

CRZ S3C6410 Mango6410-R2

Enviroment and Boot Mode

<http://www.mangoboard.com/>

<http://cafe.naver.com/embeddedcrazyboys>

Crazy Embedded Laboratory

Document History

Revision	Date	Change note

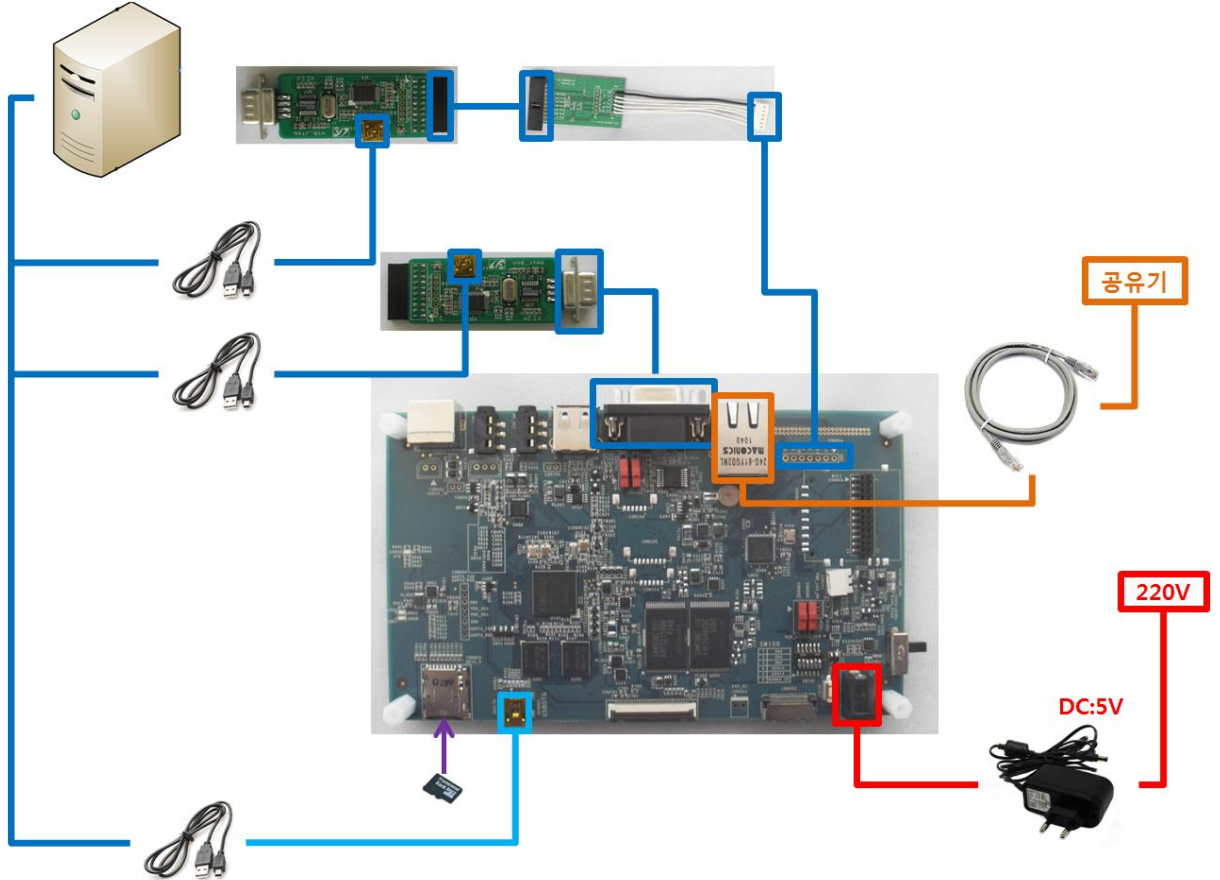
목차

1.	S3C6410 Mango-R6410 환경구성 및 부팅모드.....	4
1.1.	Mango_R6410 연결.....	4
1.1.1.	전체연결.....	4
1.1.2.	JTAG 연결.....	5
1.2.	Mango_R6410 부팅모드 설정.....	9
1.2.1.	NAND Boot.....	9
1.2.2.	NOR Boot.....	10
1.3.	Nor Flash Write하기.....	10
1.3.1.	환경구성.....	10
1.3.2.	OponOCD 연결하기.....	10
1.3.3.	Flash 정보 보기.....	11
1.3.4.	Flash 정보 및 erase 하기.....	14
1.3.5.	Flash 에 Write하기.....	16
1.3.6.	Verify image 하기.....	17

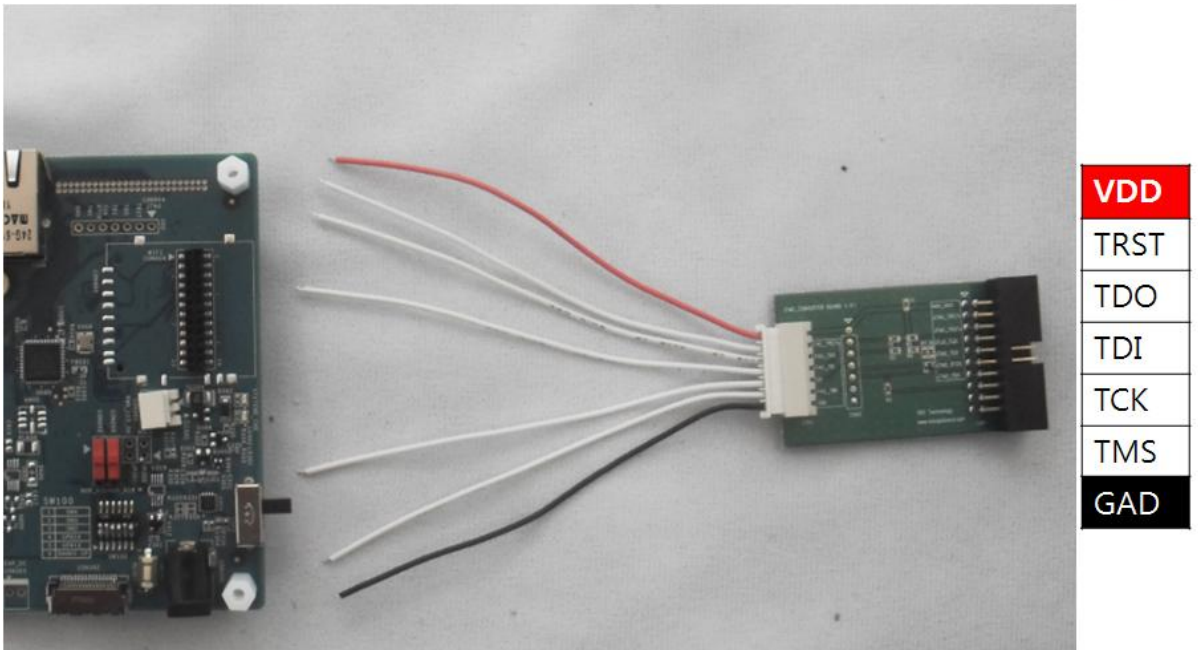
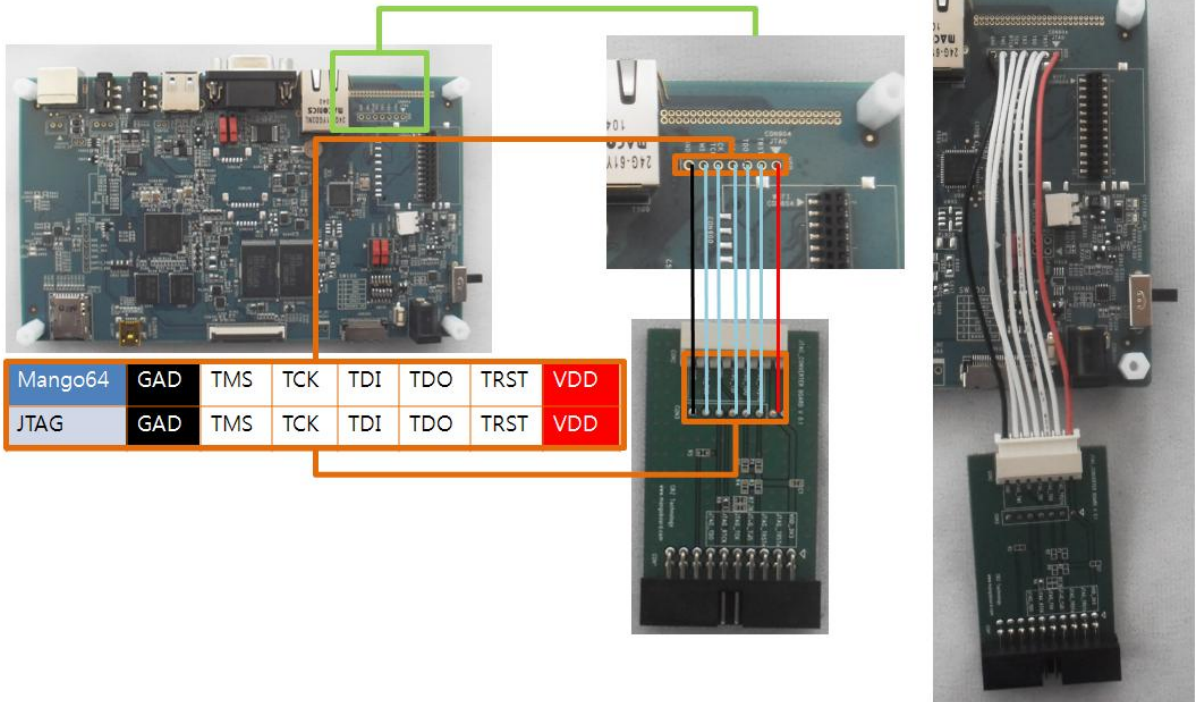
1. S3C6410 Mango-R6410 환경구성 및 부팅모드

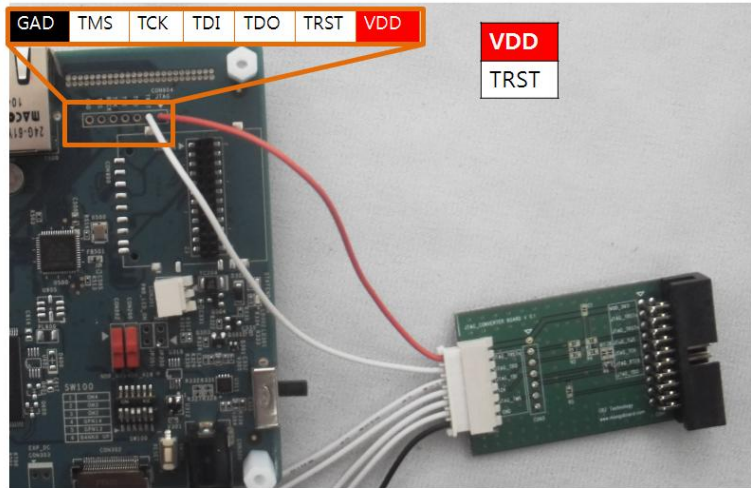
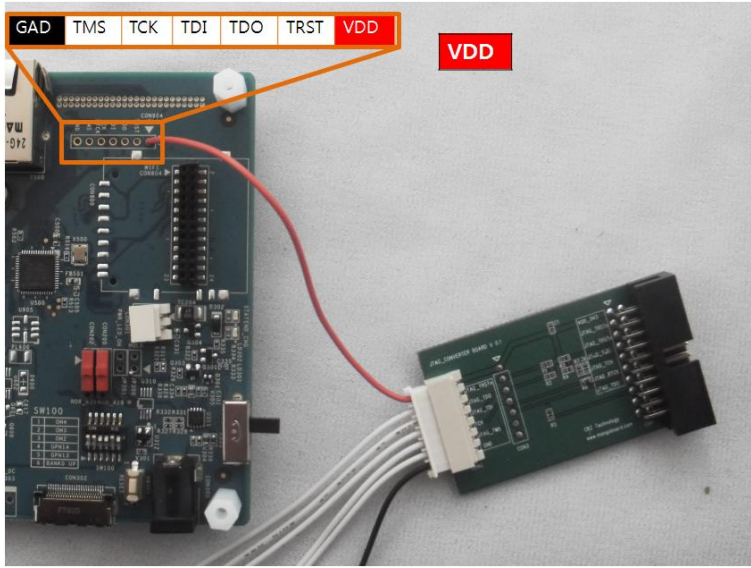
1.1. Mango_R6410 연결

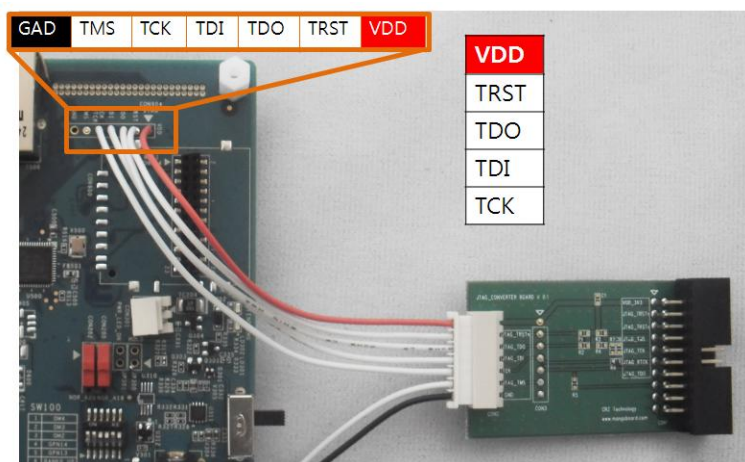
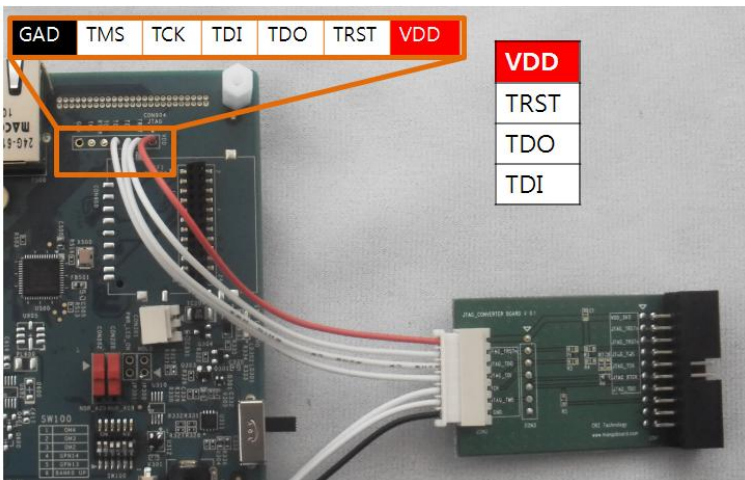
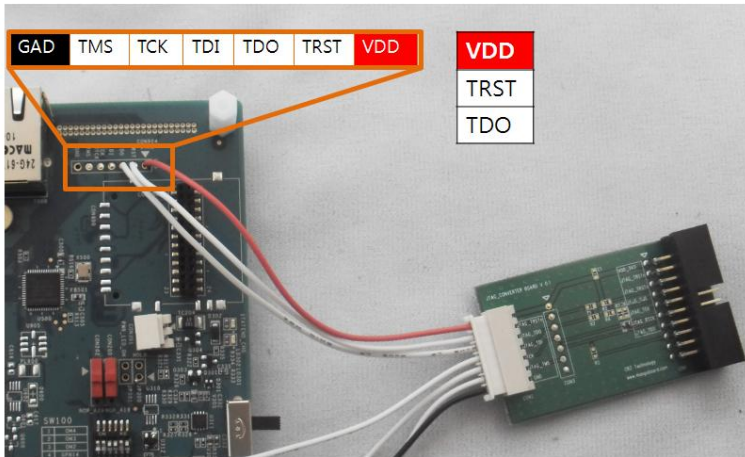
1.1.1. 전체연결

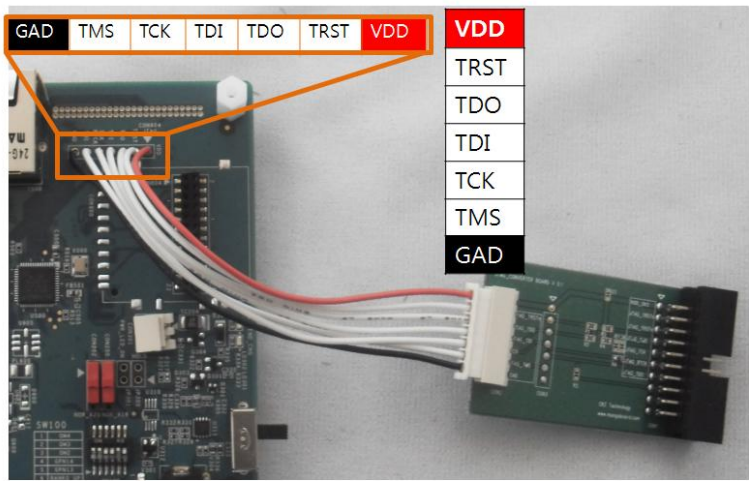
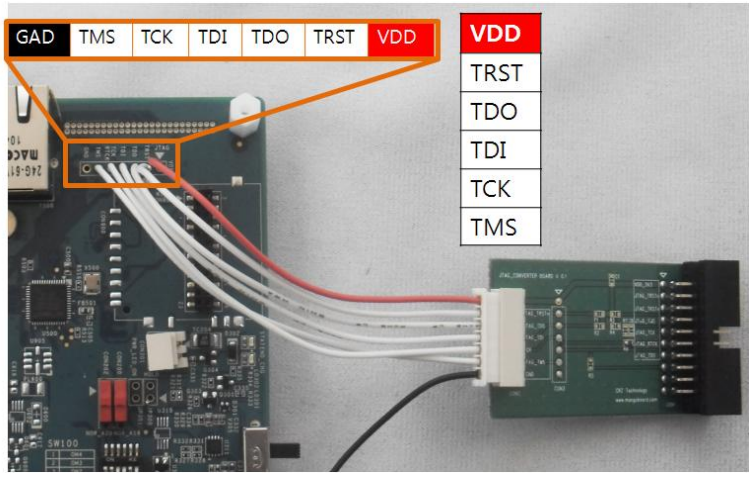


1.1.2. JTAG 연결



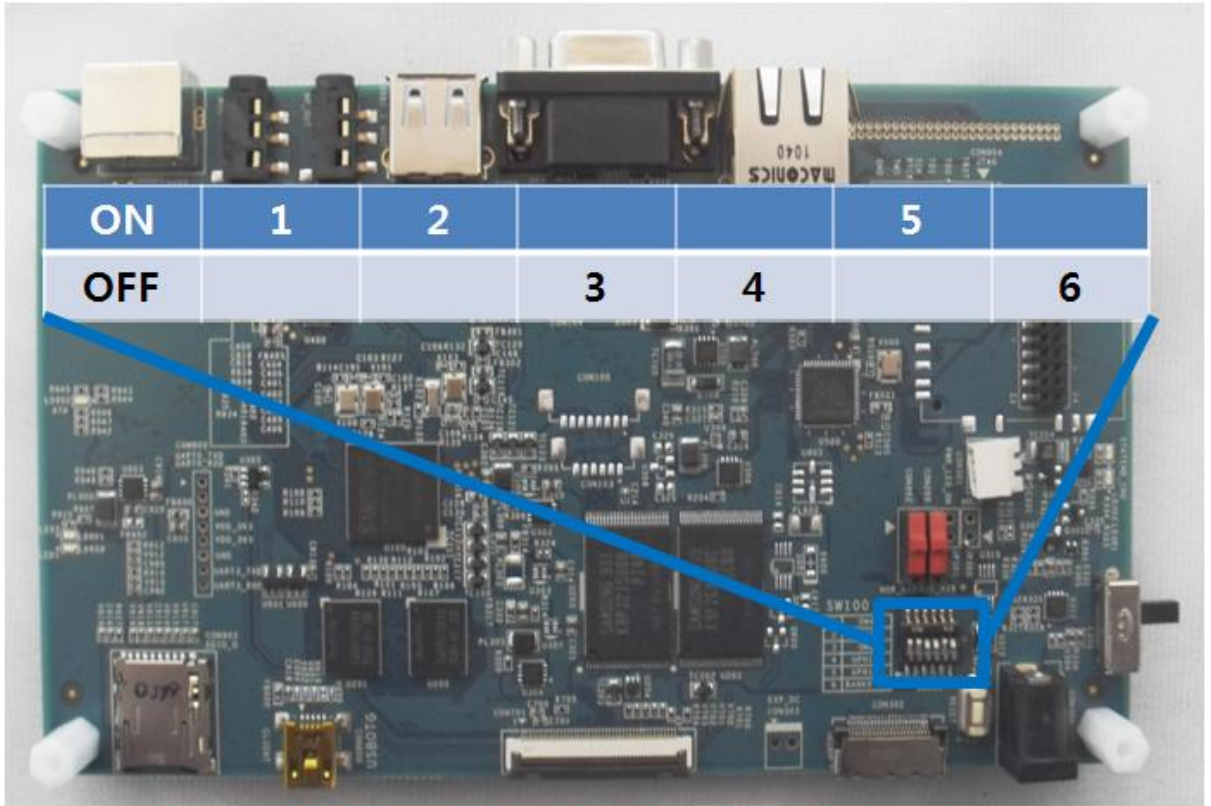




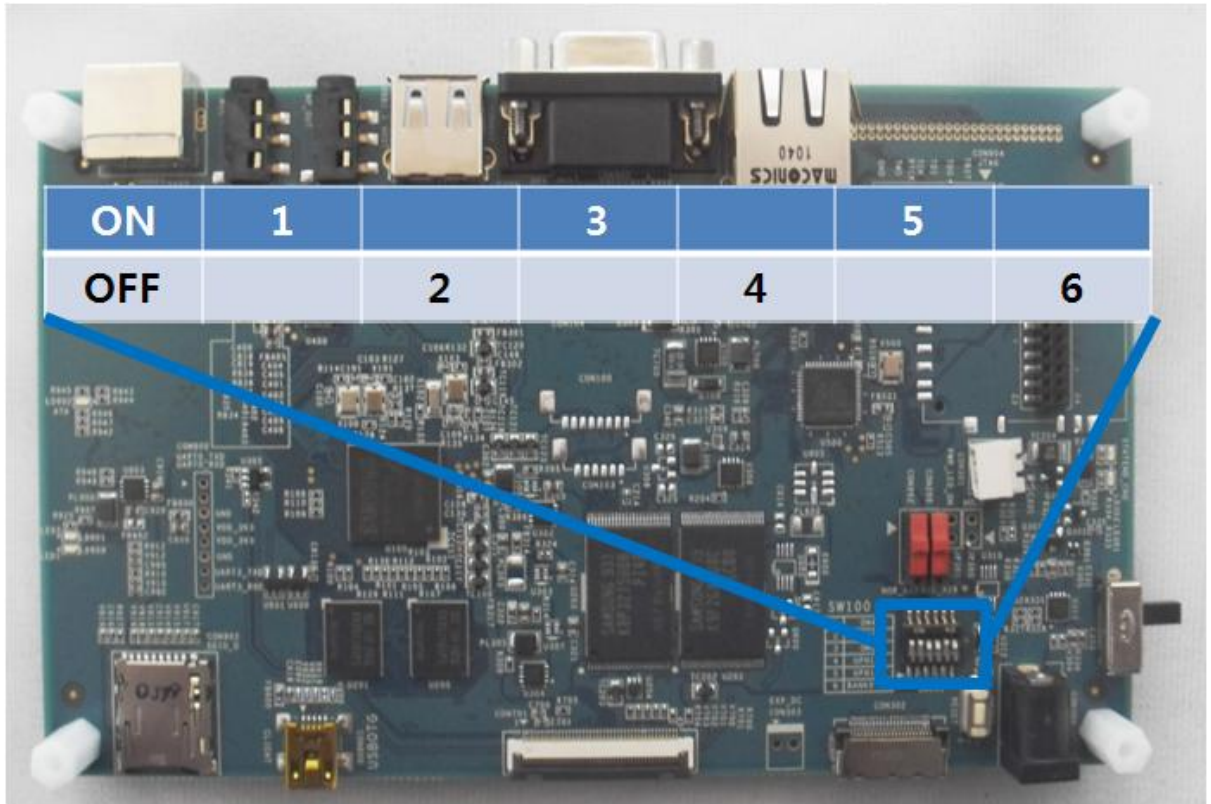


1.2. Mango_R6410 부팅모드 설정

1.2.1. NAND Boot



1.2.2. NOR Boot



1.3. Nor Flash Write하기

1.3.1. 환경구성

버전은 Openocd 0.1.0 버전입니다.

```
> version  
Open On-Chip Debugger 1.0 (2009-04-30-01:48) svn:1547
```

망고64 보드를 Nor Flash boot Mode로 SW100 스위치를 변경 및 shunt switch CON202,CON200을 0,1로 변경을 해야 합니다.

1.3.2. OponOCD 연결하기

그리고, 보드에 FT2232 usb jtag 보드를 연결합니다.

전원을 인가 후 Dos Command창을 실행합니다.

물론 , openocd.exe 파일이 있는 디렉토리에서 실행해야 합니다.

첨부파일에서

```
>openocd.exe -f mango64nor.cfg
```

입력하면 됩니다.

```
J:\Mango_project\Work\Mango-JTAG\Mango64-OpenOCD-Cygwin-091210>openocd.exe -f mango64nor.cfg
Open On-Chip Debugger 1.0 (2009-04-30-01:48) svn:1547

BUGS? Read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS

$URL: svn://svn.berlios.de/openocd/trunk/src/openocd.c $
jtag_speed: 5
1000 kHz
Info : JTAG tap: s3c6410.unknown tap/device found: 0x2b900f0f (Manufacturer: 0x787, Part: 0xb900, Version: 0x2)
Info : JTAG Tap/device matched
Info : JTAG tap: s3c6410.cpu tap/device found: 0x07b76f0f (Manufacturer: 0x787, Part: 0x7b76, Version: 0x0)
Info : JTAG Tap/device matched
Info : found ARM1176
Warn : no telnet port specified, using default port 4444
Warn : no gdb port specified, using default port 3333
Warn : no tcl port specified, using default port 6666
Info : accepting 'telnet' connection from 0
Open On-Chip Debugger 1.0 (2009-04-30-01:48) svn:1547
```

openocd 실행

Mango64nor.cfg 파일을 살펴보면,

```
source [find ./mango64.cfg]
$_TARGETNAME configure -work-area-phys 0x0c000000 -work-area-size 0x2000
flash bank cfi 0x00000000 0x00400000 2 2 $_TARGETNAME
```

mango64nor.cfg

위의 내용을 살펴보면,

```
$_TARGETNAME configure -work-area-phys 0x0c000000 -work-area-size 0x2000
```

은 OpenOCD가 steppingstone(SRAM) 영역을 사용하겠다는 의미 입니다.

```
flash bank cfi 0x00000000 0x00400000 2 2 $_TARGETNAME
```

CFI(Common Flash interface) 의 약자입니다.

0x00000000: 시작 address(정하기 나름)

0x00400000 : 망고64에 달려있는 Flash Size입니다.

2 : flash chip width

2: bus width

\$_TARGETNAME : target

<http://openocd.berlios.de/doc/html/Flash-Commands.html>

에 자세히 설명이 나와 있습니다.

또 다른 DOS Command 창을 실행 후

```
>telnet localhost 4444
```

실행합니다.

실행 후

1.3.3. Flash 정보 보기

```
>flash banks
```

명령을 입력합니다.

```
> flash banks
#0: cfi at 0x00000000, size 0x00400000, buswidth 2, chipwidth 2
>
```

flash banks 실행모습

#0 : 현재 연결 된 bank number입니다.

>flash probe 0

명령을 수행합니다.

```

> flash probe 0
Target not halted
unknown error when probing flash bank '#0' at 0x00000000

> halt
Debug entry: JTAG HALT
target state: halted
target halted due to debug-request
cpsr: 0x0000001db pc: 0x00000004

> flash probe 0
Flash Manufacturer/Device: 0x00ec 0x257e
flash 'cfi' found at 0x00000000
    
```

flash probe 실행모습

Flash에 Manufacture/Device 값을 읽어 옵니다.

이것은 flash(K8P3215UQB)의 datasheet를 참조하시면 됩니다.

Table 7. K8P3215UQB Autoselect Codes, (High Voltage Method)

Description		$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{WE}}$	A20 to A12	A10	A9	A8	A7	A6	A5 to A4	A3	A2	A1	A0	DQ15 to DQ8	DQ7 to DQ0
Manufacturer ID		L	L	H	DA	X	V _{DD}	X	L	L	X	L	L	L	L	X	ECH
Device ID	Read Cycle 1	L	L	H	DA	X	V _{DD}	X	L	L	L	L	L	L	H	25H	7EH
	Read Cycle 2											H	H	H	L	25H	03H
	Read Cycle 3											H	H	H	H	25H	01H

Manufacturer ID는 0xEC 이고, Device ID는 0x257E 라는 것을 알 수가 있습니다.

명령을 수행한 값과 동일하다는 것을 알 수 있습니다.

>flash info 0

명령을 수행 해 보겠습니다.

```

> flash info 0
#0: cfi at 0x00000000, size 0x00400000, buswidth 2, chipwidth 2
# 0: 0x00000000 <0x2000 8kB> protection state unknown
# 1: 0x00002000 <0x2000 8kB> protection state unknown
# 2: 0x00004000 <0x2000 8kB> protection state unknown
# 3: 0x00006000 <0x2000 8kB> protection state unknown
# 4: 0x00008000 <0x2000 8kB> protection state unknown
# 5: 0x0000a000 <0x2000 8kB> protection state unknown
# 6: 0x0000c000 <0x2000 8kB> protection state unknown
# 7: 0x0000e000 <0x2000 8kB> protection state unknown
# 8: 0x00010000 <0x10000 64kB> protection state unknown
# 9: 0x00012000 <0x10000 64kB> protection state unknown
# 10: 0x00013000 <0x10000 64kB> protection state unknown
# 11: 0x00014000 <0x10000 64kB> protection state unknown
# 12: 0x00015000 <0x10000 64kB> protection state unknown
# 13: 0x00016000 <0x10000 64kB> protection state unknown
# 14: 0x00017000 <0x10000 64kB> protection state unknown
# 15: 0x00018000 <0x10000 64kB> protection state unknown
# 16: 0x00019000 <0x10000 64kB> protection state unknown
# 17: 0x0001a000 <0x10000 64kB> protection state unknown
# 18: 0x0001b000 <0x10000 64kB> protection state unknown
# 19: 0x0001c000 <0x10000 64kB> protection state unknown
# 20: 0x0001d000 <0x10000 64kB> protection state unknown
# 21: 0x0001e000 <0x10000 64kB> protection state unknown
# 22: 0x0001f000 <0x10000 64kB> protection state unknown
# 23: 0x00100000 <0x10000 64kB> protection state unknown
# 24: 0x00110000 <0x10000 64kB> protection state unknown
# 25: 0x00120000 <0x10000 64kB> protection state unknown

```

flash info 실행모습

```

cfi information:
mfr: 0x00ec, id:0x257e
qry: 'QRY', pri_id: 0x0002, pri_addr: 0x0040, alt_id: 0x0000, alt_addr: 0x0000
Ucc min: 2.7, Ucc max: 3.6, Upp min: 0.0, Upp max: 0.0
typ. word write timeout: 8, typ. buf write timeout: 1, typ. block erase timeout: 512, typ. chip erase timeout: 1
max. word write timeout: 128, max. buf write timeout: 1, max. block erase timeout: 8192, max. chip erase timeout: 1
size: 0x400000, interface desc: 1, max buffer write size: 1

Spansion primary algorithm extend information:
pri: 'PRI', version: 0.0
Silicon Rev.: 0x0, Address Sensitive unlock: 0x0
Erase Suspend: 0x2, Sector Protect: 0x1
UppMin: 00.5, UppMax: 09.5

```

flash info 실행모습 2

훨씬 많은 정보를 볼수가 있습니다. Manufacture ID와 VCC 전압까지 나오고, Spansions primary algorithm을 사용 했다는 내용까지 나옵니다. 멋집니다.

1.3.4. Flash 정보 및 erase 하기

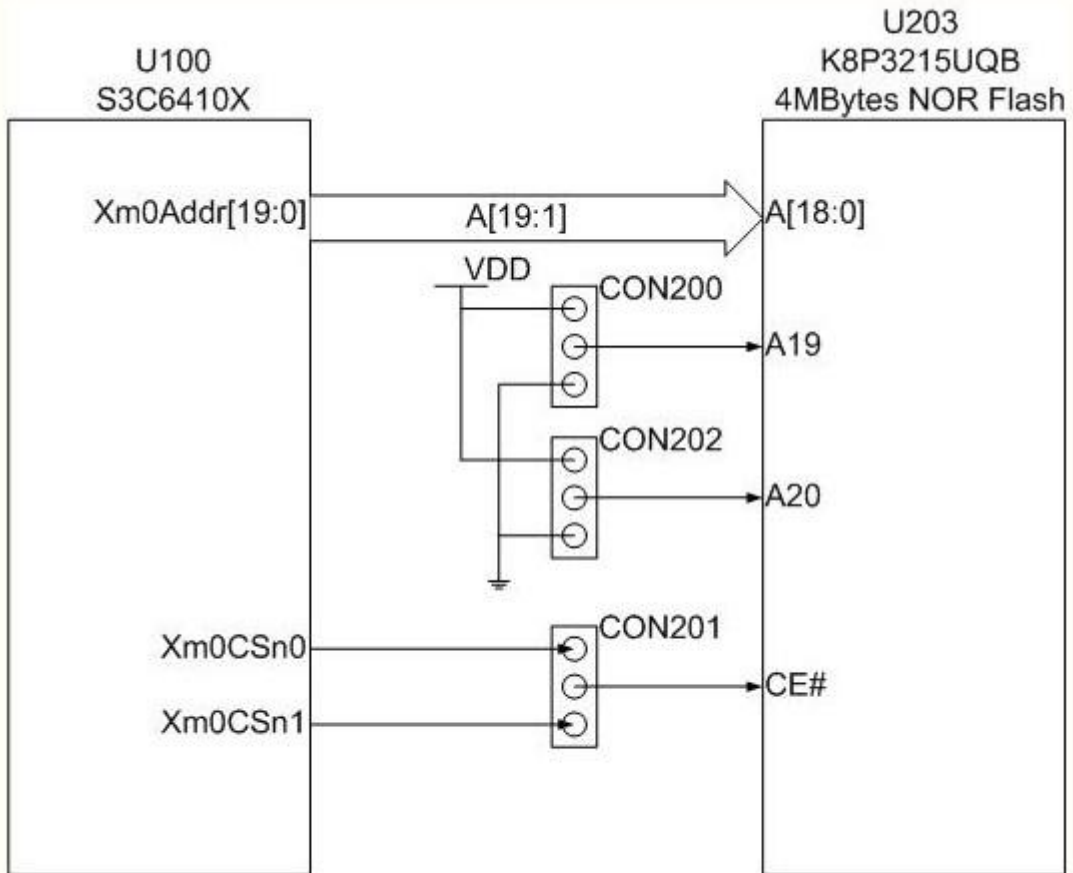
>flash info 0

명령으로 보면, 23번 sector부터 1Mbyte가 시작됩니다.

현재 Shunt Switch를 조정을 해 놓았기 때문에 0x00100000부터 Write하면 됩니다.

<http://cafe.naver.com/embeddedcrazyboys/1606>

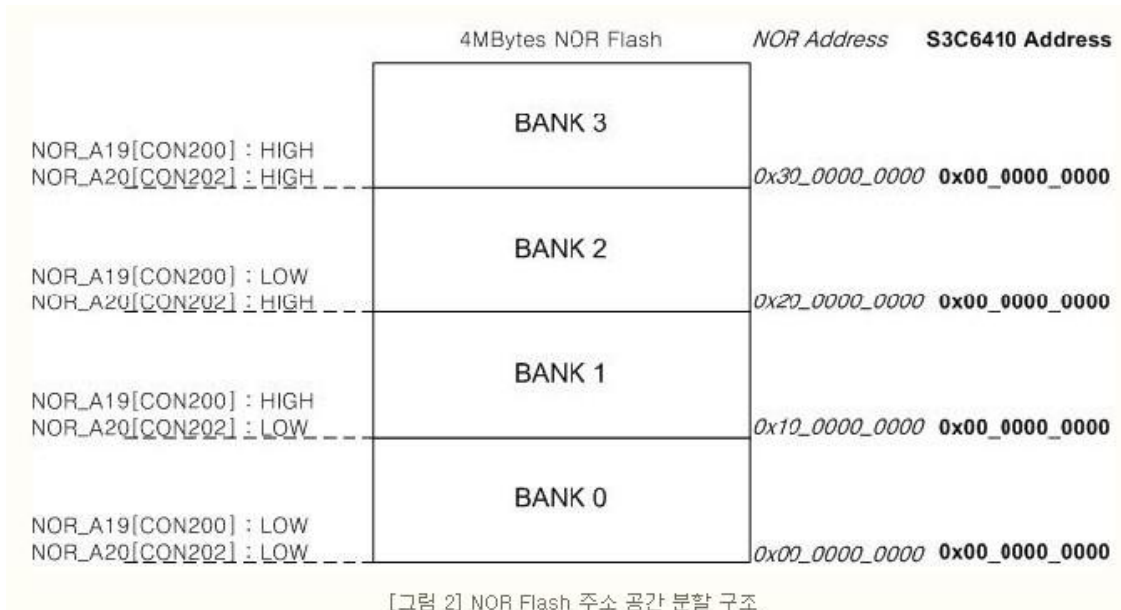
참조 하시기 바랍니다.



[그림 1] Mango64 보드 NOR flash 결선도

Mango64 보드 Nor Flash 결선도

Nor flash 메모리 MAP을 살펴보면 아래와 같습니다.



[그림 2] NOR Flash 주소 공간 분할 구조

NOR Flash 주소 분할 구조

결국, S3C6410입장에서는 시작번지는 0입니다. 현재 CON200: High, CON202: Low 상태로 해 놓았습니다.

```
> flash erase_address 0x00100000 0x10000
```

명령을 살펴보면, 0x00100000 시작해서, 0x10000(64KB) 만큼 erase합니다.

64KB만 erase하는 이유는 6410nor.bin 이 51436Byte 사이즈이기 때문입니다.

대충 봐도 64KB를 넘지 않습니다.

```
> flash erase_address 0x00100000 0x10000
erased address 0x00100000 length 65536 in 0.469000s
```

flash erase address 실행모습

또 다른 방법은 Block Number로 지우는 방법입니다.

```

> flash info 0
#0: cfi at 0x00000000, size 0x00400000, buswidth 2, chipwidth 2
# 0: 0x00000000 (0x2000 8kB) protection state unknown
# 1: 0x00002000 (0x2000 8kB) protection state unknown
# 2: 0x00004000 (0x2000 8kB) protection state unknown
# 3: 0x00006000 (0x2000 8kB) protection state unknown
# 4: 0x00008000 (0x2000 8kB) protection state unknown
# 5: 0x0000a000 (0x2000 8kB) protection state unknown
# 6: 0x0000c000 (0x2000 8kB) protection state unknown
# 7: 0x0000e000 (0x2000 8kB) protection state unknown
# 8: 0x00010000 (0x10000 64kB) protection state unknown
# 9: 0x00020000 (0x10000 64kB) protection state unknown
# 10: 0x00030000 (0x10000 64kB) protection state unknown
# 11: 0x00040000 (0x10000 64kB) protection state unknown
# 12: 0x00050000 (0x10000 64kB) protection state unknown
# 13: 0x00060000 (0x10000 64kB) protection state unknown
# 14: 0x00070000 (0x10000 64kB) protection state unknown
# 15: 0x00080000 (0x10000 64kB) protection state unknown
# 16: 0x00090000 (0x10000 64kB) protection state unknown
# 17: 0x000a0000 (0x10000 64kB) protection state unknown
# 18: 0x000b0000 (0x10000 64kB) protection state unknown
# 19: 0x000c0000 (0x10000 64kB) protection state unknown
# 20: 0x000d0000 (0x10000 64kB) protection state unknown
# 21: 0x000e0000 (0x10000 64kB) protection state unknown
# 22: 0x000f0000 (0x10000 64kB) protection state unknown
# 23: 0x00100000 (0x10000 64kB) protection state unknown
# 24: 0x00110000 (0x10000 64kB) protection state unknown
# 25: 0x00120000 (0x10000 64kB) protection state unknown

```

block number 23

Block Number 23이 64Kbyte를 영역을 가지고 있습니다.

```
>flash erase_sector 0 23 23
```

하면 됩니다. 명령 형식은 아래와 같습니다.

```

flash erase_address      erase address range <address> <length>
flash erase_check        check erase state of sectors in flash bank <num>
flash erase_sector       erase sectors at <bank> <first> <last>

```

flash erase sector 형식

실행 결과 입니다.

```

> flash erase_sector 0 23 23
erased sectors 23 through 23 on flash bank 0 in 0.500000s

```

flash erase sector 결과

1.3.5. Flash 에 Write하기

명령은

>flash write_bank 0 6410nor.bin 0x100000

```
flash write_bank write binary data to <bank> <file> <offset>
```

flash bank write 형식

살펴 보면, bank은

>flash banks 명령을 보면 0입니다.

File은 PC에 6410nor.bin 파일이름을 입력합니다.

Offset은 0x100000가 Write하기 위한 시작 address입니다.

실행 결과

```
> flash write_bank 0 6410nor.bin 0x100000
not enough working area available(requested 32768, free 8096)
not enough working area available(requested 16384, free 8096)
not enough working area available(requested 8192, free 8096)
Debug entry: breakpoint
target state: halted
target halted due to breakpoint
cpsr: 0x600001d3 pc: 0x0c00005c
Debug entry: breakpoint
target state: halted
target halted due to breakpoint
cpsr: 0x600001d3 pc: 0x0c00005c
```

flash bank write 실행 결과

1.3.6. Verify image 하기

명령은

>verify_image 6410nor.bin 0x00100000 bin

명령 형식을 보면,

```
verify_image verify_image <file> [offset] [type]
```

verify_image 형식

<file> : flash에 Write한 이미지 이름

<offset>: Write 시작 address

<type>: binary,axf, 등 format형태

실행 결과

```
> verify_image 6410nor.bin 0x00100000 bin
checksum mismatch - attempting binary compare
BUG: keep_alive() was not invoked in the 1000ms timelimit. GDB alive packet not sent! (55484)
verified 51436 bytes in 55.437000s
```

verify 실행결과

또 다른 방법은

>dump_image nordump.bin 0x00100000 0x10000

명령으로 ,nordump.bin과 6410nor.bin을 비교해 보는 방법입니다.

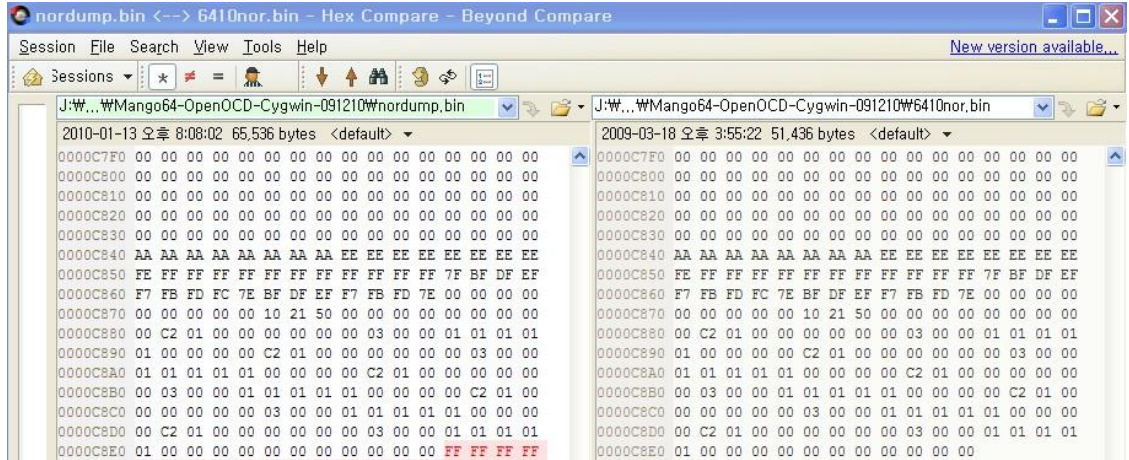
작은 이미지인 경우만 사용하세요. 이미지 크기가 크면, 하 세월입니다.

```
> dump_image nordump.bin 0x00100000 0x10000
dumped 65536 byte in 70.655998s
```

dump image 실행결과

Beyond Compare 프로그램으로 비교결과 값이 동일합니다.

0xC8EC(51436)Byte 동일합니다.



compare 결과