University of British Columbia
Electrical and Computer Engineering

CV-8052 Soft Processor in the DE1-SoC Board: Getting Started Guide

Copyright © 2008-2019, Jesus Calvino-Fraga

Introduction

This document provides step by step instructions on how to build, load, and run user code in an 8052 soft processor configured for the Altera DE1-SoC board.

Configuration Steps

These configuration steps need to be executed only once:

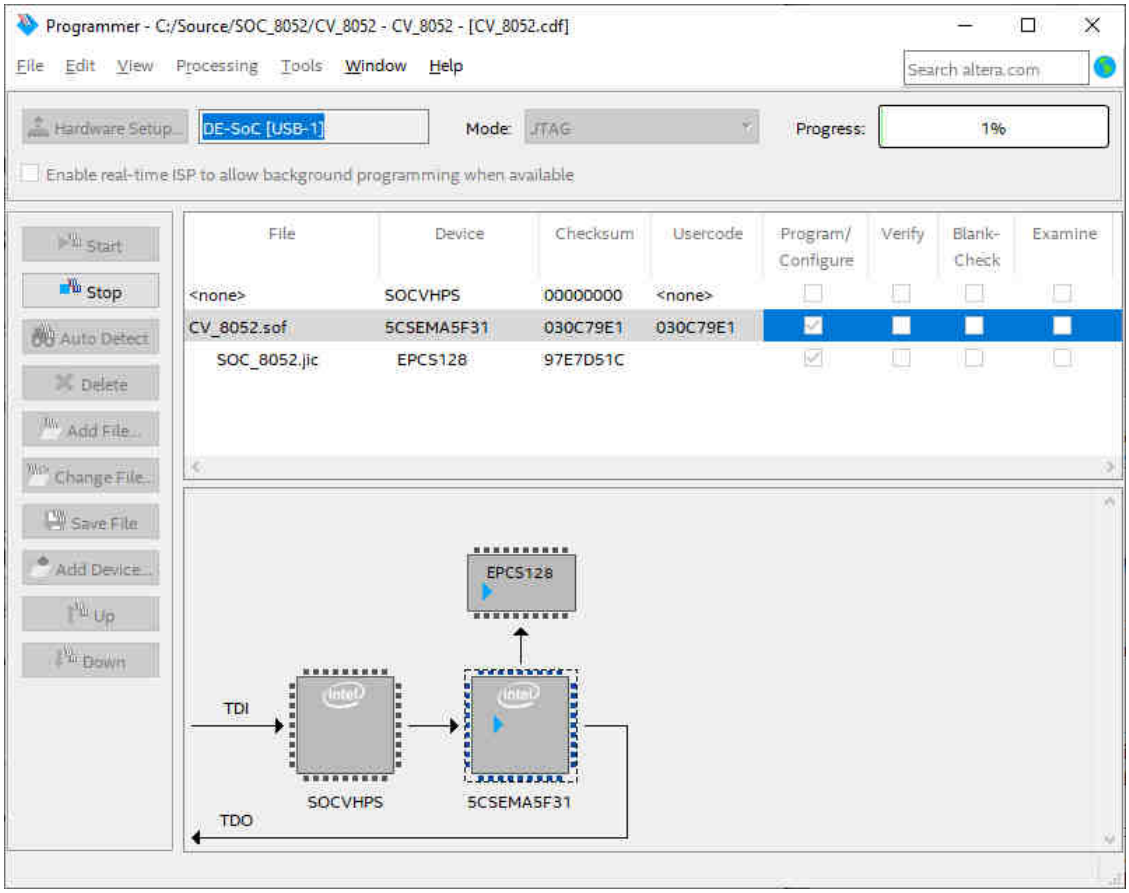
1) Download the Quartus project files for the 8052 soft processor from the course WebCT page. All the required files are compressed in the file ***'SOC_8052.zip'***. Decompress all files in a folder of your choice. If you are using the laboratory computers, remember to decompress the files in a folder into your network drive Z:

2) Start Quartus (version 16 or up). Open the project ***'CV_8052.qpf'***.

3) You may skip this step if you want, as the configuration file '***CV_8052.pof'*** is already included in the project. Otherwise, in Quartus II click ***'Processing'*** → ***'Start Compilation'***. It may take a few minutes for Quartus II to finish synthesizing the CV-8052 soft processor.

4) Connect the Altera DE1-SoC to the computer using the USB cable. Also plug in the power adapter. Turn on the DE1-SoC board.

5) To download the configuration file to the Altera DE1-SoC board click ***'Tools'*** → ***'Programmer'***. In the line where the file ***'CV_8052.sof'*** is, make sure the selection box ***'Program/Configure'*** is checked. Then press the ***'Start'*** button. A moving progress bar shows that the Altera DE1-SoC board is being configured with the 8052 soft-core. Be patient. This may take several minutes.



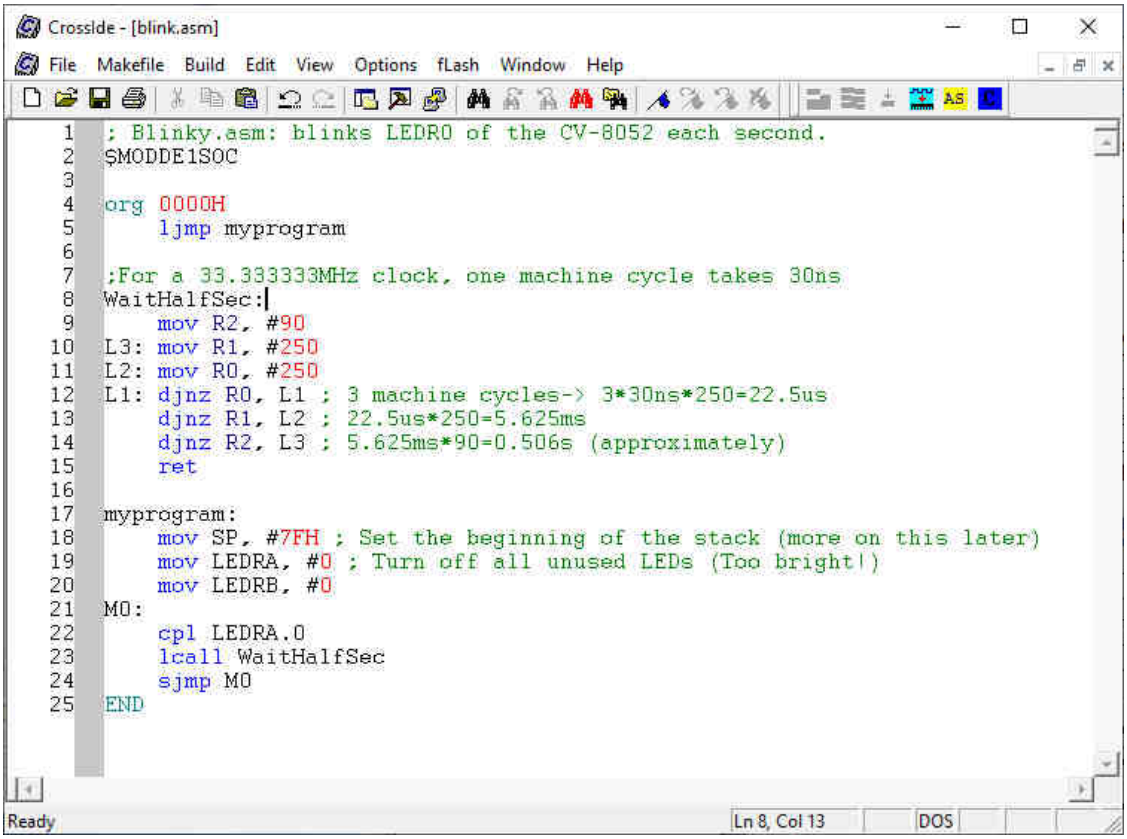
6) The CV-8052 soft processor is now loaded into the Altera DE1-SoC board and running.

Compiling and Loading Programs into the CV-8052

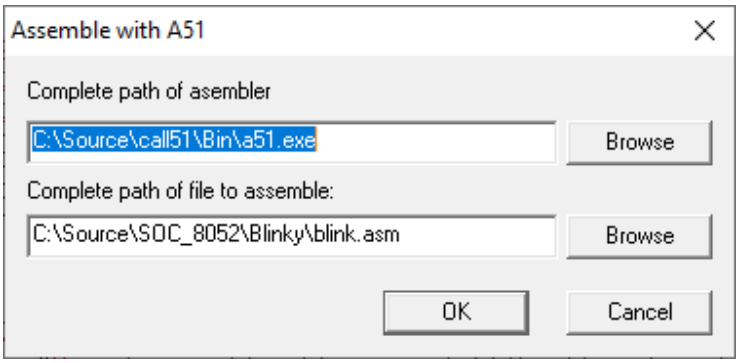
The CV-8052 soft processor includes a boot loader that can be used to load and run user programs with the CV-8052. Before doing that, we need to write and compile our program:

1) Download and install Crosside from the course web page. Crosside is a text editor with assembly and C syntax highlighting. It also includes a module to transmit files to the CV-8052 processor.

2) Start Crosside. Create a new assembly file by clicking ***'File'->'New'->'Asm source file'***. Type your assembly program and save it. The figure below shows an assembly program saved as ***'blinky.asm'***.



3) To compile the program click ***'Build'->'ASM51'***. A pop-up window appears. Click the first ***'Browse'*** button and find the program ***'a51.exe'*** in the ***'bin'*** folder of your CrosIDE\Call51 installation:



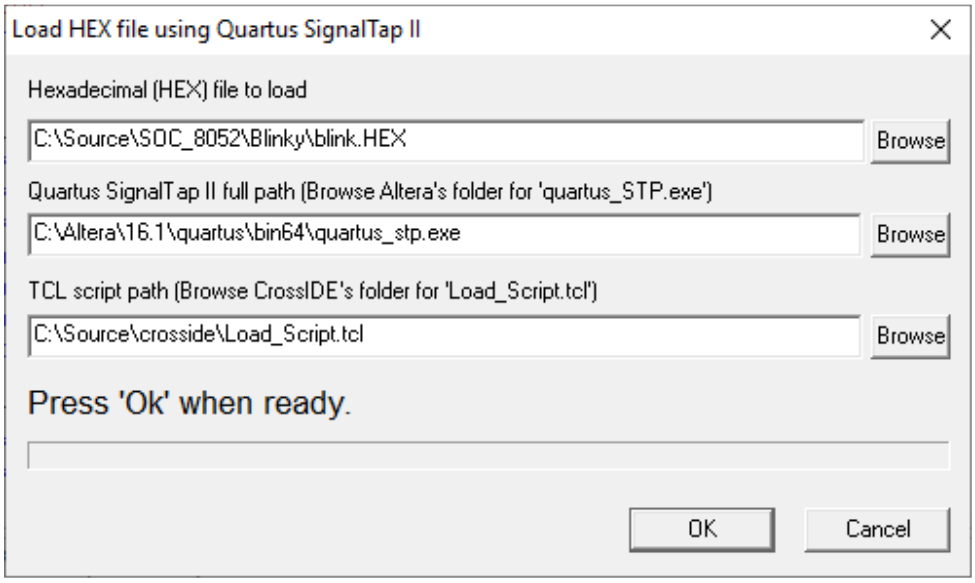
Click ***'OK'***. If your program has no errors you can proceed to the next step. Otherwise you need to fix the errors and try again.

4) After compiling your source file, a new file with the same name and extension ***'.HEX'*** is generated. This is the file we load or ***'flash'*** into the CV-8052 soft-core processor. To do that we need to activate the boot-loader in the CV-8052 by following these steps:

1. Press and hold KEY1.
2. While holding KEY1, press and release KEY0. (KEY1 is the BOOT button, KEY0 is the Reset button)
3. Release KEY1.

You will know when the boot-loader is running if LEDR0 is on and all other LEDs are off. In Crosside press ***'fLash'->'Quartus SignalTap II'***. There are three required fields you'll need to fill with valid information before proceeding further:

1. The hexadecimal (HEX) file created by the compiler.
2. The full path of the Quartus SignalTap II program (quartus_STP.exe). This program is included with Quartus II in a location similar to the one showed in the figure below.
3. The location of the Tcl script used by ***'quartus_STP.exe'***. This script is included with CrossIDE as shown in the figure below.



(Additionally the computer needs to be connected to the DE1-SoC board using a USB cable). Press ***'Ok'***. Wait until the program finishes. Check the report window in CrossIDE; the bottom of the message displayed should look like this:

```
.
.
Connecting to DE-SoC [USB-1] @2: 5CSE(BA5|MA5)/5CSTFD5D5/.. (0x02D120DD)
Sending HEX file 'C:\Source\SOC_8052\Blinky\blink.HEX'...
Sending command to copy hex file to 'flash' memory...
Done.
```

5) KEY0 is configured as the Reset button for the CV-8052. Press KEY0 and the loaded program starts running.

CV-8052 Soft-Core Special Function Registers (SFRs)

The CV-8052 soft processor includes all the standard 8052 SFRs. Some additional SFRs were added to provide access to some of the resources in the Altera DE1-SoC board. These are the additional SFRs:

SFR	Address	Description							
HEX0	91H	Seven segment display 0							
HEX1	92H	Seven segment display 1							
HEX2	93H	Seven segment display 2							
HEX3	94H	Seven segment display 3							
HEX4	8EH	Seven segment display 4							
HEX5	8FH	Seven segment display 5							
LEDRA	E8H	LEDs LEDR0 to LEDR7 (bit addressable).							
LEDRB	95H	LEDs LEDR8 to LEDR15.							
SWA	E8H	Switches SW0 to SW7 (bit addressable).							
SWB	95H	Switches SW8 to SW15.							
KEY	F8H	KEY1=KEY.1, KEY2=KEY.2, etc.							
LCD_CMD	D8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		X	X	X	X	ON	RS	EN	RW
LCD_DATA	D9H	Input/output port to LCD							
LCD_MOD	DAH	Write 0FFH to make LCD_DATA an output							
P0MOD	9AH	Input/Output mode bits for port 0							
P1MOD	9BH	Input/Output mode bits for port 1							
P2MOD	9CH	Input/Output mode bits for port 2							
P3MOD	9DH	Input/Output mode bits for port 3							

SBIT	Address	Description
ADC_MISO	F8H	SPI MISO for AD7928/LTC2308 (Read Only Bit)
ADC_MOSI	F9H	SPI MOSI for AD7928/LTC2308 (Write Only Bit)
ADC_SCLK	FAH	SPI CLK for AD7928/LTC2308 (Write Only Bit)
ADC_ENN	FBH	SPI ENABLE for AD7928/LTC2308 (Write Only Bit)

Pin Assignments

All the standard 8052 I/O pins are assigned to the expansion headers of the Altera DE1-SoC board. By default all the port pins (P0 to P3) are configured as inputs. To configure any of the pins of a port as an output write 1 to the corresponding bit in the PxMOD register described above. For example to make P0.0 and P0.7 outputs and leave P0.1 to P0.6 as inputs, write 81H to P0MOD. These are the pin assignments:

JP1

LCD_DATA[0]	**1**	**2**	*LCD_DATA[1]*
LCD_DATA[2]	**3**	**4**	*LCD_DATA[3]*
LCD_DATA[4]	**5**	**6**	*LCD_DATA[5]*
LCD_DATA[6]	**7**	**8**	*LCD_DATA[7]*
LCD_EN	**9**	**10**	*LCD_RS*
5V	**11**	**12**	*GND*
LCD_RW	**13**	**14**	*TXD*
LCD_ON	**15**	**16**	*RXD*
FL_DQ[0]	**17**	**18**	*FL_DQ[1]*
FL_DQ[2]	**19**	**20**	*FL_DQ[3]*
FL_DQ[4]	**21**	**22**	*FL_DQ[5]*
FL_DQ[6]	**23**	**24**	*FL_DQ[7]*
FL_RST_N	**25**	**26**	*FL_WE_N*
FL_OE_N	**27**	**28**	*FL_CE_N*
3.3V	**29**	**30**	*GND*
TDO	**31**	**32**	*TDI*
TCS	**33**	**34**	*TCK*
Not used	**35**	**36**	*Not used*
T0	**37**	**38**	*T1*
T2	**39**	**40**	*T2EX*

JP2

P0.0	**1**	**2**	*P0.1*
P0.2	**3**	**4**	*P0.3*
P0.4	**5**	**6**	*P0.5*
P0.6	**7**	**8**	*P0.7*
P1.0	**9**	**10**	*P1.1*
5V	**11**	**12**	*GND*
P1.2	**13**	**14**	*P1.3*
P1.4	**15**	**16**	*P1.5*
P1.6	**17**	**18**	*P1.7*
P2.0	**19**	**20**	*P2.1*
P2.2	**21**	**22**	*P2.3*
P2.4	**23**	**24**	*P2.5*
P2.6	**25**	**26**	*P2.7*
P3.0	**27**	**28**	*P3.1*
3.3V	**29**	**30**	*GND*
P3.2	**31**	**32**	*P3.3*
P3.4	**33**	**34**	*P3.5*
P3.6	**35**	**36**	*P3.7*
INT0	**37**	**38**	*INT1*
Not Used	**39**	**40**	*Not used*