AVR1934: XMEGA-A3BU Xplained Software User Guide

8-bit Atmel Microcontrollers

Application Note

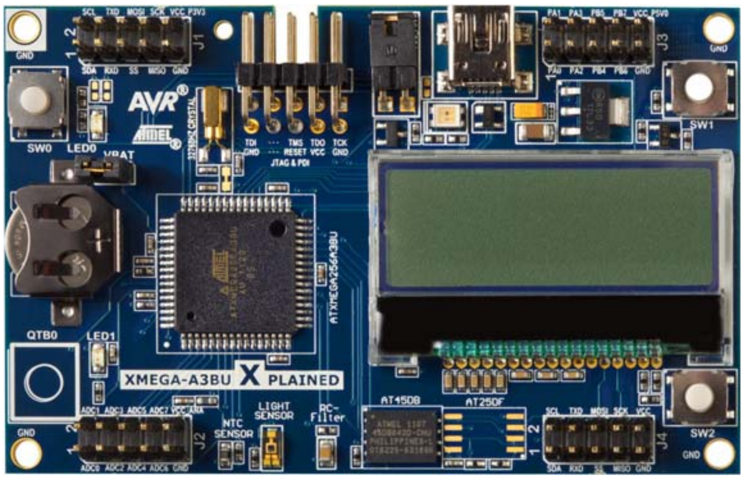
Features

- **LCD with backlight**
- **Sensors readout**
  - **Light sensor**
  - **Temperature sensor**
- **Menu system**
- **QTouch button demonstration**
- **Date and time functionality using Real Time Counter**
  - **Time since production**
  - **Current time and date**

1 Introduction

The Atmel® AVR® XMEGA®-A3BU Xplained evaluation kit demo software is created to showcase the XMEGA-A3BU, touch button and LCD, using a simple menu navigation system, and small applications showcasing different features. For detailed documentation, please refer to the source code documentation generated by the Doxygen automatic documentation tool. Application Note AVR1923 describes the XMEGA-A3BU Xplained hardware in detail.

Figure 1-1. The XMEGA-A3BU Xplained Board



Rev. 8413A-AVR-09/11

2 Modules and services

This demo application is available through the Atmel® AVR® Software Framework (ASF), version 2.6.0 or later. It is available at the Atmel.com AVR Software Framework page. The demo application is available as an example project in AVR Studio 5. This can be accessed by clicking File → New → Example Project, and selecting "Demo application for XMEGA-A3BU Xplained".

Table 2-1. List of services and modules

Module name	**Source file**	**Header file**
LCD panel and controller	`st7565r.c`	`st7565r.h`
QTouch button	-	`touch_api.h`
NTC temperature sensor	`ntc_sensor.c`	`ntc_sensor.h`
Ambient light sensor	`lightsensor.c`	`lightsensor.h`
Battery backed Real Time Counter	`rtc32.c`	`rtc32.h`
Monochrome Graphics System (gfx_mono)	`gfx_mono.c`	`gfx_mono.h`
Menu system	`menu.c`	`menu.h`
Calendar service	`calendar.c`	`calendar.h`
USB CDC service	`cdc.c`	`cdc.h`

2.1 LCD panel and controller

The XMEGA-A3BU Xplained board features a monochrome LCD (liquid crystal display), which is a bundle containing an LCD controller (ST7565R) and an LCD panel (C12832_A1Z), with white backlight.

The LCD controller offers two different interfacing modes:

1. Parallel interface mode in which display data is written by the master MCU (XMEGA for this board) to the controller memory, eight bits at a time using eight data lines for the transfer.
2. Serial interface mode where the master MCU produces a clock and a serial data signal, generally known as SPI (Serial Peripheral Interface).

The latter mode is selected for this board. The advantage of using the serial interface is that the number of pins is kept to a minimum, while the main disadvantage is that one can only write to the display, as reading of display data is only supported when using the parallel interface.

Five pins are needed to connect the master MCU to the LCD controller in serial interface mode:

- Chip select, used to tell the LCD controller that it is the intended target for the serial transmission
- Register select, used to signal whether the transmission contains data (HIGH), or a command (LOW)
- Reset (active LOW), used to perform a hard reset of the LCD controller and the default SPI data and clock pins

- SPI clock
- SPI data

The ST7565R is always initialized by performing a hard reset of the LCD controller to bring it to a known state, as its state cannot be read from the controller. Please refer to the "XMEGA-A3BU Xplained Hardware User's Guide" for more information about physical pin connections.

The ST7565R driver (`st7565r.c` and `st7565r.h`) takes care of all low level interfacing with the reset line, register select line, chip select line and the SPI driver. The ST7565R driver offers a number of functions used by higher level display stacks, such as the `gfx_mono` service. The driver may also be used directly, if needed. These functions are described and thoroughly documented in the ASF documentation.

A white LCD backlight is also available for the screen. The backlight can simply be either on or off, or controlled by a PWM signal, which in turn controls the intensity of the backlight.

2.2 QTouch button

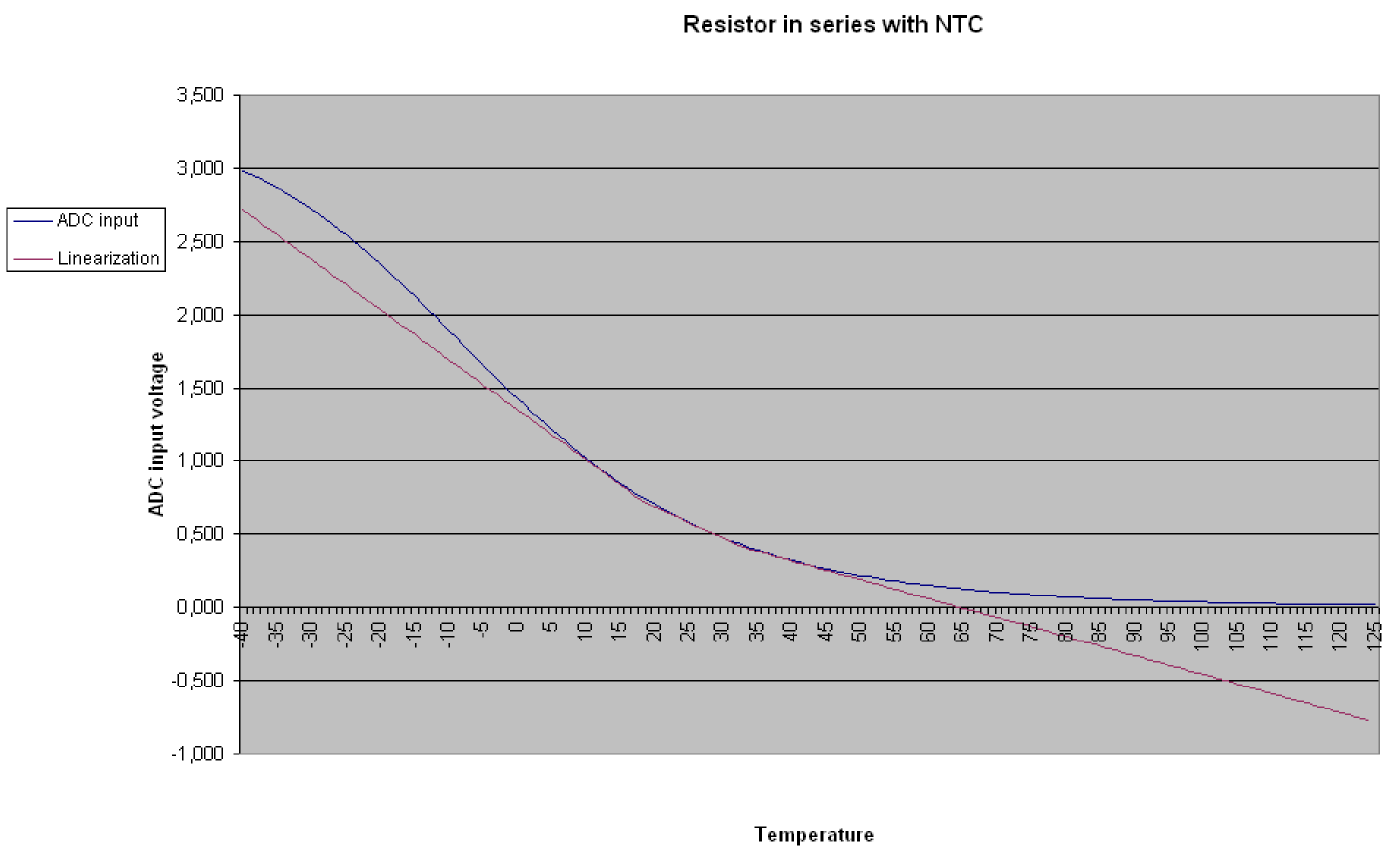
The Atmel AVR XMEGA-A3BU Xplained board contains one Atmel®QTouch® button named QTB0. This button is utilized by the inclusion of the QTouch library. Please refer to the Atmel QTouch Library User Guide for details on this library.

From the application the QTouch button functionality is initialized from main with the call to `touch_init()`, and the button is checked with the function `check_touch_key_pressed()`. It is polled together with the other buttons in the file `keyboard.c`, and will provide menu navigation to the application. The QTouch API is available under the directory `qtouch` as the file `touch_api.h` together with other files from the QTouch library. Configuration of the QTouch library is done with defines declared in `touch_qt_config.h`.

2.3 NTC temperature sensor

An NTC (Negative Temperature Coefficient) resistor is available for measurement of the ambient temperature. The NTC resistor is connected in series with another resistor, creating a voltage across the NTC resistor which varies with the ambient temperature. This voltage can in turn be measured using the Atmel AVR XMEGA ADC. The relationship between the ADC sample values and the actual ambient temperature is a complex equation shown in the plot in Figure 2-1:

Figure 2-1. Comparison of real data with linearized approximation



To calculate the actual temperature efficiently using the XMEGA device, linearization of the relationship between ADC samples and temperature is chosen, using the following equations:

```
T[°C] = -0.07776 * ADCsample + 65.513     [0   < ADCsample <  420]
T[°C] = -0.04748 * ADCsample + 53.25728   [420 ≤ ADCsample <  697]
T[°C] = -0.0295  * ADCsample + 40.50229   [697 ≤ ADCsample ≤ 2047],
```

where `ADCsample` is the value obtained from the ADC. The accuracy of the resulting temperature is only as accurate as the regression method used. As the equation which has been linearized is fairly complex, the resulting temperature should only be regarded as an approximation to the actual temperature.

2.4 Ambient light sensor

An ambient light sensor in available on the Atmel® AVR® XMEGA®-A3BU Xplained, which is sensitive to visible light, much like the human eye. A resistor is connected in series with the light sensor providing a voltage proportional to the current passing through the light sensor. An increasing current gives an increasing voltage across the series resistor, and this voltage is measured using the XMEGA ADC. Table 2-2 shows the relationship between the ambient light in lux and the voltage measured. The table also provides some indication of what the different lux values represent in daily life.

Table 2-2. Illuminance and ADC input voltage

Illuminance [lux]	**ADC input [V]**	**Illuminance scenario**[1]
1	0,0024	Full moon overhead at tropical latitudes
10	0,0243	
20	0,0485	
30	0,0728	
40	0,0971	
50	0,1213	Family living room
60	0,1456	
70	0,1699	
80	0,1941	Hallway/toilet
100	0,2426	Very dark overcast day
200	0,4853	
300	0,7279	Office lighting
400	0,9706	Office lighting
500	1,2132	Office lighting
600	1,4559	
700	1,6985	
800	1,9412	
850	2,0625	Overcast day

[1] Data obtained from Wikipedia

2.5 Battery backed Real Time Counter

The Atmel AVR XMEGA-A3BU features battery backup system support, which can be used to power an ultra low-power 32-bit Real Time Counter (RTC), and a 32.768kHz crystal oscillator, providing a 1Hz clock to the RTC. On the XMEGA A3BU Xplained board the RTC is powered by a button cell battery.

The battery is connected to the battery backup systems V_{BAT} pin, ensuring power to the RTC when the main power is disconnected, allowing the storage of current time on the board. Typical power consumption with the RTC running is 500nA.

During production of the XMEGA-A3BU Xplained board, the production time is stored in the user signature row, and the RTC is initialized to the current time in UTC (Coordinated Universal Time), allowing the demo application to show the time relative to the time zone, and time elapsed since production.

2.5.1 Usage in demo application

The data from the RTC is used in the calculation of current date and time in the Date/Time and Production date applications. This is done by interfacing the XMEGA RTC32 driver (`rtc32.c` and `rtc32.h`), and reading the counter register.

The main demo application verifies that the RTC is not running on battery power before trying to initialize it. If the V_{BAT} system check fails, the RTC is not running, and must be reinitialized. The applications will also try to verify that the RTC has not been reset by comparing it to the production time stored in the user signature row. If by chance the RTC register is cleared, such as when the backup battery is removed, the current time will be reset to whichever is later, one second after the production time stored in the user signature row, or 01.01.2011 00:00:00 UTC, allowing for correct operation of the demo application.

2.6 Monochrome Graphics System (gfx_mono)

The monochrome graphics library gfx_mono provides functions for a high level interface for drawing primitives, pixel manipulation and writing text to LCD panels. In this configuration, it is set up to interface the ST7565R LCD controller, which in turn interfaces the C12832_A1Z LCD panel. It is created so that it provides a complete abstraction layer on top of the LCD controller interface, allowing for easy porting to other LCD controllers and panels. All ST7565R and C12832_A1Z specific functions are located in `gfx_mono_c12832_a1z.c` and `gfx_mono_c12832_a1z.h`.

The library allows for drawing and filling of primitives (lines, rectangles, circles) using `gfx_mono_generic.h`, and text printing using `gfx_mono_text.c` and `gfx_mono_text.h`. The former also allows drawing of bitmaps to the graphic memory.

The library supports writing to a frame buffer for easy manipulation of pixel data, in case the LCD does not support pixel data reads. As the LCD controller serial interface chosen for this board does not allow for reading of pixel data, a frame buffer is utilized.

2.6.1 Primitives drawing

The library allows drawing of certain primitives, such as lines, rectangles and circles, defined in gfx_mono_generic.h. These functions are used in the demo applications menu system, and button splash screen. The drawing of primitives is handled by the library, but there is also support for hardware based primitives, given support for this in the controller. Primitives are handled by the library on this board.

2.6.2 Text drawing with font support

Font support is made available through the gfx_mono_text.c and gfx_mono_text.h files, which in turn relies on system font objects from sysfont.c and sysfont.h, allowing printing of text from glyph data stored in program memory using progmem.c and progmem.h.

2.6.3 Usage in demo application

The demo application uses the menu system service, which in turn uses gfx_mono, combining primitive drawing and text output to create a simple navigation menu system. See the menu system chapter for more information. Each individual application uses the library for different purposes, such as text, primitives and bitmaps.

2.7 Menu system

The module `common.services.gfx_mono.menu` provides a simple menu system for monochrome graphical displays.

Typical flow of an application using the menu system:

1. Define menu structure.
2. Call `gfx_mono_menu_init()`.
3. Get user input.
4. Update menu with user input using function `gfx_mono_menu_process_key()`.
5. Interpret `gfx_mono_menu_process_key()` return value.
6. Go to 3.

2.7.1 Menu `struct`

The menu is declared using the 'menu' struct defined in `gfx_mono_menu.h`. It consists of the following members:

- `title:` the title of the menu, shown on the top line of the display
- `strings:` an array of strings, each string defines one line in the menu
- `num_elements:` number of elements (strings) in the menu

In addition you have two members for internal use, which should be initialized to 0:

- `current_selection:` the line currently selected
- `current_page:` the active page

2.7.2 Initialize menu system

To initialize the menu system, call the `gfx_mono_menu_init()` function. Use a reference to the menu `struct` as parameter. This function will clear the display and draw the menu.

2.7.3 Update menu

Before the menu can be updated, input from the user is needed. Methods for getting input are not part of the menu module, and must be implemented separately.

As soon as input is available, the menu system is informed using the `gfx_mono_menu_process_key()` function. Use a reference to your menu struct and the user input key code as parameters.

This function will then return a status code and act depending on the given keycode:

- `MENU_KEYCODE_DOWN`: Change selection to next menu item (or first if at bottom) Returns `MENU_EVENT_IDLE`
- `MENU_KEYCODE_UP`: Change selection to previous menu item (or last if at top) Returns `MENU_EVENT_IDLE`
- `MENU_KEYCODE_ENTER`: Nothing changes in menu. Returns the line selected
- `MENU_KEYCODE_BACK`: Nothing changes in menu. Returns `MENU_EVENT_EXIT`

2.7.4 Custom keycodes

Keycodes are values describing different keys, enabling the system to differentiate which key has been pressed. These definitions are located in `conf_menu.h`, and may be changed as needed.

2.7.5 Custom indicator icon

The graphical indicator used to indicate menu selection is defined in `conf_menu.h`. This indicator can also be changed if needed.

2.7.6 Usage in demo application

The main menu consists of the following items:

- Temperature
- Light sensor
- Production Date
- Date and Time
- Toggle Backlight

2.8 Calendar service

Including `calendar.c` and `calendar.h` provides functionality for converting between UNIX timestamps and `calendar_date` structs.

A UNIX timestamp is a 32 bit integer value representing the number of seconds that has elapsed since January 1st 1970, 00:00:00 UTC.

The `calendar_date` struct consists of seconds, minutes, hours, date, month, year and day of week. The months and dates start at zero, so one has to increment the variable to get the normal readable months and dates. Days of the week are numbered from 0 to 6, where 0 is Sunday and 6 is Saturday.

The service also provides functionality for calculating the time difference between two `calendar_date` structs, as well as functionality for converting timestamps to dates in a specified time zone and dates in a specified time zone to timestamps. The time zones are provided as input parameters to the time zone functions as offset in hours (positive and negative) and minutes. Calendar functionality together with the RTC could in other applications be used to set alarm for a specific time and date.

Other functionality includes incrementing seconds in the `calendar_date` struct, without having to go through a UNIX timestamp.

2.8.1 Usage in demo application

The production time application uses the calendar service to convert the production time timestamp and the current timestamp to a date and calculate the difference between them.

The date and time application uses the service to convert the current timestamp into the local time and date.

2.9 USB CDC service

The demo application allows the XMEGA-A3BU Xplained board to communicate with a computer using USB with a CDC (Communications Device Class) setup, allowing the keyboard to be used to navigate in the demo application. This functionality is available in the files `cdc.c` and `cdc.h`. The application calls `cdc_start()`, which reads in the USB serial number from user signature row, which is later passed to the USB stack. The configuration of the USB CDC device is done in `conf_usb.h`. The keyboard interface available in `keyboard.c` and `keyboard.h` uses `cdc_getkey()` to read a key from the CDC interface, and translates it to a usable keycode.

2.9.1 Usage in demo application

The USB CDC service allows the demo application to be controlled by a computer keyboard. When the XMEGA-A3BU Xplained board is connected for the first time, Windows will try to install a driver for "Virtual CDC Com". A driver file, `XPLAINED_Virtual_Com_Port.inf`, is located in the demo application directory. If the driver is successfully installed, the board will show up in in the Device Manager, under Ports (`XPLAINED Virtual Com Port (COM**)`). This COM port can now be interfaced with your favourite terminal software.[2] The connection parameters are as follows:

Baud rate	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

If the connection has been successfully established, you should be greeted by this text on the screen of your terminal:

```
Welcome to the XMEGA-A3BU Xplained Demo CDC interface!


Key bindings for LCD menu control:
  Enter      : Enter
  Backspace  : Back
  Arrow Up   : Up
  Arrow Down : Down
```

You should now be able to control the demo application with your keyboard.

[2] If you are using Windows XP, you can use the built in HyperTerminal (Run dialog → hypertrm). If you are using Windows 7, you can download PuTTY, or RealTerm.

3 Demo application

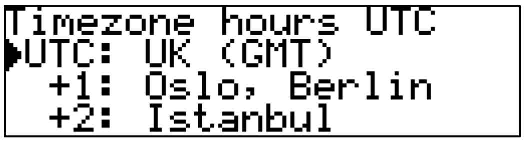
The demo application consists of different applications selectable from a list when the device is started:

Table 3-1. Available applications

Application name	Source file	Header file
Time zone application	`timezone.c`	`timezone.h`
Date time application	`date_time.c`	`date_time.h`
Production date application	`production_date.c`	`production_date.h`
Temperature application	`ntc_sensor.c`	`ntc_sensor.h`
Light sensor application	`lightsensor.c`	`lightsensor.h`
Backlight toggling	--	--

3.1 Time zone application

When the XMEGA-A3BU Xplained board is powered on for the first time, the user is prompted to supply it with the current time zone from a list of hours from UTC -12 to UTC +12, as well as minute offsets. The new time zone settings are stored in EEPROM. As the UNIX timestamps generated by the RTC is in Coordinated Universal Time (UTC), this info will be used to display the time as local time. The time zone application is available in the date and time application available from the main menu, and is located in `timezone.c` and `timezone.h` together with other time zone functionality. When the time zone application is started, it is not possible to return to the main menu before a time zone is selected.



```
Timezone hours UTC
▶UTC: UK (GMT)
  +1: Oslo, Berlin
  +2: Istanbul
```

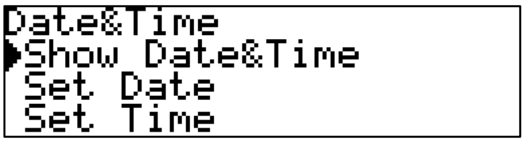
Figure 3 Timezone selection

3.2 Date and time application

The date time application presents the user with a menu with the following options:

- "Show date&time" – display local date and time in the selected time zone
- "Set date" – Set the date
- "Set time" - Set local time
- "Set timezone" – Set time zone hour and minute offset

The menu uses the menu system, and all strings and options are defined at the top of `date_time.c`. This function handles calling the correct application based on the user's choice in the menu. If the back button is pressed, the user will return to the main menu.



```
Date&Time
▶Show Date&Time
 Set Date
 Set Time
```

Figure 4 Date&time menu

The "Show date&time" application shows the current local date and time, computed from the RTC UNIX timestamp. Note that this time will not be completely accurate, due to inaccuracies in the RTC.

"Set date" and "set time" presents the user with the current date or time and spinner widgets to adjust them. The date widget does not do real-time checking of valid dates, so if the user tries to set an invalid date such as Feb. 31th; an error message will be presented, and the date will not be stored.

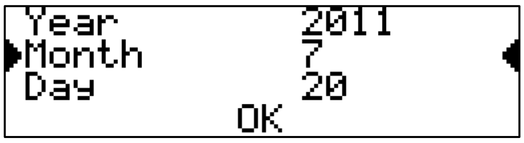

Year 2011
Month 7
Day 20
OK

Figure 5 Set date, month spinner selected

The "Set timezone" menu option is the same application that is presented to the user on initial startup, and will present the user with a list of time zones from UTC -12 to +12 hours and UTC +0, +15, +30 or +45 minutes. The selector uses the menu system and a list of strings to present the user with the available options. The selected time zone is stored in EEPROM.

The date and time menu, along with the associated applications are located in `date_time.c` and `date_time.h`. The time zone selector is located in `timezone.c` and `timezone.h`.

It is important to note that this application does not account for Daylight Saving Time (DST).

3.3 Production date application

The production date application uses the current time from the RTC and the production time stored in user signature row to calculate and show on the LCD display how much time in months, days and hours has elapsed since production. If production time in the signature row is corrupted, it will be assumed to be 01.01.1970, 00:00:00. The production date application is available from the main menu. It is located in `production_date.c` and `production_date.h`.

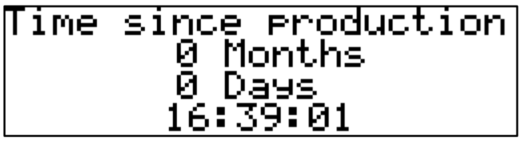

Time since production
0 Months
0 Days
16:39:01

Figure 6 "Time since production" application screenshot

3.4 Temperature application

The temperature application acquires the temperature using the functions available from `adc_sensors.c` and `adc_sensors.h`, and then displays this data on the screen, together with a thermometer picture which has a scale depending on the measured data. The temperature application is available from the main menu. It is located in the files `ntc_sensor.c` and `ntc_sensor.h`.

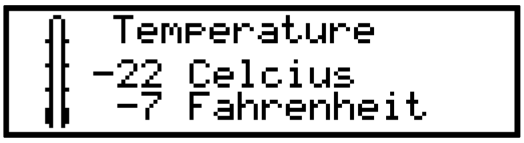
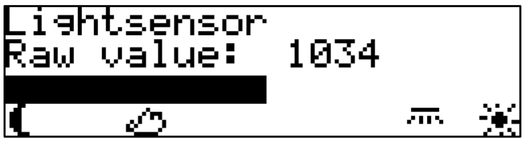

Temperature
-22 Celcius
-7 Fahrenheit

Figure 7 A cold winter day

3.5 Light sensor application

The light sensor application displays data from the light sensor as a raw ADC value text string, and a graphical bar representing illuminance. This bar increases in length as the ambient illumination increases. In contrast, the raw ADC value will decrease with increasing illumination. To avoid flickering, the text string for the raw value is only updated every 20th iteration of the application loop. The light sensor application is available from the main menu. It is located in the files `lightsensor.c` and `lightsensor.h`.

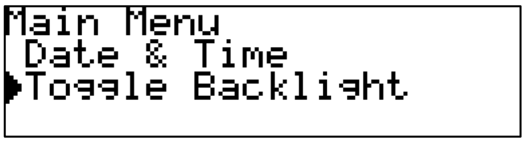


```
Lightsensor
Raw value:   1034
```

Figure 8 Lightsensor application screenshot

3.6 Backlight toggling

The backlight can be toggled on or off from the demo application menu.



```
Main Menu
 Date & Time
▶Toggle Backlight
```

Figure 9 Toggle backlight in the main menu

4 EVALUATION BOARD/KIT IMPORTANT NOTICE

5 Table of Contents

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chou-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3523-3551
Fax: (+81) 3523-7581