XGSEmu - A XGS Micro Edition Emulator

Rainer Blessing

October 22, 2005

Abstract

A brief description of the design of an XGS emulator.

1 Overview

The XGSEmu tries to emulate the XGS Micro Edition, which you will very likely be an owner of. If not take a look at XGameStation.com . I recommend buying one. It can keep you entertained for years.
There are still a lot of things which are missing to call it complete. Some details might be wrongly implemented. Test your programm every now and then on the real hardware to avoid surprises :)

2 Emulators

In the past I was impressed by emulators. I thought they were difficult to create and that is certainly true for an Amiga or a PPC emulator. Writing emulators for easier systems like the XGS is a managable task, it does not take years to be complete
The directory *XGSME_HW_CD/General_Papers* on the XGS CD contains some articles about writing emulators. I also found this Howto on the internet. To make sure I was heading in the right direction with my implementation I also looked at some emulator source code.

3 Design and Implementation

I started with writing a debugger where I could see the opcode, memory and register contents so that I could test the CPU emulation. This should be the first step if you are creating an emulator.
I implemented the emulation of the mov commands and then the rest of the commands followed. More and more programs began to work which was quite satisfying. The video emulation followed and then the emulation of joystick and SRAM.

3.1 CPU

The CPU as the heart of a computer is the most important part in an emulator. Emulation should be as fast possible and also as accurate as possible, because most of the time is spent in the cpu emulation. Speed is not an issue if you are creating a C64 emulator, but the XGS has a higher clock than the PS1 or the N64.

There are two ways to handle to emulation of the opcodes. You can either use a function table, or a switch statement. I used a switch statement which is significantly faster than a function table. In the first versions of the emulator I called the emulation function for every emulated command, which made the emulation very slow. I didn't realize that a function call to another object would consume that much time. I later put the switch statement into a loop. The loop executes one millon emulated commands before it returns.
I used the instruction set table in [1] as a reference for the emulated commands.
Writing a CPU emulation is very rewarding. You can learn all the assembler commands and can train your ability to handle binary and hex numbers.

3.2 Video

To completely measure the length of the signal fragments at the beginning of a video line, you would need to have a timer with a nanosecond resolution. Such timers exist but then it would also be neccessary to emulate the SX52 at 80MHz. That would slow the emulation down considerably. The video speed is therefore synchronous to the emulation speed. This way you also can run the emulator on a slower PC. The screen will be updated less often, but the emulator will be usable[1].
Everytime something is written to the RE register which is connected to the video hardware on the XGS a function is called. The function parameters are the current clock count of the CPU and the colour value written to the register. By subtracting the previous clock count from the current clock count you get the length of the line segment. The length is used to detect a new line, new frame and colour burst. The values are written to circular array. The visible part of the signal is drawn on the video surface in the specified colour, which is in the previous array slot. The colours are calculated every time a line segment is drawn by multipling the RGB value by the intensity. They are the left and right nibbles of the colour written to the RE register. An obvious optimization would be to precalculate all combinations.
The colour burst colour is added to the original colour and the result is wrapped at 28 (modulo 28) to model the colour wheel[3].

3.3 Joystick

Like on the real hardware the joystick values are written to a variable which is serially shifted into the RA register. The register bits are used to detect the start of the value.
Only one joystick is emulated currently, joystick emulation does not have a high priority for me.

3.4 SRAM

The SRAM emulation works like the joystick emulation. The page part of the RAM address is serially written to a bit in the RC register. It was easy to implement after the joystick emulation was working.

[1] Sound on the other hand needs to be played at the original speed.

4 Epiloque

I hope you found this article useful and also the emulator itself. There hardly are any programs for the XGS availble apart from the ones on XGS CD. It would be great if the emulator could help you in the development of yours.

References

[1] Ubicom inc.,*SX-DDS-SX4852BD-15.pdf,* Ubicom inc.,2004

[2] LaMothe, Andre', *The Black Art of Video Game Console Design,* (Sams, 2005)

[3] Jack, Keith, *Video Demystified,* (Butterworth-Heinemann, 2004)

[4] Pazera, Ernest, *Focus On SDL,* (Course Technology, 2004)

[5] Smart, Julian et al, *Cross-Platform GUI Programming With wxWidgets,* (Prentice Hall, 2005)