Revision 1 2 CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

INTRODUCTION

The purpose of this document is to present a quick tutorial on how to create an AVR® assembler project within *Atmel Studio*, how to simulate an assembly program with the software debugger built into *Atmel Studio*, and also how to emulate an assembly program with the debugger/programmer built into your μ PAD.

Additionally, the appendix of this document identifies how to do the aforementioned using the C programming language. When using C and working across a network, you will need to perform additional steps also laid out in the last page of this document.

For more information on Atmel Studio, visit the Atmel Studio User Guide.

REQUIRED MATERIALS

- <u>GPIO_Output.asm</u>
- µPAD v2.0 with USB A/B connector cable

SUPPLEMENTAL MATERIALS

• <u>µPAD v2.0 Schematic</u>

PROCEDURE

NOTE: This tutorial assumes that you already have *Atmel Studio* installed, and that you have set your workspace folder to a known location. See the <u>*Atmel Studio Installation Instructions*</u> posted on our course website to learn how to do so.

- 1. Open *Atmel Studio*, and create a new project by navigating to File \rightarrow New \rightarrow Project.
- 2. Under *Installed*, select Assembler, and then AVR Assembler Project (see *Figure 1*).
- 3. Browse to a desired location for which to save the file (using the *Location* textbox), and save the file with a meaningful name (using the *Name* textbox). For this tutorial, we will call the project "GPIO_Output". Leave everything else default, and create the project by clicking the *OK* button.
- 4. In the *Device Selection* window that automatically opens, select the correct device and click *OK*. For the entirety of this semester, we will be using the ATxmega128A1U (see *Figure 2*).

w Project				? ×	Device Selection						
Recent		Sort by: Default	· E	Search Installed Templates (Ctrl+E)							
Installed		AVR Assembler Project	Assembler	Type: Assembler Creates an AVR 8-bit Assembler project	Device Family:	All	~			Search for device	۶
Assembler					Name	App./Boot Memo	ory (Kbytes)Data Memory	(bytes)EEPROM (bytes)	Device Info		
AtmelStudio Solut	tion				ATtiny88	8	512	64	^ Device Na	me: ATxmega128A1U	
					ATtiny9	1	32	N/A	Speed:	N/A	
					ATxmega128A1	136	8192	2048	Vcc:	N/A	
					ATxmega128A1U	136	8192	2048	Family:	XMEGAA	
					ATxmega128A3	136	8192	2048	· · · · ·		
					ATxmega128A3U	136	8192	2048	Datas	sheets	
				Las Ald Drag	ATxmega128A4U	136	8192	2048			
				"call delay	ATxmega128B1	136	8192	2048	Supported		
					ATxmega128B3	136	8192	2048	Atmel-	ICE	
				Tap Loop	ATxmega128C3	136	8192	2048	AVR D	ragon	
				ASM	ATxmega128D3	136	8192	2048	AVRIS	Pinkli	
					ATxmega128D4	136	8192	2048			
					ATxmega16A4	20	2048	1024	AVR C	DNE!	
					ATxmega16A4U	20	2048	1024	X EDBG		
					ATxmega16C4	20	2048	1024	JTAGI	CE3	
					ATxmega16D4	20	2048	1024			
					ATxmega16E5	20	2048	512	JTAGI	CE mkll	
	AssemblerApplic				ATxmega192A3	200	16384	2048	A Power	Debugger	
					ATxmega192A3U	200	16384	2048	~		_
ation:	C:\Users\	\Documents\Atmel Studio\7.0	•	Browse							
ition name:	AssemblerApplic	ation1		Create directory for solution							
				OK Cancel						OK	Cance

Figure 1: Choosing the project type

Figure 2: Choosing the device type

CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

You should now see a workspace similar to what is shown in Figure 3. From this point on, we will begin to use the code from the GPIO Output.asm file mentioned above. This short program utilizes a GPIO port connected to the RGB LED package on your µPAD. A GPIO port can be defined in many capacities, though in the context of this tutorial, it suffices to relate a GPIO port with a group of physicals pins controlled by electrical signals, i.e., a low voltage signal corresponding to a binary '0', a higher voltage corresponding to a binary '1'.

- 5. Copy the code from the accompanying GPIO Output.asm file to your main program. Save the main program by navigating to **File** \rightarrow Save main.asm, or by pressing Ctrl+S (i.e., Ctrl and then S) on your keyboard. Alternatively, save all aspects of the project by navigating to File \rightarrow Save All, or by pressing Ctrl+Shift+S on your keyboard.
- 6. Build (and compile) the project solution by navigating to **Build** \rightarrow **Build** Solution (or by using the function key F7). If your code has no errors, i.e., if it was copied correctly, the Output window at the bottom should include "Build succeeded."

NOTE: If your compiler detects any errors or warnings, you can double-click on an individual listing of either type and your cursor should be brought to the offending line (sometimes, your compiler cannot associate the error or warning with a specific line of code).

NOTE: If you would like to add line numbers in the code editor (which is highly recommended), navigate to Tools \rightarrow Options \rightarrow Text Editor \rightarrow All Languages and select Line Numbers, under the Settings heading.

NOTE: Within the assembly file code editor, press Ctrl+Space on your keyboard to bring up a dialog box consisting of a list of assembly instructions for the currently chosen device. If any string of letters is typed into this dialog box, the listings shown will start with that specified string of letters. The internal program used for this function is sometimes known as Intellisense, Autocomplete, or Auto Completion.

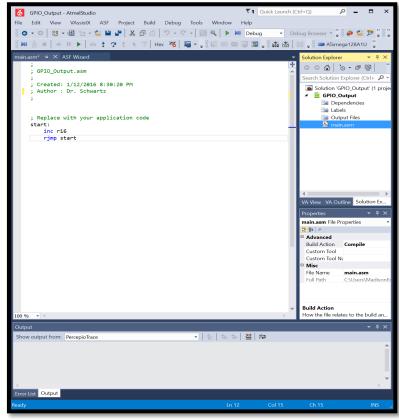


Figure 3: Atmel Studio application window, after creating project

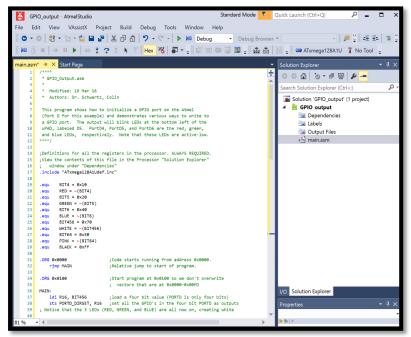


Figure 4: Atmel Studio application window, after copying contents of GPIO Output.asm

CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

Before we can execute the program, we must select the appropriate debugging tool in *Atmel Studio*. For simulation, we will select the software simulator built into *Atmel Studio*, and for emulation, we will select the programmer/debugger built into your μ PAD.

7. To perform a simulation of the program, we must first select the simulator debugging tool. Click the target selector icon (listed *No Tool* by default), as pointed to by the red arrow in *Figure 5*. When you select this icon, the project *Tool* window shown in *Figure 6* will appear. (Alternatively, the project *Tool* window can be opened by navigating to **Project** → *<Project_Name>* **Properties**, and then selecting the *Tool* option.) Under *Selected debugger/programmer*, use the dropdown menu to select *Simulator*. The target selector icon should now display *Simulator* (as also shown in *Figure 6*). Finalize these changes by saving the project (**File** → **Save All**), and then close the project properties window.

<pre>Conc_output - Anthespuble File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VAssitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Help File Edit View VassitX Project Build Debug Tools Window Window Help File Edit View VassitX Project Build Debug Tools Window Prove Prov</pre>				
Image: Start Page Image: Start Page: Start Page Image: Start Page: Start Pa	🥳 G	PIO_output - AtmelStudio	Standard Mode 🔻	Quick Launch (Ctrl+Q) 🖉 🗕 🗖 🗙
Image: Start Page Image: Start Page: Start Page Image: Start Page: Start Pa	THE P		and the	
NI Imain arm <				
<pre>main.set Stat Page //**** //**** //**** //********</pre>	· • •			· · · · · · · · · · · · · · · · · · ·
<pre>/*** * GoTQ_Output.sss * * * Modified: 19 Har 16 * Authors: Dr. Schwartz, Colin * This regress hows how to initialize a APID port on the Atmal (fort D for this example) and demonstrates various ways to write to a GPID port. The output will be host link LBD at the bottom left of the uAD, labeled DS. PortDS, and PortDE are the red, green, and blue LDS, respectively. Note that these LDS are active.ex. **** **** ***** ***** ************</pre>	j ni	a = → ▶ ↔ ‡ ? î ħ ī <mark>Hex</mark> 76 ■ - 🚚	🛱 🗰 🛱 🧊 🖪 🚽 🏙 🖄	🖞 🛛 🚽 MTxmega128A1U 🥤 No Tool 🖕
<pre>/**** * For Dutput.sss * * * Modified: 19 Har 16 * Authors: Dr. Schwartz, Colin This program shows how to initialize a GPID port on the Atmal (fort D for this example) and demonstrates various ways to write to a GPID port. The output will black LBD at the bottom left of the urAD, Isbeled DS. PortDS, and PortDE are the red, green, and blac LDS, respectively. Note that these LDB are active.exe, **** **** ***** *******************</pre>	main.asi	n* 🕫 🗙 Start Page		▼ Solution Explorer ▼ 4 ×
<pre>definitions for all the registers in the processor "Solution Explorer"</pre>	1	/****		
<pre>starth Solution Explorer (CUH=3</pre>	2	* GPIO_Output.asm		
<pre>* Modified 19 MP 18 * Modified 19 MP 18 * Authors: br. Schmartz, Colin This program shows how to initialize a GPID port on the Attel Growt D for this example and demonstrates various ways to write to a GPID port. The output will blank LBb at the bottom lart of the growth D for this program shows how to initialize a GPID port on the Attel Growth D for this example and demonstrates various ways to write to a GPID port. The output will blank LBb at the bottom lart of the growth D for this program shows how to that these LBb are active low. ***/ pofinitions for all the registers in the processor "Solution Explore" i window under "Dependencies" i window under "Dependencies" i window under "Dependencies" degram State = colle equ BUTS = colle equ BUTS = colle equ BUTS = colle i [Code starts running from address 0x0000. equ BUTS = colle i [Code starts running from start of prog</pre>		*		Search Solution Explorer (Ctrl+;)
This program shows how to initialize a GPTD port on the Attel (Port B for this example) and demonstrates various ways to write to a GPTD port. The output will blink LBB at the bottom left of the unday, latelide, respectively. Note that these LBGs are actively. """"""""""""""""""""""""""""""""""""				search solution explorer (cur+,)
<pre>This program shows how to initialize a 97D port on the Attel (over D for this seample) and demonstrates various ways to write to a 97D port. The output will blink LBD at the bottom laft of the demonstrates varies and send and there are the rad, preen, """/ perinitions for all the registers in the processor "Solution Explorer" i variou under "Demonstrates" equ at RED = (RITA) eq</pre>		* Authors: Dr. Schwartz, Colin		Solution 'GPIO output' (1 project
<pre>Inits to give hits sumpt to initialize solution usys to a G of OPD pot. The output will bink LBD at the bottom left of the usAD, labeled DS. PortDS, and PortDS are the red, green, and blue LDS. respectively. Note that these LDS are active low. """/ """/ """/ """/ """/ """/ """/ ""</pre>				A GPIO output
<pre>s defDo port. The output will blink LEDs at the bottom left of the the uRAD, labeled EX. FortEA, FortCAS, and FortEA serve the red, green, and blue LEDs, respectively. Note that these LEDs are activelow. they they they they they they they they</pre>				
<pre>ublock to up of the second secon</pre>				
<pre>in and blue LDGs, respectively. Note that these LDDs are active-low. def in the section of this file in the processor. ALWAYS REQUIRED. pives the contents of this file in the Processor "Solution Explorer" include "AffamegalERAUGEFINC" equ affa = otda equa affa = otda</pre>				
<pre>image contents of this file in the processor "Solution Explorer" if include "Altamegal28AlUder.inc" include "Altamegal28AlUder.inc" include "Altamegal28AlUder.inc" equ afto = oct0 e</pre>				🖼 Output Files
<pre>iDefinitions for all the registers in the processor. ALWAYS REQUIRED. jview the contents of this file in the Processor "Solution Explorer" introlde "AfmegaIZARIUGF.inc" introlde "Af</pre>			- 104.	and main asm
<pre>is juberinitions for all the registers in the processor. ALWAY REQUIRED. is jubles the contents of this file in the Processor "Solution Explorer" is windew under "Dependencies" include "Annegatablustrinc" include "Annegat</pre>		,		
<pre>is juike the contents of this file in the Processor "Solution Explorer" is include "Affamegal28ALUGF",inc" include "Affam</pre>		:Definitions for all the registers in the processor, ALWAYS RED	UIRED.	
<pre>16 ; window under "Dependencies" 17 ; include "ArwegelizBaldef.inc" 18 19 ; include "ArwegelizBaldef.inc" 19 10 ; equ SIT4 = 0x10 10 ; equ SIT5 = 0x0 11 ; equ SIT5 = 0x0 12 ; equ SIT6 = 0x0 13 ; equ SIT6 = 0x0 14 ; equ SIT6 = 0x0 15 ; equ SIT6 = 0x16 16 ; equ SIT6 = 0x16 17 ; equ SIT6 = 0x5 16 ; equ SIT6 = 0x5 17 ; equ SIT6 = 0x5 16 ; equ SIT6 = 0x5 17 ; equ SIT6 = 0x5 16 ; equ SIT6 = 0x5 17 ; equ SIT6 = 0x5 17 ; equ SIT6 = 0x5 18 ; equ SIT6 = 0x5 19 ; equ SIT6 = 0x5 10 ; equ SIT6</pre>	15			
<pre>19 .equ 8174 = 0x10 19 .equ 8174 = 0x10 20 .equ 8175 = x(8174) 21 .equ 8175 = x(8174) 22 .equ 0628 = x(8174) 23 .equ 0628 = x(8174) 24 .equ 0628 = x(8174) 25 .equ 06174 = 0x00 26 .equ 06174 = x(8174) 27 .equ 81745 = 0x70 28 .equ 06174 = x(8174) 29 .equ 81745 = 0x70 20 .equ 81745 = x(8174) 20 .equ 81755 = x(81745 = x(8174)) 20 .equ 81755 = x(81745 = x(8174)) 20 .equ 81755 = x(81745) 20 .equ 81755 =</pre>	16			
<pre>19 .equ 8174 = 0x18 20 .equ 8175 = 0x20 21 .equ 8175 = 0x20 22 .equ 8175 = 0x20 23 .equ 81755 = 0x20 24 .equ 81745 = 0x10 25 .equ 81745 = 0x10 26 .equ 81745 = 0x10 27 .equ 81745 = 0x20 26 .equ 81745 = 0x20 27 .equ 81745 = 0x20 28 .equ 81745 = 0x20 29 .equ 81745 = 0x20 20 .equ .</pre>	17	.include "ATxmega128A1Udef.inc"		
<pre>20 .equ 80 = -(81T4) 21 .equ 8175 = -(81T4) 22 .equ 8175 = xx28 23 .equ 8176 = xx28 24 .equ 8176 = xx28 25 .equ 8176 = xx28 26 .equ 8176 = xx217469 27 .equ 81776 = xx271469 28 .equ 8176 = xx271469 29 .equ 8176 = xx271469 20 .equ 8176 = xx271469 20 .equ 8176 = xx271469 21 .equ 8176 = xx271469 22 .equ 8176 = xx271469 23 .equ 8176 = xx271469 24 .equ 8176 = xx271469 25 .equ 8176 = xx271469 25 .equ 8176 = xx271469 26 .equ 8176 = xx271469 27 .equ 8176 = xx271469 28 .equ 8176 = xx271469 29 .equ 8176 = xx271469 20 .equ 8176 = xx271469 20 .equ 8176 = xx271469 20 .equ 8176 = xx271469 21 .equ 21 .</pre>	18			
1 .equ BITS = 0x.20 22 .equ OREEM = -(0TS) 23 .equ BITS = 0x40 24 .equ BITS = (0TS) 25 .equ BITS = (0TS) 26 .equ BITS = (0TS) 27 .equ BITS = (0TS) 28 .equ BITS = (0TS) 29 .equ BITS = (0TS) 20 .equ .equ 20 .equ .equ 20 .equ .equ 20 .equ .equ 21 .equ .equ 21 <				
22 .equ 0REW = -(6175) 23 .equ 0IUE = -(6176) 24 .equ 0IUE = -(6176) 25 .equ 0IUE = -(6176) 26 .equ 0IUE = -(6176) 27 .equ 0IUC = -(6176) 28 .equ 0IUC = 00000 29 .equ 0IUC = 00000 20 .equ 0IUC = 000000 21 .equ 0IUC = 000000 22 .equ 0IUC = 0000000 23 .equ 0IUC = 00000000000000000000000000000000000	20	.equ RED = ~(BIT4)		
23 .equ SIT6 = 0x40 24 .equ SIT6 = 0x40 25 .equ SIT755 = 0x70 26 .equ SIT755 = 0x70 27 .equ SIT755 = 0x70 28 .equ SIT755 = 0x70 29 .equ SIT64 = 0x50 29 .equ SIT64 = 0x50 20 .equ SIT64 = 0x50 21 .equ SIT64 = 0x50 22 .equ SIT64 = 0x50 23 .equ SIT64 = 0x50 24 .equ SIT64 = 0x50 25 .equ SIT64 = 0x50 23 .equ SIT64 = 0x50 24 .equ SIT64 = 0x50 25 .equ .equ 26 .equ .equ 27 .equ .equ 28 .equ .equ 29 .equ .equ 20				
24 .etu BLUE = -(BIT6) 25 .etu BIT45 = 0x70 26 .etu BIT45 = 0x70 27 .etu BIT45 = 0x70 28 .etu BIT45 = 0x70 29 .etu BIT45 = 0x64 29 .etu BLACK = 0x64 20 .etu BLACK = 0x67 30 .etu BLACK = 0x77 31 .086 0x0000 .jCode starts running from address 0x0000. 32 .etu .etu 33 .oti 0x0100 .jStart program at 0x0100 sus don't overwrite 35 .j vetors that are 4 0x0000 - 0x0000				
25 .eeu 817456 - 8X76 26 .eeu 817456 - 8X76 27 .eeu 81746 - 8X58 28 .eeu 81746 - 8X58 29 .eeu 918K - (81745) 29 .eeu 918K - (81745) 29 .eeu 918K - (81745) 20 .eeu 918K - (81745) 21 .eeu 91745 - (81745) 22 .eeu 91745 - (81745) 23 .eeu 91745 - (81745) 24 .eeu 91745 - (81745) 25 .eeu 91745 - (81745) 26 .eeu 91745 - (81745) 27 .eeu 91745 - (81745) 28 .eeu 91745 - (81745) 29 .eeu 91745 - (81745) 20 .eeu 91745 - (81745) 29 .eeu 91745 - (81745) 20 .eeu 91745 - (81745) 20 .eeu 91745 - (81745) 20 .eeu 91745 - (81745) 20 .eeu 91745 - (81745) 21 .eeu 91755 - (817455) 21 .eeu 91755 - (817455) 21 .eeu 91755 - (8174555) 21 .eeu 917555 - (81745555) 21 .eeu 917555 - (81745555555) 21 .eeu 917555 - (8174555555555555555555555555555555555555				
<pre>26 .equ MMTF = -(817456) 27 .equ MMTF = -(817456) 28 .equ PINK = -(81764) 39 .equ PINK = -(81764) 30 .equ BLACK = 0xFF 30 31 .CAG 0x0000 jCode starts running from address 0x0000. 32 rjmp MAIN jRelative jump to start of program. 33 .CAG 0x0100 jStart program at 0x0100 so us don't overwrite 35 j vectors that are at 0x0000 PADPD</pre>				
27 .etu BITG4 = 0x50 28 .etu PIK5 = (SITG4) 29 .etu PIK5 = (SITG4) 30 .etu SLKC = 0xFF 30 .cB6 0x0000 ;Code starts nunning from address 0x0000. 31 .cB6 0x0000 ;Start program at 0x0100 so us don't overwrite 35 .cB6 0x0100 ;Start program at 0x0100 so us don't overwrite 35 .cB6 0x0100 ;Start program at 0x0100 so us don't overwrite 35 .cB6 0x0100 ;Start program at 0x0100 so us don't overwrite				
28 .equ PINK = -(&ITG4) 29 .equ BLACK = 0xFF 30 31 .COG 0x0000 jCode starts running from address 0x0000. 32 rjmp MAIN jRelative jump to start of program. 33 .COG 0x0100 jStart program at 0x0100 so us don't overwrite 35 j vetors that are at 0x0000 Ac00FD				
20 .etu BLACK = ÖXFF 30 31 .OBG ÖXGOOD jCode starts running from address öxGOOD. 32 gripp MAIN jRelative jump to start of program. 34 .OBG ÖXGIDO jStart program at ØxGIDO sove don't overwrite 35 j vetors that are at 0xGOOD exOPD				
30 .060 0x0000 ;Code starts running from address 0x0000. 32 rjmp MAIN ;Relative jump to start of program. 33 .066 0x0100 ;Start program at 0x0100 so use don't overwrite 35 j vetors that are at 0x0000 Ac000 por				
31 .ORG 0x0000 ;Code starts running from address 0x0000. 32 rjmp NWIN ;Relative jump to start of program. 33 .ORG 0x0100 ;Start program at 0x0100 so we don't overwrite 35 ; y votros that are at 0x0000-0x0000		rede enter - entr		
32 rjmp HAIN ;Relative jump to start of program. 33 40.086 ex8180 ;Start program at 0x8100 so use don't overwrite 55 ; vetors that are at 0x8000-0x000		.ORG 0x0000 :Code starts running from address 0	×0000.	
33 34 .CRG 0x0100 ;Start program at 0x0100 so we don't overwrite 35 ; vectors that are at 0x0000-0x00FD	32			
35 ; vectors that are at 0x0000-0x00FD	33			
	34	.ORG 0x0100 ;Start program at 0x0100 so we don'	t overwrite	
36 MAIN: I/O Solution Explorer	35		D	
				I/O Solution Explorer
37 ldi R16, BIT456 ;load a four bit value (PORTD is only four bits)				
				Properties - 4 ×
39 ; Notice that the 3 LEDs (RED, GREEN, and BLUE) are all now on, creating white		; Notice that the 3 LEDs (RED, GREEN, and BLUE) are all now on,	creating white	-
81 % • 4	81 %	• 4	→ →	21 21 2

Figure 5: Target selector icon

GPIO_Output - A	AtmelStudio					
File Edit View	VAssistX ASF Project Build Debug Tools Window Help					
	型 🛅 🖕 🏜 🔐 🐰 🗗 台 🦻 - ペー 🔝 🔍 🕨 Debug 🔹 Debug Browser - 👘 🏓 🏓					
j j j j j j j j j j j j j j j j j j j	🛛 🕨 🗠 🍷 😤 🏦 💺 王 Hex 🧏 📓 - 🖕 編 💷 🛱 🗐 🌆 - 🌡 🏙 💭 🏙 - 🎆 🖉 - 🎆 Altxmega128A1U 🥤 Simulator 🖕					
GPIO_Output* → ×	main.asm ASF Wizard					
Build Build Events	Configuration: N/A V Platform: N/A V					
Toolchain Device Tool Components Advanced	Device Selected debugger/programmer Tool Simulator •					
	External tool Command: Select Stimuli File Stimuli File Activate stimuli when in breakmode from menu Debug->Execute Stimulifile, then continue execution					

Figure 6: Project *Tool* window

CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

Before executing the program, we will place a breakpoint. Breakpoints are used to intentionally halt your program at a specific point of execution for debugging purposes, and are optional. A given breakpoint will halt your program immediately *before* executing the instruction specified at the breakpoint.

8. Place a breakpoint in your code editor on the first assembly instruction within your *main* program, i.e., ldi R16, BIT456, by using your mouse to click the gray pane to the far left of the instruction. A red dot will appear next to this specified instruction, as shown in *Figure 7*.

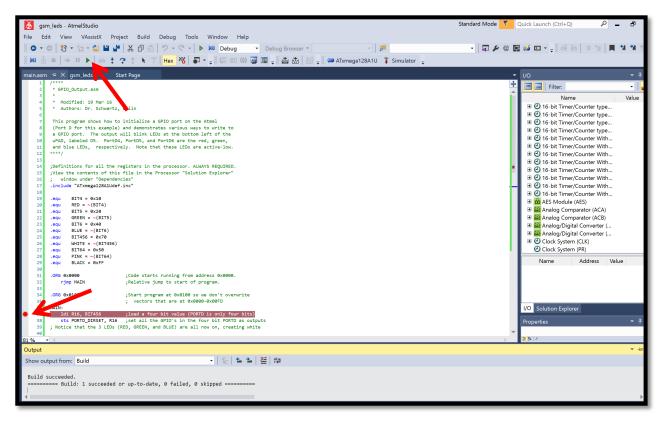


Figure 7: Breakpoint at first instruction of the given program, along with the debug Continue icon.

To start the program simulation, i.e., to start debugging, you can either select the *Start Debugging* icon (also known as the *Continue* icon) specified by the green arrow near the top of the *Atmel Studio* window (see the topmost red arrow in the *Figure 7*), or navigate to **Debug** \rightarrow **Continue**, or even simply press **F5** on your keyboard.

9. Start debugging the main program. The program should stop execution at the breakpoint you placed, and the specified line of code should be highlighted yellow, following a yellow arrow in the gray pane where the breakpoint resides. (The yellow arrow, along with the yellow highlight, indicates the next instruction to be executed.)

While debugging, you can view any of the processor's registers, I/O ports, memory locations, etc. To do so is extremely helpful when writing any embedded software. Since we are using the I/O port *PORTD* in this example program, in addition to internal GPIO registers, we will explore the *I/O* and *Processor Status* views of the debugger.

10. Open the *I/O* view by navigating to **Debug** → **Windows** → **I/O**, as shown in *Figure 8*. Open the *Processor Status* view by navigating to **Debug** → **Windows** → **Processor Status**.

⁸ Revision 1 2² CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

File	Edit View VAssistX Project	Build Debug	Тоо	ls Window Help					
30	Windows	+		Breakpoints	Alt+F9		7	- 🗔 🖋 🕼 🖬 🐝 🖬 - 🚽 - 🗄 E	비 계 개 비 게 하
8 01	Start Debugging and Break	Alt+F5	-	Data Breakpoints		M ATxmega128A1U	T Simulator		
Ř	Attach to Target			Processor Status					
m 📕	Stop Debugging	Ctrl+Shift+F5	1	I/O		→			→ ₽ >
⊳	Start Without Debugging	Ctrl+Alt+F5		Live Watch		T	Name Program Counter	Value 0x00000100	
	Disable debugWIRE and Close		Θ	Program Counter Trace			Stack Pointer	0x3FFF	
	Continue	F5	∍	Output			X Register	0x0000	
€	Execute Stimulifile			Watch	•		Y Register	0x0000	
€	Set Stimulifile		R,	Autos	Ctrl+Alt+V, A		Z Register	0x0000	
5	Restart		[4]	Locals	Alt+4		Status Register	ITHSVNZC	
11	Break All	Ctrl+F5	Σ	Immediate	Ctrl+Alt+I		Cycle Counter	2	
69	QuickWatch	Shift+F9	≣	Call Stack	Alt+7	I 1	Frequency	1.000 MHz	
*	Step Into	F11	ю.	Threads	Ctrl+Alt+H		Stop Watch	2.00 µs	
?	Step Over	F10		Modules	Ctrl+Alt+U		Registers		
1	Step Out	Shift+F11	¢ [©]	Processes	Ctrl+Shift+Alt+P		R00	0x00	
k	Run To Cursor	Ctrl+F10		Memory	•		R01	0x00	
Ì		Shift+F5	G	Disassembly	Alt+8		R02	0x00	
	Other Debug Targets	•		Registers	Alt+5		R03	0x00	
	Percepio Trace	•		so we don't overwrite 0000-0x00FD		· · · · ·	R04	0x00	
	Toggle Breakpoint	F9					R05	0x00	
C.	New Breakpoint	•		ORTD is only four bits) e four bit PORTD as outputs			R06	0x00	
	Delete All Breakpoints	Ctrl+Shift+F9	are al	ll now on, creating white			R07	0x00	
	Disable All Breakpoints		e to	the GPIO pins.			R08	0x00	
	Clear All DataTips		m, t	hen use some combinations		-	R09	0x00	
8:	Export DataTips			·		Þ	R10	0x00	
v	Import DataTips					× Memory 3			~ ₽ >
	Save Dump As				Туре	_	g APP_SECTION	 Address: 0x000000,prog 	- 0
₽	Options							. 44 44 44 44 44 44 44 44 44 44 44 44 44	<u>ŷ</u> Àŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷ
s	gsm_leds Properties							ff	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
			_			prog 0x00002 prog 0x00003		· ff	<u>ŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷ</u> ŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷ
						prog 0x00004		· FF	<u>yyyyyyyyyyyyyyyyyyyyy</u>
						prog 0x00005		· FF	<u> </u>
						prog 0x00007 prog 0x00008		· ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++	<u>9999999999999999999999999999999999999</u>
							<u> </u>	Call Stack Breakpoints Command Immediate	Output Error List Memory
						Autos Locais	Materia Memory 5	compared breakpoints command immediate	output thortist memory

Figure 8: Selecting the I/O Debug view

NOTE: When simulating, if you would like the simulated clock frequency to match the clock frequency of the actual μ PAD board (if it does not already match), i.e., the actual clock speed when you emulate, select the value listed next to *Frequency* in the *Processor Status* window, and enter the oscillator frequency that you have chosen for your device (e.g., 2.000 MHz, the default ATXMEGA128A1U clock frequency).

Now, we will utilize the I/O view to view all of registers within PORTD, starting at the point of execution specified by our chosen breakpoint.

11. Within the I/O view, filter for and select *I/O Port Configuration (PORTD)*. (To search for this, you may type something as simple as "portd" in the *Filter* textbox, as shown in *Figure 9*.) After selecting the correct port configuration, you will be able to view all of the registers associated with PORTD.

To execute the next instruction within your program, you can click the *Step Into* icon (as pointed to by the leftmost red arrow in *Figure 9*), or press F11 on your keyboard. (You may also step through each instruction by navigating to **Debug** \rightarrow **Step Into**).

12. Step through the program code, identifying changes that occur to the registers within PORTD, as well as to the microcontroller's internal GPIO registers.

NOTE: Two other useful debug stepping features are *Step Over* and *Step Out*, as pointed to by the second-rightmost and rightmost red arrows in *Figure 9*, respectively. *Step Over* will always execute the next instruction in the current procedure frame as a single unit, i.e., if the next instruction to be executed consists of a procedure call, the entire procedure will be executed in a single step. *Step Out* executes the remaining lines of a function in which the current execution point lies.

Revision 1 24 CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

				Chan de ad Maria		P	
ŵ	GPIO_Output (Debugging) - AtmelS	tudio		Standard Mode	Quick Launch (Ctrl+Q)	₽ - □	×
<u>F</u> ile	<u>E</u> dit <u>V</u> iew VAssist <u>X</u> <u>P</u> roject	<u>B</u> uild <u>D</u> ebug <u>T</u> o	ools <u>W</u> indow <u>H</u> elp				
6	🔹 🖸 🕆 🛅 🕶 🔛 💾 🔐 📗	X 口 白 ツ・ペ・	🛛 🕨 🕅 Debug 🕞 I	Debug Browser 👻	- 🗾 🎜 🚆	표 🤁 📜 📜	위끝
§ м	i ■ → □ ▶ ↔ * ?	1 k 1 Hex 🄏	🛯 • 🚚 🛤 📾 🖏 📓	📮 👪 📥 🔣 .	🖕 🖗 MTxmega128A1U 🦷 Sir	mulator 💂	
GPIO	Output.asm + ×			•	1/0		л×
	uPAD, labeled D5. /ortD4		are the red, green,	+	Filter: portd	- <i>1</i>	
	and blue LEDs, respective	ly. Note that thes	e LEDs are active-low.		Name	Value	
	****/				I/O Port Configuration		
	;Definitions for all the re	gisters in the proc	essor. ALWAYS REQUIRED.		Port Interrupt 1 Lev	•	
	;View the contents of this	file in the Process			Port Interrupt 0 Lev		
	; window under "Dependence				🗎 Output/Pull Config	urat T 0x00 ~	
	.include "ATxmega128A1Udef.	inc"			Input/Sense Config		
	.equ BIT4 = 0b00010000				Output/Pull Config		
	.equ INV4 = 0b11101111			-	Input/Sense Config Output/Pull Config		
	.equ RED = INV4				Input/Sense Config		- 11
	.equ BIT5 = 0b00100000				Output/Pull Config		
	.equ INV5 = ~BIT5 .equ GREEN = ~(BIT5)				lnput/Sense Config		
	.equ BIT6 = 0x40				🗎 Output/Pull Config		_
	.equ BLUE = ~(BIT6)				Innut/Sense Config		
	.equ BIT456 = 0x70				Name Address Va		
	.equ WHITE = ~(BIT456) .equ BIT64 = 0x50						
	.equ PINK = ~(BIT64)						
	.equ BLACK = ØxFF						
			5 II			×00 000000	
0	.ORG 0x0000 rjmp MAIN	;Code starts runni ;Relative jump to	ng from address 0x0000.	_		x00 00000C	
		Julio Julio Co	start or program.			×00 00000	
	.ORG 0x0100		0x0100 so we don't overw	rite			
		; vectors that ar	e at 0x0000-0x00FD			x00 000000	
	MAIN: ldi R16, BIT456	:load a four bit y	alue (PORTD is only four	hits)		×00 00000C	
	-		s in the four bit PORTD	· · · · · · · · · · · · · · · · · · ·		×00 00000C	_
100 %	• Notice that the 3 LEDs (R	FD GREEN and BUIE) are all now on creati	ng white	Solution Explorer Processor St	tatus I/O Properties	× 1
Local		• ¹ ×	Memory 4	F	Solution Explorer Processor Si		лх
Nar		Type	Memory: prog APP_SECTIO	v -	Address: 0x000000,prog		Υ ^ •
	I				ff ff ff ff ff ff ff ff ff	ÿÀÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ	
					ff ff ff ff ff ff ff ff	<u>ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ</u>	
			prog 0x000020 ff ff ff		ff ff ff ff ff ff ff ff	<u>ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ</u> ŸŸ	
			F - 0		ff ff ff ff ff ff ff ff	<u>ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ</u>	
			prog 0x000040 ff ff ff prog 0x000050 ff ff ff		ff ff ff ff ff ff ff ff ff	<u>ŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷ</u> ŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷŷ	
			prog 0x000060 ff ff ff		ff ff ff ff ff ff ff ff	<u>ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ</u> ÿÿ	
					ff ff ff ff ff ff ff ff	<u>ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ</u>	
A	Locals Martin 1 - Martin 2	T			ff ff ff ff ff ff ff ff		
	s Locals Watch 1 Watch 2		Call Stack Breakpoints Com		nediate Window Output Mem		
Stoppe	ed			Ln 33	Col 70 Ch 5	58 IN:	S _{all}
_							

Figure 9: Step Into, Step Over, and Step Out debug icons

We will now stop debugging and begin to emulate the program on the μ PAD. To stop debugging, you can navigate to **Debug** \rightarrow **Stop Debugging**, or click on the *Stop Debugging* icon, i.e., the red square, in the toolbar.

13. Connect the μPAD to your computer with your USB A/B connector cable. Select the on-board Atmel Embedded Debugger (EDBG) as the Selected debugger/programmer, within the project Tool menu, as done in Step 7. Also verify that the Interface is chosen to be PDI (as shown in Figure 10). Finalize these changes by saving the project.

NOTE: The PDI clock frequency is only representative of the programmer/debugger, and need not be the same value as the clock frequency of the processor. It is recommended to *not* change the default value.

110		210
	CREATING, SIMULATING, AND EMULATING IN ATMEL	STUDIO

GPIO_Output	t - AtmelStudio		Advanced Mode 🔻 Quick Launch (Ctrl+Q)
	w VAssist <u>X P</u> roject <u>B</u> uild <u>D</u> ebug <u>T</u> ool		
		Debug Debug Browser	
		🖥 🔭 🖕 0 🕼 📖 📟 📖 🗳 🖕 0 🗰 🖾 021 🛫 0 🗰 ATxm	ega128A1U 🥤 PDI on EDBG (ATMLEDBGC0000000110) 🖕
OOTB uPAD 2.0 - 0	1110 GPIO_Output Þ 🗙		· · · · · · · · · · · · · · · · · · ·
Build Build Events	Configuration: N/A	V Platform: N/A	с
Toolchain			
Device Tool	Selected debugger/programmer		^
Components	EDBG • ATMLEDBGC0000000110 V Inte	erface: PDI ~	
Advanced			
	PDI Clock		
		4 MHz Reset to default clock	
		Reset to default clock	
	Programming settings		
	Erase entire chip ~		
	✓ Preserve EEPROM		
	Erase entire chip ×		

Figure 10: Selecting the on-board µPAD debugger

- 14. If the EDBG device does not appear as shown in Figure 10, there are a few possiblities to correct this problem. Try the first one below; if this does not work then try the second one.
 - If you are CONVINCED that you properly followed the install directions (and the previous tweet did not fix it), uninstall it, turn off your antivirus, reinstall it, & turn on antivirus.
 - Use the Device Manager to remove some extra drivers that altered the USB function through Atmel Studio. The procedure can be found at <u>https://www.microchip.com/forums/m1087470.aspx</u>.
- 15. Repeat steps 9-12. The code will now be executing in hardware rather than in software; once more, this is known as emulation. Upon stepping through the program, you should see the RGB LED package at the bottom left of your μPAD (labeled D4) reflect the changes made to PORTD within the program. These LEDs are connected to PORTD of your processor, as shown in the μPAD v2.0 Schematic.

Revision 1 24 CREATING, SIMULATING, AND EMULATING IN ATMEL STUDIO

APPENDIX A: SPECIAL CONSIDERATIONS FOR PROJECTS USING C

Atmel Studio cannot work accoss networks without using a network drive. Below are instructions on how to create a network drive, for Windows 10 and versions prior.

Setting Up A Network Drive For Atmel Studio

Windows 10:

- 1. Type Windows-E (i.e., hold down the windows key and then type E) to open and Windows Exporer
- 2. In Windows Explorer, select This PC
- 3. Select Computer → Map network drive
- 4. Select a drive letter, e.g., Z:
- 5. Put the path to the folder that you want to use, e.g., \\mil.ufl.edu\tebow\3744\labs\
- 6. Select **Reconnect at sign-in**
- 7. Select Finish

Pre-Windows 10:

- 1. Type Windows-E (i.e., hold down the windows key and then type E) to open and Windows Exporer
- 2. In Windows Explorer, select This PC
- 3. Select Map network drive
- 4. Select a drive letter, e.g., Z:
- 5. Put the path to the folder that you want to use, e.g., \\mil.ufl.edu\tebow\3744\labs\
- 6. Select Reconnect at logon
- 7. Select Finish

Procedure to Create, Simulate, and Emulate in C

- 1. Open *Atmel Studio*, and create a new project by navigating to File \rightarrow New \rightarrow Project.
- 2. Under *Installed*, under the C/C++ subheading, select GCC C Executable Project.
- 3. Find the path to the proper location with the *Location* textbox. If necessary, put in the network drive (e.g., Z:), and then the correct folder name.
- 4. Create the project by clicking *OK*, and perform everything else as specified in the above tutorial. The only new limitation might be that to be able to completely step through a C program, the compiler's optimization level may need to be changed. (Optimization allows your compiler to interpret your written C code and attempt to generate more efficient assembly/machine code, i.e., some C statements written in your program might be removed by the compiler if any level of optimization is enabled.)
 - To turn off optimization, first navigate to Project → <Project_Name> Properties. Within the project properties window, select the Toolchain subheading. Under AVR/GNU C Compiler in the Toolchain, select Optimization. Within the Optimization window, select the drop-down box for Optimization Level, and choose None (-O0). In newer versions of Atmel Studio, there is an Optimize debugging experience (-Og) option.
 - Don't forget to optimization back on (Optimize (-O1), by default) when you are finished debugging.

NOTE: Details for each optimization level specified by the AVR/GNU C Compiler can be found in the <u>Atmel</u> <u>Studio User Guide</u>.