



Annotated 'C' Examples for the 'F02x Family

Relevant Devices

This application note applies to the following devices:
C8051F020, C8051F021, C8051F022, and C8051F023.

Introduction

This note contains example code written in 'C' that can be used as a starting point for the development of applications based on the C8051F02x family of devices.

Index of Programs by Peripheral

The following short descriptions provide an index to the attached programs, organized by peripheral.

ADC0 Examples

The following are example programs which use ADC0.

"ADC0_Buf1.c"

This program shows an example of using ADC0 in interrupt mode using Timer3 overflows as a start-of-conversion source to sample AIN0 <NUM_SAMPLES> times, storing the results in XDATA space. Once <NUM_SAMPLES> have been collected, the samples are transmitted out UART0. Once the transmission has completed, another <NUM_SAMPLES> of data are collected and the process repeats.

"ADC0_Int1.c"

This program shows an example of using ADC0 in interrupt mode using Timer3 overflows as a start-of-conversion to measure the output of the on-chip temperature sensor. The temperature is calculated from the ADC0 result and is transmitted out UART0.

"ADC0_Int2m.c"

This program shows an example of using ADC0 in interrupt mode using Timer3 overflows as a start-of-conversion to measure the voltages on AIN0 through AIN7 and the temperature sensor. The voltages are calculated from the resulting codes and are transmitted out UART0.

"ADC0_OSA1.c"

This program shows an example of using ADC0 in interrupt mode using Timer3 overflows as a start-of-conversion to measure the output of the on-chip temperature sensor. The ADC0 results are filtered by a simple integrate-and-dump process whose integrate/decimate ratio is given by the constant <INT_DEC>. The temperature is calculated from the ADC0 result and is transmitted out UART0.



"ADC0_Poll1.c"

This program demonstrates operation of ADC0 in polled mode. The ADC0 is configured to use writes to AD0BUSY as its start of conversion source and to measure the output of the on-chip temperature sensor. The temperature sensor output is converted to degrees Celsius and is transmitted out UART0.

DAC0 Examples

The following are example programs which use DAC0. These can easily be converted to use DAC1 if desired.

"DAC0_DTMF1.c"

Example source code which outputs DTMF tones on DAC0. DAC0's output is scheduled to update at a rate determined by the constant <SAMPLERATED>, managed and timed by Timer4.

Oscillator Examples

The following are example programs which configure the internal and external oscillators.

"OSC_Cry1.c"

This program shows an example of how to configure the External Oscillator to drive a 22.1184 MHz crystal and to select this external oscillator as the system clock source. Also enables the missing clock detector reset function. Assumes an 22.1184 MHz crystal is attached between XTAL1 and XTAL2.

"OSC_Int1.c"

This program shows an example of how to configure the internal oscillator to its maxi-

imum frequency (~16 MHz). Also enables the Missing Clock Detector reset function.

Timer Examples

"Timer0_Poll1.c"

This program shows an example of using Timer0 in polled mode to implement a delay counter with a resolution of 1 ms.



Example Code

"ADC0_Buf1.c"

```
//-----  
// ADC0_Buf1.c  
//-----  
// Copyright 2001 Cygnal Integrated Products, Inc.  
//  
// AUTH: BW  
// DATE: 27 AUG 01  
//  
// This program shows an example of using ADC0 in interrupt mode using Timer3  
// overflows as a start-of-conversion to sample AIN0 <NUM_SAMPLES> times,  
// storing the results in XDATA space. Once <NUM_SAMPLES> have been  
// collected, the samples are transmitted out UART0. Once the transmission  
// has completed, another <NUM_SAMPLES> of data are collected and the process  
// repeats.  
//  
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.  
//  
// The system clock frequency is stored in a global constant SYSCLK. The  
// target UART baud rate is stored in a global constant BAUDRATE. The  
// ADC0 sampling rate is stored in a global constant SAMPLERATE0. The number  
// of samples collected during each batch is stored in <NUM_SAMPLES>. The  
// maximum value of <NUM_SAMPLES> is 2048 on a C8051F02x device with 4096  
// bytes of XRAM (assuming no external RAM is connected to the External  
// Memory Interface).  
//  
// Target: C8051F02x  
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51  
//  
  
//-----  
// Includes  
//-----  
  
#include <c8051f020.h> // SFR declarations  
#include <stdio.h>  
  
//-----  
// 16-bit SFR Definitions for 'F02x  
//-----  
  
sfr16 DP = 0x82; // data pointer  
sfr16 TMR3RL = 0x92; // Timer3 reload value  
sfr16 TMR3 = 0x94; // Timer3 counter  
sfr16 ADC0 = 0xbe; // ADC0 data  
sfr16 ADC0GT = 0xc4; // ADC0 greater than window  
sfr16 ADC0LT = 0xc6; // ADC0 less than window  
sfr16 RCAP2 = 0xca; // Timer2 capture/reload  
sfr16 T2 = 0xcc; // Timer2  
sfr16 RCAP4 = 0xe4; // Timer4 capture/reload  
sfr16 T4 = 0xf4; // Timer4  
sfr16 DAC0 = 0xd2; // DAC0 data  
sfr16 DAC1 = 0xd5; // DAC1 data
```



```
//-----  
// Global CONSTANTS  
//-----  
  
#define SYSCLK      22118400      // SYSCLK frequency in Hz  
#define BAUDRATE    115200       // Baud rate of UART in bps  
#define SAMPLERATE0 50000        // ADC0 Sample frequency in Hz  
#define NUM_SAMPLES 2048         // number of ADC0 samples to take in  
                                // sequence  
  
#define TRUE 1  
#define FALSE 0  
  
sbit LED = P1^6;                 // LED='1' means ON  
sbit SW1 = P3^7;                 // SW1='0' means switch pressed  
  
//-----  
// Function PROTOTYPES  
//-----  
  
void SYSCLK_Init (void);  
void PORT_Init (void);  
void UART0_Init (void);  
void ADC0_Init (void);  
void Timer3_Init (int counts);  
void ADC0_ISR (void);  
  
//-----  
// Global VARIABLES  
//-----  
  
xdata unsigned samples[NUM_SAMPLES]; // array to store ADC0 results  
bit ADC0_DONE;                       // TRUE when NUM_SAMPLES have been  
                                      // collected  
  
//-----  
// MAIN Routine  
//-----  
  
void main (void) {  
    int i;                             // loop counter  
  
    WDTCN = 0xde;                       // disable watchdog timer  
    WDTCN = 0xad;  
  
    SYSCLK_Init ();                     // initialize oscillator  
    PORT_Init ();                       // initialize crossbar and GPIO  
    UART0_Init ();                     // initialize UART0  
    Timer3_Init (SYSCLK/SAMPLERATE0); // initialize Timer3 to overflow at  
                                      // desired ADC0 sample rate  
  
    ADC0_Init ();                       // init ADC  
  
    EA = 1;                             // Enable global interrupts  
  
    while (1) {  
        // collect samples...
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
ADC0_DONE = FALSE;
LED = 1; // turn LED on during sample process
EIE2 |= 0x02; // enable ADC0 interrupts
while (ADC0_DONE == FALSE); // wait for samples to be taken

// upload samples to UART0
LED = 0; // turn LED off during upload process
for (i = 0; i < NUM_SAMPLES; i++) {
    printf ("%u\n", samples[i]);
}
printf ("\n");
}
}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i; // delay counter

    OSCXCN = 0x67; // start external oscillator with
                  // 22.1184MHz crystal

    for (i=0; i < 256; i++) ; // Wait for osc. to start up

    while (!(OSCXCN & 0x80)) ; // Wait for crystal osc. to settle

    OSCICN = 0x88; // select external oscillator as SYSCLK
                  // source and enable missing clock
                  // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0 = 0x04; // Enable UART0
    XBR1 = 0x00;
    XBR2 = 0x40; // Enable crossbar and weak pull-ups
    POMDOUT |= 0x01; // enable TX0 as a push-pull output
    P1MDOUT |= 0x40; // enable P1.6 (LED) as push-pull output
}

//-----
```



```

// UART0_Init
//-----
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
    SCON0    = 0x50;           // SCON0: mode 1, 8-bit UART, enable RX
    TMOD     = 0x20;           // TMOD: timer 1, mode 2, 8-bit reload
    TH1      = -(SYSCLK/BAUDRATE/16); // set Timer1 reload value for baudrate
    TR1      = 1;             // start Timer1
    CKCON    |= 0x10;          // Timer1 uses SYSCLK as time base
    PCON     |= 0x80;          // SMOD00 = 1
    TI0      = 1;             // Indicate TX0 ready
}

//-----
// ADC0_Init
//-----
//
// Configure ADC0 to use Timer3 overflows as conversion source, to
// generate an interrupt on conversion complete, and to use left-justified
// output mode. Enables ADC end of conversion interrupt. Enables ADC0, but
// leaves ADC0 end-of-conversion interrupts disabled.
//
void ADC0_Init (void)
{
    ADC0CN = 0x05;           // ADC0 disabled; normal tracking
                                // mode; ADC0 conversions are initiated
                                // on overflow of Timer3; ADC0 data is
                                // left-justified
    REF0CN = 0x07;           // enable temp sensor, on-chip VREF,
                                // and VREF output buffer
    AMX0SL = 0x00;           // Select AIN0 as ADC mux output
    ADC0CF = (SYSCLK/2500000) << 3; // ADC conversion clock = 2.5MHz
    ADC0CF &= ~0x07;         // PGA gain = 1
    EIE2    &= ~0x02;         // disable ADC0 interrupts

    ADOEN = 1;               // enable ADC0
}

//-----
// Timer3_Init
//-----
//
// Configure Timer3 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer3_Init (int counts)
{
    TMR3CN = 0x02;           // Stop Timer3; Clear TF3;
                                // use SYSCLK as timebase
    TMR3RL = -counts;        // Init reload values
    TMR3    = 0xffff;        // set to reload immediately
    EIE2    &= ~0x01;         // disable Timer3 interrupts
    TMR3CN |= 0x04;          // start Timer3
}

```



```
//-----  
// Interrupt Service Routines  
//-----  
  
//-----  
// ADC0_ISR  
//-----  
//  
// ADC0 end-of-conversion ISR  
// Here we take the ADC0 sample and store it in the global array <samples[]>  
// and update the local sample counter <num_samples>. When <num_samples> ==  
// <NUM_SAMPLES>, we disable ADC0 end-of-conversion interrupts and post  
// ADC0_DONE = 1.  
//  
void ADC0_ISR (void) interrupt 15 using 3  
{  
    static unsigned num_samples = 0;    // ADC0 sample counter  
  
    AD0INT = 0;                        // clear ADC0 conversion complete  
                                        // indicator  
  
    samples[num_samples] = ADC0;       // read and store ADC0 value  
  
    num_samples++;                     // update sample counter  
  
    if (num_samples == NUM_SAMPLES) {  
        num_samples = 0;               // reset sample counter  
        EIE2 &= ~0x02;                 // disable ADC0 interrupts  
        ADC0_DONE = 1;                 // set DONE indicator  
    }  
}
```



“ADC0_Int1.c”

```
//-----
// ADC0_int1.c
//-----
// Copyright 2001 Cygnal Integrated Products, Inc.
//
// AUTH: BW
// DATE: 18 AUG 01
//
// This program shows an example of using ADC0 in interrupt mode using Timer3
// overflows as a start-of-conversion to measure the output of the on-chip
// temperature sensor. The temperature is calculated from the ADC0 result
// and is transmitted out UART0.
//
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.
//
// The system clock frequency is stored in a global constant SYSCLK. The
// target UART baud rate is stored in a global constant BAUDRATE. The
// ADC0 sampling rate is stored in a global constant SAMPLERATE0.
//
// Target: C8051F02x
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//
//-----
// Includes
//-----

#include <c8051f020.h>           // SFR declarations
#include <stdio.h>

//-----
// 16-bit SFR Definitions for 'F02x
//-----

sfr16 DP      = 0x82;           // data pointer
sfr16 TMR3RL  = 0x92;           // Timer3 reload value
sfr16 TMR3    = 0x94;           // Timer3 counter
sfr16 ADC0    = 0xbe;           // ADC0 data
sfr16 ADC0GT  = 0xc4;           // ADC0 greater than window
sfr16 ADC0LT  = 0xc6;           // ADC0 less than window
sfr16 RCAP2   = 0xca;           // Timer2 capture/reload
sfr16 T2      = 0xcc;           // Timer2
sfr16 RCAP4   = 0xe4;           // Timer4 capture/reload
sfr16 T4      = 0xf4;           // Timer4
sfr16 DAC0    = 0xd2;           // DAC0 data
sfr16 DAC1    = 0xd5;           // DAC1 data

//-----
// Global CONSTANTS
//-----

#define SYSCLK      22118400     // SYSCLK frequency in Hz
#define BAUDRATE    9600        // Baud rate of UART in bps
#define SAMPLERATE0 50000       // ADC0 Sample frequency in Hz
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
sbit LED = P1^6; // LED='1' means ON
sbit SW1 = P3^7; // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----

void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);

//-----
// Global VARIABLES
//-----

long result; // ADC0 decimated value

//-----
// MAIN Routine
//-----

void main (void) {
    long temperature; // temperature in hundredths of a
                    // degree C
    int temp_int, temp_frac; // integer and fractional portions of
                            // temperature

    WDTCN = 0xde; // disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init (); // initialize oscillator
    PORT_Init (); // initialize crossbar and GPIO
    UART0_Init (); // initialize UART0
    Timer3_Init (SYSCLK/SAMPLERATE0); // initialize Timer3 to overflow at
    // sample rate

    ADC0_Init (); // init ADC

    ADOEN = 1; // enable ADC

    EA = 1; // Enable global interrupts

    while (1) {
        EA = 0; // disable interrupts
        temperature = result; // get ADC value from global variable
        EA = 1; // re-enable interrupts

        // calculate temperature in hundredths of a degree
        temperature = temperature - 41380;
        temperature = (temperature * 100L) / 156;
        temp_int = temperature / 100;
        temp_frac = temperature - (temp_int * 100);
        printf ("Temperature is %+02d.%02d\n", temp_int, temp_frac);
    }
}
```



```

}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                // delay counter

    OSCXCN = 0x67;        // start external oscillator with
                        // 22.1184MHz crystal

    for (i=0; i < 256; i++) ; // XTLVLD blanking interval (>1ms)

    while (!(OSCXCN & 0x80)) ; // Wait for crystal osc. to settle

    OSCICN = 0x88;        // select external oscillator as SYSCLK
                        // source and enable missing clock
                        // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x04;        // Enable UART0
    XBR1    = 0x00;
    XBR2    = 0x40;        // Enable crossbar and weak pull-ups
    POMDOUT |= 0x01;      // enable TX0 as a push-pull output
    P1MDOUT |= 0x40;      // enable P1.6 (LED) as push-pull output
}

//-----
// UART0_Init
//-----
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
    SCON0    = 0x50;        // SCON0: mode 1, 8-bit UART, enable RX
    TMOD     = 0x20;        // TMOD: timer 1, mode 2, 8-bit reload
    TH1      = -(SYSCLK/BAUDRATE/16); // set Timer1 reload value for baudrate
    TR1      = 1;          // start Timer1
    CKCON    |= 0x10;      // Timer1 uses SYSCLK as time base
}

```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
    PCON  |= 0x80;          // SMOD00 = 1
    TI0   = 1;             // Indicate TX0 ready
}

//-----
// ADC0_Init
//-----
//
// Configure ADC0 to use Timer3 overflows as conversion source, to
// generate an interrupt on conversion complete, and to use left-justified
// output mode. Enables ADC end of conversion interrupt. Leaves ADC disabled.
//
void ADC0_Init (void)
{
    ADC0CN = 0x05;          // ADC0 disabled; normal tracking
                          // mode; ADC0 conversions are initiated
                          // on overflow of Timer3; ADC0 data is
                          // left-justified

    REF0CN = 0x07;          // enable temp sensor, on-chip VREF,
                          // and VREF output buffer

    AMX0SL = 0x0f;          // Select TEMP sens as ADC mux output
    ADC0CF = (SYSCLK/2500000) << 3; // ADC conversion clock = 2.5MHz
    ADC0CF |= 0x01;         // PGA gain = 2

    EIE2 |= 0x02;          // enable ADC interrupts
}

//-----
// Timer3_Init
//-----
//
// Configure Timer3 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer3_Init (int counts)
{
    TMR3CN = 0x02;          // Stop Timer3; Clear TF3;
                          // use SYSCLK as timebase

    TMR3RL = -counts;       // Init reload values
    TMR3    = 0xffff;       // set to reload immediately
    EIE2   &= ~0x01;       // disable Timer3 interrupts
    TMR3CN |= 0x04;         // start Timer3
}

//-----
// Interrupt Service Routines
//-----

//-----
// ADC0_ISR
//-----
//
// ADC0 end-of-conversion ISR
// Here we take the ADC0 sample and store it in the global variable <result>.
//
void ADC0_ISR (void) interrupt 15
{
```

AN022 - Annotated 'C' Examples for the 'F02x Family



```
AD0INT = 0;                // clear ADC conversion complete
                           // indicator

result = ADC0;             // read ADC value
}
```



"ADC0_Int2m.c"

```
//-----  
// ADC0_int2m.c  
//-----  
// Copyright 2001 Cygnal Integrated Products, Inc.  
//  
// AUTH: BW  
// DATE: 25 AUG 01  
//  
// This program shows an example of using ADC0 in interrupt mode using Timer3  
// overflows as a start-of-conversion to measure the the voltages on AIN0  
// through AIN7 and the temperature sensor. The voltages are calculated from  
// the resulting codes and are transmitted out UART0.  
//  
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.  
//  
// The system clock frequency is stored in a global constant SYSCLK. The  
// target UART baud rate is stored in a global constant BAUDRATE. The  
// ADC0 sampling rate is stored in a global constant SAMPLERATE0. The voltage  
// reference value is stored in a constant VREF0, and is used to convert the  
// resulting codes from the ADC0 measurements into a voltage.  
//  
// Target: C8051F02x  
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51  
//  
//-----  
// Includes  
//-----  
  
#include <c8051f020.h>           // SFR declarations  
#include <stdio.h>  
  
//-----  
// 16-bit SFR Definitions for 'F02x  
//-----  
  
sfr16 DP           = 0x82;           // data pointer  
sfr16 TMR3RL       = 0x92;           // Timer3 reload value  
sfr16 TMR3         = 0x94;           // Timer3 counter  
sfr16 ADC0         = 0xbe;           // ADC0 data  
sfr16 ADC0GT       = 0xc4;           // ADC0 greater than window  
sfr16 ADC0LT       = 0xc6;           // ADC0 less than window  
sfr16 RCAP2        = 0xca;           // Timer2 capture/reload  
sfr16 T2           = 0xcc;           // Timer2  
sfr16 RCAP4        = 0xe4;           // Timer4 capture/reload  
sfr16 T4           = 0xf4;           // Timer4  
sfr16 DAC0         = 0xd2;           // DAC0 data  
sfr16 DAC1         = 0xd5;           // DAC1 data  
  
//-----  
// Global CONSTANTS  
//-----  
  
#define SYSCLK      22118400         // SYSCLK frequency in Hz  
#define BAUDRATE    9600             // Baud rate of UART in bps
```

AN022 - Annotated 'C' Examples for the 'F02x Family



```
#define SAMPLERATE0 50000 // ADC0 Sample frequency in Hz
#define VREF0 2430 // VREF voltage in millivolts

sbit LED = P1^6; // LED='1' means ON
sbit SW1 = P3^7; // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----

void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);

//-----
// Global VARIABLES
//-----

long result[9]; // AIN0-7 and temp sensor output
// results

//-----
// MAIN Routine
//-----

void main (void) {
    long voltage; // voltage in millivolts
    int i; // loop counter
    // voltage

    WDTCN = 0xde; // disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init (); // initialize oscillator
    PORT_Init (); // initialize crossbar and GPIO
    UART0_Init (); // initialize UART0
    Timer3_Init (SYSCLK/SAMPLERATE0); // initialize Timer3 to overflow at
    // sample rate

    ADC0_Init (); // init ADC

    ADOEN = 1; // enable ADC

    EA = 1; // Enable global interrupts

    while (1) {
        for (i = 0; i < 9; i++) {
            EA = 0; // disable interrupts
            voltage = result[i]; // get ADC value from global variable
            EA = 1; // re-enable interrupts

            // calculate voltage in millivolts
            voltage = voltage * VREF0;
            voltage = voltage >> 16;
        }
    }
}
```



```
        printf ("Channel '%d' voltage is %ldmV\n", i, voltage);
    }
}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                // delay counter

    OSCXCN = 0x67;        // start external oscillator with
                        // 22.1184MHz crystal

    for (i=0; i < 256; i++) ;    // XTLVLD blanking interval (>1ms)

    while (!(OSCXCN & 0x80)) ;    // Wait for crystal osc. to settle

    OSCICN = 0x88;        // select external oscillator as SYSCLK
                        // source and enable missing clock
                        // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x04;        // Enable UART0
    XBR1    = 0x00;
    XBR2    = 0x40;        // Enable crossbar and weak pull-ups
    POMDOUT |= 0x01;      // enable TX0 as a push-pull output
    P1MDOUT |= 0x40;      // enable P1.6 (LED) as push-pull output
}

//-----
// UART0_Init
//-----
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
    SCON0   = 0x50;        // SCON0: mode 1, 8-bit UART, enable RX
    TMOD    = 0x20;        // TMOD: timer 1, mode 2, 8-bit reload
}
```

AN022 - Annotated 'C' Examples for the 'F02x Family



```
    TH1    = -(SYSCLK/BAUDRATE/16);    // set Timer1 reload value for baudrate
    TR1    = 1;                        // start Timer1
    CKCON  |= 0x10;                    // Timer1 uses SYSCLK as time base
    PCON   |= 0x80;                    // SMOD00 = 1
    TIO    = 1;                        // Indicate TX0 ready
}

//-----
// ADC0_Init
//-----
//
// Configure ADC0 to use Timer3 overflows as conversion source, to
// generate an interrupt on conversion complete, and to use left-justified
// output mode. Enables ADC end of conversion interrupt. Leaves ADC disabled.
// Note: here we also enable low-power tracking mode to ensure that minimum
// tracking times are met when ADC0 channels are changed.
//
void ADC0_Init (void)
{
    ADC0CN = 0x45;                    // ADC0 disabled; low-power tracking
                                        // mode; ADC0 conversions are initiated
                                        // on overflow of Timer3; ADC0 data is
                                        // left-justified
    REF0CN = 0x07;                    // enable temp sensor, on-chip VREF,
                                        // and VREF output buffer
    AMX0SL = 0x00;                    // Select AIN0 as ADC mux output
    ADC0CF = (SYSCLK/2500000) << 3;  // ADC conversion clock = 2.5MHz
    ADC0CF &= ~0x07;                 // PGA gain = 1

    EIE2  |= 0x02;                    // enable ADC interrupts
}

//-----
// Timer3_Init
//-----
//
// Configure Timer3 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer3_Init (int counts)
{
    TMR3CN = 0x02;                    // Stop Timer3; Clear TF3;
                                        // use SYSCLK as timebase
    TMR3RL = -counts;                 // Init reload values
    TMR3    = 0xffff;                 // set to reload immediately
    EIE2    &= ~0x01;                 // disable Timer3 interrupts
    TMR3CN  |= 0x04;                 // start Timer3
}

//-----
// Interrupt Service Routines
//-----
//-----
// ADC0_ISR
//-----
//
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
// ADC0 end-of-conversion ISR
// Here we take the ADC0 sample and store it in the global array <result>.
// We also select the next channel to convert.
//
void ADC0_ISR (void) interrupt 15
{
    static unsigned char channel = 0;    // ADC mux channel (0-8)

    AD0INT = 0;                          // clear ADC conversion complete
                                          // indicator

    result[channel] = ADC0;               // read ADC value

    channel++;                             // change channel
    if (channel == 9) {
        channel = 0;
    }

    AMX0SL = channel;                     // set mux to next channel
}
```



"ADC0_OSA1.c"

```
//-----
// ADC0_OSA1.c
//-----
// Copyright 2001 Cygnal Integrated Products, Inc.
//
// AUTH: BW
// DATE: 18 AUG 01
//
// This program shows an example of using ADC0 in interrupt mode using Timer3
// overflows as a start-of-conversion to measure the output of the on-chip
// temperature sensor. The ADC0 results are filtered by a simple integrate-
// and-dump process whose integrate/decimate ratio is given by the constant
// INT_DEC. The temperature is calculated from the ADC0 result
// and is transmitted out UART0.
//
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.
//
// The system clock frequency is stored in a global constant SYSCLK. The
// target UART baud rate is stored in a global constant BAUDRATE. The
// ADC0 sampling rate is stored in a global constant SAMPLERATE0.
//
// Target: C8051F02x
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51

//-----
// Includes
//-----

#include <c8051f020.h>           // SFR declarations
#include <stdio.h>

//-----
// 16-bit SFR Definitions for 'F02x
//-----

sfr16 DP      = 0x82;           // data pointer
sfr16 TMR3RL  = 0x92;           // Timer3 reload value
sfr16 TMR3    = 0x94;           // Timer3 counter
sfr16 ADC0    = 0xbe;           // ADC0 data
sfr16 ADC0GT  = 0xc4;           // ADC0 greater than window
sfr16 ADC0LT  = 0xc6;           // ADC0 less than window
sfr16 RCAP2   = 0xca;           // Timer2 capture/reload
sfr16 T2      = 0xcc;           // Timer2
sfr16 RCAP4   = 0xe4;           // Timer4 capture/reload
sfr16 T4      = 0xf4;           // Timer4
sfr16 DAC0    = 0xd2;           // DAC0 data
sfr16 DAC1    = 0xd5;           // DAC1 data

//-----
// Global CONSTANTS
//-----

#define SYSCLK      22118400     // SYSCLK frequency in Hz
#define BAUDRATE    9600        // Baud rate of UART in bps
#define SAMPLERATE0 50000       // ADC0 Sample frequency in Hz
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
#define INT_DEC      256                // integrate and decimate ratio

sbit LED = P1^6;                // LED='1' means ON
sbit SW1 = P3^7;                // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----

void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);

//-----
// Global VARIABLES
//-----

long result;                    // ADC0 decimated value

//-----
// MAIN Routine
//-----

void main (void) {
    long temperature;           // temperature in hundredths of a
                                // degree C
    int temp_int, temp_frac;    // integer and fractional portions of
                                // temperature

    WDTCN = 0xde;               // disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init ();            // initialize oscillator
    PORT_Init ();              // initialize crossbar and GPIO
    UART0_Init ();             // initialize UART0
    Timer3_Init (SYSCLK/SAMPLERATE0); // initialize Timer3 to overflow at
                                // sample rate

    ADC0_Init ();              // init ADC

    ADOEN = 1;                 // enable ADC

    EA = 1;                    // Enable global interrupts

    while (1) {
        EA = 0;                // disable interrupts
        temperature = result;
        EA = 1;                // re-enable interrupts

        // calculate temperature in hundredths of a degree
        temperature = temperature - 41380;
        temperature = (temperature * 100L) / 156;
        temp_int = temperature / 100;
        temp_frac = temperature - (temp_int * 100);
    }
}
```



```

    printf ("Temperature is %+02d.%02d\n", temp_int, temp_frac);

    LED = ~SW1;                // LED reflects state of switch
}
}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                    // delay counter

    OSCXCN = 0x67;           // start external oscillator with
                            // 22.1184MHz crystal

    for (i=0; i < 256; i++) ; // XTLVLD blanking interval (>1ms)

    while (!(OSCXCN & 0x80)) ; // Wait for crystal osc. to settle

    OSCICN = 0x88;          // select external oscillator as SYSCLK
                            // source and enable missing clock
                            // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x04;         // Enable UART0
    XBR1    = 0x00;
    XBR2    = 0x40;         // Enable crossbar and weak pull-ups
    POMDOUT |= 0x01;       // enable TX0 as a push-pull output
    P1MDOUT |= 0x40;       // enable P1.6 (LED) as push-pull output
}

//-----
// UART0_Init
//-----
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
    SCON0   = 0x50;        // SCON0: mode 1, 8-bit UART, enable RX
}

```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
TMOD      = 0x20;           // TMOD: timer 1, mode 2, 8-bit reload
TH1       = -(SYSCLK/BAUDRATE/16); // set Timer1 reload value for baudrate
TR1       = 1;             // start Timer1
CKCON    |= 0x10;         // Timer1 uses SYSCLK as time base
PCON     |= 0x80;         // SMOD00 = 1
TIO      = 1;             // Indicate TX0 ready
}

//-----
// ADC0_Init
//-----
//
// Configure ADC0 to use Timer3 overflows as conversion source, to
// generate an interrupt on conversion complete, and to use left-justified
// output mode. Enables ADC end of conversion interrupt. Leaves ADC disabled.
//
void ADC0_Init (void)
{
    ADC0CN = 0x05;           // ADC0 disabled; normal tracking
                           // mode; ADC0 conversions are initiated
                           // on overflow of Timer3; ADC0 data is
                           // left-justified
    REF0CN = 0x07;           // enable temp sensor, on-chip VREF,
                           // and VREF output buffer
    AMX0SL = 0x0f;           // Select TEMP sens as ADC mux output
    ADC0CF = (SYSCLK/2500000) << 3; // ADC conversion clock = 2.5MHz
    ADC0CF |= 0x01;         // PGA gain = 2

    EIE2 |= 0x02;           // enable ADC interrupts
}

//-----
// Timer3_Init
//-----
//
// Configure Timer3 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer3_Init (int counts)
{
    TMR3CN = 0x02;           // Stop Timer3; Clear TF3;
                           // use SYSCLK as timebase
    TMR3RL = -counts;        // Init reload values
    TMR3   = 0xffff;         // set to reload immediately
    EIE2   &= ~0x01;         // disable Timer3 interrupts
    TMR3CN |= 0x04;         // start Timer3
}

//-----
// Interrupt Service Routines
//-----
//-----
// ADC0_ISR
//-----
//
// ADC0 end-of-conversion ISR
```

AN022 - Annotated 'C' Examples for the 'F02x Family



```
// Here we take the ADC0 sample, add it to a running total <accumulator>, and
// decrement our local decimation counter <int_dec>. When <int_dec> reaches
// zero, we post the decimated result in the global variable <result>.
//
void ADC0_ISR (void) interrupt 15
{
    static unsigned int_dec=INT_DEC;    // integrate/decimate counter
                                        // we post a new result when
                                        // int_dec = 0
    static long accumulator=0L;        // here's where we integrate the
                                        // ADC samples

    ADOINT = 0;                        // clear ADC conversion complete
                                        // indicator

    accumulator += ADC0;               // read ADC value and add to running
                                        // total
    int_dec--;                          // update decimation counter

    if (int_dec == 0) {                // if zero, then post result
        int_dec = INT_DEC;             // reset counter
        result = accumulator >> 8;
        accumulator = 0L;              // reset accumulator
    }
}
```



"ADC0_Poll1.c"

```
//-----  
// ADC0_Poll1.c  
//-----  
// Copyright 2001 Cygnal Integrated Products, Inc.  
//  
// AUTH: BW  
// DATE: 18 AUG 01  
//  
// This program demonstrates operation of ADC0 in polled mode. The ADC0 is  
// configured to use writes to AD0BUSY as its start of conversion source and  
// to measure the output of the on-chip temperature sensor. The temperature  
// sensor output is converted to degrees Celsius and is transmitted out UART0.  
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.  
//  
// The system clock frequency is stored in a global constant SYSCLK. The  
// target UART baud rate is stored in a global constant BAUDRATE.  
//  
// Target: C8051F02x  
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51  
//  
//-----  
// Includes  
//-----  
  
#include <c8051f020.h>           // SFR declarations  
#include <stdio.h>  
  
//-----  
// 16-bit SFR Definitions for 'F02x  
//-----  
  
sfr16 DP      = 0x82;           // data pointer  
sfr16 TMR3RL  = 0x92;           // Timer3 reload value  
sfr16 TMR3    = 0x94;           // Timer3 counter  
sfr16 ADC0    = 0xbe;           // ADC0 data  
sfr16 ADC0GT  = 0xc4;           // ADC0 greater than window  
sfr16 ADC0LT  = 0xc6;           // ADC0 less than window  
sfr16 RCAP2   = 0xca;           // Timer2 capture/reload  
sfr16 T2      = 0xcc;           // Timer2  
sfr16 RCAP4   = 0xe4;           // Timer4 capture/reload  
sfr16 T4      = 0xf4;           // Timer4  
sfr16 DAC0    = 0xd2;           // DAC0 data  
sfr16 DAC1    = 0xd5;           // DAC1 data  
  
//-----  
// Global CONSTANTS  
//-----  
  
#define SYSCLK      22118400     // SYSCLK frequency in Hz  
#define BAUDRATE    9600        // Baud rate of UART in bps  
  
sbit LED = P1^6;                // LED='1' means ON  
sbit SW1 = P3^7;                // SW1='0' means switch pressed
```



```
//-----  
// Function PROTOTYPES  
//-----  
  
void SYSCLK_Init (void);  
void PORT_Init (void);  
void UART0_Init (void);  
void ADC0_Init (void);  
  
//-----  
// Global VARIABLES  
//-----  
  
//-----  
// MAIN Routine  
//-----  
  
void main (void) {  
    long temperature;           // temperature in hundredths of a  
                                // degree C  
    int temp_int, temp_frac;    // integer and fractional portions of  
                                // temperature  
  
    WDTCN = 0xde;              // disable watchdog timer  
    WDTCN = 0xad;  
  
    SYSCLK_Init ();           // initialize oscillator  
    PORT_Init ();             // initialize crossbar and GPIO  
    UART0_Init ();           // initialize UART0  
  
    ADC0_Init ();             // init and enable ADC  
  
    // *** check difference in tracking modes ***  
  
    while (1) {  
        AD0INT = 0;           // clear conversion complete indicator  
        AD0BUSY = 1;          // initiate conversion  
        while (AD0INT == 0);  // wait for conversion complete  
        temperature = ADC0;    // read ADC0 data  
  
        // calculate temperature in hundredths of a degree  
        temperature = temperature - 41380;  
        temperature = (temperature * 100L) / 156;  
        temp_int = temperature / 100;  
        temp_frac = temperature - (temp_int * 100);  
        printf ("Temperature is %+02d.%02d\n", temp_int, temp_frac);  
    }  
}  
  
//-----  
// Initialization Subroutines  
//-----  
  
//-----  
// SYSCLK_Init  
//-----  
//
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                // delay counter

    OSCXCN = 0x67;        // start external oscillator with
                        // 22.1184MHz crystal

    for (i=0; i < 256; i++) ;        // XTLVLD blanking interval (>1ms)

    while (!(OSCXCN & 0x80)) ;        // Wait for crystal osc. to settle

    OSCICN = 0x88;        // select external oscillator as SYSCLK
                        // source and enable missing clock
                        // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x04;        // Enable UART0
    XBR1    = 0x00;
    XBR2    = 0x40;        // Enable crossbar and weak pull-ups
    POMDOUT |= 0x01;        // enable TX0 as a push-pull output
    P1MDOUT |= 0x40;        // enable P1.6 (LED) as push-pull output
}

//-----
// UART0_Init
//-----
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
    SCON0   = 0x50;        // SCON0: mode 1, 8-bit UART, enable RX
    TMOD    = 0x20;        // TMOD: timer 1, mode 2, 8-bit reload
    TH1     = -(SYSCLK/BAUDRATE/16); // set Timer1 reload value for baudrate
    TR1     = 1;          // start Timer1
    CKCON   |= 0x10;        // Timer1 uses SYSCLK as time base
    PCON    |= 0x80;        // SMOD00 = 1
    TIO     = 1;          // Indicate TX0 ready
}

//-----
// ADC0_Init
//-----
//
// Configure ADC0 to use AD0BUSY as conversion source, to use left-justified
// output mode, to use normal tracking mode, and to measure the output of
```

AN022 - Annotated 'C' Examples for the 'F02x Family



```
// the on-chip temperature sensor. Disables ADC0 end of conversion interrupt
// and ADC0 window compare interrupt.
//
void ADC0_Init (void)
{
    ADC0CN = 0x81;                // ADC0 enabled; normal tracking
                                // mode; ADC0 conversions are initiated
                                // on write to AD0BUSY; ADC0 data is
                                // left-justified
    REF0CN = 0x07;                // enable temp sensor, on-chip VREF,
                                // and VREF output buffer
    AMX0SL = 0x0f;                // Select TEMP sens as ADC mux output
    ADC0CF = (SYSCLK/2500000) << 3; // ADC conversion clock = 2.5MHz
    ADC0CF |= 0x01;              // PGA gain = 2

    EIE2 &= ~0x02;                // disable ADC0 EOC interrupt
    EIE1 &= ~0x04;                // disable ADC0 window compare interrupt
}
```



"DAC0_DTMF1.c"

```
//-----  
// DAC0_DTMF1.c  
//-----  
// Copyright 2001 Cygnal Integrated Products, Inc.  
//  
// AUTH: BW  
// DATE: 27 AUG 01  
//  
// Target: C8051F02x  
// Tool chain: KEIL C51  
//  
// Description:  
// Example source code which outputs DTMF tones on DAC0. DAC0's output is  
// scheduled to update at a rate determined by the constant SAMPLERATED,  
// managed and timed by Timer4.  
//  
// Implements a 256-entry full-cycle sine table of 8-bit precision.  
//  
// The output frequency is proportional a 16-bit phase adder.  
// At each DAC update cycle, the phase adder value is added to a running  
// phase accumulator.<phase_accumulator>, the upper bits of which are used  
// to access the sine lookup table.  
//  
//-----  
// Includes  
//-----  
  
#include <c8051f020.h>          // SFR declarations  
  
//-----  
// 16-bit SFR Definitions for 'F02x  
//-----  
  
sfr16 DP          = 0x82;      // data pointer  
sfr16 TMR3RL     = 0x92;      // Timer3 reload value  
sfr16 TMR3       = 0x94;      // Timer3 counter  
sfr16 ADC0       = 0xbe;      // ADC0 data  
sfr16 ADC0GT    = 0xc4;      // ADC0 greater than window  
sfr16 ADC0LT    = 0xc6;      // ADC0 less than window  
sfr16 RCAP2     = 0xca;      // Timer2 capture/reload  
sfr16 T2        = 0xcc;      // Timer2  
sfr16 RCAP4     = 0xe4;      // Timer4 capture/reload  
sfr16 T4        = 0xf4;      // Timer4  
sfr16 DAC0      = 0xd2;      // DAC0 data  
sfr16 DAC1      = 0xd5;      // DAC1 data  
  
//-----  
// Global CONSTANTS  
//-----  
#define SYSCLK 22118400        // SYSCLK frequency in Hz  
  
#define SAMPLERATED 100000L   // update rate of DAC in Hz  
  
#define phase_precision 65536 // range of phase accumulator
```



```
// DTMF phase adder values based on SAMPLERATED and <phase_precision>

#define LOW697697 * phase_precision / SAMPLERATED
#define LOW770 770 * phase_precision / SAMPLERATED
#define LOW852 852 * phase_precision / SAMPLERATED
#define LOW941 941 * phase_precision / SAMPLERATED

#define HI1209 1209 * phase_precision / SAMPLERATED
#define HI1336 1336 * phase_precision / SAMPLERATED
#define HI1477 1477 * phase_precision / SAMPLERATED
#define HI1633 1633 * phase_precision / SAMPLERATED

//-----
// Function PROTOTYPES
//-----

void main (void);
void SYSCLK_Init (void);
void PORT_Init (void);

void Timer4_Init (int counts);
void Timer4_ISR (void);

//-----
// Global VARIABLES
//-----
unsigned phase_add1;           // holds low-tone phase adder
unsigned phase_add2;           // holds low-tone phase adder

bit tone1_en;                  // enable = 1 for tone 1
bit tone2_en;                  // enable = 1 for tone 2

char code SINE_TABLE[256] = {
    0x00, 0x03, 0x06, 0x09, 0x0c, 0x0f, 0x12, 0x15,
    0x18, 0x1c, 0x1f, 0x22, 0x25, 0x28, 0x2b, 0x2e,
    0x30, 0x33, 0x36, 0x39, 0x3c, 0x3f, 0x41, 0x44,
    0x47, 0x49, 0x4c, 0x4e, 0x51, 0x53, 0x55, 0x58,
    0x5a, 0x5c, 0x5e, 0x60, 0x62, 0x64, 0x66, 0x68,
    0x6a, 0x6c, 0x6d, 0x6f, 0x70, 0x72, 0x73, 0x75,
    0x76, 0x77, 0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7c,
    0x7d, 0x7e, 0x7e, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7e, 0x7e,
    0x7d, 0x7c, 0x7c, 0x7b, 0x7a, 0x79, 0x78, 0x77,
    0x76, 0x75, 0x73, 0x72, 0x70, 0x6f, 0x6d, 0x6c,
    0x6a, 0x68, 0x66, 0x64, 0x62, 0x60, 0x5e, 0x5c,
    0x5a, 0x58, 0x55, 0x53, 0x51, 0x4e, 0x4c, 0x49,
    0x47, 0x44, 0x41, 0x3f, 0x3c, 0x39, 0x36, 0x33,
    0x30, 0x2e, 0x2b, 0x28, 0x25, 0x22, 0x1f, 0x1c,
    0x18, 0x15, 0x12, 0x0f, 0x0c, 0x09, 0x06, 0x03,
    0x00, 0xfd, 0xfa, 0xf7, 0xf4, 0xf1, 0xee, 0xeb,
    0xe8, 0xe4, 0xe1, 0xde, 0xdb, 0xd8, 0xd5, 0xd2,
    0xd0, 0xcd, 0xca, 0xc7, 0xc4, 0xc1, 0xbf, 0xbc,
    0xb9, 0xb7, 0xb4, 0xb2, 0xaf, 0xad, 0xab, 0xa8,
    0xa6, 0xa4, 0xa2, 0xa0, 0x9e, 0x9c, 0x9a, 0x98,
    0x96, 0x94, 0x93, 0x91, 0x90, 0x8e, 0x8d, 0x8b,
    0x8a, 0x89, 0x88, 0x87, 0x86, 0x85, 0x84, 0x84,
```



AN022 - Annotated 'C' Examples for the 'F02x Family

```
0x83, 0x82, 0x82, 0x81, 0x81, 0x81, 0x81, 0x81,
0x80, 0x81, 0x81, 0x81, 0x81, 0x81, 0x82, 0x82,
0x83, 0x84, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89,
0x8a, 0x8b, 0x8d, 0x8e, 0x90, 0x91, 0x93, 0x94,
0x96, 0x98, 0x9a, 0x9c, 0x9e, 0xa0, 0xa2, 0xa4,
0xa6, 0xa8, 0xab, 0xad, 0xaf, 0xb2, 0xb4, 0xb7,
0xb9, 0xbc, 0xbf, 0xc1, 0xc4, 0xc7, 0xca, 0xcd,
0xd0, 0xd2, 0xd5, 0xd8, 0xdb, 0xde, 0xe1, 0xe4,
0xe8, 0xeb, 0xee, 0xf1, 0xf4, 0xf7, 0xfa, 0xfd
};

//-----
// MAIN Routine
//-----

void main (void) {

    WDTCN = 0xde;           // Disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init ();

    PORT_Init ();

    REF0CN = 0x03;         // enable internal VREF generator

    DAC0CN = 0x97;         // enable DAC0 in left-justified mode
                          // using Timer4 as update scheduler
    Timer4_Init(SYSCLK/SAMPLERATED); // initialize T4 to generate DAC0
                          // schedule

    tone1_en = 1;         // enable low group tones
    tone2_en = 1;         // enable high group tones

    phase_add1 = LOW697;
    phase_add2 = HI1633;

    EA = 1;               // Enable global interrupts

    while (1);           // spin forever
}

//-----
// Init Routines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;               // delay counter
```



```

OSCXCEN = 0x67;           // start external oscillator with
                          // 22.1184MHz crystal

for (i=0; i < 256; i++) ; // Wait for osc. to start up

while (!(OSCXCEN & 0x80)) ; // Wait for crystal osc. to settle

OSCICEN = 0x88;          // select external oscillator as SYSCLK
                          // source and enable missing clock
                          // detector
}

//-----
// PORT_Init
//-----
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0      = 0x00;
    XBR1      = 0x00;
    XBR2      = 0x40;           // Enable crossbar and weak pull-ups
    P1MDOUT  |= 0x40;         // enable P1.6 (LED) as push-pull output
}

//-----
// Timer4_Init
//-----
// This routine initializes Timer4 in auto-reload mode to generate interrupts
// at intervals specified in <counts>.
//
void Timer4_Init (int counts)
{
    T4CON = 0;                // STOP timer; set to auto-reload mode
    CKCON |= 0x40;           // T4M = '1'; Timer4 counts SYSCLKs
    RCAP4 = -counts;         // set reload value
    T4 = RCAP4;
    EIE2 |= 0x04;           // enable Timer4 interrupts
    T4CON |= 0x04;          // start Timer4
}

//-----
// Interrupt Handlers
//-----

//-----
// Timer4_ISR
//-----
//
// This ISR is called on Timer4 overflows. Timer4 is set to auto-reload mode
// and is used to schedule the DAC output sample rate in this example.
// Note that the value that is written to DACOH during this ISR call is
// actually transferred to DAC0 at the next Timer4 overflow.
//

```



```
void Timer4_ISR (void) interrupt 16 using 3
{
    static unsigned phase_acc1 = 0; // holds low-tone phase accumulator
    static unsigned phase_acc2 = 0; // holds high-tone phase accumulator

    char temp1; // temp values for table results
    char temp2;
    char code *table_ptr;

    T4CON &= ~0x80; // clear T4 overflow flag

    table_ptr = SINE_TABLE;

    if ((tone1_en) && (tone2_en)) {
        phase_acc1 += phase_add1; // update phase acc1 (low tone)
        temp1 = *(table_ptr + (phase_acc1 >> 8));

        phase_acc2 += phase_add2; // update phase acc2 (high tone)
        // read the table value
        temp2 = *(table_ptr + (phase_acc2 >> 8));

        // now update the DAC value. Note: the XOR with 0x80 translates
        // the bipolar table look-up into a unipolar quantity.
        DAC0H = 0x80 ^ ((temp1 >> 1) + (temp2 >> 1));
    } else if (tone1_en) {
        phase_acc1 += phase_add1; // update phase acc1 (low tone)
        // read the table value
        temp1 = *(table_ptr + (phase_acc1 >> 8));

        // now update the DAC value. Note: the XOR with 0x80 translates
        // the bipolar table look-up into a unipolar quantity.
        DAC0H = 0x80 ^ temp1;
    } else if (tone2_en) {
        phase_acc2 += phase_add2; // update phase acc2 (high tone)
        // read the table value
        temp2 = *(table_ptr + (phase_acc2 >> 8));

        // now update the DAC value. Note: the XOR with 0x80 translates
        // the bipolar table look-up into a unipolar quantity.
        DAC0H = 0x80 ^ temp2;
    }
}
```



"OSC_Cry1.c"

```
//-----
// OSC_CRY1.c
//-----
// Copyright 2001 Cygnal Integrated Products, Inc.
//
// AUTH: BW
// DATE: 25 AUG 01
//
// This program configures shows an example of how to configure the external
// oscillator to drive a 22.1184MHz crystal and to select this external
// oscillator as the system clock source. Also enables the Missing Clock
// Detector reset function. Assumes an 22.1184MHz crystal is attached
// between XTAL1 and XTAL2.
//
// The system clock frequency is stored in a global constant SYSCLK.
//
// Target: C8051F02x
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//

//-----
// Includes
//-----

#include <c8051f020.h>           // SFR declarations

//-----
// 16-bit SFR Definitions for 'F02x
//-----

sfr16 DP      = 0x82;           // data pointer
sfr16 TMR3RL  = 0x92;           // Timer3 reload value
sfr16 TMR3    = 0x94;           // Timer3 counter
sfr16 ADC0    = 0xbe;           // ADC0 data
sfr16 ADC0GT  = 0xc4;           // ADC0 greater than window
sfr16 ADC0LT  = 0xc6;           // ADC0 less than window
sfr16 RCAP2   = 0xca;           // Timer2 capture/reload
sfr16 T2      = 0xcc;           // Timer2
sfr16 RCAP4   = 0xe4;           // Timer4 capture/reload
sfr16 T4      = 0xf4;           // Timer4
sfr16 DAC0    = 0xd2;           // DAC0 data
sfr16 DAC1    = 0xd5;           // DAC1 data

//-----
// Global CONSTANTS
//-----

#define SYSCLK      22118400     // SYSCLK frequency in Hz

sbit LED = P1^6;                // LED='1' means ON
sbit SW1 = P3^7;                // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----
```



```
void SYSCLK_Init (void);

//-----
// Global VARIABLES
//-----

//-----
// MAIN Routine
//-----

void main (void) {

    WDTCN = 0xde;           // disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init ();       // initialize oscillator

    while (1);
}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                 // delay counter

    OSCXCN = 0x67;        // start external oscillator with
                          // 22.1184MHz crystal

    for (i=0; i < 256; i++) ; // Wait for osc. to start

    while (!(OSCXCN & 0x80)) ; // Wait for crystal osc. to settle

    OSCICN = 0x88;        // select external oscillator as SYSCLK
                          // source and enable missing clock
                          // detector
}
}
```



“OSC_Int1.c”

```
//-----
// OSC_INT1.c
//-----
// Copyright 2001 Cygnal Integrated Products, Inc.
//
// AUTH: BW
// DATE: 25 AUG 01
//
// This program shows an example of how to configure the internal
// oscillator to its maximum frequency (~16MHz). Also enables the Missing
// Clock Detector reset function.
//
// The system clock frequency is stored in a global constant SYSCLK.
//
// Target: C8051F02x
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//
//-----
// Includes
//-----

#include <c8051f020.h>                // SFR declarations

//-----
// 16-bit SFR Definitions for 'F02x
//-----

sfr16 DP      = 0x82;                // data pointer
sfr16 TMR3RL  = 0x92;                // Timer3 reload value
sfr16 TMR3    = 0x94;                // Timer3 counter
sfr16 ADC0    = 0xbe;                // ADC0 data
sfr16 ADC0GT  = 0xc4;                // ADC0 greater than window
sfr16 ADC0LT  = 0xc6;                // ADC0 less than window
sfr16 RCAP2   = 0xca;                // Timer2 capture/reload
sfr16 T2      = 0xcc;                // Timer2
sfr16 RCAP4   = 0xe4;                // Timer4 capture/reload
sfr16 T4      = 0xf4;                // Timer4
sfr16 DAC0    = 0xd2;                // DAC0 data
sfr16 DAC1    = 0xd5;                // DAC1 data

//-----
// Global CONSTANTS
//-----

#define SYSCLK      16000000          // SYSCLK frequency in Hz

sbit LED = P1^6;                     // LED='1' means ON
sbit SW1 = P3^7;                     // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----

void SYSCLK_Init (void);
```



```
//-----  
// Global VARIABLES  
//-----  
  
//-----  
// MAIN Routine  
//-----  
  
void main (void) {  
  
    WDTCN = 0xde;           // disable watchdog timer  
    WDTCN = 0xad;  
  
    SYSCLK_Init ();        // initialize oscillator  
  
    while (1);  
}  
  
//-----  
// Initialization Subroutines  
//-----  
  
//-----  
// SYSCLK_Init  
//-----  
//  
// This routine initializes the internal oscillator to its maximum setting and  
// selects the internal oscillator as the system clock source. Also enables  
// the missing clock detector reset function.  
//  
void SYSCLK_Init (void)  
{  
    OSCICN = 0x87;         // configure internal oscillator to  
                           // highest frequency setting; select  
                           // internal oscillator as SYSCLK source  
                           // enable missing clock detector reset  
}
```



"Timer0_Poll1.c"

```
//-----
// Timer0_Poll1.c
//-----
// Copyright 2001 Cygnal Integrated Products, Inc.
//
// AUTH: BW
// DATE: 27 AUG 01
//
// This program shows an example of using Timer0 in polled mode to implement
// a delay counter with a resolution of 1 ms.
//
// Assumes an 22.1184MHz crystal is attached between XTAL1 and XTAL2.
//
// The system clock frequency is stored in a global constant SYSCLK.
//
// Target: C8051F02x
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//
//-----
// Includes
//-----
#include <c8051f020.h>           // SFR declarations

//-----
// 16-bit SFR Definitions for 'F02x
//-----

sfr16 DP      = 0x82;           // data pointer
sfr16 TMR3RL  = 0x92;           // Timer3 reload value
sfr16 TMR3    = 0x94;           // Timer3 counter
sfr16 ADC0    = 0xbe;           // ADC0 data
sfr16 ADC0GT  = 0xc4;           // ADC0 greater than window
sfr16 ADC0LT  = 0xc6;           // ADC0 less than window
sfr16 RCAP2   = 0xca;           // Timer2 capture/reload
sfr16 T2      = 0xcc;           // Timer2
sfr16 RCAP4   = 0xe4;           // Timer4 capture/reload
sfr16 T4      = 0xf4;           // Timer4
sfr16 DAC0    = 0xd2;           // DAC0 data
sfr16 DAC1    = 0xd5;           // DAC1 data

//-----
// Global CONSTANTS
//-----

#define SYSCLK      22118400      // SYSCLK frequency in Hz

sbit LED = P1^6;                 // LED='1' means ON
sbit SW1 = P3^7;                 // SW1='0' means switch pressed

//-----
// Function PROTOTYPES
//-----
```



```
void SYSCLK_Init (void);
void PORT_Init (void);
void Timer0_Delay (int ms);

//-----
// Global VARIABLES
//-----

//-----
// MAIN Routine
//-----

void main (void) {

    WDTCN = 0xde;           // disable watchdog timer
    WDTCN = 0xad;

    SYSCLK_Init ();        // initialize oscillator
    PORT_Init ();          // initialize crossbar and GPIO

    while (1) {
        Timer0_Delay (100); // delay for 100ms
        LED = ~LED;         // change state of LED
    }
}

//-----
// Initialization Subroutines
//-----

//-----
// SYSCLK_Init
//-----
//
// This routine initializes the system clock to use an 22.1184MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
    int i;                 // delay counter

    OSCXCN = 0x67;         // start external oscillator with
                          // 22.1184MHz crystal

    for (i=0; i < 256; i++) ; // Wait for osc. to start up

    while (!(OSCXCN & 0x80)) ; // Wait for crystal osc. to settle

    OSCICN = 0x88;         // select external oscillator as SYSCLK
                          // source and enable missing clock
                          // detector
}

//-----
// PORT_Init
//-----
//
```



```
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
    XBR0    = 0x00;
    XBR1    = 0x00;
    XBR2    = 0x40;           // Enable crossbar and weak pull-ups
    P1MDOUT |= 0x40;         // enable P1.6 (LED) as push-pull output
}

//-----
// Timer0_Delay
//-----
//
// Configure Timer0 to delay <ms> milliseconds before returning.
//
void Timer0_Delay (int ms)
{
    int i;                   // millisecond counter

    TCON  &= ~0x30;          // STOP Timer0 and clear overflow flag
    TMOD  &= ~0x0f;          // configure Timer0 to 16-bit mode
    TMOD  |=  0x01;
    CKCON |=  0x08;          // Timer0 counts SYSCLKs

    for (i = 0; i < ms; i++) { // count milliseconds
        TR0 = 0;             // STOP Timer0
        TH0 = (-SYSCLK/1000) >> 8; // set Timer0 to overflow in 1ms
        TL0 = -SYSCLK/1000;
        TR0 = 1;             // START Timer0
        while (TF0 == 0);    // wait for overflow
        TF0 = 0;             // clear overflow indicator
    }
}
```