

# **MSC12xx Programming with SDCC: An Open-Source Alternative**

*Charles Repetti*  
*Russell Anderson*

*Sandpiper Associates, Inc.*  
*Data Acquisition Group – Microsystems*

## **ABSTRACT**

The MSC12xx is a “system on a chip” which embeds an 8051 microcontroller, a 24 bit analog-to-digital converter (ADC), 4-32K bytes of flash memory, and a set of peripherals in one package. Programming the 8051 is an important part of a complete product development effort. The “small device C compiler,” or SDCC [1] is one of the options for 8051 software development. SDCC is open source, and is free of charge. The entire tool flow runs under Microsoft Windows™.

## **Contents**

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
<b>2</b>	<b>Installation .....</b>	<b>2</b>
2.1	CYGWIN.....	3
2.2	SDCC .....	5
<b>3</b>	<b>Source Code Preparation.....</b>	<b>6</b>
3.1	C Source Code.....	6
3.2	Assembly Language .....	8
3.3	Libraries.....	8
<b>4</b>	<b>Running the “Hello World” Program .....</b>	<b>8</b>
4.1	Makefile .....	8
4.2	Downloading and Running .....	9
<b>5</b>	<b>Exercising the Analog to Digital Converter .....</b>	<b>10</b>
5.1	Using Floating-Point in SDCC.....	10
5.2	Using the ADC.....	11
5.3	Running the Program.....	13
<b>6</b>	<b>Conclusion.....</b>	<b>13</b>
<b>7</b>	<b>End Notes .....</b>	<b>14</b>

## **Figures**

<b>Figure 1, CYGWIN Setup. (Click on Default to change it to Install) .....</b>	<b>3</b>
<b>Figure 2, checking the CYGWIN/GCC installation .....</b>	<b>4</b>
<b>Figure 3, checking the SDCC installation.....</b>	<b>5</b>
<b>Figure 4, Running the Program.....</b>	<b>9</b>
<b>Figure 5, Running <i>adc.c</i>.....</b>	<b>13</b>

All trademarks are the property of their respective owners.

## 1 Introduction

SDCC is a compelling choice for software development on the MSC12xx for two primary reasons. First, it is available free of charge. More importantly, it is delivered with all of the source code for development tools and libraries, as well as with numerous helpful examples. This means that if a problem arises, the developer is able to fix it using the supplied source code.

In this application note, we note the differences between the formats of the source and object code TI supplies and the format that SDCC expects, and provide an automatic method for performing conversions between them. Finally, we run an example on an MSC12xxEVM.

## 2 Installation

Before beginning, it is a good idea to install and test the downloader TI ships on the CD supplied with the MSC12xxEVM [2]. We will discuss the downloader later in the process of executing our first program.

We first install *CYGWIN* [3], which runs as a client under Microsoft Windows. We install this as a binary, then install the GNU [4] compiler (known as *GCC*), and then use it to build *SDCC*. *SDCC* is bundled with “*ASXXXX*” and “*ASLINK*”, which are a freeware assembler and linker, respectively. While *SDCC* binaries are available for Microsoft Windows, we use a number of the other GNU tools in this application note.

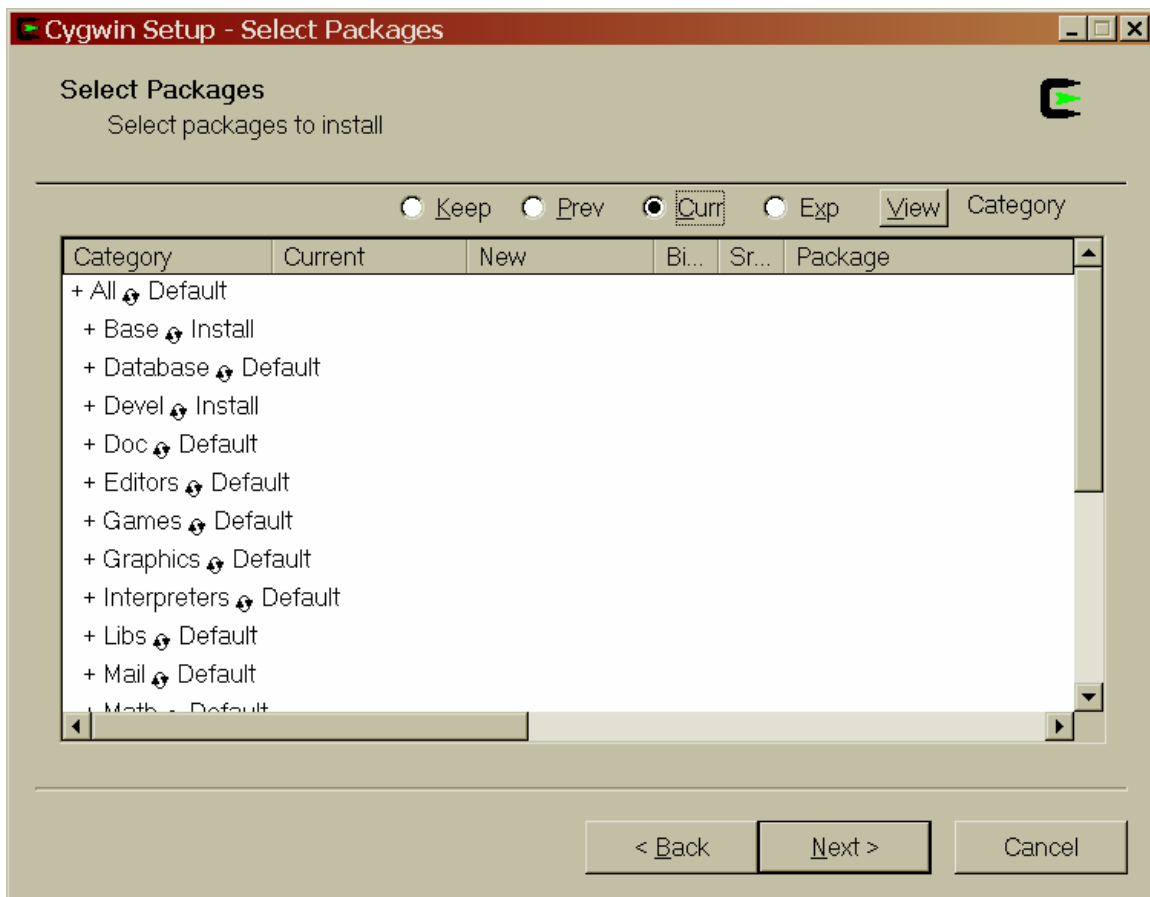
Microsoft Windows does not ship with any of the GNU tools installed, so a bit of work is required to simply prepare *CYGWIN*. This effort is conveniently accomplished with the advanced tools that are part of the *CYGWIN* package, but downloading them is time-consuming. *CYGWIN* is useful for more than development on the MSC12xx, so the effort may have additional benefits.

Beyond the base *CYGWIN* package, it is necessary to install the *CYGWIN* developer’s kit in order to get the *GCC* compiler. Then, *SDCC* itself must be obtained from Sourceforge.

Finally, the TI downloader must be installed. This may be obtained either from the MSC12xxEVM CD or from the TI web site mentioned in the End Notes.

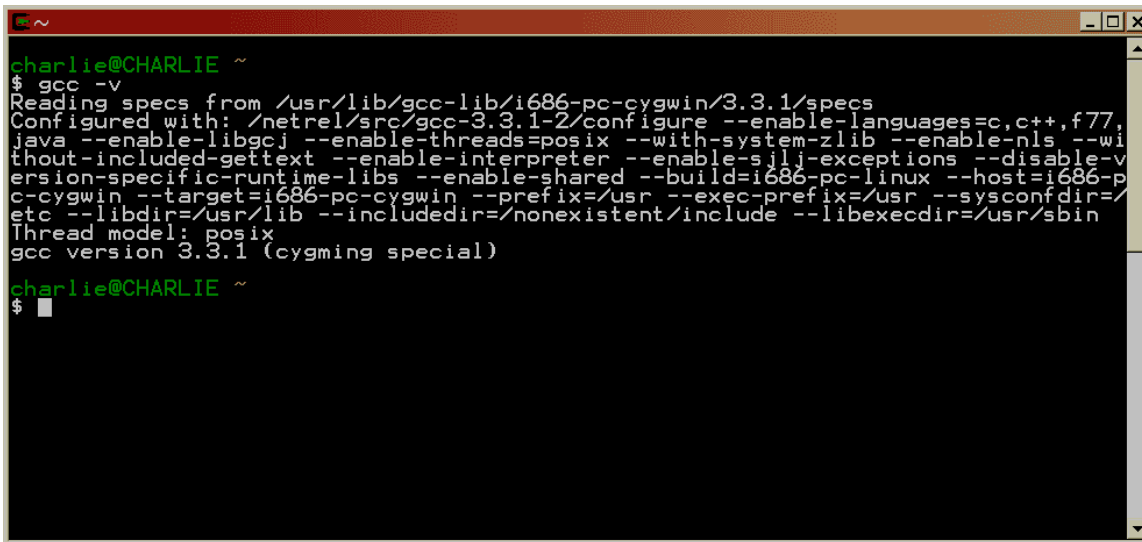
## 2.1 CYGWIN

The home page of CYGWIN, <http://www.cygwin.com>, is the place start. A program named “setup.exe” must be copied to your computer. This allows for CYGWIN components to be installed onto your computer. The “Base” category may be installed first, and the “Devel” category added next. If there are problems with timeouts over your network connection, it may be necessary to install the components piecemeal, or perhaps to download the components before installing them.



**Figure 1, CYGWIN Setup.** (Click on Default to change it to Install)

Once the CYGWIN base is installed and the development components are added, it should be possible to start a CYGWIN window on your computer. This window will be a command shell running a bash shell. Kernel calls map through to the CYGWIN DLL. Once the installation is complete, run a quick check of your work, as shown in Figure 2:



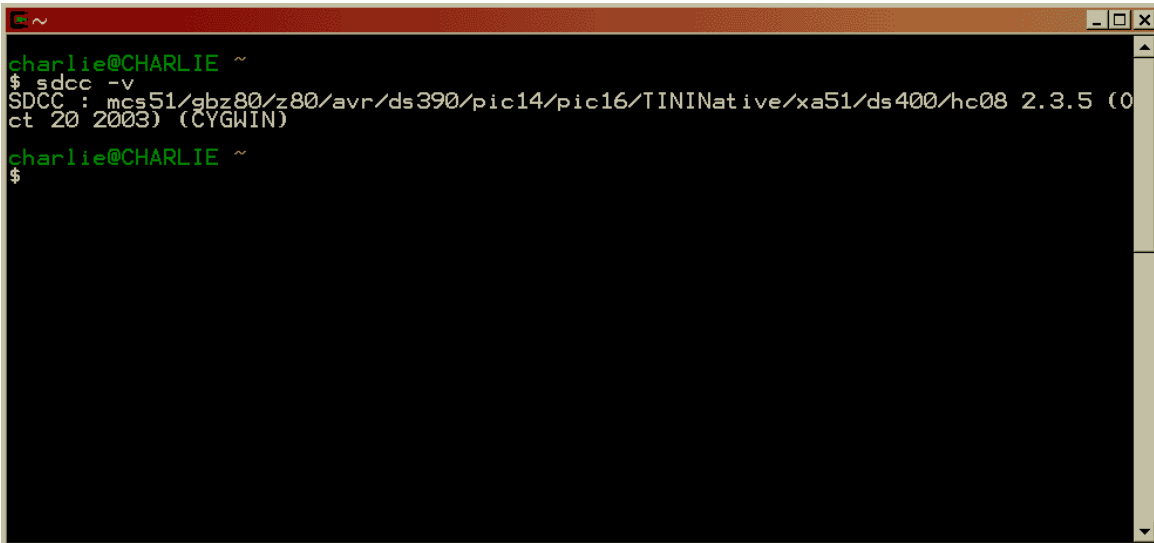
```
charlie@CHARLIE ~  
$ gcc -v  
Reading specs from /usr/lib/gcc-lib/i686-pc-cygwin/3.3.1/specs  
Configured with: /netrel/src/gcc-3.3.1-2/configure --enable-languages=c,c++,f77,  
java --enable-libgcj --enable-threads=posix --with-system-zlib --enable-nls --wi  
thout-included-gettext --enable-interpreter --enable-sjlj-exceptions --disable-v  
ersion-specific-runtime-libs --enable-shared --build=i686-pc-linux --host=i686-p  
c-cygwin --target=i686-pc-cygwin --prefix=/usr --exec-prefix=/usr --sysconfdir=  
etc --libdir=/usr/lib --includedir=/nonexistent/include --libexecdir=/usr/sbin  
Thread model: posix  
gcc version 3.3.1 (cygming special)  
charlie@CHARLIE ~  
$ █
```

Figure 2, checking the CYGWIN/GCC installation

Note that we have not only checked the fact that the CYGWIN shell runs, but we have also checked our GCC installation by starting the compiler as `gcc -v`. We had no trouble installing *X-Window* and running it as well, for those developers who want to use a graphically-based IDE.

## 2.2 SDCC

Even if you receive a version of SDCC with your CYGWIN release, you will still want to update SDCC. The version of SDCC we received with our CYGWIN did not work at all! As of this writing, the SDCC group is doing nightly builds, and we had no trouble with the latest stable release. Visit <http://sdcc.sourceforge.net/> and update your installation. Build SDCC from the source, using your GCC compiler. The steps to do this are clearly outlined in the SDCC documentation.



```

charlie@CHARLIE ~
$ sdcc -v
SDCC : mcs51/gbz80/z80/avr/ds390/pic14/pic16/TININative/xa51/ds400/hc08 2.3.5 (Oct 20 2003) (CYGWIN)
charlie@CHARLIE ~
$
  
```

Figure 3, checking the SDCC installation

Once done, check your work by exercising the compiler at the command line. Again, note that versions of SDCC prior to 2.3.5 will probably not work.

### 3 Source Code Preparation

SDCC accepts ANSI C, and includes a number of language extensions specific to the 8051. While the syntax of SDCC is slightly different than the one used by TI in its examples, it is a simple matter to automatically translate from TI "C" to SDCC "C."

#### 3.1 C Source Code

In order to compile programs for the MSC12xxEVM, it is helpful to have "msc12xx.h" available for complete register definitions. Unfortunately, some TI files use slightly different language extensions for special register and special bit definitions. Some of these cannot be remedied with "#define" style macros. For example, a TI header file might read as:

```
/*--
MSC12xx.H
Header file for TI MSC12xx microcontroller.
All rights reserved.
--*/
sfr P0    = 0x80;
sbit TF1 = TCON^7;
```

**List 1, MSC12xx.h**

This is not immediately useful for SDCC both because the syntax of the "sfr" and "sbit" language extensions is ordered differently, and because the SDCC preprocessor does not accept bit level operators for "sbit." But since we are using Linux, we have many easy ways to fix this. Rather than rewrite the compiler to accept TI's syntax, which might take some effort, we simply write a quick preprocessor to do the trick. We used Perl [5]:

```
#!/usr/bin/perl

# Perl script to translate TI 8051 Headers for use with SDCC

# Open the header and treat each line one at a time
open(FH, "msc12xx.h") || die "No File $!";
while(<FH>) {

    # if we find a "sfr" definition, save it for later
    if (/^sfr(16)?\s+(\S+)\s+=\s+(\S+);/) {
        print "sfr at $3 $2;\n";
        $sfr{$1} = $2;
    }

    # if we have an "sbit" as a "sfr" definition, use our record
    } elsif (/^sbit\s+(\S+)\s+=\s+([\^^\s]+)\s+(\S+);/) {
        printf("sbit at 0x%X %s;\n", hex($sfr{$2})+hex($3), $1);
    }

    # Any other lines are just echoed
    } else {
        print $_;
    }
}
close(FH)
```

**List 2, SDCC\_header.pl**

This program is included, and as supplied simply dumps its output to the screen. Invoking it as *parse.pl >ti.h*, though, would write a new file named *ti.h* with the proper definitions. The above header might translate as follows:

```

/*--
MSC12xx.H
Header file for TI MSC12xx microcontroller.
All rights reserved.
--*/
sfr at 0x80 P0;
sbit at 0x88 TFL;

```

**List 3, ti.h**

We copy this file to the SDCC include directory. Now we write up a quick “Hello World” program in “C”:

```

//
// Copyright 2002 Texas Instruments
//
// MSC12xx Hello World Program
#include <8052.h>
#include <ser.h>
#include <stdio.h>

extern void autobaud(void);

// We define a "putchar" for "printf" to use
void putchar(char ch) {
    ser_putc(ch);
}
// Begin the program...
void main(void) {

    // Press <Enter> for auto baudrate adjust
    autobaud();

    // SDCC requires these setup calls
    ser_init();
    EA=1;

    // Print to the terminal
    printf("Hello World\r\n");
}

```

**List 4, HelloWorld.c**

Note that the autobaud that we use is from the SDCC library, and not the TI-supplied ROM. Regardless of this, the autobaud functions the same way, timing the signal wavelength on the input channel and setting the baud rate on the 8051 appropriately. Also, note that we used the SDCC serial library, being careful to enable interrupts with EA=1 before trying to write to the port.

We also make use of the standard C library, as implemented by SDCC. By defining a *putchar* which simply writes to our terminal, we may now use the library function *printf* (which resolves, after formatting, to our *putchar*) to write a message to the host screen.

## 3.2 Assembly Language

SDCC's output is a well commented assembly language listing. This listing can be used for reference. Also, the ASXXX assembler can accept hand written code.

The command **asx8051 -los myfile.asm** will assemble *myfile.asm* to *myfile.rel*. The output file may be linked along with any output of SDCC to form a program downloadable to the MSC12xxEVM.

## 3.3 Libraries

The entire source listing for the SDCC libraries is included with the distribution. The area of library transparency and adaptability is one where the open-source movement shows its clear advantages. Tools for creating one's own libraries are also included.

Keep in mind that it is considered good practice to contribute any useful code you might produce back to the SDCC project.

## 4 Running the "Hello World" Program

All that remains is to run the program. As we mentioned, it is necessary to install the TI downloader. Instructions for doing this are included on the CD which was shipped with your MSC12xxEVM. It is a good idea to test the installation with a TI-supplied example before approaching SDCC.

### 4.1 Makefile

There is one last foible of our tool chain we must address. A CYGWIN file uses a different *end of line* sequence than does a Microsoft Windows file. This is such a common nuisance that a utility to remedy the issue is included with CYGWIN. It is the *unix2dos* command that is included in the *make file*. *Make* is a utility that has been around for years, and is a favored method for automating a tool chain's flow:

```
all:      HelloWorld.ihx

HelloWorld.ihx: HelloWorld.rel
    sdcc HelloWorld.rel
    unix2dos HelloWorld.ihx
    download /FHelloWorld.ihx /X11 /P1 /T /B19200

HelloWorld.rel: HelloWorld.c
    sdcc -c HelloWorld.c

clean:
    rm *.rel
```

**List 5, Makefile**



## 4.2 Downloading and Running

Typing the *make* command automatically looks for a file in the current directory named *Makefile*, in the current directory, as shown in List 5, and follows the rules it contains. This should result in the building and running of the *HelloWorld* program we have been discussing.

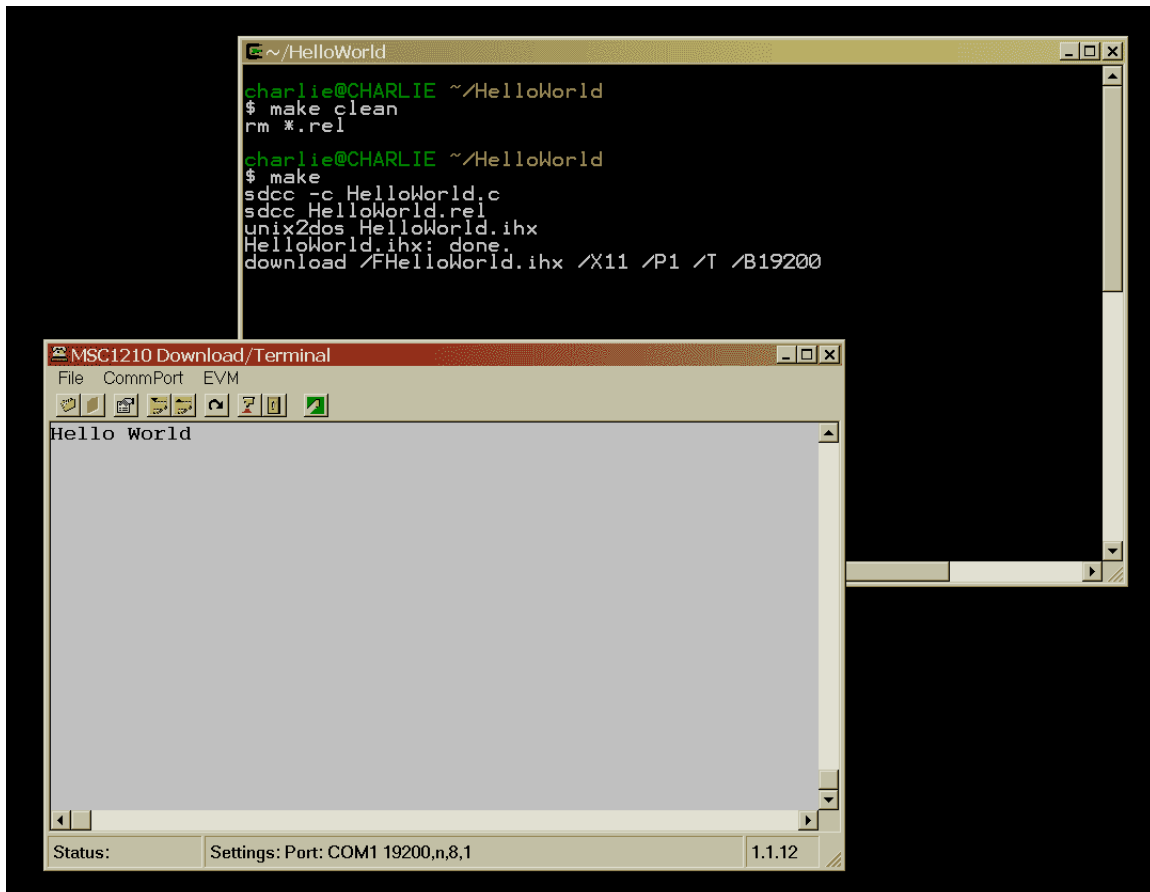


Figure 4, Running the Program

Figure 4 shows the screen after the TI downloader has run, and you have pressed *Enter* on your keyboard. The *HelloWorld* shows its output in the terminal screen, demonstrating our success.

## 5 Exercising the Analog to Digital Converter

The Analog-to-Digital Converter (ADC) is the heart of the MSC12xx . Before we set up and use the ADC, though, we want to be sure we can print the results. We also want to be able to easily analyze our results without overflow or underflow problems, and thus we need to do a little work first.

### 5.1 Using Floating-Point in SDCC

SDCC includes floating point support by default. In order to save memory in those applications which do not print floating point numbers, SDCC currently ships with the floating point support for the *printf* library function turned off. This setting may be changed with the compile-time argument *USE\_FLOATS*.

First, the correct *Makefile* for the SDCC libraries must be located. There are a few files named *Makefile* in the SDCC directories. The correct *Makefile* is located in the *device* subdirectory, usually found at */usr/share/sdcc/device/lib/Makefile*. In that file is a variable named *CFLAGS*. This variable must be changed to include *-DUSE\_FLOATS=1*. This might result in an edit such as the following:

```
lib_dir_suffix = sdcc/lib
sdcc_libdir    = $(DESTDIR)$(datadir)/$(lib_dir_suffix)
CPPFLAGS      = -I$(INCDIR) -I$(PORTINCDIR)
CFLAGS        = $(MODELFLAGS) --nostdinc -DUSE_FLOATS=1
```

**List 6, SDCC *device/lib/Makefile***

Once this edit is complete, a */usr/src/sdcc/make clean* command should be issued, followed new */usr/src/sdcc/make* and */usr/src/sdcc/make install* commands.

## 5.2 Using the ADC

Now we can write a “C” program which will read the ADC and output the results to the terminal:

```
#include <ti.h>
#include <stdio.h>
#include <ser.h>

// We will use these to scale our ADC's
#define VREF 2.5
#define GND 0.0
#define FULL_SCALE 16777215.0

// function declarations
extern void      autobaud      ( void );
extern unsigned long  unipolar  ( void );
extern void      calibrate    ( void );
// global variables for convenience
// note: SDCC places automatics on the heap,
// unless keyword "reentrant" is used
int i, n = 0, decimation = 1920, samples = 10;
unsigned long adc, adc_buffer;
float volts;

// Our entry point
void main ( void ) {

    // Set up the terminal
    autobaud();
    ser_init();
    EA=1;

    // Prepare for ADC use
    calibrate();

    // Go in circles...
    while (1) {

        // Take a few samples for noise free operation
        adc = 0;
        for ( i = 0 ; i < samples ; i++ ) {

            // wait for the next result
            while (!(AIE&0x20));

            // accumulate the results quickly, without overflow,
            // but with a bias against older samples.
            adc_buffer = unipolar() >> 1;
            adc = (adc >> 1) + adc_buffer;
        }

        // calculate the exact voltage we are seeing...
        volts = ((float) adc / FULL_SCALE) * (VREF - GND);

        // ...and print out the results
        printf("Sample %3d: V=%f      \r", ++n, volts);
    }
}
```

```

//
// Set up the ADC and throw away a few a samples as the device settles
//
void calibrate ( void ) {

    printf ("Unipolar mode (AIN0)\r\n");

    // Timer Setup
    USEC    = 10;                // 11 MHz Clock
    ACLK    = 8;                // ACLK = 11,059,000 / 9 = 1.2288MHz
                                // modclock = 1.2288MHz / 64 = 19,200 Hz

    // Setup ADC
    PDCON  &= 0xF7;            // turn on ADC
    ADMUX  = 0x08;            // Select Analog Channel 1

    // VRefOn, VRef Hi, Burnout Detect Off, PGA = 1
    ADCON0 = 0x30;

    // unipolar auto, self calibration, offset, gain
    ADCON1 = 0x41;
    // Note that if decimation is too low, noise will appear...
    ADCON2 = decimation & 0xFF;    // LSB of decimation
    ADCON3 = (decimation>>8) & 0x07; // MSB of decimation

    printf ( "Calibrating. . .\r\n");

    for ( i = 0; i<4; i++ ) {

        // Wait for four conversions for filter to settle after calibration
        while (!(AIE & 0x20));

        // dummy read to clear ADCIRQ
        unipolar();
    }
}

//
// This is for unsigned 24 bit results
//
unsigned long unipolar ( void )
{
    // Storage for in-place conversion
    union {
        char BitMap[4];
        long LongInteger; } DoubleWord ;

    // Move the register contents piecemeal to "little endian" "long" storage
    DoubleWord.BitMap[3] = 0x00;
    DoubleWord.BitMap[2] = ADRESH;
    DoubleWord.BitMap[1] = ADRESM;
    DoubleWord.BitMap[0] = ADRESL;

    // return the result as a "long"
    return DoubleWord.LongInteger;
}

//
// As in "HelloWorld", we need this for "printf"
//
void putchar(char ch) {
    ser_putc(ch);
}

```

### List 7, adc.c

### 5.3 Running the Program

With the MSC12xxEVM powered up, ground AGND, and place a voltage source across ANC and AIN0 of less than 2.5 Volts. Now, using a *Makefile* like the one we used for *Hello World*, we build and run *ADC*. The result should be a screen like the one we saw for *HelloWorld*, but with a precise read on the voltage we place on the input pin of our msc12xxEVM showing on the terminal screen.

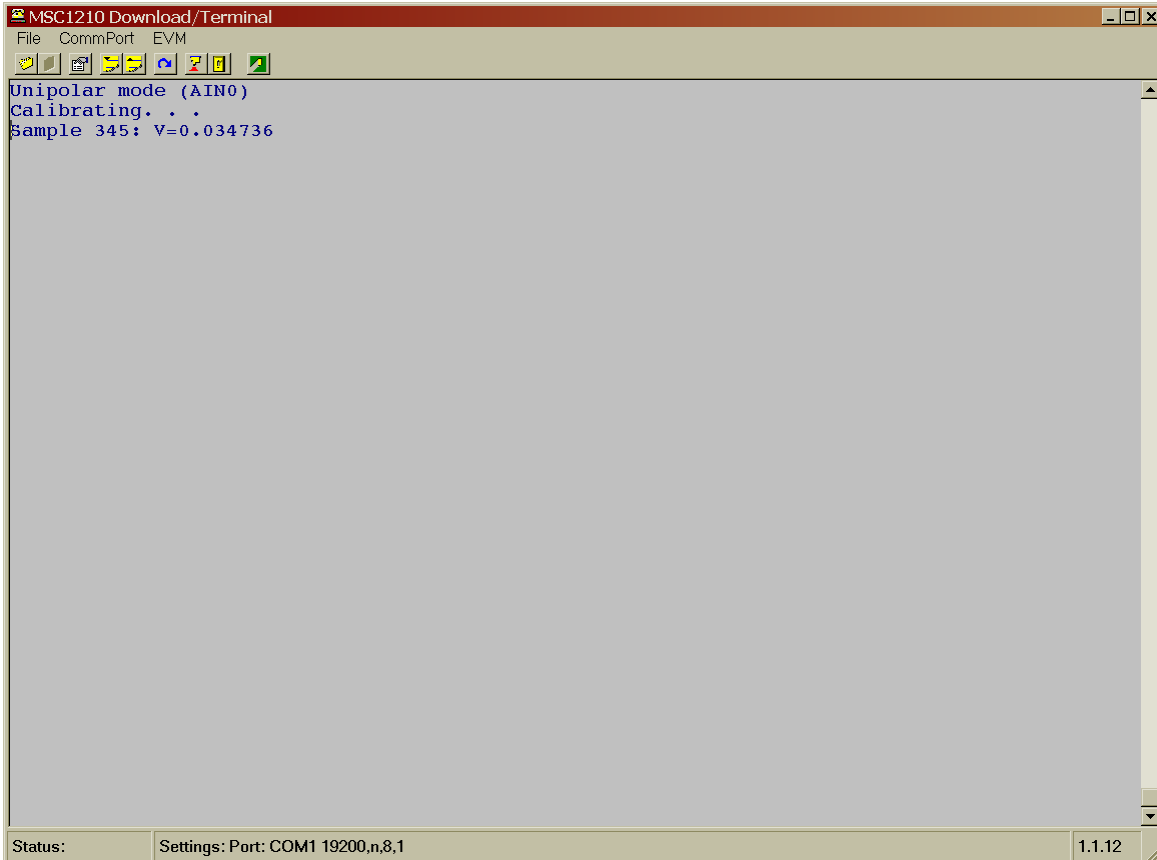


Figure 5, Running *adc.c*

## 6 Conclusion

SDCC is a viable option for software development on the TI MSC12xx family of microsystem controllers. The open-source software is available at no cost. Extending the Libraries is always an option, and the developer has the added security of knowing that software tool reliability remains under his or her control.

## 7 End Notes

---

- [1] SDCC is currently maintained by the Source Forge at <http://sdcc.sourceforge.net/>.
- [2] <http://www-s.ti.com/sc/psheets/sbac018a/sbac018a.zip> is a location for obtaining the TI downloader.
- [3] The home page for Cygwin is <http://www.cygwin.com>.
- [4] Important information is available at <http://www.gnu.org>. Note that SDCC and CYGWIN are also released under the GPL.
- [5] Perl is a widely used for text processing, particularly for Internet applications. It is included in the CYGWIN distribution. More information is available at <http://www.perl.com>, as well as on many other Internet sites.

**IMPORTANT NOTICE**

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

**Products**

Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>

**Applications**

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Broadband	<a href="http://www.ti.com/broadband">www.ti.com/broadband</a>
Digital Control	<a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a>
Military	<a href="http://www.ti.com/military">www.ti.com/military</a>
Optical Networking	<a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Telephony	<a href="http://www.ti.com/telephony">www.ti.com/telephony</a>
Video & Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
Wireless	<a href="http://www.ti.com/wireless">www.ti.com/wireless</a>

Mailing Address: Texas Instruments  
 Post Office Box 655303 Dallas, Texas 75265  
 Copyright . 2003, Texas Instruments Incorporated