Interface Development for the Point-of-care device based on SOPC

Hong Bum Son*, Sung Gun Song*, Jae Wook Jung*, Chang Su Lee*, and Seong Mo Park*

Abstract: This paper describes the development of the sensor interface and driver program for a point of care (POC) device. The proposed POC device comprises an ARM9 embedded processor and eight-channel sensor input to measure various bio-signals. It features a user-friendly interface using a full-color TFT-LCD and touch-screen, and a bluetooth wireless communication module. The proposed device is based on the system on a programmable chip (SOPC). We use Altera's Excalibur device, which has an ARM9 and FPGA area on a chip, as a test bed for the development of interface hardware and driver software.

Keywords: *Point-Of-Care, System-On-a-Programmable-Chip, Interface, Driver, Linux, μC/OS-II*

1. Introduction

With the rise of the standard of living, attention to illnesses such as diabetes, obesity and hyperpiesia has increased. For an effective medical treatment of these diseases, early detection and continuous care are very important. Such circumstances led to the invention of the POC device.

POC is a continual trend in the medical technology industry that delivers health care closer to both the patient and the health care provider. Some of the key features of POC are portability, deskilling, and near patient capability [1].

Many POC devices have entered the market. Most of them, however, have a poor user interface and the reuse of measurement data is too difficult and complex. We propose a POC device based on the 32bit ARM9 embedded processor. It has a full-color TFT-LCD measuring 320x240, and a touch screen for convenient user input. Using the ADS7859 ADC IC, the POC device can be fitted with eight sensors at the same time. In addition, we use a bluetooth module in order to transfer data to the PC and the network. This enables the user to manage bio-signal data in a wireless environment and doctors in hospitals to perform remote testing through network connection.

In Section 2, we describe hardware implementation, in Section 3, we describe software implementation, and in Section 4, we present our conclusions.

Manuscript received December 5, 2006; accepted May 22, 2007.
This work was supported by grant No. RTI04-03-03 from the RTI Program of the Ministry of Commerce, Industry and Energy (MOCIE) in Korea, and cad tools were supported by IDEC.
Corresponding Author: Seong Mo Park
* Dept. of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea
(momil@ciscom.chonnam.ac.kr,ssgun0@gmail.com,kawaii79@ciscom.chonnam.ac.kr,puppet0@empal.com, smpark@chonnam.ac.kr)

2. Hardware Implementation

2.1 System Architecture

We use the Excalibur device EPXA4, which has 400,000 gate FPGA areas. The proposed system consists of a 320x240 TFT-LCD, a touch-screen, multi-channel ADC, bluetooth and other PIO interface modules.

Fig. 1 shows a block diagram of the proposed system.

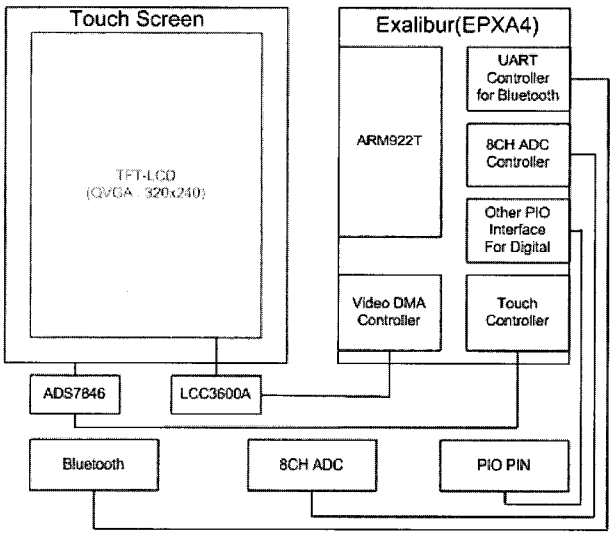


Fig. 1. Block diagram of the proposed point of care device

The proposed POC device requires an A/D converter controller for processing sensor data input, a bluetooth (UART) controller for transferring measurement data, a video DMA controller for displaying measurement data to the TFT-LCD, and a Touch-Screen controller for consuming the user input. Each controller contains control logics.

In addition, we implemented AHB slave logics to interface with the ARM9 processor.

2.2 Hardware Implementation

2.2.1 Video DMA Controller

In order to display on the TFT-LCD, we implemented a VDMA controller, as referenced in Application Note 287 (AN287) provided by Altera [2]. The VDMA controller works with three registers. The first register holds the size of the video data, the second register holds the address of the memory that has the video data, and the last register holds the control information such as start and stop. If the last register is written by '1', the DMA controller starts running. When the DMA controller is running, it reads data from the memory location pointed by the second register (the data size is 32bit, two pixels), then sends the data to the timing controller. The DMA controller has a counter to count the sent data. It compares the counted value with the video size and generates a stop signal when they are equal. It generates other control signals such as V-Blank and H-Blank as well. Fig. 2 shows a block diagram of the VDMA.

The VDMA controller supports 320x240 and 640x480 video sizes with 16bit color only.

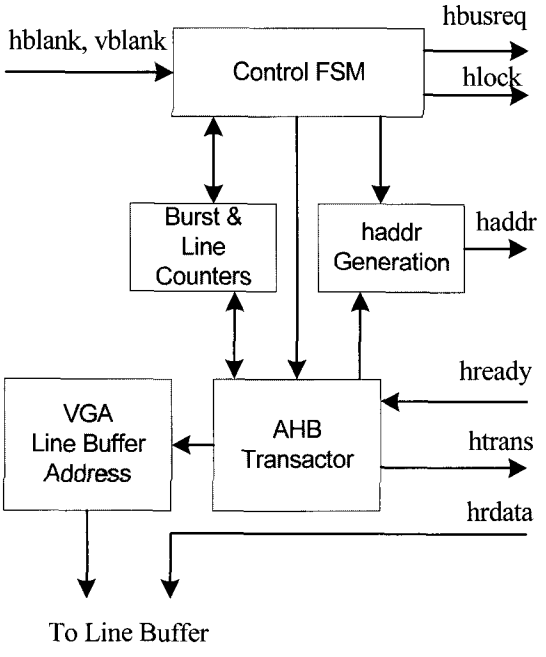


Fig. 2. Block diagram of the VDMA

To drive the TFT-LCD, a Samsung Electronics LCC3600A is used. The LCC3600A is a timing controller for 240x320, 260k-colors, Gate-IC-less TFT-LCD. From input signals such as data enable, main clock and RGB data, this IC generates a video signal and control signals for the source driver and integrated gate driver circuits [3].

The VDMA controller transfers the output video data from the video-memory to the LCC3600A according to the device input timing, as shown in Fig. 3 [3].

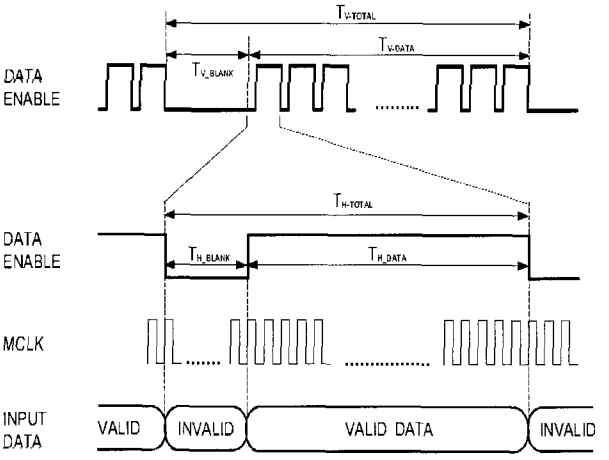


Fig. 3. LCC3600A device input timing

2.2.2 ADC Controller

The AD7859 is a high-speed, low power, 8-channel, 12-bit ADC chip which operates with a single 3 V or 5 V power supply [4]. In order to drive this IC, we must implement the interface logic. The AHB slave decoder output pin is connected to the CS pin of this IC.

The AD7859 contains a control register, ADC output data register, status register, test register and 10 calibration registers. The test register and calibration registers are read/write registers, the control register is a write-only register, and the status register and ADC output data register are read-only registers. All registers use the same pin. To read from a register, the user should write the control register first. The control register has RDSLT0 and RDSLT1 bits. These bits are decoded to determine which register should be addressed during a read operation.

The interface logic has twenty buffer registers. The register map is shown in Table 1. When the processor accesses these registers, a control word is automatically generated by the interface logic.

Table 1. Buffer register map of the interface logic

Buffer register range	Register name
0	Status register
1	Test register
2 ~ 10	ADC output data register
11 ~ 20	Calibration register

To read the ADC data, it writes '0x0' in the selected ADC register. Then, the interface logic generates a control word for the AD7859 and writes the data to the AD7859. Next, it reads data from the ADC output data register of the AD7859. Finally, it stores the data in the interface buffer register. Fig. 4 shows a connection diagram of the AD7859 with that controller.

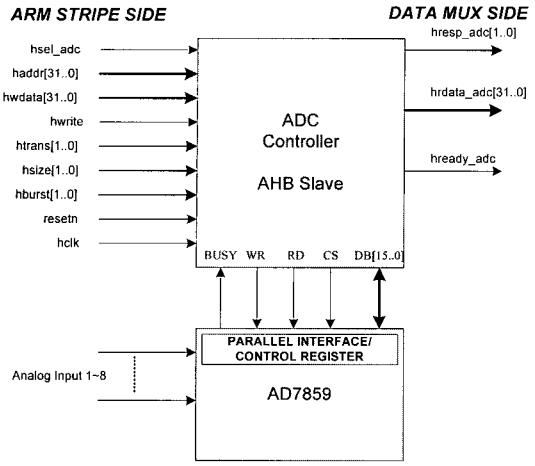


Fig. 4. Connection diagram of the AD7859 with controller

2.2.3 Touch Screen Controller

We use the touch screen to obtain data from the user. To drive the touch screen, Texas-Instrument's ADS7846 IC is used [5]. When the user touches the screen, this IC sends the voltage data of the touched point to the DOUT pin in serial format. In order to control the touch screen, we made an AHB Slave interface that has a serial interface with the ADS7846. This slave has three buffers: a control register to control the ADS7846, a bit shift register that stores the controller's serial data with the clock, and an interface data register. Once the data storage is completed, the interface logic sends a processor interrupt signal.

2.2.4 Other Interface Logics

We use bluetooth for data communication. We implemented the UART connecting the bluetooth module at its end point, and we implemented a PIO slave module in order to extend the IO.

All the interface logics are implemented with Altera's Quartus 4.0. Fig.5 shows the response data MUX and slave decoder. Table 2 shows the memory map of the interface logics.

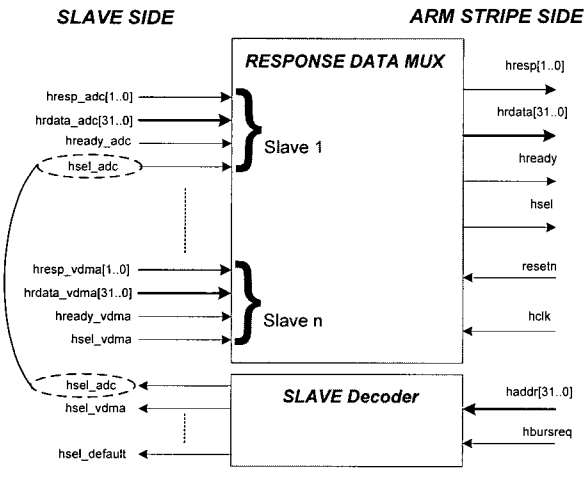


Fig. 5. Response data MUX and slave decoder

Table 2. Summary of the memory map of the interface logics

Address	Logic	Comment
0x80000000	Video-DMA	DMA Start
0x80000004		Video data address
0x80000008		Video size
0x81000000	Touch Controller	Control data write
0x81000004		Position data read
0x82000000 ~ 0x82000080	ADC Controller	ADC control data
0x83000000	UART/ BLUETOOTH	BT control Register
0x83000004		BT data read/write
0x84000000	PIO Interface	PIO data out register
0x84000004		PIO data in register

3. Software Implementation

In order to drive the hardware, it is necessary to select an operating system and implement the device drivers. To operate the POC device, we can use Linux or µC/OS-II. At first, we made a device driver using c language on the Linux. To port Linux, we used a patch script for ARM and Excalibur from www.arm.linux.org.uk. In the case of µC/OS-II, we implemented a function level device driver. To port µC/OS-II, we referred to Application Notes 1011 and 1014 from Micriµm's website [6][7].

Furthermore, in order to display the sensor data on the TFT-LCD, we implemented a graphics library which works on Linux and µC/OS-II.

3.1. µC/OS-II Porting

Fig. 6 shows the porting list of µC/OS-II. In order to port µC/OS-II, we needed to modify the source code and implement some functions. To run µC/OS-II, a timer is

Modify Data types : OS_CPU.H
BOOLEAN, INT8U ,... , OS_STK
Define CRITICAL METHOD : OS_CPU.H
OS_CRITICAL_METHOD etc.
Define Macros : OS_CPU.H
OS_ENTER_CRITICAL() etc.
Define Functions : OS_CPU_A.asm
OSCtxSW(), OSStartHighRdy(),
OSIntCtxSw(), OSTickISR()
Define Functions : OS_CPU_C.c
OSTaskStkInit(), OSInitHookBegin(),
OSInitHookEnd(), OSTaskCreateHook(),
OSTaskDelHook(), OSTaskSwHook(),
OSTaskStatHook(), OSTCBInitHook(),
OSTimeTickHook(), OSTaskIdleHook()
Other BSP

Fig. 6. µC/OS-II architecture and µC/OS-II porting list

required. The timer regularly supports a time-tick. The OS receives the time-tick, and then runs the task scheduling. Excalibur (with two timers) supports three different modes: the free-running heartbeat mode, the on-shot delay mode and the software interval timer mode [8]. We used the free -running heartbeat mode.

3.2 Graphic Library

Presenting a sensor data with VDMA requires video graphic functions such as draw line, put dot, draw box, and draw text. We developed a graphic function library at the source level that works with three methods: simple firmware, real-time OS based program and embedded Linux program. The video graphic library works with a user-defined video page which defines a resolution. The resolution of the video output is defined in the header file "draw.h." We can easily implement a double page method. When using the double page method, the user defines two video memory addresses and calls a function with arguments that include video page number.

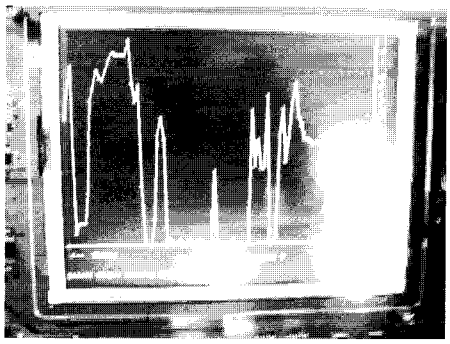
Fig. 7 shows an example of the drawline function, which has six arguments. The first argument means the video page address. Its data type is an unsigned char pointer value. The second argument means start x, the third means start y, the fourth means end x, the fifth means end y, and the last argument means the pixel value. Since the TFT-LCD only supports 16bit color, the pixel value structure is 16bit.

The drawtext function has five arguments. The first argument means the video page address, which has the same value as the video page address of the drawline function. The second argument means start x, the third start y, and the fourth the output text type char pointer value. The last argument means directions 0 or 1, where 0 means the horizontal direction and 1 means the vertical direction.

```
drawtext(pfbmap,0,300," <-- Timing ",0);
drawtext(pfbmap,220,20,"^| Output(Voltage)",1);
//drawtext(pfbmap,0,450,"Sensor Output Drawing Program v1.0",0);
//drawtext(pfbmap,610,410,"0",0);
drawline(pfbmap,0,300,210,300,makepixel(255,255,255));
drawline(pfbmap,210,0,210,300,makepixel(255,255,255));
```

Fig. 7. Example for the drawtext and drawline functions

Fig. 8 shows a screen captured image of an application with a distance sensor that uses Infrared (IR). Fig. 8(a) shows an image run under the pxa255 processor. Fig. 8(b) shows an image run under the Excalibur device.



(a) Image run under PXA255

(b) Image run under Excalibur

Fig. 8. Screen captured images of the IR distance sensor

4. Conclusions

We proposed a POC device which has an ARM9 processor and multi-channel input.

The proposed POC device has a wireless interface using bluetooth to send data to networks. Thus, an attending physician in a hospital can diagnose and treat a patient at a remote site with a wireless networking function. It has a user-friendly interface and can be operated by the user in the stand alone mode. It will support more powerful applications since it is based on the ARM9 processor.

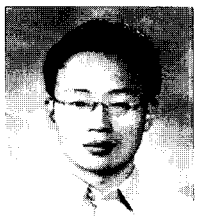
Additional technology, such as the wearable POC system [9] and POCT [10], may be added to the proposed POC device. Moreover, all interface logics and control ICs should be integrated into a single chip to be developed as a commercial product.

References

[1] T. O'Brien, N. Linton, M. Jonas, R. Linton and D. Band, "Point-of-care sensor technology for critical care applications," IEE Colloquium on New Measurements and Techniques in 3, pp. 6/1-6/3, December. 1996.

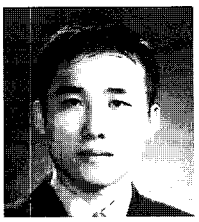
[2] Altera, Application Note 287, Using Excalibur DMA Controllers for Video Imaging Version 1.1, February 2003.

[3] Samsung Electronics, LCC3600A, March 2003.
[4] Analog Device, 3V to 5V Single Supply, 200kSPS 8-Channel, 12-Bit Sampling ADCs – AD7859/ AD7859L – REV.A, Data Sheet, 1996.
[5] Texas Instruments, ADS7846 Touch Screen Controller, Data Sheet, March 2003.
[6] Micriμm, Application Note 1011: μC/OS-II and The ARM Processor, 2004.
[7] Micriμm, Application Note 1014: μC/OS-II and The ARM Processor, 2004.
[8] Altera, Excalibur Device Hardware Reference Manual Version 3.1, November 2002.
[9] Y. Jianchu, R. Schmitz and S. Warren, "A wearable point-of-care system for home use that incorporates plug-and-play and wireless standards," IEEE Transactions on Information Technology in Biomedicine Volume 9, Issue 3, pp 363-367, September 2005.
[10] J. McGrory, O. Lynch and E. Coyle, "Design of a wireless system for patient-hospital communication and result validation in point of care testing," in Proc Computational Intelligence and Multimedia Applications, pp.212-217, August. 2005.

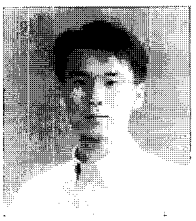


Hong Bum Son
Mr. Son received a B.S. degree in Computer Engineering from Chonnam National University in 2005. He is now undertaking a master's course as a member of the Computer Architecture Laboratory at Chonnam National University. His research interests include hardware interfacing, embedded systems and software, real-time operating systems, and digital control circuits.

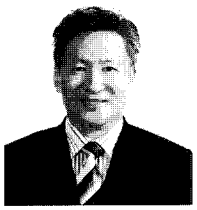
Sung Gun Song
Mr. Song received a B.S. degree in Computer Engineering from Honam University in 2004. He is now undertaking a doctorate's course as a member of the Computer Architecture Laboratory at Chonnam National University. His research interests include computer architecture and embedded systems.



Jae Wook Jung
Mr. Jung received a B.S. degree in Computer Engineering from Honam University in 2006. He is now undertaking a master's course as a member of the Computer Architecture Laboratory at Chonnam National University. His research interests include hardware interfacing and VoIP software.



Chang Su Lee
Mr. Lee received a B.S. degree in Computer Engineering from Chonnam National University in 2005. He is now undertaking a master's course as a member of the Computer Architecture Laboratory at Chonnam National University. His research interests include HDL for SOC.



Seong Mo Park
Dr. Park received his B.S. degree in Electronics Engineering at Seoul National University in 1977 and his M.S. degree in Electrical and Electronics Engineering at the Korea Advanced Institute of Science and Technology in 1979 in Seoul, Korea. He received his Ph.D. degree in Electrical and Computer Engineering at North Carolina State University in Raleigh, NC in 1988. He worked as an IC design engineer at the Korea Institute of Electronics Technology in Korea from 1979 to 1984. He was with Old Dominion University in Norfolk, VA from 1988 to 1992 as an assistant professor. He is now with the Computer Engineering Department of Chonnam National University in Gwangju, Korea as a professor. His research interests include digital signal processing, algorithm specific architecture, embedded systems and SOC design. Dr. Park is a member of IEEE, IEEK and KICS.