The Design of A Creative Engineering Robot with MCU Platform*

Hong, Seon Hack**

MCU 플랫폼 창의 공학용 로봇 설계

홍 선 학

〈Abstract〉

In this paper, the implementation of creative engineering robot with MCU platform is described. This robot, as a platform of robot system to be used as creative engineering education, has to satisfy restrictions in many aspects in order to study algorithm and apply for the processor based function and pattern recognition application.

Considering many restrictions of the mobile platform for creative robot system, we made this robot autonomous by using efficiently the LINUX embedded system. And we choose Marvell Monahan processor(PXA320) as MCU flatform, and used CentOS5.2 as an embedded OS that has the function of robustness and optimality. For flexibility and modularity, the platform has expansion ports. The results of experiment are described to show the pattern matching of creative engineering mobile robot with LINUX programming environments.

Key Words : Creative Robot, PXA320, LINUX, RTOS

Ⅰ. Introduction

The Monahan processor is an integrated system-on-a-chip microprocessor for high-performance and low-power portable device. It incorporates the Intel XSclae micro-architecture with on-the-fly voltage and frequency scaling and sophisticated power management to provide industry leading MIPS/mW performance. It complies with the ARM architecture V5TE instruction set and follows the ARM programmer's model and is available in a discrete package configuration. The Monahan processor memory architecture provides greater flexibility and higher performance than that of previous core products. It provides the configuration support for two dedicated memory interfaces to support high speed DDR SDRAM and data flash devices. This flexibility enables high performance "store and download" as well as "execute in place" system architectures. It

* 본 연구는 2008학년도 서일대학 교내학술연구비지원으로 수행되었음.

** 서일대학 컴퓨터전자과 교수

contains six 128-Kbyte banks of on-chip SRAM, which can be used for a combination of display frame buffer, program code, or multimedia data. Each bank can be configured to retain its contents when the processor enters a low-power mode[1].

128 of the peripheral pins on the Monahan processor provide software controlled general purpose I/O (GPIO) functionality. The key features of the GPIO controller are such that as inputs, GPIO pins can be sampled or programmed to generate an interrupt from either a rising or falling edge. As outputs, GPIO pins can be individually cleared or set and can be preprogrammed to either state when entering sleep mode.

The camera image-capture interface has these features: There are several supported vertical and horizontal resolutions from 176×144 to 2560×2048. Programmable interrupts for FIFO overflow, end of line and end of frame. Support for 8- and 10-bit RAW(RGGB, CMTG, etc) capture modes. Histogram unit generates statistics for image data[2, 3].

Ⅱ. Basic Theory

This chapter provides a fundamental overview of the platform creative robot system. The processor read and write the peripheral registers and FIFOs on the peripheral bus using the bridge in Fig. 1. All internal registers of the peripherals must be accessed using word access loads and stores. Internal register and FIFO space must be mapped as non-cacheable. Byte and half-word accesses to internal registers are not permitted and yield unpredictable results.

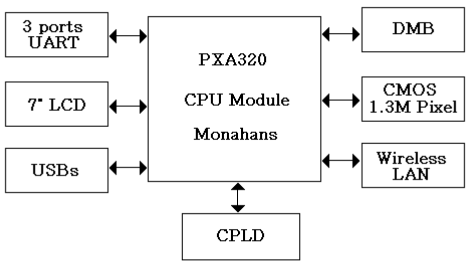

Fig. 1. TKU 320 Blockdiagram

2.1 Accessing Peripherals on the Bus

The peripheral bus modules connect to system bus via the DMA/bridge unit. Peripherals on the peripheral bus can be accessed with either programmed I/O using the bridge, or by using a DMA transfer. DMA can be programmed to transfer from the peripheral to the memory(receives) or from the memory to the peripheral(transmits).

For a read operation, there are two system-bus transactions: an initial transaction to send the request, and a separate data transfer to return the read data. During the gap between these two operations, the system bus is relinquished and can be used by other devices. For a write operation, the system-bus transaction completes when the write has transferred over to the DMA/bridge rather than waiting until it reaches the actual peripheral.

All on-chip interrupts are enabled, masked, and routed to the core FIQ or IRQ. Each peripheral-unit interrupt is enabled or disabled at the source through an interrupt-enable bit. Generally, all interrupt bits in a peripheral unit are ORed together and present a single value to the interrupt controller[4-6].

2.2 The Java Platform

A platform is the hardware or software environment in which a program runs. We used LINUX(CentOS 5.2) as platform here. Also platform can be described as a combination of the operating system and underlying hardware. The java platform differs from most other platforms in that it's a software - only platform that runs on the top of other software - based platforms.

The java platform has two components :

- The virtual machine(Virtual Box)
- The application programming interface

Fig. 2 illustrates the graphical overview of the platform[7-8].

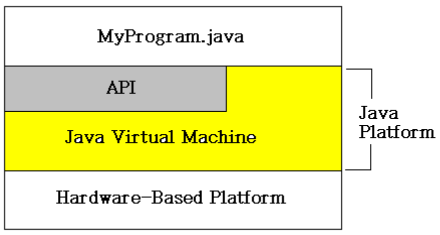

Fig. 2. Java Platform

The virtual machine is the base for the java platform and is ported onto various hardware - based platforms. The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces: these libraries are known as packages. The API provides the core functionality of the java programming language. it offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more. As a platform - independent environment, the java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

2.3 Setup the leJOS API

The name lejos means 'far' in spanish. In the name leJOS, the letters JOS are capitalized because those letters stand for Java Operating System. Since le means 'the' in several languages, this would mean the Java Operating System. The leJOS JVM(Java Virtual machine) is written in C code in a very platform independent style, which means it is easily ported to other machines. So far it has appeared on the creative robot platforms.

There are also tools on the PC side to compile and upload code to the leJOS JVM. leJOS is multi platform, and recently that means Windows, Linux, and Macintosh. leJOS software is available for each of these platforms, allowing you to develop the program under your favorite operating system. In this paper we use the Windows that is to download the latest version from www.lejos.org. One of the best open source IDE(Integrated Development Environment) is Eclipse by IBM. It's free, powerful, and easy to use. It makes sense to use a more advanced IDE with the hardware since our code can grow quite large.

As leJOS is a firmware replacement, the new firmware leJOS firmware must be flashed onto the MCU, and will replace the standard firmware. This wipes out any files currently held on the standard firmware. The leJOS is an open source project hosted in the sourceforge repository. it was originally created from the TinyVM that implemented a Java VM for the RCX system. There are many advantages of using leJOS rather than other programming environment. These include :

- It uses the industry-standard Java language.
- It provides object-oriented environment.
- It has full support for Bluetooth.
- It has advanced navigation support.
- It supports third party sensors.
- It provides trigonometry and other Math functions.
- It supports the J2ME LCD user interface
- including many graphics functions.
- It supports multi threading.
- It supports listeners and events.
- It has USB support including java stream.
- It has telerobotics support via standard TCP/IP sockets.
- It has sound support including playing 8-bit WAV files.
- It has computer vision and speech support via iCommand.

Ⅲ. Capture Interface

In this paper, the quick capture interface connects the processor and a compatible external image-capture module, which consists of a sensor providing RAW image data, a sensor with a minimal level of integrated processing on RGB or YCbCr image data, or the combination of a sensor with more sophisticated image-processing capability.

3.1 Operations of Quick Capture Interface

The quick capture interface operates in three modes : First is the preprocessed still-image/video capture mode. The image data is in YCbCr 4:2:2 color space. The image data is captured through various interface options, optionally scaled and/or bit sliced, and then formatted and packed before it is presented to system memory resources. The PXA320 processor also supports compressed JPEG streams. Second is RAW still image capture mode. Captures and formats image data to be processed using algorithms targeting display or print quality. The image data is in the four element Bayer pattern of RGGB color space. The data is captured through the various interface options, optionally corrected for dead pixels, companded, black-level clamped, and then packed before it is presented to memory resources. Third is RAW Video-image capture mode. The sensor provides image data in the RAW RGGB color. The integrated pixel processing chain supports the conversion to RGB 8:8:8 and YCbCr 4:2:2 color space through several functional units, which include : Spatial scaling unit(SSU), Pixel substitution unit(PSU), Companding/black-level clamp/gamma correction unit(CGU), Color synthesis unit(CSU) and Color management unit(CMU). The RAW still-image capture is limited to a maximum of 2560×2032 resolution. A statistical unit is introduced to support more sophisticated exposure metering, and a dedicated four-channel DMA is introduced to increase frame rate and resolution[9-11].

3.2 Features and Operations of Image process

In this paper, We use the features of the capture interface such as : Horizontal resolution(from 64 to 2560) and Vertical resolution(from 32 to 2032). Programmable processor clock output to sensor from 187kHz to 52MHz. Programmable interrupts for FIFO overflow, end of line and end of frame. Programmable interface timing signals for internal and external synchronization signaling. Three programmable 64-element look-up table(LUT) : three independent mapping functions($f_R(x)$,$f_G(x)$ and $f_B(x)$), companding from 10-bit to 8-bit RAW data, and programmable black-level clamp (BLC) offset. Histogram unit generates statistics from image data, such as : increment saturates to avoid rollovers, up to 64K pixels per data value, and 8-bit and 10-bit data stream with 32-bit result.

The quick capture interface operates in standalone mode as a still and video capture, encode, decode and display device. The four most common image/video operating modes are : video and image preview/review, video conference, video-clip generation and still-image capture. Figure 3 shows the quick capture interface connection.

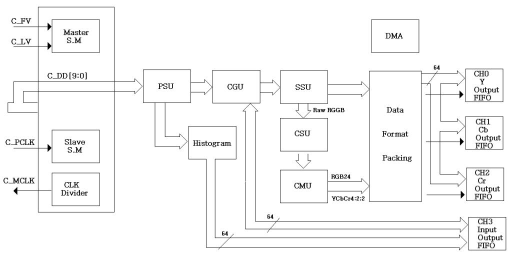

Fig. 3. Quick Capture Interface Block Diagram

Two classes of data(preprocessed and raw) are provided to the camera controller. The use of resources within the quick capture interface depends on both the level of processing provided by the external camera module and the intended purpose. The image sensor can provide raw data through parallel interfaces supporting 8-bit or 10-bit raw pixel formats[12-15].

3.3 Graphic and Input Controller

The input data from the vision sensor is packed into the required format and written into the FIFO. The capture interface has three separate FIFOs that act as temporary storage for the captured video/image data. The read channel uses the fourth FIFO to load the companding LUT and also to send the histogram data to memory. All FIFOs are read-only. Any write to the FIFOs results in a system-bus error. When an end of frame is encountered during data capture, the remaining bits in the FIFO entry after the last data is written are stuffed with zero. For example, if the last sample of a frame occupies byte 0 of the 8-byte FIFO entry, bytes 1 through 7 are written with zeros. The first valid data of the next frame is loaded to the next FIFO entry. Zero stuffing at the ned of frame takes a finite time that depends on the last valid byte position and whether planarized YCbCr mode is enabled. If the next frame capture starts before zero stuffing of the previous frame is complete, incorrect Quick capture Interface operation can result.

When a FIFO reaches threshold level, the DMA controller transfers a burst of data from the FIFO to the destination address specified in the Quick Capture Interface DMA descriptor. The DMA descriptors must be programmed to transfer the preferred number of bytes. The number of bytes in the QCI FIFO corresponding to a frame is always a multiple of 8 bytes, and the DMA controller should read the entire frame data including padded zeros before moving on to read data corresponding to the next frame. The procedures of writing to memory are as follows : 1. Program the CIDADRx register with the descriptor address to be fetched. 2 Set the preferred values in the control registers and DMA Control/Status register. 3 When a FIFO has data, the descriptor is fetched and loads the CITADRx, CISADRx, and CICMDx registers with appropriate values. 4. After the descriptor is fetched when the FIFO threshold is fetched, the data transfers occur until the length field in CICMDx is zero.

Channel 3 is the only channel that performs reads from memory. This channel reads command data for the compander lookup table. When the QCI software driver initiates a required for data, the dedicated DMA controller transfers data from a memory address space specified in the DMA descriptor. The programming model for using the Quick Capture Interface DMA as follows: 1. Program the CIDADR3 register with the descriptor address to be fetched. 2. Set the preferred values in the control registers and DMA Control/Status register. 3. When CICCR is set, the descriptor is fetched and loaded into the CISADR3, CITADR3, CIDADR3 and CICMD3 registers with appropriate values[16, 17].

Ⅳ. Image Functional Units

In this paper, we describe the functional blocks shown in Figure 3. There are 4 types of functional units.

4.1 Histogram Unit(HST)

Histograms are generated as a preprocessing step both for image metering used in exposure control and for determining the companding curve when RAW image data is captured. The histogram unit does not have to be a destination for data but instead snoops image data as is it passes by on the image data bus. After an image is directed at the histogram function, the data is stored in a RAM array accessible to Quick capture Interface software driver. The operation of the histogram unit is simply to take the data(up to 10 bits) directed to it as an index into the array and increment that location[18, 19].

4.2 Pixel Substitution Unit(PSU)

The PSU repairs known bad pixels of the sensor array. It performs this task with a sorted list of all bad pixels within the PSU. The list contains the row and column address of each pixel to be substituted. To replace the pixels, the nearest neighbor pixel of the same color is used as a substitute. In most cases the nearest left pixel (n-1) is used. When this is not possible, the nearest upper neighbor is used.

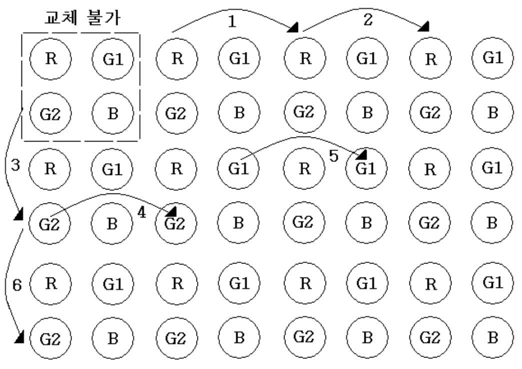

Fig. 4. Pixel Substitution to Bayer Pattern

The PSU restricts the number of bad pixels to a maximum of 128, and the very first pixel of each color cannot be marked as bad even if it is damaged, the choices are to re-adjust the cropping window or to ignore the bad pixel, with the consequence that it can be used in or copied into other bad pixels. The pixel substitution to a Bayer pattern is illustrated by Fig. 4[20-21].

In any color plane (4 pixels RGGB), the algorithm for substitution is for the PSU to use the left neighbor if at all possible (case 1 and 5) ($P_i - 1$). If this is nat possible (case 3) (first column, i=0), then the upper pixel is used P(i, j-1).

This substitution can cascade in both ways(case 2, 4, and 6). If two bad pixels of one row are next to each other P(i, j) and P(i+1, j), the first bad pixel P(i, j) is substituted with its left neighbor P(i-1, j) if i>0, and the next bad pixel P(i+1, j) becomes the value of the first substitution P(i-1, j), Similarly, the vertical substitution cascaded, if required[22-23].

Figure 5 shows the procedure of overall image processing flow, at first it would start with the open of camera device driver, and then determine the image format with configuration. The captured image are saved with overlay file, and then process the binary work, expansion, contraction and partition of captured image. These functions consisted the Embedded hardware and PXA camera application. We experimented the image pattern recognition of creative robot platform in terms of avoiding the obstacles which had several shapes.

Figure 6 shows the experimental results the noised input image and median filtering image. We had a desired results through the several experiments.

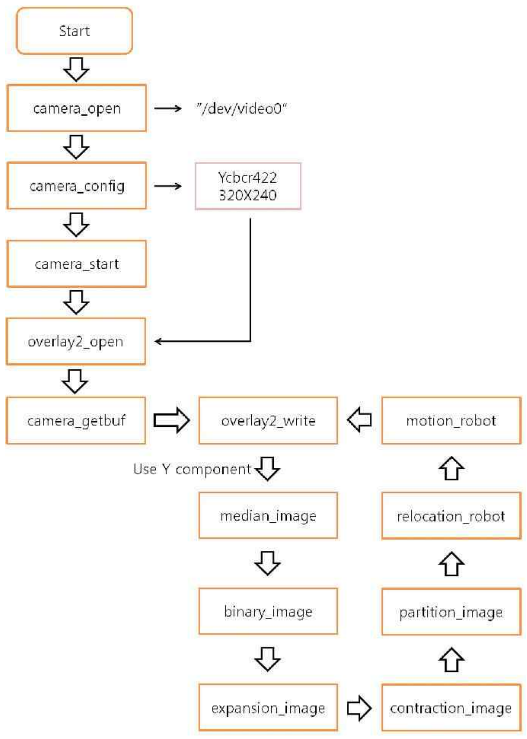

Fig. 5. Overall Function Flow

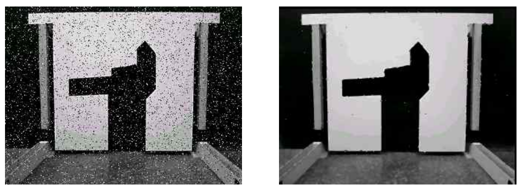

Fig. 6. Image processing results

Ⅴ. Conclusions

In this paper, we made the creative engineering robot with MCU platform. This robot, as a platform based robot system to be used as creative engineering education, has to accomplish the several works with 1.3 M pixel CMOS camera visionary pattern probability recognition application.

Considering a lot of restrictions of the embedded platform for creative robot system, we made this robot autonomous by using VirtualBox(Ver 3.0.4) development system efficiency. And we choose Marvell Monahan processor(PXA320) as MCU, RTOS as embedded OS that has the robustness and optimality. For flexibility and modularity, the platform has expansion ports. The results of experiment are described to show the density and maximum likelihood method of creative engineering mobile robot for data processing and pattern recognition verification.

References

[1] PXA3200TKU _Linux_Manual_V1. 7. Monahan P Processor. November 7, 2006.

[2] Marvell PXA320 Processor, Graphics and Input Controller, December 14, 2006.

[3] Marvell PXA320 Processor, Serial Controller Configuration, December 15, 2006.

[4] Mary Campione, Kathy Walrath, The Java Tutorial, Object-Oriented programming for the Internet, Addison-Wesley, 2001.

[5] Beck, K.. Pattern and Software Development. Dr. Dobbs Journal. 2. 1994.

[6] Jason Gu, Max Meng, Al Cook, Peter X, Liu. Sensor Fusion in Mobile Robot, Proceedings of the 4th World Congress on Intelligent Control and Automation, June 2002. pp. 10-14.

[7] R. Gartshore, A. Aguado, C. Galambos. Incremental Map Building using an Occupancy Grid for an Autonomous Monocular robot, 7th International Conference on Control, Automation, Robotics & Vision. Dec, 2002. Robot, MIIT Press, 1991.

[8] Brian Bagnall, Maximum LEGO NXT, Building Robots with Java Brains, Varian Press, 2007.

[9] 홍선학, UML기반의 창의공학용 로봇설계, 한국통신학회, 제33권, 제10호, 2008, pp. 343-349.

[10] 홍선학, GUI환경을 갖는 퍼지기반 이동로봇제어, 대한전자공학회, 제43권, IE편, 제4호, 2006, pp. 340-347.

[11] 홍선학, 센서결합을 이용한 이동로봇제어 대한전자공학회, 제42권, TE편, 제2호, 2005, pp. 91-98.

[12] 홍선학, 영상 추적을 이용한 이동로봇제어 대한전자공학회, 제42권, TE편, 제4호, 2005, pp. 201-208.

[13] Martin, R. Designing Object-oriented C++ Application Using the Booch Method, Englewood Cliffs, NJ: Prentice-Hall.

[14] Frank L. Lewis, Optimal Estimation, John Wiley & Sons, 1986.

[15] Gordon McComb, Robot Builder's Bonanza, TAB Books, 1987.

[16] Dave Prochnov, Mindstorms Hacker's Guide, McGraw Hill, 2007.

[17] Benjamin Erwin, Creative Projects with Mindstorms. Addison Wesley, 2001.

[18] Jim Leden, Embedded Control Systems in C/C++, CMP Books, 2004.

[19] Fred G. Martin, Robotic Explorations, A Hand-On Introduction to Engineering, 2001.

[20] An Introduction to ROBOT TECHNOLOGY, Philippe Coiffet, 1982.

[21] PXA320_Monahans_P_DM_Vol[1]+Rev_0.95 System and Timer Developers, 11, 7, 2006.

[22] Richard O. Duda, Peter E. Hart, David G. Strok, Pattern Classification, 2nd Edition Jouh Wiley & sons, INC, 2001.

[23] Wirfs-Brock,. R., Wilkerson. Designing Object-Oriented Software, Englewood Cliffs, NJ. : Prentice-Hall. 2002.

■ 저자소개 ■

홍 선 학
Hong, Seon Hack

1992년~현재
서일대학 컴퓨터전자과 교수
1994년 광운대학교 대학원 박사 졸업.
1988년 광운대학교 전기공학과 석사졸업.
1985년 광운대학교 전기공학과 학사졸업.

관심분야 : 제어, 컴퓨터응용, 로봇분야등
E-mail : hongsh@seoil.ac.kr

논문접수일 : 2009년 10월 20일
수 정 일 : 2009년 11월 15일 (1차)
2009년 11월 25일 (2차)
2009년 12월 6일 (3차)
게재확정일: 2009년 12월 10일