Portable Programmer

Team: *Abhishek Srivastava, Kapil Dev Advani, Sakaar Khurana and Vatsal Sharan*

Mentor : *Pragyandesh Narayan Tripathi*

Under the guidance of Ganesh Pitchiah, Rajat Arora, Rishabh Maheshwari and other members of Electronics Club, IITK.

Abstract:

To design a console having the ability to deploy different programs on diverse embedded systems in a field.A functional model with a graphical lcd,keyboard and mouse input was developed to test multiple problems on top of it.

Hardware Configurations:

The final project consisted of two modules- the programming module and the application module.

Programming module- The programming module consists of an Atmega 32, a 16*2 Character LCD and SD Card interface through fabricated SD Card adapter. Micro SD cards can be interfaced through a micro SD adapter. UART interface is used for communication with microcontroller to be programmed. The programming module can program any Atmega loaded with the bootloader.

Application module:

The application module has PS/2 interface for developing interactive, GUI based applications. A 128*64 Monochrome Graphical LCD is used for display. PS/2 ports are provided for interface with PS/2 mouse and PS/2 Keyboard. The central unit is an Atmega 32 which is programmed through the programming module through UART.

The PS/2 interface:

The application module uses input from PS/2 Mouse and PS/2 Keyboard to run interactive applications.

The PS/2 mouse and keyboard implement a bidirectional synchronous serial protocol. The bus is "idle" when both lines are high . This is the state where the keyboard/mouse is allowed begin transmitting data.

The device always generates the clock signal. If the host wants to send data, it first pulls the Clock low. The host then pulls Data low and releases Clock. This is the "Request-to-Send" state and signals the device to start generating clock pulses. The device then generates the clock and receives the data.

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11-12 bits. These bits are:

- 1 start bit. This is always 0.
- 8 data bits, least significant bit first.
- 1 parity bit (odd parity). The parity bit is set such that the number of 1's in the data bits plus the parity bit always add up to an odd number (odd parity).
- 1 stop bit. This is always 1.
- 1 acknowledge bit (host-to-device communication only)

If the keyboard's processor finds that any key is being pressed, released, or held down, the keyboard will send a packet of information, unique to the key, known as a "scan code" to the host. There are two different types of scan codes: "make codes" and "break codes". A make code is sent when a key is pressed or held down. A break code is sent when a key is released.

There are two common modes in which a PS/2 Mouse is operated:

- Stream - The mouse issues movement data packets when movement occurs or button state changes.
- Remote – The mouse issues data packets only when host sends the command.

We used the remote mode for our project. The data packet consists of 3 bytes:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X movement							
Byte 3	Y movement							

Byte 2 and Byte 3 are the changes in movement or relative movement since last data packet is issued.

SD Card:

SD Card functionality was added to allow the user to store multiple hex files in one place and then program an Atmega with the hex file of their choice. FAT32 was implemented so that the hex files could be used directly after copying them into the SD card. We used FAT32 and SD card libraries the Internet (visit the References section), which had been used before by the Club.

A problem faced was interfacing the SD card, a 3.3 V device, with our Atmega 32, a 5.0 V device. We could not provide the proper current and voltage values for our SD card and had to use a fabricated SD card adapter to interface with the Atmega 32.

Software Configurations:

Bootloader:

To program our application module we used a bootloader. A bootloader is not very different from a normal program, except it has the ability to write into the flash memory. The bootloader has to be programmed in the program memory of the microcontroller just once, using a conventional programmer. Once in the microcontroller, the bootloader is such programmed that each time after reset it starts running like any conventional program. But a UART bootloader will receive data from UART, checking for incoming bytes. If the bytes start arriving, the bootloader will write them in the program. Once all bytes have been received, the bootloader executes a jump at the start of the memory zone it has received and then the “normal” program starts running.

We programmed the Butterfly Bootloader into the Atmega and first programmed it from the PC through UART. Visit the References for a tutorial prepared by us on this.

To program an Atmega loaded with the bootloader from another Atmega the INTEL HEX file format was understood:

INTEL HEX File:

```
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

🟨 Start code: 10 for data, 00 for end of file
🟩 Byte count, in hexadecimal
🟪 Address
🟥 Record type: 00 for data, 01 for end of file
🟦 Data, in hexadecimal format
🟪 Checksum

The data needs to be sent to the bootloader according to a specified protocol, for this we studied the Butterfly bootloader code. A difficulty encountered was that according to the bootloader the data has to be sent 16 bytes at a time, except in the end, whereas the hex file generated could have any number of hex files in a line.

Portable programmer:

The portable programmer displays a list of all the hex files stored in the SD card. The user presses a button to select a hex file. On pressing the button the other Atmega is programmed with the hex file through the UART interface.

Applications:

Two applications were developed for our application module.

Calculator:

The calculator application implements a GUI based calculator, with an interface not unlike the one found on most computers. The screen displays icons with options for the digits, trigonometric and logarithmic functions, clear, graph and evaluate. The user can click on the box with the PS/2 mouse to select a particular option. It features the ability to evaluate mathematical expressions of any size according to BODMAS rule. A very useful feature is the ability to draw the graphs of mathematical functions. It plots the graph on a domain and range specified by the user.

Hurdle Game:

The idea behind this game was to implement a real time based application and the implementation of PS/2 mouse and PS/2 keyboard. The game is designed in three levels and you are expected to clear the obstacles to reach a pre-designated position to win. The cursor is controlled with the PS/2 mouse. The speed with which the cursor moves increases progressively on clearing each level. The user enters his name through the PS/2 keyboard. The game makes use of interrupts to increase the speed.

Conclusion:

The programming module implements a portable programming capable of programming any Atmega with the bootloader at a very high Baud rate of 15200 bps directly from a hex file stored in the SD card. This has wide applications as multiple hex files can be stores with the ability to run them without the need for a PC to program them. We were also able to develop interactive applications and implement PS/2 mouse and keyboard interface for running the applications.

References:

https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B-rX2_85L6DXZjMxZmVjYzAtNjYwNi00ZTM5LThhOWItMzM3NmM3NzE0ZDdk&hl=en_US: For a tutuorial on Bootloader prepared by us.

http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html: For FAT and SD card libraries.

http://www.computer-engineering.org/ps2protocol/ : For information on PS/2 protocol and PS/2 mouse and keyboard.

http://dev.emcelettronica.com/what-microcontroller-bootloader-and-how-it-works: For general information on bootloaders.

http://en.wikipedia.org/wiki/Intel_HEX: For INTEL HEX file description.