



---

## User's Manual



**LINUX Co., Ltd.**

## LOC에 대한 이해

최근 내장형 제품으로의 인터넷 기술 및 Embedded Linux의 적용이 최대 관심사가 되었고, 각 기업들은 관련 기술을 어떻게 적용할 수 있을 것인가에 큰 관심을 가지고 있습니다.

인터넷 기술 및 Embedded Linux의 적용은 크게 다음과 같은 목적을 가지고 있다고 판단됩니다.

1. 제품의 동작이나 기능 구현이란 목표에서 제품의 정보화, 정보 전달의 유연성, 수평적 정보 전달 및 정보 관리등으로 기술적 관점이 변하고 있다고 생각됩니다. 또한 정보의 관리나 활용이 여러 계층에서 요구되고, 다양한 범위에서 활용됩니다. 이에 따라 계층간 정보 전달이 유연하고, 각 계층을 하나로 통합할 수 있는 일반적이고 성공적인 네트워크 구성이 요구되고 있습니다.

현재로서 가장 적절한 네트워크는 인터넷 망이라고 판단되며, 이를 기반으로 한 다양한 기술의 적용이 가능하기 때문에 최근의 기술적 관심이 모아지고 있다고 판단됩니다.

2. 상용 운영 체제를 이용한 제품의 개발은, 고비용과 기술의 폐쇄성 때문에 다양한 발전이 어렵습니다. 따라서 공개되어 있고, 구성이 표준화되어 있는 Linux의 응용이 큰 관심사가 되고 있습니다. Linux는 강력한 기능에 비해 작은 용량으로도 내장형 제품에 적용 가능한 형태로 기술이 발전되고 있으므로, 이를 내장형 제품에 적용하고자 하는 요구가 늘어나고 있습니다.

Linux는 TCP/IP를 포함하고 있으므로, 인터넷 연동에 대한 확실한 해답이 될 수 있습니다. TCP/IP가 Unix를 기반으로 개발/발전되었고, Linux는 Unix를 기반으로 발전된 운영 체제 이므로, 기타 인터넷 연동 솔루션에 비해 가장 발전된 해답을 제시할 수 있습니다.

Linux가 비록 공개되어 있지만, 실제 내장형 제품에 적용하기에는 많은 기술적 문제점이 존재하고, 이를 해결할 수 있는 전문 인력이나 기술도 다양하지 않기 때문에 손쉽게 적용되지 못하고 있는 것이 현실입니다. 대부분의 Embedded Linux에 대한 솔루션은 PC architecture를 기반으로 한 제품이 대부분인 까닭에 내장형 제품으로 구현하기에는 비용이나 크기면에서 쉽게 적용하기 어려운 것이 사실입니다.

이에 (주) 리누덱스에서는 Linux가 탑재된 CPU chip을 공급하여 Embedded Linux가 필요한 기업에서 손쉽게 기능을 구현할 수 있도록 하고자 합니다. LOC (LinuxOnChip)는 최소 크기, 최소 비용으로 기능 구현이 가능합니다. 이를 채택한 기업에서는 LOC를 기반으로 하여, 응용 H/W 및 S/W만을 개발하는 것만으로 필요한 기능을 모두 구비한 제품을 손쉽게 개발할 수 있습니다. LOC는 인터넷 연동 제품, Embedded Linux에 대한 가장 손쉬운 해답입니다.

## 제품 구성

- LOC
  - . 900mil DIP 형태의 CPU chip
  - . CPU, ROM, RAM, Ethernet controller가 내장된 Linux chip
  
- 개발 환경 구축용 CD
  - . GCC m68k cross compiler
  - . GDB debugger
  - . 예제
  
- Evaluation board
  - . LOC 기능 시험을 위한 기본적인 응용 H/W가 내장된 형태의 평가용 보드

## 연락처

- E-mail : linudix@LinuxOnChip.com
- Telephone : 02-3482-0991~2
- FAX : 02-3482-0993

## LOC 사용을 위한 기초 사항

- Linux 응용 프로그래밍이 가능하기 위해서는 Linux에 대한 기본 지식을 가지고 있어야 합니다. 이는 어떠한 운영 체제를 기반으로 한 응용 프로그램 구현에 있어서도 반드시 필요한 사항입니다.
- C-language에 익숙하여야 합니다. Linux 자체가 C를 기반으로 하고 있으며, 제공되는 개발 tool 역시 C-compiler 입니다. 대부분의 embedded system 개발자가 이 언어에 익숙할 것으로 판단됩니다.
- 인터넷 연동을 위한 프로그래밍이나, 홈페이지 구축 등을 하기 위해서는 해당 기술에 대한 지식이 필요합니다. (주) 리누덱스에서는 이에 대한 기술 지원을 하기 어렵습니다. 모든 사용자에게 대한 응답을 충분히 해 드릴 수 없기 때문입니다. 이점에 대해서 충분한 양해를 구합니다. 필요한 부분에 대해서는 웹상에서 많은 예제를 구할 수 있고, 관련 기술 서적을 통해 쉽게 얻을 수 있습니다.
- 기타 기술적인 질문이나 의문 사항들은 홈페이지의 "질문과 대답"을 통해서 응답하고자 합니다.

## LinuxOnChip 특징

- Embedded 제품에 Linux 적용을 위한 최적의 solution
- Internet 정보 가전, 산업용 제어기 등 인터넷 접속이 필요한 전기/전자 제품에 적용
- Real time kernel 구현(Interrupt driven control 가능)
- TCP/IP protocol stack
- Flash ROM에 구현된 file system
- NFS(Network File System)를 통한 개발 program의 저장 및 실행 가능
- On line(target board) debugging이 구현
- LAN controller 내장
- LCD controller 내장- Web server 내장
- PPP 접속 기능

## H/W 설명

구성품	설명
CPU	Motorola MC68EZ328
ROM	2MB, 4MB flash memory
RAM	8MB DRAM
Ethernet port	10base-T
Interrupt pin	2 external pin
etc	2 external chip select LCD controller
Type	900mil width, 2.54mm pitch,50pin DIP
Power	3.3V
Serial port	1 UART 1 SPI

## CPU

Motorola의 68000 계열 MC68EZ328 CPU가 사용되었습니다. 일명 Dragon ball이라고 불리는 이 CPU는 각종 주변장치를 내장한 고집적 CPU로서 내장형 시스템에 적용하기에 알맞은 구조를 가지고 있으며, 전력 소모도 아주 작습니다. LOC의 중앙 처리 장치로서 내장형 시스템에 적용하기에 충분한 퍼포먼스를 가지고 있습니다. LOC를 이용하여 응용 프로그램이나 하드웨어를 구성할 경우에는 Motorola site를 방문하여 사용설명서를 다운로드 받아 참고 하시기 바랍니다.

\* 참고로 Strong ARM 등 고속 CPU 제품을 곧 출시할 계획입니다.

## Flash ROM

본 제품의 Linux kernel이 저장되는 곳입니다. Kernel은 출하시 Flash ROM 에 탑재되어 제공됩니다. Flash ROM은 2Mbytes 제품과 4Mbytes 제품으로 나뉘어 집니다.

Kernel은 약 1Mbytes를 점유하므로, 사용자 영역은 각각 1Mbytes, 3Mbytes 입니다. LOC에는 flash file system이 설치되어 있으므로, 사용자 프로그램을 사용자 영역에 로드할 수 있습니다.

LOC에는 3개의 파티션이 기본으로 마운트되어 있습니다.

/ (root) : 루트 파티션입니다. Kernel 및 LOC의 기본 프로그램과 환경설정 파일들을 담는 용도로 사용됩니다.

/var (Ram disk) : 임시저장용도로 사용할수 있는 휘발성 램디스크 입니다.

/usr : 사용자 파티션입니다. 사용자 응용 프로그램과 설정 파일등을 설치할 수 있는 영역 입니다.

각 파티션의 크기는 flash ROM의 용량에 따라 아래와 같습니다. (아래의 용량은 포맷 후의 사용할수 있는 실제 용량입니다.)

2M bytes flash ROM

/ : 428KB

/usr : 620KB

/var : 500KB

4M bytes flash ROM / : 428KB

/usr : 2660KB

/var : 500KB

Note: 램디스크의 용량은 필요에 따라 128K, 512K, 1M, 2M, 4M 로 변경이 가능합니다.

Flash file로의 저장 방법은 "부록 : Flash file"을 참조하시기 바랍니다.

## DRAM

LOC가 부팅된 이후 Flash ROM에 탑재된 kernel 이미지가 복사되어 운전되는 영역입니다. 총 8Mbytes로 구성되어 있으며, kernel 및 allocated variable 등 시스템의 기본 메모리 영역으로 사용됩니다.

## Ethernet controller

10base-T ethernet controller가 내장되어 있습니다. 사용자는 외부에 transformer와 connector만 실장함으로써 10base-T port를 확보할 수 있습니다. 실장 방법 및 권장 사항은 "부록 : Evaluation board" 를 참조하시기 바랍니다.

## Serial port

한개의 UART port를 제공합니다. LOC 자체에는 RS-232C level converter가 장착되어 있지 않습니다. 사용자는 LOC 외부에 level converter chip(MAX232등)과 필요한 형태의 connector(Dsub 9pin)를 장착하여 RS-232C port를 확보할 수 있습니다. 실장 방법은 "부록 : Evaluation board' 를 참조하시기 바랍니다. 내장된 serial port는 개발을 위한 console port로도 사용됩니다.

## SPI port(Serial Periperal port)

직렬 버스를 통하여 각종 주변 chip을 연결할 수 있는 port 입니다. SPI는 3개의 신호선으로 CPU와 주변 장치를 interface 할 수 있는 버스입니다. A/D, D/A, DSP, UART등을 접속하여 사용할 수 있습니다. SPI port로 사용하지 않을 경우에는 general I/O port로 사용가능합니다. 자세한 설명은 MC68EZ328 사용설명서를 참조하시기 바랍니다.

## Interrupt port

2개의 interrupt pin을 제공합니다. Interrupt 우선 순위 결정등을 위해서는 MC68EZ328 사용설명서를 참조하시기 바랍니다. Interrupt pin으로 사용하지 않을 경우에는 general I/O port로 사용가능합니다. CPU의 pin 번호와 LOC의 pin 번호간의 관계는 "부록 : pin 배치" 를 참조하시기 바랍니다.

## External chip select pin

LOC 외부에 memory device를 장착하고자 하는 사용자를 위하여 2개의 chip select pin 이 제공됩니다. 사용자는 별도의 address decode 회로를 사용하지 않고, 2개의 chip select 신호를 사용할 수 있습니다. CPU의 pin 번호와 LOC의 pin 번호간의 관계는 "부록 : pin 배치"를 참조하시기 바랍니다.

## External memory bus

총10개의 address line이 제공되므로, chip select당 512bytes의 외부 메모리 access가 가능합니다. 대부분의 내장형 제품 회로 설계에 있어서 큰 address 공간은 sytem ROM과 RAM을 위해 사용되고, 실제 주변 periperal은 많은 address 공간을 차지 않습니다. LOC는 자체에 큰 용량의 ROM과 RAM을 가지고 있으므로, 외부에는 그렇게 큰 address 공간이 필요하지 않습니다. 외부로의 data bus는 8bit를 지원합니다. 이는 LOC의 형상에 따른 pin 수의 제약에 의한 것이지만, 주변 회로의 특성을 고려하여 8bit만으로도 외부 회로 이용에 충분하다는 점이 고려 되었습니다.

CPU의 data bus중 상위 8bit(D8 ~ D15)가 외부로 제공되므로, 사용자 프로그램 개발시에는 "부록 : Data bus 사용시 주의사항"을 참고하여 주시기 바랍니다.

\* 참고로 MC68EZ328이 제공하는 모든 address 공간과 16 bit 외부 data bus를 구현한 형태의 제품의 출시를 계획하고 있습니다.

## LCD controller & 8 general I/O port

8개의 general I/O port가 제공됩니다. 이 port는 내장 LCD controller와 기능을 공유합니다. LCD controller를 사용하지 않을 때는 일반 I/O pin으로 사용할 수 있습니다.

흑백일 경우 640\*512, Gray level일 경우 320\*240 크기의 LCD controller가 내장되어 있습니다. 추가 하드웨어 없이 LCD controller의 구현이 가능합니다. 자세한 사항은 MC68EZ328 사용설명서를 참조하시기 바랍니다.

## 3.3V operating voltage

채택한 MC68EZ328을 비롯하여 최근 CPU들은 대부분 3.3V 전원을 채택하고 있으며, 컴팩트하게 설계된 CPU 일수록 3.3V 전원을 채택하는 경우가 더 많습니다. LOC도 3.3V 전원으로 구동되고 외부 Pin 역시 3.3V 로 구성되어 있습니다.

대부분의 사용자들이 5V로 구동되는 응용 회로와의 인터페이스 부분에서 고민할 것으로 생각됩니다. 아래에 3.3V와 5V 회로간의 접속 방법을 간단히 소개합니다.

- 출력 : TTL level로 출력되므로, 수신 5V측에 TTL IC(74LSxx)를 사용하면 됩니다.
- 입력 : 입력 pin은 3.3V 정격이므로, 5V를 직접 입력하지 말아야 합니다. 쉬운 방법은 LVC나 LVT series buffer를 사용하는 것이겠지만, 원가가 문제될때는 voltage divider를 구성하는 것으로 문제를 해결할 수 있을 것입니다.
- 양방향 port : 입력이나 출력만의 해결은 비교적 간단하지만, data bus와 같은 양방향 pin을 3.3V와 5V 전원 간에 접속하기는 조금 복잡합니다. 면적이 허락한다면, 입력과 출력을 분리하여 각각 tri-state buffer를 사용하여 구현이 가능하지만, 회로가 조잡해질 가능성이 있습니다. 이러한 곳에는 LVT245와 같은 부품을 사용하는 것이 적절한 방법이라고 판단됩니다. LVC, LVT 등 3.3V to 5V interface에 관한 제품이나 설명은 Motorola나 TI의 web site에서 쉽게 application note나 제품 설명서를 구할 수 있습니다.

## 개발 환경 구축

### - 개발 장비 요구 조건

LOC를 이용한 S/W 를 개발하기 위한 장비로서 필요한 것은 리눅스가 탑재된 데스크탑 PC 입니다. 대부분의 PC가 기본적으로 CD ROM이나 충분한 메모리등을 내장하고 있으므로 개발 장비로서는 충분하다고 판단됩니다.

제공되는 CD에는

- . kernel uClinux 2.0.38 . GCC m68k(Motorola 68000) compiler tool
- . 68000 계열 지원 source
- . GDB debugging tool
- . 각종 예제

등이 포함 되어 있습니다.

제공되는 CD가 RPM을 기반으로 구성되어 있기 때문에, 개발 장비로 사용될 PC에는 xx86 계열 RPM(Redhat Package Manager)이 설치되어 있어야 합니다. 리눅스가 설치되어 있지 않을 경우에는 레드햇 리눅스 배포판이나 알파 리눅스 배포판을 구하여 설치하시면 됩니다.

### - m68k 개발 환경 구축

사용자는 개발 장비인 PC 상에서 프로그래밍을 하고, compile을 한 후 타겟 보드인 LOC 에 개발 프로그램을 저장하여 원하는 기능을 구현하게 됩니다. 따라서 개발 PC 상에 m68k compiler가 동작할 수 있는 환경을 구축하여야 합니다.

개발 PC 에는 리눅스가 설치되어 있어야 하며 RedHat 5.2 이상이 설치되어 있는 것으로 가정합니다. (최근의 리눅스 배포본을 설치하셨다면 아무 문제가 없을것입니다) 개발 플랫폼에 설치된 리눅스와 개발 툴과의 호환성 은 아래와 같이 확인하시기 바랍니다.

```
# ldd /bin/ls  
libc.so.6 => /lib/lib.so.6 (xxxx)
```

만약 libc.so.5 => /lib/lib.so.5 (xxxx) 와 같이 출력된다면 재설치가 필요합니다.

먼저 제공된 cdrom 을 마운트 하시기 바랍니다.

```
#mount -t iso9660 /dev/cdrom /mnt
```

개발작업을 할 디렉토리를 만드시기 바랍니다. 예에서는 /home/develop 디렉토리를 가정하겠습니다.

```
#mkdir /home/develop
```

기본 컴파일러를 설치 합니다.

```
#cd /mnt/cdrom/RPMS  
#make
```

이제 호스트 PC 에 기본 컴파일러가 설치되었습니다. m68k cross compiler 사용법은 온라인 매뉴얼을 참고 하시기 바랍니다.

```
#man m68k-pic-coff-gcc
```

다음 LOC 를 위한 커널소스를 설치합니다. 리눅스 커널수정이 필요하지 않으신 사용자께서는 이 과정을 건너뛰셔도 됩니다.

```
#cd /home/develop  
#buildenv  
#make
```

이 과정을 마치면 /home/develop 디렉토리에는 linux, romdisk, src 디렉토리가 생성됩니다. Linux 디렉토리에는 uClinux 2.0.38 커널의 소스가 담겨져 있습니다. Src 디렉토리에는 시스템툴들의 소스가 담겨져 있습니다. Romdisk 디렉토리내의 파일들은 현재 LOC 에서는 사용하지 않습니다.

이제 원래의 uClinux 2.0.38 커널을 LOC 의 하드웨어에 맞게 수정하는 작업을 하셔야 합니다. 아래와 같이 실행하시기 바랍니다.

```
#cp /mnt/cdrom/LOC/LOC.diff.gz /tmp  
#cd /tmp  
#gzip -d LOC.diff.gz  
#cd /home/develop/linux  
#patch -p1 < /tmp/LOC.diff
```

(추후 커널 재컴파일하실 때 사용하실 configuration 값은 (make config 시) /mnt/cdrom/LOC/kernel\_config.loc 를 참고 하시기 바랍니다.)

이제 LOC 를 위한 리눅스 소스코드의 설치가 완료 되었습니다.

지금부터 여러분은 실제 LOC 를위한 프로그램의 작성,컴파일,실행을 하실수 있습니다.

## Evaluation board를 통한 개발 환경 시험

구축된 개발 환경을 시험하기 전에 evaluation board에 대한 간단한 설명을 한 후에 시험 예제를 설명하겠습니다.

### LinuxOnChip evaluation board

LOC의 기능을 시험해 볼 수 있으며, 간단한 개발 도구로 사용 가능한 평가용 board 입니다. Terminal emulator program이 장착된 PC의 RS-232C port와 evaluation board의 RS-232C port를 연결하고, 전원을 투입하면 terminal 화면에 LOC의 booting 과정이 표시 되고, Login prompt가 최종 표시됩니다. LOC는 desktop linux와 같은 interface를 제공하면서 linux machine으로서 동작을 시작합니다.

### Evaluation board를 통한 boot

1. 개발용 PC와 evaluation board간을 RS-232C 케이블로 연결합니다(케이블 배선도는 부록Evaluation board를 참조).
2. 개발용 PC에 minicom을 띄우고, 9600bps, no parity, 1 stop bit로 minicom 환경을 설정합니다.
3. 9 ~ 12V 전원을 투입합니다. booting 과정이 표시된 후 Log in prompt가 표시됩니다. Login ID는 root, password는 uClinux로 초기 지정되어 있습니다.
4. 기본적인 Linux 명령이 내장되어 있으므로, 적절히 사용하여 기능을 시험해 보십시오.

### 시험 예제

가장 쉬운 예제인 "Hello! LinuxOnChip."을 console 화면에 print 해보는 예제를 설명하겠습니다.

"m68k 환경 구축"항에서 설명한 순서에 따라, host PC의 원하는 directory 아래에 작업 환경을 구축하면, development라는 directory에 개발 환경이 구축됩니다. 자동 생성된 file 들과의 혼동을 피하기 위해 sample이라는 작업 directory를 만들어 시험용 code를 작성해 보겠습니다.

```
#pwd
/home/xxx/development
#mkdir sample
#cd sample
#buildenv
```

과정이 끝났으면, 아래 프로그램을 coding 하여 저장합니다.

```
main(int argc, char *argv[]){  
    puts("Hello! LinuxOnChip.");  
}
```

coding이 끝났으므로 compile을 수행하여야 합니다. 아래와 같이 compile option을 주어 실행합니다.

```
#m68k-pic-coff-gcc test.c -o test
```

이제 test란 이름으로 LOC에서 실행가능한 file이 만들어졌습니다. 이 file을 LOC로 전송하여 실행시켜야 하는데, file의 전송은 NFS(Network File System)를 이용하여 수행합니다. Host PC의 작업 directory를 LOC에 mount하여, LOC 입장에서 host PC의 특정 directory를 LOC의 local directory인 것 처럼 사용할 수 있도록 설정해주어야 합니다. Host PC의 /etc/exports file을 열어서 내용을 추가합니다. 예제에서는 /home/xxx/development/sample directory를 작업 directory로 가정하였으므로, 아래와 같이 추가합니다.

```
/home/xxx/development/sample (ro,insecure)
```

그다음 Host PC 의 /etc/hosts 파일에 LOC 에 할당한 ip address 를 아래와 같이 추가 합니다. (예에서는 LOC 의 ip address 가 192.168.1.200 라고 가정했고 Host PC의 ip address 는 192.168.1.7 이라고 가정합니다.)

```
192.168.1.200 LOC.linux.private LOC
```

NFS 설정 방법은 대부분의 Linux 설명서등에 자세히 설명되어 있습니다. /etc/exports file 을 수정하였으면,

```
#/etc/rc.d/init.d/nfsserver stop  
#/etc/rc.d/init.d/nfsserver start
```

를 수행하여, 수정내용이 효력을 발휘하도록 합니다. 이제 LOC에서 host PC의 작업 directory를 mount하여 호스트 PC 의 디스크를 LOC 에 마운트 합니다. 이 과정을 거치면 호스트 PC 의 디스크는 LOC 입장에서 볼 때 로컬 디스크와 동일하게 사용할수 있게 됩니다

여기서 부터는 Minicom등으로 연결된 console을 통해 LOC에 대하여 명령을 내립니다. 마운트 명령으로 현재의 디스크 마운트 상태를 확인합니다.

```
#mount
```

```
/dev/root on / type umsdos (rw)
```

```
/dev/ram0 on /var type ext2 (rw)
```

```
proc on /proc type proc (rw)
```

```
/dev/vsbb on /usr type umsdos (rw)
```

마운트 명령을 이용하여 호스트 PC 의 sample 디렉토리를 LOC 의 /mnt 디렉토리에 마운트 합니다.

```
#mount -t nfs 192.168.1.7:/home/xxx/development/sample /mnt
```

마운트 명령이 성공적으로 완료되면 마운트 명령으로 그 결과를 확인하시기 바랍니다.

```
#mount
```

```
/dev/root on / type umsdos (rw)
```

```
/dev/ram0 on /var type ext2 (rw)
```

```
proc on /proc type proc (rw)192.168.1.7:/home/xxx/development/sample/mnt on /mnt  
type nfs (rw,addr=192.168.1.7)
```

```
/dev/vsbb on /usr type umsdos (rw)
```

상기와 같은 작업을 마치면, LOC는 host PC를 자신의 local directory 처럼 사용 가능합니다. 192.168.1.7은 host PC의 IP address 입니다.

이제 프로그램을 실행시켜 봅니다.

```
#cd /mnt
```

```
#./test
```

```
Hellow! LinuxOnChip.
```

```
#
```

예상한대로 결과가 출력되었습니다. 만약 이 프로그램을LOC가 재부팅될 때 마다 수행되도록 하기 위해서는 먼저 이 실행화일을 LOC 의 플래시 디스크에 복사를 하셔야 합니다.

이 실행화일을 플래시 디스크 (예를 들면 /usr/bin 디렉토리) 에 저장하시고 싶으시면 아래와 같이 하시면 됩니다.

```
#cd /mnt
```

```
#cp test /usr/bin
```

```
#sync
```

(주의: 플래시메모리에 파일을 복사 하신후에는 반드시 sync 명령을 사용해 주시기 바랍니다. 이 sync 명령은 복사한 파일이 확실히 플래시에 써졌음을 보증합니다. 리눅스와 같은 유닉스 계열의 운영체제는 내부적으로 파일들에 대한 버퍼링(캐싱이라고도 합니다) 을 하여 디스크 액세스 속도를 최적화 하도록 되어있습니다. 따라서 cp 명령후에 sync 명령을 입력 하셔야만 확실히 플래시 디스크에 해당 파일이 저장됩니다. 쉽게 이해하시려면 sync 명령을 플래시롬에 write 하는명령이라고 생각 하시면 됩니다.)

파일을 LOC 의 플래시 디스크에 복사 하셨다면 이제 호스트 PC 없이 도 이 프로그램을 실행 하실수 있습니다. LOC 가 부팅될때마다 자동으로 이 프로그램을 실행시키고 싶으시면 LOC 의 /usr/rc.d/rc file에 아래와 같이 추가합니다. (/usr/rc.d/rc 파일은 LOC 가 부팅후 자동으로 실행되는 쉘 스크립트(shell script) 입니다. DOS 에 익숙하신 분은 autoexec.bat 파일과 같다고 생각하시면 됩니다.)

```
/usr/bin/test &
```

/usr/rc.d/rc file의 수정은 "부록 : Flash file"을 참조 하십시오. **부록 : DC characteristic**

- Supply voltage : 최소 2.7V, 최대 3.6V
- Input voltage : 최소 2.7V, 최대 3.6V
- Operating temp. : 최소 0°C, 최대 70°C
- Power rating : 최대 0.5W

공급 전압은 3.3V를 유지하여야 합니다. 사용된 chip의 공급 전압이 3.3V이기 때문에 3.3V 이상의 공급 전원이나, 3.3V이상의 입력 전압을 접속할 경우에는 예측할 수 없는 상태가 발생할 수 있습니다.

**부록 : Memory map**

영역	Address range
System RAM	0x00000000 ~ 0x00800000
System Flash ROM	0x11000000 ~ 0x11400000
External CSA*	0x11400000 ~ 0x11400800
Ethernet controller	0x10000300 ~ 0x11400800
External CSB*	0x10020000 ~ 0x10020800

## 부록 : Flash file system

LOC는 Flash memory에 구현된 file system을 가지고 있으므로, 하드 디스크를 사용하는 것과 마찬가지로 file의 생성, 제거, 이동, 복사, 수정등이 가능합니다.

"mount" 명령을 key in 하면, 아래와 같이 LOC 의 로컬 디스크의 할당을 확인할 수 있습니다. (/ , /var, /usr)

```
# mount
/dev/root on / type umsdos (rw)
/dev/ram0 on /var type ext2 (rw)
proc on /proc type proc (rw)
/dev/vsbb on /usr type umsdos (rw)
```

개발용 PC의 작업 directory를 NFS(Network File System)를 이용하여 mount 하였다면,

```
# mount
/dev/root on / type umsdos (rw)
/dev/ram0 on /var type ext2 (rw)
proc on /proc type proc (rw)
192.168.1.7:/home2 on /mnt type nfs (rw,addr=192.168.1.7)
/dev/vsbb on /usr type vfat (rw)
```

으로 표시되며, 192.168.1.7 host의 /home2 directory가 mount 되었음을 알 수 있습니다. LOC는 192.168.1.7:/home2 directory를 자신의 local directory 처럼 사용 가능합니다.

Root directory는 Linux system에 대한 중요한 내용을 가지고 있습니다. 루트 디렉토리의 /etc 디렉토리 밑에는 LOC 의 환경을 설정하는 파일들이 있습니다.

/etc/rc : LOC 부팅때 가장먼저 실행을 시작하는 shell script 입니다.

/etc/rc.inet: 네트워크 설정 파일 입니다. LOC 의 IP 어드레스 게이트웨이등을 설정하는 파일 입니다. 네트워크 설정을 변경하고 싶으시면 이 파일을 수정하시면 됩니다.

/etc/resolv.conf: 네임 서버를 설정하는 파일 입니다. DNS 를 사용하신다면 적절한 네임서버를 설정해 주시기 바랍니다.

플래시 디스크의 파일을 변경하신후에는 반드시 sync 명령을 사용하셔야 실제로 플래시 디스크로의 쓰기 동작이 완료가 보증 됩니다.

```
#sync
```

Directory의 생성/제거, file의 생성, 제거, 이동, 복사등이 /, /usr partition에 대해 행해진 경우에는 반드시 상기 명령을 실행 하십시오. 해당 명령이 실행되지 않으면, flash disk 로의 file write가 실행되지 않을 수 있습니다.

주의) LOC를 이용하여 프로그램의 개발이 끝난후에는 반드시 플래시 디스크를 읽기 전용으로 설정을 하시기 바랍니다. 플래시 디스크를 읽기 전용으로 전환하면 일반적인 ROM과 동일하게 되며, 실수에 의한 시스템파일과 응용프로그램의 손상을 방지 할수 있습니다.

플래시 디스크를 읽기전용으로 설정하는 방법은 아래와 같습니다.

/usr 디렉토리의 읽기 전용전환

/etc/rc 파일내의

/bin/mount /dev/vsbb /usr 라인을 아래와 같이 변경하십시오.

/bin/mount -o ro /usr

2. / (root) 디렉토리의 읽기 전용전환

LOC 를 모니터 모드로 전환 하시기 바랍니다. (아래의 '모니터 모드로의 전환' 부를 참고 하시기 바랍니다.

모니터 모드상에서 아래와 같이 입력합니다.

Mon>setenv cmdline Arg!ro

실행이 완료되면 LOC 를 재부팅 하시기 바랍니다.

## 모니터 모드로의 전환

LOC 를 모니터 모드에 진입시키는 방법입니다. LOC 가 부팅할때 ESC 코드를 콘솔 포트를 3 회 이상 송신하면 리눅스 부팅을 중지 하고 모니터 모드에 진입합니다. 간단한 방법은 minicom 과 같은 터미널에서 ESC 키를 계속 누른 상태에서 (ESC 코드를 계속 전송) LOC 의 리셋버튼 을 누르면 됩니다. 'mon>' 이라는 프롬프트가 떨어지면 모니터 모드에 진입한것이므로 ESC 를 더이상 누르지 않아도 됩니다.

모니터 모드의 주 기능중의 하나는 환경변수의 설정입니다. 사용자는 리눅스가 부팅할 때 미리 약속된 인자를 커널에 전달할수 있으며 이 환경변수의 이름은 cmdline 입니다. 모니터모드에서는 LOC 의 환경변수를 60 개까지 설정/삭제 가 가능합니다.

환경변수를 설정하는 명령어는 setenv 입니다. 명령줄의 구조는

setenv [변수명 [변수값]]

입니다.

변수명과 변수값을 생략하면 현재 LOC 에 기록된 환경변수의 리스트를 출력합니다.

변수명을쓰고 변수값을 생략하면 해당 변수가 삭제 됩니다.

변수명과 변수값을 모두 지정하시면 해당 변수와 그 값이 LOC 에 기록 됩니다.

예를 들면 cmdline 이라는 환경변수를 만들고 싶고 그 값을 Arg!ro 로 하고 싶다면

mon>setenv cmdline Arg!ro

라고 입력하면 됩니다.

만약 이 환경변수를 지우고 싶으시면 아래와 같이 입력 하시면 됩니다.

해당 변수명의 환경변수가 삭제 됩니다.

예) mon>setenv cmdline

현재의 환경변수를 보고 싶을 때는 그냥 `setenv` 만 입력하면 됩니다.

예) `mon>setenv`

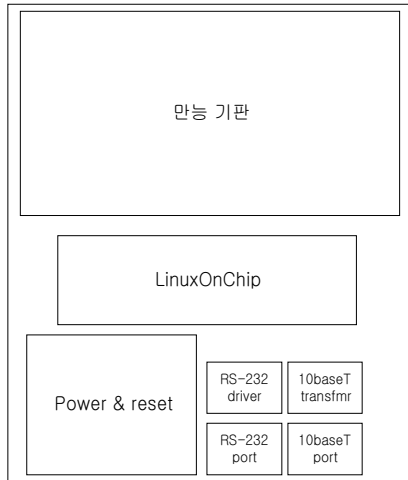
환경변수명 중에서 `cmdline` 이라는 변수명은 리눅스에 인자를 전달하기 위해 할당되어 있습니다. 변수값은 반드시 `Arg!` 로 시작 해야 합니다. 리눅스에 전달하는 인자중의 루트 디스크를 읽기전용(read only) 로 설정을 지정하는 인자는 `ro` 입니다. 만약 `LOC` 의 루트 디렉토리를 읽기전용으로 설정하고 싶으시다면 아래와 같이 입력하시면 됩니다.  
`Mon>setenv  
cmdline Arg!ro`

만약 리눅스 부팅과정에 시리얼 포트로 부팅과정에 대한 출력을 원하지 않으시면 `noprintk` 를 입력하시면 됩니다. 그리고 부팅후 로그인 프롬프트를 출력하고 대기하는 `getty` 를 실행시키지 않기를 원하시면 (시리얼 콘솔을 완전히 응용프로그램에서 제어하기를 원할 때 필요할 것입니다.) `noshell` 인자를 리눅스에 전달하시면 됩니다.

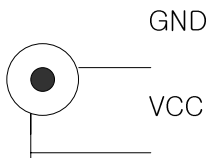
예를 들어 루트디스크를 읽기전용으로 하고 부팅메시지를 없애고 시리얼포트를 통한 로그인을 원하지 않으시다면 아래의 명령을 사용하시면 됩니다.

`Mon>setenv cmdline Arg!ro noprintk noshell`

## 부록 : Evaluation board

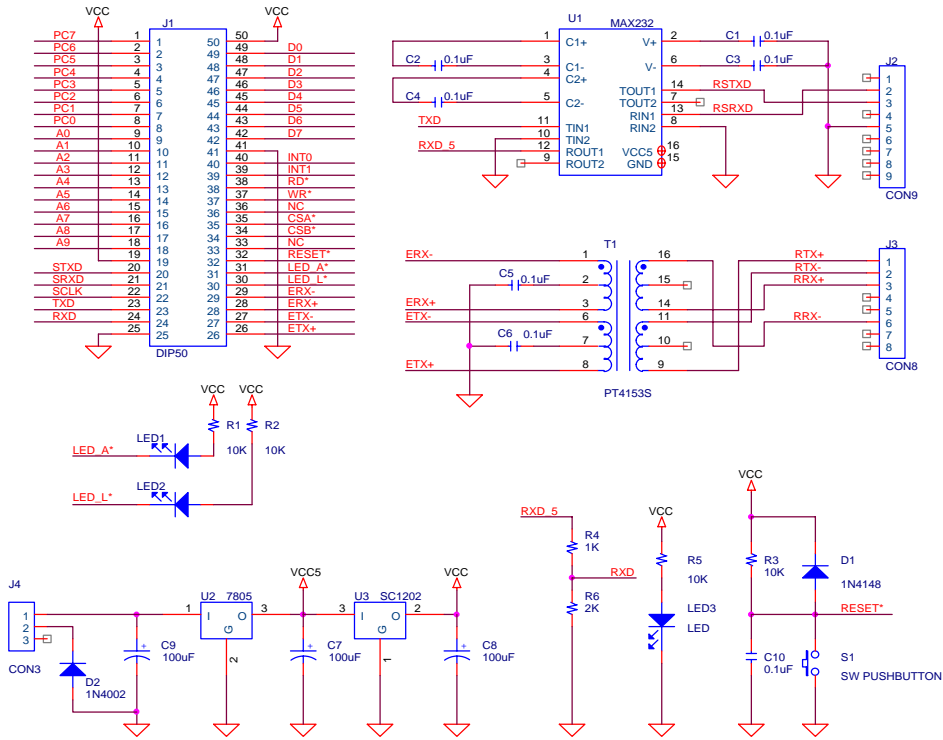


- 입력 전압 : DC 9 ~ 12V

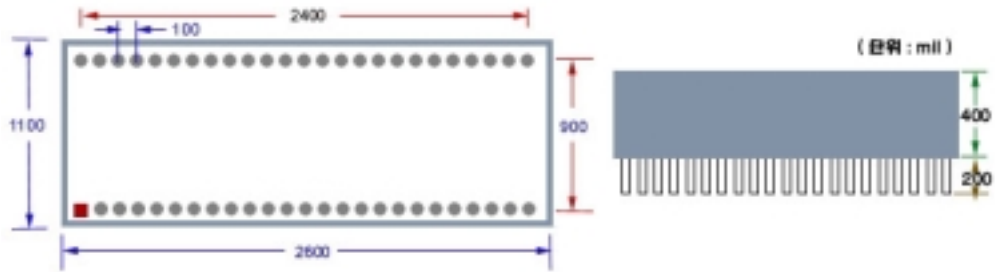


- 1 RS-232C serial port
- 1 10base-T ethernet port
- Reset key
- LOC(LinuxOnChip)를 board에 장착시, 1번 pin의 방향에 주의하십시오.  
Evaluation board의 왼쪽 아래쪽이 1번 pin 입니다. LOC의 케이스 상단에 돌기가 있는 부분이 1번 pin입니다.
- RS-232C level converter 부분에만 5V 전압이 사용되고, 나머지 부분은 3.3V 전원입니다.
- 10base-T용 transformer는 수신측 1:1, 송신측 1:2.5 입니다.  
해당하는 Transformer는 : Halo electronics의 TG92-2006N1,TG41-2006N  
: Pulse engineering의 E2023,EX2024  
: Valor electronics(Halo에 합병)의 PT4153S

Evaluation board 회로도



부록 : 외형도



**부록 : Pin 배치 및 설명**

CPU(MC68EZ328)		LinuxOnChip	
Pin number	Pin name	Pin number	Pin name
38,39,41 ~ 46	PC7 ~ PC0	1 ~ 8	PC7 ~ PC0
83 ~ 86,88 ~ 93	A1 ~ A109 ~ 18	A0 ~ A9	
		19, 50	VCC
9	SPMTXD	20	STXD
10	SPMRXD	21	SRXD
11	SPMCLK	22	SCLK
14	TXD	23	TXD
13	RXD	24	RXD
		25, 41	GND
Lan controller I/O pin		26	ETX+
		27	ETX-
		28	ERX+
		29	ERX-
		30	LED_L*
		31	LED_A*
77	RESET*	32	RESET*
		33, 36	Dont' care
54	CSB1*	34	CSB*
56	CSA1*	35	CSA*
80	UWE*	37	WR*
79	OE*	38	RD*
28	PD7	39	INT1
29	PD6	40	INT0
59 ~ 66	D15 ~ D8	42 ~ 49	D7 ~ D0

- 이름뒤에 \* 가 표시되는 pin은 low active를 의미합니다.- LOC의 address bus A[0:9]는 실제 CPU의 A[1:10]에 대치되어 있습니다. LOC 내부에서는 16bit로 access 하기 때문에 A0(CPU 측)가 의미가 없습니다. 사용자는 LOC 외부에 memory device를 장착할 경우, LOC의 pin name A[0:9]를 memory device의 A[0:9]에 접속하면 됩니다.

- Data bus에 대한 설명은 "부록 : Data bus 사용시 주의 사항"을 참조하시기 바랍니다.

## 부록 : Data bus 사용시 주의 사항

CPU의 data bus D[15:8]이 LOC의 D[7:0]에 할당되어 있습니다. 아래의 설명을 참고하여 응용 프로그램 작성시 주의하시기 바랍니다. LOC의 D[7:0]에 접속된 memory device의 data는 CPU 내부에서는 D[15:8]로 인식됩니다.

아래는 data bus를 16bit와 8bit로 각각 access할 때의 차이점을 표시하기 위한 프로그램의 출력입니다. 0x1234는 D[15:0]로 상응됩니다.

```
#Databus
Write word = 0x1234,      Read word    = 0x1200
Write word = 0x1234,      Read byte[0] = 0x12
Write word = 0x1234,      Read byte[1] = 0xa7
Write byte[0:1]=0x12,0x34, Read word    = 0x1200
Write byte[0:1]=0x12,0x34, Read byte[0] = 0x12Write byte[0:1]=0x12,0x34, Read
byte[1] = 0xa7
#
```

보시는 바와 같이 word로 0x1234를 write하고, word로 읽으면 0x1234가 읽혀야 하지만, 8bit 외부 데이터에 대한 프로그램이므로 34위치가 00으로 읽힙니다. 하지만 byte로 즉, 8bit로 읽으면 0번 byte가 0x12로 읽히고, 1번 byte가 0x34로 읽히게 됩니다. 출력에는 1번 byte가 0xa7로 읽혔는데, 이는 외부로 8bit data bus만 나와 있기 때문에 임의의 data가 읽혀서 이렇게 표시된 것입니다. 내부 16bit memory에 대해 같은 방식으로 access해 보면, 0x34로 읽힐 것입니다. 마찬가지로 8bit로 0번 byte에 0x12, 1번 byte에 0x34를 적은 후, word로 같은 address 영역을 access해 보면 0x1234로 읽힐 것입니다.

이런 현상은 Motorola 계열 CPU를 사용하는 경우에 발견할 수 있는 결과인데, 대부분 Intel 계열 CPU를 사용하던 사용자들은 처음에 약간의 혼동을 가져올 수 있습니다. word write, word read 또는 byte write, byte read 시에는 혼동이 생기지 않지만, word write 후 byte read 식으로 access 할 경우에는 위 사항을 주의하여 program 하셔야 합니다.

참고로 상기 결과는 외부에 memory device를 장착한 상태에서 운전한 것입니다.

## 부록 : Directory 구조

LOC 의 플래시 디스크에 담긴 디렉토리 구조와 파일들에 대한 자료 입니다.

### 1. 디렉토리 구조

/ 디렉토리에 아래의 디렉토리 와 파일이 있습니다.

DOS: 빈디렉토리 입니다. Umsdos 파일시스템을 사용하면 자동생성됩니다.

Bin: LOC 의 운용에 필요한 기본적인 유틸리티가 들어 있습니다.

Dev: 디바이스 파일들이 들어 있습니다.

Sbin: 시스템에 필요한 실행파일들이 들어 있습니다.

Etc: LOC 시스템 설정파일들이 들어 있습니다.

Lib: 라이브러리의 에러값에 대한 정의가 있는 파일이 있습니다.

Usr: 사용자 응용프로그램을 담기 위한 디렉토리 입니다. / 와는 별도의 플래시디스크 파티션에 할당되어 있습니다.

Var: 램디스크가 마운트된 디렉토리 입니다. 램디스크는 기본으로 512KB 로 잡혀 있습니다.

Mnt: 네트워크 드라이브를 마운트하기 위한 디렉토리 입니다.

Tmp: 임시파일저장용 디렉토리 입니다. 램디스크에 할당되어 있습니다.

Proc: 리눅스 커널의 정보를 담고 있는 파일이 들어 있습니다.

Ramfs.img: 램디스크의 이미지 파일 입니다. 512KB 에 해당하는 빈 디스크 이미지 입니다.

### 2. /usr 디렉토리 구조

사용자의 응용프로그램을 담기위한 디렉토리로써 아래의 구조를 가지고 있습니다.

/usr/bin: 작성하신 응용프로그램을 이 디렉토리에 복사 하시면 됩니다.

/usr/rc.d: 사용자 startup script 가 담긴 디렉토리 입니다. 이 디렉토리에는 rc 라는 파일이 있습니다. 이 rc 파일은 LOC 부팅후 자체 시스템 설정을 실행한 직후 실행되는 shell script 파일 입니다. 여기에 응용프로그램을 기동하는 명령을 추가하시면 됩니다.

### 3. /etc 파일들에 대한 자료.

사용자에게 관심있는 파일은 아마 /etc 디렉토리에 있는 시스템 설정화일 일것입니다.

아래의 중요한 파일들이 있습니다.

/etc/rc: 리눅스 부팅후 가장 먼저 실행되는 shell script 입니다. 시스템의 설정을 수행합니다. 이 script 에서 네트워크 설정 script 인 /etc/rc.inet 을 실행하도록 되어 있습니다. 사용자가 변경할 필요가 없습니다. 사용자 startup shell script는 /usr/rc.d/rc 파일입니다.

/etc/rc.inet: 네트워크 설정을 담당하는 shell script 입니다. IP 주소 서브넷 마스크설정 등이 이 shell script 에서 실행됩니다. 이 파일은 직접 에디팅 하셔도 되지만 제공되는 netconfig 명령을 사용하시면 손쉽게 네트워크 설정을 하실수 있습니다. Netconfig 명령은 사용자 입력을 받아들여 이 rc.inet 파일을 자동 생성하는 프로그램 입니다.

/etc/resolv.conf: 이 파일은 DNS 서버를 설정하는 파일 입니다. 역시 Netconfig 명령을 사용하시면 직접 에디팅하는 수고를 덜수 있습니다.

/etc/inettab: LOC 부팅후 로그인을 받아들이는 getty 를 기동시키는 startup table 입니다. 사용자가 변경할 필요가 없습니다.

/ etc/inetd.conf, /etc/services: inetd 라는 네트워크 데몬에 필요한 파일입니다. 사용자가 변경할 필요가 없습니다.

/etc/passwd: 패스워드 설정 파일입니다. LOC 에서는 참조되지 않습니다. LOC 의 root passwd 는 /bin/login 실행파일에 직접 하드코딩 되어 있습니다. 따라서 패스워드를 수정하시려면 개발 플랫폼에 있는 login 프로그램의 소스를 수정하신후 재 컴파일하여 LOC 의 /bin/login 을 대치하시면 됩니다.

## 부록 : LOC 의 개발환경을 설치한후의 개발 플랫폼 (데스크탑 PC) 내의 LOC 관련 디렉토리

LOC 기본 매뉴얼의 순서대로 개발킷 CDROM 내의 툴을 개발 플랫폼에 설치하신 후에는 아래의 내용이 담깁니다. 설치된 주 파일은 컴파일러 툴과 커널 소스 그리고 시스템 유틸리티의 소스 파일입니다.

설치한 내용은 모두 /opt/uClinux 라는 디렉토리 하에 생성됩니다.  
/opt/uClinux 디렉토리 의 주요 내용은 아래와 같습니다.

/opt/uClinux/bin: 컴파일러 툴들의 실행파일이 담겨 있습니다.

/opt/uClinux/linux: LOC에 탑재된 리눅스 커널의 소스입니다.

/opt/uClinux/src: LOC 에 탑재된 시스템 유틸리티들의 소스가 담겨 있는 디렉토리 입니다.

/opt/uClinux/m68k-coff : 커널 컴파일에 필요한 컴파일러 (유저 응용프로그램 개발용 컴파일러와는 다릅니다.) 에 필요한 툴들이 담겨 있는 디렉토리 입니다.

/opt/uClinux/m68k-pic-coff: 사용자 프로그램의 개발용 컴파일러에 필요한 디렉토리 구조를가지고 있습니다. ./bin ./include ./lib 세개의 디렉토리가 있습니다.

./bin: 컴파일러, 어셈블러, 링커와 라이브러리를 위한 툴(실행파일) 이 들어 있습니다.  
이 실행파일들은 설치과정중에 /usr/bin 에 그 링크가 만들어집니다. 개발툴들의 명칭은 아래와 같습니다.

m68k-pic-coff-gcc : GNU C compiler

m68k-pic-coff-ld : 링커

m68k-pic-coff-as: 어셈블러

m68k-pic-coff-ar, m68k-pic-coff-nm, m68k-pic-coff-ranlib, m68k-pic-coff-strip: 라이브러리 와 오브젝트파일 처리를 위한 툴들 입니다. 온라인 매뉴얼이 기본 설치되므로 각 실행화일에 대한 매뉴얼을 참고 하시기 바랍니다.

예) man m68k-pic-coff-gcc

./include: 표준 헤더 파일이 담긴 디렉토리 입니다. 예를 들면 #include <stdio.h> 라고 하셨다면 이 헤더파일 stdio.h 는 /opt/uClinux/m68k-pic-coff/include/stdio.h 를 의미 합니다.

./lib : 표준 라이브러리 가 담긴 디렉토리 입니다. (Libc.a)

LOC 기본 설치과정에서는 설치되지 않지만 CDROM 에는 프로그램 예제가 담겨 있습니다. 이 예제는 /cdrom mount point/LOC/examples 디렉토리에 담겨져 있습니다.