Features

- Joystick Game Controller Atmel® ATxmega32A4U microcontroller
- In System Programming (ISP) and debugging via Program and Debug Interface (PDI)
- Analog inputs
  - Two analog joysticks w/ two axis movement
- Digital I/Os
  - Six digital push buttons

Introduction

This document shows the implementation of a joystick game controller based on the Atmel ATxmega32A4U microcontroller. This design allows users to control the mouse cursor and simulate left/right clicks from a personal computer (PC) or laptop via push buttons on the joystick of a game controller.
Table of Contents

1. Glossary .................................................................................................................. 3
2. Overview .................................................................................................................. 3
3. ATxmega Base Board .............................................................................................. 3
4. Joystick Board ........................................................................................................... 4
5. Software Implementation .......................................................................................... 5
   5.1 Joystick program flowchart .................................................................................. 5
   5.2 System initialization .............................................................................................. 5
   5.3 ADC initialization ............................................................................................... 6
   5.4 ADC sampling / start conversion ....................................................................... 7
6. User Interface (UI) ................................................................................................... 8
   6.1 HID Descriptor file ............................................................................................. 8
   6.2 Mouse operations .............................................................................................. 8
   6.3 Defines and prototypes ..................................................................................... 10
7. Hardware Design ..................................................................................................... 11
   7.1 Atmel ATxmega base board .............................................................................. 11
   7.2 Joystick daughter board .................................................................................. 12
   7.3 Base board and daughter board setup ............................................................ 13
8. References ............................................................................................................... 14
9. Revision History ....................................................................................................... 15
1. Glossary

ISP  In-System Programming
Atmel Studio 6  Integrated Development Environment (IDE) for Atmel AVR® applications
USB  Universal Serial Bus
HID  Human Interface Device

2. Overview

This application requires the user to provide analog signals to the Atmel ATxmega32A4U in which passes the obtained values to coordinates to the HID device which in this case is a standard PC mouse. The hardware design will be split between three separate boards:

- ATxmega Base Board – Assembled with an Atmel ATxmega device which has the USB connector, 3.3V regulator, and appropriate headers pins
- Joystick Daughter Board – This board was meant to mount on top of the ATxmega baseboard. This board has two analog joysticks that are equipped with pushbutton switches. In addition, there are four tactile pushbuttons connected to the standard I/O of the ATMEGA device

3. ATxmega Base Board

The base board block diagram is shown in Figure 3-1. The board itself consists of the following components:

- ATxmega324AU device – Handles user input from either joystick, pushbutton, touch pad
- Micro USB connector – Provides supply (5V) to the game controller board. In addition, provides the connection medium between the game controller board and the PC
- Voltage regulator – The ATmega32A4U device operates at a VCC range between 1.6V to 3.6V. This regulator converts the 5V provided by the USB to a 3.3V for the ATmega32A4U device
- ISP/PDI header – ISP programming interface to update the firmware on the device
- Six 2x5 header – These headers pins are for the I/O from the ATxmega device to communicate with the other daughter boards. In addition, these provide an anchor when both boards are connected to one another

Figure 3-1. ATxmega base board, block diagram.
4. **Joystick Board**

The joystick daughter board block diagram is shown on Figure 4-1 and consists of the following components:

- Two analog joysticks
  - Directional movements are simply two potentiometers (10kΩ) – one for each axis
  - Joystick has a select button that is actuated when the joystick is pressed down
- Four tactile pushbuttons
  - These pushbuttons are directly connected to the headers of the board which are connected to ports E and D of the Atmel ATxmega device

![Figure 4-1. Joystick board, block diagram.](image-url)
5. **Software Implementation**

5.1 **Joystick program flowchart**

![Joystick program flowchart](image)

5.2 **System initialization**

The Atmel ATxmega32A4U has a large number of clock sources to choose from. For the fastest ADC conversion time while saving space on our reference board, the code will initialize and calibrate the onboard 32Mhz internal RC oscillator. The internal oscillators do not require any external components to run. For more details concerning the characteristics and accuracy of the internal oscillators, refer to the devices’ datasheet.

The GPIO ports of the ATxmega32A4U need to be configured properly in order to handle the incoming signals from the joystick and pushbuttons. Each port pin on the microcontroller can be configured as an input or output with a selectable driver and pull settings. For this application, we will have a total of four analog signals and six digital signals from the two joysticks and four tactile pushbuttons. For the tactile push buttons, the ATxmega32A4U I/O pins will be set as inputs with internal pull-up enabled.
5.3 ADC initialization

This reference design will utilize the on-board ADC of the Atmel ATxmega32A4 to handle the analog signals that are generated by the two joysticks. Each joystick contains two variable resistors (X- and Y-position) that will increase/decrease in value in comparison of the joystick’s position. The ADC will be configured in the following mode to ensure that the signals from the joysticks are interpreted correctly.

The ADC will be configured as follows:

- Single Ended mode
- 12-bit resolution, unsigned results
- Port A input channels
- External \( V_{\text{REF}} \) @ 3.3V
- Manual conversion triggering

In `main.c` the full ADC initialization can be found in under the `adc_hw_init` function. Each input channel used must be initialized properly.

The code has defined the input port channels as follows:

```c
// Analog Input Port Channels
#define JOYSTICK_INPUT_0   ADCCH_POS_PIN1 //PA1
#define JOYSTICK_INPUT_1   ADCCH_POS_PIN2 //PA2
#define JOYSTICK_INPUT_2   ADCCH_POS_PIN3 //PA3
#define JOYSTICK_INPUT_3   ADCCH_POS_PIN4 //PA4
```

In addition to properly setting up the ADC to receive the analog values, the ATxmega32AU must have a defined threshold of values that can ensure that the joystick movement is correctly represented. In addition, by applying a threshold we can give users a faster cursor movement when the joystick is being fully accelerated in one direction. By applying \( V_{\text{CC}} \) and GND to the joystick the voltage levels and appropriate converted values can be seen in Figure 5-2.

**Figure 5-2.** Joystick threshold values.
5.4 ADC sampling / start conversion

Once the ADC has been properly been configured, joystick movement can be then be detected. The ADC will setup as free-running to provide constant updates to the user with real time position of the mouse cursor.

In Figure 5-2, the ADC thresholds are given in each position axis along with the threshold of when the joystick is not being used at all (REST). The ADC will constantly sample the values on the input ports and will provide real time movement updates to the mouse cursor when the joystick is moved in any direction.

The push button implementation is pretty straight forward. Each push button is connected to an I/O port to the Atmel ATxmega32A4U. When the push button is activated, the I/O pin will be pulled low signaling an active button press. Two of the buttons that are located on the joystick will simulate the standard left and right click of a mouse.

The ADC values and GPIO detection will be passed to UI (User Interface Block) that will handle the standard mouse interface. This block will be discussed in detailed later in the document.
6. User Interface (UI)

6.1 HID Descriptor file

Once system initialization has been completed, the user can then initiate the game controller and begin cursor simulating cursor movements with the joystick/touchpad. The following source files will be discussed in detail concerning the implementation of the HID (Human Interface Device) class:

- udi_hid_mouse.c
- udi_hid_mouse_desc.c
- ui.c

When a HID device is connected to the PC, the device (mouse) sends data is predefined blocks or reports at regular intervals back to the computer. These reports contain critical information about which buttons are pressed or how far the joysticks have moved. However, in order for the host (PC) to understand the data format, the device (Atmel ATxmega) must “describe” the reports to the host which we will call a “Descriptor File.” In the udi_hid_mouse.c the descriptor file is created and initialized in Figure 6-1.

Figure 6-1. Descriptor files implementation.

This standard descriptor file is sent recurring as long as the device is connected to the PC. This is the framework of data that the mouse will send that will correspond to its movements, buttons, and acceleration.

6.2 Mouse operations

A standard mouse has three main outputs: position, left mouse button, right mouse button. The file ui.c will describe how these outputs are defined and implemented.

For the position of the cursor, standard Cartesian coordinates will be used. The file ui.c contains the function used to correctly represent the ADC value to the cursor position. Figure 6-2 shows the full function declaration.
Figure 6-2. Cursor position definition.

The function takes two inputs and returns true when the new position has been sent. The first input, pos, would be the actual position which has been obtained through the ADC sampling. The second input, index_report would be to determine which axis movement will be updated. A value of “1” would refer to movement along the X-axis while a value of “2” refers movement along the Y-axis. Once the position and axis has been set, the updated coordinates will be sent in the descriptor format and to the host (PC).

As previously mentioned, there the device has set thresholds that will detect when a “full move” is being issued or if a “fine move” is wanted. After seeing what ADC values have been related to the joystick position, ui.c will constantly be checking whether a “full move” is being generated. The following Figure 6-3 shows the threshold check and corresponding movement of the left analog joystick (PA4).

Figure 6-3. Threshold check.

In ui.c, we have certain define thresholds that must be taken in account. Initially, when the joystick is not being moved/pushed, the ADC is still providing a corresponding value to this “center” position. The value of joystick_lower_value is a set middle range of the “center position.” Any values between the joystick_lower_value and joystick_higher_value will not affect movement of the cursor. When a value which is lower than the lower middle value, a position changed is wanted by the user. The threshold value of joystick_min_move determines if the user would like a “full move” which will call the hid movement function with the position of the newly obtained value. However, if the lower threshold is not satisfied, the “fine move” is being requested and a different position is calculated before passing it hid movement function. This implementation is for all axis movement is quite similar. The only values that need to be taken account would be the threshold values and how they relate to which axis (X or Y).

Similarly, the buttons implementation operates in comparable fashion. Standard mouses come equipped with both a left and right button click functions. In ui.c the button function is defined in Figure 6-4.
Figure 6-4. Mouse button implementation.

```c
static bool udi_hid_mouse_btn(bool b_state, uint8_t btn)
{
    // Modify buttons report
    if (HID_MOUSE_BTN_DOWN == b_state)
        udi_hid_mouse_report[0] |= btn;
    else
        udi_hid_mouse_report[0] &= ~btn;
    // Use mouse move routine
    return udi_hid_mouse_move(0, 1);
}
```

For this function, there are two inputs: `b_state` and `btn`. The first input will check whether the mouse button is actually being pressed. The second input, will determine which button is being enabled (0x01 = Left Button, 0x02 = Right Button). For this project we have assigned PE3 and PE4 as the left and right buttons, respectively. However, this can be reassigned to whichever GPIO the user may like.

Today, computer mouses have a few more buttons/features that can be used differently in an application. Functions such as scrolling can implemented with this descriptor file already in place. The function for a mouse scroll feature can be found in the source file `udi_hid_mouse.c`. Similarly, the scroll function does call the same mouse move described in Figure 6-3.

6.3 Defines and prototypes

Table 6-1 shows the critical function prototypes that are used with the game controller application. Table 6-2 shows the threshold values.

Note: The following threshold will depend on each system and should be adjusted accordingly.

**Table 6-1. Function definitions.**

<table>
<thead>
<tr>
<th>Prototypes</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>udi_hid_mouse_move(int8_t pos, uint8_t index_report)</td>
<td>Cursor position update</td>
</tr>
<tr>
<td>udi_hid_mouse_btn(bool b_state, uint8_t btn)</td>
<td>Mouse button detection</td>
</tr>
<tr>
<td>sysclk_init(void)</td>
<td>CLK initialization</td>
</tr>
<tr>
<td>adc_hw_init(int ch, int pin)</td>
<td>ADC initialization</td>
</tr>
<tr>
<td>board_init(void)</td>
<td>GPIO initialization</td>
</tr>
</tbody>
</table>

**Table 6-2. Threshold values.**

<table>
<thead>
<tr>
<th>Parameter definitions</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define JOYSTICK_MIDDLE_VALUE 2900</td>
<td>Threshold for REST Position</td>
</tr>
<tr>
<td>#define JOYSTICK_LOWER_VALUE 2450</td>
<td>Threshold for REST Position</td>
</tr>
<tr>
<td>#define JOYSTICK_MAX_MOVE 4000</td>
<td>Threshold for Full Move</td>
</tr>
<tr>
<td>#define JOYSTICK_MIN_MOVE 290</td>
<td>Threshold for Full Move</td>
</tr>
</tbody>
</table>
7. Hardware Design

7.1 Atmel ATxmega base board

Figure 7-1 shows the ATxmega base board.

Figure 7-1. ATxmega base board.

- Micro USB (U2 / RED)
  This is the connection medium between the host (PC) and the Atmel ATxmega32A4 device. In addition, this provides the Vcc supply voltage to the base board at 5V

- Voltage Regulator (U3 / YELLOW)
  This regulator is needed to step down voltage provided by the USB connector for the ATxmega since the operating voltage of the device is 1.8V to 3.3V. The input voltage of 5.0V will produce the desired 3.3V from the output of the regulator

- Power LED (D1 / GREEN)
  Power LED to ensure Vcc to the board is properly provided

- PDI / ISP (U4 / PURPLE)
  Programming header for the ATxmega32A to update the firmware of the device. A programming tool, such as an Atmel JTAGICE3 can be used to program the IC while in-system

- I/O Headers (J1 – J6 / ORANGE)
  These board headers serve two purposes: the first one be to anchor daughters board to the baseboard to add stability when they are conjoined, the second being to properly route the mix signal outputs from the joystick to the ATxmega32A4
7.2 Joystick daughter board

Figure 7-2 shows the joystick daughter board.

Figure 7-2. Joystick daughter board.

- Analog Joystick (Joy_Left and Joy_Right / RED)
  Two axis joysticks with push button switches
- Push Buttons (SW0 – SW3 / YELLOW)
  Tactile push buttons switches
- I/O Header (J1 – J6 / GREEN)
  Similar to the base board. These headers provide the routing of mix signals to the base board and device (ATxmega32A4) and provide stability when boards are connected to one another
7.3 Base board and daughter board setup

Figure 7-3. Base board and daughter board.

In Figure 7-3, the base board and daughter board are shown as a single unit.
8. References

## 9. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42059A</td>
<td>01/2013</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>