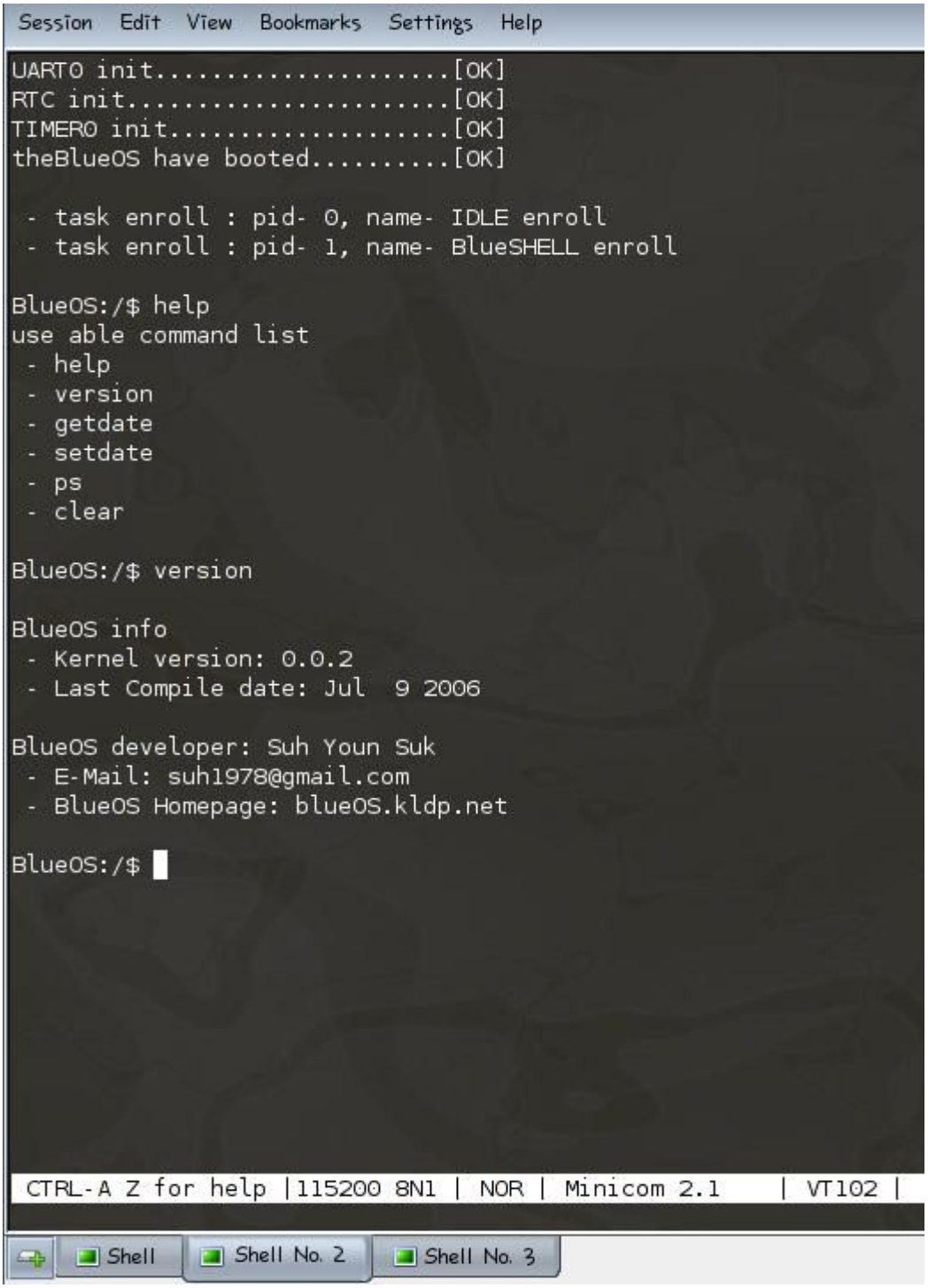
1. BlueOS

BlueOS는 임베디드 시스템을 공부하면서 테스트나 바로 적용해보려는 용도와 임베디드 관련 회사에 면접때 가지고 갈 포트폴리오를 목적으로 시작하게 된 프로젝트입니다. 구성은 부트로더인 **BlueBoot와** 마이크로 커널인 **BlueOS**, 마지막으로 BlueOS 위에서 동작하게될 태스크인 **BlueTask**로 이루어져 있습니다.
타겟보드는 블루보드입니다. 코어는 ARM계열인 S3C2440입니다. 프로그램 개발은 데비안 리눅스상에서 진행하였습니다.



부팅된 모습을 minicom을 통해 캡쳐한 화면입니다

2. 구성

(1) **BlueBoot**

```
|-- Makefile
|-- clksetup.S        - FCLK, HCLK, PCLK 설정
|-- led.S             - 디버깅용 LED on, off, wait
|-- memsetup.S        - 메모리 컨트롤러 설정
`-- start.S           - startup 코드
```

(2) **BlueOS**

```
|-- Makefile
|-- aborthandler.c    - abort exception 처리 부분 (미사용)
|-- exceptions.S      - 익셉션을 받게 되는 부분
|-- fiqhandler.c      - fiq exception 처리 부분 (미사용)
|-- head.S            - 모드별 스택설정, 커널메인으로 점프
|-- include           - 각 화일별 헤더파일
|   |-- interrupt.h
|   |-- mcu_clock.h
|   |-- printk.h
|   |-- rtc.h
|   |-- schedule.h
|   |-- stdarg.h
|   |-- string.h
|   |-- syscall.h
|   |-- system.h
|   |-- tcb.h
|   |-- timer.h
|   `-- uart.h
|-- interrupt.c       - 인터럽트 초기화
|-- irqhandler.c      - interrupt request 처리 루틴
|-- led.S             - 커널 디버깅용으로 printk 구현후 미사용
|-- load_task.S       - 태스크 로딩
|-- main.c            - 커널메인
|-- printk.c          - 리눅스의 printk와 거의 흡사
|-- rtc.c             - real time clock 디바이스 드라이버
|-- schedule.c        - 라운드 로빈 방식의 스케줄링
|-- string.c          - 문자열 관련 라이브러리
|-- swihandler.c      - software interrupt 처리 루틴
|-- tcb.c             - task control block
|-- timer.c           - timer 디바이스 드라이버로 tick로 사용
`-- uart.c            - uart0 디바이스 드라이버
```

(3) **BlueTASK0** - IDLE 태스크

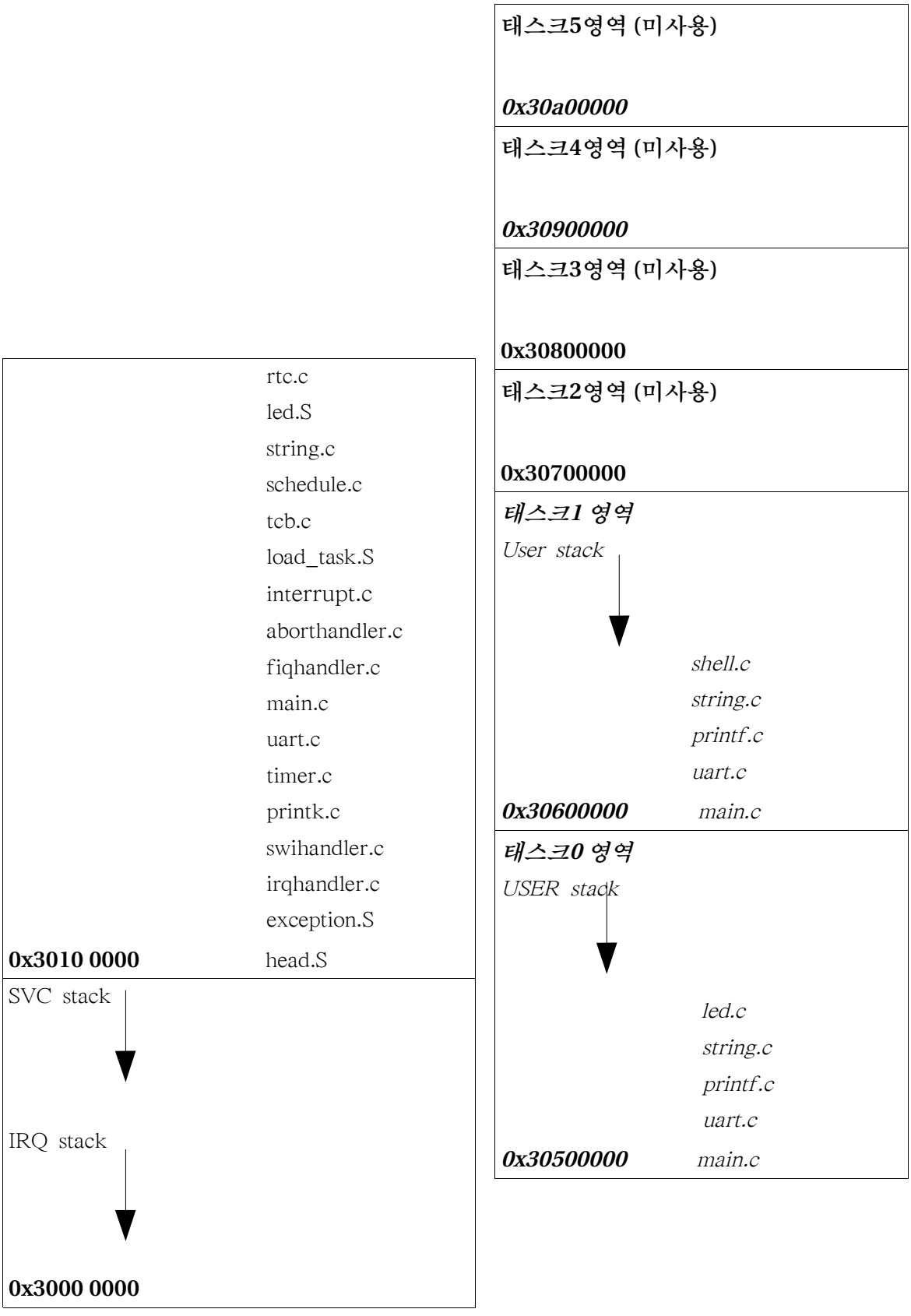
(4) **BlueTASK1** - BlueSHELL 태스크

3. 부트로더, 커널, 태스크의 NOR 영역(1MB)의 배치도

태스크5영역 (미사용)	
0xf0000	
태스크4영역 (미사용)	
0xe0000	
태스크3영역 (미사용)	
0xd0000	
태스크2영역 (미사용)	
0xc0000	
태스크1 영역	
0xbfffc	*shell.c*
	string.c
~	*printkf.c*
	uart.c
0xb0000	*main.c*
태스크0 영역	
0xafffc	*led.c*
	string.c
~	*printf.c*
	uart.c
0xa0000	*main.c*

커널영역	
0x9fffc	rtc.c
	led.S
	string.c
	schedule.c
	tcb.c
	load_task.S
	interrupt.c
	aborthandler.c
~	fiqhandler.c
	main.c
	uart.c
	timer.c
	printk.c
	swihandler.S
	irqhandler.S
	exception.S
0x10000	head.S
부트로더 영역	
0x0fffc	memsetup.S
~	clksetup.S
	led.S
0x00000	start.S

4. **부트로더, 커널, 태스크의 RAM 영역(64MB)의 배치도**



5. 부팅과정

(1) 전원이 On이 되면 익셉션중 Reset이 발생한다

(2) start.S의 startup 코드중에서 _start에서 시작해서 Reset이에 해당하는 벡터주소인 0x00000000번지에서 시작한다

(3) 디버깅용으로 사용할 LED에 관련된 GPIO등을 초기화하고, 만약 LED가 On되어 있으면 모두 Off한후에 첫번째 LED를 On시킨다

(4) 현재모드를 SVC모드로 설정하고, 혹시 발생할지 모르는 irq, fiq 발생하지 못하게 마스킹한다

(5) watchdog timer를 비활성화 시킨다

(6) FCLK, HCLK, PCLK, UCLK등의 CPU나 주변장치들의 버스로 공급되는 clock등의 속도를 설정한다. 두번째 LED를 On시킨다

(7) 메모리 콘트롤러를 초기화한다. 세번째 LED를 On시킨다

(8) NOR 영역중 0x10000 ~ 0x9fffc 에 존재하는 커널을 RAM영역인 0x30100000으로 로드한다. 네번째 LED를 On시킨다

(9) 커널로 점프한다. 정확하게는 0x30100000번지로에 해당하는 head.S로 점프한다.

(10) head.S중 exception_vector_table에서 시작하게 된다. 여기까지 온걸 확인시키기 위해 모든 LED 를 Off 한다. 먼저 stack_setup를 실행하면서 svc, irq모드의 스택을 설정한다. svc 스택은 0x30100000번지에서 아래로 작아지는 스택이다. irq 스택은 svc 스택에서 0x6000 이 작은 영역에서부터 시작해서 마찬가지로 아래로 작아지는 스택이다

(11) 커널의 kmain으로 점프한다. 스택이 만들어졌으니 이제부터는 어셈이 아닌 C언어를 사용할수 있다

(12) 인터럽트 관련 사항을 초기화한다. 기본적으로 모든 인터럽트를 마스크할것이다. 그 후에 사용하려는 인터럽트를 초기화 할때 마스크를 지워주면 된다.

(13) uart 를 초기화한다. 이제부터는 printk를 사용할수 있다.

(14) rtc 를 초기화한다. tick으로 사용할것은 timer0이다

(15) timer 를 초기화한다. tick은 200HZ로 한다

(16) 커널의 초기화하는 다 끝났다. 이제 사용할 태스크를 NOR 영역에서 RAM영역으로 로드하고 TCB를 초기화한다.

(17) task0에 해당하는 IDLE 태스크를 수행한다. 이때부터는 유저모드이고, 정상적이라면 커널로 오지는 않을것이다.

(18) 등록된 태스크 0, 1번을 돌아가면서 수행한다. 각 태스크는 등록될때 주어진 slice 값만큼에 해당하는 tick동안만 수행하고 나서 다음 tick부터는 다음 태스크를 수행하기 위해 전의 태스크의 컨텍스트 정보를 tcb에 저장하고, 스케줄링에 의해 새로운 태스크의 컨텍스트들을 복구하여 새로운 태스크를 수행한다.

6. 개발하면서 참고했던 자료 / 서적

(1) ARM System Developers Guide

- ARM 어셈에 대한 명령어등의 전반적인 지식 습득에 활용
- 9장 인터럽트 처리부분중에서 익셉션 벡터테이블, 컨텍스트 정보의 저장 및 복원에 대한 예제 참고
- 10장 펌웨어와 부트로더인에서 부트로더에 대한 예제 참고
- 11장 임베디드 운영체제에서 PCB(저는 TCB 라고 명명하였습니다.)에 컨텍스트를 저장하고 복원하는 예제

(2) MicroC/OS-II 실시간 커널

- 임베디드 운여체제에 대한 이해에 큰 도움을 주었다.
- tcb 구성할때 참고
- ENTER_CRITICAL(), EXIT_CRITICAL()를 참고
- 세마포어, 태스크 우선순위등을 참고 (개발진행중)

(3) 유영창씨의 연재기사인 부트로더 제작기 참고

(4) Blob, u-boot 부트로더, 리눅스 커널 0.0x, 블루보드에 맞게 포팅된 리눅스 커널소스 2.4 등을 참고하여 timer 등의 디바이스등을 개발하면서는데 참고

7. 공부와 포트폴리오를 목적으로 시작한 BlueOS 를 오픈 프로젝트로 하려고 합니다. 프로젝트로트의 홈페이지는 http://blueos.kldp.net 입니다. 현재까지의 모든 소스는 이곳에 등록되어 있습니다.