DIFFERENCES BETWEEN THE C8051F300 AND THE C8051T60X DEVICE FAMILIES

1. Introduction

The C8051T60x devices are low-cost, byte-programmable EPROM microcontrollers based on the Silicon Laboratories CIP-51 microcontroller core. As with Silicon Laboratories' C8051F-series Flash-based microcontrollers, the C8051T60x are highly-integrated mixed-signal devices, incorporating communication peripherals as well as analog-to-digital converter (ADC) technology. Also included is the C2 2-wire debugging and programming interface, allowing designers to rapidly develop and debug firmware.

Because the devices in the C8051T60x family are one-time programmable, and there is no means of erasing the EPROM memory, a new device is required for testing every time the firmware changes. This could make the code development process more difficult and time consuming. Fortunately, the features found on the C8051T60x family are closely related to the features of the Flash-based C8051F30x family. In most cases, this allows the C8051F300 to be used for the entire code development process, and the firmware image will be interchangeable between the two device families. For systems which take advantage of the additional features found on the C8051T60x family, the majority of the firmware can still be developed on the C8051F300 device, and then ported to the C8051T60x during the final stages of code development.

This document details the hardware differences between the two device families and provides some guidelines for using the C8051F300 to develop code for the C8051T60x.

2. Key Points

- Although the majority of device features are identical between the C8051F300 and the C8051T60x, there are some hardware differences.
- When developing code for the C8051T60x device family, the majority of the code development process can be done on the Flash-based C8051F300.
- Each device has a set of unique features that are not present on the other family. Recognizing these differences is the key to successfully developing common code (code which will work on either family) as well as code that uses some of the additional features found on the C8051T60x.

3. Code Memory Storage

The most obvious difference between the C8051F30x and the C8051T60x devices is the code memory storage technology. On the C8051F30x devices, Flash memory is used, while on the C8051T60x devices, a one-time byte-programmable EPROM memory architecture is used. Table 1 details some parameters of interest related to the code storage technology.

Table 1. Code Memory Storage

Feature	C8051F300	C8051T60x
Code memory can be programmed multiple times	YES	NO
Programming voltage (V_{PP}) required to program code memory	NO	YES
Code memory can be written or erased from firmware on the device	YES	NO
Code memory can be read from firmware on the device	YES	

The impact of the code storage technology on the development of C8051T60x firmware is minimal. When developing firmware for the C8051T60x on the C8051F300 or porting an existing design, make certain that there are no firmware routines intended to write or erase areas of code memory, as they will not have any effect on the C8051T60x.

4. Special Function Registers

The special function register (SFR) memory map of the C8051T60x is very similar to the SFR memory map of the C8051F300. However, there are a few differences, related to functionality or features found on only one of the two device families. Fortunately, SFRs which exist on one family but not on the other can be safely written and read on the other device family without causing a problem. Likewise, certain registers have additional bits defined that are not present on both devices. In these cases, the default bit settings are safe to write, and the read values of those bits are defined in the datasheet. Figure 1 shows the combined SFR map of the two device families. The locations of SFRs which differ between the two families and those with only bitwise differences are highlighted. Note that three of the register locations associated with the output word of the ADC have different names on the two device families. The functionality of these registers is similar on both device families, and is discussed in more detail in Section 5.1.

0xF8	CPT0CN	PCA0L	PCA0H	PCA0CPL0	PCA0CPH0			
0xF0	B	P0MDIN					EIP1	
0xE8	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2			RSTSRC
0xE0	ACC	XBR0	XBR1	XBR2	IT01CF		EIE1	
0xD8	PCA0CN	PCA0MD	PCA0CPM0	PCA0CPM1	PCA0CPM2			
0xD0	PSW	**REF0CN**						
0xC8	TMR2CN							
0xC0	SMB0CN	SMB0CF	SMB0DAT	**ADC0GTL**	**ADC0GT** **ADC0GTH**	**ADC0LTL**	**ADC0LT** **ADC0LTH**	**REG0CN**
0xB8	IP			**AMX0SL**	**ADC0CF**	**ADC0L**	**ADC0** **ADC0H**	
0xB0		**OSCXCN**	OSCICN	OSCICL			**FLSCL**	**FLKEY**
0xA8	IE							
0xA0			**TOFFL**	**TOFFH**	P0MDOUT			
0x98	SCON0	SBUF0				CPT0MD		CPT0MX
0x90								
0x88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	**PSCTL**
0x80	P0	SP	DPL	DPH				PCON
	0(8) Bit-Addressable	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
	C8051F300 Registers			**C8051T60x Registers**			**Registers with Bit Differences**	

Figure 1. SFR Memory Map Differences

5. ADC and Temperature Sensor

The ADC peripheral is different between the two device families. The C8051F300 features an 8-bit, 500 ksps SAR, while the C8051T60x family has a 10-bit, 500 ksps SAR. A list of ADC differences which may affect the system and code design are detailed in Table 2.

Table 2. ADC and Temperature Sensor

Feature	C8051F300	C8051T60x
Output Word Resolution	8 bits	10 or 8 bits*
Throughput (Sampling Rate)	500 ksps	
Minimum Tracking Time	300 ns	
Conversion Time in SAR Clocks	11 Clocks	13 or 11 Clocks*
Maximum SAR Clock Speed	6 MHz	8.33 MHz
Gain Settings	1x, 2x, 4x, 0.5x	1x, 0.5x
Differential Inputs (AIN+ and AIN–)	YES	NO
Calibrated Temperature Sensor Offset	NO	YES
Voltage Reference (VREF) Options	V_{DD}, External Pin	V_{DD}, External Pin, LDO Output

***Note:** The normal output word resolution is 10 bits, which results in a conversion time of 10 SAR clocks. An “8-bit Compatibility Mode” is included in these devices for better code and hardware compatibility with the timing of the C8051F300 ADC. 8-bit Compatibility Mode reduces the output word resolution to 8 bits and reduces the conversion time to 11 SAR clocks.

5.1. ADC Output

The C8051T60x’s 10-bit ADC output word is stored in two 8-bit registers, ADC0H and ADC0L (locations 0xBE and 0xBD, respectively). The output word is left-justified, meaning that the most significant 8 bits are stored in register ADC0H, while the two LSBs of the ADC output word are bits 7 and 6 of the ADC0L register. On the C8051F300 device, the 8-bit output word is stored in a register called ADC0, which has the same SFR address as ADC0H on the 'T60x device. The SFR at location "ADC0L" on the C8051F300 is unused, and will read back 0x00 whenever it is read by code. This enables code which uses the full 10-bits of the C8051T60x ADC output to be developed on the C8051F300. On the C8051F300, the code will simply execute as though the two LSBs of the 10-bit data word are always zero.

The Window Comparator registers for the ADC are implemented in the same way, with the data left-justified on the C8051T60x, and the low-byte registers unused on the C8051F300. For example, the C8051T60x ADC0GTH register is in the same location as the C8051F300 ADC0GT register, and it holds the 8 MSBs of the 10-bit comparison value, while the ADC0GTL on the C8051T60x device holds the two least significant bits of the 10-bit comparison value, and is unused on the C8051F300. Reading the locations associated with the C8051T60x’s ADC0GTL or ADC0LTL register locations on the C8051F300 will read back a value of 0x00, and writing these locations will not have any effect.

When the C8051T60x is configured for 8-bit compatibility mode, the 8-bit output of the ADC is associated with the ADC0H register, and the ADC0L register will read 0x00. When using the Window Comparator function in 8-bit compatibility mode, the ADC0GTL and ADC0LTL registers should be explicitly set to 0x00 by firmware.

5.2. Analog Multiplexer and Gain Settings

The ADC on the C8051T60x contains a subset of the multiplexing and gain features found on the C8051F300. On the C8051T60x, only the positive channel (AIN+) of the ADC is available, meaning that only single-ended measurements are possible (AIN+ – GND). When developing code for the C8051T60x on the C8051F300, the four MSBs of the AMX0SL register should always be written to 1111b (binary). This will select GND as the negative input on the C8051F300's ADC, and perform a single-ended measurement.

The gain settings on the C8051T60x ADC can be chosen from 1x or 0.5x. When developing code on the C8051F300, the AMP0GN1-0 bits in register ADC0CF should always be set to 00b for 0.5x or 01b for 1x. The settings 10b (2x) and 11b (4x) are not valid on the C8051T60x. It is important to note that the LSB size (in Volts) for the C8051T60x ADC in 10-bit, 1x gain mode is equivalent to the LSB size (in Volts) of the 8-bit C8051F300 ADC in 4x gain mode, thus the same level of precision can be measured on either device.

5.3. SAR Timing

During a conversion, the SAR ADC is normally in one of two different phases; "tracking" or "converting", as shown in Figure 2.

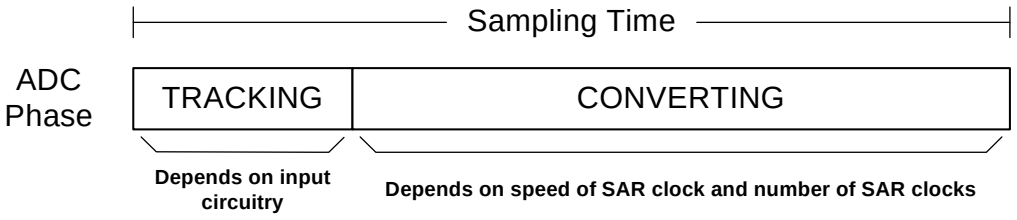


Figure 2. ADC Sampling Phases: Tracking and Converting

Establishing the same conversion time between the 'T60x and 'F30x requires programming the 'T60x to a faster SAR clock frequency. During the tracking phase, the ADC's sampling capacitor is connected to the external pin input through the analog multiplexer. When a conversion is initiated by the selected start-of-conversion source, the sampling capacitor is disconnected from the input, and the SAR conversion is performed. Even though both the C8051F300's ADC and the C8051T60x's ADC operate at a maximum sampling rate of 500 ksps, the number of SAR clocks required for the conversion phase in 10-bit mode on the C8051T60x is larger than the number required for either ADC in 8-bit mode. At a given SAR clock frequency, this leaves less of the total sampling time for tracking. The ADC on both devices requires a minimum 300 ns tracking time before each conversion, and additional tracking time may be necessary depending on the nature of the front-end circuitry (see the "Settling Time Requirements" section of the data sheet for more details). Fortunately, the C8051T60x SAR can operate with a faster SAR clock than the C8051F300 to compensate for the extra SAR clocks needed to complete a 10-bit conversion, and to provide comparable tracking/settling time at the same sampling rate. If the application requires the C8051T60x's SAR clock speed to be faster than the C8051F300 SAR clock specification, the switch to the faster SAR clock should be implemented as one of the final development steps after the majority of other code development is completed on the C8051F300.

5.4. Temperature Sensor

The usage of the temperature sensor on the C8051T60x is very similar to the C8051F300. However, the transfer function of the two temperature sensors is different. Different values for the offset and slope of the temperature sensor are necessary to accurately calculate the temperature on each device family. Additionally, the C8051T60x temperature sensor output has been measured during production test for each device, to help remove some of the error due to part-to-part variations. The results of the measurement are stored in the registers TOFFH and TOFFL. These register values represent the output of the temperature sensor at an ambient temperature of zero degrees Celsius, as measured by the ADC using the internal regulator as a reference voltage.

5.5. Voltage Reference Options

On the C8051F300, the ADC can use one of two different voltage reference options: the V_{DD} supply pin, or an external reference applied to the VREF pin. On the C8051T60x, an additional voltage reference option is available. By setting the REGOVR bit in the REF0CN register, the ADC can use the internal regulator as the voltage reference for the ADC.

5.6. External Conversion Start (CNVSTR) Timing

If the CNVSTR pin is used to begin conversions on the ADC, it is important to note the differences in timing between the C8051F300 and the C8051T60x family. On the C8051F300 the rising edge of CNVSTR always ends tracking mode and begins a conversion. On the C8051T60x family, when the AD0TM bit is set to ‘0’, conversions are initiated on the rising edge of CNVSTR, and tracking will occur when CNVSTR is low. On the C8051T60x, if f AD0TM is set to ‘1’, tracking will occur any time a conversion is not in progress, and it will last an additional three SAR clocks after the rising edge of CNVSTR.

6. Supply and I/O Pin Voltages

The C8051T60x is implemented in a different process technology than the C8051F30x. Consequently, there are some additional features and restrictions related to the supply voltage and allowable I/O pin voltages, which are detailed in Table 3.

Table 3. Supply and I/O Pin Voltages

Feature	C8051F300	C8051T60x
Supply Voltage Range	2.7–3.6 V	1.8–3.6 V
LDO Regulator for Internal Core Voltage	NO	YES
Maximum Voltage on any I/O Pin	5.8 V	V_{DD} + 3.6 V*
***Note:** Up to a maximum of 5.8 V		

The internal LDO regulator included on the C8051T60x family is used to regulate the V_{DD} supply voltage down to 1.7 V for the controller core. The regulated core voltage is only used internally. All external voltages and analog circuits on the device are powered from the V_{DD} voltage, so the I/O logic levels and the allowable ADC and comparator input ranges are all relative to V_{DD}. The LDO regulator can be bypassed if the external supply to the device is from a 1.8 V power source. The regulator output to the internal circuitry may also be turned off when the device enters Stop mode to conserve power.

6.1. In-System Code Development for Lower Voltage Systems (Below 2.7 V)

In a system which uses a 2.7–3.6 V supply voltage, the C8051F300 can be used as a substitute for the C8051T60x for in-system code development. However, if the supply voltage in the final system using the C8051T60x is less than 2.7 V, it may be necessary to add additional circuitry to the prototype board when using the C8051F300, to prevent damage to other devices in the system. Any required additional circuitry will be dictated by the specifications of the other circuits in the system:

1. If all of the other circuitry in the system can also operate at 2.7 V or higher, raising the supply voltage for the entire system during development is an easy solution.
2. If the other devices cannot operate at a higher supply voltage, but the I/O pins connected to the C8051F300 are tolerant of higher voltages, simply using a separate regulator for the C8051F300 may be an option.
3. If the I/O pins of the other devices in the system also cannot tolerate higher voltages, level-shifting circuitry may be necessary. An alternate to level shifters would be to use the C8051F300's outputs in open-drain mode, with an external pull-up resistor to the lower supply voltage. Be aware that this will result in extra supply current on the 'F300, as well as slower rise times of the C8051F300 output signals.

6.2. Special Considerations for Higher Voltage Systems (Above 3.6 V)

The C8051T60x devices can interface to logic levels that are higher than its supply voltage. However, special care must be taken in any system where the C8051T60x interfaces to logic that uses a supply voltage higher than 3.6 V. The C8051T60x I/O pins can only tolerate up to 3.6 V above the voltage present at the V_{DD} pin, or 5.8 V, whichever is lower. This means that when the device is powered off, and V_{DD} is 0 V, the maximum voltage on any I/O pin is 3.6 V. When VDD is 2.2 V or higher, the maximum voltage at any I/O pin is 5.8 V. It may be necessary to either control the order in which power supplies turn on in the system or add external protection circuitry to ensure that the pin voltages remain within tolerable limits at all times.

6.3. Regulator Control

The internal LDO regulator is an additional feature of the C8051T60x which is not found on the C8051F300. The special function register REG0CN is used to control some of the regulator's features. The REG0CN register is located at address 0xC7 in the C8051T60x devices. On the C8051F300, register location 0xC7 is not used, and it will not cause any harm to write to this location on a C8051F300 device. This allows the user to run code intended for a C8051T60x device on a C8051F300 device without modification.

If the V_{DD} supply voltage to the C8051T60x is regulated externally with a 1.8 V regulator, then the internal regulator

may be placed in bypass mode for power savings. The BYPASS bit in REG0CN can be set to '1' by firmware to use an external regulator. It is very important that the BYPASS function only be used if the external V_{DD} supply voltage is within the limits for "Voltage on V_{DD} / Regulator in Bypass Mode" specified in the "Absolute Maximum Ratings" table in the datasheet. The device will be damaged if bypass mode is used when V_{DD} is higher than this specification.

For slower clock frequencies (<2 MHz), the amount of current consumed by the device can be reduced by setting the Memory Power Control bit (MPCE) in register REG0CN to '1'. This setting allows the device to save power by only turning on the EPROM read buffers for the amount of time necessary to read the memory contents. Note that when this feature is enabled, the ADC should use a SAR clock divider of 2 or more for proper operation.

On the C8051T60x devices, there are two versions of Stop mode (as opposed to one on the C8051F30x). The normal Stop mode leaves the internal regulator on, and is identical to the C8051F30x Stop mode in that any reset source will be accepted and bring the device back out of Stop mode. The contents of RAM are retained if the regulator is left on in Stop mode. However, for additional power savings, the output of the regulator may also be disabled when the device enters Stop mode. Writing a '1' to the STOPCF bit in the REG0CN register enables this feature. Only a reset initiated by the /RST pin, a power-on reset, or a VDD monitor reset can wake the device from this mode. Note that the contents of RAM are lost if this option is used, since the power to the internal RAM is supplied by the regulator.

6.4. VDD Monitor / Brown-out Detector

The VDD monitor on the C8051T60x behaves in the same way as the VDD monitor for the C8051F300. The only difference is the threshold below which the VDD monitor will reset the device. The C8051F300's VDD monitor threshold is set to operate below 2.7 V, while the C8051T60x's VDD monitor threshold is set to operate below 1.8 V.

7. Clocking Options

The clocking options on the C8051T60x are very similar to those offered on the C8051F300. The only exception, as shown in Table 4, is that the C8051F300 includes an external crystal oscillator option, which is not available on the C8051T60x.

Table 4. Clocking Options

Feature	C8051F300	C8051T60x
Internal Calibrated 24.5 MHz Oscillator (divided by 1, 2, 4, or 8)	YES	YES
External CMOS clock (digital input)	YES	YES
External Oscillator in RC or Capacitor Mode	YES	YES
External Oscillator in Crystal Oscillator Mode	YES	NO

Because the external crystal oscillator option is not offered on the C8051T60x, any port of an existing C8051F300 design which relies on the precision of a crystal oscillator should be modified to use an external CMOS oscillator instead.

8. RAM Size

Most of the C8051T60x device family has 256 bytes of on-chip RAM like their Flash counterparts. The C8051T606 devices however, have only 128 bytes of on-chip RAM. When developing firmware for C8051T606 devices, the code should be restricted to only use the 128 bytes of directly-addressable memory (addresses 0x00 through 0x7F) which is on both the Flash and EPROM-based devices.

9. Available Port I/O Pins

The C8051F30x and the C8051T600/1/2/3/4/5 all include 8 GPIO pins that can be assigned to peripheral functions. The C8051T606 devices have only 6 of these I/O pins. Port pins P0.0 and P0.6 do not exist on the C8051T606 devices. When developing firmware for the C8051T606, care should be taken to restrict functions only to port I/O pins P0.1, P0.2, P0.3, P0.4, P0.5, and P0.7.

10. Other Peripherals

All other peripherals and features not discussed in the previous sections are functionally the same between the two device families. Code written for these peripherals will operate the same way on either device family, so there are no special considerations when developing code for the other peripherals.

11. Code Example

A simple code example is included in this section which highlights some of the differences in the ADC modules between the C8051F300 and the C8051T60x. In the code example, the ADC is configured to read the temperature sensor, average a number of samples together, and calculate the temperature from the result in tenths of degrees Celsius. The main differences to be noted are in the setup of the ADC and the calculation of temperature from the ADC result. Some portions of code are conditionally compiled based on the processor being used. There are two definitions for the processor selection: *C8051F300* and *C8051T600*. The code can be compiled for either processor by commenting out the opposite processor's definition.

To compile for the C8051F300:

```
#define C8051F300
//#define C8051T600
```

To compile for the C8051T600:

```
//#define C8051F300
#define C8051T600
```

There are only two places in code which use the above constants to conditionally compile code. These are in the ADC and VREF setup routine, and in the pre-compiler definitions for the temperature sensor offset and slope values.

11.1. ADC and VREF Setup Differences

On the C8051F300, the VDD supply is used as the ADC's voltage reference. It is assumed that the code is running on a board which supplies 3.3 V for the VDD supply. To obtain better resolution on the temperature sensor's output voltage, the gain of the ADC when compiling for the C8051F300 is set to 2, which configures the input range for the ADC for 0 to 1.65 V.

On the C8051T60x family of devices, the temperature sensor offset at 0 degrees Celsius has been calculated in production test under the condition that the ADC is using the internal 1.8 V regulator as VREF and the 1x gain range. This value is stored in the TOFFH and TOFFL registers in the C8051T60x devices. To take advantage of this pre-measured temperature sensor value, the internal 1.8 V regulated supply is used as the voltage reference and the 1X gain range is used. Thus, the input range for the ADC on the C8051T60x is 0 to 1.8 V.

11.2. Temperature Calculation Differences

The C8051F300 and C8051T60x temperature sensors have slightly different characteristics for offset and slope. When compiling the source code for the C8051F300, the offset and slope values are calculated by the pre-compiler and stored as constants. When compiling the source code for the C8051T60x, the slope value is calculated and stored as a constant, but the offset value is read from the TOFFH and TOFFL registers.

In both cases, the offset value used by the code represents the output of the ADC when measuring the temperature sensor at 0 degrees Celsius, to an accuracy of 10 bits. The slope value used by the code represents the temperature sensor slope per 100 degrees Celsius, also assuming a 10-bit ADC output word.

11.3. Example Code Listing

```
//-----------------------------------------------------------------------------
// T60x_ADC_TemperatureSensor.c
//-----------------------------------------------------------------------------
// Copyright (C) 2007 Silicon Laboratories, Inc.
//
// This program is intended to run on either a C8051F30x or C8051T60x device,
// and is an example of developing code on an 'F300 for the 'T60x.
//
// #if statements in the code are used to define necessary differences in the
// setup and operation of the two devices.
//
// This software prints the device die temperature out the hardware
// UART at 115200bps. The calibrated internal 24.5MHz oscillator is used as the
// system clock source.
//
// The ADC is configured to look at the on-chip temp sensor.  The sampling
// rate of the ADC is determined by the constant <SAMPLE_RATE>, which is given
// in Hz.
//
// The ADC0 End of Conversion Interrupt Handler retrieves the sample
// from the ADC and adds it to a running accumulator.  Every <INT_DEC>
// samples, the ADC updates and stores its result in the global variable
// <result>, which contains the accumulated result. The sampling technique of
// adding a set of values and decimating them (posting results every (n)th
// sample) is called 'integrate and dump.'  It is easy to implement and
// requires very few resources.
//
// For each power of 4 of <INT_DEC>, you gain 1 bit of noise resolution.
// For example, <INT_DEC> = 16 gains you 2 bits of resolution: 4^2 = 16.

// Target:         C8051F30x, C8051T60x
// Tool chain:     Keil C51 7.50 / Keil EVAL C51
// Command Line:   None
//
// Release 1.0
//    -Initial Revision (BD)
//    -7 FEB 2007
//
```

```
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <C8051T600_defs.h>            // SFR declarations
#include <stdio.h>

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

// Comment out one of the following two lines to indicate which device is used
//#define C8051F300
#define C8051T600


// ** Definitions for the C8051F30x family **
// Below are definitions to calculate basic temperature sensor parameters for
// the 'F30x device.
// When using the 'F30x family, VREF is set to VDD, and assumed to be 3.3V.
// The gain of the ADC in this example is set to 2x, so wherever VREF is used
// it will be divided by 2.
// The temperature sensor offset is typically 897 mV at 0 degrees C.
// The temperature sensor slope  is typically 3.35 mV / degree C.
//
// Based on the VREF voltage and the offset number, the temperature sensor
// offset <TOFF_LSB10> is calculated in output codes from the ADC.
// Based on the VREF voltage and the slope number, the temperature sensor
// slope is calculated in ADC output codes per 100 degrees C (to gain better
// accuracy with fixed-point mathematics).
// Although the ADC on the 'F30x devices is only 8 bits, 10-bit numbers are
// calculated here, to make the code more portable to the 'T60x.
#ifdef C8051F300
#define VREF_UV      3300000
#define TOFF_UV      897000
#define TSLOPE_UV    3350

#define TOFF_LSB10   ((((TOFF_UV)*1024L)/(VREF_UV/2L))<<6)
#define LSB10_P_100C ((100L*(TSLOPE_UV)*1024L)/(VREF_UV/2L))
#endif
// ** End definitions for the C8051F30x family **
```

```
// ** Definitions for the C8051T60x family **
// Below are definitions to calculate basic temperature sensor parameters for
// the 'T60x device.
// When using the 'T60x family, VREF is set to the internal 1.8V regulator.
// The gain of the ADC in this example is set to 1x.
// The temperature sensor slope  is typically 3.20 mV / degree C.
//
// The temperature sensor offset in the 'T60x is pre-measured under the VREF
// and ADC conditions above, and stored in the registers TOFFH and TOFFL.
// The temperature sensor offset <TOFF_LSB10> is simply set to read TOFF.
// Based on the VREF voltage and the slope number, the temperature sensor
// slope is calculated in ADC output codes per 100 degrees C (to gain better
// accuracy with fixed-point mathematics).
#ifdef C8051T600
#define VREF_UV      1800000
#define TSLOPE_UV    3200

#define TOFF_LSB10   (TOFF)
#define LSB10_P_100C ((100L*(TSLOPE_UV)*1024L)/(VREF_UV))
#endif
// ** End definitions for the C8051T60x family **


#define SYSCLK       24500000          // SYSCLK frequency in Hz
#define BAUDRATE     115200            // Baud rate of UART in bps
#define SAMPLE_RATE  10000             // Sample frequency in Hz

#define INT_DEC      256               // integrate and decimate ratio
#define ADD_BITS     4                 // ADD_BITS is calculated from INT_DEC
                                       // it should be equal to the number of
                                       // additional bits of noise resolution
                                       // obtained by averaging INT_DEC samples
                                       // 1 = 0 bit
                                       // 4 = 1 bit
                                       // 16 = 2 bits
                                       // 64 = 3 bits
                                       // etc.

sbit LED = P0^6;                       // LED='1' means ON
sbit SW1 = P0^3;                       // SW1='0' means switch pressed
```

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

void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer2_Init (int counts);
void ADC0_ISR (void);

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

long result;                            // ADC0 decimated value
bit new_temp_data = 0;

//-----------------------------------------------------------------------------
// 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

   // Disable Watchdog timer
   PCA0MD &= ~0x40;                     // WDTE = 0 (clear watchdog timer
                                        // enable)

   SYSCLK_Init ();                      // initialize oscillator
   PORT_Init ();                        // initialize crossbar and GPIO
   UART0_Init ();                       // initialize UART0
   Timer2_Init (SYSCLK/SAMPLE_RATE);    // initialize Timer2 to overflow at
                                        // <SAMPLE_RATE>

   ADC0_Init ();                        // init ADC0

  AD0EN = 1;                           // enable ADC0

   EA = 1;                              // Enable global interrupts
```

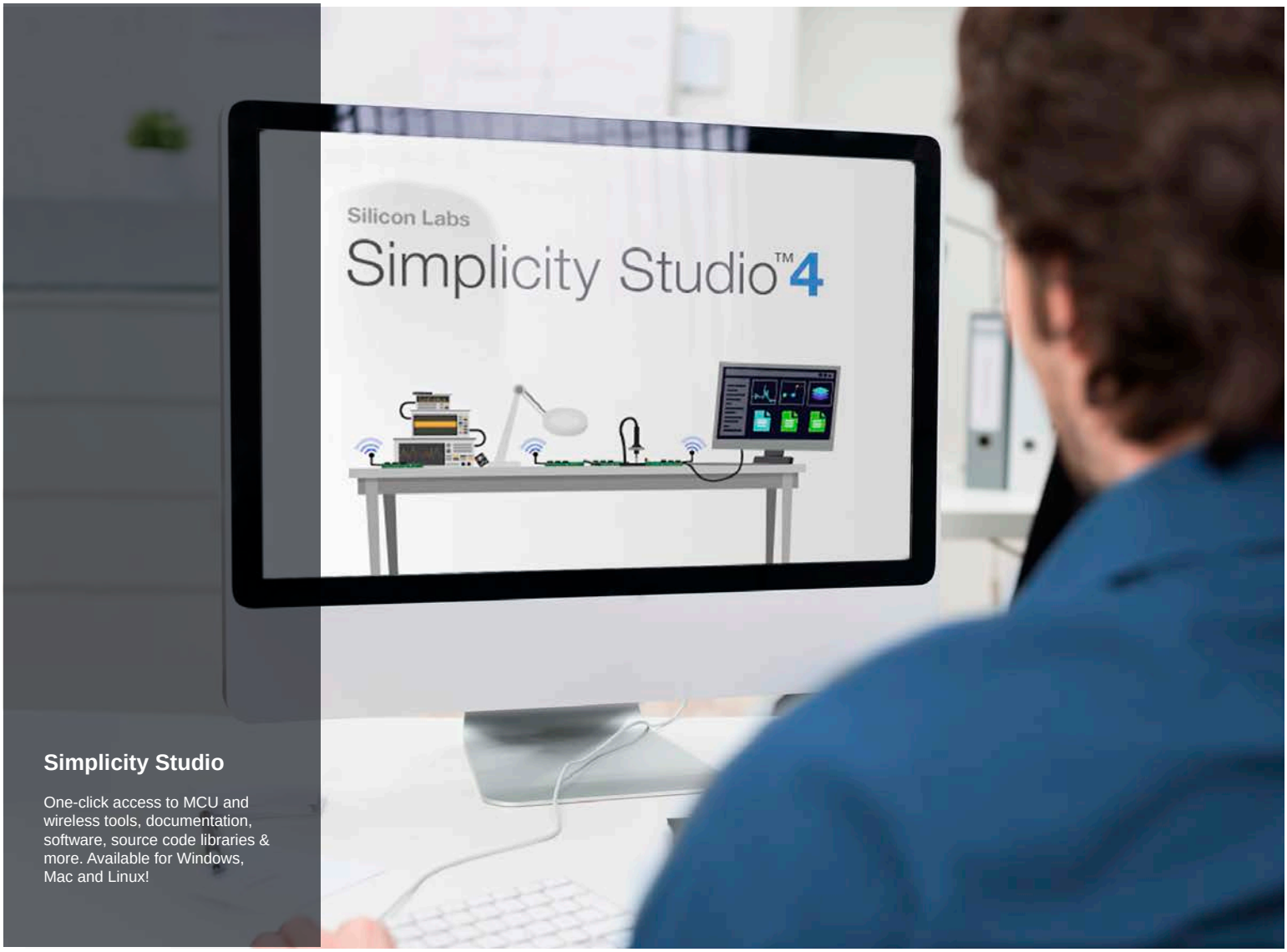
```
  while (1)
   {
      while (new_temp_data == 0);
      EA = 0;                           // disable interrupts
      // read result, and calculate temperature x 10.
      temperature = (result * 1000L) / (LSB10_P_100C << ADD_BITS);
      EA = 1;                           // re-enable interrupts

      new_temp_data = 0;

      // calculate temperature in tenths of a degree Celsius
      temp_int = temperature / 10;
      temp_frac = temperature % 10;
     printf ("Temperature is %+02d.%01d\n", temp_int, temp_frac);

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

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

//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// This routine initializes the system clock to use the internal 24.5MHz
// oscillator as its clock source.  Also enables missing clock detector reset.
//
void SYSCLK_Init (void)
{
   OSCICN |= 0x03;                      // configure internal oscillator for
                                        // its maximum frequency}
   RSTSRC = 0x04;                       // enable missing clock detector
}
```

```
//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports.
// P0.0 -
// P0.1 -
// P0.2 -
// P0.3 - SW1
// P0.4 - UART TX (push-pull)
// P0.5 - UART RX
// P0.6 - LED (push-pull)
// P0.7 - C2D
//
void PORT_Init (void)
{
   XBR0    = 0x48;                     // skip LED and Switch in crossbar
                                       // assignments
   XBR1    = 0x03;                     // UART0 TX and RX pins enabled
   XBR2    = 0x40;                     // Enable crossbar and weak pull-ups
   P0MDOUT |= 0x50;                    // enable TX0 and LED as a push-pull
                                       // outputs
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <BAUDRATE> and 8-N-1.
//
void UART0_Init (void)
{
   SCON0 = 0x10;                       // SCON0: 8-bit variable bit rate
                                       //        level of STOP bit is ignored
                                       //        RX enabled
                                       //        ninth bits are zeros
                                       //        clear RI0 and TI0 bits
   if (SYSCLK/BAUDRATE/2/256 < 1) {
      TH1 = -(SYSCLK/BAUDRATE/2);
      CKCON |= 0x10;                   // T1M = 1; SCA1:0 = xx
   } else if (SYSCLK/BAUDRATE/2/256 < 4) {
      TH1 = -(SYSCLK/BAUDRATE/2/4);
      CKCON |=  0x01;                  // T1M = 0; SCA1:0 = 01
```

```
      CKCON &= ~0x12;
   } else if (SYSCLK/BAUDRATE/2/256 < 12) {
      TH1 = -(SYSCLK/BAUDRATE/2/12);
      CKCON &= ~0x13;                  // T1M = 0; SCA1:0 = 00
   } else {
      TH1 = -(SYSCLK/BAUDRATE/2/48);
      CKCON |=  0x02;                  // T1M = 0; SCA1:0 = 10
      CKCON &= ~0x11;
   }

   TL1 = 0xff;                         // set Timer1 to overflow immediately
   TMOD |= 0x20;                       // TMOD: timer 1 in 8-bit autoreload
   TMOD &= ~0xD0;                      // mode
   TR1 = 1;                            // START Timer1
   TI0 = 1;                            // Indicate TX0 ready
}

//-----------------------------------------------------------------------------
// ADC0_Init
//-----------------------------------------------------------------------------
//
// Configure ADC0 to use Timer2 overflows as conversion source, to
// generate an interrupt on conversion complete, and to sense the output of
// the temp sensor.  Enables ADC end of conversion interrupt. Leaves ADC
// disabled.
//
void ADC0_Init (void)
{
   ADC0CN = 0x02;                      // ADC0 disabled; normal tracking
                                       // mode; ADC0 conversions are initiated
                                       // on overflow of Timer2;
   AMX0SL = 0xf8;                      // select temp sensor as mux input
   ADC0CF = (SYSCLK/4000000) << 3;     // ADC conversion clock <= 4MHz

#ifdef C8051F300
   ADC0CF |= 0x02;                     // PGA gain = 2
   REF0CN = 0x0e;                      // enable temp sensor, VREF = VDD, bias
                                       // generator is on.
#endif
```

```
#ifdef C8051T600
   ADC0CF |= 0x01;                     // PGA gain = 1
   REF0CN = 0x14;                      // enable temp sensor, VREF = Regulator
#endif

   EIE1 |= 0x04;                       // enable ADC0 EOC interrupt
}

//-----------------------------------------------------------------------------
// Timer2_Init
//-----------------------------------------------------------------------------
//
// Configure Timer2 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer2_Init (int counts)
{
   TMR2CN = 0x00;                      // STOP Timer2; Clear TF2H and TF2L;
                                       // disable low-byte interrupt; disable
                                       // split mode; select internal timebase
   CKCON |= 0x20;                      // Timer2 uses SYSCLK as its timebase

   TMR2RL  = -counts;                  // Init reload values
   TMR2    = 0xffff;                   // set to reload immediately
   ET2 = 0;                            // disable Timer2 interrupts
   TR2 = 1;                            // start Timer2
}

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

```
//-----------------------------------------------------------------------------
// ADC0_ISR
//-----------------------------------------------------------------------------
//
// ADC0 end-of-conversion ISR
// 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 INTERRUPT_ADC0_EOC using 3
{
   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

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

   // read ADC value and add to running total. Also subtract out offset now.
   accumulator += ((ADC0_16 - TOFF_LSB10)>>6);



   int_dec--;                          // update decimation counter

   if (int_dec == 0) {                 // if zero, then post result
      int_dec = INT_DEC;               // reset counter
      result = accumulator >> ADD_BITS;
      accumulator = 0L;                // reset accumulator
      new_temp_data = 1;
   }
}
```



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com