   .Enable IRQ on 9th clock falling edge.
;=====================================================================
;         Input:    none
;         Output: none
;=====================================================================*/
void InitI2C ()
{
//;
//;         Set SCL to output direction
//;
          SDAPort = 1;// ;Latch output high for SDA
          SCLKPort = 1;// ;Latch output high for SCLK

          SCLKDirBit = 0; // ;Serial clock set to output direction
          SDADirBit = 0;// ;Serial data to output direction
/*;************************************************************
; Serial operating mode register 0 (CSIM0)
;*                                                           *
;*         1 0 0 1 1 1 1 1                                   *
;*         ^ ^ ^ ^ ^ ^ ^ ^                                   *
;*         | | | | | | |_|__ 11 - Clock specified with bits 0 to 3 of TCL3*
;*         | | | | | | |                                     *
;*         | | | |_|_|_____ 111 I2C bus mode (P25 = SDA0)  *
;*         | | |____________ 0 IRQ with each serial transfer *
;*         | |_____________ 0 Read only - slave addr. not equal to serial I/O shift register data.
;*         |______________ 1 Enable operation              *
;************************************************************/
          CSIM0 = 0b10011111;
```

```
/*;************************************************************
;*         Interrupt timing specification register (SINT) *
;*                                                        *
;*          0 0 0 0 1 0 1 1                               *
;*          ^ ^ ^ ^ ^ ^ ^ ^                               *
;*          | | | | | | |_|__ 11 9-clock wait              *
;*          | | | | | |_____ 0 Indicates that the wait state has *
;*          | | | | |  released                           *
;*          | | | | |________ 1 Other than serial transfer SCL *
;*          | | | |    pin, output is set to high impedance *
;*          | | | |__________ 0 Bits 0 to 7               *
;*          | | |                                       *
;*          | | |_____________ 0 CSIIF0 is set to 1 after end of *
;*          | |                                 serial transfer. *
;*          | |_______________ 0 SCL line level low (read only) *
;*          |                                             *
;*          |_________________ 0                          *
;*                                                        *
;************************************************************/
          SINT = 0b00001011;
//;
//;         Clock selection = 1000b (39.1 kHz)
//;
          TCL3 = 0b1001000;
//;
//;         Serial bus interface control register (SBIC)
//;         (SBIC)
//;
          RELT = 1;  //;SO Latch is set to 1 after setting automatically clear to 0
//;
//;         Slave address for 24C01A
//;         1010 A2 A1 A0 R/W = 10100000b
//;
          SVA0 = SlaveAddr;
}
```

NEC

LCD Controller/Driver 10

Applicable Devices

- µPD78030x
- µPD7806x

Description

Liquid crystal displays (LCDs) are widely used in handheld embedded applications as reliable, low-power, compact readouts. Many of these applications require drivers/controllers on-board.

The example program is an LCD driver using 4-time-division and 1/2 bias. The program displays the time of day: HOURS:MINUTES:SECONDS. Default time is 12:00:00. Every 15.6 milliseconds, the program checks whether a second has passed and, if so, updates the display.

LCD Controller/Driver Features

1. Automatic output of segment signals and common signals is possible by automatic reading of the display data memory.

2. Any of five display modes can be selected:
   - Static
   - 3/4 duty (1/2 bias)
   - 1/3 duty (1/2 bias)
   - 1/3 duty (1/3 bias)
   - 1/4 duty (1/3 bias)

3. Any of four frame frequencies can be selected in each display mode.

4. There are a maximum of 40-segment signal outputs (S0 to S39 = 160 bits) and four common signal outputs (COM0 to COM3).

5. Sixteen of the segment signal outputs can be switched to input/output ports in units of two (P80/S39 to P87/S32 and P90/S31 to P97/S24).

6. In mask ROM versions, split resistors for LCD driver voltage generation can be incorporated by mask option.

7. Operation on the subsystem clock is also possible.

Maximum Number of Displayable Pixels in Each Display Mode

Bias Method	Time Division	COM Signals	Max. Pixels
–	Static	COM0	40 (40 segments x 1 common)
1/2	2	COM0 – COM1	80 (40 segments x 2 commons)
	3	COM0 – COM2	120 (40 segments x 3 commons)
1/3	3		
	4	COM0 – COM3	160 (40 segments x 4 commons)

LCD Controller/Driver Configuration

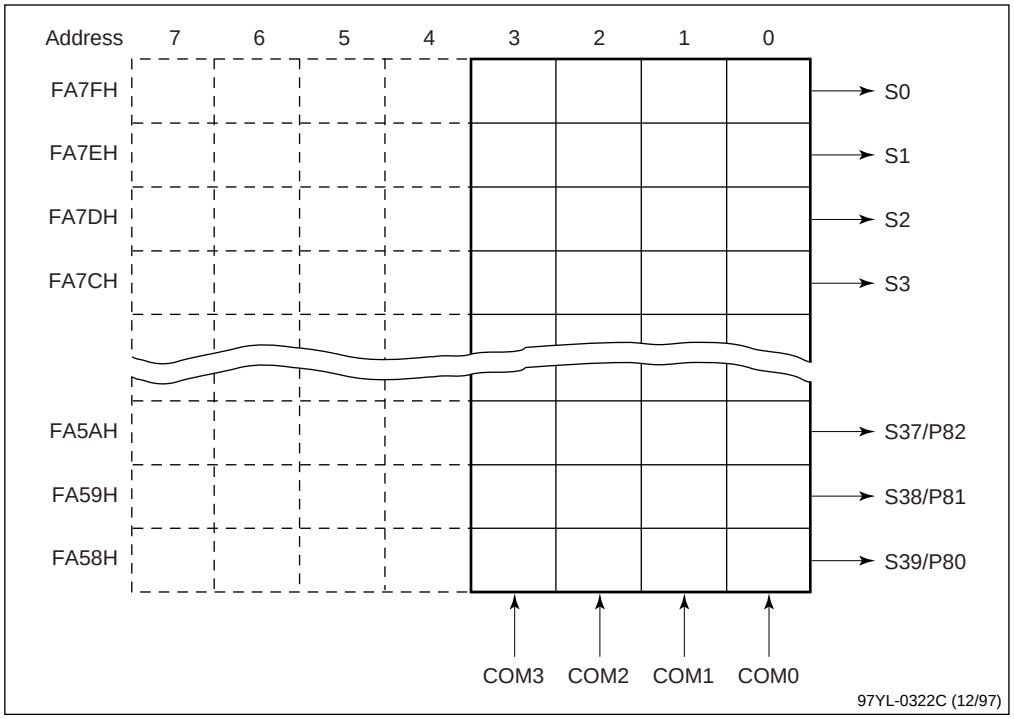
Display outputs	Dedicated segment signals: 24
	Segment signal, input/output port dual function: 16
	Common signals 4 (COM0 to COM3)
Control registers	LCD display mode register (LCDM)
	LCD display control register (LCDC)

LCD Display Data Memory

The LCD display data memory is mapped onto addresses FA58h to FA7Fh. The data stored in the memory can be displayed on an LCD panel by the LCD controller/driver.

Figure 10-1 shows the relationship between the LCD display data memory contents and the segment/common outputs. Any area not used for display can be used as normal RAM.

Figure 10-1. Display Data Memory



LCD Display Mode Register (LCDM)

1. This LCDM register (Figure 10-2) sets the display mode, supply voltage, LCD clock frequency, and display on or off.

2. The LCD clock is supplied from the clock timer. When LCD display is performed, set TMC2 bit 1 of the watch timer mode control register to 1.

3. To reduce power consumption, clear LCDM3 to 0 when LCD display is not performed. Before manipulating LCDM3, be sure to turn off the LCD display.

Figure 10-2. LCD Display Mode Register (LCDM)

7	6	5	4	3	2	1	0
LCDON	LCDM6	LCDM5	LCDM4	LCDM3	LCDM2	LCDM1	LCDM0

Bit(s)	Name	Description
2–0	LCDM2–LCDM0	LCD controller display mode
		000 Time div = 4, bias = 1/3
		001 Time div = 3, bias = 1/3
		010 Time div = 2, bias = 1/2
		011 Time div = 3, bias = 1/2
		100 Static
		Other = prohibited
3	LCDM3	LCD controller supply voltage
		0 = normal (2.7–5.5 V)
		1 = low voltage (2–3.4 V)
6–4	LCDM6–LCDM4	LCD clock frequency
		000 76 Hz (5.0 MHz), 64 Hz (4.19 MHz), 64 Hz (32.768 kHz)
		001 153 Hz (5.0 MHz), 128 Hz (4.19 MHz),128 Hz (32.768 kHz)
		010 305 Hz (5.0 MHz), 256 Hz (4.19 MHz), 256 Hz (32.768 kHz)
		011 610 Hz (5.0 MHz), 512 Hz (4.19 MHz), 512 Hz (32.768 kHz)
		100 Setting prohibited
7	LCDON	LCD display: 1 = on and 0 = off

LCD Display Control Register (LCDC)

The LCDC register (Figure 10-3) sets cutoff of the current flowing to split resistors for LCD drive voltage generation and switchover between segment output and input/output port functions.

Figure 10-3. LCD Display Control Register (LCDC)

7	6	5	4	3	2	1	0
LCD7	LCD6	LCD5	LCD4	0	0	LEPS	LIPS

Bits	Name	Description	
1–0	LEPS, LIPS	Power to LCD	
		00 Does not supply power to LCD	
		01 From V_{DD} pin	
		10 From BIAS pin	
7–4	LCD7–LCD4	Port pins	Segment pins
	0 0 0 0	P80–P97	None
	0 0 0 1	P80–P95	S24 – S25
	0 0 1 0	P80–P93	S24 – S27
	0 0 1 1	P80–P91	S24 – S29
	0 1 0 0	P80–P87	S24 – S31
	0 1 0 1	P80–P85	S24 – S33
	0 1 1 0	P80–P83	S24 – S35
	0 1 1 1	P80–P81	S24 – S37
	1 0 0 0	None	S24 – S39

LCD Controller/Driver Settings

LCD controller/driver settings should be performed as listed below. When the LCD controller/driver is used, the watch timer should be set to the operational state beforehand.

1. Set “Watch Operation enabled” in timer clock selection register 2 (TCL2) and the watch timer mode control register (TMC2).
2. Set the initial value in the display data memory (FA58h to FA7Fh).
3. Set the pins to be used as segment outputs in the LCD display control register (LCDC).
4. Set the display mode, operating mode, and LCD clock in the LCD display mode register (LCDM).
5. Set data in the display data memory according to the display contents.

Common Signals and Segment Signals

1. An individual pixel on an LCD panel lights when the potential difference of the corresponding common signal and segment signal reaches or exceeds a given voltage (the LCD drive voltage V_{LCD}).
2. For common signals, the selection timing order is COM0, 1, 2, and 3. With 2-time-division, pins COM2 and COM3 are left open; with 3-time-division, the COM3 pin is left open.
3. Segment signals correspond to a 40-byte LCD display data memory. Each memory bit 0, 1, 2, and 3 is read in synchronization with the COM0, 1, 2, and 3 timings, respectively, and if the value of the bit is 1, it is converted to the selection voltage. If the value of the bit is 0, it is converted to the non-selection voltage and output to a segment pin (S0 to S39).
4. LCD display data memory bits 1 and 2 are not used with the static display mode, bits 2 and 3 are not used with the 2-time-division method, and bit 3 is not used with the 3-time-division method. These bits can be used for purposes other than display. Bits 4 to 7 are fixed at 0.

LCD Drive Voltages

Split resistors for producing the LCD drive voltages can be incorporated in the mask ROM product by mask option. Also, an LCD drive voltage can be externally supplied from the BIAS pin to produce other LCD drive voltages. With the 1/2 bias method, the $V_{LC}1$ and $V_{LC}2$ pins must be connected externally.

Pins	No Bias	1/2 Bias	1/3 Bias
$V_{LC}0$	V_{LCD}	V_{LCD}	V_{LCD}
$V_{LC}1$	2/3 V_{LCD}	1/2 V_{LCD}	2/3 V_{LCD}
$V_{LC}2$	1/3 V_{LCD}	1/2 V_{LCD}	1/3 V_{LCD}

Example LCD Driver

The example LCD driver uses a 4-time-division display with 1/3 bias. All 40 segments (160 pixels) are used. A 10 x 14-bit segment displays time of day (hours:minutes:seconds) or other messages. The first segment is connected to the S0 output pin and the last segment to the S39 output pin. The time-of-day display is updated every second. Default time is 12:00:00.

Flowcharts

Figures 10-4, 10-5, and 10-6 are flowcharts of the LCD application programs.

Figure 10-4. Flowcharts of Character Display and LCD Module Routines

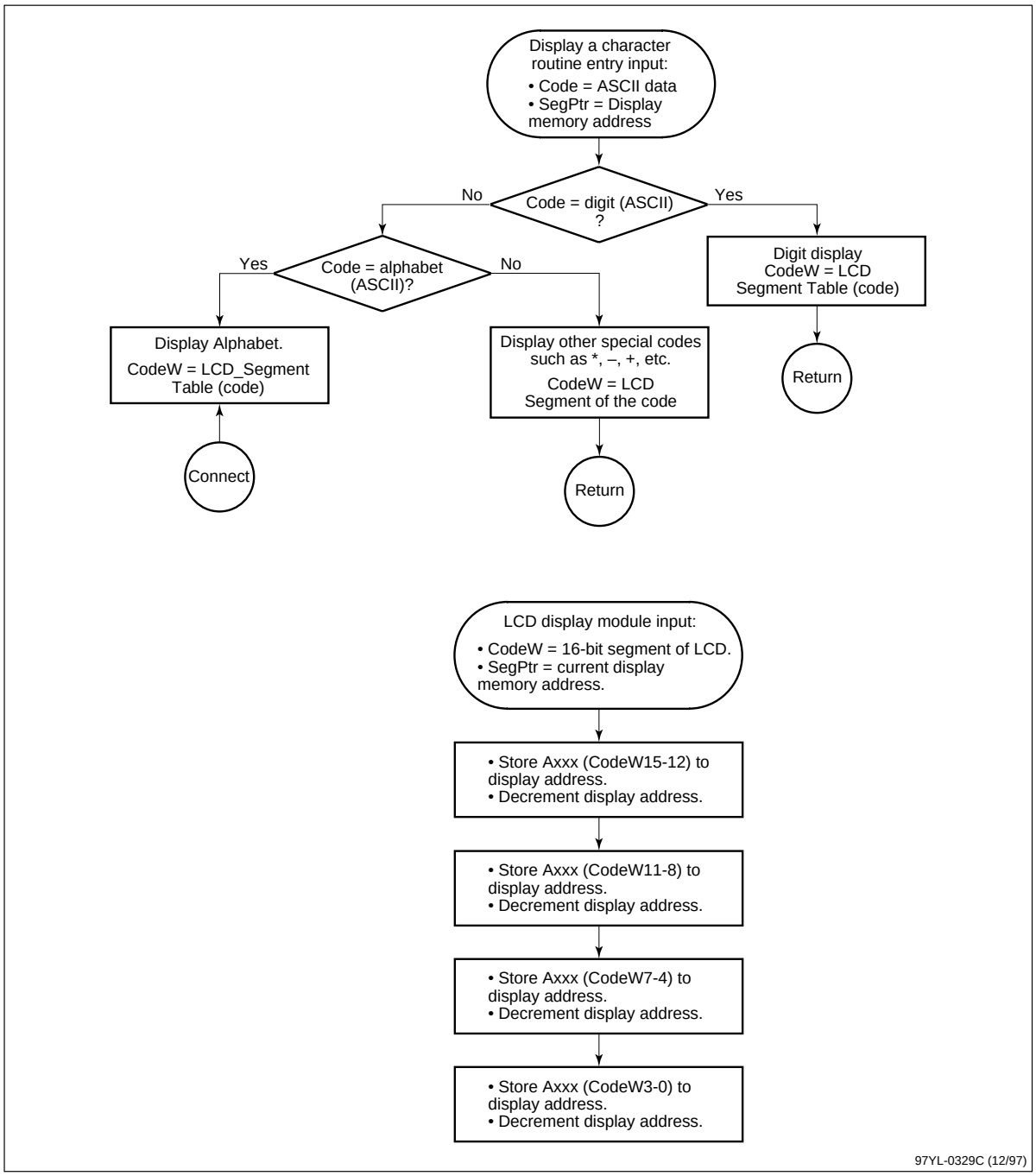


Figure 10-5. Flowchart of Prescaler Interrupt Interval Routine

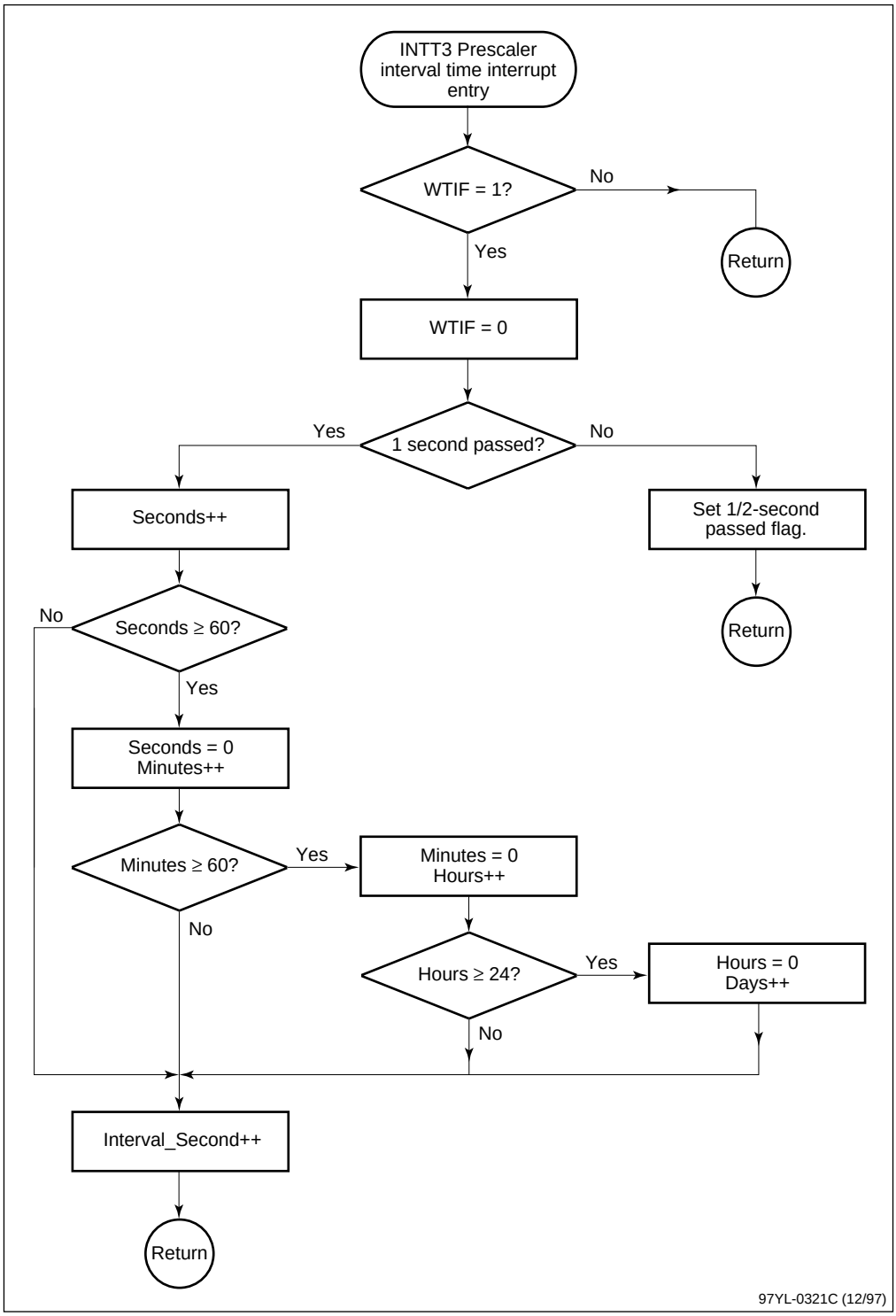
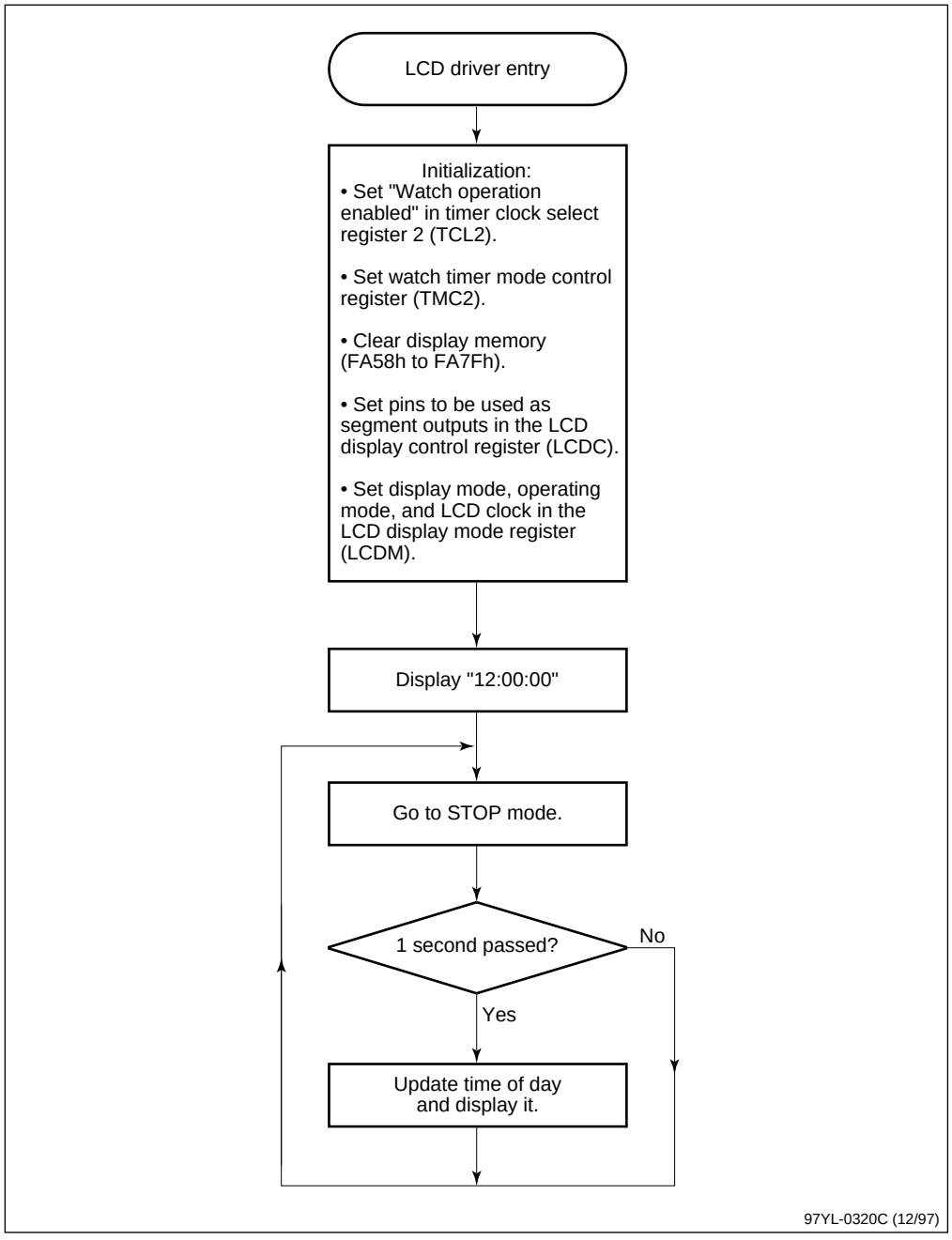


Figure 10-6. Flowchart of LCD Driver Program



Assembly Language: LCD Application Example

```
;*********************************************************************
;
;                                          File name: LCD.ASM
;                                          Date: 10/29/97
;*********************************************************************
;         Description:
;
;
;The example program is an LCD driver using 4-time-division and 1/2 bias.
;The program displays the time of day: 'HOURS:MINUTES:SECONDS,'
;Default time is 12:00:00. Every 15.6 milliseconds,
;the program checks whether a second has passed. If so, it updates the display.
;
;====================================================================
;         The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;         The LCD and watch timer are run under the subsystem clock.
;         Normal operation is under the subsystem clock, and when a key is depressed, the
;         main system clock is activated.
;====================================================================
;         for only DDB-K0070A
;
$set (DDB)
;====================================================================
;         Equates
;====================================================================
;
;         LCD display memory
;
LCDBuffer            EQU      0FA7Fh
SegmentSize          EQU      40

HOURS_10th_Pt        EQU      0FA7Fh
HOURS_Unit_Pt        EQU      HOURS_10th_Pt - 4
COLON1_Pt            EQU      HOURS_Unit_Pt - 4    ;offset 4 segments
MINUTES_10th_Pt      EQU      COLON1_Pt - 4
MINUTES_Unit_Pt      EQU      MINUTES_10th_Pt - 4
COLON2_Pt            EQU      MINUTES_Unit_Pt - 4
SECONDS_10th_Pt      EQU      COLON2_Pt -4
SECONDS_Unit_Pt      EQU      SECONDS_10th_Pt -4

AMPM_Pt              EQU      0FA7Fh

;====================================================================
;         saddr area
;         FWA = 0fe20h
;====================================================================
          dseg      saddr
;
;         Time of day by watch timer mode
;
Interval_Seconds: ds  1 ;Interval seconds

Seconds:     ds       1      ;Seconds
Minutes:     ds       1      ;Minutes
Hours:       ds       1      ;Hours
Days:        ds       1      ;Days

;====================================================================
;         Even boundary pair
;====================================================================
          dseg      saddrp
```

```
;=====================================================================
;          Bit Segment
;=====================================================================
           bseg
HalfSecondFlag         dbit    ;1/2-second toggle flag
ASecondFlag            dbit    ;1 second pass

;=====================================================================
;
;          Internal High RAM allocation
;
           dseg       IHRAM
           ds         20h
Stack:

           $ej
;=====================================================================
;          Reset vector
;=====================================================================
Vector     cseg       at 0000h
           dw         ResetStart

           org        01Eh
           dw         Watch_timer             ; Watch timer interval time interrupt

           $ej
;=====================================================================
;          Main
;=====================================================================
Main       cseg       at 80h

           public     ResetStart
ResetStart:
           di                                 ;Disable interrupt
;
;          Select register bank 0
;
           sel        RB0                     ;Select register bank 0 for background
;
;          Oscillation mode select register
;          . Does not use divider circuit
;
           mov        OSMS,#00000001b
;
;          Set stack pointer (IHMEM + 20H)
;
;;         movw       SP,#Stack               ;Stack ptr
           movw       SP,#0fe00h              ;Stack ptr
;
;          Clear display memory (FA7Fh (SEG0) - FA58h (SEG39))
;
           movw       hl,#LCDBuffer
           mov        a,#0
           mov        b,#SegmentSize
clrmem:
           mov        [hl],a
           decw       hl
           dbnz       b,$clrmem
```

```
;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 1 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 1, Select 32.768 kHz subsystem clock)
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================
          mov       TCL2,#00010000b

;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter oweration control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for Watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"                        )
;
;=====================================================================
          mov       TMC2,#01010110b

;=====================================================================
;
;         LCDC - LCD display control register
;
;         1 0 0 0 0 0 1 0
;         | | | | | | | |__ LCDC.0 (= 0, Power to LCD from BIAS pin)
;         | | | | | | |____ LCDC.1 (= 1, Power to LCD from BIAS pin)
;         | | | | | |______ LCDC.2 (= 0, Reserved)
;         | | | | |________ LCDC.3 (= 0, Reserved)
;         | | | |__________ LCDC.4 (= 0, Select S24-S39)
;         | | |____________ LCDC.5 (= 0, Select S24-S39)
;         | |______________ LCDC.6 (= 0, Select S24-S39)
;         |________________ LCDC.7 (= 1, Select S24-S39)
;
;=====================================================================
          mov       LCDC,#10000010b

;=====================================================================
;
;         LCDM - LCD display mode register
;
;         1 0 0 1 0 0 0 0
;         | | | | | | | |__ LCDM.0 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | | | |____ LCDM.1 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | | |______ LCDM.2 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | |________ LCDM.3 (= 0, Normal mode (2.7 to 5.5 V)
;         | | | |__________ LCDM.4 (= 1, LCD clock 128 Hz)
;         | | |____________ LCDM.5 (= 0, LCD clock 128 Hz)
;         | |______________ LCDM.6 (= 0, LCD clock 128 Hz)
;         |________________ LCDM.7 (= 1, Display on)
;
```

```
;=====================================================================
          mov       LCDM,#10010000b
;
;         Clear time variables
;
          mov       Seconds,#0
          mov       Minutes,#0
          mov       Hours,#12
          mov       Days,#0
          clr1      ASecondFlag
          clr1      HalfSecondFlag
;
;         Enable INTP0 for DDB-K0070A
;
$if (DDB)
          clr1      PMK0
$endif
;         Clear IRQ flags
          clr1      TMIF3
;         Enable watch timer internal timer
          clr1      TMMK3
          ei
;=====================================================================
;         Display current time of day
;         Default: 12:00:00
;=====================================================================
;
;         Set to normal operation
;
          mov       PCC,#00000000B
;
;         Go to STANDBY MODE and if wake up then
;         Check second on
;
main_loop:
          nop
          nop
          STOP
          nop
;
;         a second passed?
;
          bf        ASecondFlag,$main_loop

          clr1      ASecondFlag
;
;         Display update
;
          call      !DisplayTOD

          br        $main_loop

;=====================================================================
;
;         Default time of day
;         Default:12:00:00
;
;=====================================================================
DisplayTOD:
          movw      hl,#HOURS_10th_Pt
          mov       a,Hours
          cmp       a,#10
          bnc       $dtod10                 ;less than 10
; 0 AM to 9 AM
```

```
          movw      ax,#00
          call      !DigitDisplay
          movw      hl,#HOURS_Unit_Pt
          mov       a,Hours
          call      !DigitDisplay
          br        dtod20
dtod10:
          cmp       a,#12+1
          bnc       $dtod12                 ;less than 12
; 10 AM to 12 PM
          movw      ax,#1
          call      !DigitDisplay
          movw      hl,#HOURS_Unit_Pt
          mov       a,Hours
          sub       a,#10
          call      !DigitDisplay
          br        dtod20
;         13 to 23
dtod12:
; 13 hr to 21 hr
          cmp       a,#12+9
          bnc       $dtod14                 ;less than 12
;
          movw      ax,#00
          call      !DigitDisplay
          movw      hl,#HOURS_Unit_Pt
          mov       a,Hours
          sub       a,#12
          call      !DigitDisplay
          br        dtod20
dtod14:
; 22 hr to 23 hr
          movw      ax,#1
          call      !DigitDisplay
          movw      hl,#HOURS_Unit_Pt
          mov       a,Hours
          sub       a,#22
          call      !DigitDisplay
          br        dtod20
;
dtod20:
          movw      hl,#COLON1_Pt
          movw      ax,#'|'
          call      !DisplayOther
;
;         MINUTE: xx
;
          movw      hl,#MINUTES_10th_Pt

          mov       a,Minutes
          mov       x,#0
mloop:
          cmp       a,#10
          bc        $mless10
          sub       a,#10
          inc       x
          br        mloop
;
mless10:
          xch       a,x
;         A = 10th , X = unit
          push      ax
          call      !DigitDisplay
          pop       ax
```

```
          xch       a,x
          call      !DigitDisplay
;
;          (:)
;
          movw      hl,#COLON2_Pt
          movw      ax,#'|'
          call      !DigitDisplay
;
;         Second: xx
;
          movw      hl,#SECONDS_10th_Pt

          mov       a,Seconds
          mov       x,#0
sloop:
          cmp       a,#10
          bc        $sless10
          sub       a,#10
          inc       x
          br        sloop
;
sless10:
          xch       a,x
;         A = 10th , X = unit
          push      ax
          call      !DigitDisplay
          pop       ax
          xch       a,x
          call      !DigitDisplay
          br        DisplayOther


;=====================================================================
;         Display a string to the LCD
;         Terminated either 00 or 0FFH
;
;         Input:
;                   DE = String ptr
;         Return:
;                   ptr = ptr++;
;=====================================================================
DisplayAString:
          movw      hl,#LCDBuffer
dlop:
          mov       a,[de]
          cmp       a,#00
          bz        $ds10                   ;done
          cmp       a,#0ffh
          bz        $ds10                   ;done
;
          push      de
          call      !DisplayACharacter

          pop       de
          incw      de
          br        dlop
ds10:
          ret

;=====================================================================
;         Display a character to the LCD
;
;         Input: a reg
```

```
;                hl = Position
;=====================================================================
DisplayACharacter:
          push      hl

          mov       b,a

          movw      hl,#SegA14              ;Alphabet
          cmp       a,#30h
          bc        $achar25
;
          cmp       a,#3Ah
          bnc       $achar20                ;Alphabet
;
;         Digit
;
          sub       a,#30h
          pop       hl
;
;         Digit display
;
DigitDisplay:
          push      hl
          movw      hl,#SegD14              ;Digit
          br        $achar30
;
;         Check Alphabets
;
achar20:
          cmp       a,#41h
          bc        $achar25
          cmp       a,#41H+26
;
;         Alphabet
;
          sub       a,#41h
          pop       hl
;
;         Alphabet display
;
AlphabetDisplay:
          push      hl
          br        $achar30
;
;         Not Alphabet or Digits
;
achar25:
          pop       hl
;
;
;
DisplayOther:
          push      hl
          cmp       a,#':'
          bnz       $ck10
          movw      ax,#L_COLON
          br        achar80
ck10:
          cmp       a,#'-'
          bnz       $ck20
          movw      ax,#L_DASH
          br        achar80
ck20:
          movw      ax,#00                  ;blank
```

```
          br        achar80
;
;         Read 2 bytes
;
achar30:
          clr1      cy
          rolc      a,1                     ;* 2
          mov       b,a
          mov       a,[hl+b]
          xch       a,x
          inc       b
          mov       a,[hl+b]
          xch       a,x                     ;a = upper, x = lower byte
;
;         a = upper byte/b = lower byte
;
;         X---h
achar80:
          pop       hl
;
;         HL = data ptr
;         AX = data
;
DisplayData:
;         X---h
          push      ax
          ror       a,1
          ror       a,1
          ror       a,1
          ror       a,1
          and       a,#0fh
          mov       [hl+0],a
;         -X--h
          pop       ax
          and       a,#0fh
          mov       [hl+1],a
;         --X-h
          mov       a,x
          push      ax

          ror       a,1
          ror       a,1
          ror       a,1
          ror       a,1
          and       a,#0fh
          mov       [hl+2],a
;         ---Xh
          pop       ax
          and       a,#0fh
          mov       [hl+3],a
;
          decw      hl
          decw      hl
          decw      hl
          decw      hl
          ret
;* ********************************************
;*          Define COM Line #
;* ********************************************
COM1      equ 1
COM2      equ 2
COM3      equ 4
COM4      equ 8
;*
```

```
;*          Segment Definitions
;*
s16_G      equ COM1
s16_J      equ COM2
s16_L      equ COM3
s16_D      equ COM4

s16_O      equ COM1*010h
s16_F      equ COM2*010h
s16_E      equ COM3*010h
s16_P      equ COM4*010h

s16_I      equ COM1*0100h
s16_B      equ COM2*0100h
s16_C      equ COM3*0100h
s16_N      equ COM4*0100h

s16_A      equ COM1*01000h
s16_H      equ COM2*01000h
s16_K      equ COM3*01000h
s16_M      equ COM4*01000h
;* ******************************************
;          Table for 14-Segment LCDs
;* ******************************************
SegD14:
           dw         s16_A+s16_B+s16_C+s16_D+s16_E+s16_F                          ;0
           dw         s16_B+s16_C                                                 ;1
           dw         s16_A+s16_B+s16_K+s16_J+s16_E+s16_D                          ;2
           dw         s16_A+s16_B+s16_K+s16_J+s16_C+s16_D                          ;3
           dw         s16_F+s16_J+s16_K+s16_B+s16_C                                ;4
           dw         s16_A+s16_F+s16_J+s16_K+s16_C+s16_D                          ;5
           dw         s16_A+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K                    ;6
           dw         s16_A+s16_B+s16_C                                           ;7
           dw         s16_A+s16_B+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K              ;8
           dw         s16_A+s16_B+s16_C+s16_D+s16_F+s16_J+s16_K                    ;9
;*
;*          Alphabet
;*
SegA14:
           dw         s16_F+s16_E+s16_A+s16_B+s16_C+s16_J+s16_K                    ;A
           dw         s16_A+s16_B+s16_C+s16_D+s16_H+s16_M+s16_K                    ;B
           dw         s16_A+s16_F+s16_E+s16_D                                      ;C
           dw         s16_A+s16_B+s16_C+s16_D+s16_H+s16_M                          ;D
           dw         s16_A+s16_F+s16_E+s16_D+s16_J+s16_K                          ;E
           dw         s16_A+s16_F+s16_E+s16_J+s16_K                                ;F
           dw         s16_A+s16_F+s16_E+s16_D+s16_C+s16_K                          ;G
           dw         s16_F+s16_E+s16_J+s16_K+s16_B+s16_C                          ;H
           dw         s16_A+s16_H+s16_M+s16_D                                      ;I
           dw         s16_E+s16_B+s16_C+s16_D                                      ;J
           dw         s16_F+s16_E+s16_J+s16_I+s16_N                                ;K
           dw         s16_F+s16_E+s16_D                                           ;L
           dw         s16_F+s16_E+s16_G+s16_I+s16_B+s16_C                          ;M
           dw         s16_F+s16_E+s16_G+s16_N+s16_B+s16_C                          ;N
           dw         s16_F+s16_E+s16_A+s16_B+s16_C+s16_D                          ;O
           dw         s16_F+s16_E+s16_A+s16_B+s16_K+s16_J                          ;P
           dw         s16_E+s16_F+s16_A+s16_B+s16_C+s16_D+s16_N                    ;Q
           dw         s16_F+s16_E+s16_A+s16_B+s16_K+s16_J+s16_N                    ;R
           dw         s16_A+s16_F+s16_J+s16_K+s16_C+s16_D                          ;S
           dw         s16_A+s16_H+s16_M                                            ;T
           dw         s16_F+s16_E+s16_D+s16_C+s16_B                                ;U
           dw         s16_F+s16_E+s16_L+s16_I                                      ;V
           dw         s16_F+s16_E+s16_L+s16_N+s16_C+s16_B                          ;W
           dw         s16_G+s16_N+s16_I+s16_L                                      ;X
           dw         s16_F+s16_J+s16_K+s16_B+s16_M                                ;Y
```

```
          dw        s16_A+s16_I+s16_L+s16_D                                    ;Z

L_DASH    equ       s16_J+s16_K             ;dash (-)
L_UNDER   equ       s16_D                   ;underline (_)
L_SLASH   equ       s16_I+s16_L             ;slash (/)
L_BSLASH  equ       s16_G+s16_N             ;Back slash (\)
L_STAR    equ       s16_I+s16_L+s16_G+s16_N ;star (*)
R_ARROW   equ       s16_G+s16_L+s16_J       ;Right arrow (->)
L_ARROW   equ       s16_I+s16_N+s16_N       ;Left arrow (<-)
L_BRACK   equ       s16_I+s16_N             ;<
R_BRACK   equ       s16_G+s16_L             ;>
L_APOSTR  equ       s16_I                   ;apostrophe (')
L_COLON   equ       s16_H+s16_M             ;colon (:)
;* ******************************************
;*         7-Segment LCDs
;* ******************************************
s7_E   equ COM1
s7_G   equ COM2
s7_F   equ COM3
s7_K   equ COM4
s7_C   equ COM1*010h
s7_B   equ COM2*010h
s7_A   equ COM3*010h
s7_D   equ COM4*010h
;* ******************************************
;*         7-Segment LCDs
;* ******************************************
Tab7Seg:
          ds        s7_A+s7_B+s7_C+s7_D+s7_E+s7_F          ;0
          ds        s7_B+s7_C                              ;1
          ds        s7_A+s7_B+s7_G+s7_E+s7_D               ;2
          ds        s7_A+s7_B+s7_G+s7_C+s7_D               ;3
          ds        s7_F+s7_G+s7_B+s7_C                    ;4
          ds        s7_A+s7_F+s7_G+s7_C+s7_D               ;5
          ds        s7_A+s7_F+s7_G+s7_C+s7_D+s7_E          ;6
          ds        s7_A+s7_B+s7_C                         ;7
          ds        s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G     ;8
          ds        s7_A+s7_B+s7_C+s7_D+s7_F+s7_G          ;9

          ds        s7_A+s7_B+s7_E+s7_F+s7_G               ;P in open
;
;         Must be less than 15
;
S7_S   equ     11
          ds        s7_A+s7_C+s7_D+s7_F+s7_G               ;S
S7_E   equ     12
          ds        s7_A+s7_D+s7_F+s7_E+s7_G               ;E
S7_P   equ     13
          ds        s7_A+s7_B+s7_F+s7_E+s7_G               ;P

dash_7    equ       s7_G
          ds        s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G     ;A

Colon7seg:
          ds        s7_B+s7_C                              ;: - colon

          $ej
;=======================================================================
;         Module name: INTTM3 watch timer interval time interrupt
;
;         Interrupted every 15.6 ms.
;
;         DESCRIPTION:
;                   Set watch time variables
```

```
;
;         OPERATION:
;                   . Flip the HalfSecondFlag
;                   . If HalfSecondFlag = 0, then incr. seconds
;                   . If Seconds == 60, incr. Minutes
;                   . If Minutes == 60, incr. Hours
;                   . If Hours   == 24, incr. Days
;
;=====================================================================
          public    Watch_timer
Watch_timer:
;
;         Test half second flag
;
          bf        WTIF,$wt20              ;not yet
          clr1      WTIF
;
;         1/2 second passed
;
          bf        HalfSecondFlag,$wt10  ;1/2 second passed
;         Clear half-second flag
          clr1      HalfSecondFlag
;
          set1      ASecondFlag
;
          inc       Interval_Seconds
          inc       Seconds
          cmp       Seconds,#60
          bc        $wt20
          mov       Seconds,#0
;
          inc       Minutes
          cmp       Minutes,#60
          bc        $wt20
          mov       Minutes,#0
;
          inc       Hours
          cmp       Hours,#24
          bc        $wt20
          mov       Hours,#0
;         Day
          inc       Days
          cmp       Days,#30
          bc        $wt20

          br        $wt20
;
wt10:
          set1      HalfSecondFlag
wt20:
          reti

;=====================================================================
          end
_

#pragma interrupt INTTM3 WatchTimer RB1
```

C Language: LCD Application Examples

```
/*;*************************************************************************
;
;                                          File name: LCD.C
;                                          Date: 10/29/97
;*************************************************************************
;         Description:
;
;
;The example program is an LCD driver using 4-time-division and 1/2 bias.
;The program displays the time of day, 'HOURS:MINUTES:SECONDS,'
;Default time is 12:00:00. Every 15.6 milliseconds,
;the program checks whether a second has passed. If so, it updates the display;
;=======================================================================
;         The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;         The LCD and watch timer are run under the subsystem clock.
;         Normal operation is under the subsystem clock, and when a key is depressed, the
;         main system clock is activated.
;=======================================================================*/
#include "define.h"
/*;=======================================================================
;         Equates
;=======================================================================*/
//;
//;          LCD Display Memory
//;
#define LCDBuffer             0xFA7F
#define SegmentSize           40

#define HOURS_10th_Pt         0xFA7F
#define HOURS_Unit_Pt         HOURS_10th_Pt- 4
#define COLON1_Pt             HOURS_Unit_Pt - 4    //;offset 4 segments
#define MINUTES_10th_Pt       COLON1_Pt - 4
#define MINUTES_Unit_Pt       MINUTES_10th_Pt - 4
#define COLON2_Pt             MINUTES_Unit_Pt - 4
#define SECONDS_10th_Pt       COLON2_Pt -4
#define SECONDS_Unit_Pt       SECONDS_10th_Pt -4

#define AMPM_Pt               0xFA7F

//;
//;          Time of day by Watch timer mode
//;
__sreg        Interval_Seconds;
__sreg        unsigned char Seconds          ; //Seconds
__sreg        unsigned char Minutes          ; //Minutes
__sreg        unsigned char Hours            ; //Hours
__sreg        unsigned char Days             ; //Days
__sreg        unsigned char *SegPtr          ; //Segment ptr
__sreg        unsigned int CodeW;
bit HalfSecondFlag                           ; // 1/2 -second toggle flag
bit ASecondFlag                              ; // 1-second pass flag
void DisplayACharacter (unsigned char code);
void DisplayAString (char *);
void DisplayTOD (void);
void DisplayOther (unsigned char code);
void LCDDisplay (void);
void DigitDisplay (char code);
void AlphabetDisplay (char code);

/* ;* ****************************************
;*          Define COM Line #
;* **************************************** */
```

```
#define COM1            1
#define COM2            2
#define COM3            4
#define COM4            8
//*
//*                       Segment Definitions
//*
#define s16_G           COM1
#define s16_J           COM2
#define s16_L           COM3
#define s16_D           COM4

#define s16_O           COM1*0x10
#define s16_F           COM2*0x10
#define s16_E           COM3*0x10
#define s16_P           COM4*0x10

#define s16_I           COM1*0x100
#define s16_B           COM2*0x100
#define s16_C           COM3*0x100
#define s16_N           COM4*0x100

#define s16_A           COM1*0x1000
#define s16_H           COM2*0x1000
#define s16_K           COM3*0x1000
#define s16_M           COM4*0x1000

/*;* *********************************************
;         Table for 14-Segment LCDs
;* ********************************************* */
unsigned int const SegD14[] =
{
       s16_A+s16_B+s16_C+s16_D+s16_E+s16_F                 // ;0
       ,s16_B+s16_C                                       // ;1
       ,s16_A+s16_B+s16_K+s16_J+s16_E+s16_D                // ;2
       ,s16_A+s16_B+s16_K+s16_J+s16_C+s16_D                // ;3
       ,s16_F+s16_J+s16_K+s16_B+s16_C                      // ;4
       ,s16_A+s16_F+s16_J+s16_K+s16_C+s16_D                // ;5
       ,s16_A+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K          // ;6
       ,s16_A+s16_B+s16_C                                  // ;7
       ,s16_A+s16_B+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K    //;8
       ,s16_A+s16_B+s16_C+s16_D+s16_F+s16_J+s16_K          //;9
};

//;*
//;*      Alphabet
//;*
const unsigned int SegA14[] =
{
       s16_F+s16_E+s16_A+s16_B+s16_C+s16_J+s16_K           //;A
       ,s16_A+s16_B+s16_C+s16_D+s16_H+s16_M+s16_K          //;B
       ,s16_A+s16_F+s16_E+s16_D                            //;C
       ,s16_A+s16_B+s16_C+s16_D+s16_H+s16_M                //;D
       ,s16_A+s16_F+s16_E+s16_D+s16_J+s16_K                //;E
       ,s16_A+s16_F+s16_E+s16_J+s16_K                      //;F
       ,s16_A+s16_F+s16_E+s16_D+s16_C+s16_K                //;G
       ,s16_F+s16_E+s16_J+s16_K+s16_B+s16_C                //;H
       ,s16_A+s16_H+s16_M+s16_D                             //;I
       ,s16_E+s16_B+s16_C+s16_D                             //;J
       ,s16_F+s16_E+s16_J+s16_I+s16_N                       //;K
       ,s16_F+s16_E+s16_D                                   //;L
       ,s16_F+s16_E+s16_G+s16_I+s16_B+s16_C                 //;M
       ,s16_F+s16_E+s16_G+s16_N+s16_B+s16_C                 //;N
```

```
       ,s16_F+s16_E+s16_A+s16_B+s16_C+s16_D                  //;O
       ,s16_F+s16_E+s16_A+s16_B+s16_K+s16_J                  //;P
       ,s16_E+s16_F+s16_A+s16_B+s16_C+s16_D+s16_N            //;Q
       ,s16_F+s16_E+s16_A+s16_B+s16_K+s16_J+s16_N            //;R
       ,s16_A+s16_F+s16_J+s16_K+s16_C+s16_D                  //;S
       ,s16_A+s16_H+s16_M                                    //;T
       ,s16_F+s16_E+s16_D+s16_C+s16_B                        //;U
       ,s16_F+s16_E+s16_L+s16_I                              //;V
       ,s16_F+s16_E+s16_L+s16_N+s16_C+s16_B                  //;W
       ,s16_G+s16_N+s16_I+s16_L                              //;X
       ,s16_F+s16_J+s16_K+s16_B+s16_M                        //;Y
       ,s16_A+s16_I+s16_L+s16_D                              //;Z
};

#define L_DASHs16_J+s16_K                                   //;dash (-)
#define L_COLON s16_H+s16_M                                 //;colon (:)

/* ;* ********************************************
;*         7-Segment LCDs
;* ******************************************** */

#define s7_E   COM1
#define s7_G   COM2
#define s7_F   COM3
#define s7_K   COM4
#define s7_C   COM1*0x10
#define s7_B   COM2*0x10
#define s7_A   COM3*0x10
#define s7_D   COM4*0x10

/*;* ********************************************
;*         7-Segment LCDs
;* ******************************************** */
const unsigned char Tab7Seg[] =
{
       s7_A+s7_B+s7_C+s7_D+s7_E+s7_F                        //;0
       ,s7_B+s7_C                                           //;1
       ,s7_A+s7_B+s7_G+s7_E+s7_D                            //;2
       ,s7_A+s7_B+s7_G+s7_C+s7_D                            //;3
       ,s7_F+s7_G+s7_B+s7_C                                 //;4
       ,s7_A+s7_F+s7_G+s7_C+s7_D                            //;5
       ,s7_A+s7_F+s7_G+s7_C+s7_D+s7_E                       //;6
       ,s7_A+s7_B+s7_C                                      //;7
       ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G                  //;8
       ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_G                       //;9
//;
//;      Must be less than 15
//;
       ,s7_A+s7_C+s7_D+s7_F+s7_G                            //;S
       ,s7_A+s7_D+s7_F+s7_E+s7_G                            //;E
       ,s7_A+s7_B+s7_F+s7_E+s7_G                            //;P
       ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G                  //;A
       ,s7_B+s7_C                                           //;: - colon
};


//;=====================================================================
//;         Main
//;=====================================================================
void main ()
{

char i;
char *dptr;
```

```
//;
//;         Oscillation mode select register
//;          . Does not use divider circuit
//;
       OSMS = 0b00000001;            // select 5 MHz
       PMK0 = 0;                     // for DDB BOARD - INTP0 has to be set
//;
//;         Clear Display Memory (FA7Fh (SEG0) - FA58h (SEG39))
//;
          dptr = LCDBuffer;
          for (i=0;i<SegmentSize;i++) *dptr-- = 0;
/*;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 1 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 1, Select 32.768 kHz subsystem clock)
;         | | |____________ TCL2.5 (Buzzer Output Frequency selection)
;         | |______________ TCL2.6 (Buzzer Output Frequency selection)
;         |________________ TCL2.7 (Buzzer Output Frequency selection)
;
;=====================================================================*/
          TCL2= 0b00010000;

/*;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch Operating Mode Selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler Operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)
;
;=====================================================================*/
          TMC2 = 0b01010110;

/*;=====================================================================
;
;         LCDC - LCD display control register
;
;         1 0 0 0 0 0 1 0
;         | | | | | | | |__ LCDC.0 (= 0, Power to LCD from BIAS pin)
;         | | | | | | |____ LCDC.1 (= 1, Power to LCD from BIAS pin)
;         | | | | | |______ LCDC.2 (= 0, Reserved)
;         | | | | |________ LCDC.3 (= 0, Reserved)
;         | | | |__________ LCDC.4 (= 0, Select S24-S39)
;         | | |____________ LCDC.5 (= 0, Select S24-S39)
;         | |______________ LCDC.6 (= 0, Select S24-S39)
;         |________________ LCDC.7 (= 1, Select S24-S39)
;
;=====================================================================*/
          LCDC = 0b10000010;
/*;=====================================================================
;
;         LCDM - LCD Display mode register
;
```

```
;         1 0 0 1 0 0 0 0
;         | | | | | | | |__ LCDM.0 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | | | |____ LCDM.1 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | | |______ LCDM.2 (= 0, 4-time-division, 1/3 bias mode)
;         | | | | |________ LCDM.3 (= 0, Normal mode (2.7 to 5.5 v)
;         | | | |__________ LCDM.4 (= 1, LCD clock 128 Hz)
;         | | |____________ LCDM.5 (= 0, LCD clock 128 Hz)
;         | |______________ LCDM.6 (= 0, LCD clock 128 Hz)
;         |________________ LCDM.7 (= 1, Display on)
;
;=====================================================================*/
          LCDM = 0b10010000;
//;
//;         Clear time variables
//;
          Seconds=0;
          Minutes=0;
          Hours=0;
          Days= 0;
          ASecondFlag = 0;
          HalfSecondFlag = 0;
//;         Clear IRQ flags
          TMIF3 = 0;
//;         Enable watch timer Internal timer
          TMMK3 = 0;

          EI ();
/*;=====================================================================
;         Display current time of day
;         Default: 12:00:00
;=====================================================================*/
//;
//;         Set to normal operation
//;
          PCC = 0b00000000;
//;
//;         Go to STANDBY MODE and if wake up then
//;         Check Second on
//;
          while (TRUE)
          {
                    NOP ();
                    NOP ();
                    STOP ();
                    NOP ();
//;
//;         a second passed?
//;
                    if ( ASecondFlag )
                    {
                                              ASecondFlag = 0;
//;
//;         Display update
//;
                                              DisplayTOD ();
                    }
          }

}
/*;=====================================================================
;         Default Time of day
;         Default:12:00:00
;=====================================================================*/
void DisplayTOD (void)
```

```
{
          SegPtr = HOURS_10th_Pt;
          if (Hours < 10)
          {
//; 0 AM to 9 AM
                    DigitDisplay (0);
                    SegPtr = HOURS_Unit_Pt;
                    DigitDisplay (Hours);
          }
          if (Hours < 12)
          {
//; 10 AM to 12 AM
                    DigitDisplay (1);
                    SegPtr = HOURS_Unit_Pt;
                    DigitDisplay (Hours-10);
          }
          if (Hours < 21)
          {
//; 1 PM to 9 PM
                    DigitDisplay (0);
                    SegPtr = HOURS_Unit_Pt;
                    DigitDisplay (Hours-12);
          }
          if (Hours >= 22)
          {
//; 10 PM to 12 PM
                    DigitDisplay (1);
                    SegPtr = HOURS_Unit_Pt;
                    DigitDisplay (Hours-22);
          }
// :
          SegPtr = COLON1_Pt;
          DisplayOther ('|');
//; minutes
          SegPtr = MINUTES_10th_Pt;
          DigitDisplay (Minutes/60);
          SegPtr = MINUTES_Unit_Pt;
          DigitDisplay (Minutes%60);


// :
          SegPtr = COLON2_Pt;
          DisplayOther ('|');
//; Seconds
          SegPtr = SECONDS_10th_Pt;
          DigitDisplay (Seconds/60);
          SegPtr = SECONDS_Unit_Pt;
          DigitDisplay (Seconds%60);
}


/*;=====================================================================
;         Display a string to the LCD
;         Terminated either 00 or 0FFH
;
;         Input:
;                   HL = String ptr
;         Return:
;                   ptr = ptr++;
;=====================================================================*/
void DisplayAString (char Buffer[])
{

char *dptr;
```

```
          SegPtr =Buffer;
          while (!*dptr)
          {
                    DisplayACharacter (*dptr++);
          }
}

/*;=====================================================================
;         Display a character to the LCD
;
;         Input: a reg
;                hl = Position
;=====================================================================*/
void DisplayACharacter (unsigned char code)
{

          if (code >= 0x30 && code <= 0x39 )
          {
                    DigitDisplay (code*2);
          }
          else
          {
           if (code >= 0x41 && code <= 0x59 )
           {
                    AlphabetDisplay (code*2);
           }
           else
                    DisplayOther (code);
          }
}
void DigitDisplay (char code)
{
          CodeW = SegD14[  (code-0x30) * 2];
          LCDDisplay ();
}
void AlphabetDisplay (char code)
{
          CodeW = SegA14[  (code-0x41) * 2];
          LCDDisplay ();
}

void DisplayOther (unsigned char code)
{
          if (code == 0)
          {
                    CodeW = 0;
                    LCDDisplay ();
          }
          if (code == ':')
          {
                    CodeW = L_COLON;
                    LCDDisplay ();
          }
          if (code == '-')
          {
                    CodeW = L_DASH;
                    LCDDisplay ();
          }
}
/*;=====================================================================
;         Display a code
;         INPUT: CodeW = 16 bits
;
```

```
;=====================================================================*/
void LCDDisplay (void)
{
unsigned int c;
//;          X---h
          c =  (CodeW & 0xf000) >> 12;
          c &= 0xf;
          *SegPtr-- = (char)c;
//;          -X--h
          c =  (CodeW & 0x0f00) >> 8;
          c &= 0xf;
          *SegPtr-- = (char)c;
//;          --X-h
          c =  (CodeW & 0x00f0) >> 4;
          c &= 0xf;
          *SegPtr-- = (char)c;
//;          ---Xh
          c =  (CodeW & 0x000f);
          *SegPtr-- = (char)c;
}

/*;=====================================================================
;          Module name: INTTM3 Watch timer Interval Time Interrupt
;
;          DESCRIPTION:
;                    Set watch time variables
;
;          OPERATION:
;                    . Flip the HalfSecondFlag.
;                    . If HalfSecondFlag = 0, then incr. seconds
;                    . If Seconds = 60, incr. Minutes
;                    . If Minutes = 60, incr. Hours
;                    . If Hours   = 24, incr. Days
;
;=====================================================================*/
void WatchTimer ()
{
//;
//;          Test half second flag
//;
          if ( !WTIF) return;
//;
          WTIF = 0;
//;
//;         1/2 second passed
//;
          if ( !HalfSecondFlag)
          {
                    HalfSecondFlag = 1;
                    return;;
          }

//;
          HalfSecondFlag = 0;
//;
          ASecondFlag = 1;

          Interval_Seconds++;

          Seconds++;
          if (Seconds < 60) return;;

          Seconds = 0;
          Minutes++;
```

```
            if (Minutes < 60) return;;

            Minutes = 0;
            Hours++;
            if (Hours < 24) return;;

            Hours = 0;
            Days++;
}
_
```

NEC

Keyboard Input 11

Applicable K0 Devices

- µPD7801xF
- µPD7801xH
- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A
- µPD7806x
- µPD78030x

Description

The keyboard input application is one of the most useful applications in consumer products because the user interface requires a data or command input from a keyboard. The other important function with key input is the standby mode. Most applications are activated by a key input, and if a key is not depressed, the system is in a standby mode to save power. In the K0 family, port 11 is designated as the key input port for the standby mode of the microcontroller. (The µPD7801xF, µPD7801xH, µPD7805x, µPD7805xF, µPD78005x, µPD7807x, and µPD78070A use port 4.)

The key return mode register KRM (Figure 11-1) enables/disables the standby mode release by a key return signal. When falling-edge detection of port 11 is used, KRIF should be cleared to 0 (it is not cleared automatically).

Figure 11-1. Key Return Mode Register (KRM)

7	6	5	4	3	2	1	0
0	0	0	0	KRM3	KRM2	KRMK	KRIF

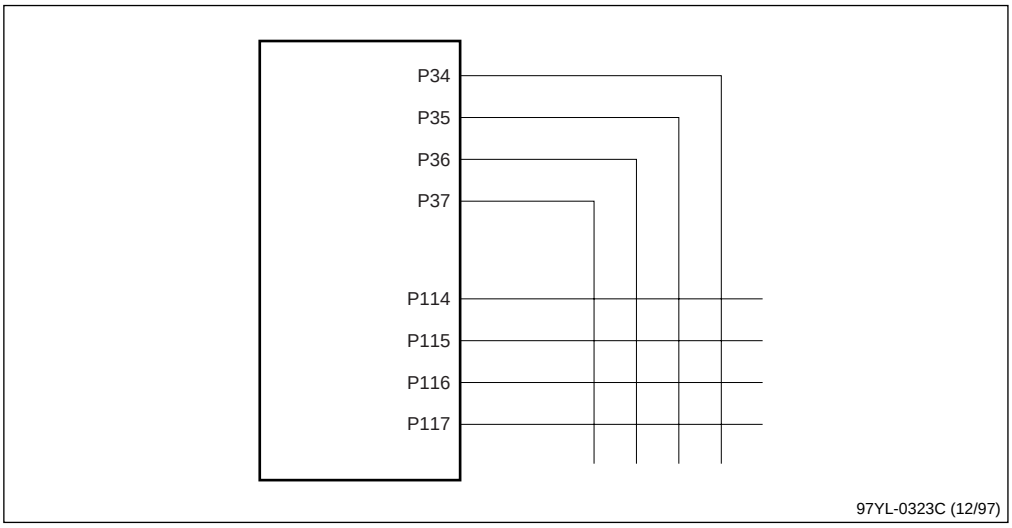
Bit(s)	Name	Description
0	KRIF	Key return signal detection flag
		0 = Not detected
		1 = Detected (falling edge on port 11)
1	KRMK	Standby mode control by key return signal
		0 = Standby mode release enabled
		1 = Standby mode release disabled
3–2	KRM3–KRM2	Selection of port 11 falling edge input
		00 P117
		01 P114–P117
		10 P112–P117
		11 P110–P117

Example Program

The example program is a module that inputs signals from a matrix of 4 x 4 keys (Figure 11-2). The keys can be pressed successively, and two or more keys can be pressed simultaneously. In the circuit shown in this example, the higher 4 bits of port 3 (P34–P37) are used as key scan signals, and port 11 is used for key return signals. As the pullup resistor of port 11 for key return, use the internal pullup resistor set by software.

Port 11 of the K0 family has a function to detect the falling edges of the eight port pins in parallel. If port 11 is used for key return signals, therefore, the standby mode can be released through detection of a falling edge by key input.

Figure 11-1. 4 x 4 Key Matrix

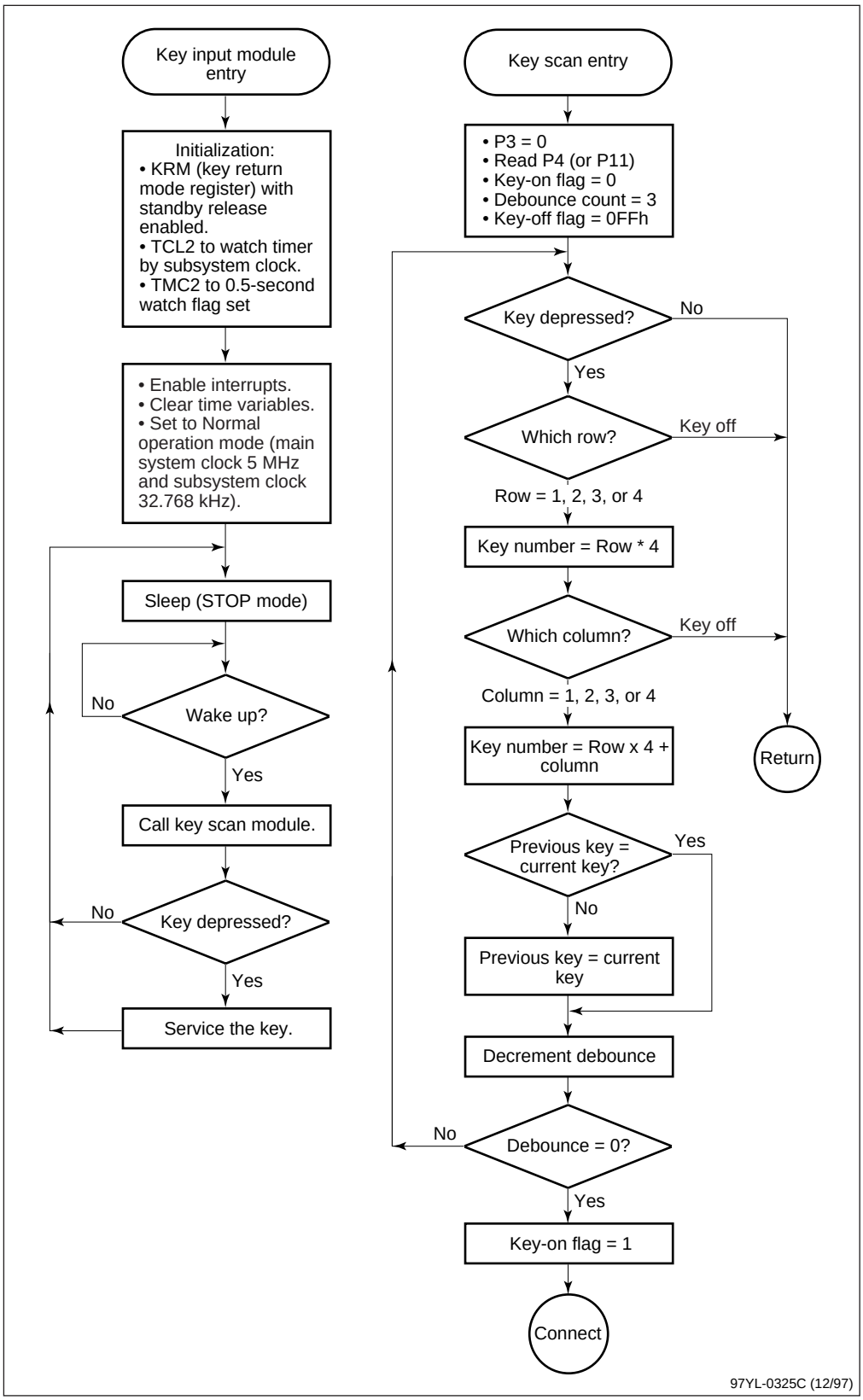


The input keys are stored to RAM on a bit/key basis. The RAM bit corresponding to a pressed key is set and the bit corresponding to a released key is cleared. By testing the RAM data on a one-bit basis starting from the first bit, the key status can be checked. To absorb key debouncing, the key is assumed to be valid when four successive key codes coincide with a given code. For example, if a key code is sampled every 16 ms, debouncing of 48 to 64 ms can be absorbed.

Key Number

0 = P114 + P34	8 = P116 + P34
1 = P114 + P35	9 = P116 + P35
2 = P114 + P36	10 = P116 + P36
3 = P114 + P37	11 = P116 + P37
4 = P115 + P34	12 = P117 + P34
5 = P115 + P35	13 = P117 + P35
6 = P115 + P36	14 = P117 + P36
7 = P115 + P37	15 = P117 + P37

Figure 11-1. Flowchart of Key Input Routine



Assembly Language Program: Example of Key Matrix

```
;************************************************************************
;
;                                          File name: key.ASM
;                                          Date: 10/27/97
;************************************************************************
;         Description:
;
;
;The example program is a key input module that inputs signals from a 4 x 4 key matrix.
;The keys can be pressed successively, and two or more keys can
;be pressed simultaneously. In this example, the higher 4
;bits of port 3 (P34 - P37) are used as key scan signals, and port 11 (P4 for µPD78054 or µPD78078)
;is used for key return signals. As the pullup resistor of port 11 for key return, the
;internal pullup resistor set by software is used.
;
;Port 11 of the K0 series has a function to detect the falling edges of the
;eight port pins in parallel. If port 11 is used for key return signals,
;therefore, the standby mode can be released through detection of a falling
;edge by key input.
;
;The input keys are stored to RAM on a one bit/key basis. The RAM bit
;corresponding to a pressed key is set and the bit corresponding to a released
;key is cleared. By testing the RAM data on a 1-bit basis starting
;from the first bit, the key status can be checked. To absorb key debouncing,
;the key is assumed to be valid when four successive key codes coincide with a
;given code. For example, if a key code is sampled every 16 ms, debouncing of
;48 ms to 64 ms can be absorbed.
;
; Key number:
;        0 = P114 + P34
;        1 = P114 + P35
;        2 = P114 + P36
;        3 = P114 + P37
;
;        4 = P115 + P34
;        5 = P115 + P35
;        6 = P115 + P36
;        7 = P115 + P37
;
;        8 = P116 + P34
;        9 = P116 + P35
;        10 = P116 + P36
;        11 = P116 + P37
;
;        12 = P117 + P34
;        13 = P117 + P35
;        14 = P117 + P36
;        15 = P117 + P37
;
;         Port 4 is used instead of port 11 for µPD78054 or µPD78078.
;
;=======================================================================
;         The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;         The LCD and watch timer are run under the subsystem clock.
;         Normal operation is under the subsystem clock, but when a key is depressed,
;         the main system clock is activated.
;=======================================================================
;         for only DDB-K0070A
;
$set (DDB)
$set (D054)
;=======================================================================
```

```
;
DebugFlag set  1
;=========================================================================
;          Equates
;=========================================================================
;=========================================================================
;          saddr area
;          FWA = 0fe20h
;=========================================================================
           dseg       saddr
;
;          Time of day by watch timer mode
;
Interval_Seconds:   ds      1     ;Interval seconds

Seconds:            ds      1     ;Seconds
Minutes:            1             ;Minutes
Hours:              ds      1     ;Hours
Days:               ds      1     ;Days

KeyNumber:          ds      1     ;Key number

;=========================================================================
;         Even boundary pair
;=========================================================================
          dseg     saddrp

;=========================================================================
;         Bit segment
;=========================================================================
          bseg
HalfSecondFlag      dbit    ;1/2-second toggle flag

;=========================================================================
;
;         Internal High RAM allocation
;
          dseg     IHRAM
          ds       20h
Stack:

          $ej
;=========================================================================
;         Reset vector
;=========================================================================
Vector    cseg     at 0000h
          dw       ResetStart

          org      01Eh
          dw       Watch_timer    ; Watch timer interval time interrupt

          $ej
;=========================================================================
;         Main
;=========================================================================
Main      cseg     at 80h

          public   ResetStart
ResetStart:
          di                      ;Disable interrupt
;
;         Select register bank 0
;
          sel      RB0            ;Select register bank 0 for background
```

```
;
;        Oscillation mode select register
;         . Does not use divider circuit
;
         mov      OSMS,#00000001b
;
;        Set stack pointer (IHMEM + 20H)
;
;;       movw     SP,#Stack      ;Stack ptr
         movw     SP,#0fe00h     ;Stack ptr

;=====================================================================
;          Key return mode register (KRM)
;
;          0 0 0 0 0 1 0 0
;          | | | | | | | |__ KRM.0 (= 0, Not detected key return signal)
;          | | | | | | | |____ KRM.1 (= 1, Standby mode release enabled)
;          | | | | | | |______ KRM.2 (= 1, Select P114-P117 as key return signal port)
;          | | | | | |________ KRM.3 (= 0, Select P114-P117 as key return signal port)
;          | | | |___________ KRM.4 (= 0, Reserved)
;          | | |_____________ KRM.5 (= 0, Reserved)
;          | |_______________ KRM.6 (= 0, Reserved)
;          |_________________ KRM.7 (= 0, Reserved)
;
;          If µPD78054 or µPD78078, KRM.2 and KRM.3 cannot be defined.
;
;=====================================================================

           mov       KRM,#00000100b
; Set scan ports
           mov       PM3,#11110000B          ;P34-P37 output
           mov       P3,#00000000b           ;Set P34-P37 to low
; Set sense ports
$if (D054)
$else
           mov       PM11,#11110000B ;P114-P117 as input
$endif
;
;          Pull-up resistor option on P11
;
$if (D054)
           mov       PUOL,#00010000B
$else
           mov       PUOH,#00001000B
$endif

;=====================================================================
;          TCL2 - Timer clock select register 2
;
;          x x x 1 x x x x
;          | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;          | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;          | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;          | | | | |________ TCL2.3 (Reserved)
;          | | | |___________ TCL2.4 (= 1, Select 32.768-kHz subsystem clock)
;          | | |_____________ TCL2.5 (Buzzer output frequency selection)
;          | |_______________ TCL2.6 (Buzzer output frequency selection)
;          |_________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================
           mov       TCL2,#00010000b

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

```
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"
;
;=====================================================================
          mov       TMC2,#01010110b
;
;         Clear time variables
;
          mov       Seconds,#0
          mov       Minutes,#0
          mov       Hours,#0
          mov       Days,#0
;
;         Enable INTP0 for DDB-K0070A
;
$if (DDB)
          clr1      PMK0
$endif
;         Clear IRQ flags
          clr1      TMIF3
;         Enable watch timer internal timer
          clr1      TMMK3
          ei
;
main_loop:
;=====================================================================
;         Normal operation
;         Main system clock with subsystem clock operation
;=====================================================================
NormalMode:
;
;         Set to normal operation
;
          mov       PCC,#00000000B
;
;         Scan keyboard - every 15.6 ms will wake it up
;
          nop
          nop
          STOP
          nop
          call      !KeyScan
;         cf = 1 key detected; cf = 0 no key found
          bc        $ServiceKey
;
;         no key
;
          nop
          br        $main_loop
;
;         Key service modules
;
ServiceKey:
          nop
          mov       a,KeyNumber
```

```
;
;         Service
;
          br        $main_loop

;=====================================================================
;         Key scan module
;
;         The key matrix is 4 x 4. There are four scan ports (P34-P37) and
;         four key return signal ports (P114-P117).
;         Debounce count is set to 4.
;
;         Key number:
;         0 = P114 + P34
;         1 = P114 + P35
;         2 = P114 + P36
;         3 = P114 + P37
;         4 = P115 + P34
;         5 = P115 + P35
;         6 = P115 + P36
;         7 = P115 + P37
;         8 = P116 + P34
;         9 = P116 + P35
;         10 = P116 + P36
;         11 = P116 + P37
;         12 = P117 + P34
;         13 = P117 + P35
;         14 = P117 + P36
;         15 = P117 + P37
;
;         Return:
;                   cf = 1 if a key is depressed
;                     and KeyNumber with key number
;
;                   cf = 0 if no key is depressed
;
;         Registers:
;                   a = Scratch
;                   b = Total row or column numbers
;                   c = Sense port
;                   d = Row x 4 + column
;                   e = Column port
;                   h = Current key number
;                   l = Number of debounces
;                   x = Find a row
;=====================================================================
KeyScan:
;
;         Find a key
;
          mov       h,#0ffh                 ;no key flag
          mov       l,#3

KeyScanLoop:
          and       P3,#11110000b
$if (D054)
          mov       a,P4
$else
          mov       a,P11
$endif
          and       a,#11110000b            ;mask low nibble
          cmp       a,#11110000b            ;all high
          bz        $NotFound               ;no key depressed
;
```

```
          mov       c,a
          mov       d,#0                    ;key number
          mov       x,#00010000b            ;find a row
          mov       b,#4                    ;4 rows
CheckRow:
          mov       a,x
          and       a,c
          bz        $RowFind                ;find the row
; shift to left
          mov       a,x
          clr1      cy
          rol       a,1
          mov       x,a
; update row x column
          mov       a,d
          add       a,#4
          mov       d,a
;
          dbnz      b,$CheckRow
;
;         Not Found
;
NotFound:
          clr1      cy
          ret
;
;         Row find. Find column.
;         d = row x column
;         x = row position
;
RowFind:
          mov       e,#00000001b            ;find a column
          mov       b,#4                    ;4 columns
CheckCol:
          mov       a,e
          mov       p3,a                    ;set to high
$if (D054)
          mov       a,P4
$else
          mov       a,P11
$endif
          and       a,x
          bnz       $ColFind
; shift to left
          clr1      cy
          mov       a,e
          rol       a,1
          mov       e,a
;
;         Incr. key number
;
          inc       d
          dbnz      b,$CheckCol
          br        NotFound                ;not found
;
ColFind:
          mov       a,d
          cmp       a,h
          bz        $CheckDebounce           ; ok
;
;         Restart key scan
;
          mov       h,a
;
```

```
;          Decr. debounce count
;
CheckDebounce:
           dec        l
           bnz        $KeyScanLoop
;
KeyFound:
           mov        a,h
           mov        KeyNumber,a
           set1       cy
           ret

           $ej
;=========================================================================
;          Module name: INTTM3 watch timer interval time interrupt
;
;          Interrupted every 15.6 ms
;
;          DESCRIPTION:
;                     Set watch time variables
;
;          OPERATION:
;                     . Flip the HalfSecondFlag.
;                     . If HalfSecondFlag = 0, then incr. seconds.
;                     . If Seconds = 60, incr. minutes.
;                     . If Minutes = 60, incr. Hours.
;                     . If Hours  = 24, incr. days.
;
;=========================================================================
           public     Watch_timer
Watch_timer:
;
;          Test 0.5-second flag
;
           bf         WTIF,$wt20              ;not yet
           clr1       WTIF
;
;          0.5 second passed
;
           bf         HalfSecondFlag,$wt10 ;0.5 second passed
;          Clear      0.5-second flag
           clr1       HalfSecondFlag
;
           inc        Interval_Seconds

           inc        Seconds
           cmp        Seconds,#60
           bc         $wt20
           mov        Seconds,#0
;
           inc        Minutes
           cmp        Minutes,#60
           bc         $wt20
           mov        Minutes,#0
;
           inc        Hours
           cmp        Hours,#24
           bc         $wt20
           mov        Hours,#0
;          Day
           inc        Days
           cmp        Days,#30
           bc         $wt20
```

```
          br         $wt20
;
wt10:
          set1       HalfSecondFlag
wt20:
          reti
;=========================================================================
          end

#pragma interrupt INTTM3 WatchTimer RB1
```

C Language Program: Example of Key Matrix

```
/*;************************************************************************
;
;                                          File name: key.c
;                                          Date: 10/27/97
;************************************************************************
;         Description:
;
;
;The example program is a key input module that inputs signals from a 4 x 4 key matrix.
;The keys can be pressed successively, and two or more keys can
;be pressed simultaneously. In this example, the higher 4
;bits of port 3 (P34 - P37) are used as key scan signals, and port 11 (P4 for µPD78054 or µPD78078)
;is used for key return signals. As the pullup resistor of port 11 for key return, the
;internal pullup resistor set by software is used.
;
;Port 11 of the K0 series has a function to detect the falling edges of the
;eight port pins in parallel. If port 11 is used for key return signals,
;therefore, the standby mode can be released through detection of a falling
;edge by key input.
;
;The input keys are stored to RAM on a bit/key basis. The RAM bit
;corresponding to a pressed key is set and the bit corresponding to a released
;key is cleared. By testing the RAM data on a 1-bit basis starting
;from the first bit, the key status can be checked. To absorb key debouncing,
;the key is assumed to be valid when four successive key codes coincide with a
;given code. For example, if a key code is sampled every 16 ms, debouncing of
;48 ms to 64 ms can be absorbed.
;
; Key number:
;        0 = P114 + P34
;        1 = P114 + P35
;        2 = P114 + P36
;        3 = P114 + P37
;
;        4 = P115 + P34
;        5 = P115 + P35
;        6 = P115 + P36
;        7 = P115 + P37
;
;        8 = P116 + P34
;        9 = P116 + P35
;        10 = P116 + P36
;        11 = P116 + P37
;
;        12 = P117 + P34
;        13 = P117 + P35
;        14 = P117 + P36
;        15 = P117 + P37
;
;         Port 4 is used instead of port 11 for µPD78054 or µPD78078.
;
;=========================================================================
;         The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;         The LCD and watch timer are run under the subsystem clock.
;         Normal operation is under the subsystem clock, but when a key is depressed,
;         the main system clock is activated.
;=========================================================================*/
#include "define.h"
#define D0541        //for µPD78054 and µPD78078
//;
//;         Time of day by watch timer mode
//;
```

```
__sreg    unsigned char Seconds; //Seconds
__sreg    unsigned char Minutes; //Minutes
__sreg    unsigned char Hours;//Hours
__sreg    unsigned char Days;//Days
__sreg    unsigned char KeyNumber;
bit HalfSecondFlag; //1/2-second toggle flag
bit KeyOnFlag;  // key-on flag if it is 1
void KeyScan (void);
/*;=====================================================================
;         Main
;=====================================================================*/
void main ()
{
//;
//;         Oscillation mode select register
//;          . Does not use divider circuit
//;
          OSMS = 0b00000001;                // select 5 MHz
          PMK0 = 0;                         // for DDB Board - INTP0 has to be set

/*;=====================================================================
;         Key Return mode register (KRM)
;
;         0 0 0 0 0 1 0 0
;         | | | | | | | |__ KRM.0 (= 0, Not detected key return signal)
;         | | | | | | |____ KRM.1 (= 1, Standby mode release enabled)
;         | | | | | |______ KRM.2 (= 1, Select P114-P117 as key return signal port)
;         | | | | |________ KRM.3 (= 0, Select P114-P117 as key return signal port)
;         | | | |__________ KRM.4 (= 0, Reserved)
;         | | |____________ KRM.5 (= 0, Reserved)
;         | |______________ KRM.6 (= 0, Reserved)
;         |________________ KRM.7 (= 0, Reserved)
;
;         If µPD78054 or µPD78078, KRM.2 and KRM.3 cannot be defined.
;
;=====================================================================*/
          KRM = 0b00000100;
//; Set scan ports
          PM3 = 0b11110000; //P34-P37, output
          P3 = 0b00000000; //Set P34-P37 to low
//; Set sense ports
#if D054
#else
          PM11 = 0b11110000; //P114-P117 as input
#endif
//;
//;       Pullup resistor option on P11
//;
#if D054
          PUOL = 0b00010000;
#else
          PUOH = 0b00001000;
#endif
```

```
/*;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 1 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 1, Select 32.768-kHz subsystem clock
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================*/
          TCL2 = 0b00010000;

/*;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch Operating Mode Selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for Watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)
;
;=====================================================================*/
          TMC2 = 0b01010110;
//;
//;         Clear time variables
//;
          Seconds = 0;
          Minutes = 0;
          Hours = 0;
          Days = 0;
//;         Clear IRQ flags
          TMIF3 = 0;
//;         Enable watch timer interval timer
          TMMK3 = 0;

          EI ();

          while (TRUE)
          {
//;
//;         Set to normal operation
//;
                    PCC = 0B00000000;
//;
//;         Scan keyboard - every 15.6 ms will wake it up
//;
                    NOP();
                    NOP();
                    STOP();
                    NOP();
                    KeyScan();
                    if (KeyOnFlag)
                    {
                                        //
                                        // Service the key (user-defined function)
```

```
                                        //
                    }
          }
}
/*;=========================================================================
;         Key scan module
;
;         The key matrix is 4 x 4. There are four scan ports (P34-P37) and
;         four key return signal ports (P114-P117).
;         Debounce count is set to 4.
;
;         Key number:
;         0 = P114 + P34
;         1 = P114 + P35
;         2 = P114 + P36
;         3 = P114 + P37
;         4 = P115 + P34
;         5 = P115 + P35
;         6 = P115 + P36
;         7 = P115 + P37
;         8 = P116 + P34
;         9 = P116 + P35
;         10 = P116 + P36
;         11 = P116 + P37
;         12 = P117 + P34
;         13 = P117 + P35
;         14 = P117 + P36
;         15 = P117 + P37
;
;         Return:
;                   cf = 1 if a key is depressed
;                     and KeyNumber with key number
;
;                   cf = 0 if no key is depressed
;
;         Registers:
;                   a = Scratch
;                   b = Total row or column numbers
;                   c = Sense port
;                   d = Row x 4 + column
;                   e = Column port
;                   h = Current key number
;                   l = Number of debounces
;                   x = Find a row
;=========================================================================*/
void KeyScan (void)
{
#asm
$set (D054)
;
;         Find a key
;
          push      ax
          push      bc
          push      de
          push      hl
          clr1      _KeyOnFlag
          mov       h,#0ffh                 ;no key flag
          mov       l,#3

KeyScanLoop:
          and       P3,#11110000b
$if (D054)
          mov       a,P4
```

```
$else
          mov       a,P11
$endif
          and       a,#11110000b            ;mask low nibble
          cmp       a,#11110000b            ;all high
          bz        $NotFound               ;no key depressed
;
          mov       c,a
          mov       d,#0                    ;key number
          mov       x,#00010000b            ;find a row
          mov       b,#4                    ;4 rows
CheckRow:
          mov       a,x
          and       a,c
          bz        $RowFind                ;find the row
; shift to left
          mov       a,x
          clr1      cy
          rol       a,1
          mov       x,a
; update row * column
          mov       a,d
          add       a,#4
          mov       d,a
;
          dbnz      b,$CheckRow
;
;         Not Found
;
NotFound:
          push      hl
          push      de
          push      bc
          push      ax
          ret
;
;         Row find. Find column.
;         d = row x column
;         x = row position
;
RowFind:
          mov       e,#00000001b            ;find a column
          mov       b,#4                    ;4 columns
CheckCol:
          mov       a,e
          mov       p3,a                    ;set to high
$if (D054)
          mov       a,P4
$else
          mov       a,P11
$endif
          and       a,x
          bnz       $ColFind
; shift to left
          clr1      cy
          mov       a,e
          rol       a,1
          mov       e,a
;
;         Incr. key number
;
          inc       d
          dbnz      b,$CheckCol
          br        NotFound                ;not found
```

```
;
ColFind:
          mov       a,d
          cmp       a,h
          bz        $CheckDebounce          ; ok
;
;         Restart key scan
;
          mov       h,a
;
;         Decr. debounce count
;
CheckDebounce:
          dec       l
          bnz       $KeyScanLoop
;
KeyFound:
          mov       a,h
          mov       _KeyNumber,a
          set1      _KeyOnFlag
          push      hl
          push      de
          push      bc
          push      ax
          ret
#endasm
}
/*;=====================================================================
;         Module name: INTTM3 Watch Timer Interval Time Interrupt
;
;         Interrupted every 15.6 ms
;
;         DESCRIPTION:
;                   .Set watch time variables
;
;         OPERATION:
;                   . Flip the HalfSecondFlag.
;                   . If HalfSecondFlag = 0, then incr. seconds.
;                   . If seconds == 60, incr. minutes.
;                   . If minutes == 60, incr. hours.
;                   . If hours  == 24, incr. days.
;
;=====================================================================*/
void WatchTimer ()
{
//;
//;         Test 0.5-second flag
//;
          if ( !WTIF) return;
//;
          WTIF = 0;
//;
//;          0.5 second passed
//;
          if ( !HalfSecondFlag)
          {
                    HalfSecondFlag = 1;
                    return;;
          }
//;
          HalfSecondFlag = 0;
//;
          Seconds++;
          if (Seconds < 60) return;;
```

```
          Seconds = 0;
          Minutes++;
          if (Minutes < 60) return;;
          Minutes = 0;
          Hours++;
          if (Hours < 24) return;;
          Hours = 0;
          Days++;
}
```

NEC

Watch Timer Function 12

Applicable K0 Devices

- µPD7801xF
- µPD7801xH
- µPD7805x
- µPD7805xF
- µPD78005x
- µPD7807x
- µPD78070A
- µPD7806x
- µPD78030x
- µPD7804xH
- µPD7804xF
- µPD78020x

Description

The watch timer of the K0 family has a function that causes the timer to overflow every 0.5 or 0.25 second by using the main system clock (4.19 MHz) or subsystem clock (32.768 kHz) as the clock source. An interval timer function allows a user to set the interval time to check the watch timer overflow flag (WTIF).

The watch timer function is set by timer clock select register 2 (TCL2) and the watch timer mode control register (TMC2). Because there is no interrupt vector for a watch timer overflow condition, the WTIF is either polled constantly or checked from the INTTM3 (watch time interval time overflow interrupt) service routine. The watch timer function contains these registers:

- Timer clock select register 2 (Figure 12-1)
- Watch timer mode control register (Figure 12-2)

Figure 12-1. Timer Clock Select Register 2 (TCL2)

7	6	5	4	3	2	1	0
TCL27	TCL26	TCL25	TCL24	0	TCL22	TCL21	TCL20

Bit(s)	Name	Description
2 – 0	TCL22 – TCL20	Watchdog timer count clock select
4	TCL24	Watch timer count clock selection
		0 = $fx/2^7$ (39.1 kHz)
		1 = fxt (32.768 kHz)
7 – 5	TCL27 – TCL25	Buzzer output frequency selection

Figure 12-2. Watch Timer Mode Control Register (TMC2)

7	6	5	4	3	2	1	0
0	TMC26	TMC25	TMC24	TMC23	TMC22	TMC21	TMC20

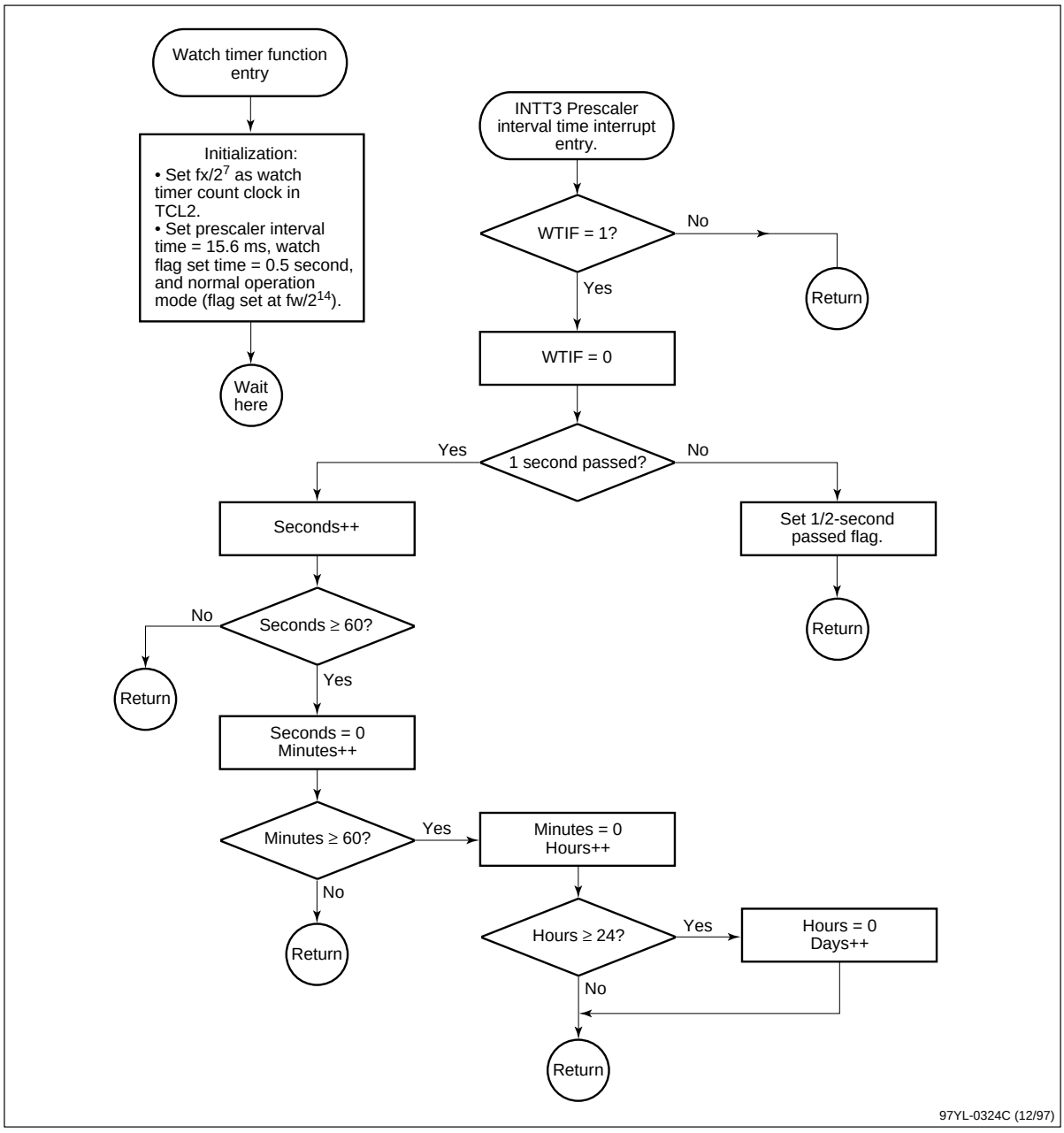
Bit(s)	Name	Description
0	TMC20	Operating mode
		0 = Normal (fw/2^{14})
		1 = Fast feed (fw/2^5)
1	TMC21	Prescaler
		0 = Clear after operation stop
		1 = Operation enable
2	TMC22	5-bit counter
		0 = Clear
		1 = Enable
3	TMC23	WTIF flag set
		0 = 2^{14}/fw (0.5 sec)
		1 = 2^{13}/fw (0.25 sec)
6–4	TMC26–TMC24	Prescaler interval timer selection at fxx = 4.19 MHz
		0 0 0 = 2^4/fw (488 µs)
		0 0 1 = 2^5/fw (977 µs)
		0 1 0 = 2^6/fw (1.95 ms)
		0 1 1 = 2^7/fw (3.91 ms)
		1 0 0 = 2^8/fw (7.82 ms)
		1 0 1 = 2^9/fw (15.6 ms)
		Other = prohibited

Example Program

The watch timer example program demonstrates the watch timer function in the K0 family. Figure 12-3 is the program flowchart. The timer clock select register (TCL2) is set to fx/2^7 (39.1 kHz) as the watch timer count clock selection. The watch timer mode control register (TMC2) is set to 15.6-ms prescaler interval time, the WTIF flag to 0.5 second, and the operating mode to normal.

When the interval timer interrupt (INTTM3) occurs, the interrupt service routine checks the WTIF (0.5-second overflow flag). If WTIF is set, flag and time values such as second, minute, hour, and day are updated.

Figure 12-3. Flowchart of Watch Timer Example Program



Assembly Language Program: Watch Timer Example

```
;*************************************************************************
;                                          File name: TOD.ASM
;                                          Date: 10/15/97
;*************************************************************************
;         Description:
;
; This program demonstrates the watch timer in the µPD78308 microcontroller.
; The watch timer control registers (TCL2 and TMC2) are used to configure
; the watch timer function.
;
; The time base resolution of the watch timer is selectable as 0.25
; or 0.5 second when the 4.19 MHz (standard: 4.194304 MHz) main system clock is
; used.
;
; Since there is no interrupt vector for the 0.5-second interval, the watch
; timer's interval timer interrupt (INTT3) should be used to check whether
; 0.5 second has passed.
;=========================================================================
; The sample program will set minute, hour, and day with the 0.5-second time base.
; The interval time is set to 15.6 ms. The interval time interrupt service
; routine is used to check the 0.5-second flag (WTIF).
;=========================================================================
;         for only DDB-K0070A
;
$set (DDB)
$set (D054)
;=========================================================================
;
DebugFlag  set   1
;=========================================================================
;         Equates
;=========================================================================
;=========================================================================
;         saddr Area
;         FWA = 0fe20h
;=========================================================================
          dseg      saddr
;
;         Time of day by watch timer mode
;
Seconds:  ds        1                     ;Seconds
Minutes:  ds        1                     ;Minutes
Hours:    ds        1                     ;Hours
Days:     ds        1                     ;Days

;=========================================================================
;         Even boundary pair
;=========================================================================
          dseg      saddrp

;=========================================================================
;         Bit segment
;=========================================================================
          bseg
HalfSecondFlag          dbit      ;1/2-second toggle flag

;=========================================================================
;
;         Internal High RAM allocation
;
          dseg      IHRAM
          ds        20h
```

```
Stack:

          $ej
;=========================================================================
;         Reset vector
;=========================================================================
Vector    cseg      at 0000h
          dw        ResetStart

          org       01Eh
          dw        Watch_timer             ; Watch timer interval time interrupt

          $ej
;=========================================================================
;         Main
;=========================================================================
Main      cseg      at 80h

          public    ResetStart
ResetStart:
          di                                ;Disable interrupt
;
;         Select register bank 0
;         sel       RB0                     ;Select register bank 0 for background
;
;         Oscillation mode select register
;          . Does not use divider circuit
;
          $if (D054)
          mov       OSMS,#00000001b
          $endif
;
;         Set stack pointer (IHMEM + 20H)
;
;;        movw      SP,#Stack               ;Stack pointer
          movw      SP,#0fe00h              ;Stack pointer
;
;         Init. Watch timer control register
;
;;        Call      !InitWatchTimer
;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 0 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 0, Select 4.19 MHz/2^7)
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================
          mov       TCL2,#00000000b
;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
```

```
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch Operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)
;
;=====================================================================
          mov       TMC2,#01010110b
;
;         Clear time variables
;
          mov       Seconds,#0
          mov       Minutes,#0
          mov       Hours,#0
          mov       Days,#0
;
;         Enable INTP0 for DDB-K0070A
;
$if (DDB)
          clr1      PMK0
$endif
;         Clear IRQ flags
          clr1      TMIF3
;         Enable watch timer internal timer
          clr1      TMMK3

          ei
;
main_loop:
          nop
          br        $main_loop ;string received flag = 0


          $ej
;=====================================================================
;         Module name: INTTM3 wWtch timer interval time interrupt
;
;         DESCRIPTION:
;                   Set watch time variables
;
;         OPERATION:
;                   . Flip the HalfSecondFlag.
;                   . If HalfSecondFlag = 0, then incr. seconds.
;                   . If seconds = 60, incr. minutes.
;                   . If minutes = 60, incr. hours.
;                   . If hours   = 24, incr. days.
;
;=====================================================================
          public    Watch_timer
Watch_timer:
;
;         Test 0.5-second flag
;
          bf        WTIF,$wt20              ;not yet
          clr1      WTIF
;
;         0.5 second passed
;
          bf        HalfSecondFlag,$wt10  ;1/2 second passed
;         Clear half-second flag
```

```
          clr1      HalfSecondFlag
;
          inc       Seconds
          cmp       Seconds,#60
          bc        $wt20
          mov       Seconds,#0
;
          inc       Minutes
          cmp       Minutes,#60
          bc        $wt20
          mov       Minutes,#0
;
          inc       Hours
          cmp       Hours,#24
          bc        $wt20
          mov       Hours,#0
;         Day
          inc       Days
          cmp       Days,#30
          bc        $wt20

          br        $wt20
;
wt10:
          set1      HalfSecondFlag
wt20:
          reti

;=========================================================================
          end
_
```

C Language Program: Watch Timer Example

```
#pragma interrupt INTTM3 WatchTimer RB1

/*;****************************************************************************
;                                          File name: TOD.C
;                                          Date: 10/15/97
;****************************************************************************
;         Description:
;
; This program demonstrates the watch timer in the µPD78308 microcontroller.
; The watch timer control registers (TCL2 and TMC2) are used to configurate
; the watch timer function.
;
; The time base resolution of the watch timer is selectable as 0.25
; or 0.5 seconds when the 4.19 MHz (standard: 4.194304 MHz) main system clock is
; used.
;
; Since there is no interrupt vector for the 0.5-second interval, the watch
; timer's interval time interrupt (INTT3) should be used to check whether
; 0.5 second has passed.
;============================================================================
; The sample program will set minute, hour and day with the 0.5-second time base.
; The interval time is set to 15.6 ms. The interval timer is used to
; check 0.5-second flag, (WTIF).
;============================================================================*/
#include "define.h"
//;
//;          Time of day by watch timer mode
//;
__sreg      unsigned char Seconds; //Seconds
__sreg      unsigned char Minutes; //Minutes
__sreg      unsigned char Hours;          //Hours
__sreg      unsigned char Days;           //Days

bit HalfSecondFlag;                       //1/2-second toggle flag


//;=========================================================================
//;         Main
//;=========================================================================
void main()
{
//;
//;         Oscillation mode select register
//;           . Does not use divider circuit
//;
          OSMS = 0b00000001;              // select 5 MHz
          PMK0 = 0;                       // for DDB BOARD - INTP0 has to be set
/* ;=======================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 0 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 0, Select 4.19 MHz/2^7)
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=======================================================================*/
          TCL2 = 0b00000000;
```

```
/*;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4(= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | | |____________ TMC2.5(= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6(= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7(Reserved = 0)"                          )
;
;=====================================================================*/
          TMC2 = 0b01010110;
//;
//;         Clear time variables
//;
          Seconds = 0;
          Minutes = 0;
          Hours = 0;
          Days = 0;
//;         Clear IRQ flags
          TMIF3 = 0;
//;         Enable watch timer interval timer
          TMMK3 = 0;

          EI();
//;
          while (TRUE);
}
/*;=====================================================================
;       Module name: INTTM3 watch timer interval time interrupt
;
;         DESCRIPTION:
;                   Set watch time variables
;
;         OPERATION:
;                   . Flip the HalfSecondFlag.
;                   . If HalfSecondFlag = 0, then incr. seconds.
;                   . If seconds = 60, incr. minutes.
;                   . If minutes = 60, incr. hours.
;                   . If hours   = 24, incr. days.
;
;=====================================================================*/
void WatchTimer()
{
//;
//;         Test 0.5-second flag
//;
          if ( !WTIF) return;
//;
          WTIF = 0;
//;
//;         0.5 second passed
//;
          if ( !HalfSecondFlag)
          {
                    HalfSecondFlag = 1;
                    return;;
          }

//;
```

```
            HalfSecondFlag = 0;
//;
            Seconds++;
            if (Seconds < 60) return;;

            Seconds = 0;
            Minutes++;
            if (Minutes < 60) return;;

            Minutes = 0;
            Hours++;
            if (Hours < 24) return;;
            Hours = 0;
            Days+;
}
```

NEC

Standby Function 13

Applicable Devices

- μPD7801xF
- μPD7801xH
- μPD7805x
- μPD7805xF
- μPD78005x
- μPD7807x
- μPD78070A
- μPD7806x
- μPD78030x
- μPD7804xH
- μPD7804xF
- μPD78020xl

Description

The standby function is designed to decrease system power consumption. The microcontroller has two power-saving modes: HALT and STOP.

HALT Mode

The HALT mode is intended to stop the CPU operation clock while the system clock continues oscillation. The HALT mode is valid to restart immediately upon interrupt request and to carry out intermittent operations such as in watch applications. The HALT instruction can be used with either the main system clock or the subsystem clock.

HALT Mode Settings and Operation Status

Parameter	Main System Clock	Subsystem Clock
Clock generator	Both main and subsystem clocks can be in oscillation.	
CPU	Operation stops	
Port	Status before HALT mode setting is held	
16-bit timer/event counter	Operable	Operable when watch counter timer output with fxt is selected as count clock
8-bit timer/event counter	Operable	Operable when TI1 and TI2 are selected as count clock
Watch timer	Operable if $fxx/2^7$ selected as count clock	Operable if fxt selected as count clock
Watchdog timer	Operable if $fxx/2^7$ selected as count clock	Operation stops
A/D converter	Operable	Operation stops
Serial interface	Operable	Operable at external SCK
LCD controller/driver	Operable if $fxx/2^7$ selected as count clock	Operable if fxt selected as count clock
INTP0	Operable when a clock ($fxx/2^5$, $fxx/2^6$, $fxx/2^7$) for the peripheral hardware is selected as sampling clock	Operation stops
INTP1 to INTP5	Operable	

There are four ways to clear HALT mode:

1. Clear upon unmasked interrupt request.
2. Clear upon non-maskable interrupt request.
3. Clear upon unmasked test input.
4. Clear upon RESET input.

STOP Mode

When the STOP instruction is executed, the main system clock oscillator stops. In the STOP mode, CPU current consumption can be considerably decreased. Data memory low-voltage hold (down to V_{DD} = 1.8 V) is possible. Thus, the STOP mode is effective to hold data memory contents with ultra-low current consumption. Because this mode can be cleared upon interrupt request, it enables intermittent operations to be carried out. However, because a wait time is necessary to secure oscillator stabilization after the STOP mode is cleared, select the HALT mode if it is necessary to start processing immediately upon interrupt request.

The STOP mode can be used only when the system operates with the main system clock (subsystem clock oscillation cannot be stopped). When proceeding to the STOP mode, be sure to stop the peripheral hardware operation and execute the STOP instruction.

STOP Mode Settings and Operation Status

Parameter	With Subsystem Clock	Without Subsystem Clock
Clock generator	Only main system clock stops oscillation	
CPU	Operation stops	Operation stops
Port	Status before STOP mode setting is held	
16-bit timer/event counter	Operable when watch timer output with fxt is selected as count clock	Operation stops
8-bit timer/event counter	Operable when TI1 and TI2 are selected as count clock	
Watch timer	Operable if fxt is selected as count clock	
Watchdog timer	Operation stops	Operation stops
A/D converter	Operation stops	Operation stop.
Serial interface	Operable at external SCK	Operable at external SCK
LCD controller/driver	Operable if Fxt is selected as count clock	Operation stops
INTP0	Operation impossible	
INTP1 to INTP5	Operable	

There are two ways to clear the STOP mode:

1. Release by unmasked interrupt request
2. Release by RESET input

In any mode, the contents of the register, flag, and data memory just before standby mode setting are retained. The input/output port output latch and output buffer status are also held.

Standby Function Control Register

After the STOP mode is cleared upon interrupt request, wait time is controlled with the oscillation stabilization time select register (OSTS). The default stabilization time is 26.2 ms (MCS = 1). The wait time after STOP mode clear does not include the time from STOP mode clear to clock oscillation start, regardless of clearance by the RESET input or by interrupt generation.

Application Program

This program demonstrates standby functions HALT and STOP mode in the K0 family. Refer to the control register (Figure 13-1) and the flowchart (Figure 13-2). To measure power consumption for about 30 seconds in each mode, the program runs as follows:

1. Normal mode operation with system main clock (5.00 MHz) and subsystem clock
2. HALT mode operation with system main clock (5.00 MHz) and subsystem clock
3. STOP mode operation with subsystem clock (32.768 kHz)

Each mode runs with a toggle port to indicate the current mode of operation.

1. P1.0 = Normal mode
2. P1.1 = HALT mode
3. P1.2 = STOP mode

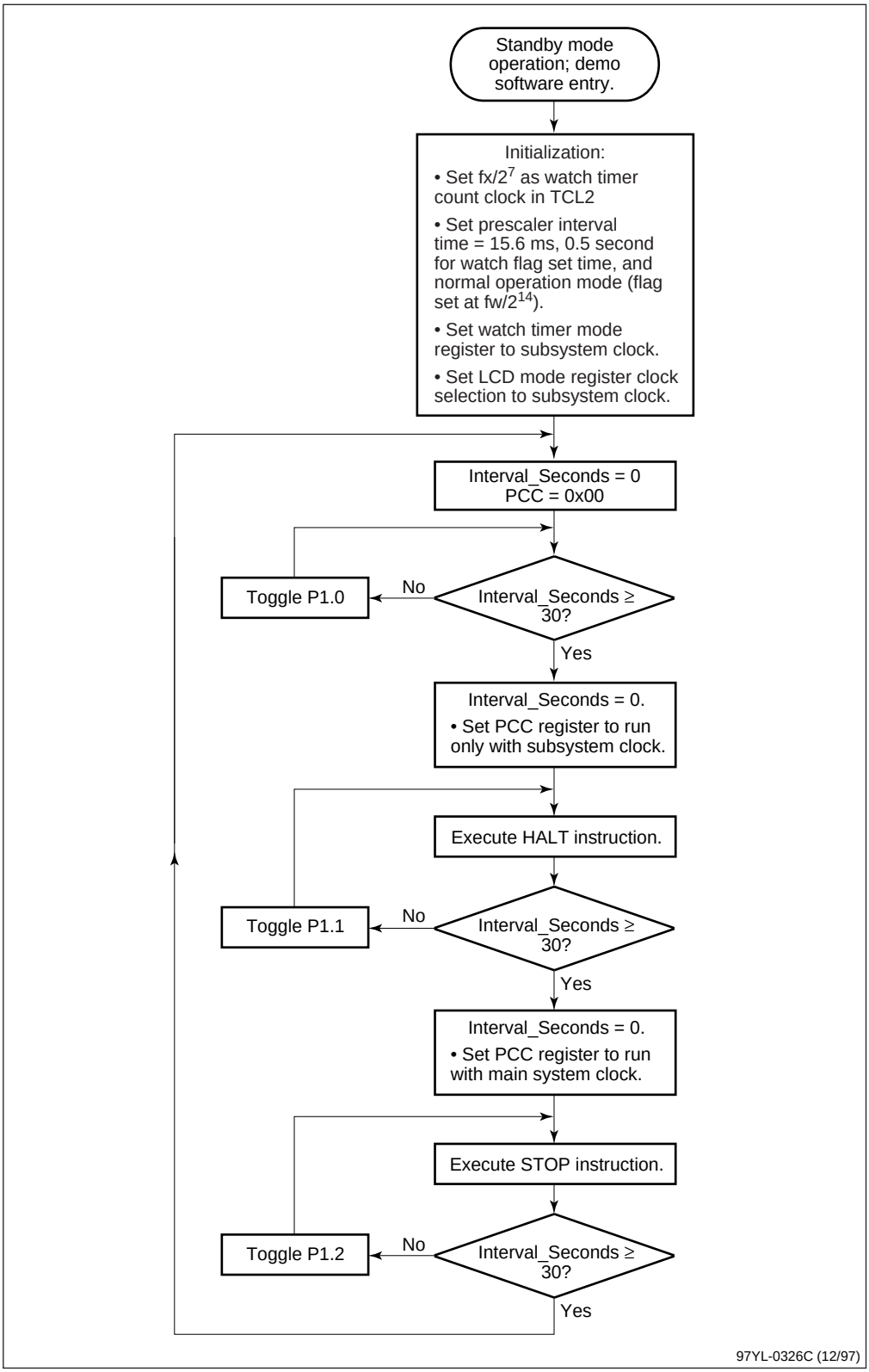
To run the STOP mode, a subsystem clock (32.768 kHz) must be installed.

Figure 13-1. Processor Clock Control Register (PCC) Setup for Each Mode

7	6	5	4	3	2	1	0
MCC	FRC	CLS	CSS	0	PCC2	PCC1	PCC0

Bit(s)	Name	Description
2–0	PCC2–PCC0	CPU clock
		0 0 0 = fx (0.4 µs)
4	CSS	CPU clock
		0 = fx (Normal, STOP)
		1 = fxt (HALT)
5	CLS	Read CPU clock status
		0 = Main system clock
6	FRC	Feedback resistor
		0 = Internal
7	MCC	Main system clock
		0 = Oscillation possible (Normal, STOP)
		1 = Oscillation stopped (HALT)

Figure 13-2. Flowchart of the Program Example of Standby Modes



Assembly Language Program: Example of Standby Modes

```
;*************************************************************************
;                                       File name: STANDBY.ASM
;                                       Date: 10/17/97
;*************************************************************************
;         Description:
;
; This program  demonstrates the standby HALT and STOP modes
; in the K0 family.
;
; The program is set in three modes to check power consumption:
;         1. Normal operation with system main clock (5 MHz) for about 30 seconds.
;         2. HALT mode operation with system main clock (5 MHz) for about 30 seconds.
;         3. STOP mode operation with subsystem clock (32.768 kHz) for about 30 seconds.
; Each time the mode is switched, toggle a port associated with
; the mode.
;                   P1.0 = Normal mode
;                   P1.1 = HALT mode
;                   P1.2 = STOP mode
;
;         By toggling the port, the user knows which mode is currently set to measure
;         power consumption.
;         To run the STOP mode, the subsystem clock (32.768 kHz) must be
;         operational.
;
;=======================================================================
;         for only DDB-K0070A
;
$set(DDB)
$set(D054)
;=======================================================================
;
DebugFlag  set   1
;=======================================================================
;         Equates
;=======================================================================
;=======================================================================
;         saddr area
;         FWA = 0fe20h
;=======================================================================
          dseg      saddr
;
;         Time of day by watch timer mode
;
Interval_Seconds:    ds                   1             ;Interval Seconds

Seconds:             ds                   1             ;Seconds
Minutes:             ds                   1             ;Minutes
Hours:               ds                   1             ;Hours
Days:                ds                   1             ;Days

;=======================================================================
;         Even boundary pair
;=======================================================================
          dseg      saddrp

;=======================================================================
;         Bit segment
;=======================================================================
          bseg
HalfSecondFlag       dbit                 ;1/2-second toggle flag

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

```
;          Internal High RAM allocation
;
           dseg      IHRAM
           ds        20h
Stack:

           $ej
;=========================================================================
;          Reset vector
;=========================================================================
Vector     cseg      at 0000h
           dw        ResetStart

           org       01Eh
           dw        Watch_timer             ; Watch timer interval time interrupt

           $ej
;=========================================================================
;          Main
;=========================================================================
Main       cseg      at 80h

           public    ResetStart
ResetStart:
           di                                ;Disable interrupt
;
;          Select register bank 0
;
           sel       RB0                     ;Select register bank 0 for background
;
;          Oscillation mode select register
;           . Does not use divider circuit
;
           mov       OSMS,#00000001b
;
;          Set stack pointer(IHMEM + 20H)
;
;;         movw      SP,#Stack               ;Stack ptr
           movw      SP,#0fe00h              ;Stack ptr
;
;          Init. watch timer control register
;
;;         Call      !InitWatchTimer
;=========================================================================
;          TCL2 - Timer clock select register 2
;
;          x x x 1 x x x x
;          | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;          | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;          | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;          | | | | |________ TCL2.3 (Reserved)
;          | | | |__________ TCL2.4 (= 1, Select 32.768 kHz subsystem clock)
;          | | |____________ TCL2.5 (Buzzer output frequency selection)
;          | |______________ TCL2.6 (Buzzer output frequency selection)
;          |________________ TCL2.7 (Buzzer output frequency selection)
;
;=========================================================================
           mov       TCL2,#00010000b
;=========================================================================
;
;          TMC2 - Watch timer mode control register
;
```

```
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control-enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"                        )
;
;=====================================================================
          mov       TMC2,#01010110b
;
;         Clear time variables
;
          mov       Seconds,#0
          mov       Minutes,#0
          mov       Hours,#0
          mov       Days,#0
;
;         Enable INTP0 for DDB-K0070A
;
$if(DDB)
          clr1      PMK0
$endif
;         Clear IRQ flags
          clr1      TMIF3
;         Enable watch timer internal timer
          clr1      TMMK3
;
;         SET P1 to all output and low
;
          mov       PM1,#0
          MOV       P1,#0

          ei
;
main_loop:
;=====================================================================
;         Normal operation
;         Main system clock with subsystem clock operation
;=====================================================================
NormalMode:
          mov       PCC,#00000000B
          mov       Interval_Seconds,#00
;
NormalModeLoop:
          cmp       Interval_Seconds,#30
          bnc       $HaltMode               ;run for HALT mode test
;         Toggle p1.0
          mov       P1,#001b
          nop
          mov       P1,#000b
          br        NormalModeLoop
;=====================================================================
;         HALT operation
;         Running with only subsystem clock operation
;=====================================================================
HaltMode:
          mov       PCC,#10010000B
          mov       Interval_Seconds,#00
;
HaltModeLoop:
          cmp       Interval_Seconds,#30
```

```
          bnc       $StopMode
; halt
          HALT
          nop
          nop
; Toggle p1.1
          mov       P1,#010b
          nop
          mov       P1,#000b
          br        HaltModeLoop
;=========================================================================
;         STOP operation
;         Main system clock stopped and with subsystem clock operation
;=========================================================================
StopMode:
          mov       PCC,#00000000B
          mov       Interval_Seconds,#00
; halt
StopModeLoop:
          cmp       Interval_Seconds,#30
          bnc       $Done                   ;done

          STOP
          nop
          nop
;
          mov       P1,#100b
          nop
          mov       P1,#000b
          br        StopModeLoop
;
;         Loop back to top
;
Done:
          br        $main_loop


          $ej
;=========================================================================
;         Module name: INTTM3 Watch timer Interval Time Interrupt
;
;         DESCRIPTION:
;                   Set watch time variables
;
;         OPERATION:
;                   . Flip the HalfSecondFlag
;                   . If HalfSecondFlag = 0, then incr. seconds.
;                   . If seconds = 60, incr. minutes.
;                   . If minutes = 60, incr. hours.
;                   . If hours   = 24, incr. days.
;
;=========================================================================
          public    Watch_timer
Watch_timer:
;
;         Test 0.5-second flag
;
          bf        WTIF,$wt20              ;not yet
          clr1      WTIF
;
;         .5 second passed
;
          bf        HalfSecondFlag,$wt10    ;1/2 second passed
;         Clear half-second flag
```

```
          clr1      HalfSecondFlag
;
          inc       Interval_Seconds

          inc       Seconds
          cmp       Seconds,#60
          bc        $wt20
          mov       Seconds,#0
;
          inc       Minutes
          cmp       Minutes,#60
          bc        $wt20
          mov       Minutes,#0
;
          inc       Hours
          cmp       Hours,#24
          bc        $wt20
          mov       Hours,#0
;         Day
          inc       Days
          cmp       Days,#30
          bc        $wt20

          br        $wt20
;
wt10:
          set1      HalfSecondFlag
wt20:
          reti

;=====================================================================
          end
_
```

C Language Program: Example of Standby Modes

```
#pragma interrupt INTTM3 WatchTimer RB1

/*;****************************************************************************
;                                          File name: STANDBY.C
;                                          Date: 10/15/97
;****************************************************************************
;         Description:
;
; This program demonstrates the standby HALT and STOP modes
; in the K0 family.
;
; The program is set in three modes to check power consumption.
;         1. Normal operation with system main clock (5 MHz) for about 30 seconds.
;         2. HALT mode operation with system main clock (5 MHz) for about 30 seconds.
;         3. STOP mode operation with subsystem clock (32.768 kHz) for about 30 seconds.
; Each time the mode is switched, toggle a port associated with
; the mode.
;                   P1.0 = Normal mode
;                   P1.1 = HALT mode
;                   P1.2 = STOP mode
;
;         By toggling the port, the user knows which mode is currently set to measure
;         power consumption.
;         To run the STOP mode, the subsystem clock (32.768 kHz) must be
;         operational.
;


;=============================================================================
;
; The sample program will set minute, hour, and day with a 0.5-second time base.
; The interval time is set to 15.6 ms. The interval timer is used to
; check the 0.5-second flag, with WTIF set. If so, increment the 0.5 second count.
;
;=============================================================================*/
#include "define.h"
//;
//;          Time of day by watch timer mode
//;
__sreg     unsigned char Interval_Seconds;                 //Seconds
__sreg     unsigned char Seconds;                          //Seconds
__sreg     unsigned char Minutes;                          //Minutes
__sreg     unsigned char Hours;                            //Hours
__sreg     unsigned char Days;                             //Days

bit HalfSecondFlag;
//1/2-second toggle flag


//;=============================================================================
//;          Main
//;=============================================================================
void main()
{
//;
//;          Oscillation mode select register
//;            . Does not use divider circuit
//;
          OSMS = 0b00000001;// select 5 MHz
          PMK0 = 0;                         // for DDB BOARD - INTP0 has to be set
/* ;=================================================================
```

```
;         TCL2 - Timer clock select register 2
;
;         x x x 1 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved))
;         | | | |__________ TCL2.4 (= 1, Select 32.768 kHz subsystem clock
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================*/
          TCL2 = 0b00010000;
/*;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch operating mode selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms, fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"                        )
;
;=====================================================================*/
          TMC2 = 0b01010110;
//;
//;         Clear time variables
//;
          Seconds=0;
          Minutes=0;
          Hours=0;
          Days= 0;
//;         Clear IRQ flags
          TMIF3 = 0;
//;         Enable watch timer interval timer
          TMMK3 = 0;

//;
          PM1 = 0X00;
          P1 = 0X00;

          EI();
 while(TRUE)
 {
//
//        Normal mode
//        Main system clock with subsystem clock operation
//
          PCC = 0b00000000;
          Interval_Seconds = 0;
          while (Interval_Seconds < 30)
          {
                    P1.0 = 1;
                    P1.0 = 0;
          }
//
//        Halt Mode
//        Set to run only with subsystem clock operation
//
```

```
          PCC = 0b10010000;
          Interval_Seconds = 0;
          while(Interval_Seconds < 30)
          {
                    HALT();
                    NOP();
                    NOP();
                    P1.1 = 1;
                    P1.1 = 0;
          }
//
//        STOP mode
//        Main system clock stopped with subsystem clock operation
//
          PCC = 0b00000000;
          Interval_Seconds = 0;
          while(Interval_Seconds < 30)
          {
                    STOP();
                    NOP();
                    NOP();
                    P1.2 = 1;
                    P1.2 = 0;
          }
 }

}
/*;=====================================================================
;         Module name: INTTM3 watch timer interval time interrupt
;
;         DESCRIPTION:
;                   Set watch time variables
;
;         OPERATION:
;                   . Flip the HalfSecondFlag.
;                   . If HalfSecondFlag = 0, then incr. seconds.
;                   . If seconds == 60, incr. minutes.
;                   . If minutes == 60, incr. hours.
;                   . If hours   == 24, incr. days.
;
;=====================================================================*/
void WatchTimer()
{
//;
//;         Test 0.5-second flag
//;
          if (!WTIF) return;
//;
          WTIF = 0;
//;
//;         0.5 second passed
//;
          if (!HalfSecondFlag)
          {
                    HalfSecondFlag=1;
                    return;;
          }

//;
          HalfSecondFlag=0;
//;
          Interval_Seconds++;

          Seconds++;
```

```
            if (Seconds < 60) return;;

            Seconds = 0;
            Minutes++;
            if (Minutes < 60) return;;

            Minutes = 0;
            Hours++;
            if (Hours < 24) return;;

            Hours = 0;
            Days++;
}
_
```

NEC

Software for the LCD Demo Application 14

Description

This program, written for an imaginary application, consists of four modules, each demonstrating one of the following functions in the µPD78P0308 and µPD78F9418 microcontrollers.

- LCD display function
- Keyboard function
- Time-of-day function
- Power-saving mode

The demo program displays the message “NEC DEMO” on the LCD when the demo PC board is powered. The message clears after about 3 seconds, and the time-of-day is displayed (default time is 12:00). The time is updated every minute. Figure 14-1 is the program flowchart.

Edit the Time-of-Day

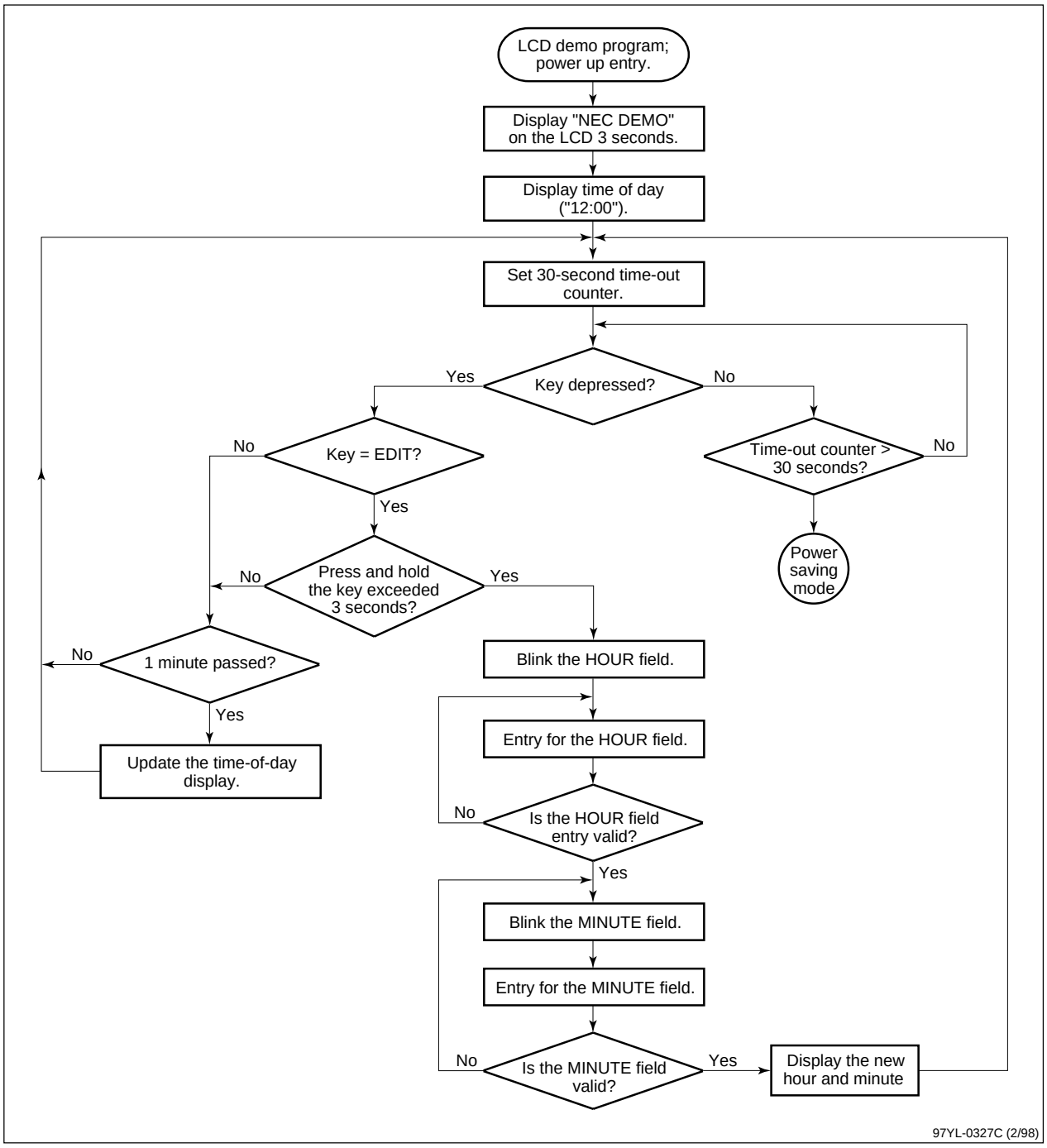
To change the time-of-day, press and hold the EDIT button for about 3 seconds until the hour field blinks. Type in the new hour and press <ENTER>. If the entry is invalid, it will be rejected and a valid number must be entered.

Once the hour field is set, the minute field blinks. Type in the new minute and press <ENTER>. If the entry is invalid, a valid number must be entered. When the hour and minute have been entered successfully, the new time-of-day will be displayed.

Power-Saving Mode

If no key is pressed, after about 30 seconds the system will go into the power-saving mode. The system will be awakened from the power-saving mode when any key is pressed.

Figure 14-1. Flowchart of the Demo Program



Assembly Language Program: DEMO_CLK.ASM

```
;*********************************************************************
;
;               File name: DEMO_CLK.ASM
;               Date: 1/29/98
;*********************************************************************
;        Description:
;
;        This program is to demonstrate the following functions in the K0/K0S family:
;                LCD display function
;                Keyboard function
;                Time-of-day function
;                Power down mode (standby mode)
;        Each module is written in a file that can be used as an application
;        note.
;
;=====================================================================
;        The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock and when a key is depressed, the
;        main system clock is activated.
;=====================================================================
;        for only DDB-K0070A
;
$set(DDB)

;=====================================================================
;        Extern reference
;=====================================================================

         EXTRN  DIGIT_0
         EXTRN  DIGIT_1
         EXTRN  DIGIT_2
         EXTRN  DIGIT_3
         EXTRN  DIGIT_4
         EXTRN  DIGIT_5
         EXTRN  DIGIT_6
         EXTRN  DIGIT_7
         EXTRN  DIGIT_8
         EXTRN  DIGIT_9
         EXTRN  KEY_SET
         EXTRN  KEY_ENTER
         EXTRN  KEY_CLEAR
         EXTRN  KEY_STOP
         EXTRN  KEY_SPARE1
         EXTRN  KEY_SPARE2

         extbit ASecondFlag

         extrn  Hours
         extrn  Interval_timer
         extrn  KeyNumber
         extrn  Minutes
         extrn  Seconds

         extrn  ClearLCDDisplay
         extrn  DisplayTOD
         extrn  DisplayAString
         extrn  Init_LCD
         extrn  Init_Key
         extrn  Init_TOD
         extrn  KeyScan
```

```
;=========================================================================
;         bit Area
;         FWA = 0fe20h
;=========================================================================

          bseg
DoneFlag  dbit   ;done

;=========================================================================
;         saddr Area
;         FWA = 0fe20h
;=========================================================================

          dseg   saddr

          public Curr_Hours
          public Curr_Minutes
          public Curr_Field_Number

Curr_Hours:ds   1
Curr_Minutes:ds 1

Curr_Field_Number: ds 1;Field number

HOUR_FieldEQU   0
MINUTE_FieldEQU 1
SECOND_FieldEQU 2

;=========================================================================
;         Even Boundary Pair
;=========================================================================
          dseg   saddrp

Curr_Field_Ptr: ds 2;Current working field ptr

;=========================================================================
;
;Internal High RAM allocation
;
          dseg   IHRAM
          ds     20h
Stack:
          $ej
;=========================================================================
;         Reset Vector
;=========================================================================
Vector    cseg   at 0000h
          dw        ResetStart

          org       01Eh
          extrn     Watch_timer
          dw        Watch_timer; Watch timer interval time interrupt

          $ej
;=========================================================================
;         MAIN
;=========================================================================
Main      cseg      at 80h

          public    ResetStart
ResetStart:
          di           ;Disable interrupt
;
;         Select Register Bank 0
```

```
;
          sel         RB0           ;Select register bank 0 for background
;
;         Oscillation mode select register
;          . Does not use divider circuit
;
          mov    OSMS,#00000001b
;
;         Set stack pointer (IHMEM + 20H)
;
;;        movw        SP,#Stack     ;Stack ptr
          movw        SP,#0fe00h    ;Stack ptr
;
;         Init.
;
          call        !Init_Key
          call        !Init_TOD
          call        !Init_LCD
;
;         Enable INTP0 for DDB-K0070A
;
$if(DDB)
          clr1        PMK0
$endif
;         Clear IRQ flags
          clr1        TMIF3
;         Enable Watch Timer Internal Timer
          clr1        TMMK3
          ei
;=====================================================================
;         Display current time of day
;         Default: 12:00:00
;=====================================================================
;
;         Set to normal operation
;
          mov         PCC,#00000000B
;
;         Display a greeting message
;         'NEC DEMO'
;
          movw        de,#NEC_DEMO
          call        !DisplayAString
Wait5sec:
          cmp         Seconds,#5
          bnz         $Wait5sec
          mov         Seconds,#0
;
;         Display Default Time of day(12:00:00)
          call        !ClearLCDDisplay
          call        !DisplayTOD
          clr1        ASecondFlag
;
;         Go to STANDBY Mode and if wake up then
;         Check Second on
;
main_loop:
;
;         a second passed?
;
          bf          ASecondFlag,$main10
          clr1        ASecondFlag
;
;         Display Update
```

```
;
          call        !DisplayTOD

;
;         Set to Normal Operation
;
main10:
;
;         Scan keyboard - every 15.6 ms will wake it up
;
          nop
          nop
          STOP
          nop
;
;         Scan keys
;
          call        !KeyScan
;         cf = 1 key detected cf = 0 no key found
          bnc         $main_loop
;
;         call the Key service module
;
          call        !ServiceKey
          br          main_loop
;
;         Greeting message
;
NEC_DEMO:
          db          'NEC_DEMO',0

;=====================================================================
;         Service a key
;
;         Input: KeyNumber
;
;
;=====================================================================
          public      ServiceKey
ServiceKey:
          mov         a,KeyNumber
          cmp         a,#KEY_SET
          bz          $DoSetTOD
;         Return
          ret

;=====================================================================
;         Set time of day.
;         Clear the LCD display.
;         Blink the HOUR field and wait for hours set.
;         Blink the MIMUTE field and wait for minutes set.
;         After a minutes is set exit.
;         If STOP then abort an action.
;         If CLEAR then start from Hour entry.
;=====================================================================
BaseTime  equ         156           ;15.6 ms

DoSetTOD:
          call        !ClearLCDDisplay
;         Display dash(--) and blink
          extrn       Display_DASH
          call        !Display_DASH
;
          mov         Interval_timer,#5000/BaseTime
```

```
          extrn       SetBlink
          call        !SetBlink
;
;
;
Retry:
          clr1        DoneFlag
          mov         Curr_Hours,#0
          mov         Curr_Minutes,#0
          mov         Curr_Field_Number,#0
          movw        Curr_Field_Ptr,#Curr_Hours
;
;         Wait for key released
;
          extrn       WaitKeyReleased
;
NextKeyLp:
          call        !WaitKeyReleased
srv_lop:
          call        !KeyScan
          bnc         $srv_lop
;
          mov         a,KeyNumber
          mov         b,a
          cmp         a,#KEY_CLEAR
          bz          $Retry        ;retry
;
;         Service
;
          call        !Branch
;
;         Done?
;
          bf          DoneFlag,$NextKeyLp
;
;         Blink off
;
          extrn       ClearBlink
          br          ClearBlink
;
;
;
KeyServiceTable:
          dw          KeyDigit0
          dw          KeyDigit1
          dw          KeyDigit2
          dw          KeyDigit3
          dw          KeyDigit4
          dw          KeyDigit5
          dw          KeyDigit6
          dw          KeyDigit7
          dw          KeyDigit8
          dw          KeyDigit9
          dw          KeySet
          dw          KeyEnter
          dw          KeyClear
          dw          KeySpare1
          dw          KeySpare2
;
;
;
KeyDigit0:
KeyDigit1:
KeyDigit2:
```

```
KeyDigit3:
KeyDigit4:
KeyDigit5:
KeyDigit6:
KeyDigit7:
KeyDigit8:
KeyDigit9:
          mov         b,#0
          mov         a,KeyNumber
          cmp         a,#DIGIT_0
          bz          $de10
          inc         b
          cmp         a,#DIGIT_1
          bz          $de10
          inc         b
          cmp         a,#DIGIT_2
          bz          $de10
          inc         b
          cmp         a,#DIGIT_3
          bz          $de10
          inc         b
          cmp         a,#DIGIT_4
          bz          $de10
          inc         b
          cmp         a,#DIGIT_5
          bz          $de10
          inc         b
          cmp         a,#DIGIT_6
          bz          $de10
          inc         b
          cmp         a,#DIGIT_7
          bz          $de10
          inc         b
          cmp         a,#DIGIT_8
          bz          $de10
          inc         b
          cmp         a,#DIGIT_9
          bz          $de10
de10:
          movw        ax,Curr_Field_Ptr
          xchw        ax,hl
          mov         a,[hl]
; value * 10 + unit
          clr1        CY
          rolc        a,1
          mov         c,a
          clr1        CY
          rolc        a,1
          clr1        CY
          rolc        a,1
          add         a,c
          add         a,b
          mov         [hl],a
          ret

KeySet:
          ret

KeyEnter:
          cmp         Curr_Field_Number,#HOUR_Field
          bnz         $enter10
; set
          mov         a,Curr_Hours
          mov         Hours,a
```

```
          movw        Curr_Field_Ptr,#Curr_Minutes
          inc         Curr_Field_Number
          ret
; set
enter10:
          mov         a,Curr_Minutes
          mov         Minutes,a
          set1        DoneFlag
          ret

KeyClear:
          ret

KeySpare1:
KeySpare2:
          ret

;=====================================================================
;         Branch
;         Input:
;                     ax = fwa of table entry
;                     b = index #
;=====================================================================
          public      Branch
Branch:
;         index * 2
          push        ax
          mov         a,b
          clr1        CY
          rolc        a,1
          mov         b,a
          pop         ax
;
          xch         a,x
          add         a,b
          add         a,b
          xch         a,x
          addc        a,#0
          xchw        ax,hl
          mov         a,[hl]
          xch         a,x
          mov         a,[hl+1]
          xch         a,x
          br          ax
;=====================================================================
          end
```

Assembly Language Program: DEMO_LCD.ASM

```
;*********************************************************************
;
;                                File name: DEMO_LCD.ASM
;                                Date: 1/29/98
;
;*********************************************************************
;        Description:
;
;
;The example program is an LCD Driver using 4-time-division and 1/2 bias.
;The program displays the time of day, HOURS:MINUTES;SECONDS.
;Default time is 12:00:00. Every 15.6 milliseconds the program checks whether a second has passed.
;If so, the display is updated.
;
;=====================================================================
;        The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock, but when a key is depressed,
;        the main system clock is activated.
;=====================================================================
;        for only DDB-K0070A
;
$set(DDB)
;=====================================================================
;        Equates
;=====================================================================
;
;        LCD Display Memory
;
LCDBuffer EQU          0FA7Fh
SegmentSizeEQU         40

HOURS_10th_PtEQU       0FA7Fh
HOURS_Unit_PtEQU       HOURS_10th_Pt- 4
COLON1_Pt EQU          HOURS_Unit_Pt - 4    ;offset 4 segments
MINUTES_10th_Pt EQUCOLON1_Pt - 4
MINUTES_Unit_Pt EQUMINUTES_10th_Pt - 4
COLON2_Pt EQU          MINUTES_Unit_Pt - 4
SECONDS_10th_Pt EQUCOLON2_Pt -4
SECONDS_Unit_Pt EQUSECONDS_10th_Pt -4

AMPM_Pt   EQU          0FA5Fh

;=====================================================================
;=====================================================================
;
;        External reference
;
          extrn        Branch
          extrn        Curr_Hours
          extrn  Curr_Minutes
          extrn  Days
          extrn  Hours
          extrn  Minutes
          extrn  Seconds

;=====================================================================
;        bit
;=====================================================================
          bseg
          public BlinkFlag
          public BlinkOnOffFlag
```

```
;
BlinkFlag dbit   ; blink LCD display
BlinkOnOffFlagdbit; blink on/off(1 = on)

;=======================================================================
;         saddr
;=======================================================================
          dseg   saddr
          public BlinkOnType
          public BlinkOffType
;
BlinkOnType:ds  1; blink ON
BlinkOffType:ds 1; blink OFF

;=======================================================================
;=======================================================================
          cseg
;=======================================================================
;
;         Default Time of day
;         Default:12:00:00
;
;=======================================================================
          public DisplayTOD
DisplayTOD:
          mov    a,Hours
          call   !DisplayHours

          movw   hl,#COLON1_Pt
          mov    a,#':'
          call   !DisplayOther
;
;         MINUTE: xx
;
          mov    a,Minutes
          call   !DisplayMinutes
;
;         (:)
;
          movw   hl,#COLON2_Pt
          mov    a,#':'
          call   !DisplayOther
;
;         Second: xx
;
          mov    a,Seconds
          br     DisplaySeconds

;=======================================================================
;
;         Display Dash( --:-- ) on the HOUR:MINUTE field
;
;=======================================================================
          public Display_DASH
Display_DASH:
;         '--'
          movw   hl,#HOURS_10th_Pt
          mov    a,#'-'
          call   !DisplayACharacter
          movw   hl,#HOURS_Unit_Pt
          mov    a,#'-'
          call   !DisplayACharacter
;         ':'
          movw   hl,#COLON1_Pt
```

```
          mov    a,#':'
          call   !DisplayOther
;         '--'
          movw   hl,#MINUTES_10th_Pt
          mov    a,#'-'
          call   !DisplayACharacter
          movw   hl,#MINUTES_Unit_Pt
          mov    a,#'-'
          br     !DisplayACharacter

;=====================================================================
;         Display a string to the LCD
;         Terminated either 00 or 0FFH
;
;         Input:
;                DE = String ptr
;         Return:
;                ptr = ptr++;
;=====================================================================
          public DisplayAString
DisplayAString:
;
;         Clear LCD display
;
          call   !ClearLCDDisplay
;
;
;
          movw   hl,#LCDBuffer
dlop:
          mov    a,[de]
          cmp    a,#00
          bz     $ds10;done
          cmp    a,#0ffh
          bz     $ds10;done
;
          push   de
          call   !DisplayACharacter

          pop    de
          incw   de
          br     dlop
ds10:
          ret

;=====================================================================
;         Display a character to the LCD
;
;         Input: a reg
;                hl= Position
;=====================================================================
          public DisplayACharacter
DisplayACharacter:
          push   hl

          mov    b,a

          movw   hl,#SegA14;Alphabet
          cmp    a,#30h
          bc     $achar25
;
          cmp    a,#3Ah
          bnc    $achar20;Alphabet
;
```

```
;         Digit
;
          sub    a,#30h
          pop    hl
;=========================================================================
;
;         Digit display
;
;=========================================================================
DigitDisplay:
          push   hl
          movw   hl,#SegD14;Digit
          br     $achar30
;
;         check Alphabets
;
achar20:
          cmp    a,#41h
          bc     $achar25
          cmp    a,#41H+26
;
;         Alphabet
;
          sub    a,#41h
          pop    hl
;=========================================================================
;
;         Alphabet display
;
;=========================================================================
AlphabetDisplay:
          push   hl
          br     $achar30
;
;         Not Alphabet or Digits
;
achar25:
          pop    hl
;
;=========================================================================
;
;         Input: a reg = ASCII data
;                HL = display memory address
;=========================================================================
DisplayOther:
          push   hl
          cmp    a,#':'
          bnz    $ck10
          movw   ax,#L_COLON
          br     achar80
ck10:
          cmp    a,#'-'
          bnz    $ck20
          movw   ax,#L_DASH
          br     achar80
ck20:
          movw   ax,#00;blank
          br     achar80
;
;         Read 2 Bytes
;
achar30:
          clr1   cy
          rolc   a,1;* 2
```

```
          mov    b,a
          mov    a,[hl+b]
          xch    a,x
          inc    b
          mov    a,[hl+b]
          xch    a,x;a = upper, x = lower byte
;
;         a = upper byte/b = lower byte
;
;         X---h
achar80:
          pop    hl
;
;         HL = data ptr
;         AX = data
;
DisplayData:
;         X---h
          push   ax
          ror    a,1
          ror    a,1
          ror    a,1
          ror    a,1
          and    a,#0fh
          mov    [hl+0],a
;         -X--h
          pop    ax
          and    a,#0fh
          mov    [hl+1],a
;         --X-h
          mov    a,x
          push   ax

          ror    a,1
          ror    a,1
          ror    a,1
          ror    a,1
          and    a,#0fh
          mov    [hl+2],a
;         ---Xh
          pop    ax
          and    a,#0fh
          mov    [hl+3],a
;
          decw   hl
          decw   hl
          decw   hl
          decw   hl
          ret

;* ********************************************
;*         Define COM Line #
;* ********************************************
COM1      equ 1
COM2      equ 2
COM3      equ 4
COM4      equ 8
;*
;*         Segment Definitions
;*
s16_G     equ COM1
s16_J     equ COM2
s16_L     equ COM3
s16_D     equ COM4
```

```
s16_O     equ COM1*010h
s16_F     equ COM2*010h
s16_E     equ COM3*010h
s16_P     equ COM4*010h

s16_I     equ COM1*0100h
s16_B     equ COM2*0100h
s16_C     equ COM3*0100h
s16_N     equ COM4*0100h

s16_A     equ COM1*01000h
s16_H     equ COM2*01000h
s16_K     equ COM3*01000h
s16_M     equ COM4*01000h

;* ******************************************
;         Table for 14-Segment LCDs
;* ******************************************
SegD14:
          dw     s16_A+s16_B+s16_C+s16_D+s16_E+s16_F;1
          dw     s16_B+s16_C;1
          dw     s16_A+s16_B+s16_K+s16_J+s16_E+s16_D  ;2
          dw     s16_A+s16_B+s16_K+s16_J+s16_C+s16_D  ;3
          dw     s16_F+s16_J+s16_K+s16_B+s16_C  ;4
          dw     s16_A+s16_F+s16_J+s16_K+s16_C+s16_D  ;5
          dw     s16_A+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K  ;6
          dw     s16_A+s16_B+s16_C;7
          dw     s16_A+s16_B+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K;8
          dw     s16_A+s16_B+s16_C+s16_D+s16_F+s16_J+s16_K;9
;*
;*         Alphabet
;*
SegA14:
          dw     s16_F+s16_E+s16_A+s16_B+s16_C+s16_J+s16_K;A
          dw     s16_A+s16_B+s16_C+s16_D+s16_H+s16_M+s16_K;B
          dw     s16_A+s16_F+s16_E+s16_D   C
          dw     s16_A+s16_B+s16_C+s16_D+s16_H+s16_M;D
          dw     s16_A+s16_F+s16_E+s16_D+s16_J+s16_K;E
          dw     s16_A+s16_F+s16_E+s16_J+s16_K;F
          dw     s16_A+s16_F+s16_E+s16_D+s16_C+s16_K;G
          dw     s16_F+s16_E+s16_J+s16_K+s16_B+s16_C;H
          dw     s16_A+s16_H+s16_M+s16_D ;I
          dw     s16_E+s16_B+s16_C+s16_D ;J
          dw     s16_F+s16_E+s16_J+s16_I+s16_N;K
          dw     s16_F+s16_E+s16_D;L
          dw     s16_F+s16_E+s16_G+s16_I+s16_B+s16_C;M
          dw     s16_F+s16_E+s16_G+s16_N+s16_B+s16_C;N
          dw     s16_F+s16_E+s16_A+s16_B+s16_C+s16_D;O
          dw     s16_F+s16_E+s16_A+s16_B+s16_K+s16_J;P
          dw     s16_E+s16_F+s16_A+s16_B+s16_C+s16_D+s16_N;Q
          dw     s16_F+s16_E+s16_A+s16_B+s16_K+s16_J+s16_N;R
          dw     s16_A+s16_F+s16_J+s16_K+s16_C+s16_D;S
          dw     s16_A+s16_H+s16_M;T
          dw     s16_F+s16_E+s16_D+s16_C+s16_B;U
          dw     s16_F+s16_E+s16_L+s16_I ;V
          dw     s16_F+s16_E+s16_L+s16_N+s16_C+s16_B;W
          dw     s16_G+s16_N+s16_I+s16_L ;X
          dw     s16_F+s16_J+s16_K+s16_B+s16_M;Y
          dw     s16_A+s16_I+s16_L+s16_D ;Z

L_DASH      equ  s16_J+s16_K;dash(-)
L_UNDER   equs16_D;underline(_)
L_SLASH   equs16_I+s16_L;slash(/)
```

```
L_BSLASH  equs16_G+s16_N;Back slash(\)
L_STAR     equ  s16_I+s16_L+s16_G+s16_N ;star(*)
R_ARROW   equs16_G+s16_L+s16_J;Right arrow(->)
L_ARROW   equs16_I+s16_N+s16_N
L_BRACK   equs16_I+s16_N;Left arrow (<-)
R_BRACK   equs16_G+s16_L;>
L_APOSTR  equs16_I;apostrope(')
L_COLON   equs16_H+s16_M;colon(:)



;* ******************************************
;*         7-Segment LCDs
;* ******************************************
s7_E   equ COM1
s7_G   equ COM2
s7_F   equ COM3
s7_K   equ COM4
s7_C   equ COM1*010h
s7_B   equ COM2*010h
s7_A   equ COM3*010h
s7_D   equ COM4*010h
;* ******************************************
;*         7-Segment LCDs
;* ******************************************
Tab7Seg:
          ds     s7_A+s7_B+s7_C+s7_D+s7_E+s7_F;0
          ds     s7_B+s7_C;1
          ds     s7_A+s7_B+s7_G+s7_E+s7_D;2
          ds     s7_A+s7_B+s7_G+s7_C+s7_D;3
          ds     s7_F+s7_G+s7_B+s7_C;4
          ds     s7_A+s7_F+s7_G+s7_C+s7_D;5
          ds     s7_A+s7_F+s7_G+s7_C+s7_D+s7_E;6
          ds     s7_A+s7_B+s7_C;7
          ds     s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G;8
          ds     s7_A+s7_B+s7_C+s7_D+s7_F+s7_G;9

          ds     s7_A+s7_B+s7_E+s7_F+s7_G;P in OPEN
;
;         Must be less than 15
;
S7_S   equ     11
          ds     s7_A+s7_C+s7_D+s7_F+s7_G      ;S
S7_E   equ     12
          ds     s7_A+s7_D+s7_F+s7_E+s7_G      ;E
S7_P   equ     13
          ds     s7_A+s7_B+s7_F+s7_E+s7_G      ;P

dash_7    equ    s7_G
          ds     s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G;A

Colon7seg:
          ds     s7_B+s7_C;: - colon

;=====================================================================
;
;         CLEAR DISPLAY MEMORY(FA7Fh(SEG0) - FA58h(SEG39))
;
;=====================================================================
          public Init_LCD
Init_LCD:
          call   !ClearLCDDisplay
```

```
;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 1 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 1, Select 32.768-kHz) subsystem clock
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================
          mov    TCL2,#00010000b

;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0,Watch Operating Mode Selection)
;         | | | | | | |____ TMC2.1 (= 1,Prescaler Operation control-Enable)
;         | | | | | |______ TMC2.2 (= 1,5-bit counter operation control - enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for watch flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4 .19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"                          )
;
;=====================================================================
          mov    TMC2,#01010110b

;=====================================================================
;
;         LCDC - LCD Display control register
;
;         1 0 0 0 0 0 1 0
;         | | | | | | | |__ LCDC.0 (= 0, Power to LCD from BIAS pin)
;         | | | | | | |____ LCDC.1 (= 1, Power to LCD from BIAS pin)
;         | | | | | |______ LCDC.2 (= 0, Reserved)
;         | | | | |________ LCDC.3 (= 0, Reserved)
;         | | | |__________ LCDC.4 (= 0, Select S24-S39)
;         | | |____________ LCDC.5 (= 0, Select s24-s39)
;         | |______________ LCDC.6 (= 0, Select s24-s39)
;         |________________ LCDC.7 (= 1, Select s24-s39)
;
;=====================================================================
          mov    LCDC,#10000010b
          ret

;=====================================================================
;         Clear LCD display
;=====================================================================
          public ClearLCDDisplay
ClearLCDDisplay:
          push   hl
          push   ax
          push   bc

          movw   hl,#LCDBuffer
          mov    a,#0
          mov    b,#SegmentSize
clrmem:
```

```
          mov    [hl],a
          decw   hl
          dbnz   b,$clrmem

          pop    bc
          pop    ax
          pop    hl
          ret

;=========================================================================
;         Blink on
;=========================================================================
          public SrvBlinkOn
SrvBlinkOn:
          mov    a,BlinkOnType
          mov    b,a
          movw   ax,#ServiceOn
          br     !Branch

;=========================================================================
ServiceOn:
          dw     HourOn
          dw     MinuteOn

;=========================================================================
;         Blink off
;=========================================================================
          public SrvBlinkOff
SrvBlinkOff:
          mov    a,BlinkOffType
          mov    b,a
          movw   ax,#ServiceOff
          br     !Branch

;=========================================================================
ServiceOff:
          dw     HourOff
          dw     MinuteOff

;=========================================================================
;         HOUR ON
;=========================================================================
          public HourOn
HourOn:
          movw   hl,#HOURS_10th_Pt
          mov    a,Curr_Hours
          cmp    a,#0
          bnz    $houron10;has hours
          br     Display2Dashs

houron10:
          br     DisplayHours

;=========================================================================
;         MINUTE ON
;=========================================================================
          public MinuteOn
MinuteOn:
          movw   hl,#MINUTES_10th_Pt
          mov    a,Curr_Minutes
          cmp    a,#0
          bnz    $minute10;has hours
          br     Display2Dashs
minute10:
```

```
          br     DisplayMinutes

;=========================================================================
;         HOUR Off
;=========================================================================
          public HourOff
HourOff:
          movw   hl,#HOURS_10th_Pt
          mov    a,#0
          mov    [hl],a
          decw   hl
          mov    [hl],a
          movw   hl,#HOURS_Unit_Pt
          mov    [hl],a
          decw   hl
          mov    [hl],a
          ret

;=========================================================================
;         MINUTE Off
;=========================================================================
          public MinuteOff
MinuteOff:
          movw   hl,#MINUTES_10th_Pt
          mov    a,#0
          mov    [hl],a
          decw   hl
          mov    [hl],a
          movw   hl,#MINUTES_Unit_Pt
          mov    [hl],a
          decw   hl
          mov    [hl],a
          ret

;=========================================================================
;         Display '--'
;         input: hl = display memory address
;=========================================================================
          public Display2Dashs
Display2Dashs:
          mov    a,#'-'
          call   !DisplayACharacter
          mov    a,#'-'
          br     DisplayACharacter

;=========================================================================
;         Display hours in HOUR field
;         Input: a = hours
;=========================================================================
          public DisplayHours
DisplayHours:
          movw   hl,#HOURS_10th_Pt
          cmp    a,#10
          bnc    $dtod10 ;less than 10
; 0 AM to 9 AM
          push   ax
          movw   ax,#00
          call   !DigitDisplay
          pop    ax
          movw   hl,#HOURS_Unit_Pt
          br     DigitDisplay
dtod10:
          cmp    a,#12+1
          bnc    $dtod12 ;less than 12
```

```
; 10 AM to 12 NOON
          push   ax
          movw   ax,#1
          call   !DigitDisplay
          movw   hl,#HOURS_Unit_Pt
          pop    ax
          sub    a,#10
          br     DigitDisplay
;         13 to 23
dtod12:
; 13 hr to 21 hr
          cmp    a,#12+9
          bnc    $dtod14 ;less than 12
;
          push   ax
          movw   ax,#00
          call   !DigitDisplay
          movw   hl,#HOURS_Unit_Pt
          pop    ax
          sub    a,#12
          br     DigitDisplay
dtod14:
; 22 hr to 23 hr
          push   ax
          movw   ax,#1
          call   !DigitDisplay
          movw   hl,#HOURS_Unit_Pt
          pop    ax
          sub    a,#22
          br     DigitDisplay

;=====================================================================
;         Display minutes in MINUTE field
;         Input: a = minutes
;=====================================================================
          public DisplayMinutes
DisplayMinutes:
;
;         MINUTE: xx
;
          movw   hl,#MINUTES_10th_Pt
          br     DisplayAField

;=====================================================================
;         Display seconds in SECOND field
;         Input: a = seconds
;=====================================================================
          public DisplaySeconds
DisplaySeconds:
          movw   hl,#SECONDS_10th_Pt
          br     DisplayAField

;=====================================================================
;
;
;
;=====================================================================
          public DisplayAField
DisplayAField:
          mov    x,#0
mloop:
          cmp    a,#10
          bc     $mless10
          sub    a,#10
```

```
          inc    x
          br     mloop
;
mless10:
          xch    a,x
;         A = 10th , X = unit
          push   ax
          call   !DigitDisplay
          pop    ax
          xch    a,x
          br     DigitDisplay

;=========================================================================
          end
```

Assembly Language Program: DEM_TOD.ASM

```
;*************************************************************************
;               File name: DEM_TOD.ASM
;               Date: 1/29/98
;*************************************************************************
;        Description:
;
; This program is to demonstrate the Watch timer in the uPD78308 microcontroller.
; The watch timer control registers (TCL2 and TMC2)) are used to configure
; the watch timer function.
; The resolution of the watch timer and time base is selectable as either 0.25 second
; or 0.5 second when 4.19 MHz (standard: 4.194304 MHz) main system clock is
; used.
; Since there is no interrupt vector for 0.5-second interval, the watch
; timer's interval timer interrupt (INTT3) should be used to check whether
; 0.5 second has passed.
;=========================================================================
; The sample program will set minute, hour, and day with 0.5-second time base.
; The interval time is set to 15.6 ms. The interval time interrupt service
; routine is used to check whether the 0.5-second flag WTIF is set. If so, increment the
; 0.5-second count.
;=========================================================================

;=========================================================================
;        Equates
;=========================================================================

         public ASecondFlag
         public BlinkPeriod
         public Days
         public Hours
         public HalfSecondFlag
         public Interval_timer
         public Minutes
         public Seconds
         public Watch_timer

         extbit BlinkFlag
         extbit BlinkOnOffFlag

         extrn  BlinkOnType
         extrn  BlinkOffType
         extrn  Curr_Field_Number

;=========================================================================
;        saddr Area
;        FWA = 0fe20h
;=========================================================================
         dseg   saddr
;
;        Time of day by Watch timer mode
;
Interval_timer: ds1;Interval timer counter (15.6 ms)
BlinkPeriod:ds  1;blink time

Seconds:  ds     1;Seconds
Minutes:  ds     1;Minutes
Hours:    ds     1;Hours
Days:     ds     1;Days

;=========================================================================
;        Even Boundary Pair
;=========================================================================
```

```
        dseg   saddrp

;=====================================================================
;       Bit Segment
;=====================================================================
        bseg
HalfSecondFlagdbit;1/2-second toggle flag
ASecondFlagdbit ;1-second pass flag

        cseg
;=====================================================================
;       Init. Time of day register
;=====================================================================

        public Init_TOD
Init_TOD:

;=====================================================================
;       TCL2 - Timer clock select register 2
;
;       x x x 0 x x x x
;       | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;       | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;       | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;       | | | | |________ TCL2.3 (Reserved)
;       | | | |__________ TCL2.4 (= 0, Select 4.1 9MHz/2^7)
;       | | |____________ TCL2.5 (Buzzer output frequency selection)
;       | |______________ TCL2.6 (Buzzer output frequency selection)
;       |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================
        mov    TCL2,#00000000b
;=====================================================================
;
;       TMC2 - Watch timer mode control register
;
;       0 1 0 1 0 1 1 0
;       | | | | | | | |__ TMC2.0 (= 0, Watch Operating Mode Selection)
;       | | | | | | |____ TMC2.1 (= 1, Prescaler Operation control-Enable)
;       | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control - enable)
;       | | | | |________ TMC2.3 (= 0, 0.5 second for Wach flag set time)
;       | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx=4.19 MHz)
;       | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx=4.19 MHz)
;       | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx=4.19 MHz)
;       |________________ TMC2.7 (Reserved = 0)"                        )
;
;=====================================================================
        mov    TMC2,#01010110b
;
;       Clear time variables
;       Default time: 12:00:00
;
        mov    Seconds,#0
        mov    Minutes,#0
        mov    Hours,#12
        mov    Days,#0

        clr1   HalfSecondFlag
        clr1   ASecondFlag
        ret

        $ej
;=====================================================================
```

```
;         Module name: INTTM3 Watch timer Interval Time Interrupt
;
;         DESCRIPTION:
;                .Set Watch time variables
;
;         OPERATION:
;                . Flip the HalfSecondFlag
;                . If HalfSecondFlag = 0 then incr. seconds
;                . If Seconds == 60 incr. minutes
;                . If Minutes == 60 incr. hours
;                . If Hours   == 24 incr. days
;
;=====================================================================
Watch_timer:
;
;         decr. interval counter
;
          cmp    Interval_timer,#0
          bz     $watch10
;
          dec    Interval_timer
          bnz    $watch10
;
;         Blink on
;
          bf     BlinkFlag,$watch10;No blink
;
;         reset blink period
;
          mov    a,BlinkPeriod
          mov    Interval_timer,a
;
          bf     BlinkOnOffFlag,$bk10 ;Blink on
;
;         Blink OFF
;
          clr1   BlinkOnOffFlag
          extrn  SrvBlinkOff
          call   !SrvBlinkOff
          br     watch10
;
;         Blink ON
;
bk10:
          set1   BlinkOnOffFlag
          extrn  SrvBlinkOn
          call   !SrvBlinkOn
          br     watch10
;
;         Test .5 second flag
;
watch10:
          bf     WTIF,$wt20;Not yet
          clr1   WTIF
;
;         .5 seconds passed
;
          bf     HalfSecondFlag,$wt10  ;1/2 second passed
;         Clear half seconds flag
          clr1   HalfSecondFlag

          set1   ASecondFlag
;
          inc    Seconds
```

```
          cmp    Seconds,#60
          bc     $wt20
          mov    Seconds,#0
;
          inc    Minutes
          cmp    Minutes,#60
          bc     $wt20
          mov    Minutes,#0
;
          inc    Hours
          cmp    Hours,#24
          bc     $wt20
          mov    Hours,#0
;         Day
          inc    Days
          cmp    Days,#30
          bc     $wt20

          br     $wt20
;
wt10:
          set1   HalfSecondFlag
wt20:
          reti

;=========================================================================
;         Delay 1/2 second
;         If key is depressed, then quit.
;         CY = 1 else CY = 0
;=========================================================================
BaseTime  equ    156;15.6 ms
          public Delay500Msec
Delay500Msec:
          mov    Interval_timer,#5000/BaseTime
D500mslp:
          extrn  CheckKeyOn
          call   !CheckKeyOn
          bnz    $d500msrtn;Key on
;
          cmp    Interval_timer,#0
          bnz    $D500mslp
          clr1   CY
          ret
;
;         CF = 1 if key on
;
d500msrtn:
          set1   CY
          ret

;=========================================================================
;         Set Blink parameters
;         Input: Interval_timer
;=========================================================================
          public SetBlink
SetBlink:
          mov    a,BlinkPeriod
          mov    Interval_timer,a
          set1   BlinkFlag
          clr1   BlinkOnOffFlag
          mov    a,Curr_Field_Number
          mov    BlinkOnType,a
          ret
```

```
;=======================================================================
;
;         Clear Blink parameters
;
;=======================================================================
          public ClearBlink
ClearBlink:
          clr1   BlinkFlag
          clr1   BlinkOnOffFlag
          ret

;=======================================================================
;=======================================================================
;=======================================================================
          end
```

Assembly Language Program: DEMO_KEY.ASM

```
;*********************************************************************
;
;               File name: DEMO_KEY.ASM
;               Date: 1/29/98
;*********************************************************************
;        Description:
;
;
;The example program is a key input module that inputs signals from a key matrix
;of 4 x 4 keys. The keys can be pressed successively, and two or more keys can
;be pressed simultaneously. In the circuit shown in this example, the lower 4
;bits of port3 (P30 - P33) are used as key scan signals, and port 11 (P4 if µPD78054 or µPD78078) is
;used for key return signals. As the pullup resistor of port 11 for key return, the
;internal resistor set by software is used.
;
;Port 11 of the K0 family has a function to detect the falling edges of the
;eight port pins in parallel. If port 11 is used for key return signals,
;therefore, the standby mode can be released through detection of a falling
;edge, by key input.
;
;The input keys are stored to RAM on a one-key-to 1 bit basis. The RAM bit
;corresponding to a pressed key is set and the bit corresponding to a released
;key is cleared.  By testing the RAM data on a 1-bit basis starting
;from the first bit, the key status can be checked. To absorb key debouncing,
;the key is assumed to be valid when four successive key codes coincide with a
;given code. For example, if a key code is sampled every 16 ms, debouncing of
;48 ms to 64 ms can be absorbed.
;
;
; Key number:
;               0 = P114 + P30
;               1 = P114 + P31
;               2 = P114 + P32
;               3 = P114 + P33
;
;               4 = P115 + P30
;               5 = P115 + P31
;               6 = P115 + P32
;               7 = P115 + P33
;
;               8 = P116 + P30
;               9 = P116 + P31
;                10 = P116 + P32  SET
;                11 = P116 + P33  ENTER
;
;                12 = P117 + P30  CLEAR/STOP
;                13 = P117 + P31  STOP
;                14 = P117 + P32
;                15 = P117 + P33
;
;        P4 is used instead of P11 if µPD78054 or µPD78078.
;
;=====================================================================
;        The system has a main system clock( 5 MHz) and a subsystem clock( 32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock and when a key is depressed, the
;        main system clock is activated.
;=====================================================================

         public KeyNumber

         public Init_Key
```

```
         public KeyScan


;=====================================================================
;        Key Assignment
;=====================================================================

         PUBLIC DIGIT_0
         PUBLIC DIGIT_1
         PUBLIC DIGIT_2
         PUBLIC DIGIT_3
         PUBLIC DIGIT_4
         PUBLIC DIGIT_5
         PUBLIC DIGIT_6
         PUBLIC DIGIT_7
         PUBLIC DIGIT_8
         PUBLIC DIGIT_9
         PUBLIC KEY_SET
         PUBLIC KEY_ENTER
         PUBLIC KEY_CLEAR
         PUBLIC KEY_STOP
         PUBLIC KEY_SPARE1
         PUBLIC KEY_SPARE2

DIGIT_0  EQU   0
DIGIT_1  EQU   1
DIGIT_2  EQU   2
DIGIT_3  EQU   3
DIGIT_4  EQU   4
DIGIT_5  EQU   5
DIGIT_6  EQU   6
DIGIT_7  EQU   7
DIGIT_8  EQU   8
DIGIT_9  EQU   9
KEY_SET  EQU   10
KEY_ENTEREQU   11
KEY_CLEAREQU   12
KEY_STOP EQU   13
KEY_SPARE1EQU  14
KEY_SPARE2EQU  15

;=====================================================================
;=====================================================================
         dseg   saddr

KeyNumber:ds    1;Key number

;=====================================================================
;        Even Boundary Pair
;=====================================================================
         dseg   saddrp

;=====================================================================
;        Bit Segment
;=====================================================================
         bseg

;=====================================================================
;        Init key return port
;=====================================================================
         cseg

Init_Key:
```

```
;=====================================================================
;         Key Return Mode register (KRM)
;
;         0 0 0 0 0 1 0 0
;         | | | | | | | |__ KRM.0 (= 0, Not detected key return signal)
;         | | | | | | |____ KRM.1 (= 1, Standby mode release enabled)
;         | | | | | |______ KRM.2 (= 1, Select p114-p117 as key return signal port)
;         | | | | |________ KRM.3 (= 0, Select p114-p117 as key return signal port)
;         | | | |__________ KRM.4 (= 0, Reserved)
;         | | |____________ KRM.5 (= 0, Reserved)
;         | |______________ KRM.6 (= 0, Reserved)
;         |________________ KRM.7 (= 0, Reserved)
;
;         If µPD78054 or µPD78078, KRM.2 and KRM.3 cannot be defined.
;
;=====================================================================

          mov    KRM,#00000100b
; Set Scan Ports
          mov    PM3,#11110000B;P30-P33 Output
          mov    P3,#00000000b;Set P30-P33 to Low
; Set Sense Ports
$if(D054)
$else
          mov    PM11,#11110000B ;P114-P117 as input
$endif
;
;         pull-up resistor option on P11
;
$if(D054)
          mov    PUOL,#00010000B
$else
          mov    PUOH,#00001000B
$endif
          ret

;=====================================================================
;         Key scan module
;
;         The key matrix is 4 x 4. There are four scan ports (P30-P33) and
;         four key return signal ports (P114-P117).
;         Debounce count sets to 4
;
;         Key number:
;                0 = P114 + P30
;                1 = P114 + P31
;                2 = P114 + P32
;                3 = P114 + P33
;                4 = P115 + P30
;                5 = P115 + P31
;                6 = P115 + P32
;                7 = P115 + P33
;                8 = P116 + P30
;                9 = P116 + P31
;                10 = P116 + P32
;                11 = P116 + P33
;                12 = P117 + P30
;                13 = P117 + P31
;                14 = P117 + P32
;                15 = P117 + P33
;
;         Return:
;                cf = 1 if a key depressed
;                   and KeyNumber with key number
```

```
;
;               cf = 0 if no key depressed.
;
;         Registers:
;               a = Scratch
;               b = Total row or column numbers
;               c = Sense port
;               d = Row * 4 + column
;               e = Column port
;               h = Current key number
;               l = Number of debounces
;
;=======================================================================
KeyScan:
;
;         Find a key
;
          mov    h,#0ffh ;no key flag
          mov    l,#3

KeyScanLoop:
          and    P3,#11110000b
          nop
          nop
          nop
          nop
$if(D054)
          mov    a,P4
$else
          mov    a,P11
$endif
          and    a,#11110000b; mask low nibble
          cmp    a,#11110000b;all high
          bz     $NotFound;no key depressed
;
          mov    c,a
          mov    d,#0;key number
          mov    x,#00010000b;find a row
          mov    b,#4;4 rows
CheckRow:
          mov    a,x
          and    a,c
          bz     $RowFind;find the row
; shift to left
          mov    a,x
          clr1   cy
          rol    a,1
          mov    x,a
; update row * column
          mov    a,d
          add    a,#4
          mov    d,a
;
          dbnz   b,$CheckRow
;
;         Not Found
;
NotFound:
          clr1   cy
          ret
;
;         Row Find. Find Column
;         d = row*column
;         x = row position
```

```
;
RowFind:
          mov    e,#00000001b;find a column
          mov    b,#4;4 columns
CheckCol:
          mov    a,e
          mov    p3,a;set to high
          nop
          nop
          nop
          nop
$if(D054)
          mov    a,P4
$else
          mov    a,P11
$endif
          and    a,x
          bnz    $ColFind
; shift to left
          clr1   cy
          mov    a,e
          rol    a,1
          mov    e,a
;
;         Incr. key number
;
          inc    d
          dbnz   b,$CheckCol
          br     NotFound;not found
;
ColFind:
          mov    a,d
          cmp    a,h
          bz     $CheckDebounce ; ok
;
;         Restart key scan
;
          mov    h,a
;
;         Decr. debounce count
;
CheckDebounce:
          dec    l
          bnz    $KeyScanLoop
;
KeyFound:
          mov    a,h
          mov    KeyNumber,a
          set1   cy
          ret

;=====================================================================
;         Wait for a key released
;=====================================================================
          public WaitKeyReleased
WaitKeyReleased:
          nop
Waitlp:
          call   !CheckKeyOn
          bnz    $Waitlp ;key on
          ret

;=====================================================================
;         Check key on or off
```

```
;         return with ZF = 1 if no key is on
;=========================================================================
          public CheckKeyOn
CheckKeyOn:
          and    P3,#11110000b
          nop
          nop
          nop
$if(D054)
          mov    a,P4
$else
          mov    a,P11
$endif
          and    a,#11110000b;mask low nibble
          cmp    a,#11110000b;all high
          ret
;=========================================================================
          end
```

C Language Program: DEMO.H

```
/*************************************************************
*               File name: DEMO.H
*               Date:11/10/97
*        Include file for definition
*************************************************************/
#define TRUE         1
#define FALSE        0
//
//                   extended functions
//
#pragma sfr          // SFR
#pragma asm          // ASM
#pragma opc          // DATA INSERTION

#ifdef K0S
#pragma realregister // using in K0S compiler
#endif

#pragma HALT
#pragma STOP
#pragma NOP
#pragma DI
#pragma EI

#pragma access       // PEEK, POKE

#define CR           13
#define LF           10
//
// DDB board used to debug
//
#define DDB          0
//
//       KEY DEFINITIONS
//
#define DIGIT_0              0
#define DIGIT_1              1
#define DIGIT_2              2
#define DIGIT_3              3
#define DIGIT_4              4
#define DIGIT_5              5
#define DIGIT_6              6
#define DIGIT_7              7
#define DIGIT_8              8
#define DIGIT_9              9
#define KEY_SET              10
#define KEY_ENTER            11
#define KEY_CLEAR            12
#define KEY_STOP             13
#define KEY_SPARE1           14
#define KEY_SPARE2           15

#define BaseTime             156//;15.6 ms

// $set(D054)
//;
//;        LCD Display Memory
//;
#define LCDBuffer            0xFA7F
#define SegmentSize          40

#define HOURS_10th_Pt        0xFA7F
```

```
#define HOURS_Unit_Pt              HOURS_10th_Pt- 4
#define COLON1_Pt                  HOURS_Unit_Pt - 4    //;offset 4 segments
#define MINUTES_10th_Pt            COLON1_Pt - 4
#define MINUTES_Unit_Pt            MINUTES_10th_Pt - 4
#define COLON2_Pt                  MINUTES_Unit_Pt - 4
#define SECONDS_10th_Pt            COLON2_Pt -4
#define SECONDS_Unit_Pt            SECONDS_10th_Pt -4

#define AMPM_Pt                    0xFA5F


#define COM1                1
#define COM2                2
#define COM3                4
#define COM4                8
//*
//*                            Segment Definitions
//*
#define s16_G               COM1
#define s16_J               COM2
#define s16_L               COM3
#define s16_D               COM4

#define s16_O               COM1*0x10
#define s16_F               COM2*0x10
#define s16_E               COM3*0x10
#define s16_P               COM4*0x10

#define s16_I               COM1*0x100
#define s16_B               COM2*0x100
#define s16_C               COM3*0x100
#define s16_N               COM4*0x100

#define s16_A               COM1*0x1000
#define s16_H               COM2*0x1000
#define s16_K               COM3*0x1000
#define s16_M               COM4*0x1000

#define L_DASH              s16_J+s16_K//;dash(-)
#define L_COLON s16_H+s16_M  //;colon(:)

/* ;* ******************************************
;*        7-Segment LCDs
;* ****************************************** */

#define s7_E   COM1
#define s7_G   COM2
#define s7_F   COM3
#define s7_K   COM4
#define s7_C   COM1*0x10
#define s7_B   COM2*0x10
#define s7_A   COM3*0x10
#define s7_D   COM4*0x10
```

C Language Program: DEMO_CLK.C

```
/*;*************************************************************************
;
;               File name: DEMO_CLK.C
;               Date: 1/29/98
;
;        MAIN module
;*************************************************************************
;        Description:
;
;        This program demonstrate, the following functions in the K0/K0S series:
;               LCD display function
;               Keyboard function
;               Time of day function
;               Power down mode(standby mode)
;        Each module is written in a file that can be used as an application
;        note.
;
;=========================================================================
;        The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock but when a key is depressed, the
;        main system clock is activated.
;=========================================================================*/

#include "DEMO.H"

/*;=========================================================================
;        Extern reference
;=========================================================================*/

extern   bit  ASecondFlag;

extern __sreg unsigned char   Hours;
extern __sreg unsigned char   Interval_timer;
extern __sreg unsigned char   KeyNumber;
extern __sreg unsigned char   Minutes;
extern __sreg unsigned char   Seconds;


extern void ClearBlink();
extern void ClearLCDDisplay();
extern void DisplayTOD();
extern void DisplayAString();
extern void DoSetTOD();
extern void Init_LCD();
extern void Init_Key();
extern void Init_TOD();
extern void KeyScan();
extern void Display_DASH();
extern void SetBlink();
extern void WaitKeyReleased();


void Branch(int *, unsigned char);
void KeyClear();
void KeyDigit();
void KeyEnter();
void KeySet();
void KeySpare();
void ServiceKey();

void hdwinit();
```

```
void exit();
//
void hdwinit()
{
}

void exit()
{
}

/*;=========================================================================
;         bit Area
;         FWA = 0fe20h
;=========================================================================*/
bit DoneFlag;

/*;=========================================================================
;         saddr Area
;         FWA = 0fe20h
;=========================================================================*/
__sreg unsigned char Curr_Hours;
__sreg unsigned char Curr_Minutes;
__sreg unsigned char Curr_Field_Number;
__sreg unsigned char digit;
__sreg unsigned char *Curr_Field_Ptr;

#define HOUR_Field           0
#define MINUTE_Field         1
#define SECOND_Field         2

/*;=========================================================================
;         Even Boundary Pair
;=========================================================================*/
const unsigned char NEC_DEMO[] = "NEC DEMO";

const unsigned char ConvertToDigit[] ={

         DIGIT_0,
         DIGIT_1,
         DIGIT_2,
         DIGIT_3,
         DIGIT_4,
         DIGIT_5,
         DIGIT_6,
         DIGIT_7,
         DIGIT_8,
         DIGIT_9
};

//;
//;
//;
const int * KeyServiceTable[] = {
              KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeyDigit
         ,    KeySet
```

```
          ,   KeyEnter
          ,   KeyClear
          ,   KeySpare
          ,   KeySpare
};
/*;=====================================================================
;         MAIN
;=====================================================================*/
void main()
{
          //;
          //;      Oscillation mode select register
          //;       . Does not use divider circuit
          //;
          OSMS = 0b00000001;
          //;
          //;      Init.
          //;
          Init_Key();
          Init_TOD();
          Init_LCD();
          //;      Clear IRQ flags
          TMIF3 = 0;
          //;      Enable Watch Timer Interval Timer
          TMMK3 = 0;
          EI();
/*;=====================================================================
;         Display current time of day
;         Default: 12:00:00
;=====================================================================*/
          PCC=0b00000000;
          //;
          //;      Display a greeting message
          //;      'NEC DEMO'
          //;
          DisplayAString("NEC DEMO");
          while(Seconds < 5);
          Seconds = 0;
          //;
          //;      Display default time of day (12:00:00)
          //;
          ClearLCDDisplay();
          DisplayTOD();
          ASecondFlag = 0;
          //;
          //;      Go to STANDBY MODE and if wake up then
          //;      Check second on
          //;
          while(TRUE)
          {
           //;
           //;      a second passed?
           //;
           if( ASecondFlag)
           {
            ASecondFlag = 0;
            DisplayTOD();
           }

//;
//;         Scan Keyboard - every 15.6 ms will wake it up
//;
           NOP();
           NOP();
```

```
          STOP();
          NOP();
          KeyScan();
          if(KeyNumber != 0xff)
          {
               DI();
               ServiceKey();
               EI();
          }
         }
}

/*;=========================================================================
;         Service a key
;
;         Input: KeyNumber
;
;=========================================================================*/
void ServiceKey()
{
         if( KeyNumber != KEY_SET) return;

         /*;=========================================================================
         ;      Set time of day.
         ;      Clear the LCD display.
         ;      Blink the HOUR field and wait for hours set.
         ;      Blink the MINUTE field and wait for minutes set.
         ;      After minutes is set, exit.
         ;      If STOP then abort an action
         ;      If CLEAR then start from Hour entry
         ;=========================================================================*/
         ClearLCDDisplay();
         //;      Display dash(--) and blink
         Display_DASH();
         //;
         Interval_timer = 5000/BaseTime;
         SetBlink();
         while(TRUE)
         {
          DoneFlag = 0;
          Curr_Hours = 0;
          Curr_Minutes = 0;
          Curr_Field_Number = 0;
          Curr_Field_Ptr = &Curr_Hours;
          //;
          //;      Wait for key released
          //;
NextKeyLp:
          WaitKeyReleased();
srv_lop:
          KeyScan();
          if(KeyNumber == 0) goto srv_lop;
          if(KeyNumber == KEY_CLEAR) continue;
          //;
          //;      Service
          //;
          Branch( (int *)KeyServiceTable, KeyNumber);
          if( !DoneFlag ) goto NextKeyLp;
          //;
          //;      Blink off
          //;
          ClearBlink();
         }
}
```

```
/*;=========================================================================
*         Digit Process
*;=========================================================================*/
void KeyDigit()
{

          for(digit = 0 ; digit < 10; digit++)
          {
           if( KeyNumber == ConvertToDigit[digit]) break;
          }
          *Curr_Field_Ptr = (char)(*Curr_Field_Ptr * 10 + digit);
}
//
void KeySet()
{
}

void KeyEnter()
{
          switch(Curr_Field_Number)
          {
                 case HOUR_Field:
                 Hours = Curr_Hours;
                 Curr_Field_Ptr = &Curr_Minutes;
                 Curr_Field_Number++;
                 break;
                 case MINUTE_Field:
                 Minutes = Curr_Minutes;
                 DoneFlag = 1;
                 break;
          }
}


void KeyClear()
{
}

void KeySpare()
{
}


/* ************************************************************************
;         Branch
************************************************************************ */
void Branch(int *TblPtr, unsigned char IndexNumber)
{
          TblPtr + IndexNumber*2;

          __asm(" pop AX");
          __asm(" pop HL");

          __asm(" br ax ");
}
```

Language Program: DEMO_LCD.C

```
/*;*************************************************************************
;
;               File name: DEMO_LCD.C
;               Date: 1/29/98
;
;*************************************************************************
;        Description:
;
;
;The example program is an LCD Driver using 4-time-division and 1/2 bias.
;The program displays the time of day, HOURS:MINUTES;SECONDS. Default time is 12:00:00. Every 15.6
;milliseconds, the program checks whether a second has passed. If so, the display is updated.
;
;=========================================================================
;        The system has a main system clock (5 MHz) and a subsystem clock (32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock, but when a key is depressed the
;        main system clock is activated.
;=========================================================================*/
#include "DEMO.H"
/*;=========================================================================
;        Equates
;=========================================================================*/
//;
//;         Time of day by Watch timer mode
//;
extern __sreg  Interval_Seconds;
extern __sregunsigned char Seconds; //Seconds
extern __sregunsigned char Minutes; //Minutes
extern __sregunsigned char Hours;   //Hours
extern __sregunsigned char Days;    //Days

extern __sregunsigned char Curr_Hours;
extern __sregunsigned char Curr_Minutes;


bit BlinkFlag;    // blink LCD display
bit BlinkOnOffFlag; // blink on/off(1 = on)

__sreg    unsigned int CodeW;
__sreg    char *SegPtr; //Segment ptr
__sreg    unsigned char BlinkOnType;  //
__sreg    unsigned char BlinkOffType; //



void Branch( int *, unsigned char );
void DisplayACharacter(unsigned char );
void Display2Dashs();
void DisplayAField(unsigned char );
void DisplayHours(unsigned char);
void DisplayMinutes(unsigned char);
void DisplaySeconds(unsigned char);
void DisplayAString(char *);
void DisplayTOD(void);
void DisplayOther(unsigned char );
void LCDDisplay(void);
void DigitDisplay(unsigned char );
void AlphabetDisplay(unsigned char );

void  HourOn();
void  HourOff();
```

```
void  MinuteOn();
void  MinuteOff();
void  ClearLCDDisplay();

//;=========================================================================
//       Service ON
//;=========================================================================
const int * ServiceOn[] = {
         HourOn,
         MinuteOn
};

//;=========================================================================
//       Service OFF
//;=========================================================================
const int * ServiceOff[] = {
         HourOff,
         MinuteOff
};


/* ;* ********************************************
/* ;* ********************************************


/* ;* ********************************************
/* ;* ********************************************
;*         Define COM Line #
;* ******************************************** */
/*;* ********************************************
;        Table for 14-segment LCD
;* ******************************************** */
unsigned int const SegD14[] =
{
         s16_A+s16_B+s16_C+s16_D+s16_E+s16_F// ;0
         ,s16_B+s16_C// ;1
         ,s16_A+s16_B+s16_K+s16_J+s16_E+s16_D// ;2
         ,s16_A+s16_B+s16_K+s16_J+s16_C+s16_D// ;3
         ,s16_F+s16_J+s16_K+s16_B+s16_C// ;4
         ,s16_A+s16_F+s16_J+s16_K+s16_C+s16_D// ;5
         ,s16_A+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K// ;6
         ,s16_A+s16_B+s16_C// ;7
         ,s16_A+s16_B+s16_C+s16_D+s16_E+s16_F+s16_J+s16_K //;8
         ,s16_A+s16_B+s16_C+s16_D+s16_F+s16_J+s16_K //;9
};

//;*
//;*        Alphabet
//;*
const unsigned int SegA14[] =
{
         s16_F+s16_E+s16_A+s16_B+s16_C+s16_J+s16_K//;A
         ,s16_A+s16_B+s16_C+s16_D+s16_H+s16_M+s16_K //;B
         ,s16_A+s16_F+s16_E+s16_D //;C
         ,s16_A+s16_B+s16_C+s16_D+s16_H+s16_M //;D
         ,s16_A+s16_F+s16_E+s16_D+s16_J+s16_K //;E
         ,s16_A+s16_F+s16_E+s16_J+s16_K //;F
         ,s16_A+s16_F+s16_E+s16_D+s16_C+s16_K //;G
         ,s16_F+s16_E+s16_J+s16_K+s16_B+s16_C //;H
         ,s16_A+s16_H+s16_M+s16_D //;I
         ,s16_E+s16_B+s16_C+s16_D //;J
         ,s16_F+s16_E+s16_J+s16_I+s16_N //;K
         ,s16_F+s16_E+s16_D //;L
         ,s16_F+s16_E+s16_G+s16_I+s16_B+s16_C //;M
```

```
        ,s16_F+s16_E+s16_G+s16_N+s16_B+s16_C //;N
        ,s16_F+s16_E+s16_A+s16_B+s16_C+s16_D //;O
        ,s16_F+s16_E+s16_A+s16_B+s16_K+s16_J //;P
        ,s16_E+s16_F+s16_A+s16_B+s16_C+s16_D+s16_N //;Q
        ,s16_F+s16_E+s16_A+s16_B+s16_K+s16_J+s16_N //;R
        ,s16_A+s16_F+s16_J+s16_K+s16_C+s16_D //;S
        ,s16_A+s16_H+s16_M //;T
        ,s16_F+s16_E+s16_D+s16_C+s16_B //;U
        ,s16_F+s16_E+s16_L+s16_I //;V
        ,s16_F+s16_E+s16_L+s16_N+s16_C+s16_B //;W
        ,s16_G+s16_N+s16_I+s16_L //;X
        ,s16_F+s16_J+s16_K+s16_B+s16_M //;Y
        ,s16_A+s16_I+s16_L+s16_D //;Z
};
/* ;* ******************************************
;*        7-segment LCDs
;* ****************************************** */
const unsigned char Tab7Seg[] =
{
        s7_A+s7_B+s7_C+s7_D+s7_E+s7_F//;0
        ,s7_B+s7_C //;1
        ,s7_A+s7_B+s7_G+s7_E+s7_D //;2
        ,s7_A+s7_B+s7_G+s7_C+s7_D //;3
        ,s7_F+s7_G+s7_B+s7_C //;4
        ,s7_A+s7_F+s7_G+s7_C+s7_D //;5
        ,s7_A+s7_F+s7_G+s7_C+s7_D+s7_E //;6
        ,s7_A+s7_B+s7_C  //;7
        ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G //;8
        ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_G //;9
//;
//;        Must be less than 15
//;
        ,s7_A+s7_C+s7_D+s7_F+s7_G //;S
        ,s7_A+s7_D+s7_F+s7_E+s7_G //;E
        ,s7_A+s7_B+s7_F+s7_E+s7_G //;P
        ,s7_A+s7_B+s7_C+s7_D+s7_F+s7_E+s7_G //;A
        ,s7_B+s7_C //;: - colon
};

//;=====================================================================
//;      LCD Initialization
//;=====================================================================
void Init_LCD()
{
        ClearLCDDisplay();
/*;=====================================================================
;
;        LCDC - LCD Display control register
;
;        1 0 0 0 0 0 1 0
;        | | | | | | | |__ LCDC.0 (= 0, Power to LCD from BIAS pin)
;        | | | | | | |____ LCDC.1 (= 1, Power to LCD from BIAS pin)
;        | | | | | |______ LCDC.2 (= 0, Reserved)
;        | | | | |________ LCDC.3 (= 0, Reserved)
;        | | | |__________ LCDC.4 (= 0, Select S24-S39)
;        | | |____________ LCDC.5 (= 0, Select S24-S39)
;        | |______________ LCDC.6 (= 0, Select S24-S39)
;        |________________ LCDC.7 (= 1, Select S24-S39)
;
;=====================================================================*/

        LCDC = 0b10000010;

/*;=====================================================================
```

```
;
;        LCDM - LCD Display mode register
;
;        1 0 0 1 0 0 0 0
;        | | | | | | | |__ LCDM.0 (= 0, 4-time-division, 1/3 bias mode)
;        | | | | | | |____ LCDM.1 (= 0, 4-time-division, 1/3 bias mode)
;        | | | | | |______ LCDM.2 (= 0, 4-time-division, 1/3 bias mode)
;        | | | | |________ LCDM.3 (= 0, Normal mode(2.7 to 5.5 v)
;        | | | |__________ LCDM.4 (= 1, LCD clock 128 Hz)
;        | | |____________ LCDM.5 (= 0, LCD clock 128 Hz)
;        | |______________ LCDM.6 (= 0, LCD clock 128 Hz)
;        |________________ LCDM.7 (= 1, Display ON)
;
;=====================================================================*/
         LCDM = 0b10010000;
}

/*;=====================================================================
*        Clear display memory
*;=====================================================================*/
void     ClearLCDDisplay()
{

         unsigned char i;
         char *dptr;

         dptr = (char *)LCDBuffer;
         for(i=0;i<SegmentSize;i++) *dptr--= 0;
}

/*;=====================================================================
;        Default Time of day
;        Default:12:00:00
;=====================================================================*/
void DisplayTOD(void)
{
         SegPtr = (char *)(HOURS_10th_Pt);
         DisplayHours(Hours);

         SegPtr = (char *)(COLON1_Pt);
         DisplayOther(':');
         //;
         //;    MINUTE: xx
         //;
         DisplayMinutes(Minutes);
         //;
         //;      (:)
         //;
         SegPtr = (char *)(COLON2_Pt);
         DisplayOther(':');
         //;
         //;      Second: xx
         //;
         DisplaySeconds(Seconds);
}

/*;=====================================================================
;
;        Display Dash( --:-- ) on the HOUR:MINUTE field
;
;=====================================================================*/
void     Display_DASH()
{
//;        '--'
```

```
        SegPtr = (char *)(HOURS_10th_Pt);
        DisplayACharacter('-');
        SegPtr = (char *)(HOURS_Unit_Pt);
        DisplayACharacter('-');
        SegPtr = (char *)(COLON1_Pt);
        DisplayOther(':');
//      '--'
        SegPtr = (char *)(MINUTES_10th_Pt);
        DisplayACharacter('-');
        SegPtr = (char *)(MINUTES_Unit_Pt);
        DisplayACharacter('-');
}

/*;=====================================================================
;
;       Display a string to the LCD
;       Terminated either 0x00
;
;=====================================================================*/
void DisplayAString( char Buffer[])
{

char *dptr;

        SegPtr =Buffer;
        while(!*dptr)
        {
              DisplayACharacter(*dptr++);
        }
}

/*;=====================================================================
;       Display a character to the LCD
;=====================================================================*/
void DisplayACharacter(unsigned char code)
{

        if(code >= 0x30 && code <= 0x39 )
        {
              DigitDisplay(code);
        }
        else
        {
         if(code >= 0x41 && code <= 0x59 )
         {
              AlphabetDisplay(code);
         }
         else
              DisplayOther(code);
        }
}

/*;=====================================================================
;       Display a DIGIT( 0 to 9 or ASCII digits) to the LCD
;=====================================================================*/
void DigitDisplay(unsigned char code)
{
        CodeW = SegD14[ ( code & 0x0f) * 2];
        LCDDisplay();
}
/*;=====================================================================
;       Display an ALPHABET to the LCD
;=====================================================================*/
void AlphabetDisplay(unsigned char code)
```

```
{
        CodeW = SegA14[ (code - 0x41) * 2];
        LCDDisplay();
}

/*;=====================================================================
;       Display a SYMBOL to the LCD
;=====================================================================*/
void DisplayOther(unsigned char code)
{
        if(code == 0)
        {
              CodeW = 0;
              LCDDisplay();
        }
        if(code == ':')
        {
              CodeW = L_COLON;
              LCDDisplay();
        }
        if(code == '-')
        {
              CodeW = L_DASH;
              LCDDisplay();
        }
}

/*;=====================================================================
;
;       Display a Code
;       INPUT: CodeW = 16-bits
;
;=====================================================================*/
void LCDDisplay(void)
{
unsigned int c;

//;        X---h
        *SegPtr-- =  (unsigned char)(( (CodeW & 0xf000) >> 12 ) & 0xf);
//;        -X--h
        *SegPtr-- =  (unsigned char)(( (CodeW & 0x0f00) >> 8 ) & 0xf);
//;        --X-h
        *SegPtr-- =  (unsigned char)(( (CodeW & 0x00f0) >> 4 ) & 0xf);
//;        ---Xh
        *SegPtr-- =  (unsigned char )( CodeW & 0x000f);
}

/*;=====================================================================
;       Blink on
;=====================================================================*/
void SrvBlinkOn()
{
        Branch( (int *)ServiceOn, BlinkOnType);
}
/*;=====================================================================
;       Blink Off
;=====================================================================*/
void SrvBlinkOff()
{
        Branch( (int *)ServiceOff, BlinkOffType);
}

/*;=====================================================================
;       HOUR On
```

```
;=====================================================================*/
void  HourOn()
{
         SegPtr = (char *)( HOURS_10th_Pt);
         if(Curr_Hours != 0)
                DisplayHours(Curr_Hours);
         else
                Display2Dashs();
}

/*;=====================================================================
;         MINUTE On
;=====================================================================*/
void  MinuteOn()
{
         SegPtr = (char *)( MINUTES_10th_Pt);
         if(Curr_Minutes != 0)
                DisplayMinutes(Curr_Minutes);
         else
                Display2Dashs();
}

/*;=====================================================================
;         HOUR Off
;=====================================================================*/
void     HourOff()
{
         SegPtr = (char *)(HOURS_10th_Pt);
         *SegPtr-- = 0;
         *SegPtr = 0;
         SegPtr = (char *)(HOURS_Unit_Pt);
         *SegPtr-- = 0;
         *SegPtr = 0;
}

/*;=====================================================================
;         MINUTE Off
;=====================================================================*/
void     MinuteOff()
{
         SegPtr = (char *)(MINUTES_10th_Pt);
         *SegPtr-- = 0;
         *SegPtr = 0;
         SegPtr = (char *)(MINUTES_Unit_Pt);
         *SegPtr-- = 0;
         *SegPtr = 0;
}

/*;=====================================================================
;         Display '--'
;         input: hl = display memory address
;=====================================================================*/
void Display2Dashs()
{
         DisplayACharacter('-');
         DisplayACharacter('-');
}

/*;=====================================================================
;         Display an hour in HOUR field
;         Input: a = hours
;=====================================================================*/
void     DisplayHours(unsigned char hrs)
{
```

```
        if(hrs < 10)
        {
              //; 0 AM to 9 AM
              DigitDisplay(0);
              SegPtr = (char *)(HOURS_Unit_Pt);
              DigitDisplay(hrs);
        }
        if( hrs > 9 && hrs <= 12)
        {
              //; 10 AM to 12 NOON
              DigitDisplay(1);
              SegPtr = (char *)(HOURS_Unit_Pt);
              DigitDisplay(hrs-10);
        }
        if( hrs < 12 && hrs < 21)
        {
              //; 1 PM to 9 PM
              DigitDisplay(0);
              SegPtr = (char *)(HOURS_Unit_Pt);
              DigitDisplay(hrs-12);
        }
        if(hrs >= 22)
        {
              //; 10 PM to 12 MIDNIGHT
              DigitDisplay(1);
              SegPtr = (char *)(HOURS_Unit_Pt);
              DigitDisplay(hrs-22);
        }

}

/*;=========================================================================
;       Display minutes in MINUTE field
;       Input: a = minutes
;=========================================================================*/
void    DisplayMinutes( unsigned char min)
{
//;
//;         MINUTE: xx
//;
        SegPtr = (char *)( MINUTES_10th_Pt);
        DisplayAField(min);
}
/*;=========================================================================
;       Display seconds in SECONDS field
;       Input: a = seconds
;=========================================================================*/
void    DisplaySeconds(unsigned char sec)
{
        SegPtr = (char *)(SECONDS_10th_Pt);

        DisplayAField(sec);
}
/*;=========================================================================
;
;=========================================================================*/
void    DisplayAField(unsigned char cdata )
{
        DigitDisplay( cdata/10);
        DigitDisplay( cdata%10 );
}
```

C Language Program: DEMO_TOD.C

```
#pragma interrupt INTTM3 WatchTimer RB1

/*;****************************************************************************
;               File name: DEMO_TOD.C
;               Date: 1/29/98
;****************************************************************************
;        Description:
;
; This program is to demonstrate the watch timer in the µPD78308 microcontroller.
; The watch timer control registers (TCL2, and TMC2)) are used to configure
; the watch timer function.
; The resolution of the Watch timer time base is selectable as either 0.25
; or 0.5 seconds, when 4.19 MHz (standard: 4.194304 MHz) main system clock is
; used.
; Since there is no interrupt vector for 0.5-second interval, the watch
; timer's interval time interrupt (INTT3) should be used to check if
; 0.5 second has passed.
;============================================================================
; The sample program will set minute, hour, and day with 0.5-second time base.
; The interval time is set to 15.6 ms. The interval timer is used to
; check 0.5-second flag WTIF set. If so, increment the 0.5-second count.
;============================================================================*/
#include "DEMO.h"

char t2 = 20;

//;
//;       Time of day by Watch timer mode
//;
__sreg   unsigned char Seconds; //Seconds
__sreg   unsigned char Minutes; //Minutes
__sreg   unsigned char Hours;   //Hours
__sreg   unsigned char Days;    //Days
__sreg   unsigned char Interval_timer;
__sreg   unsigned char BlinkPeriod;

bit HalfSecondFlag; //1/2 seconds toggle flag
bit ASecondFlag;


extern   bit BlinkFlag;
extern   bit BlinkOnOffFlag;

extern __sreg unsigned char  BlinkOnType;
extern __sreg unsigned char  BlinkOffType;
extern __sreg unsigned char  Curr_Field_Number;

extern   void  SrvBlinkOff();
extern   void  SrvBlinkOn();
extern   char  CheckKeyOn();

//;============================================================================
//;       Init. Time of day
//;============================================================================
void Init_TOD()
{
```

```
/* ;=====================================================================
;         TCL2 - Timer clock select register 2
;
;         x x x 0 x x x x
;         | | | | | | | |__ TCL2.0 (Watchdog timer count clock selection)
;         | | | | | | |____ TCL2.1 (Watchdog timer count clock selection)
;         | | | | | |______ TCL2.2 (Watchdog timer count clock selection)
;         | | | | |________ TCL2.3 (Reserved)
;         | | | |__________ TCL2.4 (= 0, Select 4.19 MHz/2^7)
;         | | |____________ TCL2.5 (Buzzer output frequency selection)
;         | |______________ TCL2.6 (Buzzer output frequency selection)
;         |________________ TCL2.7 (Buzzer output frequency selection)
;
;=====================================================================*/
         TCL2 = 0b00000000;
/*;=====================================================================
;
;         TMC2 - Watch timer mode control register
;
;         0 1 0 1 0 1 1 0
;         | | | | | | | |__ TMC2.0 (= 0, Watch Operating Mode Selection)
;         | | | | | | |____ TMC2.1 (= 1, Prescaler Operation control enable)
;         | | | | | |______ TMC2.2 (= 1, 5-bit counter operation control enable)
;         | | | | |________ TMC2.3 (= 0, 0.5 second for Wach flag set time)
;         | | | |__________ TMC2.4 (= 1, Prescaler interval timer - 15.6 ms (fxx =4.19 MHz)
;         | | |____________ TMC2.5 (= 0, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         | |______________ TMC2.6 (= 1, Prescaler interval timer - 15.6 ms (fxx = 4.19 MHz)
;         |________________ TMC2.7 (Reserved = 0)"                          )
;
;=====================================================================*/
         TMC2 = 0b01010110;
//;
//;         Clear time variables
//;
         Seconds=0;
         Minutes=0;
         Hours=0;
         Days= 0;

         HalfSecondFlag = 0;
         ASecondFlag = 0;
}
/*;=====================================================================
;         Module name: INTTM3 Watch timer Interval Time Interrupt
;
;         Description:
;                .Set Watch time variables
;
;         Operation:
;                . Flip the HalfSecondFlag
;                . If HalfSecondFlag = 0 then incr. seconds
;                . If Seconds == 60 incr. Minutes
;                . If Minutes == 60 incr. Hours
;                . If Hours   == 24 incr. Days
;
;=====================================================================*/
void WatchTimer()
{
         if( Interval_timer ) Interval_timer--;
         if( BlinkFlag )
         {
                Interval_timer = BlinkPeriod;
                if(BlinkOnOffFlag)
```

```
                {
                //; Blink OFF
                BlinkOnOffFlag = 0;
                SrvBlinkOff();
                }
                else
                {
                //; Blink ON
                BlinkOnOffFlag = 1;
                SrvBlinkOn();
                }
        }
        //;
        //;     Test .5 second flag
        //;
        if( !WTIF) return;
        //;
        WTIF = 0;
        //;
        //;      .5 seconds passed
        //;
        if( !HalfSecondFlag)
        {
               HalfSecondFlag=1;
               return;;
        }
        //;
        HalfSecondFlag=0;
        //;
        Seconds++;
        if(Seconds < 60) return;;

        Seconds = 0;
        Minutes++;
        if(Minutes < 60) return;;

        Minutes = 0;
        Hours++;
        if(Hours < 24) return;;

        Hours = 0;
        Days++;
}

/*;=========================================================================
;       Delay 1/2 Seconds
;       If key is depressed then quit
;       not zero,
;=========================================================================*/
char Delay500Msec()
{
        Interval_timer = 5000/BaseTime;
        while( !CheckKeyOn())
        {
               if( !Interval_timer)
               return 0;
        }
        return 1;
}
/*;=========================================================================
;
;       Set Blink Parameters
;       Input: Interval_timer
;
```

```
;=========================================================================*/
void SetBlink()
{
         Interval_timer = BlinkPeriod;
         BlinkOnType = Curr_Field_Number;
         BlinkOnOffFlag= 0;
         BlinkFlag  = 1;
}
/*;=========================================================================
;
;        Clear Blink Parameters
;
;=========================================================================*/
void     ClearBlink()
{
         BlinkOnOffFlag= 0;
         BlinkFlag  = 0;
}
```

C Language Program: DEMO_KEY.C

```
/*;*****************************************************************************
;
;               File name: DEMO_KEY.C
;               Date: 1/29/98
;*****************************************************************************
;        Description:
;
;
;The example program is a key input module that inputs signals from a key matrix
;of 4 x 4 keys. The keys can be pressed successively, and two or more keys can
;be pressed simultaneously. In the circuit shown in this example, the lower 4
;bits of port3 (P30-P33) are used as key scan signals, and port 11 (port 4 if µPD78054 or µPD78078)
;is used as key return signals. As the pull-up resister of port 11 for key return, the
;internal resistor set by software is used.
;
;Port 11 of the K0 family has a function to detect the falling edges of the
;eight port pins in parallel. If port 11 is used for key return signals,
;therefore, the standby mode can be released through detection of a falling
;edge, by key input.
;
;The input keys are stored to RAM on a bit/key basis. The RAM bit
;corresponding to a pressed key is set and the bit corresponding to a released
;key is cleared.  By testing the RAM data on a 1-bit basis starting
;from the first bit, the key status can be checked. To absorb key debouncing,
;the key is assumed to be valid when four successive key codes coincide with a
;given code. For example, if a key code is sampled every 16 msec, debouncing of
;48 to 64 ms can be absorbed.
;
;
; Key number:
;               0 = P114 + P30
;               1 = P114 + P31
;               2 = P114 + P32
;               3 = P114 + P33
;
;               4 = P115 + p30
;               5 = P115 + P31
;               6 = P115 + P32
;               7 = P115 + P33
;
;               8 = P116 + P30
;               9 = P116 + P31
;                10 = P116 + P32
;                11 = P116 + P33
;
;                12 = P117 + P30
;                13 = P117 + P31
;                14 = P117 + P32
;                15 = P117 + P33
;        P4 is used instead of P11 if µPD78054 or µPD78078.
;
;================================================================ ====
;        The system has a main system clock (5 MHz) and a subsystem clock(32.768 kHz).
;        The LCD and Watch timer are run under the subsystem clock.
;        Normal operation is under the subsystem clock and when a key is depressed, the
;        main system clock is activated.
;=====================================================================*/
#include "DEMO.H"
//#define D054  1  //for µPD78054 and 078

bit KeyOnFlag;    // key on flag if it is 1
```

```
__sreg unsigned char KeyNumber;

char CheckKeyOn();
void KeyScan(void);
void Init_Key();

/*;=====================================================================
;        MAIN
;=====================================================================*/
void Init_Key()
{
/*;=====================================================================
;        Key Return mode register(KRM)
;
;        0 0 0 0 0 1 0 0
;        | | | | | | | |__ KRM.0 (= 0, Not detected key return signal)
;        | | | | | | | |____ KRM.1 (= 1, Standby mode release enabled)
;        | | | | | | |______ KRM.2 (= 1, Select p114-p117 as key return signal port)
;        | | | | | |________ KRM.3 (= 0, Select p114-p117 as key return signal port)
;        | | | |__________ KRM.4 (= 0, Reserved)
;        | | |____________ KRM.5 (= 0, Reserved)
;        | |______________ KRM.6 (= 0, Reserved)
;        |________________ KRM.7 (= 0, Reserved)
;
;        If µPD78054 or µPD78078 KRM.2 and KRM.3 cannot be defined.
;
;=====================================================================*/

         KRM = 0B00000100;

//; Set Scan Ports

         PM3=0B11110000;  //P30-P33 Output
         P3 = 0B00000000; //Set P30-P33 to Low
//; Set Sense Ports
#ifdef D054
#else
         PM11= 0b11110000; //P114-P117 as input
#endif
//;
//;       Pull-up resistor option on P11
//;
#ifdef D054
         PUOL = 0b00010000;
#else
         PUOH = 0b00001000;
#endif
}

/*;=====================================================================
;        Key scan module
;
;        The key matrix is 4 x 4. There are four scan port (P30-P33) and
;        four key return signal ports (P114-P117).
;        Debounce count sets to 4
;
;        Key number:
;               0 = P114 + P30
;               1 = P114 + P31
;               2 = P114 + P32
;               3 = P114 + P33
;               4 = P115 + P30
;               5 = P115 + P31
;               6 = P115 + P32
```

```
;               7 = P115 + P33
;               8 = P116 + P30
;               9 = P116 + P31
;               10 = P116 + P32
;               11 = P116 + P33
;               12 = P117 + P30
;               13 = P117 + P31
;               14 = P117 + P32
;               15 = P117 + P33
;
;       Return:
;               cf = 1 if a key depressed
;                  and KeyNumber with key number
;
;               cf = 0 if no key depressed.
;
;       Registers:
;               a = Scratch
;               b = Total row or column numbers
;               c = Sense port
;               d = Row x 4 + column
;               e = Column port
;               h = Current key number
;               l = Number of debounces
;
;=========================================================================*/
void KeyScan(void)
{

#asm
;
;       Find a key
;
        push    ax
        push    bc
        push    de
        push    hl

        clr1    _KeyOnFlag
        mov     h,#0ffh ;no key flag
        mov     l,#3

KeyScanLoop:
        and     P3,#11110000b
$if(D054)
        mov     a,P4
$else
        mov     a,P11
$endif
        and     a,#11110000b; mask low nibble
        cmp     a,#11110000b;all high
        bz      $NotFound;no key depressed
        mov     c,a
        mov     d,#0;key number
        mov     x,#00010000b;find a row
        mov     b,#4;4 rows
CheckRow:
        mov     a,x
        and     a,c
        bz      $RowFind;find the row
; shift to left
        mov     a,x
        clr1    cy
        rol     a,1
```

```
          mov    x,a
; update row * column
          mov    a,d
          add    a,#4
          mov    d,a
          dbnz   b,$CheckRow
;
;         Not Found
;
NotFound:
          mov    _KeyNumber,#0
          push   hl
          push   de
          push   bc
          push   ax
          ret
;
;         Row Find. Find Column
;         d = row*column
;         x = row position
;
RowFind:
          mov    e,#00000001b;find a column
          mov    b,#4;4 columns
CheckCol:
          mov    a,e
          mov    P3,a;set to high
$if(D054)
          mov    a,P4
$else
          mov    a,P11
$endif
          and    a,x
          bnz    $ColFind
; shift to left
          clr1   cy
          mov    a,e
          rol    a,1
          mov    e,a
;
;         Incr. key number
;
          inc    d
          dbnz   b,$CheckCol
          br     NotFound;not found
;
ColFind:
          mov    a,d
          cmp    a,h
          bz     $CheckDebounce ; ok
;
;         Restart key scan
;
          mov    h,a
;
;         Decr. debounce count
;
CheckDebounce:
          dec    l
          bnz    $KeyScanLoop
;
KeyFound:
          mov    a,h
          mov    _KeyNumber,a
```

```
        set1   _KeyOnFlag

        push   hl
        push   de
        push   bc
        push   ax
        ret
#endasm
}

/*;=========================================================================
;       Wait for a key released
;=========================================================================*/
void WaitKeyReleased()
{
        while( !CheckKeyOn );
}

/*;=========================================================================
;       Check key on or off
;       return with ZF = 1 if no key is on
;=========================================================================*/
char CheckKeyOn()
{
unsigned char i;

        P3 &= 0b11110000;
        NOP();
        NOP();
        NOP();

#ifdef D054
        i = P4;
#else
        i = P11;
#endif
        i &= 0b11110000;
        if( i == 0b11110000)
              return 0;
        else
              return 1;
}
```

NEC

Some of the information contained in this document may vary from country to country. Before using any NEC product in your application, please contact a representative from the NEC office in your country to obtain a list of authorized representatives and distributors who can verify the following:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, the Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC

For literature, call **1-800-366-9782** 7 a.m. to 6 p.m. Pacific time
or FAX your request to **1-800-729-9288**
or visit our web site at **www.nec.com**

NEC Electronics Inc.

CORPORATE HEADQUARTERS
2880 Scott Boulevard
Santa Clara, CA 95050-2554
TEL 408-588-6000

U13218EU1V0AN00