Programming Firmware using the Small Device C Compiler (SDCC)

PRESENTED BY RICHARD GOWEN (@alt_bier)

Created for BSidesDFW 2020 HHV

This Slide Deck Is Available at https://altbier.us/SDCC/

What is SDCC?

- SDCC is a retargettable, optimizing Standard C (ANSI C89, ISO C99, ISO C11) compiler suite
- It targets the Intel MCS51 based microprocessors (8031, 8032, 8051, 8052, etc.), Maxim (formerly Dallas) DS80C390 variants, Freescale (formerly Motorola) HC08 based (hc08, s08), Zilog Z80 based MCUs (z80, z180, gbz80, Rabbit 2000/3000, Rabbit 3000A, TLCS-90), Padauk (pdk14, pdk15) and STMicroelectronics STM8.
- Information and downloads: `http://sdcc.sourceforge.net/`

Download and Install

SDCC is available for several platforms including Windows, MacOS, and Linux.

You can download the version for your platform from SourceForge:
`https://sourceforge.net/projects/sdcc/files/`

In the case of Windows, this download will be an installation .exe file.

In the case of Linux, this download will be a tarball containing the binaries and an installation script that needs to be extracted and run.

For Linux the binaries may be available within your platform's application repositories (e.g. `apt install sdcc`).

Download Supporting Packages

Depending upon your operating system you may need some additional software packages to work effectively with SDCC.

For Linux, you are going to want the build-essential package of tools by whichever name you flavor of linux calls them (e.g. `apt install build-essential`).

For Windows, the support tools you will use will depend upon how you work with code on your system. I tend to work in a Linux like environment when coding on my Windows system, and here are the packages I use:

Git for Windows: `https://gitforwindows.org/`

MinGW - Minimalist GNU compiler: `https://sourceforge.net/projects/mingw/`

Make sure you set your PATH environment variable within the Git for Windows environment so it can find SDCC and make tools

```
# SDCC compiler tools
export PATH=$PATH:/c/Program\ Files/SDCC/bin


# Mingw tools (for Make)
export PATH=$PATH:/c/MinGW/bin


alias make=mingw32-make.exe
```

Download IC Chip SDK & Toolchain

Depending upon which chip you are working with you will need to download its software development kit (SDK) and its toolchain for firmware deployment.

Since different chips have different hardware specification that are called out in SDK libraries and have differing methods of firmware upload it is important to research the correct products for the chip you are working with.

For this presentation we will be using the CH552G chip in our examples. This IC chip uses an MCS51 series E8051 core processor which is compatible with SDCC.

The CH55x SDK for this IC is available from Blinkinlabs who ported the Keil SDK to SDCC: **https://github.com/Blinkinlabs/ch554_sdcc/**

Loading the firmware onto the chip can be done using the official WCH program WCHISPTool (Windows) **http://www.wch.cn/cn/** (service -->downloads) or via multiple open-source tools such as:

- LibreCH551 **https://github.com/rgwan/librech551**
- ch552tool **https://github.com/MarsTechHAN/ch552tool**

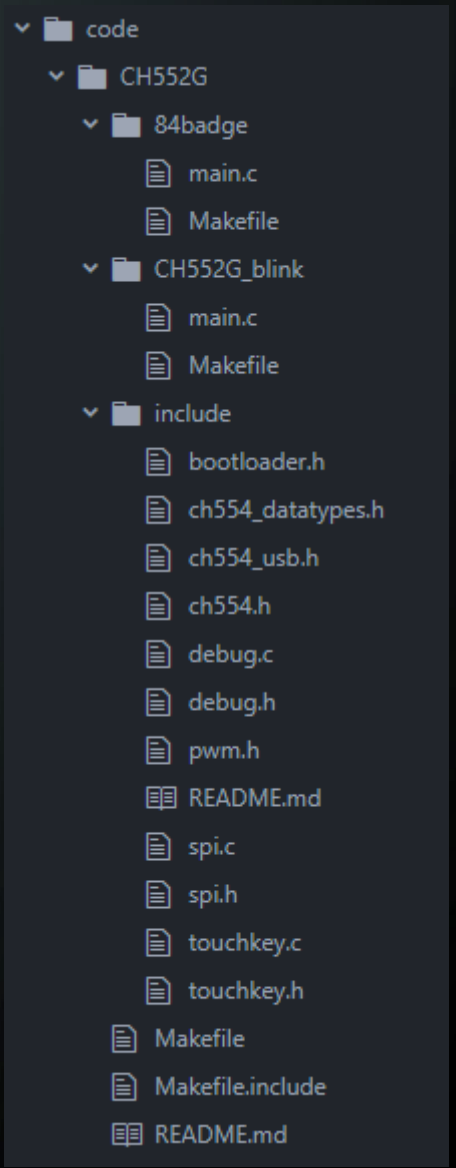
Code File Organization

While code file organization is a matter of the programmer's taste, I tend to keep the IC specific SDK libraries in their own directory and include them from Make files.

You can see from the file list on the right that the include directory holds my SDK files including the chip specific headers and two different program main.c files that are each in their own directory.

There is a main Makefile that can compile everything, or I can use the Makefile in each program to compile only its code.

There is a common Makefile.include that sets up the toolchain parameters.



The Makefile

The Makefile for my firmware programs are simple.

Each code directory has a Makefile that will name a target, call main.c, include debug.c from the SDK, and include the toolchain Makefile.include

The Makefile in project directory will reach into each subdirectory and execute the Makefile there.

Makefile

```
TARGET = blink

C_FILES = \
  main.c \
  ../include/debug.c

pre-flash:


include ../Makefile.include

```

Makefile

```
SUBDIRS = $(shell ls -d */)

.PHONY: $(SUBDIRS)


$(SUBDIRS):
  $(MAKE) -C $@

.DEFAULT_GOAL := all
all: $(SUBDIRS)


print-%  : ; @echo $* = $($*)

```

The Toolchain Makefile.include

The Toolchain Makefile.include was provided by the SDK.

It sets up our CC as SDCC and the WCHISP tool as well as various parameter settings.

This should be reviewed for the system you are working on and updated as needed.

Makefile.include

```
#######################################################

# toolchain
CC = sdcc
OBJCOPY = objcopy
PACK_HEX = packihx
WCHISP = wchisptool

#######################################################

ifndef FREQ_SYS
FREQ_SYS = 16000000
endif

ifndef XRAM_SIZE
XRAM_SIZE = 0x0400
endif

ifndef XRAM_LOC
XRAM_LOC = 0x0000
endif

ifndef CODE_SIZE
CODE_SIZE = 0x2800
endif

ROOT_DIR := $(dir $(abspath $(lastword $(MAKEFILE_LIST))))

CFLAGS := -V -mmcs51 --model-small \
  --xram-size $(XRAM_SIZE) --xram-loc $(XRAM_LOC) \
  --code-size $(CODE_SIZE) \
  -I$(ROOT_DIR)/include -DFREQ_SYS=$(FREQ_SYS) \
  $(EXTRA_FLAGS)

LFLAGS := $(CFLAGS)

RELS := $(C_FILES:.c=.rel)

print-%  : ; @echo $* = $($*)
```

The main.c

The main.c file for the firmware program will include the SDK header files necessary for what it is using.

In this example I am including ch554.h and debug.h from the SDK.

These SDK headers will provide the chip specific definitions that we will use in our code.

In this example I am calling P3_DIR_PU which is defined in the ch554.h header that we included.

main.c

```
// Blink LEDs alternating between groups
// LEDs connected to pins 1.4, 1.5, 1.6, 1.7, 3.0, 3.1,
// Tactile switch connected to pin 3.4

#include <ch554.h>
#include <debug.h>

#define LED30_PIN 0
#define LED31_PIN 1
#define LED32_PIN 2
#define LED33_PIN 3
#define LED14_PIN 4
#define LED15_PIN 5
#define LED16_PIN 6
#define LED17_PIN 7

// P1 = 0x90 P3 = 0xB0
SBIT(LED30, 0xB0, LED30_PIN);
SBIT(LED31, 0xB0, LED31_PIN);
SBIT(LED32, 0xB0, LED32_PIN);
SBIT(LED33, 0xB0, LED33_PIN);
SBIT(LED14, 0x90, LED14_PIN);
SBIT(LED15, 0x90, LED15_PIN);
SBIT(LED16, 0x90, LED16_PIN);
SBIT(LED17, 0x90, LED17_PIN);

void main() {
    CfgFsys();

    P3_DIR_PU &= 0x0C;
    // Configure pin 3.0 as GPIO output
    P3_MOD_OC = P3_MOD_OC & ~(1<<LED30_PIN);
    P3_DIR_PU = P3_DIR_PU | (1<<LED30_PIN);
    // Configure pin 3.1 as GPIO output
    P3_MOD_OC = P3_MOD_OC & ~(1<<LED31_PIN);
    P3_DIR_PU = P3_DIR_PU | (1<<LED31_PIN);
    // Configure pin 3.0 as GPIO output
    P3_MOD_OC = P3_MOD_OC & ~(1<<LED32_PIN);
    P3_DIR_PU = P3_DIR_PU | (1<<LED32_PIN);
```

Compile

When you are ready to compile the code run make from the code directory.

```
richa@DatA55 MINGW64 ~/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/code/CH552G/CH552G_blink (master)
$ ll
total 5
-rw-r--r-- 1 richa 197609 2278 Oct  4  2019 main.c
-rw-r--r-- 1 richa 197609  110 Oct  9 08:31 Makefile

richa@DatA55 MINGW64 ~/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/code/CH552G/CH552G_blink (master)
$ make
sdcc -c -V -mmcs51 --model-small --xram-size 0x0400 --xram-loc 0x0000 --code-size 0x2800 -IC:/Users/richa/OneDrive/BSide
sDFW2020/BSidesDFW_2020_HHV/code/CH552G//include -DFREQ_SYS=16000000  main.c
+ C:\PROGRA~1\SDCC\bin\sdcpp.exe -nostdinc -Wall -std=c11 -I"C:/Users/richa/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/co
de/CH552G//include" -D"FREQ_SYS=16000000" -obj-ext=.rel -D__SDCC_CHAR_UNSIGNED -D__SDCC_MODEL_SMALL -D__SDCC_FLOAT_REENT
 -D__SDCC=3_9_0 -D__SDCC_VERSION_MAJOR=3 -D__SDCC_VERSION_MINOR=9 -D__SDCC_VERSION_PATCH=0 -DSDCC=390 -D__SDCC_REVISION=
11195 -D__SDCC_mcs51 -D__STDC_NO_COMPLEX__=1 -D__STDC_NO_THREADS__=1 -D__STDC_NO_ATOMICS__=1 -D__STDC_NO_VLA__=1 -D__STD
C_ISO_10646__=201409L -D__STDC_UTF_16__=1 -D__STDC_UTF_32__=1 -isystem "C:\Program Files\SDCC\bin\..\include\mcs51" -isy
stem "C:\Program Files\SDCC\bin\..\include"  "main.c"
+ C:\PROGRA~1\SDCC\bin\sdas8051.exe -plosgffw "main.rel" "main".asm
sdcc -c -V -mmcs51 --model-small --xram-size 0x0400 --xram-loc 0x0000 --code-size 0x2800 -IC:/Users/richa/OneDrive/BSide
sDFW2020/BSidesDFW_2020_HHV/code/CH552G//include -DFREQ_SYS=16000000  ../include/debug.c
+ C:\PROGRA~1\SDCC\bin\sdcpp.exe -nostdinc -Wall -std=c11 -I"C:/Users/richa/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/co
de/CH552G//include" -D"FREQ_SYS=16000000" -obj-ext=.rel -D__SDCC_CHAR_UNSIGNED -D__SDCC_MODEL_SMALL -D__SDCC_FLOAT_REENT
 -D__SDCC=3_9_0 -D__SDCC_VERSION_MAJOR=3 -D__SDCC_VERSION_MINOR=9 -D__SDCC_VERSION_PATH=0 -DSDCC=390 -D__SDCC_REVISION=
11195 -D__SDCC_mcs51 -D__STDC_NO_COMPLEX__=1 -D__STDC_NO_THREADS__=1 -D__STDC_NO_ATOMICS__=1 -D__STDC_NO_VLA__=1 -D__STD
C_ISO_10646__=201409L -D__STDC_UTF_16__=1 -D__STDC_UTF_32__=1 -isystem "C:\Program Files\SDCC\bin\..\include\mcs51" -isy
stem "C:\Program Files\SDCC\bin\..\include"  "../include/debug.c"
../include/debug.c:270: warning 158: overflow in implicit constant conversion
+ C:\PROGRA~1\SDCC\bin\sdas8051.exe -plosgffw "debug.rel" "debug".asm
sdcc main.rel debug.rel -V -mmcs51 --model-small --xram-size 0x0400 --xram-loc 0x0000 --code-size 0x2800 -IC:/Users/rich
a/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/code/CH552G//include -DFREQ_SYS=16000000  -o blink.ihx
+ C:\PROGRA~1\SDCC\bin\sdld.exe -nf "blink.lk"
objcopy -I ihex -O binary blink.ihx blink.bin
packihx blink.ihx > blink.hex
packihx: read 34 lines, wrote 54: OK.

richa@DatA55 MINGW64 ~/OneDrive/BSidesDFW2020/BSidesDFW_2020_HHV/code/CH552G/CH552G_blink (master)
$
```

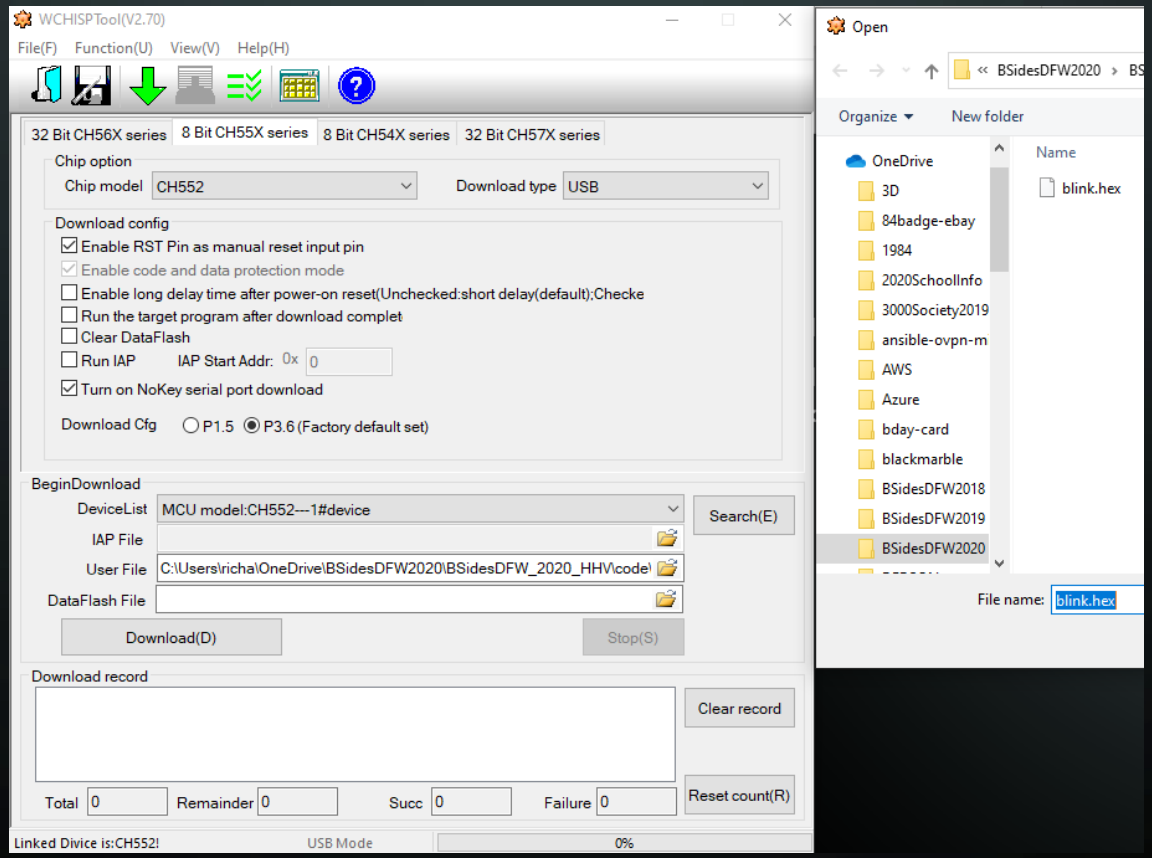
```
main.c
Makefile
```

```
blink.bin
blink.hex
blink.ihx
blink.lk
blink.map
blink.mem
debug.asm
debug.lst
debug.rel
debug.rst
debug.sym
main.asm
main.c
main.lst
main.rel
main.rst
main.sym
Makefile
```

Upload Firmware to Chip

If your code compiled without error, you are ready to upload it to your chip.

In my case I will be using the WCHISPTool which takes the .hex file and uploads it to the CH552G chip via a USB connection.



THANK YOU

I hope you enjoyed this presentation and learned something from it.

-- @alt_bier

This Slide Deck – https://altbier.us/SDCC/

Code – https://github.com/gowenrw/BSidesDFW_2020_HHV/