LKV-080A
SW 사용 설명서

Version Information

BSP Version : Version 1.3.1

True Leader of Technology

|주|엘케이일레븐

알 림

여기에 실린 내용은 제품의 성능 향상과 신뢰도의 증대를 위하여 예고없이 변경될 수도 있습니다.

㈜엘케이일레븐
경기도 성남시 중원구 상대원동 191-1 SKn 테크노파크 메가동 1306 호

목차

알 림 2
1. Introduction 5
1.1. LKV-080A B/D 의 소개 5
1.2. 본 문서의 내용 5
2. Boot Parameter 6
2.1. Boot Parameter 정보 6
2.2. Boot Parameter 입력 예 6
2.3. Second Ethernet 을 설정할 경우 IP 입력하는 방법 8
2.4. Second Ethernet 을 이용하여 부팅하는 경우 10
2.5. PMC Ethernet Card 를 사용할 경우 IP 입력 방법 11
3. LKV-080A Control Method 13
3.1. LKV-080A B/D Memory Map 13
3.1.1. NVRAM 13
3.1.2. Flash Memory 14
3.2. The contents of a LKV-080A B/D IO Control 16
3.2.1. Watch dog time out reset control 16
3.2.2. Front Panel Led Control Method 16
3.2.3. Dip Switch Read method 16
4. LKV-080A Serial Control method 17
4.1. LKV-080A B/D Serial Driver Initialization 17
4.2. vxWorks IO System 17
4.3. RS-485 통신 사용시 셋팅 방법(Port 2,3,4) 19
5. LKV-080A B/D VMEBus Control method 20
5.1. LKV-080A B/D VME Memory Map 20
5.2. LKV-080A B/D VME Bus Access 20
5.2.1. Extended Mode 일 경우 VMEBus Access 20
5.2.2. Standard Mode 일 경우 VMEBus Access 20
5.2.3. Short Mode 일 경우 VMEBus Access 20
5.3. LKV-080A VME Bus Access Example 20
5.3.1. Extended Mode Access Example 21
5.3.2. Standard Mode Access Example 21
5.4. LKV-080A B/D AM Code Setting Function 22
5.4.1. VME AM Code Setting Function 22

그림 목차

그림 1. LKV-080A B/D 의 Boot Parameter 입력정보.....7
그림 2. Ethernet 1 만 설정 했을 경우의 Boot From Network 8
그림 3. Ethernet 2 사용시 설정된 화면 9
그림 4. Ethernet 2 사용한 부팅 10
그림 5. PMC Ethernet Card 를 사용했을 경우 Boot From Network 12

표 목차

표 1. LKV-080A Memory Map.....13

1. Introduction

1.1. LKV-080A B/D 의 소개

LKV-080A B/D 는 8245 processor 를 사용한 VME Single Board Computer 이다. MPC8245 는 MPC603e Core 가 내장되어 있으며, 내부 bus 는 33MHz 의 32bit PCI Bus 와 133MHz 의 SDRAM Memory Bus 로 구성되어 있다. Parallel Interface 는 16Mbyte 의 Flash Memory 와 128Mbyte 의 SDRAM module, 512kbyte 의 NVRAM 으로 되어 있으며, VME Bus Access 기능은 FPGA 로 구현되어 있다.

1.2. 본 문서의 내용

LKV-080A B/D 는 Board 내에 128Mbyte 의 SDRAM, 16Mbyte 의 Flash Memory, 512Kbyte 의 RTC/NVRAM, 1Mbyte 의 EPROM 으로 구성되어 있다. 또한 Serial Port4 개와 VME Bus Interface 가 구현되어 있어 Master/Slave 보드로 사용 가능하다. 본 문서 에서는 위와 같은 보드 구성에 따라

1 장에서는 LKV-080A B/D 에 대한 소개 및 구성에 대한 내용이 수록되어 있으며,
2 장부터는 보드 사용자가 LKV-080A B/D 사용 시 Setting 해야 할 Boot Parameter 에 관해 자세한 설명이 되어 있다.
3 장에서는 본 Board 에서 지원되는 다양한 memory type 의 control 방법 및 예가 주어지며,
4 장에서는 Serial Port 설정 및 사용 방법에 대해서 소개되며 그에 관한 예가 주어진다.
5 장에서는 VME Bus 의 control 방법이 설명되어 있다.

2. Boot Parameter

2.1. Boot Parameter 정보

```
boot device          : fei
unit number          : 0
processor number     : 0
host name            : 080
file name            : vxWorks
inet on ethernet (e) : 220.76.45.2
host inet (h)        : 220.76.45.200
user (u)             : 080
ftp password (pw)    : 080
flags (f)            : 0x8
target name (tn)     : LKV-080
other (o)            :
```

위의 내용은 Boot Parameter 이다. 먼저 "boot device"는 Ethernet driver 의 name 을 가리키고 Board 에 따라 달라지며, Device Driver 의 구성도 또한 다르다. File name 은 VxWorks image 가 있는 경로명을 적어준다. [그림 2-1]은 VxWorks Boot 를 하기위한 boot parameter 의 실제 입력 정보이다.

inet on Ethernet(e)는 Board 의 IP Address 를 적어주며,
host inet 은 VxWorks image 를 download 할 host 의 IP Address 를 적어준다.
user(u)와 ftp password 에는 ftp demon 의 ID 와 Password 를 적어주면 된다.

여기서 Debug 용 Console program 은 Tera Term 이나 HyperTerminal 을 사용한다. BSP 는 FTP 를 이용하여 host 로부터 OS + Application image 를 download 한 후에 이를 실행한다.

2.2. Boot Parameter 입력 예

VxWorks Boot Prompt([VXWorks Boot]:) 상에서 'p'를 입력하면 그림 [2-1]과 같은 Boot Parameter 에 관한 정보를 볼 수 있다. 또한 Prompt 상에서 'c'를 입력하면 Boot Parameter 에 대한 정보를 사용자의 환경에 맞게 setting 가능하도록 되어 있다. Boot Parameter 의 입력이 끝난 후 Prompt 상에서 '@' 를 입력하면 Ethernet 을 통해 부팅이 실행되는 것을 확인 할 수 있다.

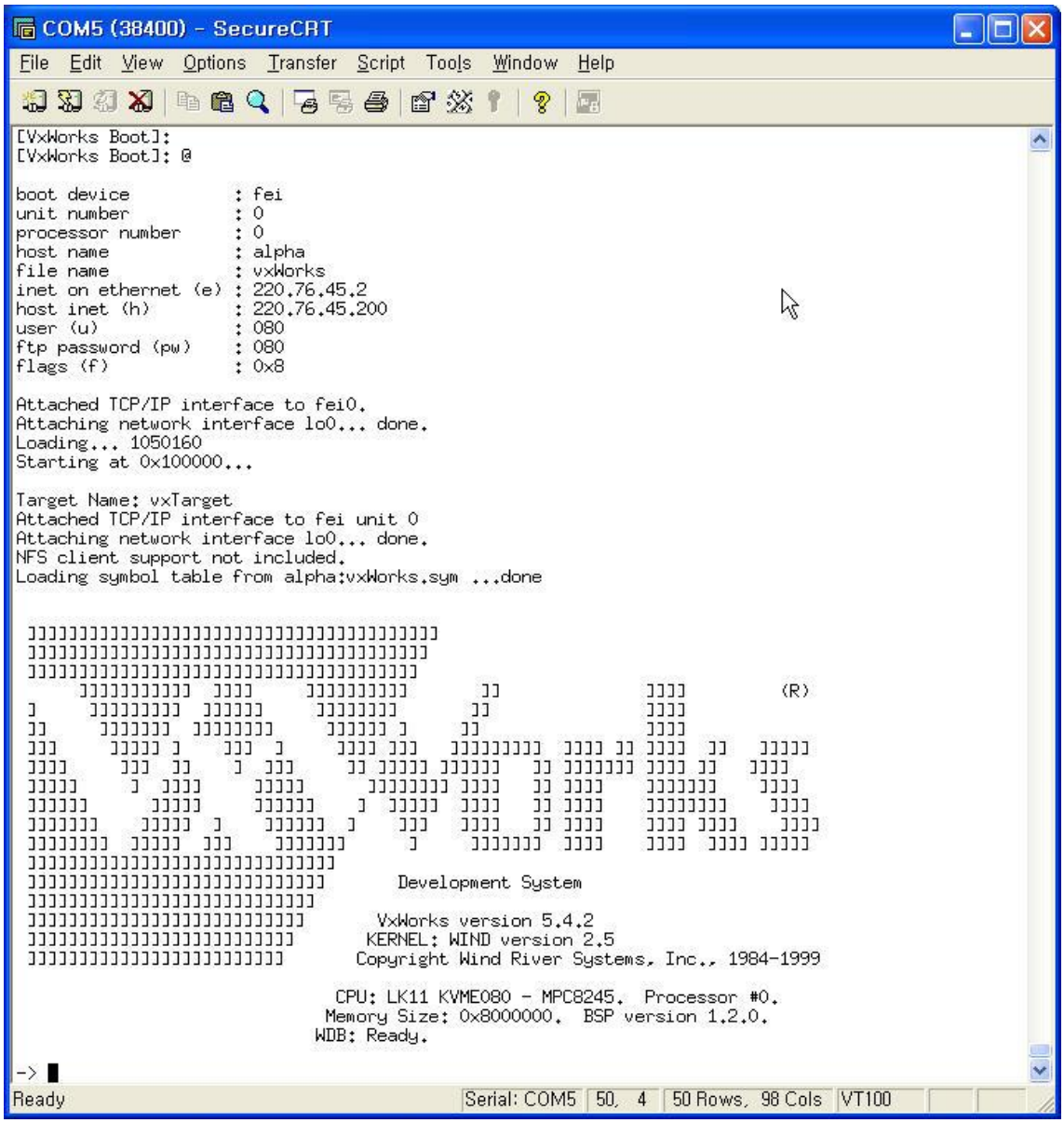

그림 1. LKV-080A B/D 의 Boot Parameter 입력정보

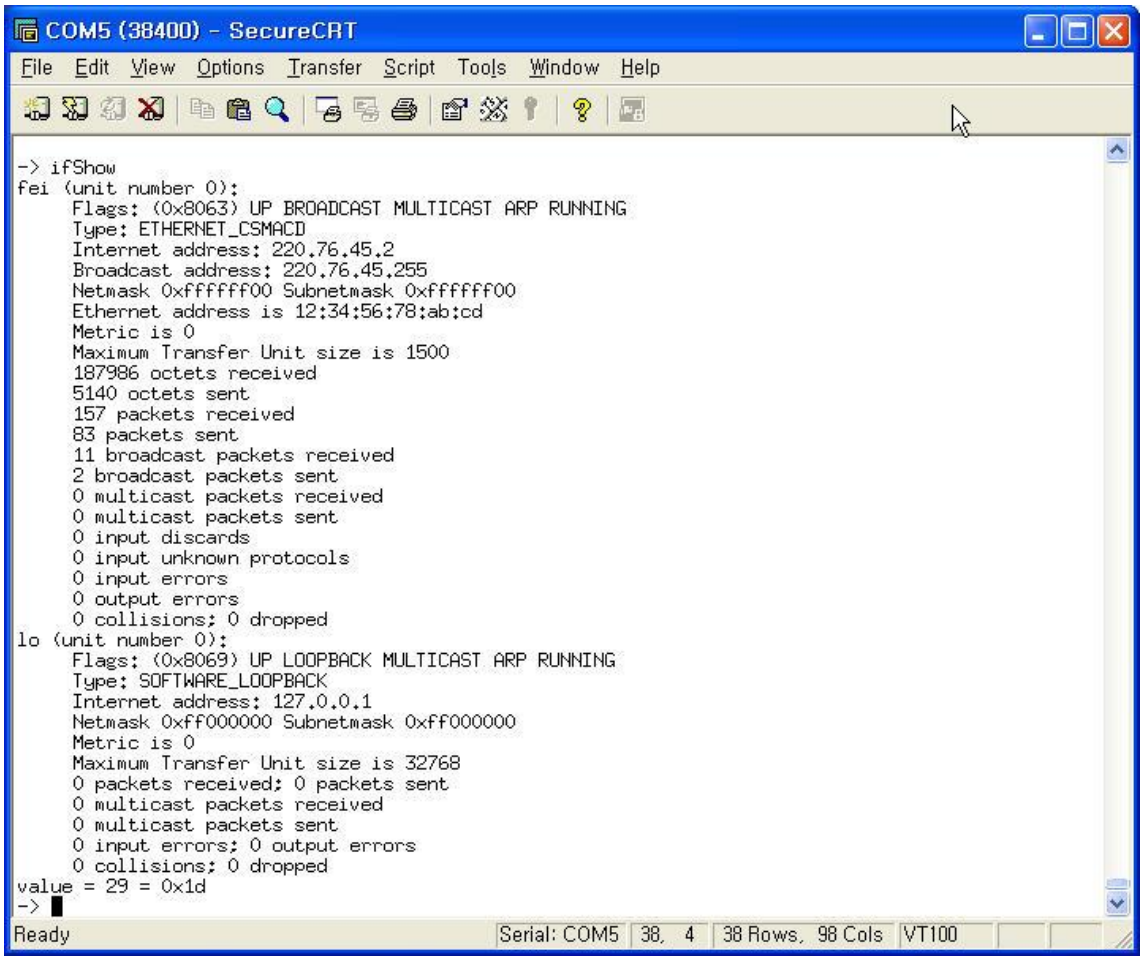

그림 2. Ethernet 1 만 설정 했을 경우의 Boot From Network

2.3. Second Ethernet 을 설정할 경우 IP 입력하는 방법

boot device	: fei
unit number	: 0
processor number	: 0
host name	: 080
file name	: vxWorks
inet on ethernet (e)	: 220.76.45.2
host inet (h)	: 220.76.45.200
user (u)	: 080
ftp password (pw)	: 080
flags (f)	: 0x8
other (o)	:

LAN2 의 IP Setting 방법은 프롬프트 상에서 EnetAttach 함수를 사용한다.

Ex)->EnetAttach(1,2,3)
1 : Ethernet device 의 번호를 입력한다.
2 : IP Address 를 입력한다.
3 : subnet mask 를 입력한다.

Ex)->EnetAttach (1,"220.76.46.2",0xffffff00)

Booting 이 된 후 Dos 상에서 다음과 같은 명령을 실행한다.

Ex) ping –t 220. 76. 45. 2–l 2000

console 창에서 다음과 같이 입력하여 확인을 한다.

Ex) ->ifShow

그림 3. Ethernet 2 사용시 설정된 화면

[그림 2-3] 에서 보는 바와 같이 unit 0 에는 Internet Address 가 220.76.45.2 로 셋팅 되어 있으며, unit 1 은 Internet Address 가 220.76.46.2 로 설정된 것을 확인할 수 있다.

2.4. Second Ethernet 을 이용하여 부팅하는 경우

첫번째 Ethernet 포트가 불량이거나 어떠한 이유로 인하여 두번째 Ethernet 포트로 부팅해야 할 경우에는 boot device:fei1 로입력하고 나머지 값들은 그대로 두고 부팅을 시도하면 되다.

```
boot device          : fei
unit number          : 1
processor number     : 0
host name            : 080
file name            : vxWorks
inet on ethernet (e) : 220.76.45.2
host inet (h)        : 220.76.45.200
user (u)             : 080
ftp password (pw)    : 080
flags (f)            : 0x8
other (o)            :
```

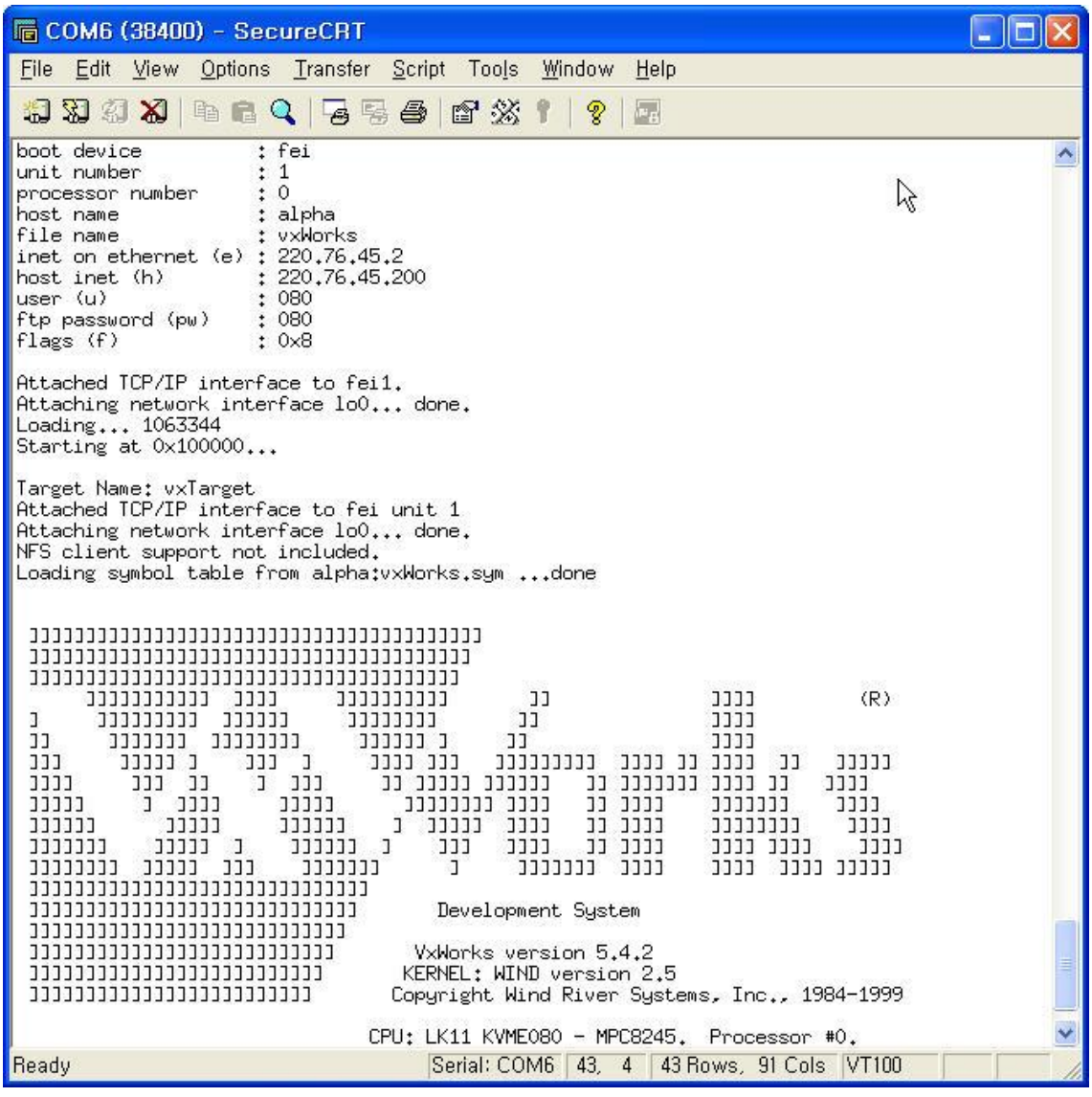

그림 4. Ethernet 2 사용한 부팅

2.5. PMC Ethernet Card 를 사용할 경우 IP 입력 방법

boot device : fei
unit number : 0
processor number : 0
host name : 080
file name : vxWorks
inet on ethernet (e) : 220.76.45.2
host inet (h) : 220.76.45.200
user (u) : 080
ftp password (pw) : 080
flags (f) : 0x0
other (o) :

PMC Ethernet Card 의 IP Setting 방법은 Board 에서 Lan2 를 setting 하는 것과 같다. 주의할 점은 device number 가 2,3 번으로 인식되는 것이다. 다음은 PMC Ethernet Card 의 IP setting 방법이다.

Ex)->EnetAttach(2,"xxx,xxx,xxx,xxx",0xffffff00) : PMC Ethernet 1
Ex)->EnetAttach(3,"xxx,xxx,xxx,xxx",0xffffff00) : PMC Ethernet 2

IP 를 입력한 후에 다음과 같이 입력하여 확인을 한다.

Ex)->ifShow

그림 5. PMC Ethernet Card 를 사용했을 경우 Boot From Network

3. LKV-080A Control Method

3.1. LKV-080A B/D Memory Map

Memory range	Size(Byte)	Description
0x00000000 ~ 0x07FFFFFF	128M	SDRAM
0x7C000000 ~ 0x7CFFFFFF	16M	Flash Memory
0x80000000 ~ 0xFDFFFFFF		PCI Memory Space
0xFE000000 ~ 0xFE00FFFF	8M	PCI I/O Space
0xFE800000 ~ 0xFEBFFFFF	4M	PCI I/O Spzce
0xFEC00000 ~ 0xFEDFFFFF		PCI configuration address register
0xFEE00000 ~ 0xFEFFFFFF		PCI configuration data register
0xFF000000 ~ 0xFF07FFFF	512K	NVRAM
0xFF080000 ~ 0xFF08001F		UART(16C554)
0xFF080120 ~ 0xFF080120	1	DIP LED, Watch dog(8-bit)
0xFFF00000 ~ 0xFFFFFFFF	1M	EPROM(8-bit)

표 1. LKV-080A Memory Map

3.1.1. NVRAM

(1) NVRAM 의 기능

NVRAM 은 Boot Parameter 의 저장을 주 목적으로 한다. 비휘발성 메모리로 Rom 은 아니지만 내부에 battery 가 내장되어 있어 특정 data 를 저장하기 위한 용도로 사용 가능하다. 또한 RTC 기능으로 사용 되며, NVRAM 에 설정한 시간 값에 의해 정확한 시간을 알아 낼 수 있다. 현재 LKV-080A B/D 는 Dallas 사의 DS9034 를 사용하며, 메모리는 512Kbyte 이다. NVRAM 을 Access 할 경우 MPC8245 의 특성으로 인하여 8bit 단위로 Access 되며, Read 시에는 16bit 나 32bit 단위로 Access 가능하다. NVRAM 의 Access Range 는 0xFF000300 ~ 0xFF07ff00 까지 Access 가능하다. 현재 0xFF000200 부터 0xFF000300 까지는 Boot Parameter 의 정보가 저장되어 있다.

(2) NVRAM Control Function

- STATUS date(str)
  RTC 의 시간정보를 설정하는 함수이다. 설정 값의 순서는 YY MM DD hh mm ss 의 순서로 설정된다.

  Ex) date("051001114800")

  YY : Year
  MM : Month
  DD : Day
  hh : Hour
  mm : Minutes
  ss : Second

  Ex) date
  WED OUG 01 11:48:00 2005

ADDRESS	DATA								FUNCTION	
	B7	B6	B5	B4	B3	B2	B1	B0		
7FFFF	-	-	-	-	-	-	-	-	YEAR	00-99
7FFFE	0	0	0	-	-	-	-	-	MONTH	01-12
7FFFD	0	0	-	-	-	-	-	-	DATE	01-31
7FFFC	0	FT	0	0	0	-	-	-	DAY	01-07
7FFFB	KS	0	-	-	-	-	-	-	HOUR	00-23
7FFFA	0	-	-	-	-	-	-	-	MINUTES	00-59
7FFF9	ST	-	-	-	-	-	-	-	SECONDS	00-59
7FFF8	W	R	S	-	-	-	-	-	CONTROL	A

ST : Stop Bit R : Read Bit FT : Frequency Test
W : Write Bit S : Sign Bit KS : Kick Start

표[3-1] DS9034(RTC) Register Map

(3) NVRAM Read/Write Function Control Method

NVRAM 은 하드웨어 특성상 Write 시 Byte 단위로만 Access 된다. 따라서 NvRam 의 데이터를 접근할 때에는 vxWorks 에서 제공하는 bcopyBytes 를 사용하여 접근하면 된다.

```
void bcopyBytes(
        char * source, /* pointer to source buffer */
        char * destination, /* pointer to destination buffer */
        int nbytes /* number of bytes to copy */
)
```

3.1.2. Flash Memory

(1) Flash Memory 의 기능

LKV-080A Board 의 Flash Memory 는 16Mbyte 의 용량을 가지고 있으며 flash Memory 를 이용하여 Flash Boot 기능으로 사용 가능 하다.

(2) Flash Memory Control Function

Flash Boot 이미지를 만드는 방법

순서 1) config.h 다음과 같이 정의 한다.

```
#define FLASH_BOOT
```

순서 2) Dos 상에서 make 를 실행한다.

```
make vxWorks.st_rom.hex
```

순서 3) 다음과 같이 실행 해서 Binary 파일을 만든다.

```
ElfToBin < vxWorks.st_rom > vxWorks.bin
```

순서 4) 생성된 Binary 파일을 Target Board 에서 Load 할 수 있는 위치로 옮긴다.

순서 5) Console 상에서 "ls" 명령 시 다음과 같이 Binary 파일이 보이는지 확인한다.

```
Ex) -> ls
        vxWroks.bin
```

순서 6) makeFlashBoot 함수를 이용하여 Binary 파일을 Flash Memory 에 기록한다.

```
Ex) makeFlashBoot("vxWorks.bin")

-> makeFlashBoot("vxWorks.bin")
Start binary file to Flash
Load File Name : vxWorks.bin - size : 520420 byte
Open & read binary file OK
sysFlashSet => offset : 0x0, strLen : 0x7f0e4, flash_size : 0x1000000
while end
flashBlockerase : s_num :0, e_num : 2
Flash Block 1 Erase Complete !!
Flash Block Erase End
flashBockErase OK
Flash Write 520420 byte OK!!
value = 0 = 0x0
->
```

순서 7) Flash Boot 방법은 Dip switch 의 첫 번째 bit 를 off 로 setting 후 Booting 을 하면 된다. Booting Parameter 는 동일하다.

(3) Flash Memory R/W Fucntion

1) Flash Erase Function

-- STATUS sysFlashEraseSector (int s_start, int s_end)

플래시 메모리의 섹터를 지우는 함수로 s_start 는 처음 섹터, s_end 는 마지막 섹터이다.

-- STATUS sysFlashErase (INT8 flashType)

플래시 메모리 전체를 지우는 함수 이다. 플래시 메모리는 16Mbyte 이며, 128 개의 섹터를 가지고 있다. 각 섹터의 크기는 128K 바이트이다. flashType 은 0 으로 한다.

2) flash Write Function

-- STATUS sysFlashSet(char * string, int strLen, int offset)
플래시 메모리에 바이트 단위로 쓰는 함수이다. string 은 쓸 데이터의 포인터, strLen 은 크기, offset 은 플래시 메모리 오프셋이다.

3.2. The contents of a LKV-080A B/D IO Control

3.2.1. Watch dog time out reset control

(1) Watchdog 타이머 기능
Watchdog 타이머를 이용하면 보드에서 특정 프로그램이 제대로 기능하지 않을 때 보드를 Reset 시킬 수 있다.

(2) Watchdog 타이머 제어 함수

1) void wdtEnable ()
Watchdog 타이머를 활성화 시킨다.

2) STATUS wdtClear ()
Watchdog 타이머를 Clear 시킨다. 이 함수를 호출한 시점에서 1.6 초 동안 Watchdog 타이머에 의한 Reset 이 걸리지 않는다. 1.6 초 이내에 이 함수를 다시 호출 하면 다시 그 시점에서부터 1.6 초 동안 Watchdog 타이머에 의한 Reset 이 걸리지 않는다.

3) STATUS wdtDisable ()
Watchdog 타이머를 비활성화 시킨다.

3.2.2. Front Panel Led Control Method

1) void ledOn(int led)
LED 를 On 시킨다. 여기서 led 는 DIAG0 LED 가 0, DIAG1 LED 가 1, RUN LED 가 2, FAIL LED 가 3 이다.

2) void ledOff(int led)
LED 를 Off 시킨다. 여기서 led 는 DIAG0 LED 가 0, DIAG1 LED 가 1, RUN LED 가 2, FAIL LED 가 3 이다.

3.2.3. Dip Switch Read method

(1) Function Description
LKV-080A Board 는 1 개의 Dip Switch 가 있으며, Switch On 시 data 는 0 으로 Read 되며, Switch Off 시 data 는 1 로 read 된다.

(2) Dip Switch Read Function

1) unsigned char dswGet ()
Dip Switch Read 시 char 값이 return 되며, 값의 범위는 0x00 ~ 0x0F 이다.

4. LKV-080A Serial Control method

4.1. LKV-080A B/D Serial Driver Initialization

LKV-080A B/D 는 2 개의 Ethernet Port 와 4 개의 Serial Port 가 장착되어 있으며, Serial Device 로는 XR16L784 를 사용한다. 또한 Baud rate generation 을 위한 기본 clock 으로는 14.7456MHz 가 공급되며, 4 개의 Serial Port 중 B/D Monitor 용 Console Port 는 COM1 이 사용되고. 나머지 Port 는 점퍼 setting 에 따라(LKV-080A H/W Manual 참조) Serial Port 는 RS-232 나 RS485 통신을 사용자가 선택하여 사용할 수 있는 Sio Port 이다.

4.2. vxWorks IO System

LKV-080A B/D 는 IO System 을 통해서 create(), remove() open(), close(), read(), write(), ioctl() 함수를 지원하며 vxWorks 에서 일반적으로 지원하는 기능과 같다.

(1) open ()
1) Synopsis
int open (const char * name, int flag, int mode)
2) Description
ⓐ name
현재 생성되어야 될 프로세서의 드라이버를 가리킨다. 드라이버의 이름은 각각의 프로세서마다 다르며 파일단위로 생성됨. Name 을 통해서 드라이버에 접근 가능함.
ⓑ flag
O_RDONLY(0) : 드라이버 읽기 전용으로 생성
O_WRONLY(1) : 드라이버 쓰기 전용으로 생성
O_RDWR(2) : 드라이버 읽기/쓰기로 생성
O_CREATE(0x0200) : 드라이버를 파일단위로 생성
ⓒ mode
UNIX 시스템에서 파일접근 허가를 나타내는 숫자를 수록함.
3) Return Value
ⓐ number
파일단위로 생성된 디바이스의 개수를 나타냄
ⓑ ERROR
해당 디바이스가 없거나 파일네임과 맞지 않을 경우를 나타냄

(2) close ()
1) Synopsis
STATUS close (int fd)
2) Description
ⓐ fd
시스템으로부터 드라이버 파일의 사용이 끝났음을 알림.
3) Return Value
ⓐ OK
호출이 성공적으로 이루어 졌을 경우를 나타냄
ⓑ ERROR
해당 드라이버가 없을 경우 또는 파일 기술자가 아닐 경우.

(3) read ()
1) Synopsis
int read (int fd, char * buffer, size_t maxbytes)
2) Description
개방된 driver file 로 부터 일정 수의 byte 를 buffer 로 복사하기 위해 사용됨.

ⓐ fd
buffer 로 읽어 들일 driver 의 file 명
ⓑ buffer
읽어 들일 file 의 저장 공간으로 char type 의 pointer 로 정식 선언되며 1 문자와 1byte 는 구별 없이 사용할 수 있다. 즉 buffer 는 자료가 복사될 문자 배열에 대한 pointer 임.
ⓒ maxbytes
file 로 부터 읽혀질 바이트의 수를 나타내는 양의 정수임.

3) Return Value
ⓐ number
file 단위로 생성된 driver 에서 buffer 에 저장된 byte 의 개수를 나타냄
ⓑ ERROR
해당 device 가 없거나 file name 과 맞지 않을 경우를 나타냄

(4) write ()
1) Synopsis
int write (int fd, char * buffer, size_t nbytes)
2) Description
write 의 호출은 read 와는 반대로 문자배열로 선언된 program buffer 로 부터 개방된 file driver 를 통해 일정 수의 byte 를 출력하거나 쓰기 위해 사용됨.
ⓐ fd
출력하거나 쓰기 위한 driver 의 file 명
ⓑ buffer
문자배열로 선언된 program buffer 로 char type 의 pointer 로 정식 선언되며 1 문자와 1byte 는 구별 없이 사용할 수 있다. 즉 buffer 는 출력될 자료의 문자배열에 대한 pointer 임.
ⓒ nbytes
파일로부터 출력되어야 할 바이트의 수를 나타내는 양의 정수임.
3) Return Value
ⓐ number
program buffer 로부터 device 에 쓰여진 byte 의 개수를 나타냄
ⓑ ERROR
해당 device 가 없거나 file name 과 맞지 않을 경우를 나타냄

(5) ioctl ()
1) Synopsis
int ioctl (int fd, int function, int arg)
2) Description
Device 의 I/O control 함수로 매우 유용하게 쓰인다.
ⓐ fd
control 할 device 의 terminal 명
ⓑ function
I/O control 함수는 deivce driver 에 따라 각 함수에 대한 옵션이 정해져 있으며 제공되는 드라이버의 옵션은 다음과 같다.
- FIOBAUDRATE
  Baud Rate 을 Set 한다.
- FIOGETOPTIONS
  각 채널에 대한 옵션을 얻어옴
- FIOSETOPTIONS
  각 채널에 대한 옵션을 setting 함
- FIOCANCEL
  각 채널에 대한 read/write 요청을 거부함
- FIOFLUSH

RX Buffer clear
- FlOWFLUSH
TX Buffer clear

4.3. RS-485 통신 사용시 셋팅 방법(Port 2,3,4)

LKV-080A 의 시리얼 포트 사용시 RS-232 방식과 RS485 방식을 제공한다. RS-232/RS-485 방식으로 사용시 하드웨어 매뉴얼을 참조하여 점퍼를 설정하는데, RS-485 방식을 사용할 경우에는 SW 에서 추가적으로 레지스터를 설정해 주어야 한다. 이를 위하여

(1) RtsSet ()
1) Synopsis
Int RtsSet (int ch, int set)
2) Description
RS-485 통신에 사용하는 RTS 시그널을 제어한다.
ⓐ ch
통신을 수행할 시리얼 포트 번호 (1~3). 0번 포트는 주로 콘솔 용도로 사용한다. 1번을 RS-485 통신에 사용하려면, BSP의 sysSerial.c를 수정해야 한다.
ⓑ set
0 : RTS Disable
1 : RTS Enable
3) Return Value
입력 포트 번호가 잘못된 경우에 ERROR 리턴

4) 사용법
송신시 : RTS enable후에 Write, Write 완료 후에는 반드시 RTS disable할 것.
수신시 : RTS disable후에 Read.

이외에 포트를 open/write/read 하는 것은 RS-232 통신 방식과 동일하다.

♣ VxWorks Programmer's Guide 3 장 참조

5. LKV-080A B/D VMEBus Control method

5.1. LKV-080A B/D VME Memory Map

NAME	Address	Size(Byte)	Access	Description
VME_A32_MSTR_BUS	0x00000000 ~ 0xEFFFFFFF	3.84G	R/W	VME A32 Master Access
VME_A23_MSTR_BUS	0xF0000000 ~ 0xF0FFFFFF	16M	R/W	VME A24 Master Access
VME_A16_MSTR_BUS	0xFF000000 ~ 0xFF00FFFF	64K	R/W	VME A16 Master Access
VME_IACK	0xFF100000 ~ 0xFF10001f	32	R/W	VME IACK
VME_REG	0xFFF00000 ~ 0xFFF0003F	64	R/W	VME Configuration Register

표[5-1] VME Bus Memory Map

5.2. LKV-080A B/D VME Bus Access

	VME_A32	VME_A24	VME_A16
PCI Bus Address	0x90000000	0x82000000	0x81000000
VME Bus Address	0x10000000	0xF0000000	0xFF000000

표[5-2] PCI Bus Address -> VME Bus Address 변환 쌍

5.2.1. Extended Mode 일 경우 VMEBus Access

Address 0x00000000 ~ 0xEFFFFFFF 는 VME32 Master Access 영역이다. 이 영역 안에 들어오는 데이터 전송은 VME A32 규격에 맞게 변형되어 VME 버스로 출력 된다. PCI 버스에서 VME 버스에 A32 모드로 접근하기 위해서는 P_BA, P_MA, P_TA 레지스터를 설정하고, P_IMG_CTRL 레지스터에서 변환 가능 비트를 1 로 설정해야 된다.

5.2.2. Standard Mode 일 경우 VMEBus Access

Address 0x00000000 ~ 0xF0FFFFFF 는 VME24 Master Access 영역이다. 이 영역 안에 들어오는 데이터 전송은 VME A24 규격에 맞게 변형되어 VME 버스로 출력 된다. PCI 버스에서 VME 버스에 A24 모드로 접근하기 위해서는 P_BA, P_MA, P_TA 레지스터를 설정하고, P_IMG_CTRL 레지스터에서 변환 가능 비트를 1 로 설정해야 된다.

5.2.3. Short Mode 일 경우 VMEBus Access

Address 0x00000000 ~ 0xFF00FFFF 는 VME32 Master Access 영역이다. 이 영역 안에 들어오는 데이터 전송은 VME A16 규격에 맞게 변형되어 VME 버스로 출력 된다. PCI 버스에서 VME 버스에 A16 모드로 접근하기 위해서는 P_BA, P_MA, P_TA 레지스터를 설정하고, P_IMG_CTRL 레지스터에서 변환 가능 비트를 1 로 설정해야 된다.

5.3. LKV-080A VME Bus Access Example

5.3.1. Extended Mode Access Example

```
void Test_VME32(unsigned int address, unsigned int size, unsigned int l)
{
 unsigned int i, j, vme32_size;
 unsigned long *vme32_addr;
 unsigned int r_buf[size];

 vme32_addr = (unsigned long*)address;
 vme32_size = size/4;

 printf("LKV-080A VME32 test program!! \ n");

 for(j=0; j<l; j++)
 {

        for(i=0; i<vme32_size; i++, vme32_addr++)
        {
                *(unsigned int *)vme32_addr = i;
                                }

        vme32_addr = (unsigned int *)address;

        for(i=0; i<vme32_size; i++, vme32_addr++)
        {
                r_buf[i] = *vme32_addr;
        }
 }
 return OK;
}
```

위 예제에서 Access 가능 어드레스는 Extended Mode 이며 Start Address 를 0x98000000 으로 설정하면 End Address 는 0x98050000 이 된다. Byte 는 4byte 를 한번에 Access 하며, End Address 까지 Write 를 한 후 Start Address 부터 End Address 까지 값을 읽게 된다.

5.3.2. Standard Mode Access Example

```
void Test_VME24(unsigned int address, unsigned int size, unsigned int l)
{
 unsigned int i, j, vme24_size;
 unsigned long *vme24_addr;
 unsigned int r_buf[size];

 vme24_addr = (unsigned short *)address;
 vme24_size = size/2;

 printf("LKV-080A VME24 test program!! \ n");

 for(j=0; j<l; j++)
 {
```

```
        for(i=0; i<vme24_size; i++, vme24_addr++)
        {
                *(unsigned short *)vme24_addr = i;
                                }

        vme24_addr = (unsigned short *)address;

        for(i=0; i<vme24_size; i++, vme24_addr++)
        {
                r_buf[i] = *vme24_addr;
        }
 }
 return OK;
}
```

위 예제에서 Access 가능 어드레스는 Standard Mode 이며 Start Address 를 0x82000000 으로 설정하면 End Address 는 0x82050000 이 된다. Byte 는 2byte 를 한번에 Access 하며, End Address 까지 Write 를 한 후 Start Address 부터 End Address 까지 값을 읽게 된다.

5.4. LKV-080A B/D AM Code Setting Function

VME 버스에는 어드레스 라인 이외에 어드레스 모디 파이어(Address Modifier)라고 불리 우는 신호선이 6 개(AM5~AM0) 존재하며, 마스터보드에서 슬레이브 보드로 AM 코드라고 하는 일정 정보를 전송한다. 또한 표[5-2]에 보는 바와 같이 AM Code 에 따라서 VME 버스를 여러 개의 시스템으로 분리하여 사용할 수 있다.

AM Code							Function
HEX CODE	5	4	3	2	1	0	
09	0	0	1	0	0	1	A32 non privileged data access
0A	0	0	1	0	1	0	A32 non privileged program access
0D	0	0	1	1	0	1	A32 supervisory data access
0E	0	0	1	1	1	0	A32 supervisory program access
39	1	1	1	0	0	1	A24 non privileged data access
3A	1	1	1	0	1	0	A24 non privileged program access
3D	1	1	1	1	0	1	A24 supervisory data access
3E	1	1	1	1	1	0	A24 supervisory program access
29	1	0	1	0	0	1	A16 non privileged access
2D							A16 supervisory access

표[5-2] VME Bus AM Code 일람표

여러 개의 슬레이브 보드가 존재할 때 슬레이브 보드에서 응답하는 AM 코드를 바꾸어 두면 마스터측은 메모리 뱅크를 전환하도록 AM 코드를 변화시켜 메모리 맵핑을 할 수 있다.

5.4.1. VME AM Code Setting Function

- int multivAmCodeSet(int mode, unsigned char AMCode_data)

Mode : mode 는 Slave AM Code 나 Master AM Code 를 결정하며, 또한 값에 따라 Address 라인이 Short Address, Standard Address, Extended Address 로 나뉜다.

Mode Value :

0x01 → VME32 종속 AM Code Setting
0x02 → VME24 종속 AM Code Setting
0x03 → VME16 종속 AM Code Setting
0x04 → VME32 마스터 AM Code Setting
0x05 → VME24 마스터 AM Code Setting
0x06 → VME16 마스터 AM Code Setting

Ex) multivAmCodeSet(0x01, 0x0D);

예제와 같이 함수 수행 시 setting 결과는 multivVmeRegShow() 통해서 확인 가능하며, 위의 함수를 통해서 다음과 같은 메시지가 출력되면 성공적으로 setting 된 것이다. Mode Value 가 위의 값과 같지 않을 경우는 기본값으로 VME32 는 0x0D, VME24 는 0x3D 로 VME16 은 0x2D 로 setting 된다.

Result) Extended(Slave) AM Code Setting Value = 0x0D