TRAINING MANUAL

From Maker to Manufacture: Bridging the Gap from Arduino to AVR

AN-12077

Prerequisites

Atmel

Hardware Prerequisites

- ATmega328P Xplained Mini Board
- IO1 Xplained Pro extension board
- Arduino Xplained Pro board
- Micro-USB cable
- Software Prerequisites
 - Atmel[®] Studio version 6.2 or later
 - Arduino IDE 1.6.0
 - Arduino Extension for Atmel Studio
 - Terminal Window Extension
- Estimated Completion Time
 - Two hours

Introduction

This hands-on will demonstrate how to develop Arduino using Atmel Studio along with the rich user interface and other great development tools that it provides.

Arduino is an open-source electronics prototyping platform based on flexible, easy- to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Key fact about Arduino is that these boards are based on the Atmel microcontroller family and underlying software is based on Atmel development tools.

Answer for 'Why should I switch from Arduino?' is:

The Arduino IDE:

- quite limiting for experienced programmers capabilities
- · lack of compiler warnings and debugging capabilities
- (Serial.println() after every statement doesn't count), make life hard when working on advanced projects
- Atmel Studio is a great choice for users that have outgrown the integrated Arduino IDE
- some of those Arduino libraries are just so darn convenient

So why not have the best of both worlds? Arduino: a wrapper on top of C/C++ with debugging capabilities. It is possible to combine any Arduino sketch or library with your own custom code on advanced projects.

Atmel has a unique, privileged position in Arduino and responsibility to the "going pro" community to provide a bridge, help transition from hobbyist to developer.

The Atmel Xplained Mini family is a perfect "bridge" for easy transition to C and C++. It has very similar architecture to Arduino and offers most of the features of an Arduino board. It is possible to run Arduino sketches when the IDE is set up properly. On-board hardware debugger/programmer is also available. It uses the incredibly popular AVR[®] microcontroller family and it is relatively inexpensive.

Figure 1. ATmega328P Xplained Mini Board



The training material is composed of four assignments.

In the **first assignment** we will see how to connect ATmega328P Xplained Mini to Arduino IDE.

In the **second assignment** we will create Arduino sketches in Atmel Studio using Studio's Arduino Extension.

In the **third assignment** we will configure Atmel Studio to directly take in sketches. We will discuss how to transit to Atmel Studio with existing Arduino sketches. Also, we will check how to insert break points and how to debug in the Atmel studio.

In the **fourth assignment** we will edit the file and create a simple application with peripherals: ADC, I²C, and SPI. The application is to read a light sensor through ADC, read a temperature sensor through an I²C interface, and store data in the SD card through a SPI interface.



Table of Contents

1	Trair	ning Module Architecture	6
	1.1	Atmel Studio Extension (.vsix)	6
	1.2	Atmel Training Executable (.exe)	6
2	Assi the A	gnment 1: How to connect the ATmega328P Xplained N Arduino IDE	/lini to 7
	2.1	Arduino IDE	7
	2.2	mEDBG Firmware upgrade on the ATmega328P Xplained Mini	7
	2.3	Set the Bootloader Fuses in ATmega328P	8
	2.4	Program the Bootloader	9
	2.5	Configure Arduino IDE	10
	2.6	Upload the Program	12
3	Assi	gnment 2: Creating Arduino Sketches in Atmel Studio	13
	3.1	Download Extension	13
	3.2	Create Sketch	14
4	Tran	sit to the Atmel Studio IDE with Existing Arduino Sketch	nes17
	4.1	Project Creation	17
	4.2	Configuring Compiler Symbols	19
	4.3	Configuring Compiler Directories	19
	4.4	Add Arduino Dependency Files	20
	4.5	Build Solution	21
	4.6	Plug in ATmega328P Xplained Mini Board	21
	4.7	Debugging	22
5	ATm	ega328P Application	26
	5.1	Compiler Setup	26
	5.2	Add Dependency Files	27
	5.2.1	For Wire Library	27
	5.2.2	For SD Library	28
	5.3	Developing Application	29
	5.4	Hardware Connection	34
	5.4.1	Connection: IO1 Xplained Pro – Arduino Xplained Pro	35
	5.4.2	Connection: ATmega328P Xplained Mini - Arduino Xplained Pro	o35
	5.4.3	Connection: USB cable	36
	5.5	Debugging the Application	38
Ар	pendix	A. Complete Solution to Assignment 4	43



6	Conclusion	46
7	Revision History	47



Icon Key Identifiers

i	INFO	Delivers contextual information about a specific topic.
	TIP	Highlights useful tips and techniques.
	TO DO	Highlights objectives to be completed.
\checkmark	RESULT	Highlights the expected result of an assignment step.
	WARNING	Indicates important information.
•	EXECUTE	Highlights actions to be executed of the target when necessary.



1 Training Module Architecture

This training material can be retrieved through different Atmel deliveries:

- As an Atmel Studio Extension (**.vsix** file) usually found on the Atmel Gallery web site (http://gallery.atmel.com/) or using the Atmel Studio Extension manager
- As an Atmel Training Executable (.exe file) usually provided during Atmel Training sessions

Depending on the delivery type, the different resources needed by this training material (hands-on documentation, datasheets, application notes, software, and tools) can be found on different locations.

1.1 Atmel Studio Extension (.vsix)

Once the extension is installed, you can open and create the different projects using "New Example Project from ASF..." in Atmel Studio.



1

1

The projects installed from an extension are normally found under "Atmel Training > Atmel Corp. Extension Name".

There are different projects which can be available depending on the extension:

- Hands-on Documentation: contains the documentation as required resources
- Hands-on Assignment: contains the initial project that may be required to start
- Hands-on Solution: contains the final application which is a solution for this hands-on



1.2 Atmel Training Executable (.exe)

Depending on where the executable has been installed, you will find the following architecture which is composed by two main folders:

- AN-12077_Hands-on: contains the initial project that may be required to start and a solution
- **Resources**: contains required resources (datasheets, software, tools...)



Unless a specific location is specified, each time a reference is made to some resources in the following pages, the user must refer to this Resources folder.



2 Assignment 1: How to connect the ATmega328P Xplained Mini to the Arduino IDE

2.1 Arduino IDE

Download the Arduino IDE from www.arduino.cc.

2.2 mEDBG Firmware upgrade on the ATmega328P Xplained Mini

- 1. Go to Atmel spaces http://spaces.atmel.com/gf/project/avr_xp_mini/frs/ .From the package "medbgdebugger" select medbg_fw.zip and download.
- 2. Overwrite the zip package medbg_fw.zip in the Atmel Studio installation folder (e.g.: C:\Program Files (x86)\Atmel\Atmel Studio 6.2\tools\mEDBG).
- 3. Start Atmel Studio.
- 4. Connect the ATmega328P Xplained Mini to the computer.
- 5. In Atmel Studio, select Tools \rightarrow Device programming (alt.: Ctrl+Shift+P).
- 6. In the Device Programming window, set Tool to mEDBG and click "Apply".

mEDBG (ATML2323040200	011546) - Device Programming		? 🗙
Tool Device mEDBG ATmega32	Interface	Device signature Target Voltage 0x1E950F Read 5,0 V Read	
Interface settings Tool information Device information Oscillator Calibration Memories Fuses Lock bits Production file	mEDBG Debug host Debug port Serial number Connection Firmware Version Hardware Version External Link:	127.0.0.1 51086 ATML2323040200011546 com.atmel.avrdbg.connection.cmsis-dap 1.6 0	Copy to clipboard
 Getting tool info 	ОК		Close





Atmel Studio will now ask you if you want to upgrade the firmware.

Firmware Upgrade		[X]
EDBG firmware is upo	lated	
	On Tool	On Disk
Firmware Version	1.0	1.6
EDBG firmware succe	essfully upgraded	
		Ungrada Close

7. Select Upgrade.

INFO There is a bug in the serial number on some of the ATmega328P Xplained Mini boards making them not recognizable by Atmel Studio programming/debugging. If your board has an unknown character in the serial number there is a fix in the "Releases" folder. To see your board serial number, start Atmel Studio and go to Tools → Device programming → select tool, this will list the mEDBG with serial number, if some of the characters have a "?" on black background, download the package from serial number fix and follow the instruction from How_to_change_serial_Number.pdf.

2.3 Set the Bootloader Fuses in ATmega328P

- 1. Now In the 'Device Programming' window, select 'Fuses'.
- Change value on EXTENDED, HIGH, and LOW as below and click Program.
 EXTENDED = 0xFF
 HIGH = 0xD8
 LOW= 0xE0



ool Device	Inte	erface		Device signature		Target Vo	ltage	
mEDBG 🔻 ATmega	328P 🔻 ISI	•	Apply	0x1E950F	Read	5,0 V	Read	
nterface settings	Fuse Name		Val	ue				
fool information	SPIEN							
	WDTON							
Device information	Ø EESAVE							
Oscillator Calibration	BOOTSZ	2048	W 3800	-				
Memories	BOOTRST		-					
uses	CKDIV8							
ock bits	CKOUT							
Production file	Fuse Register	Value						
	EXTENDED	0xFF						
	HIGH	0xD8						
	LOW	0xE0						
	 Auto read Verify after 	program	ming			Pro	ogram	Copy to clipboard
interegistersOK inting operation verify in ify register EXTENDED ify register HIGHOK ify register LOWOK ify registersOK	registers OK							

Figure 2-1. Device Programming: Fuses

2.4 Program the Bootloader

i info

The bootloader hex file is located in the Arduino IDE folder: C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\atmega*.hex. Bootloader can be selected according to board configurations as listed in Table 2-1.

Table 2-1. Bootloader

Xplained Mini	Bootloader
ATmega328P/5V/16MHz	ATmegaBOOT_168_atmega328.hex
ATmega168PB/5V/16MHz	ATmegaBOOT_168_ng.hex
ATmega168PB/3.3V/8MHz	ATmegaBOOT_168_pro_8MHz.hex

- 1. In the 'Tools \rightarrow Device Programming' window, select tab "Memories".
- 2. Browse for C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\atmega\ ATmegaBOOT_168_atmega328.hex.
- 3. Click program.



Figure 2-2. Device Programming: Memories

mEDBG (ATML2323040200	00011546) - Device Programming	×
Tool Device mEDBG ATmega32	Interface Device signature Target Voltage	
Interface settings Tool information Device information Oscillator Calibration Memories	Device Erase Chip Erase now Flash (32 KB) C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\atmega\ATmegaBOOT_168_atmega328.hex C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\atmega\ATmegaBOOT_168_atmega328.hex Read	
Fuses Lock bits Production file	EEPROM (1 KB) Verify EEPROM after programming Verify EEPROM after programming Verify Read	
● OK	Close	

2.5 Configure Arduino IDE

- 1. Start the Arduino IDE from the Windows[®] start menu, or the folder the Arduino IDE was installed in.
- From menu Tools → Port select the correct COM port for the mEDBG. (See tip below on how to verify the COM port used for the mEDBG.)

Figure 2-3.	Arduino IDE: Serial Port
-------------	--------------------------

💿 sketch_feb25c	Arduino 1.6.0			
File Edit Sketch To	ools Help		~	
<pre>sketch_feb25c void setup() { // put your se</pre>	Auto Format Archive Sketch Fix Encoding & Reload Serial Monitor	Ctrl+T Ctrl+Shift+M		
}	Board Processor		•	
// put your ma	Port	1		COM3
)	Programmer Burn Bootloader		~	COM27





mEDBG com port can be viewed from Start \rightarrow Control panel \rightarrow Device Manager \rightarrow Ports as below:

File Action View Help Image: Second Sec	🚔 Device Manager					
 NORLT0561 Batteries Bluetooth Radios Computer Disk drives Display adapters Monitors Keyboards Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) ECP Printer Port (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	File Action View Help					
 NORLT0561 Batteries Bluetooth Radios Computer Disk drives Display adapters Human Interface Devices IDE ATA/ATAPI controllers IDE Grand Connectivity Keyboards Mice and other pointing devices Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 						
 Batteries Bluetooth Radios Computer Disk drives Display adapters Human Interface Devices IDE ATA/ATAPI controllers Monitors Monitors Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Intel(R) Ethernet IDB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	4 🚽 NORLT0561					
 Bluetooth Radios Computer Disk drives Display adapters Human Interface Devices IDE ATA/ATAPI controllers IDE ATA/ATAPI controllers Imaging devices Jungo Connectivity Keyboards Monitors Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Intel(R) Ethernet Is (USB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	Datteries					
 Computer Disk drives Display adapters Display adapters Human Interface Devices IDE ATA/ATAPI controllers ING IDE Connectivity Keyboards Monitors Monitors Network adapters IDE Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Intel(R) Ethernet IDB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	Bluetooth Radios					
 Disk drives Display adapters Display adapters Display adapters Display adapters IDE ATA/ATAPI controllers IDE ATA/ATAPI controllers Imaging devices Imaging devices Imaging devices Imaging devices Imaging devices Mice and other pointing devices Monitors Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Intel(R) Etrial Bus (USB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	🖂 🖓 Computer					
 Display adapters Human Interface Devices IDE ATA/ATAPI controllers IDE ATA/ATAPI controllers Imaging devices Jungo Connectivity Keyboards Monitors Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	Disk drives					
 Human Interface Devices IDE ATA/ATAPI controllers Imaging devices Imaging devices Imaging connectivity Keyboards Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Intel(R) Ethernet Isus (USB) Controller Ports (COM & LPT) Intel(R) Active Management Technology - SOI (COM3) 	Display adapters					
 IDE ATA/ATAPI controllers Imaging devices Imaging devices Imaging connectivity Keyboards Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Other devices Other devices Intel(R) Ethernet Connection I218-LM Tortal Controller Tortal Context (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	Human Interface Devices					
 Imaging devices Jungo Connectivity Keyboards Mice and other pointing devices Monitors Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Other devices Other devices Intel(R) Active Management Technology - SOI (COM3) 	IDE ATA/ATAPI controllers					
 Jungo Connectivity Keyboards Mice and other pointing devices Monitors Network adapters Personal Area Network) Personal Area Network Area Network Area Network) Personal Area Network Area Network Area Network Area Network) Personal Area Network A	Imaging devices					
 Keyboards Mice and other pointing devices Monitors Network adapters Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) ECP Printer Port (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	🛛 🖳 🖳 Jungo Connectivity					
 Mice and other pointing devices Monitors Metwork adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) ECP Printer Port (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	Keyboards					
 Monitors Network adapters Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) ECP Printer Port (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	Mice and other pointing devices					
 Network adapters Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Cisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Other devices Universal Serial Bus (USB) Controller Ports (COM & LPT) ECP Printer Port (LPT1) Intel(R) Active Management Technology - SOI (COM3) 	De Monitors					
Bluetooth Device (Personal Area Network) Bluetooth Device (RFCOMM Protocol TDI) Scisco Systems VPN Adapter for 64-bit Windows Intel(R) Dual Band Wireless-AC 7260 Device Intel(R) Ethernet Connection I218-LM Dother devices Diversal Serial Bus (USB) Controller Ports (COM & LPT) FCP Printer Port (LPT1) Title(R) Active Management Technology - SOL (COM3)	A Providence Andrews					
Bluetooth Device (RFCOMM Protocol TDI) Gisco Systems VPN Adapter for 64-bit Windows Portection I218-LM Dother devices Dother devices Ports (COM & LPT) FCP Printer Port (LPT1) Total (R) Active Management Technology - SOL (COM3)	Bluetooth Device (Personal Area Network)					
Cisco Systems VPN Adapter for 64-bit Windows Cisco Systems VPN Adapter for 64-bit Windows Cisco Systems VPN Adapter for 64-bit Windows	Bluetooth Device (RFCOMM Protocol TDI)					
Intel(R) Dual Band Wireless-AC 7260 Intel(R) Ethernet Connection I218-LM Image: Serial Bus (USB) Controller Image: Se	Cisco Systems VPN Adapter for 64-bit Windows					
Intel(R) Ethernet Connection I218-LM Gother devices Goth	Intel(R) Dual Band Wireless-AC 7260					
Other devices Other d	Intel(R) Ethernet Connection I218-LM					
Universal Serial Bus (USB) Controller Ports (COM & LPT) FCP Printer Port (LPT1) Trtel(R) Active Management Technology - SOI (COM3)	 Other devices 					
Ports (COM & LPT) Printer Port (LPT1) Printer Port (LPT1) Printel(R) Active Management Technology - SOI (COM3)	Inversal Serial Bus (USB) Controller					
''''''''''''''''''''''''''''''''''	Ports (COM & LPT)					
Intel(R) Active Management Technology - SOL (COM3)	ECP Printer Port (LPT1)					
	Intel(K) Active Management Technology - SOL (COM3)					
mEDBG Virtual COM Port (COM27)	mEDBG Virtual COM Port (COM27)					

3. Board can be selected according to Table 2-2.

Table 2-2. Arduino IDE: Board

Xplained Mini	Board to select in Arduino IDE
ATmega328P Xplained Mini	Arduino Nano: ATmega328
ATmega168PB Xplained Mini	Arduino Nano: ATmega168

Here, select: Tools \rightarrow Boards \rightarrow Arduino Nano.

Select: Tools \rightarrow Processor \rightarrow ATmega328.

Figure 2-4. Arduino IDE: Board and Processor





2.6 Upload the Program



ATmega328P Xplained Mini board will be connected to Arduino IDE and will start blinking LED.

1. In Arduino IDE, select File \rightarrow Examples \rightarrow 01.Basics \rightarrow Blink.

Figure 2-5. Arduino IDE: Examples-Blink

	sketch_feb25c Arduino 1.6.0)				
File	Edit Sketch Tools Help		_			
	New Open Sketchbook	Ctrl+N Ctrl+O				
	Examples			01.Basics	1	AnalogReadSerial
	Close Save	Ctrl+W Ctrl+S		02.Digital 03.Analog		BareMinimum Blink
	Save As Upload Upload Using Programmer	Ctrl+Shift+S Ctrl+U Ctrl+Shift+U		04.Communication 05.Control 06.Sensors	1	DigitalReadSerial Fade ReadAnalogVoltage
	Page Setup Print	Ctrl+Shift+P Ctrl+P		07.Display 08.Strings 09.USB	* *	
	Preferences	Ctrl+Comma		10.StarterKit	•	
	Quit	Ctrl+Q		ArduinoISP		
				Audio Bridge EEPROM Esplora Ethernet	· · · · ·	

2. Upload the sketch by clicking "Upload" button in Arduino IDE.





RESULT Messages will be appeared at the bottom of window 'Compiling Sketch', 'Uploading', 'Done uploading'.

- 3. Select File \rightarrow Save As... and save the sketch 'Blink'. You can select any path to save sketch.
- 4. Observe the blinking LED in ATmega328P Xplained Mini board.

We have successfully connected ATmega328P Xplained Mini board to the Arduino IDE.



3 Assignment 2: Creating Arduino Sketches in Atmel Studio



Atmel Studio's Arduino Extension allows any Arduino sketch to be written, compiled and uploaded to any Arduino while inside Atmel Studio with rich user interface and professional features that it provides.

3.1 Download Extension

- 1. Open Atmel Studio.
- Select Tools → Extension Manager. The Extension Manager Window opens and by default it shows the installed extensions.
- 3. Click the "Available Downloads" option.

Figure 3-1. Atmel Studio: Extension Manager



- 4. Select "Arduino IDE for Atmel Studio" and click the "Download" icon.
- 5. The "Sign in to Extension Manager Dashboard" window opens and asks you to sign-in/register. Then Sign-in using the email-id and password provided while registering.

INFO If you have not registered already, kindly register. Then close the "Sign in to Extension Manager Dashboard" window. Atmel will send you a confirmation email to the email-id provided. Click the link to confirm your email-id. Repeat steps 2 to 4 and in the "Sign in to Extension Manager Dashboard" and Sign-in using the email-id and password provided while registering.

6. The "Arduino IDE for Atmel Studio" extension will be downloaded, then install it.



3.2 Create Sketch

1. Open Atmel Studio.

()		
	TO	DO

Verify Arduino 1.6 is listed in Atmel Studio and select it.



i info

If it is not listed, click <Configuration Manager>. Select Arduino 1.6. Add the Arduino installation directory path as shown in below figure. Select 'OK'.

Figure 3-2. Configure Visual Micro

Config	ure Visual Micro							
Vin	Please specify a micro-controller Ide location Visual Micro needs to know where, on your computer, application(s) such as the Arduino.exe are located. If an application is not already installed then please download it using the 'Download/Install' button. Support for some platforms is still under development and we value feed back in our forum Arduino 1.6 There the ide folder location (example: c:\arduino) c:\Program Files (x86)\Arduino Ontionally enter a stetchbook location (also affects location of custom libraries (hardwarel)							
¥ н н ц	Coverage Cancel Download/Install Ide OK Cancel Ok elp and information Ok ow to test a new installation Ok etting_started Ok ow to debug arduino Ok							

- 2. Select File \rightarrow New \rightarrow Sketch Project.
- 3. Enter name for sketch project as shown in below figure and select 'OK'.







led_toggle.ino sketch will be created at location c:\Arduino as it is a default location as shown in Figure 3-2.

4. Edit the code to toggle the LED.

```
int led = 13;
void setup()
{
    /* add setup code here */
    pinMode(led, OUTPUT);
}
void loop()
{
    /* add main program code here */
    digitalWrite(led,HIGH);
    delay(500);
    digitalWrite(led, LOW);
    delay(500);
}
```

5. Select the board as Arduino Nano w/ATmega328.



6. Select mEDBG COM port number.



秦 led_toggle - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
🗄 🕶 🖶 📨 🍃 🛃 🎒 👗 💺 隆 🖉 - 🔍 - 💭 - 🖳 🔚 🔍 🍞 🜗 💹 Debug
🐑 📨 🐝 🍋 🔓 🥵 🕰 📲 📲 🔛 💷 🇇 🗉 🕨 💷 🖓 🖓 🐨 🐨 🐨 Hex 🖼 🗸
🕴 Arduino 1.6 🔹 Arduino Nanc 🔹 🎖 🚽 COM3 🔹 😓 🖬 🖓 🚽
COM3 - Intel(R) Active Management Technology - SOL
COM27 - mEDBG Virtual COM Port

7. Download program by clicking icon .

RESULT LED on ATmega328P Xplained Mini board will start toggling.



4 Transit to the Atmel Studio IDE with Existing Arduino Sketches

4.1 **Project Creation**

- 1. Open Atmel Studio and create a new "GCC C++ Executable Project give the Project", give the project a reasonable name and select the ATmega328P as the device.
- 2. Make sure to make a note of the folder where the project is created.



3. In the resulting solution, right click and remove the project cpp file (here *ATmega328P_1.cpp*) as shown in below figure.



- 4. In Windows Explorer:
 - a. Copy your sketch (earlier saved Arduino sketch Blink/Blink.ino) and place it to your new Atmel Studio project subdirectory, where *ATmega328P_1.cpp* is (still there in Explorer, even though it was removed from the Atmel Studio project).



- ➤ Computer ➤ Local Disk (C:) ➤ workspace ➤ ATmega328P_1 ➤ ATmega328P_1 ➤ \leftarrow Organize • Include in library -Share with -New folder Name Date modified 🚖 Favorites 💻 Desktop 👌 Debug 25.02.2015 12:15 🐌 Downloads Tmega328P_1.cpp 25.02.2015 12:15 laces Recent Places ATmega328P_1.cppproj 25.02.2015 12:15 8 Blink.cpp 25.02.2015 10:48 💻 Desktop 🚞 Libraries Documents 🕜 Git
- b. Rename your sketch extension to .cpp: Blink.ino \rightarrow Blink.cpp. See below figure.

5. In the Atmel Studio select Project \rightarrow Show All Files.

ATmega_328P_1 - AtmelStudio							
File Edit View VAssistX ASF	Project	Build	Debug	Tools	Window	Help	
🛐 • 🎒 💷 • 📂 📓 🌒 👗 🗸	A	dd New S	ketch Ite	m			•
: 🔁 🖾 🖓 🍋 🔓 🖓 🐝 🗆	A	dd/Impor	t Sketch	Library			•
Arduino 1.6 • Arduino Nanc •	80 A	SF Wizard	I.				
	SI SI	now All Fi	les				
	Se	et as Start	Up Proje	ct			
	A	Tmega_32	28P_1 Pro	perties		Alt+F7	
	Ma SI	now Sketc	h Core a	nd Libra	ry Sources		

 In the Solution Explorer window ,right click Blink.cpp file and select "Include In Project" as shown in figure below:





4.2 Configuring Compiler Symbols

- In the Solution Explorer right click the project and select Properties or go to Project → [project name] Properties... in the file menu (Alt+F7).
- 2. Under Toolchain \rightarrow AVR/GNU C Compiler \rightarrow Symbols , add:
 - F_CPU=16000000L
 - USB_VID=null
 - USB_PID=null
 - ARDUINO=160

INFO These are Arduino Specific Symbols.

3. Repeat for Toolchain \rightarrow AVR/GNU C++ Compiler \rightarrow Symbols.

ATTREgaszor_1 - Attrestudio		
File Edit View VAssistX ASF Project Build Debu	g Tools Window Help	
i 🛅 • 🕮 🖬 • 💕 🔜 🐼 X 🐁 🛍 🕫 • 🦊 •] • 🖾 🔢 🗣 🕪 🖗 Debug 🔹 🔯 main 🔹 🔹 🖓 🔁 🔮 🝻 🗔 • ෫ 🗄 🚎	[글일] 그 뒤로 취로 성용 및 .
i 🔁 🔂 🗞 🗠 🔓 여 🗛 🔬 🖬 🖕 i 💵 💷 🌩 🗉	🕨 🕼 🕫 🗊 🖆 🖅 🗄 Hex 📓 🗸 👷 🖾 💷 🥥 💷 💭 🗳 🚽 🛗 📥 🖕 🗑 🛲 ATmega328P 🥤 No Tool 🖕	
Arduino 1.0.x • • 😵 🚽 COM3	- 😸 💷 🖕	
ATmega328P 1* ×		 Solution Explorer
Build Configuration: Active (Debug) Configuration Manager	Platform: Active (AVR)	Solution 'ATmega328P_1' (1 project) ATmega328P_1 Dependenctes
Toolchain Configuration Manager Device Image: Configuration Manager Tool Image: Configuration Manager Advanced Image: Completer Image: Completer Image: Completer Image: Completer	AVR/GNU C Compiler Symbols Defined symbols (-D) DEBUG F, CPU-1600000L USB yID=null USB yID=null URDefined symbols (-U)	 ⇒ Dependentles ⊇ Urburt Files > ⊇ Libraries ☆ Blinkcpp

4.3 Configuring Compiler Directories

- 1. Under Toolchain \rightarrow AVR/GNU C Compiler \rightarrow Directories add.
- 2. C:\Program Files (x86)\Arduino\hardware\arduino\avr\cores\arduino.
- 3. C:\Program Files (x86)\Arduino\hardware\arduino\avr\variants\standard.
- 4. C:\Program Files (x86)\Arduino\hardware\arduino\avr\variants\eightanaloginputs.
- 5. Repeat for Toolchain \rightarrow AVR/GNU C++ Compiler \rightarrow Directories.

WARNING Make sure you deselect "Relative Path" while adding these paths.



ATmega328P_1*	× _
Build Build Events	Configuration: Active (Debug) Platform: Active (AVR)
Toolchain*	Configuration Manager
Device	
Tool	▲ I AVR/GNU Common AVR/GNU C Compiler ⇒ Directories
Advanced	 General AVR/GNU C Compiler AVR/GNU C Compiler General Symbols Directories Optimization Debugging Warnings Miscellaneous Symbols Directories Optimization General Symbols Symbols Miscellaneous

4.4 Add Arduino Dependency Files

- 1. Right click the project; go to "Add \rightarrow Existing Item..."
- 2. In the dialog box that opens, aim the browser at C:\Program Files (x86)\Arduino\hardware\arduino\avr\cores\arduino
 - a. In the "File Name" box, type *.c* <enter>
 - b. Multi-select all files and select "Add as Link"

🔊 🗢 📙 « Local Dis	sk (C:) • Program Files (x86) • Arduin	o ► hardware ► arduino ► av	r 🕨 cores 🕨 arduino	-	Search arduino	P	
organize 🔹 New fold	er					0	-
Atmel Studio 6.2	Name	Date modified	Туре	Size		*	- Solution Explore
6.2	💓 abi.cpp	08.02.2015 23:39	C++ Source File	2 KB			D L D
	👸 CDC.cpp	08.02.2015 23:39	C++ Source File	6 KB			Colution 141
Favorites	👸 HardwareSerial.cpp	08.02.2015 23:39	C++ Source File	8 KB			A ATmega
E Desktop	HardwareSerial0.cpp	08.02.2015 23:39	C++ Source File	3 KB			🔤 Depe
Downloads	👔 HardwareSerial1.cpp	08.02.2015 23:39	C++ Source File	3 KB			🔤 Outpu
Recent Places	👔 HardwareSerial2.cpp	08.02.2015 23:39	C++ Source File	2 KB			Librar
	WardwareSerial3.cpp	08.02.2015 23:39	C++ Source File	2 KB			Blink.e
Desktop	👔 HID.cpp	08.02.2015 23:39	C++ Source File	14 KB			
Libraries	a hooks.c	08.02.2015 23:39	C Source File	2 KB			
Documents	IPAddress.cpp	08.02.2015 23:39	C++ Source File	2 KB			
a Git	👔 main.cpp	08.02.2015 23:39	C++ Source File	2 KB			
Music	👔 new.cpp	08.02.2015 23:39	C++ Source File	2 KB		=	
Pictures	👔 Print.cpp	08.02.2015 23:39	C++ Source File	6 KB			
 Subversion 	👔 Stream.cpp	08.02.2015 23:39	C++ Source File	8 KB			
Judeos	👔 Tone.cpp	08.02.2015 23:39	C++ Source File	15 KB			
Honrao Rupali	👸 USBCore.cpp	08.02.2015 23:39	C++ Source File	14 KB			
Computer	WInterrupts.c	08.02.2015 23:39	C Source File	9 KB			
Network	👩 wiring.c	08.02.2015 23:39	C Source File	10 KB			
Control Panel	øiring_analog.c	08.02.2015 23:39	C Source File	8 KB			
Recycle Bin	øiring_digital.c	08.02.2015 23:39	C Source File	5 KB			
	iviring_pulse.c	08.02.2015 23:39	C Source File	4 KB			
	øi wiring_shift.c	08.02.2015 23:39	C Source File	2 KB			
	👸 WMath.cpp	08.02.2015 23:39	C++ Source File	2 KB			
	@ WString.cpp	08.02.2015 23:39	C++ Source File	17 KB		Ŧ	



4.5 Build Solution

1. Add # include "arduino.h" at the top of Blink.cpp



i info

Now all the Arduino dependencies have been added and project can be compiled and loaded into the target.

2. In the file menu, select Build \rightarrow Build Solution. The build should finish successfully with no errors.



4.6 Plug in ATmega328P Xplained Mini Board

- In the Solution Explorer right click the project and select Properties or go to Project→ [project name] Properties... in the file menu (Alt=F7).
- 2. Under Tools, select mEDBG and debugWIRE as interfaces.

File Edit View VAsistX ASF Project Build Debug Tools Window Help Image: Construction of the set of t	🖗 ATmega328P_1 - AtmelStudio	
We - the set of the set	File Edit View VAssistX ASF Project Build Debug Tools Window Help	
Arduino 10.x Configuration N/A Platform: Selected Advanced Programming settings Ease only program area Platform: Coccpp Glatfatfatform Platform: Selected Selected Settings Platform: Selected Settings Platform: Selected Settings Platform: Settings Platform: Settings Setings Settings Settings Settings Settings Setin	i 🐜 = 🖽 🖼 = 💕 🛃 🕼 🙏 🐁 🖄 🖉 = 🔍 = 💭 = 🖏 🖳 🍕 🍞 🔹 🕨 Debug 💿	• 🙆 main 🔹 💀 💀 🕾 🌌 🏄 🏼 • 🚽 龍 🖀 😫 🗆 🗣 🖓 🗛 😣
Arduno 10x Solution Explorer Build Build Solution XImega328P.1* X Build Events Solution XImega328P.1* (A Torchain Selected debugger/programmer Device Output Files Tool Advanced Programming settings Erase only program area Image: Proserve EEPROM Bebug settings Image: Reproduct Settings Image: Reproduct Settings Image: Reproduct Settings	i 🗒 🖾 🗞 😭 😭 🗛 🔬 🕼 📄 💹 🖬 🗢 🖬 🖉 🖓 🗐 🗐 🗐 🐨 🐨 Hex 📓 - 📑	🕴 🐖 🛒 🔤 📮 🗄 📅 🚟 👗 🚽 ATmega328P 🦹 debugWIRE on mEDBG (ATML2323040200011546) 😑
Blinkcpp ATmega328P_1* × Solution Explorer Build Build Configuration: N/A Platform: N/A Imagea328P_1* (Build Selected debugger/programmer Imagea328P_1* (Imagea328P_1* (Device Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Toolchain Device Imagea328P_1* (Imagea328P_1* (Programming settings Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Programming settings Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Programming settings Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (Imagea328P_1* (<		
Blinkcpp Afmega328P_1* × Solution Explorer Build Build Configuration: NA Platform: NA Build Build Events Selected debugger/programmer Selected debugger/programmer Selected debugger/programmer Device Tool Advanced Programming settings Selected debugwliRE • Selected debugwliRE • Programming settings Erase only program area • Preserve EEPROM Selected all flash memory except HID.cpp Witche Build Settings Ø Cache all flash memory except Selected all flash memory except Selecter all flash memory except		
Build Configuration: NA Platform: NA Imaga328P1:1(1) Build Events Selected debugger/programmer Imaga328P1:1(1) Imaga328P1:1(1) Device Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Toolchain Device Imaga328P1:1(1) Imaga328P1:1(1) Advanced Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Programming settings Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Programming settings Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga328P1:1(1) Imaga	Blink.cpp ATmega328P_1* ×	- Solution Explorer
Build Configuration: N/A Platform: N/A Platform: N/A Image: Configuration: N/A Platform: N/A Build Build Events Selected debugger/programmer Image: Configuration: N/A Image: Configuration: N/A <th></th> <th></th>		
Build Events > Imagea328P_1 Toolchain Device Tool Imagea328P_1 Output Files Imagea328P_1 Tool Imagea328P_1 Advanced Programming settings Erase only program area • Imagea328P_1 Preserve EEPROM Imagea328P_1 Debug settings Imagea328P_1 Imagea328P_1 Imagea328P_1	Build Configuration: N/A Platform: N/A	Solution 'ATmega328P_1' (1 proj
Toolchain Dependences Device Ibraries Tool Advanced Programming settings Ibraries Erase only program area • Ibraries Preserve EEPROM Ibraries Debug settings Ibraries V Keep timers running in stop mode Ibraries V Cache all flash memory except Ibraries	Build Events	ATmega328P_1
Device becue decuger/programmer Tool Advanced Programming settings Frase only program area • Preserve EEPROM Debug settings Verserve EEPROM Verserve EEPROM Ver	Toolchain Salastad dahuggar/programmar	Dependencies Dependencies
Tool Advanced ImEdBG • ATML2323040200011546 • Interface (debugWIRE •) ImedBG • ATML2323040200011546 • Interface (debugWIRE •) Advanced Programming settings ImedBG • ATML232304020011546 • Interface (debugWIRE •) Erase only program area • ImedBG • ATML232304020011546 • Interface (debugWIRE •) ImedBG • ATML232304020011546 • Interface (debugWIRE •) ImedBG • ATML232304020011546 • Interface (debugWIRE •) ImedBG • ATML232304020011546 • Interface (debugWIRE •) ImedBG • ATML232304020011546 • ImedBG • Imed	Device	▶ ⊇ Libraries
Advanced Programming settings Advanced Programming settings Ease only program area Preserve EEPROM Debug settings Keep times running in stop mode Cache all flash memory except Tornecpp Tornecpp Tornecpp Tornecpp Tornecpp Tornecpp Tornecpp Tornecpp	Tool mEDBG • ATML2323040200011546 • Interface: debugWIRE •	abi.cpp
Advanced Programming settings Frase only program area Proserve EEPROM Debug settings Cache all flash memory except Torcep Torcep Torcep Torcep Torcep Torcep Torcep		🖹 Blink.cpp
Programming settings Frase only program area • Preserve EEPROM Debug settings Keep times running in stop mode Cache all flash memory except HardwareSerial3.cpp HardwareSe	Advanced	a) CDC.cpp
Image: Stating and Sta	Programming settings	all HardwareSerial.cpp
Erase only program area Initial dividual Section 2.000 Preserve EEPROM Initial dividual Section 2.000 Debug settings Initial dividual Section 2.000 If Activity Section 2.000 Initial dividual Section 2.000 If Activity Section 2.00	riogramming security.	al HardwareSerial1.cpp
■ Preserve EEPROM ■ HardwareSerial3.cpp Debug settings ■ holos c ☑ Keep timers running in stop mode ■ holos c ☑ Cache all flash memory except ■ main.cpp ④ holos c ■ holos c ☑ Timer, cpp ■ Timer, cpp ☑ Timer, cpp ■ timer,	Erase only program area 🔻	a HardwareSerial2.cpp
A HD.cpp A HD.cp	Preserve FEPROM	al HardwareSerial3.cpp
Debug settings		a HID.cpp
☑ Keep timers running in stop mode ⓐ Il PAdoress.cpp ☑ Cache all flash memory except ⓐ Il nin.cpp ☑ Stream.cpp ⓐ Stream.cpp ⑥ Stream.cpp ⓐ Stream.cpp ⑧ Tone.cpp ⓐ Tone.cpp	Debug settings	in hooks.c
	Keep timers running in stop mode	圖 IPAddress.cpp 剧 main.cop
al Print.cpp al Stream.cpp al Tone.cpp	Cache all flash memory except	al namepp
al Stream.cpp al Tome.cpp		副 Print.cpp
얇 Tone.cp		🛐 Stream.cpp
		a Tone.cpp
al USBCORE.cpp		all USBCore.cpp
a) Wilnerruptsc		Munterrupts.c
gi wingo		a wing.c
a wring analysis		a) wiring analogic

i info

At some point the system may want to update the debugger. Let it update.



- 3. Under Toolchain → AVR/GNU C Compiler → Optimization. Select Optimization level 'Optimize for size(-Os)'.
- Under Toolchain → AVR/GNU C++ Compiler → Optimization. Select Optimization level 'Optimize for size(-Os)'.



4.7 Debugging

1. Select debug and click 'Start Debugging and Break'.

ATmega328P_1 - AtmelStudio			
File Edit View VAssistX ASF Project Build	Deb	ug Tools Window Help	
🗛 • 🔁 🗉 • 📂 🛃 🗿 🐰 🖦 🖄 🔊 • 🤉		Windows	•
🐑 🗁 🎇 🍋 🔓 🥵 💁 📲 📜 💷 📔		Start Debugging and Break	Alt+F5
Arduino 1.0.x 🔹 🔹 😵 🚽 COM3		Stop Debugging	Ctrl+Shift+F5
Blink.cpp × ATmega328P_1*	∉⊳	Start Without Debugging	Ctrl+Alt+F5
→ Blink.cpp - C:\workspa		Disable debugWIRE and Close	
pin the on-board LED is connected	•	Continue Execute Stimulifile	F5
	Ę	Set Stimulifile	
This example code is in the publi		Restart	
modified 8 May 2014 by Scott Fitzgerald	61	QuickWatch	Shift+F9
	۶I	Step Into	F11
		Step Over	F10
<pre># include "arduino.h"</pre>	Ĩ	Step Out	Shift+F11
<pre>// the setup function runs once whe proid setup() {</pre>		Run To Cursor	Ctrl+F10
// initialize digital pin 13 as a	Î	Reset	Shift+F5
pinMode(13, OUTPUT);		Percepio Trace	•
_ }		Toggle Breakpoint	F9
<pre>// the loop function runs over and</pre>		New Breakpoint	•
□ void loop() {	2	Delete All Breakpoints	Ctrl+Shift+F9
digitalwrite(13, HiGH); // turr delay(1000); // wait		Clear All DataTips	
digitalWrite(13, LOW); // turr		Export DataTips	
delay(1000); // wait		Import DataTips	
		Options and Settings	
	_		





RESULT

If the DWEN fuse is not enabled and error message is displayed. Click 'Yes' and Studio will use the ISP to set the fuse as shown below.



The Debugger is started and breaks in main. You are now ready to start debugging.





- C 🖄 🗗 🖌 🔛 🔂 - C		Windows	•	SPL_QUARTER_SPEED - 😒 💆
i 🔁 🍜 🎘 🍋 🎖 🎖 0, 🔬 🍱 🚽 💷 🔒	911	Start Debugging and Break	Alt+F5	a 🖬 🗃 📓 📲 🗃 🖶 📥 🚛 I 🛥 A
🕴 Arduino 1.0.x 🝷 Arduino Nanc 🗣 🧋 COM28		Stop Debugging	Ctrl+Shift+F5	
Disassembly main.cpp × Blink.cpp Atmo	4	Start Without Debugging Disable debugWIRE and Close	Ctrl+Alt+F5	
→ main		Continue	F5	
#Include CALIDITIO.	4	Execute Stimulifile		
⊟int main(void)	Ś	Set Stimulifile		
{ init():		Restart		
20120(7)	11	Break All	Ctri+F5	
<pre>#if defined(USBCON)</pre>	64	QuickWatch	Shift+F9	
USBDevice.attach();	®I	Step Into	F11	
WEIGHT	(J)	Step Over	F10	
<pre>setup();</pre>	°1	Step Out	Shift+F11	
for (···) I	•1	Run To Cursor	Ctrl+F10	
loop();	Ť	Reset	Shift+F5	
if (serialEventRun) serialE		Percepio Trace	•	
3		Toggle Breakpoint	F9	
return 0;		New Breakpoint		
3	2	Delete All Breakpoints	Ctrl+Shift+F9	
		Clear All DataTips		
		Export DataTips		
		Import DataTips		
		Options and Settings		

 (In the 'Solution Explorer' file Blink.cpp) Go to the line in the source code where you want to insert a breakpoint, right click and select Breakpoint → Insert Breakpoint.







3. Run to Breakpoint by clicking "continue". You can pause and continue the execution as per requirement.



4. Exit debug mode: Select Debug \rightarrow Disable debugWIRE and Close.

File Edit View VAssistX ASF Project Build	Deb	ug Tools Window Help	
9 - C 🙇 📣 🔏 😹 😂 - C		Windows	•
i 😨 🖾 🖓 🍋 🔓 😚 😘 🔬 🗳 🚽 i 💷 📔	M	Start Debugging and Break	Alt+F5
😳 Arduino 1.0.x 👻 Arduino Nanc 👻 🎖 🖕		Stop Debugging	Ctrl+Shift+F5
Disassembly main con Blink con X Atm	₽	Start Without Debugging	Ctrl+Alt+F5
		Disable debugWIRE and Close	
Blink.cpp C:\workspa	Þ.	Continue	F5
Blink	-	Execute Stimulifile	
Turns on an LED on for one second	4	Set Stimulifile	
This example code is in the publi		Restart	
*/	п	Break All	Ctrl+F5
<pre>#include "arduino.h"</pre>	63	QuickWatch	Shift+F9
// D1 424 100 111	91	Step Into	F11

WARNING It is important to disable debugWIRE.

Disabling debugWIRE resets the target and the DWEN fuse is reset, you will be able to use the ISP interface again. Having the DWEN fuse programmed enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption of the AVR while in sleep modes. The DWEN Fuse should therefore always be disabled when debugWIRE is not used.

5. Reset the board and observe that the LED blinking.

INFO



5 ATmega328P Application

We are going to create a simple application to read a light sensor using the ADC, read a temperature sensor using the I²C interface and store data in the SD card using the SPI interface. Also the light sensor value and temperature value is transmitted through the EDBG COM port.

For the I²C the "Wire" library from Arduino is required and for SD card the "SD library" is required to be included.

5.1 Compiler Setup

- 1. Right Click Project (here ATmega328P_1) and select Properties.
- 2. Under Toolchain → AVR/GNU C Compiler → Directories add (note that the previously added directories should still be in the list)

C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\Wire

C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\Wire\utility

C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\SPI

C:\Program Files (x86)\Arduino\libraries\SD\src

C:\Program Files (x86)\Arduino\libraries\SD\src\utility

3. Repeat for Toolchain \rightarrow AVR/GNU C++ Compiler \rightarrow Directories.

WARNING Make sure you uncheck "Relative Path" while adding these paths.

Add Include Paths (-I)	
Include Paths (-I)	
Relative Path	
	OK Cancel





Figure 5-1. Atmel Studio: Compiler Setup



5.2 Add Dependency Files



Add the .cpp source files from libraries to your actual Atmel Studio project.

5.2.1 For Wire Library

- 1. Right click the project; go to "Add \rightarrow Existing Item..."
- 2. In the dialog box that opens, aim the browser at
 - a. C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\Wire
 - b. Select Wire.cpp file and add 'As a link'
 - c. Repeat the same for "wire\utility" subdirectory, select twi.c file and add 'As a link'



Add Existing Item - ATmeg	ga328P_1				×
🕞 💬 🛡 📕 « Program F	iles (x86) ► Arduino ► hardware ► arduino ►	avr ► libraries ► Wi	ire 🕨 🔻	Search Wire	م
Organize New folde	r			• = = • •	
Atmel Studio 6.2 🗂	Name	Date modified	Туре	Size	
1.6.2	🐌 examples	25.02.2015 10:34	File folder		
=	💩 utility	25.02.2015 10:34	File folder		
涬 Favorites	keywords.txt	08.02.2015 23:39	TXT File	1 KB	
E Desktop	Ø library.properties	08.02.2015 23:39	PROPERTIES File	1 KB	
🗼 Downloads 👘	👸 Wire.cpp	08.02.2015 23:39	C++ Source File	8 KB	
Skecent Places	🕢 Wire.h	08.02.2015 23:39	H File	3 KB	
Desktop Libraries Cit					
File nam	ne: Wire.cpp		•	All Files (*.*)	
				Add 🔻	Cancel
	General			Add As Link	

5.2.2 For SD Library

- 1. Right click the project; go to "Add \rightarrow Existing Item..."
- 2. In the dialog box that opens, aim the browser at
 - a. C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\SPI
- 3. Select SPI.cpp file and add 'As a link'
- 4. Repeat the same for C:\Program Files (x86)\Arduino\libraries\SD\src
 - a. Multiple Select SD.cpp, File.cpp file.
 - b. Repeat the same for same for "SD\utility" subdirectory, multiple select SdVolume.cpp, Sd2Card.cpp, SdFile.cpp, and add 'As a link'.



DesktopDate modifiedTypeSizeIbraries <th></th> <th>Nama</th> <th>Data modified</th> <th>Turne</th> <th>Sizo</th> <th></th> <th></th>		Nama	Data modified	Turne	Sizo		
 i bisktop i bi	Deskton	Name	Date modified	туре	Size		
introduced introduced <td< td=""><td></td><td>FatStructs.h</td><td>08.02.2015 23:39</td><td>H File</td><td>17 KB</td><td></td><td></td></td<>		FatStructs.h	08.02.2015 23:39	H File	17 KB		
Image: Solution of Side Side Side Side Side Side Side Side		💕 Sd2Card.cpp	08.02.2015 23:39	C++ Source File	21 KB		
Image: Solution of the solution	Git	👩 Sd2Card.h	08.02.2015 23:39	H File	10 KB		
Indict Image: A stress of SdFat.h 08.02.2015 23:39 H File 22 KB Image: A stress of SdFat.h 08.02.2015 23:39 H File 9 KB Image: A stress of SdFat.h 08.02.2015 23:39 H File 9 KB Image: A stress of SdFat.h 08.02.2015 23:39 H File 3 KB Image: A stress of SdFat.h 08.02.2015 23:39 C++ Source File 42 KB Image: A stress of SdFile.cpp 08.02.2015 23:39 H File 8 KB Image: A stress of SdFile.cpp 08.02.2015 23:39 C++ Source File 10 KB Image: A stress of SdFat.h 08.02.2015 23:39 C++ Source File 10 KB	Music	Sd2PinMap.h	08.02.2015 23:39	H File	13 KB		
Includes Image: Solution of SdFattmainpage.h 08.02.2015 23:39 H File 9 KB Image: SdFattutil.h 08.02.2015 23:39 H File 3 KB Image: SdFattutil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFattutil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFattutil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFattutil.h 08.02.2015 23:39 C++ Source File 10 KB Image: SdFattutil.h 08.02.2015 23:39 C++ Source File 10 KB	Pictures	👩 SdFat.h	08.02.2015 23:39	H File	22 KB		
Image: Solution of Videos Image: Solution of SdFatUtil.h 08.02.2015 23:39 H File 3 KB Image: SdFatUtil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFatUtil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFatUtil.h 08.02.2015 23:39 C++ Source File 42 KB Image: SdFatUtil.h 08.02.2015 23:39 C++ Source File 10 KB Image: SdVolume.cpp 08.02.2015 23:39 C++ Source File 10 KB	Subversion	👩 SdFatmainpage.h	08.02.2015 23:39	H File	9 KB		
Windows Microsoft 08.02.2015 23:39 C++ Source File 42 KB Computer SdInfo.h 08.02.2015 23:39 H File 8 KB Metwork SdVolume.cpp 08.02.2015 23:39 C++ Source File 10 KB	Videos	SdFatUtil.h	08.02.2015 23:39	H File	3 KB		
Computer Image: SdInfo.h 08.02.2015 23:39 H File 8 KB Network SdVolume.cpp 08.02.2015 23:39 C++ Source File 10 KB Control Panel Control Panel C++ Source File 10 KB	Honrao Runali	💕 SdFile.cpp	08.02.2015 23:39	C++ Source File	42 KB		
Network Ø SdVolume.cpp 08.02.2015 23:39 C++ Source File 10 KB Control Panel 0	Computer	Ø SdInfo.h	08.02.2015 23:39	H File	8 KB		
Control Panel	Network	💕 SdVolume.cpp	08.02.2015 23:39	C++ Source File	10 KB		
,	Control Panel						
Recycle Bin	Recycle Bin						
	~						
~	File na	ame: "SdVolume.cpp" "Sd2Card.cpp" "S	dFile.cop"		 All Files 	(* *)	

5.3 Developing Application

EXECUTE Rename current Blink.cpp file to sensors.cpp file and delete LED blinking code. (I.e. delete setup() and loop().)

EXECUTE Include header files for SD and Wire library at the top in sensors.cpp file.

#include <Wire.h>
#include <SD.h>

```
EXECUTE
```

FE Add definitions regarding temperature sensors.

<pre>#define AT30TSE_TEMPERATURE_TWI_ADDR #define AT30TSE_TEMPERATURE_REG #define AT30TSE_TEMPERATURE_REG_SIZE #define AT30TSE NON VOLATILE REG</pre>	0x4F 0x00 2 0x00
#define Al30ISE_NON_VOLATILE_REG	0x00
<pre>#define AT30TSE_CONFIG_RES_9_bit</pre>	0
<pre>#define AT30TSE_CONFIG_RES_10_bit</pre>	1
<pre>#define AT30TSE_CONFIG_RES_11_bit</pre>	2
<pre>#define AT30TSE_CONFIG_RES_12_bit</pre>	3
<pre>uint16_t resolution = AT30TSE_CONFIG_RE</pre>	S 12 bit;





```
Wire.requestFrom(AT30TSE_TEMPERATURE_TWI_ADDR, reg_size);
while(Wire.available())
{
        buffer[i] = Wire.read(); // receive a byte as character
        i++;
    }
    return (buffer[0] << 8) | buffer[1];
}</pre>
```





```
double at30tse_read_temperature()
{
       /* Read the 16-bit temperature register. */
       uint16_t data = at30tse_read_register(AT30TSE_TEMPERATURE_REG,
       AT30TSE NON VOLATILE REG,
       AT30TSE TEMPERATURE REG SIZE);
       double temperature = 0;
       int8_t sign = 1;
       /*Check if negative and clear sign bit. */
       if (data & (1 << 15)){
              sign *= -1;
              data &= ~(1 << 15);</pre>
       }
       /* Convert to temperature */
       switch (resolution){
              case AT30TSE_CONFIG_RES_9_bit:
              data = (data >> 7);
              temperature = data * sign * 0.5;
              break;
              case AT30TSE_CONFIG_RES_10_bit:
              data = (data >> 6);
              temperature = data * sign * 0.25;
              break;
              case AT30TSE_CONFIG_RES_11_bit:
              data = (data >> 5);
              temperature = data * sign * 0.125;
              break;
              case AT30TSE_CONFIG_RES_12_bit:
              data = (data >> 4);
              temperature = data * sign * 0.0625;
              break;
              default:
              break;
       }
       return temperature;
}
```

EXECUTE Add initializing function setup().

i info

As soon as we start writing a function name in Atmel Studio possible function names are listed as shown below.





Atmel

EXECUTE Add a variable and function loop().

```
bool sensor_flg=0;
void loop() {
      sensorValue = analogRead(analogInPin);
                                                          // read the ADC:
      if (sensorValue>=500)
      {
             sensor_flg=1;
      }
      if (sensor_flg==1)
                                     // write data in file on particular sensor vale
      {
             sensor_flg=0;
             myFile = SD.open("data.txt", FILE_WRITE);
                                // if the file opened okay, write to it:
             if (myFile)
             {
                    myFile.print("sensor = " );
                    myFile.print(sensorValue);
                    myFile.print(" temp = ");
                    myFile.print(temp result);
                    myFile.print("\n");
                    myFile.close(); // close the file:
             }
             else
             {
                    // if the file didn't open, print an error:
                    Serial.println("error opening data.txt");
             }
             // re-open the file for reading:
             myFile = SD.open("data.txt");
             if (myFile)
             {
                    // read from the file until there's nothing else in it:
                    while (myFile.available()) {
                          Serial.write(myFile.read());
                    }
                    // close the file:
                    myFile.close();
                    } else {
                    // if the file didn't open, print an error:
                    Serial.println("error opening data.txt");
             }
       }
      delay(500);
}
```



泰 ATmega_328P_1 - AtmelStudio			
File Edit View VAssistX ASF Project	Build	d Debug Tools Window	Help
🛅 = 🕮 🗉 = 📂 🛃 🍠 🐰 🛍 🛍 !	₩	Build Solution	F7
i 🐑 🐷 🖓 🍋 🔓 😭 💁 🔬 🗳 🍦 🗯		Rebuild Solution	Ctrl+Alt+F7
Arduino 1.6 🝷 Arduino Mini 🝷 🎖 🚽		Clean Solution	
sensors con X ASE Wizard ATmega22		Build ATmega_328P_1	
Sensols.cpp X Asi Wizard Annegasz		Rebuild ATmega_328P_1	
🔸 sensors.cpp 🛛 👻 🚔 🌳 C:\\		Clean ATmage 229D 1	
		Clean ATmega_328P_1	
		Configuration Manager	
		Compile	Ctrl+F7

RESULT The build should finish successfully with zero errors.

5.4 Hardware Connection

Light sensor and temperature senor are present on IO1 Xplained Pro board.

IO1 Xplained Pro is an extension board to the Atmel Xplained Pro evaluation platform. It is designed to give a wide variety of functionality to Xplained Pro MCU boards, including a microSD card, a temperature sensor, a light sensor and more. The IO1 extension board is shown in Figure 5-2.



Figure 5-2. IO1 Xplained Pro

Actual connections of IO1 Xplained Pro board to ATmega328P Xplained Mini are as shown in Table 5-1.



IO1 Xplained Pro		ATmega328P Xplained Mini		
Pin number	Name	Pin	MCU pin	
3	ADC	PC0	A0	
11	TWI_SDA	PC4	SDA	
12	TWI_SCL	PC5	SCL	
2	GND	GND		
20	VCC	3V3		
15	SPI_SS_A	PB2	SPI_SS(D10)	
16	SPI_MOSI	PB3	SPI_MOSI	
17	SPI_MISO	PB4	SPI_MISO	
18	SPI_SCK	PB5	SPI_SCK	

Table 5-1. Connection: IO1 Xplained Pro- ATmega328P Xplained Mini

5.4.1 Connection: IO1 Xplained Pro – Arduino Xplained Pro

Connect the IO1 Xplained Pro board to Arduino Xplained Pro board as shown in below figure.



5.4.2 Connection: ATmega328P Xplained Mini - Arduino Xplained Pro

Now we need to connect ATmega328P Xplained Mini to the Arduino Xplained Pro board. Connect so that PC5 from column 1 on the Xplained Mini board goes in A5 of the Arduino Xplained Pro (red colored connection) and PC5 from column 11 on the Xplained Mini goes in SCL of to the Arduino Xplained Pro (green colored connection).





Connection would be as below.



5.4.3 Connection: USB cable

Connect the USB cable to the ATmega328P Xplained Mini board and place the board as shown below.

NG Make sure SD card is properly inserted in the socket.







Board is placed in this way that the light sensor wouldn't be in dark.

i info

INFO

f

Alternative way of connection: User can also connect IO1 Xplained Pro board using nine male to female connectors without using Arduino Xplained Pro board as shown in figure below.





5.5 Debugging the Application

Now we will debug the application. Code has been written such that at particular condition (when sensor value > 500) the light sensor value and temperature value will be stored in the SD card. We will place a breakpoint at this condition and check that the breakpoint is hit.



Select Debug and click 'Start Debugging and Break'.



	Windows	
11	Start Debugging and Break	c Alt+F5
1	Stop Debugging	Ctrl+Shift+F5
Þ	Start Without Debugging	Ctrl+Alt+F5
	Disable debugWIRE and C	ose
	Continue	F5
ê	Execute Stimulifile	
Ś	Set Stimulifile	
3	Restart	
63	QuickWatch	Shift+F9
•I	Step Into	F11
ÇI	Step Over	F10
°1	Step Out	Shift+F11
•1	Run To Cursor	Ctrl+F10
1	Reset	Shift+F5
	Percepio Trace	
	Toggle Breakpoint	F9
	New Breakpoint	
2	Delete All Breakpoints	Ctrl+Shift+F9
	Clear All DataTips	
	Export DataTips	
	Import DataTips	
	Options and Settings	

WARNING

If the DWEN fuse is not enabled and error message is displayed. Click 'Yes' and Studio will use the ISP to set the fuse as shown below.



RESULT The Debugger is started and breaks in main. You are now ready to start debugging.

TO DO In Atmel Studio select View \rightarrow Terminal Window. Select the mEDBG COM Port, Baud: 9600 and select "Connect".









🔡 то do

To view the sensor value while debugging, select Debug \rightarrow QuickWatch... Enter the expression 'sensorValue' and select button 'Add Watch'.

RESULT

TO DO

In Watch1 window value of variable 'sensorValue' will be displayed.

ATmega_328P_1 (Debugging) - AtmelStudio File Edit View VAssistX ASF Project Build Debug Tools Window H	elp	
🐜 - 🖽 := 💋 📓 🖉 X 🖏 🖄 🤚 - (* - 💭 - 🖏 😨 .	🖒 🕅 Debug 🕞 💋 spi_ini	t 🚽 🖓 🚰 🛃 🎒
i 🗖 📴 🐼 🍋 🔓 🥵 🕰 🛓 💷 🗉 🍦 🗉 🕨 ຝ 🕾 💷 📜 🛸	🛾 🛨 Hex 🛛 🖬 🗸 🔡 🐼 💷	🔤 🖪 🔡 🛗 🛗 👗 📲 🛲 ATmega32
Arduino 1.6 • Arduino Mini • 💡 = COM3 • 🔜 🖬 📮		
Disassembly CDC.cpp wiring.c main.cpp sensors.cpp × Ardui	no.h ATmega_328P_1	
→ loop.if ↓ if (sensorValue>=500)	QuickWatch	
⊟void loop() {	Expression:	Reevaluate
<pre>sensorValue = analogRead(analogInPin); // rea</pre>	sensorValue	✓
<pre>temp_result = at30tse_read_temperature(); // rea</pre>	Value:	Add Watch
if (sensorValue>=500)	Name	Value Type ^
<pre>\$ sensor_flg=1; }</pre>		
<pre>if (sensor_flg==1) // write data in file on p {</pre>		
sensor_flg=0;		
<pre>myFile = SD.open("data.txt", FILE_WRITE);</pre>		
if (myFile) // if the file opened okay, wr		T
<pre>{ myFile.print("sensor = "); mvFile.print(sensorValue);</pre>		Close Help
Watch 1		
Name		Value
✓ sensorValue		516
TO DO Close the 'Watch1' windo	ow and 'QuickWatch'	window.

Continue the execution by pressing F5 on keyboard.





Sensor value and temperature value will be stored in SD card also transmitted through EDBG COM port. Terminal window will show sensor values and temperature values.

Figure 5-4.	Atmel	Studio:	Terminal	Window

Terminal Window	<u>-</u> □ ×
Disconnect COM27 x Paude 9600 x ASCII x 🖸 🎽 Esua to file Options	
Save to me Options	
Receive	
SD Card initialization done. sensor = 516 temp = 24.00	×
Sand History	
Send	ASCII - LF CR Send
mEDBG Virtual COM Port (COM27)	
TO DO Exit the debug mode by selecting Debug \rightarrow Disable debugWIRE.	

Exit the debug mode by selecting Debug \rightarrow Disable debugWIRE.

WARNING It is important to disable debugWIRE.

We have successfully added Arduino project on an ATmega328P Xplained Mini with the Atmel Studio.



Appendix A. Complete Solution to Assignment 4

Sensors.cpp file.

```
#include "arduino.h"
#include <Wire.h>
#include <SD.h>
#define AT30TSE_TEMPERATURE_TWI_ADDR
                                         0x4F
#define AT30TSE_TEMPERATURE_REG
                                         0x00
#define AT30TSE_TEMPERATURE_REG_SIZE
                                         2
#define AT30TSE_NON_VOLATILE_REG
                                         0x00
#define AT30TSE CONFIG RES 9 bit
                                         0
#define AT30TSE CONFIG RES 10 bit
                                         1
#define AT30TSE_CONFIG_RES_11_bit
                                         2
#define AT30TSE_CONFIG_RES_12_bit
                                         3
uint16_t resolution = AT30TSE_CONFIG_RES_10_bit;
double temp_result;
int sensorValue = 0;
                     // value read from ADC A0
const int analogInPin = A0; // Analog input pin
const int chipSelect = 10;
// set up variable to use SD utility library functions:
File myFile;
void setup()
{
       Serial.begin(9600); // Open serial communications
       if (!SD.begin(chipSelect)) {
              Serial.println("SD Card initialization failed!");
              return;
       }
       Serial.println("SD Card initialization done.");
       SD.remove("data.txt");
       Wire.begin();
}
```



```
uint16_t at30tse_read_register(uint8_t reg, uint8_t reg_type, uint8_t reg_size)
{
       uint8_t buffer[2],i=0;
       buffer[0] = reg | reg_type;
       buffer[1] = 0;
       /* Internal register pointer in AT30TSE */
      Wire.beginTransmission(AT30TSE_TEMPERATURE_TWI_ADDR);
      Wire.write(buffer[0]);
      Wire.endTransmission();
      Wire.requestFrom(AT30TSE TEMPERATURE TWI ADDR, reg size);
      while(Wire.available())
       {
              buffer[i] = Wire.read();
              i++;
       }
       return (buffer[0] << 8) | buffer[1];</pre>
}
double at30tse_read_temperature()
{
       /* Read the 16-bit temperature register. */
      uint16 t data = at30tse_read_register(AT30TSE_TEMPERATURE_REG,
      AT30TSE_NON_VOLATILE_REG,
      AT30TSE TEMPERATURE REG SIZE);
       double temperature = 0;
       int8_t sign = 1;
       /*Check if negative and clear sign bit. */
       if (data & (1 << 15)){
              sign *= -1;
              data &= ~(1 << 15);</pre>
       }
       /* Convert to temperature */
       switch (resolution){
              case AT30TSE_CONFIG_RES_9_bit:
              data = (data >> 7);
              temperature = data * sign * 0.5;
              break;
              case AT30TSE_CONFIG_RES_10_bit:
              data = (data >> 6);
              temperature = data * sign * 0.25;
              break;
              case AT30TSE_CONFIG_RES_11_bit:
              data = (data >> 5);
              temperature = data * sign * 0.125;
              break;
              case AT30TSE_CONFIG_RES_12_bit:
              data = (data >> 4);
              temperature = data * sign * 0.0625;
              break;
              default:
              break;
       }
       return temperature;
}
```



```
bool sensor flg=0;
void loop() {
       sensorValue = analogRead(analogInPin);
                                                              // read the ADC:
      temp_result = at30tse_read_temperature(); // read temperature
      if (sensorValue>=500)
       {
             sensor_flg=1;
       }
      if (sensor_flg==1)
                               // write data in file on particular sensor vale
       {
             sensor_flg=0;
              myFile = SD.open("data.txt", FILE_WRITE);
              if (myFile)
                                  // if the file opened okay, write to it:
               {
                     myFile.print("sensor = " );
                     myFile.print(sensorValue);
                     myFile.print("
                                       temp = ");
                     myFile.print(temp result);
                     myFile.print("\n");
                     myFile.close(); // close the file:
               }
              else
              {
                     // if the file didn't open, print an error:
                     Serial.println("error opening data.txt");
               }
              // re-open the file for reading:
              myFile = SD.open("data.txt");
              if (myFile)
               {
                     // read from the file until there's nothing else in it:
                     while (myFile.available()) {
                            Serial.write(myFile.read());
                     }
                     // close the file:
                     myFile.close();
                     } else {
                     // if the file didn't open, print an error:
                     Serial.println("error opening data.txt");
              }
       }
       delay(500);
}
```



6 Conclusion

In this hands-on training we have accomplished the following tasks:

- added support form Arduino style coding to Atmel studio
- added debugging capability to Arduino project
- based an Arduino project on Atmel Xplained Mini and Atmel Studio without changing any code
- added the ability to write more complex and complicated programs



7 Revision History

Doc. Rev.	Date	Comments
42439A	08/2015	Initial document release



Atmel Enabling Unlimited Possibilities[®]



www.atmel.com

L

 Atmel Corporation
 1600 Technology Drive, San Jose, CA 95110 USA
 T: (+1)(408) 441.0311
 F: (+1)(408) 436.4200

© 2015 Atmel Corporation. / Rev.: Atmel-42439A-From-Maker-to-Manufacture-Bridging-the Gap-from-Arduino-to-AVR_TrainingManual_082015.

Atmel[®], Atmel logo and combinations thereof, AVR[®], Enabling Unlimited Possibilities[®], and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Windows[®] is a registered trademark of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained here in. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.