JTAG Hacking

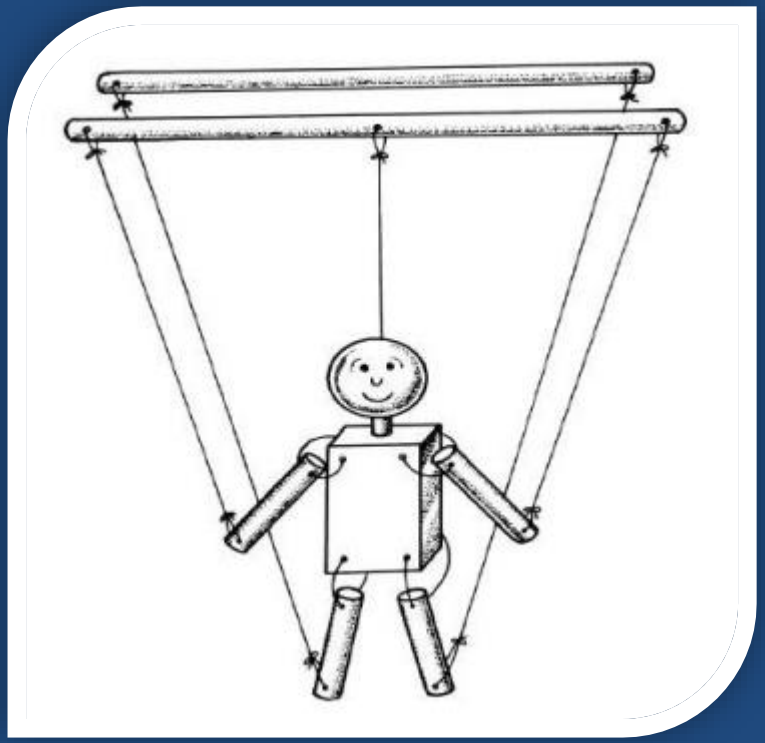
mongii@grayhash

개요

- 하드웨어 디버깅 표준인 JTAG의 개념에 대해 이해한다.
- AVR을 이용하여 JTAG을 직접 제어하는 도구를 개발해 봄으로써 작동원리를 이해한다.
- 동적 디버깅, Flash Memory Dump 등 각종 JTAG 활용 예제에 대해 알아본다.

JTAG의 개념 이해하기

- JTAG = 하드웨어 디버깅의 표준
- CPU를 마음대로 제어할 수 있다.
  - 즉, JTAG 연결에 성공하면, 모든 것을 다 할 수 있다.



JTAG의 탄생

- JTAG : Joint Test Action Group
  - 기술이름이자 동시에 그룹이름
  - 하드웨어 디버깅의 표준
    - 1990년 : IEEE에서 표준화

- 기능
  - 각 Pin들의 상태 모니터링
  - Firmware 코드를 line by line으로 디버깅 가능
  - Firmware dump, writing 가능
  - 그 외 하드웨어적인 모든 것 가능

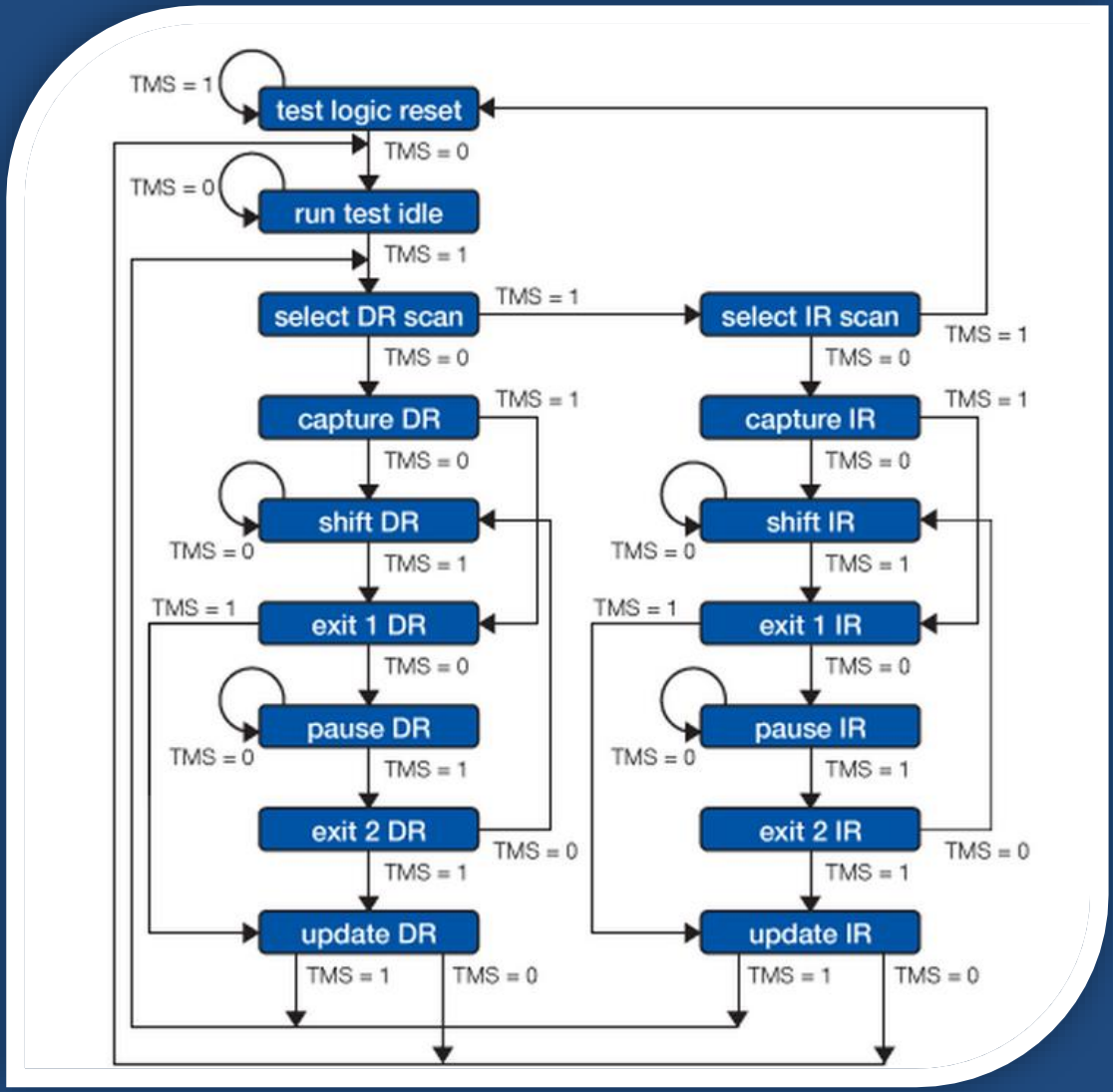
JTAG 연결에 사용되는 핀들

- TDI : Test Data In
- TDO : Test Data Out
- TMS : Test Mode Select
- TCK : Test Clock
- TRST : Test Reset (Optional)

TAP Controller

- JTAG를 제어하는 핵심 장치
- JTAG이 무엇을 할지 명령을 내릴 수 있다.
- 줄여서 TAPC라고 한다.

TAPC 상태도(State machine)



TMS와 TCK를 먼저 이해하자

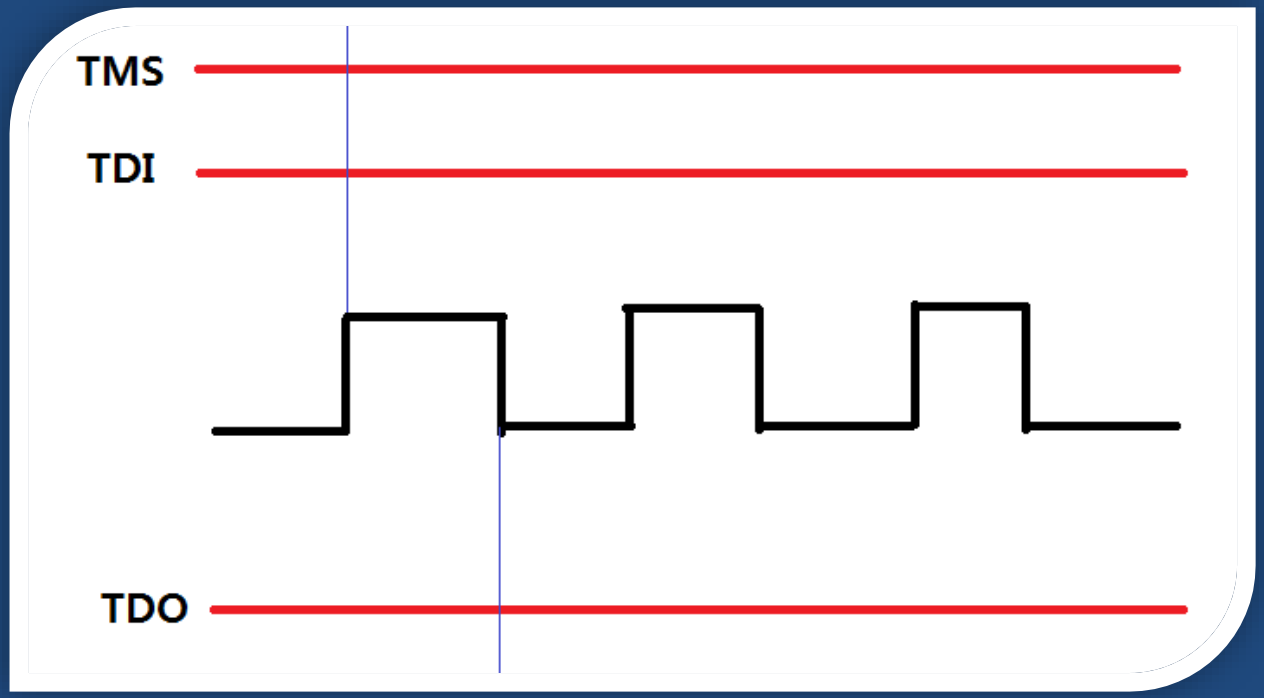
- TMS와 TCK는 TAPC의 상태를 변화시킨다.
- TMS : 방향 전환
- TCK : 이동

TDO, TDI, TRST

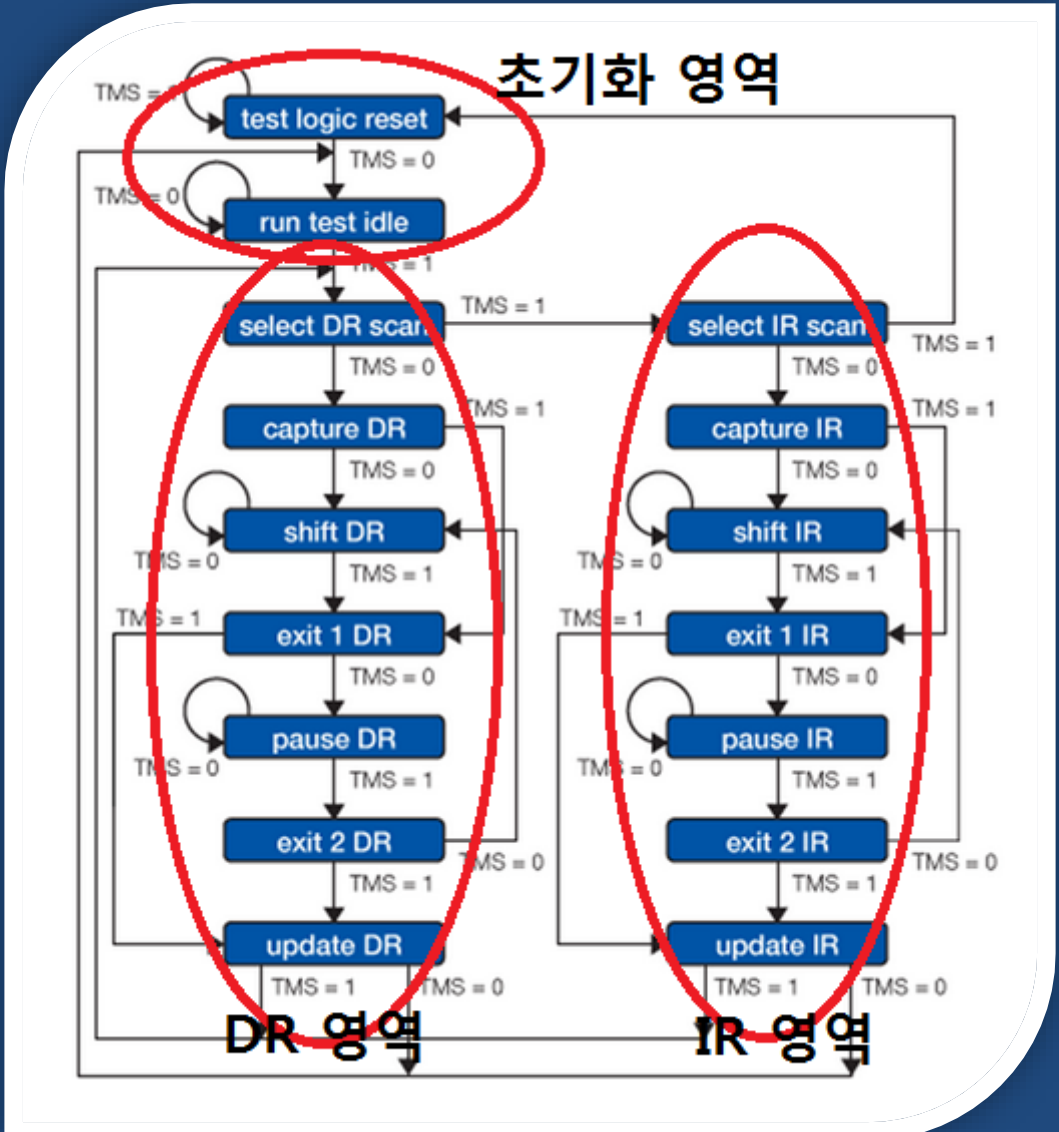
- TDO : JTAG으로부터 출력이 나오는 핀
- TDI : JTAG이 입력을 받는 핀
- TRST (필수는 아니다)
  - 리셋 핀
    - Test-Logic Reset
  - LOW 신호를 주면 JTAG이 초기화 됨
    - 즉, TAPC와 레지스터들이 초기화 됨

Rising edge와 Falling edge

- TCK : Clock 제공 (해커 -> 대상 장비)
  - TDI/TMS : Rising Edge에 실어 보낸다.
  - TDO : Falling Edge 직후에 받는다.



TAPC의 상태 설명


초기화 영역
test logic reset
run test idle
select DR scan
capture DR
shift DR
exit 1 DR
pause DR
exit 2 DR
update DR
select IR scan
capture IR
shift IR
exit 1 IR
pause IR
exit 2 IR
update IR
TMS = 0
TMS = 1
DR 영역
IR 영역

IR과 DR

- IR : Instruction Register (명령 레지스터)
  - 4비트 크기
  - JTAG을 어떤 모드로 작동시킬 것인가?
  - 외부로 노출된 TDI와 TDO를 내부의 어디에 연결시킬지 결정한다.
  - 연결 대상 : 내부 Boundary Cell 혹은 특정 레지스터

- DR : Data Register (데이터 레지스터)
  - 실제 데이터 전송을 위한 레지스터
  - 위 IR에 의해 연결된 대상의 값이 DR에 저장된다.
    - IR의 값에 따라 자동으로 연결 된다
  - 종류
    - Boundary Cell의 값
    - 그 외 레지스터들의 값 (ex. ID Register, Bypass Register)

Instruction의 종류
(AVR 기준)

- 1111 : BYPASS
  - JTAG 제어 없이 신호를 그냥 bypass 시킴
- 0001 : TDI와 TDO를 ID 레지스터에 연결
- 0010 : TDI와 TDO를 Cell에 연결, SAMPLE, 모니터링
- 0000 : TDI와 TDO를 Cell에 연결, EX-Test
  - MCU의 출력 핀을 제어한다. 가장 많이 사용되는 기능
- 1100 : RESET

대상 CPU, MCU의 JTAG 이해하기

JTAG Interface and On-chip Debug System

Features

- JTAG (IEEE std. 1149.1 Compliant) Interface
- Boundary-scan Capabilities According to the IEEE std. 1149.1 (JTAG) Standard
- Debugger Access to:
  - All Internal Peripheral Units
  - Internal and External RAM
  - The Internal Register File
  - Program Counter
  - EEPROM and Flash Memories
- Extensive On-chip Debug Support for Break Conditions, Including
  - AVR *Break* Instruction
  - Break on Change of Program Memory Flow
  - Single Step Break
  - Program Memory Breakpoints on Single Address or Address Range
  - Data Memory Breakpoints on Single Address or Address Range
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- On-chip Debugging Supported by AVR Studio

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface can be used for

- Testing PCBs by using the JTAG Boundary-scan capability
- Programming the non-volatile memories, Fuses and Lock bits
- On-chip debugging

Atmega128A datasheet 254~255p

EXTEST; $0

Mandatory JTAG instruction for selecting the Boundary-scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-scan chain is driven out as soon as the JTAG IR-register is loaded with the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

IDCODE; $1

Optional JT
consists o
This is th

The act

- Cap
- Shi

SAMPLE_PRELOAD; $2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latched are not connected to the pins. The Boundary-scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-scan Chain.
- Shift-DR: The Boundary-scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-scan Chain is applied to the output latches. However, the output latches are not connected to the pins.

AVR_RESET; $C

The AVR specific public JTAG instruction for forcing the AVR device into the Reset mode or releasing the JTAG Reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the Reset will be active as long as there is a logic 'one' in the Reset Chain. The output from this chain is not latched.

The active states are:

- Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; $F

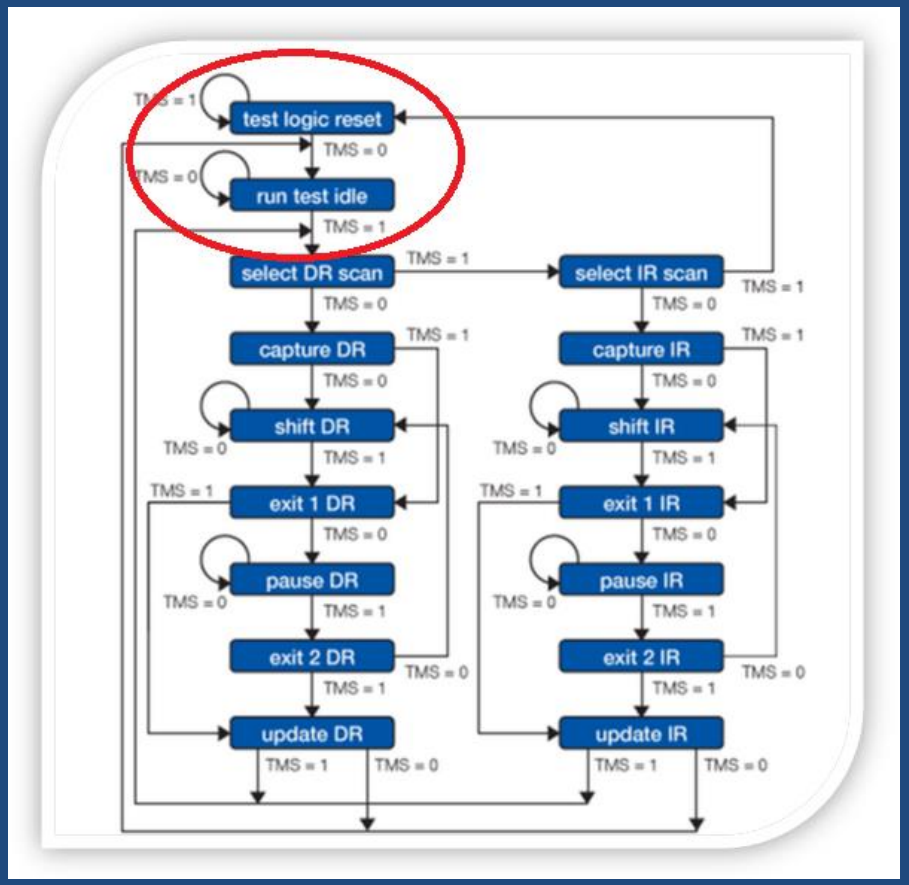
Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic “0” into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

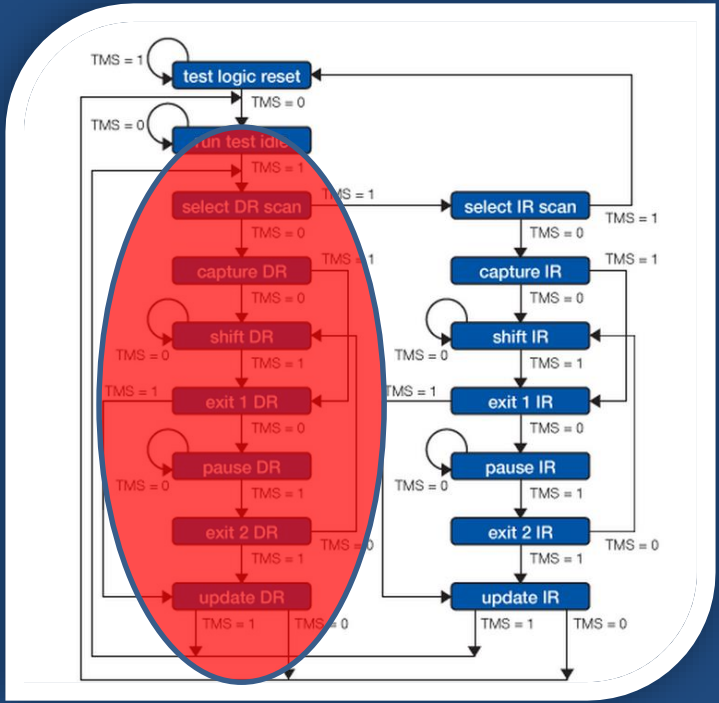
TAPC 상태 설명

- Test-Logic-Reset
  - JTAG 초기화
- Run-Test/Idle
  - JTAG 작동 시작



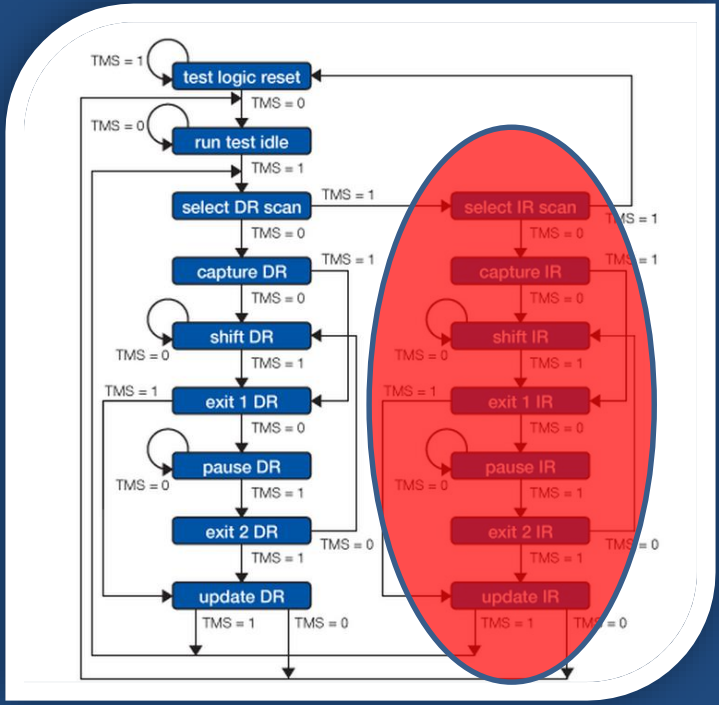
TAPC 상태 설명

- Select DR-Scan
  - DR 영역과 IR 영역에 대한 분기점
- Capture DR
  - 현재 Cell의 내용을 캡쳐
  - 이 때, 값이 업데이트 된다
- Shift DR
  - 해당 값을 TDO으로 1비트씩 출력
- Exit1 DR, Pause DR, Exit2 DR : 분기점
- Update DR
  - Shift하면서 새로 들어간 DR 값 업데이트



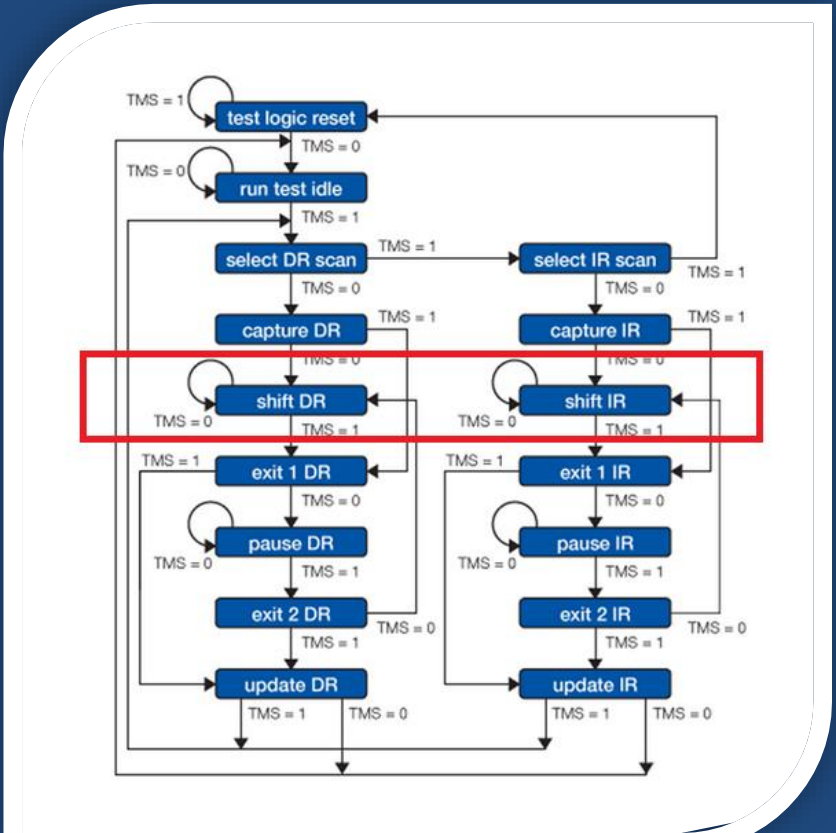
TAPC 상태 설명

- Select IR-Scan : 분기점
- Capture IR
  - 현재 IR의 값을 가져온다
- Shift IR
  - 새로운 IR 값을 넣고, 동시에 출력
- Exit1 IR, Pause IR, Exit2 IR : 분기점
- Update IR
  - 앞서 받은 IR 값 업데이트



SHIFT-IR과 SHIFT-DR

- TMS=0일 때 자신을 다시 실행한다.
- 즉, 자신을 다시 실행할 때마다 값이 shift 된다.
- 이 때 새로운 값을 넣어주면(TDI) 동시에 기존 값이 밖으로(TDO) 출력된다.



무엇을 어디로 Shift 시키는가?

- SHIFT-IR
  - TDI가 IR로 연결된다.
  - 즉, 우리가 보내는 입력이 IR로 전달된다.
  - 동시에 기존 값이 TDO로 나온다.
- SHIFT-DR
  - TDI가 DR로 연결된다.
  - 즉, 우리가 보내는 입력이 DR로 전달된다.
  - 동시에 기존 값이 TDO로 나온다.

Capture-IR과 Shift-IR

- Capture-IR
  - IR 값을 빼내올 준비를 한다. (값 획득)
  - 실제 빼오는 건 SHIFT-IR에서 한다.

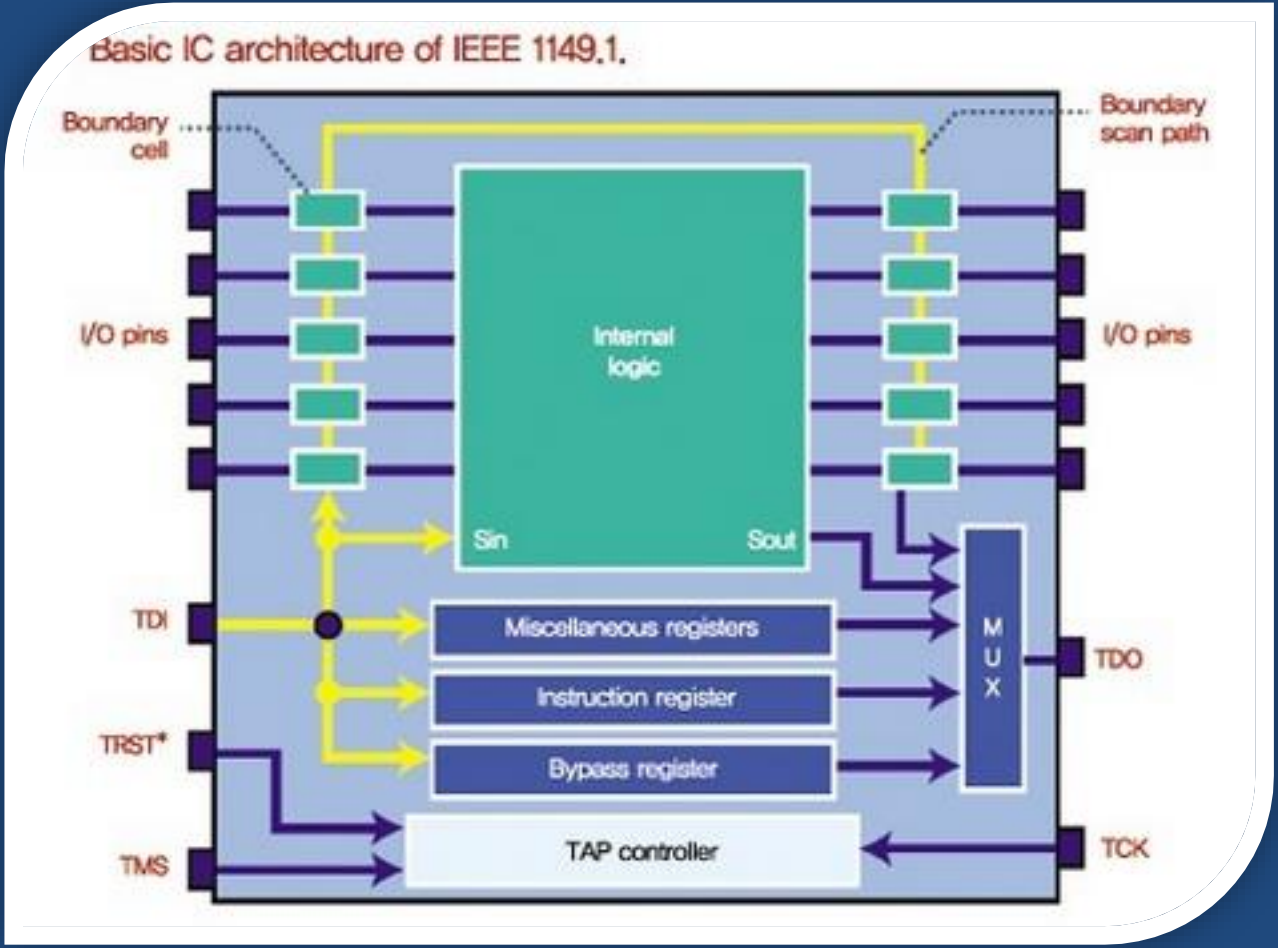
- Shift-IR
  - 쓰기와 읽기를 동시에 한다
  - 새 값을 넣으면 기존 값은 나온다.

TAPC를 reset하는 방법

1. /TRST에 LOW 신호를 준다.
   * TRST 핀이 있을 경우에 해당

2. TMS를 1로 둔 채 최대 6번 이동한다.
   - 현재 어떤 상태에 있던지 무조건 reset으로 간다.

JTAG 시스템의 회로도



JTAG tool 개발 실습
- IDCODE 읽기 -

IDCODE 읽기 과정 요약

- IR에 IDCODE 명령 패턴을 집어 넣는다.
  - Shift-IR, Update-IR 이용
- DR 값을 읽어서 꺼내온다.

ID 레지스터란?

- Identification Register
  - 32비트
  - 비트 0 : IDCODE 존재 여부
  - 비트 1~11 : 제조사
  - 비트 12~26 : 부품 번호
  - 비트 27~31 : 버전 번호

ID 레지스터란?

ice Identification egister

Figure 122 shows the structure of the Device Identification Register.

Figure 122. The Format of the Device Identification Register

	MSB			LSB
Bit	31 28	27 12	11 1	0
Device ID	Version	Part Number	Manufacturer ID	1
	4 bits	16 bits	11 bits	1-bit

Version

Version is a 4-bit number identifying the revision of the component. The JTAG version number follows the revision of the device, and wraps around at revision P (0xF). Revision A and Q is 0x0, revision B and R is 0x1 and so on.

Part Number

The part number is a 16-bit code identifying the component. The JTAG Part Number for Atmel® AVR® ATmega128 is listed in Table 100.

Table 100. AVR JTAG Part Number

Part Number	JTAG Part Number (Hex)
ATmega128	0x9702

Manufacturer ID

The Manufacturer ID is a 11-bit code identifying the manufacturer. The JTAG manufacturer ID for ATMEL is listed in Table 101.

Table 101. Manufacturer ID

Manufacturer	JTAG Manufacturer ID (Hex)
ATMEL	0x01F

IDCODE 요청

- IR 레지스터
  - 0001이 되어야 함
  - 즉, JTAG의 명령 모드를 0001로 바꾸어야 함

- 어떻게?
  - TAPC의 상태를 Shift-IR까지 옮긴 후,
  - TDI에 0001을 순서대로 입력한다.
    - 최하위 비트부터 입력, 즉 반대로 1000

- 그럼 DR 레지스터에 IDCODE가 담길 준비가 된다.

IDCODE 획득

- Shift-DR로 이동
  - DR <-> TDI/TDO 연결
- TCK가 1->0으로 변할 때 값이 나옴
- TDI에 아무값(그냥 0)을 32번 밀어 넣어줘서 32비트의 IDCODE가 나온게 만든다.

실습 문제

- 래빗 보드 IDCODE를 가져오는 JTAG 툴을 개발해 보세요.

- 절차
  - JMOD128 보드의 입출력 핀을 래빗 보드의 JTAG 핀에 입력한다. (TDO, TDI, TMS, TCK)
  - TAPC 상태 변화를 통해 JTAG을 제어한다.
  - TDO로 IDCODE 값을 얻어온다.

주의사항

- 래빗 보드의 AVR 퓨즈 비트들 중 JTAGEN 값이 ENABLE 되어있어야 한다.
- 퓨즈 비트란? MCU의 기본 상태를 설정하는 값
- JTAGEN 상태에선 PORTF의 상위 4비트를 사용하지 못하게 된다.

핀들의 현재 값 알아내기
(Monitoring)

핀들의 현재 값 알아내기

- IR => SAMPLE(Monitor) 모드(0010)로 선택
  - DR에 boundary-scan cell chain이 연결된다
- Shift-DR : 204개의 0을 밀어넣기
  - 동시에 TDI로 나오는 값을 빼온다

결과 출력 예제

```
readbit[0] : 0
readbit[1] : 0
readbit[2] : 0
readbit[3] : 0
readbit[4] : 0
readbit[5] : 0
readbit[6] : 0
readbit[7] : 0
readbit[8] : 0
readbit[9] : 0
readbit[10] : 0
readbit[11] : 0
readbit[12] : 0
readbit[13] : 1
readbit[14] : 0
readbit[15] : 0
readbit[16] : 1
readbit[17] : 0
readbit[18] : 0
readbit[19] : 1
readbit[20] : 0
readbit[21] : 0
readbit[22] : 1
```

```
readbit[0] : 0
readbit[1] : 0
readbit[2] : 0
readbit[3] : 0
readbit[4] : 0
readbit[5] : 0
readbit[6] : 0
readbit[7] : 0
readbit[8] : 0
readbit[9] : 0
readbit[10] : 0
readbit[11] : 0
readbit[12] : 0
readbit[13] : 1
readbit[14] : 1
readbit[15] : 0
readbit[16] : 1
readbit[17] : 1
readbit[18] : 0
readbit[19] : 1
readbit[20] : 1
readbit[21] : 0
readbit[22] : 1
```

22	PA3.Control	
21	PA3.Pullup_Enable	
20	PA2.Data	
19	PA2.Control	
18	PA2.Pullup_Enable	
17	PA1.Data	
16	PA1.Control	
15	PA1.Pullup_Enable	
14	PA0.Data	
13	PA0.Control	
12	PA0.Pullup_Enable	
11	PF3.Data	Port F
10	PF3.Control	
9	PF3.Pullup_Enable	
8	PF2.Data	
7	PF2.Control	
6	PF2.Pullup_Enable	
5	PF1.Data	
4	PF1.Control	
3	PF1.Pullup_Enable	
2	PF0.Data	
1	PF0.Control	
0	PF0.Pullup_Enable	

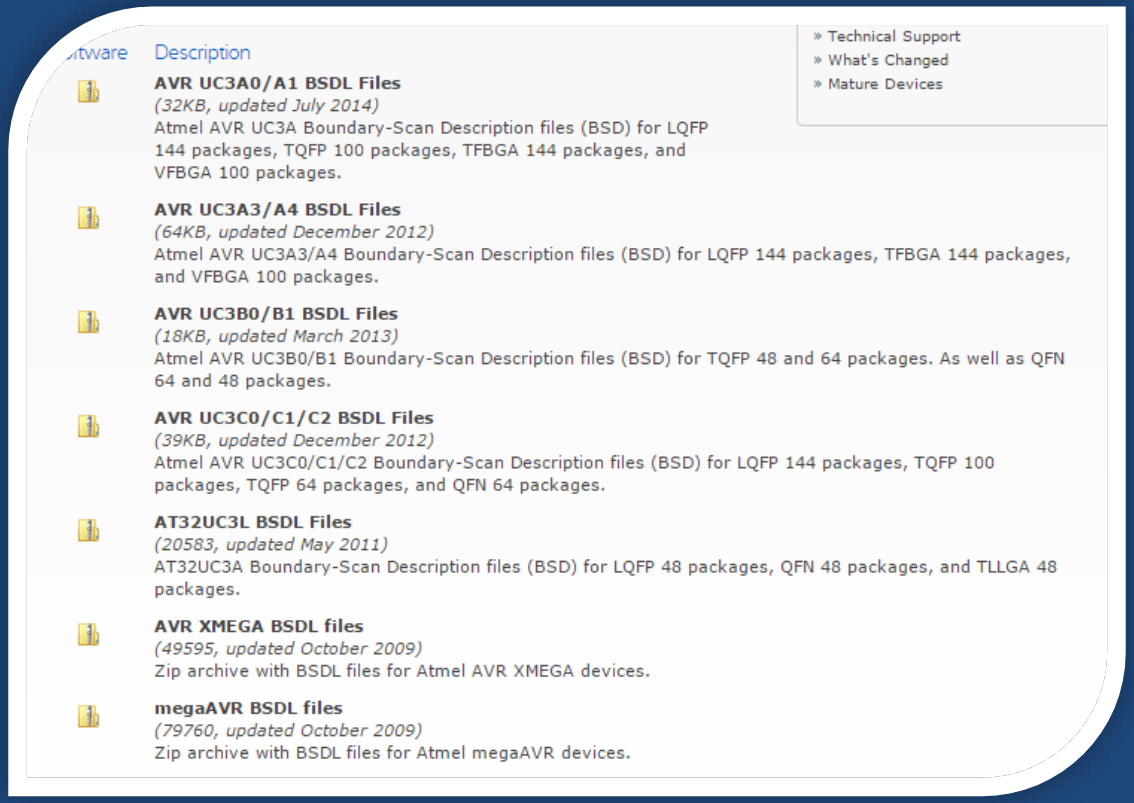
Boundary-Scan Order

- 몇 번째 Boundary가 어떤 핀에 연결 되어있는지를 보여주는 지도

- Atmega128A : 총 204개
- 254page에 있다.
- http://www.atmel.com/Images/Atmel-8151-8-bit-AVR-ATmega128A_Datasheet.pdf

BSDL

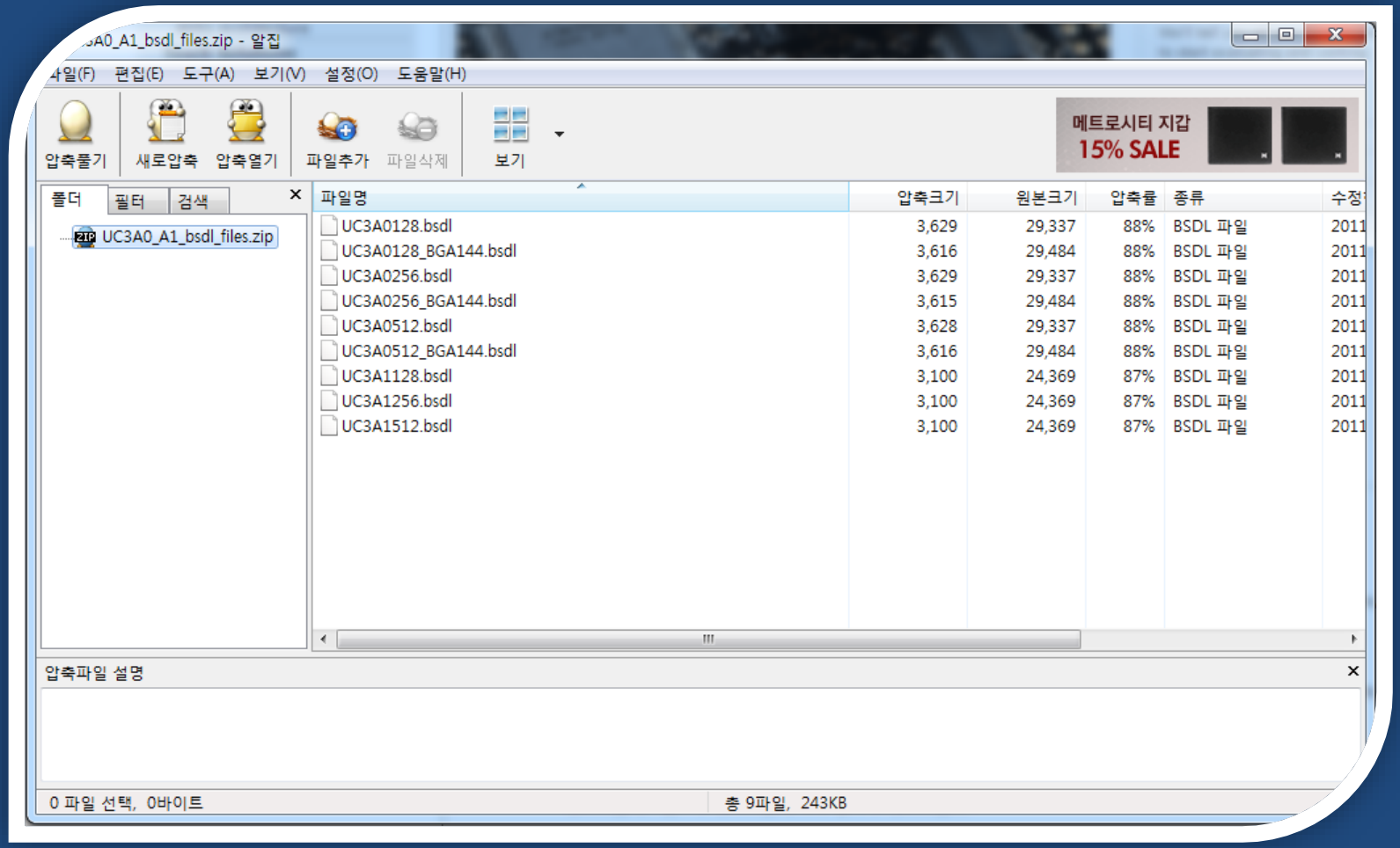
- Boundary-scan Description Language Files (BSDL)
- Boundary Scan order 정보를 담고 있는 공식 파일 포맷
- AVR : http://www.atmel.com/tools/avrbsdlfiles.aspx



» Technical Support
» What's Changed
» Mature Devices

...ftware	Description
	AVR UC3A0/A1 BSDL Files *(32KB, updated July 2014)* Atmel AVR UC3A Boundary-Scan Description files (BSD) for LQFP 144 packages, TQFP 100 packages, TFBGA 144 packages, and VFBGA 100 packages.
	AVR UC3A3/A4 BSDL Files *(64KB, updated December 2012)* Atmel AVR UC3A3/A4 Boundary-Scan Description files (BSD) for LQFP 144 packages, TFBGA 144 packages, and VFBGA 100 packages.
	AVR UC3B0/B1 BSDL Files *(18KB, updated March 2013)* Atmel AVR UC3B0/B1 Boundary-Scan Description files (BSD) for TQFP 48 and 64 packages. As well as QFN 64 and 48 packages.
	AVR UC3C0/C1/C2 BSDL Files *(39KB, updated December 2012)* Atmel AVR UC3C0/C1/C2 Boundary-Scan Description files (BSD) for LQFP 144 packages, TQFP 100 packages, TQFP 64 packages, and QFN 64 packages.
	AT32UC3L BSDL Files *(20583, updated May 2011)* AT32UC3A Boundary-Scan Description files (BSD) for LQFP 48 packages, QFN 48 packages, and TLLGA 48 packages.
	AVR XMEGA BSDL files *(49595, updated October 2009)* Zip archive with BSDL files for Atmel AVR XMEGA devices.
	megaAVR BSDL files *(79760, updated October 2009)* Zip archive with BSDL files for Atmel megaAVR devices.

Atmega128 BSDL



Atmega128 BSDL

```
205 attribute COMPONENT_CONFORMANCE of UC3A0128 : entity is " STD_1149_1_2001 ";
206
207 attribute PIN_MAP              of UC3A0128 : entity is PHYSICAL_PIN_MAP ;
208
209 constant  VQFP144 : PIN_MAP_STRING:=
210  "  ADVREF     : 82   , " &
211  "  ADVREFN    : 83   , " &
212  "  DM         : 70   , " &
213  "  DP         : 71   , " &
214  "  GNDCORE00  : 8    , " &
215  "  GNDCORE03  : 28   , " &
216  "  GNDCORE06  : 52   , " &
217  "  GNDCORE09  : 94   , " &
218  "  GNDCRNBR   : 109  , " &
219  "  GNDCRNUL   : 37   , " &
220  "  GNDIO00    : 18   , " &
221  "  GNDIO03    : 72   , " &
222  "  GNDIO06    : 117  , " &
223  "  PA00       : 25   , " &
224  "  PA01       : 27   , " &
225  "  PA02       : 30   , " &
226  "  PA03       : 32   , " &
```

실습 문제

- 래빗 보드 전 핀의 현재 상태를 가져오는 JTAG 툴을 개발해 보세요.

핀들의 값 제어하기

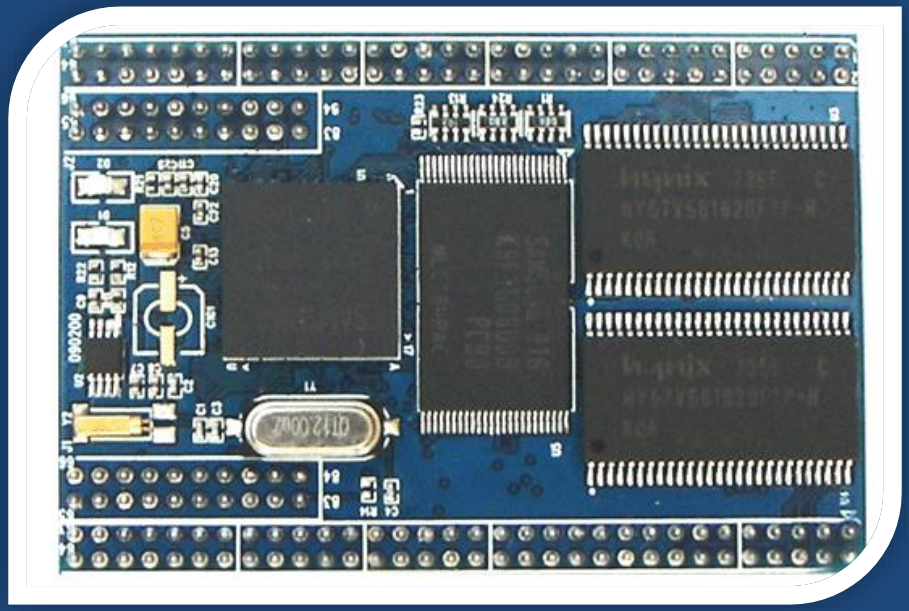
핀들의 값 제어하기

- 앞서 얻어낸 비트 배열을 가지고 있는다.
- 제어를 원하는 비트를 변조한다.
- IR => EXTEST 모드로 변경한다.
- Shift-DR : 204개의 새로운 패턴을 밀어 넣는다.
- Update-DR을 해준다.

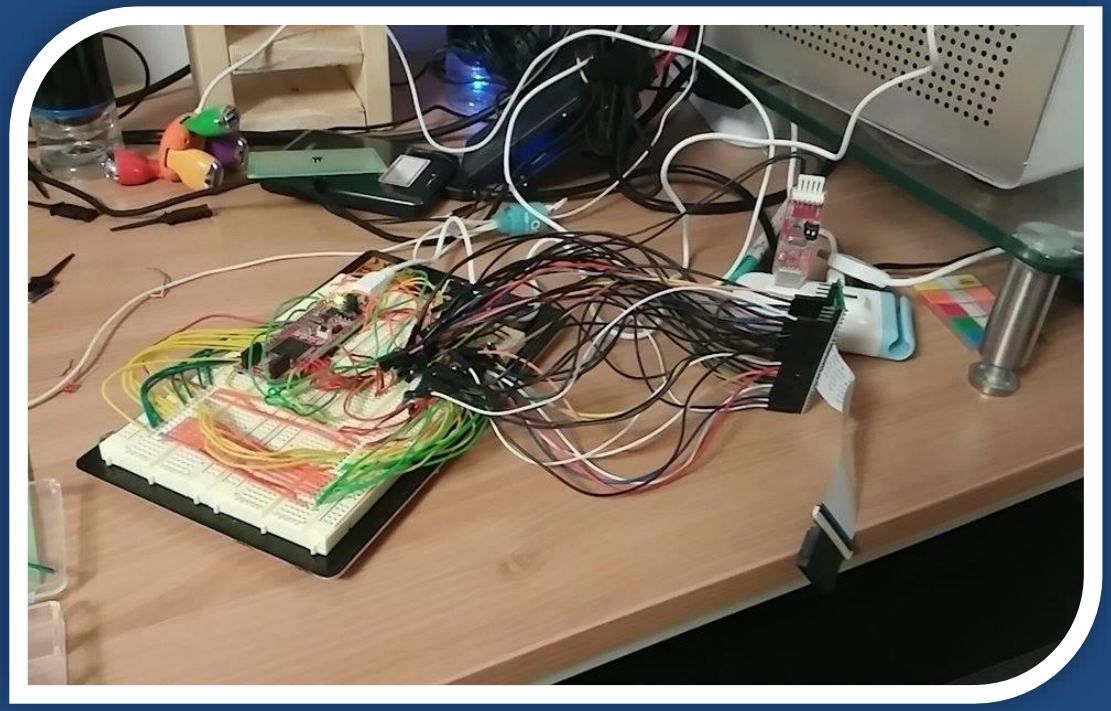
JTAG을 이용한 Firmware 획득

일반적인 Firmware 획득

- Flash memory를 PCB에서 desoldering한다.

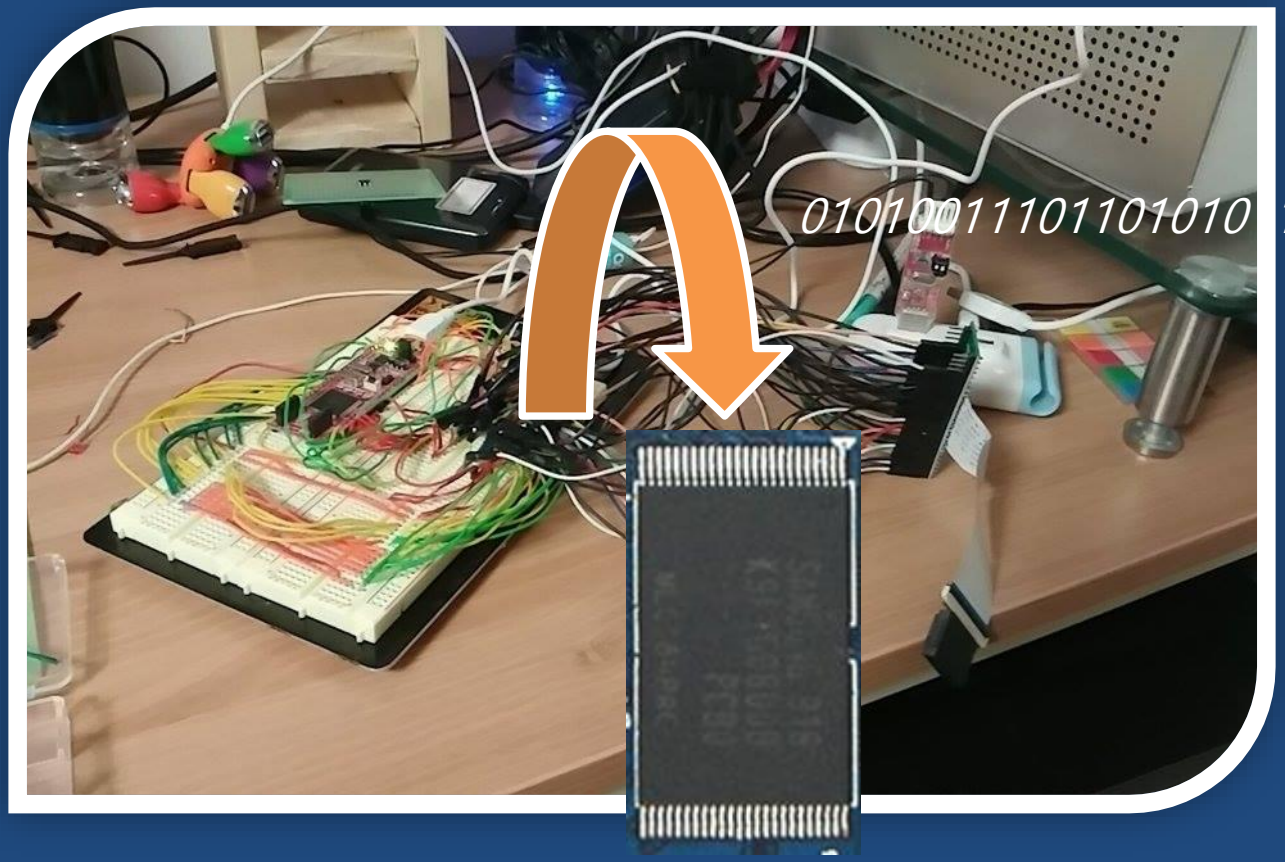


일반적인 Firmware 획득



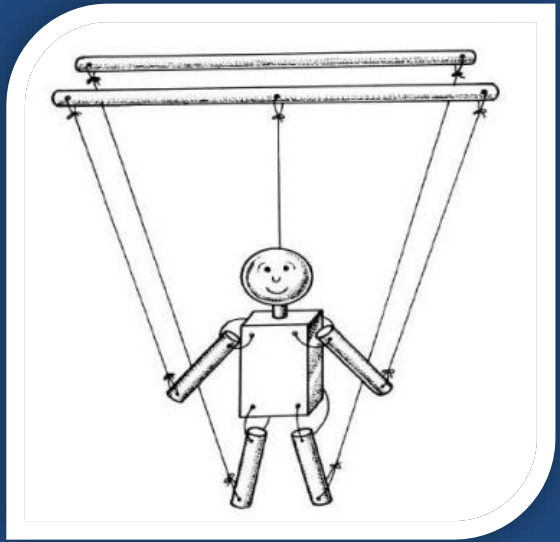
Rom writer 혹은
직접 개발한 도구를 이용하여
Firmware를 획득한다.

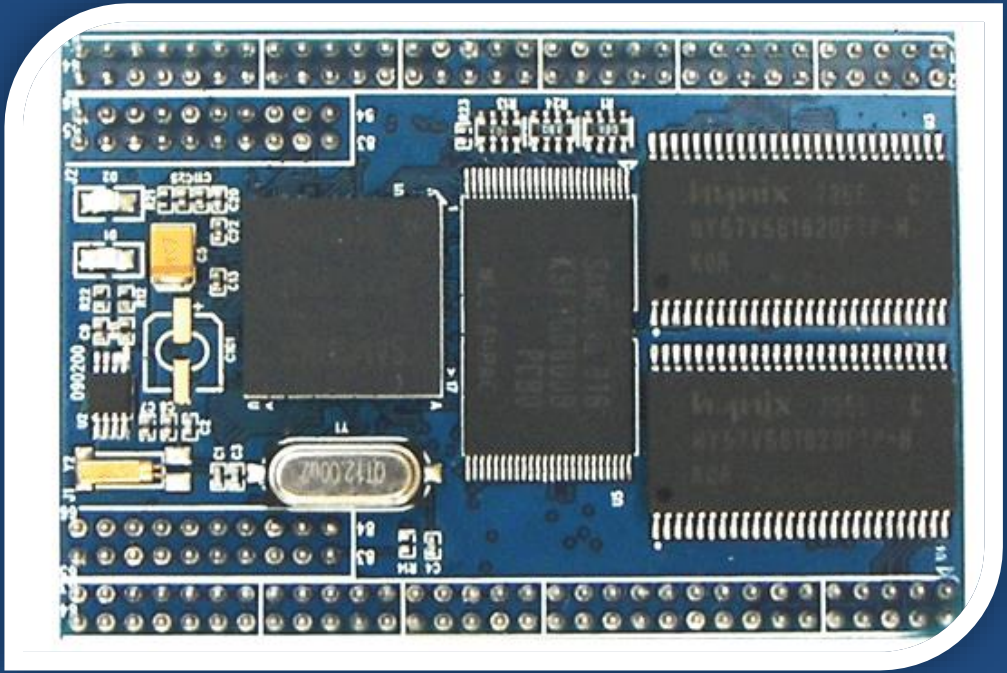
일반적인 Firmware 획득



Flash memory 프로토콜에 맞게 전기신호를 전송한다.

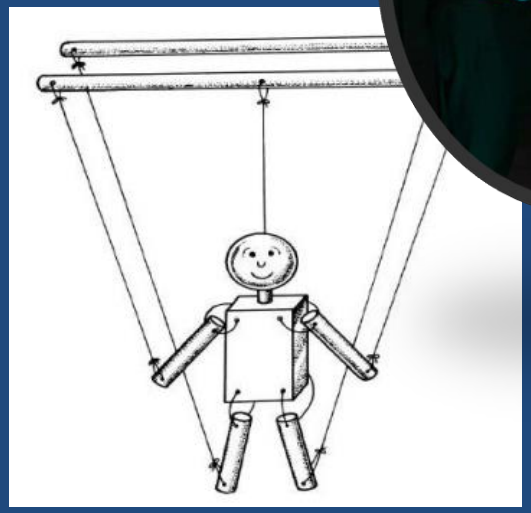
JTAG을 이용한 Flash Memory 덤프

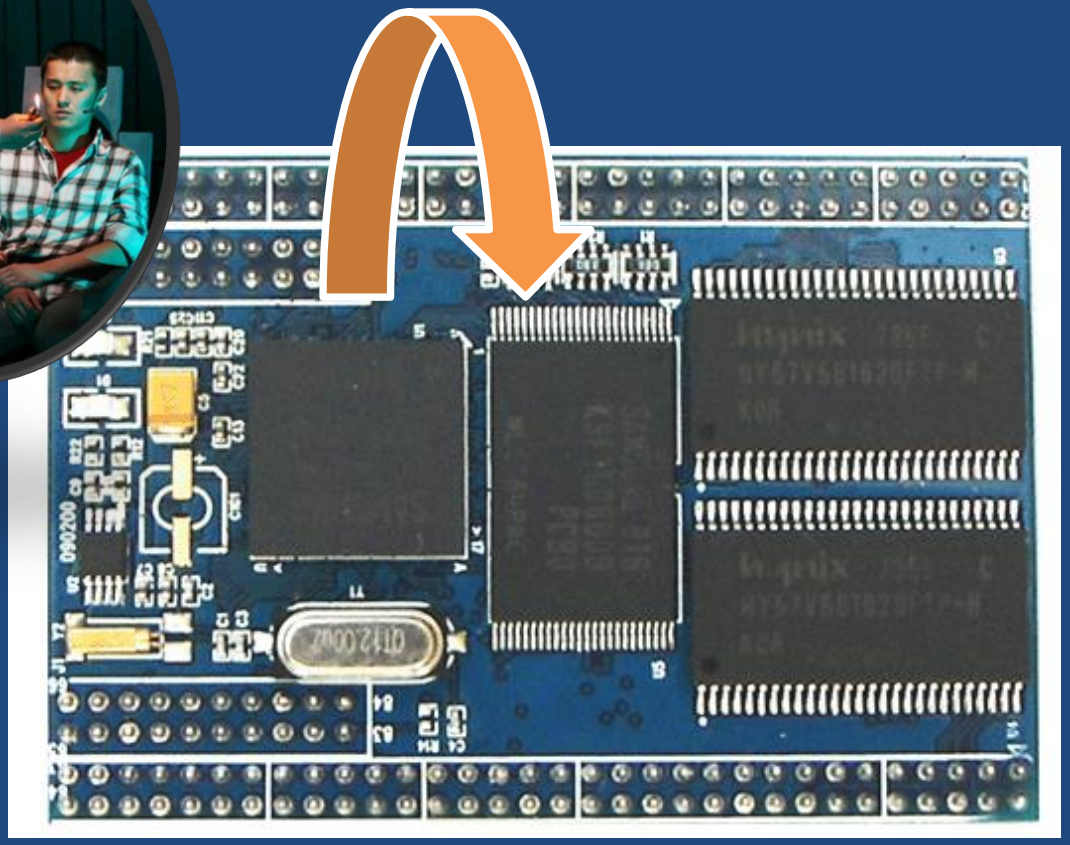




JTAG을 이용한 Flash Memory 덤프

0101001110110101011 1~~



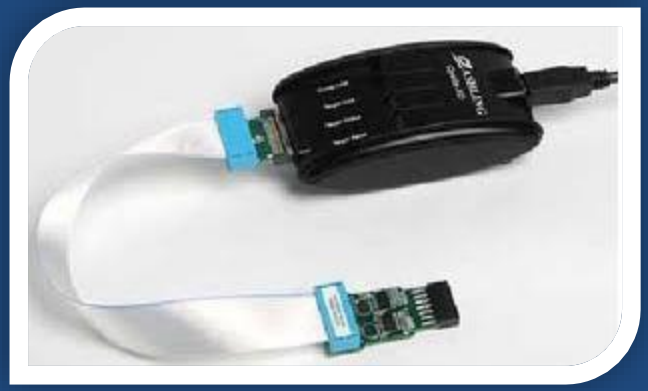


CPU야 너는 이제부터
Flash dumper다~

CPU별 JTAG 장비들 예시

ARM용

AVR용



MIPS용

ARM용 JTAG (JK JTAG)

QUICK

[JK전자] **ARM-JTAG Pro**

ARM-JTAG USB 제품은 ARM Core(ARM7/9/11, Cortex-M3, M4, A8)등을 위한 디버깅 및 플래시 프로그램이 가능한 USB JTAG 에뮬레이터 제품

780,000 원 적 0원

› 장바구니

› 관심상품

QUICK

[JK전자] **ARM-JTAG Standard**

ARM-JTAG USB 제품은 ARM Core(ARM7/9/11, Cortex-M3, M4, A8)등을 위한 디버깅 및 플래시 프로그램이 가능한 USB JTAG 에뮬레이터 제품

420,000 원 적 0원

› 장바구니

› 관심상품

QUICK

[JK전자] **ARM-JTAG Light**

ARM-JTAG USB 제품은 ARM Core(ARM7/9/11, Cortex-M3, M4, A8)등을 위한 디버깅 및 플래시 프로그램이 가능한 USB JTAG 에뮬레이터 제품

100,000 원 적 0원

› 장바구니

› 관심상품

ARM-JTAG Edition		Light	Standard	Professional
Speed	Download	750 KB/S	750 KB/S	750 KB/S
	Upload	550 KB/S	550 KB/S	550 KB/S
Support Core	ARM-7	YES	YES	YES
	ARM-9	NO	YES	YES
	CORTEX-M3	YES	YES	YES
	CORTEX-M4	YES	YES	YES
	ARM11	NO	YES	YES
	CORTEX-A8	NO	NO	YES
Support IDE		ADS1.2, RVDS, KEIL/MDK, IAR		
Support OS		WIN2000 / XP / VISTA / WINDOWS 7		

ARM용 JTAG (H-JTAG)

QUICK

[JK전자] **H-JTAG USB Pro** [상품코드:22164]

H-JTAG USB 제품은 ARM Core(S3C2440, S3C6410, XScale, Cortex-M3)등을 위한 디버깅 및 플래시 프로그램이 가능한 USB JTAG 에뮬레이터 제품 입니다.

~~600,000 원~~
540,000 원

준비 2~3일
m.o.q 1
단위 1

1
› 장바구니
› 관심상품

QUICK

[Jk전자] **H-JTAG USB Standard** [상품코드:15635]

ARM Core를 위한 디버깅 및 플래시 프로그램이 가능한 USB JTAG 에뮬레이터 제품 입니다.

~~245,000 원~~
198,000 원

준비 2~3일
m.o.q 1
단위 1

1
› 장바구니
› 관심상품

H-JTAG Edition		Light	Standard	Professional
Speed	Download	LPT x 5	750 KB/S	750 KB/S
	Upload		550 KB/S	550 KB/S
ARM Core	ARM-7	YES	YES	YES
	ARM-9	YES	YES	YES
	CORTEX-M3	YES	YES	YES
	XSCALE	NO	YES	YES
	ARM11	NO	NO	YES
	PXA3XX	NO	NO	YES
Support IDE		ADS1.2, RVDS, KEIL/MDK, IAR		
Support OS		WIN2000 / XP / VISTA / WINDOWS 7 (각 32비트, 64비트 운영체제에서 사용가능)		

ARM용 JTAG (J-Link)

	J-Link BASE	J-Link PLUS	J-Link ULTRA+	J-Link PRO	J-Trace for Cortex-M	J-Trace ARM
USB	yes	yes	yes	yes	yes	yes
Ethernet	no	no	no	yes	no	no
Supported cores	ARM7/9/11, Cortex-A5/A8/A9/R4, Cortex-M0/M0+/M1/M3/M4, Renesas RX				Tracing: Cortex-M3/M4 No tracing: ARM7/9/11, Cortex-M0/M0+/M1 Cortex-A5/A8/A9/R4	ARM 7/9
JTAG	yes	yes	yes	yes	yes	yes
SWD	yes	yes	yes	yes	yes	no
SWO	yes	yes	yes	yes	yes	no
ETM Trace	no	no	no	no	yes	yes

JTAG을 이용한 갤럭시S1 Firmware 획득 실습

갤럭시S1 Spec

ARM Cortex A8 : 이에 맞는 JTAG 장비 필요

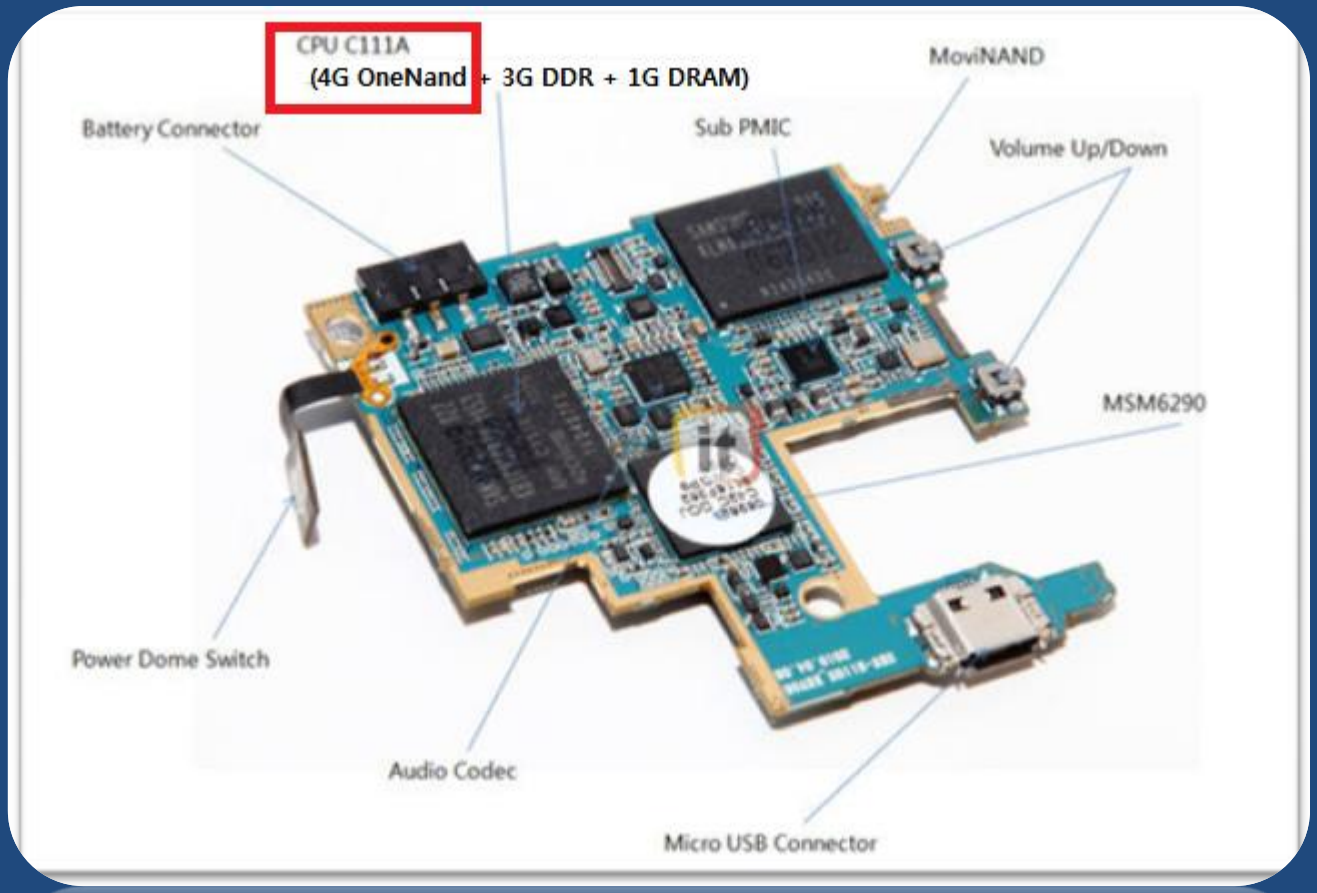
갤럭시S 스펙

그렇다며, 현존 최강의 갤럭시S의 스펙은 어떨까요? 해외에 공개된 갤럭시S의 스펙은 아래와 같습니다.

모델 Samsung Galaxy S
프로세서 ARM Cortex A8 @ 1 Ghz - Hummingbird
메모리 8GB, 16GB (32GB까지 장착)
램 512M
크기 122 x 64 x 10 mm
무게 118g
네트워크 2.5G (GSM/ GPRS/ EDGE) : 850/ 900/ 1800/ 1900 MHz 3G (HSDPA 7.2Mbps, HSUPA 5.76Mbps) : 900/ 1900/ 2100 MHzGPS 장착
GPS 장착
와이파이 장착(b/g/n)
블루투스 3.0 가능
밧데리 1500 mAh
카메라 500백만 화소, HD720p 재생 및 녹화 가능
디스플레이 4.0" WVGA SUPER AMOLED (800x480) with mDNIe
운영체제 Android Android 2.1(2.2 프로요로 업데이트 예정)

출처 : http://www.frandroid.com/18192/prise-en-main-du-samsung-galaxy-s-i9000/

갤럭시 S1의 하드웨어 구성



갤럭시S의 저장장치 구성

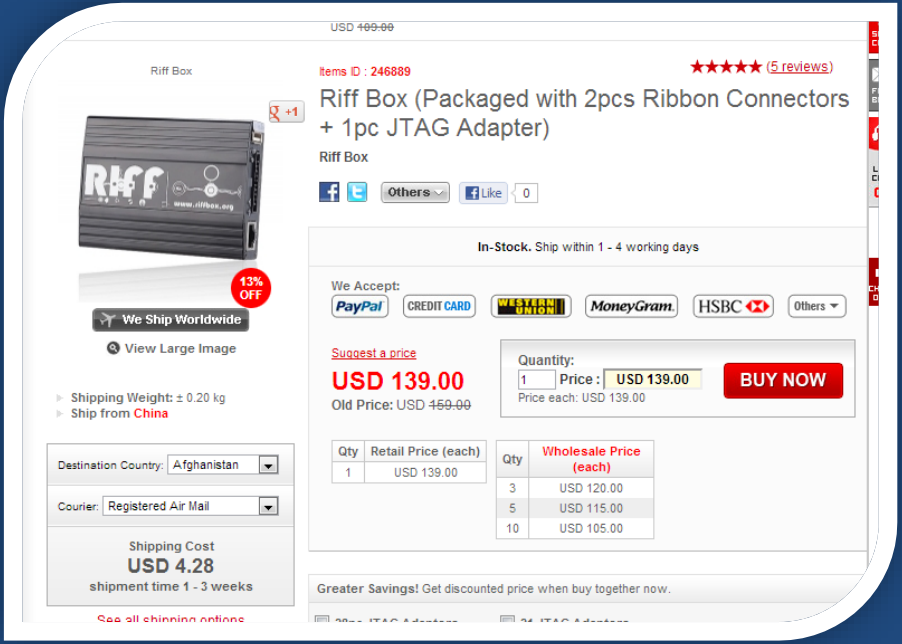
- oneNand
  - 여러 개의 파티션으로 분할
    - /system/ (300메가)
    - /dbdata/ (100메가)
    - /cache/ (30메가)
    - ...
- moviNand
  - 2개의 파티션으로 분할
    - 내장 SD카드(14기가)
    - /data/ (2기가)
  - /dev/block/mmcXXX

실습 장비 소개

- RIFFBOX JTAG

무적의 RIFFBOX JTAG

- 핸드폰, 스마트폰 복구 용도의 JTAG
- 일반 디버깅 용도로도 사용 가능
- 상대적으로 저렴하다.



RIFFBOX JTAG 호환 목록

rently supported memory controllers are:

- OneNAND Memory (connected directly to the MCU's address space);
- CFI Compliant NOR Memory with CFI Command sets 0×0001, 0×0002, 0×0200 and 0×0003;
- NAND Controller in MSM6250, MSM6250A;
- NAND Controller in QSC6055, QSC6085, QSC6240, QSC6270;
- NAND Controller in MDM6085, MDM6200, MDM6600;
- NAND Controller in MSM6245, MSM6246, MSM6270, MSM6275, MSM6280, MSM6280A, MSM6281, MSM6290, MSM6800A, MSM6801A;
- NAND Controller and OneNAND Controller in MSM7225, MSM7227, MSM7625, MSM7627;
- NAND Controller in MSM7200, MSM7200A, MSM7201A, MSM7500, MSM7500A, MSM7501A, MSM7600;
- NAND Controller in QSD8250, QSD8650;
- eMMC Controller #2 in MSM7230, MSM8255, MSM8255T;
- eMMC Controller #0 in S5PV310;

Currently supported chipsets and cores for JTAG I/O operations:

- Generic ARM Cores: ARM7, ARM9 (ARM920, ARM926, ARM946), ARM11, CORTEX-A8,CORTEX-A9;
- Qualcomm QSC Family: QSC1100, QSC1110, QSC6010, QSC6020, QSC6030, QSC6055, QSC6085, QSC6240, QSC6270;
- Qualcomm MSM Family: MSM6000, MSM6150, MSM6245, MSM6246, MSM6250, MSM6250A, MSM6260, MSM6275, MSM6280, MSM6280A, MSM6281, MSM6800A, MSM6801A, MSM6290, MSM7225, MSM7227, MSM7625, MSM7627, MSM7230, MSM8255, MSM8255T, MSM8260;
- Qualcomm QSD Family: QSD8250, QSD8650;
- Qualcomm ESM Family: ESM7602A;
- Qualcomm MDM Family: MDM6085 MDM6200, MDM6600;
- OMAP Family: OMAP1710, OMAP3430, OMAP3630, OMAP4430;
- NVIDIA Family: TEGRA2;
- Marvell/XScale Family: PXA270, PXA271, PXA272, PXA310, PXA312, PXA320.
- Samsung Processors: S5P6422, S5PV310.

RIFFBOX JTAG 호환 목록

Samsung Phones:

- Samsung B7330 Boot Repair/Unbrick
- Samsung C5510 Boot Repair/Unbrick
- Samsung F480 Boot Repair/Unbrick
- Samsung F500 modem Boot Repair/Unbrick
- Samsung G810 modem Boot Repair/Unbrick
- Samsung G810 PDA Boot Repair/Unbrick
- Samsung i450 modem Boot Repair/Unbrick
- Samsung i450 PDA Boot Repair/Unbrick
- Samsung i550 modem Boot Repair/Unbrick
- Samsung i550 PDA Boot Repair/Unbrick
- Samsung i560 PDA Boot Repair/Unbrick
- Samsung i568 PDA Boot Repair/Unbrick
- Samsung i560 modem Boot Repair/Unbrick
- Samsung i568 modem Boot Repair/Unbrick
- Samsung i5500 Galaxy Boot Repair/Unbrick
- Samsung i5510 Galaxy Boot Repair/Unbrick
- Samsung I5700 Boot Repair/Unbrick
- Samsung i5800 Boot Repair/Unbrick
- Samsung i710 Boot Repair/Unbrick
- Samsung i718 Boot Repair/Unbrick
- Samsung i740 PDA Boot Repair/Unbrick
- Samsung i740 Modem Boot Repair/Unbrick
- Samsung I7500 Boot Repair/Unbrick
- Samsung i780 modem Boot Repair/Unbrick
- Samsung i780 PDA Boot Repair/Unbrick
- Samsung i897 Captivate Boot Repair/Unbrick
- Samsung I8000 Boot Repair/Unbrick
- Samsung i8320 modem Boot Repair/Unbrick

LG RD3000 Repair, Read/Write NOR flash

- LG RD3100 Repair, Read/Write NOR flash
- LG RD3500 Repair, Read/Write NOR flash
- LG RD3510 Repair, Read/Write NOR flash
- LG RD3540 Repair, Read/Write NOR flash
- LG RD3610 Repair, Read/Write NOR flash
- LG RD3640 Repair, Read/Write NOR flash
- LG RD6100 Repair, Read/Write NOR flash
- LG LSP450T Repair, Read/Write NOR flash
- LG VM265 Repair, Read/Write NOR flash
- LG TD636 Repair, Read/Write NOR flash
- ZTE-C N600 Repair, Read/Write NAND flash
- ZTE S100 Repair, Read/Write NAND flash
- ZTE F285 Repair, Read/Write NAND flash
- ZTE Striker (S1602) Repair, Read/Write NAND flash
- ZTE C370 Repair, Read/Write NOR flash
- ZTE C355 Repair, Read/Write NOR flash
- ZTE WP829 Repair, Read/Write NOR flash
- ZTE R518 Repair, Read/Write NOR flash
- ZTE AC2776 Repair, Read/Write NOR flash
- ZTE AC2746 Repair, Read/Write NOR flash
- ZTE F290 Repair, Read/Write NOR flash
- ZTE S185 Repair, Read/Write NOR flash
- ZTE S170 Repair, Read/Write NOR flash
- ZTE S165 Repair, Read/Write NOR flash
- ZTE FWPC26 Repair, Read/Write NOR flash
- HTC Droid Eris A6376 Boot Repair, ModelID repair
- HTC Droid Incredible ADR6300 (HTC Incredible C) Unbrick
- HAIER C2076 Repair, Read/Write NAND flash
- HAIER C300 Repair, Read/Write NOR flash

OneNAND 제어 가능

Here are ARM cores currently supported by the RIFF BOX 2 JTAG firmware:

- – ARM926
- – ARM920T
- – ARM1136
- – PXA312
- – PXA270
- – OMAP3x (has TAP Router module)

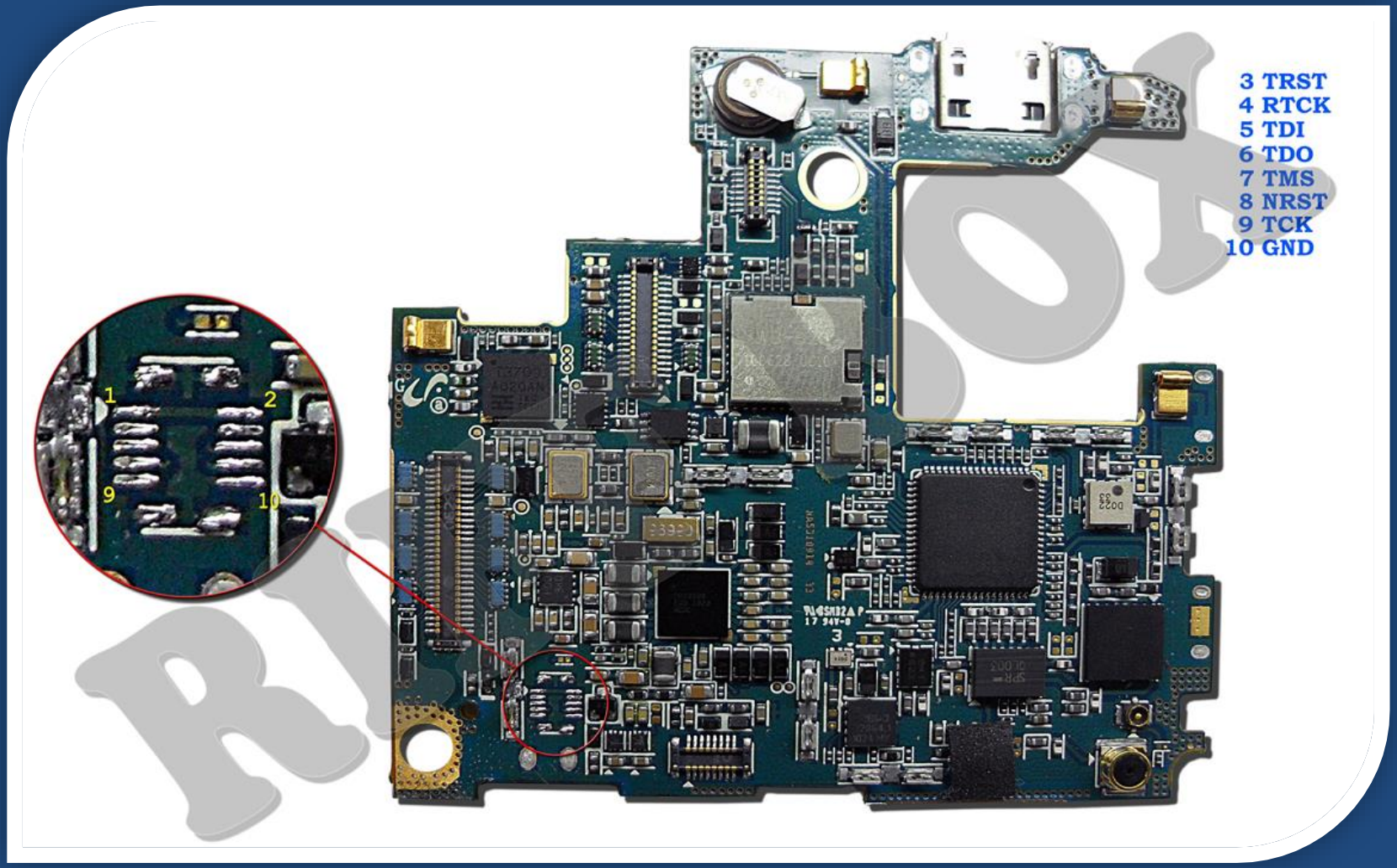
Supported chipsets based on those ARM cores with RIFF DCC Loader functionality (that is NAND memory operations through custom chip's NAND controller):

- – Intel XScale PXA312
- – Intel XScale PXA270
- – Qualcomm MSM62xx (except MSM6250x group)
- – Qualcomm MSM6250x
- – Samsung S3C2440
- – Samsung S3C6410
- – Broadcomm BCM21xxx;
- – OneNAND (not chipset, but still...).

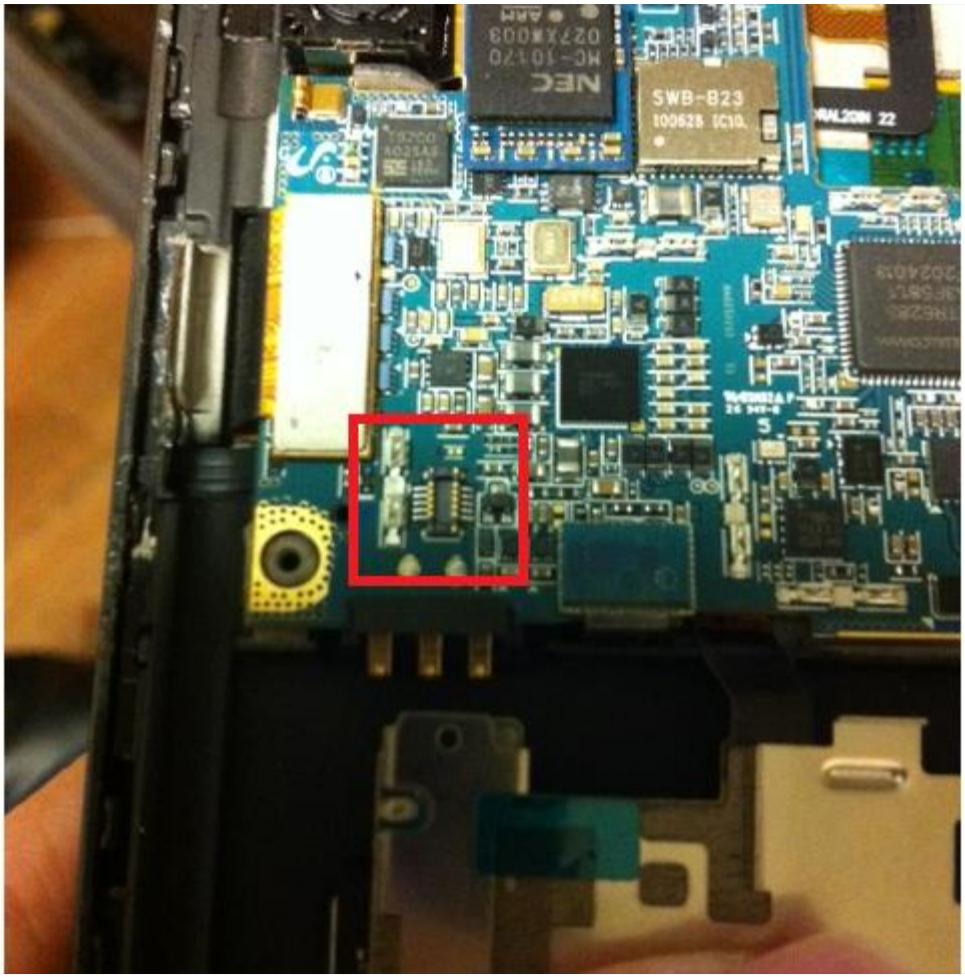
In short, if you have a device in hands which has supported chipset inside and if this chipset's core belongs to the supported ARM cores list, then you can connect and read/write memory of your device over JTAG link.

갤럭시S1 JTAG 포트 찾기

갤럭시S1 JTAG 포트 찾기



갤럭시S1 JTAG 포트 찾기

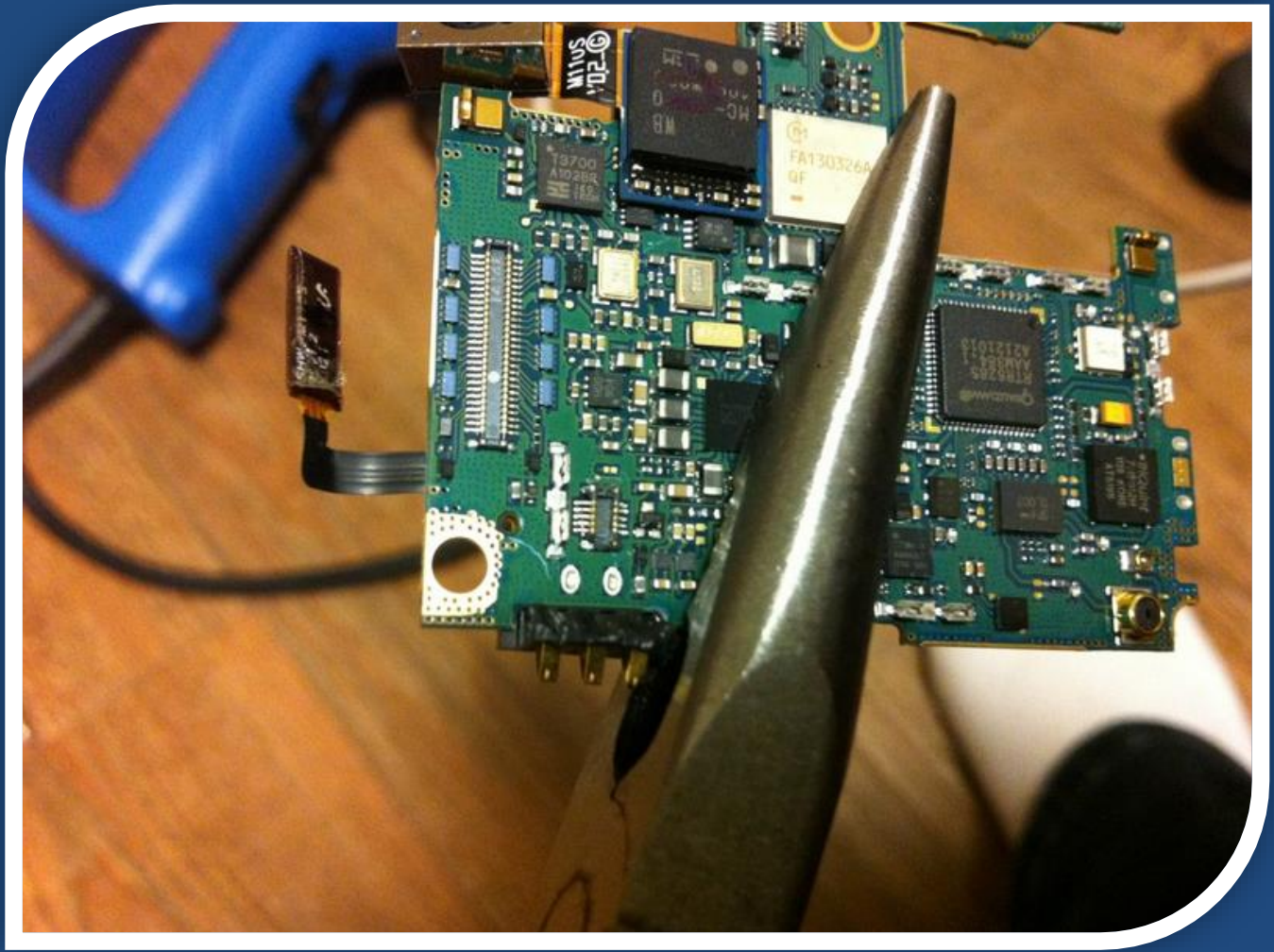


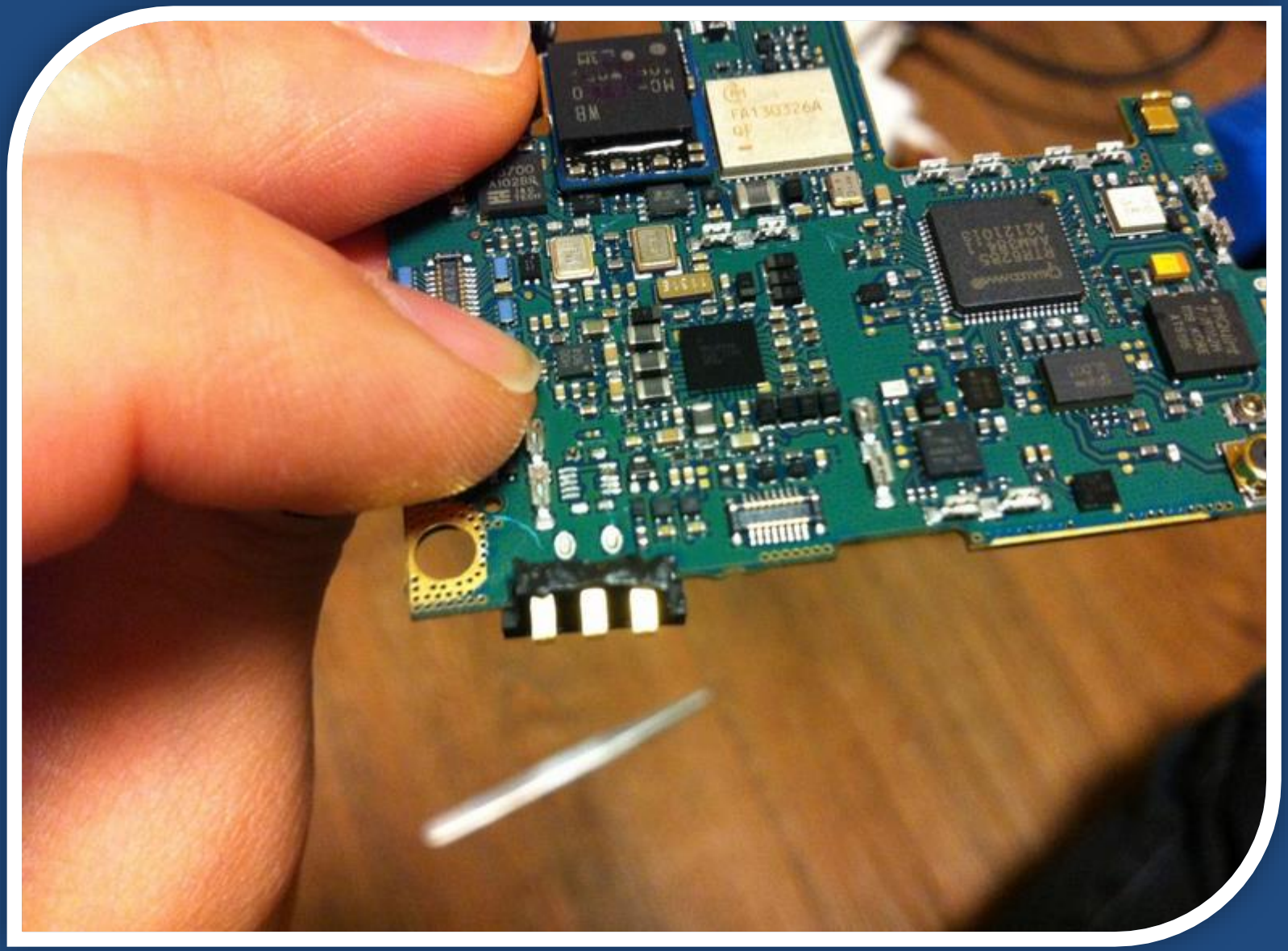
어떻게 연결하지?

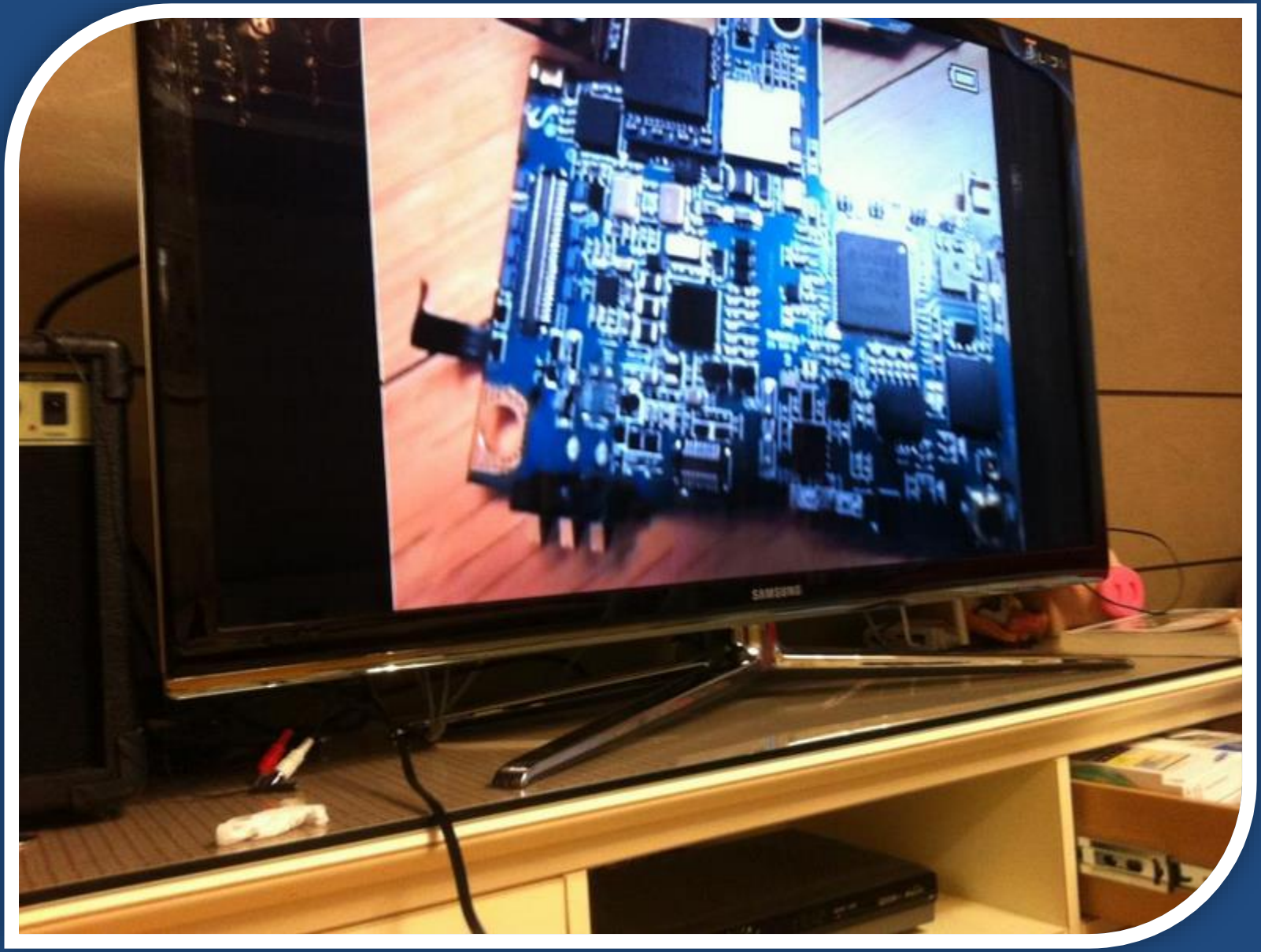
- 1. 납땜
  - 실납
  - 확대 도구
  - 본인은 세 번의 실패 후에 포기하였습니다..

- 2. 전용 JIG 이용
  - 해외 쇼핑몰에서 구매 가능
  - http://www.aliexpress.com/item/GPG-33-IN-1-EASY-JIGS-33-IN-1-GPG-EASY-JIGS/1708309400.html

JTAG 납땜 시도 이야기




FA130326A

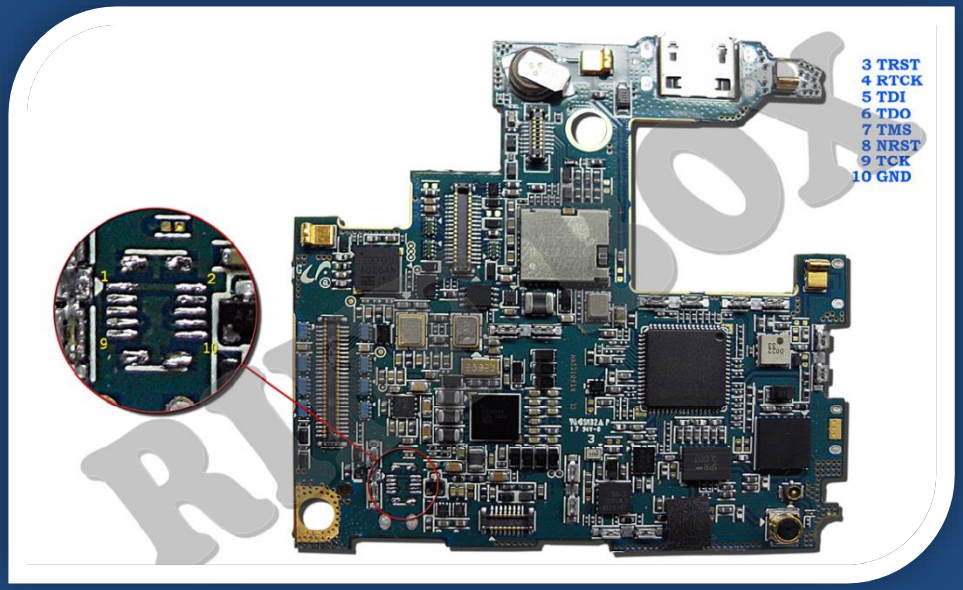

SAMSUNG

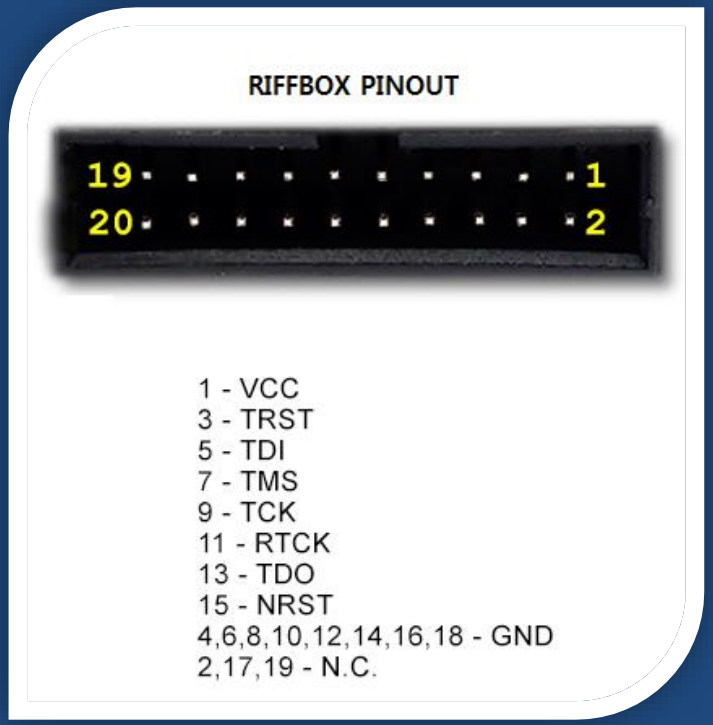
JTAG 전용 JIG

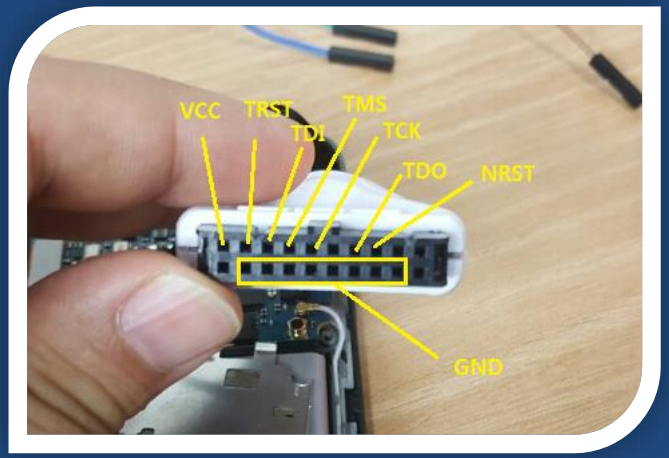
3 IN 1 GPG EASY JIGS Riff Z3x Pro Easy jtag
Store No: 1113184
http://www.aliexpress.com/store/111318

갤럭시 S1 <-> RIFFBOX 연결

갤럭시 S1 <-> RIFFBOX 연결

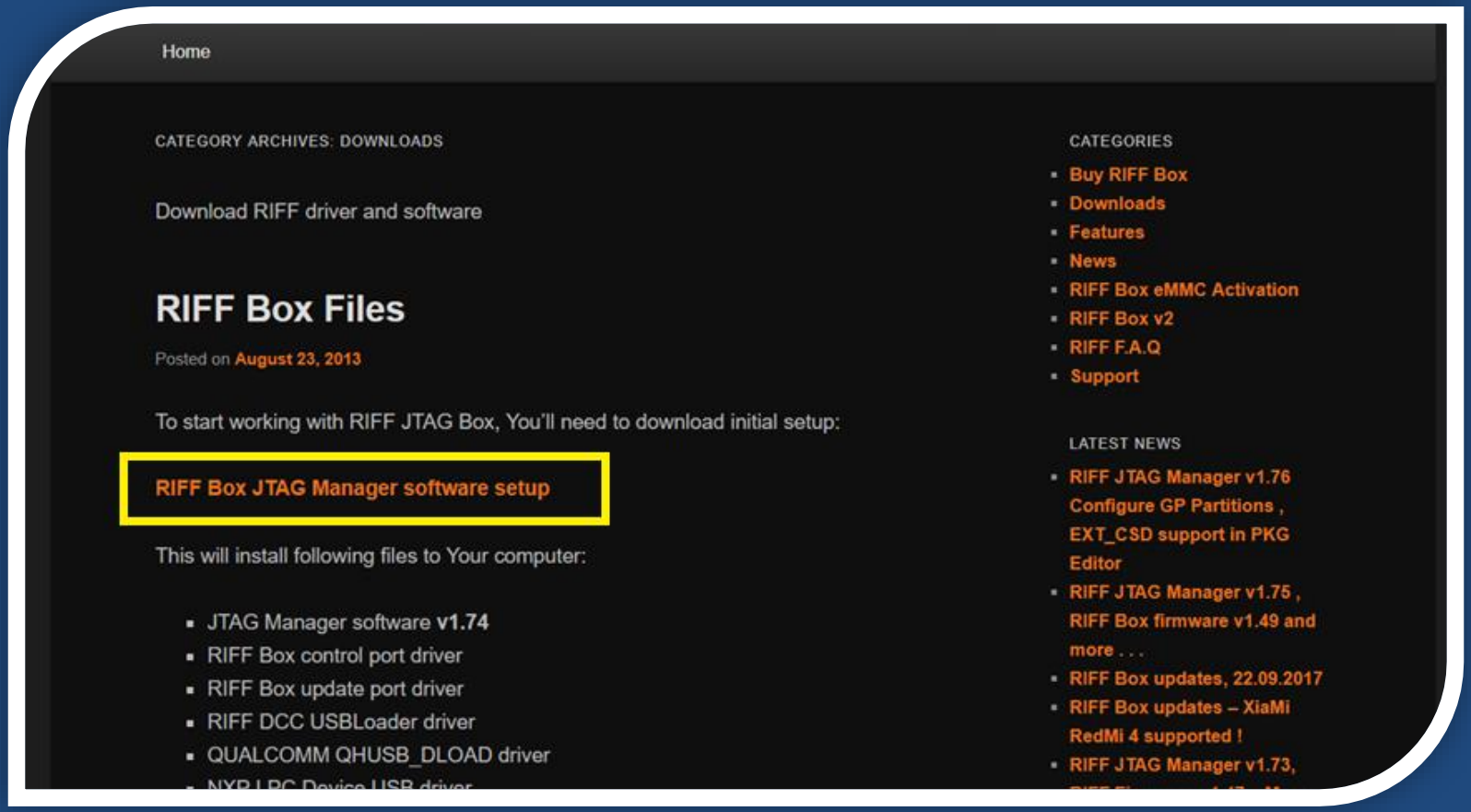






RIFFBOX JTAG 설치

- http://www.riffbox.org/

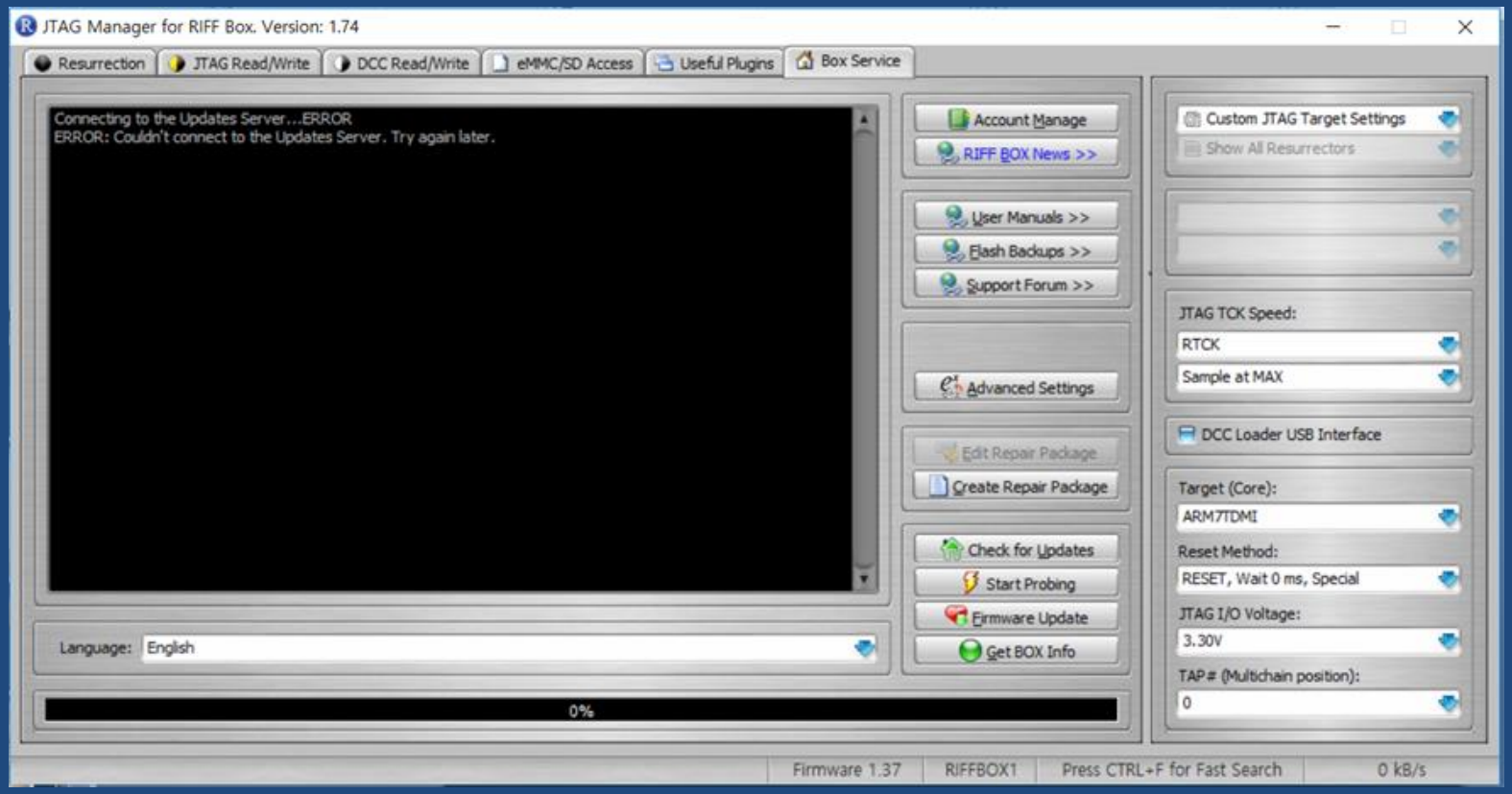


RIFFBOX JTAG 설치

- Drive 미인식 시 수동 설치
  - ₩RIFF JTAG Manager₩Drivers₩

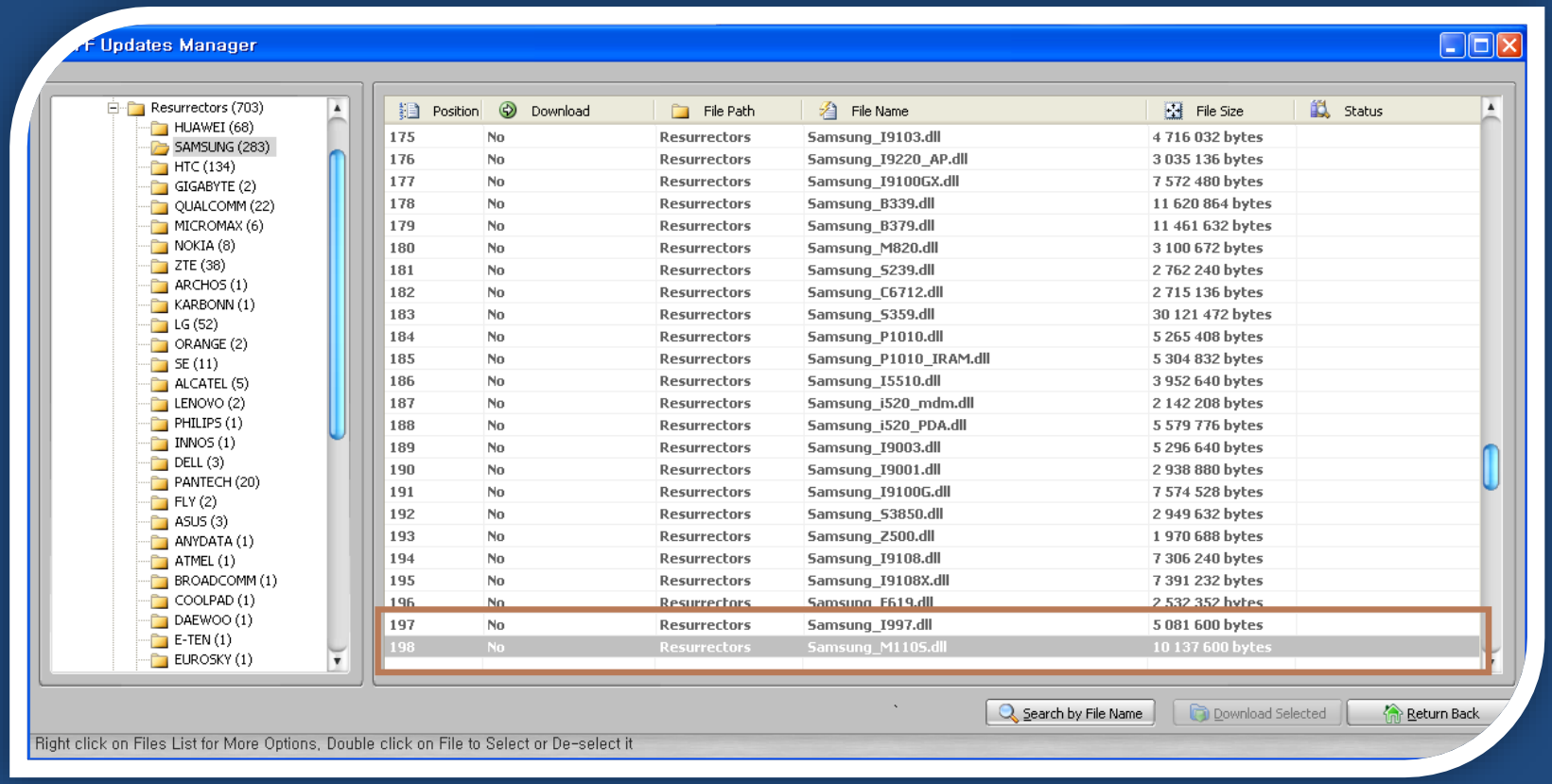
RIFFBOX JTAG 설치

- RIFF Box JTAG Manager 실행
  - Box Service -> Firmware Update

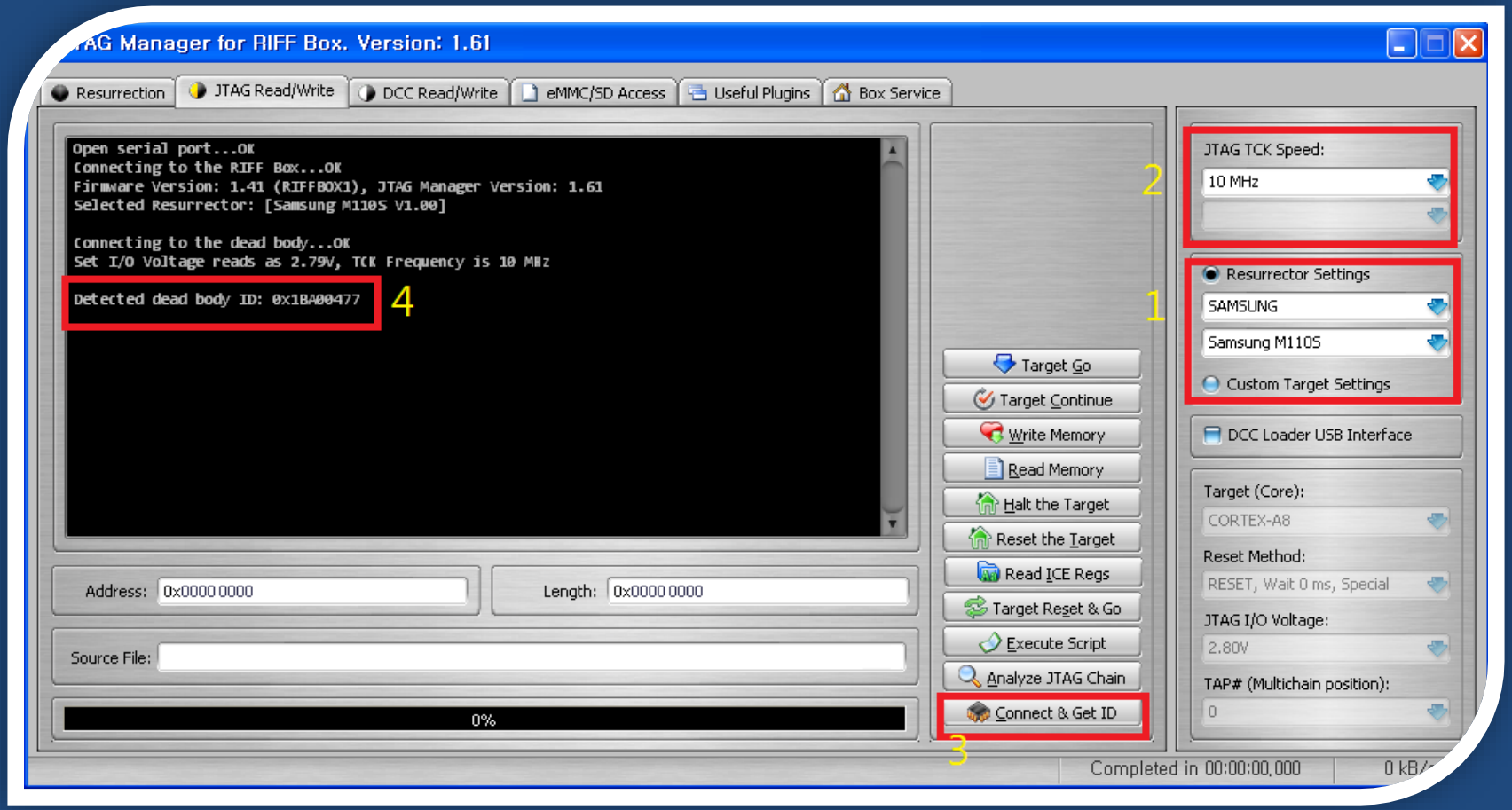


갤럭시 S1 (M110S) 모듈 설치

- Box Service -> Check for Updates

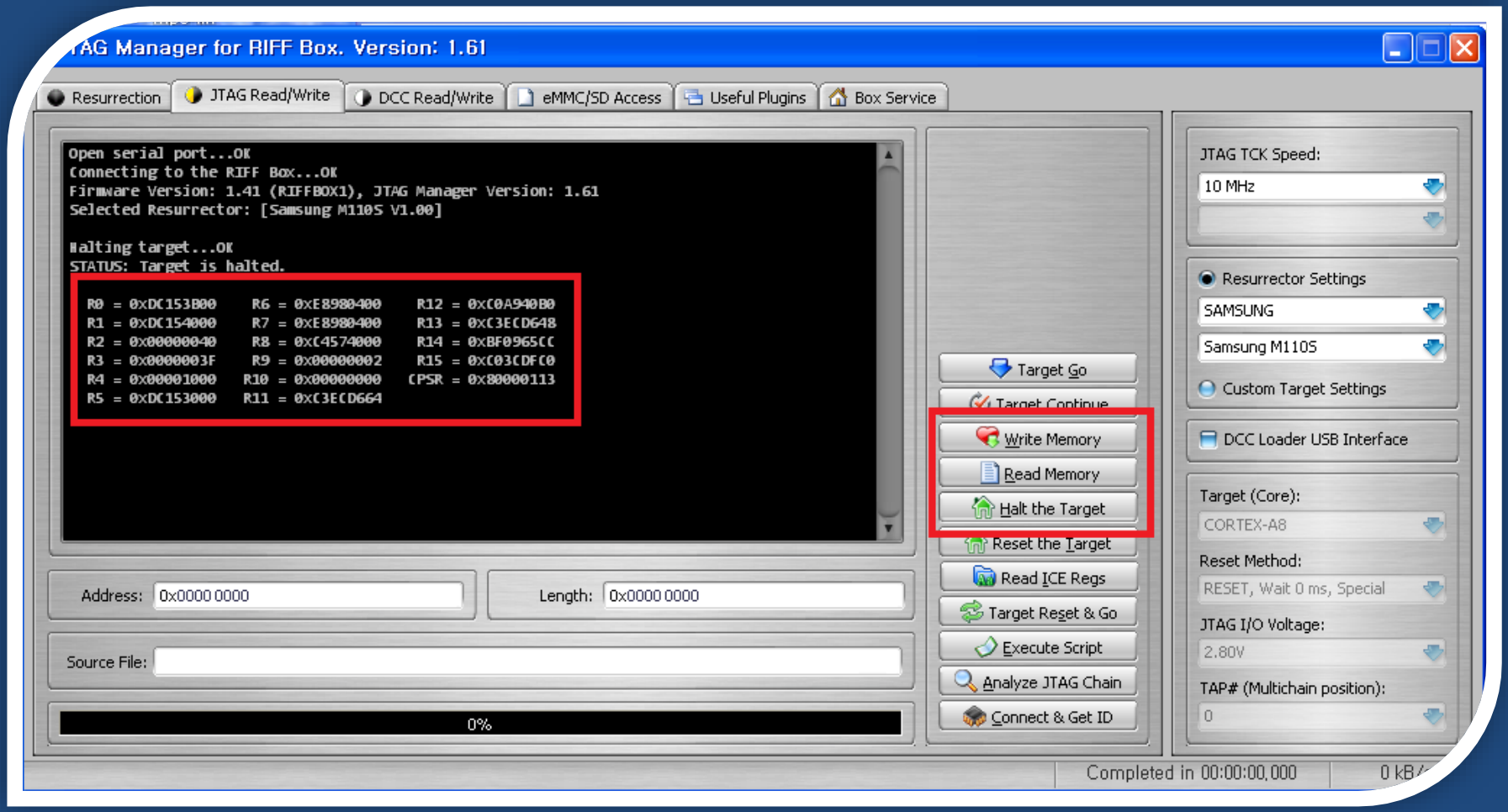


JTAG 연결 테스트



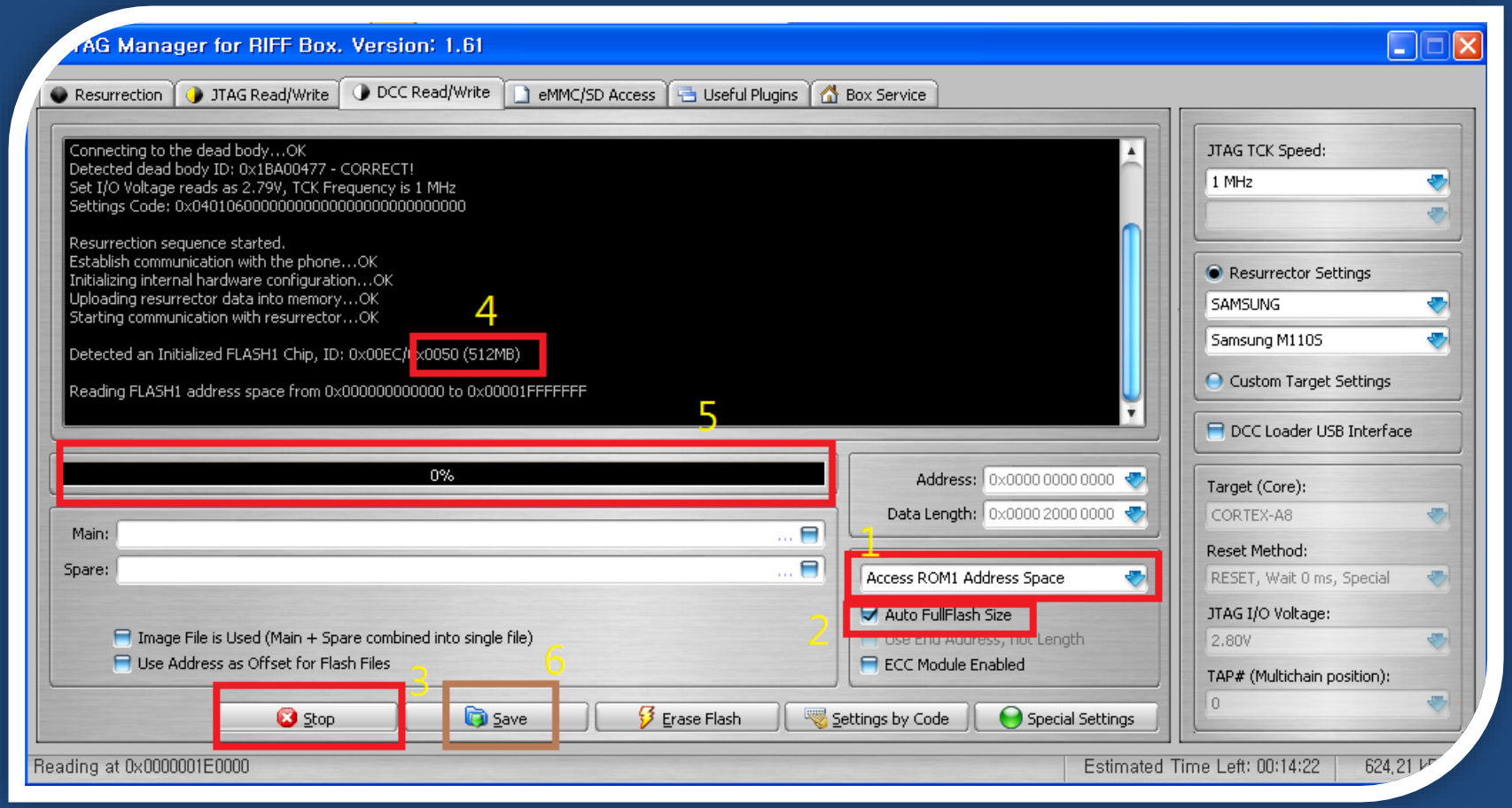
• JTAG connect 후에도 target device는 정상적으로 작동한다. (Halt the Target을 실행하지 않는 한)

메모리 읽기/쓰기 테스트



- Memory Read/Write는 올바른 주소를 Address에 먼저 설정해야 함
- Target Go도 마찬가지

펌웨어 덤프 실습


AG Manager for RIFF Box. Version: 1.61
Resurrection
JTAG Read/Write
DCC Read/Write
eMMC/SD Access
Useful Plugins
Box Service
Connecting to the dead body...OK
Detected dead body ID: 0x1BA00477 - CORRECT!
Set I/O Voltage reads as 2.79V, TCK Frequency is 1 MHz
Settings Code: 0x0401060000000000000000000000000
Resurrection sequence started.
Establish communication with the phone...OK
Initializing internal hardware configuration...OK
Uploading resurrector data into memory...OK
Starting communication with resurrector...OK
Detected an Initialized FLASH1 Chip, ID: 0x00EC/0x0050 (512MB)
Reading FLASH1 address space from 0x000000000000 to 0x00001FFFFFFF
4
5
0%
Main:
Spare:
Image File is Used (Main + Spare combined into single file)
Use Address as Offset for Flash Files
Address: 0x0000 0000 0000
Data Length: 0x0000 2000 0000
1
Access ROM1 Address Space
2
Auto FullFlash Size
ECC Module Enabled
3
6
Stop
Save
Erase Flash
Settings by Code
Special Settings
JTAG TCK Speed:
1 MHz
Resurrector Settings
SAMSUNG
Samsung M110S
Custom Target Settings
DCC Loader USB Interface
Target (Core):
CORTEX-A8
Reset Method:
RESET, Wait 0 ms, Special
JTAG I/O Voltage:
2.80V
TAP# (Multichain position):
0
Reading at 0x0000001E0000
Estimated Time Left: 00:14:22
624,21

덤프 된 펌웨어 파일의 포맷

- Samsung RFS(Robust File System)

문서 토론

향후 15년의 위키미디어의 미래를 논의하는 위키미디어 2030 전략 토론이 진행되고 있습니다.

RFS

위키백과, 우리 모두의 백과사전.

RFS(Robust File System)는 삼성에서 자체 개발한 플래시 메모리를 위한 파일 시스템이다. FAT32 파일시스템에 기반을 두고 있다. FAT32 에 저널링과 웨어 레벨링 기능을 추가했다.

이 글은 컴퓨터에 관한 토막글입니다. 서로의 지식을 모아 알차게 문서를 완성해 갑시다.

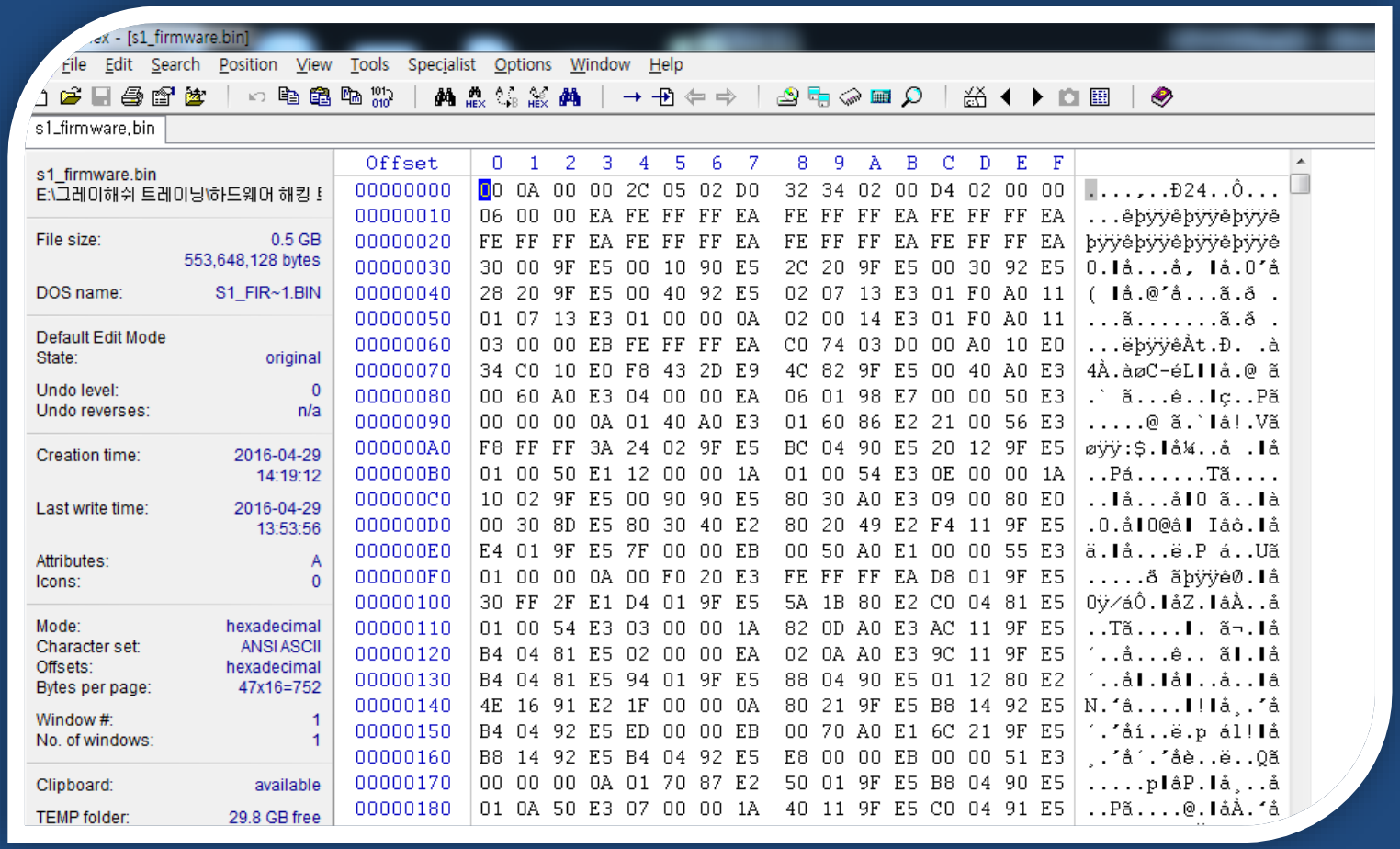
분류: 파일 시스템

이 문서는 2012년 5월 5일 (토) 19:54에 마지막으로 편집되었습니다.

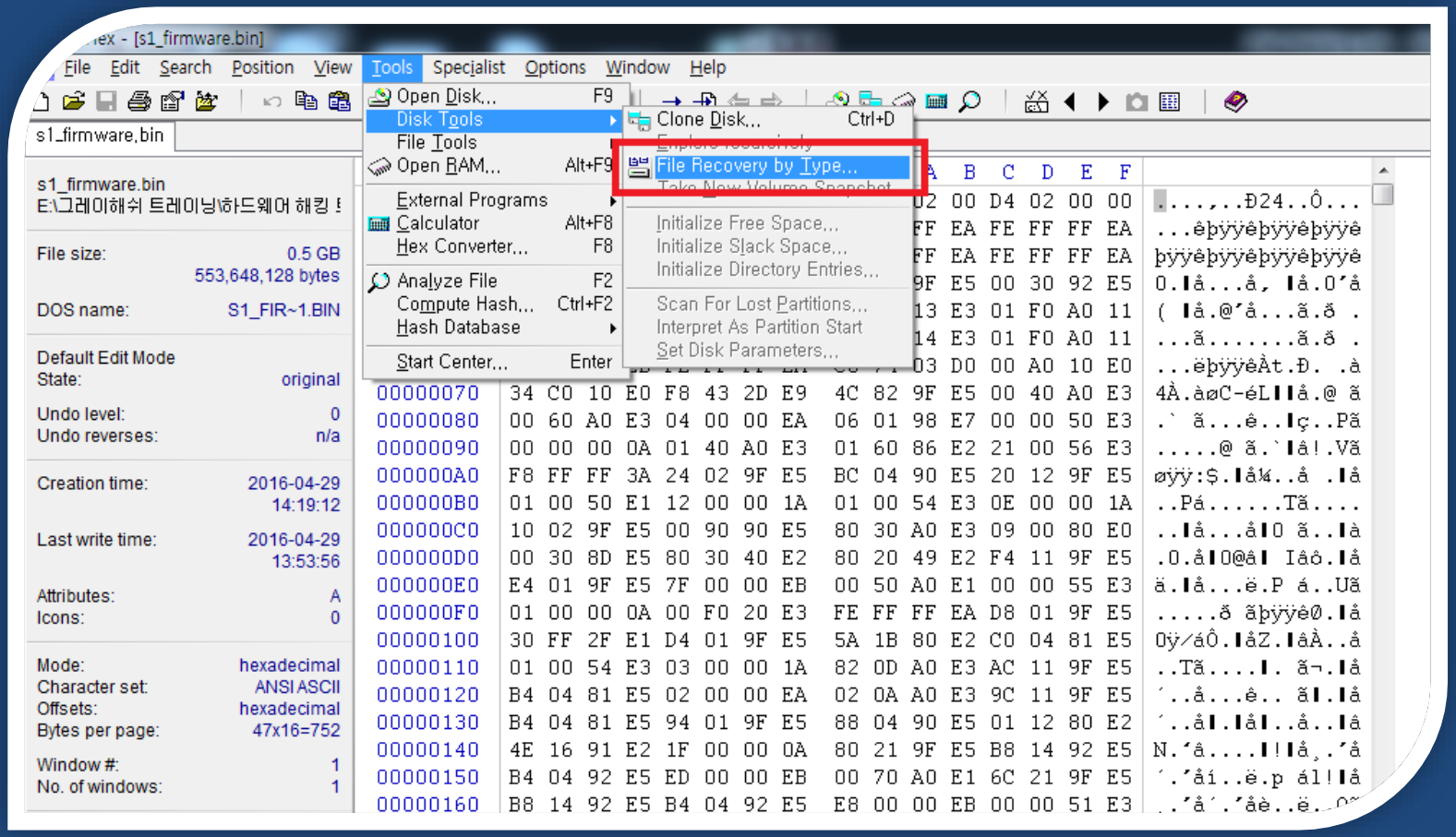
모든 문서는 크리에이티브 커먼즈 저작자표시-동일조건변경허락 3.0에 따라 사용할 수 있으며, 추가적인 조건이 적용될 수 있습니다. 자세한 내용은 이용 약관을 참고하십시오.
Wikipedia®는 미국 및 다른 국가에 등록되어 있는 Wikimedia Foundation, Inc. 소유의 등록 상표입니다.

개인정보 정책 위키백과 소개 면책 조항 개발자 쿠키 정책 모바일 보기

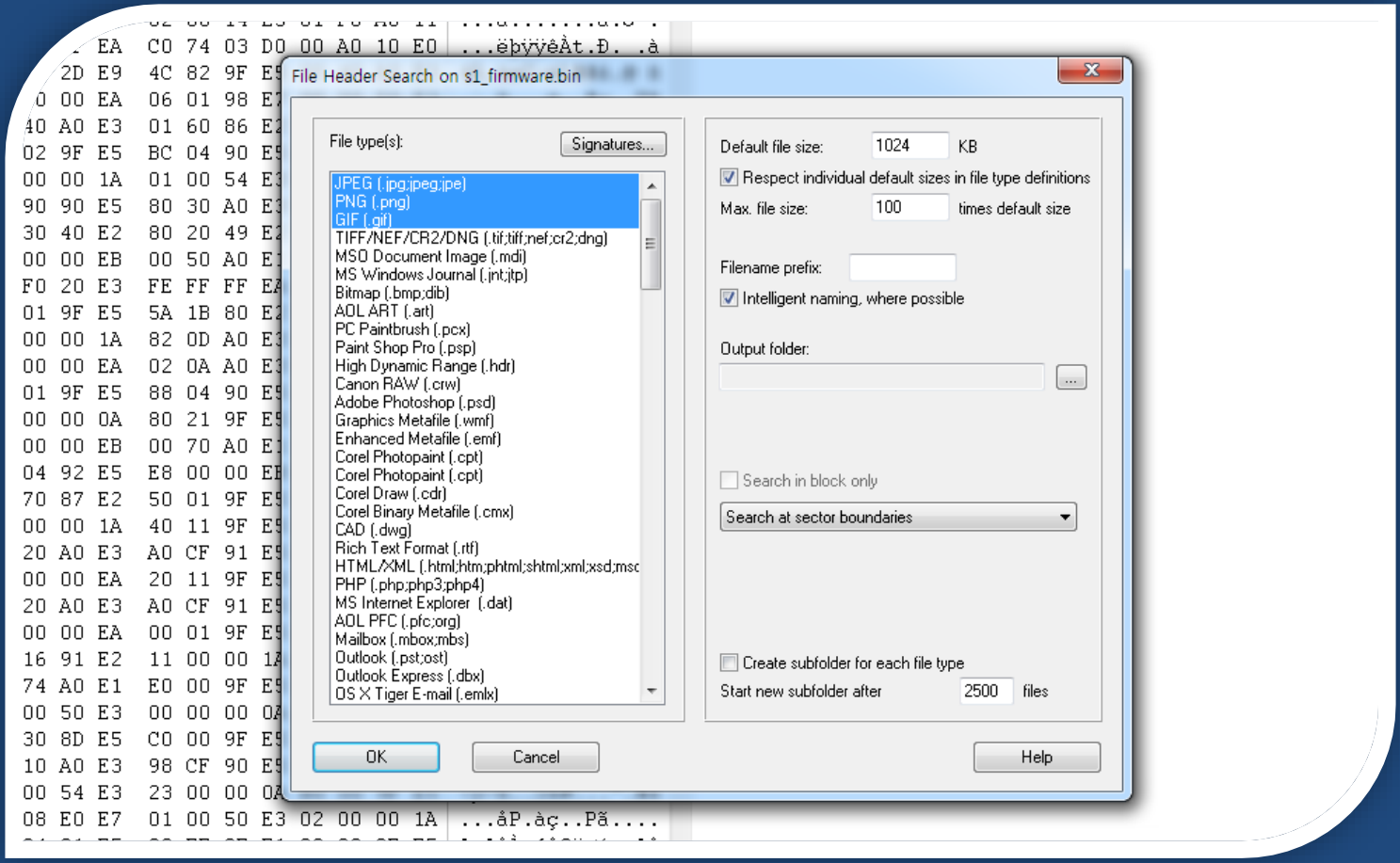
펌웨어 바이너리 내의 파일 추출



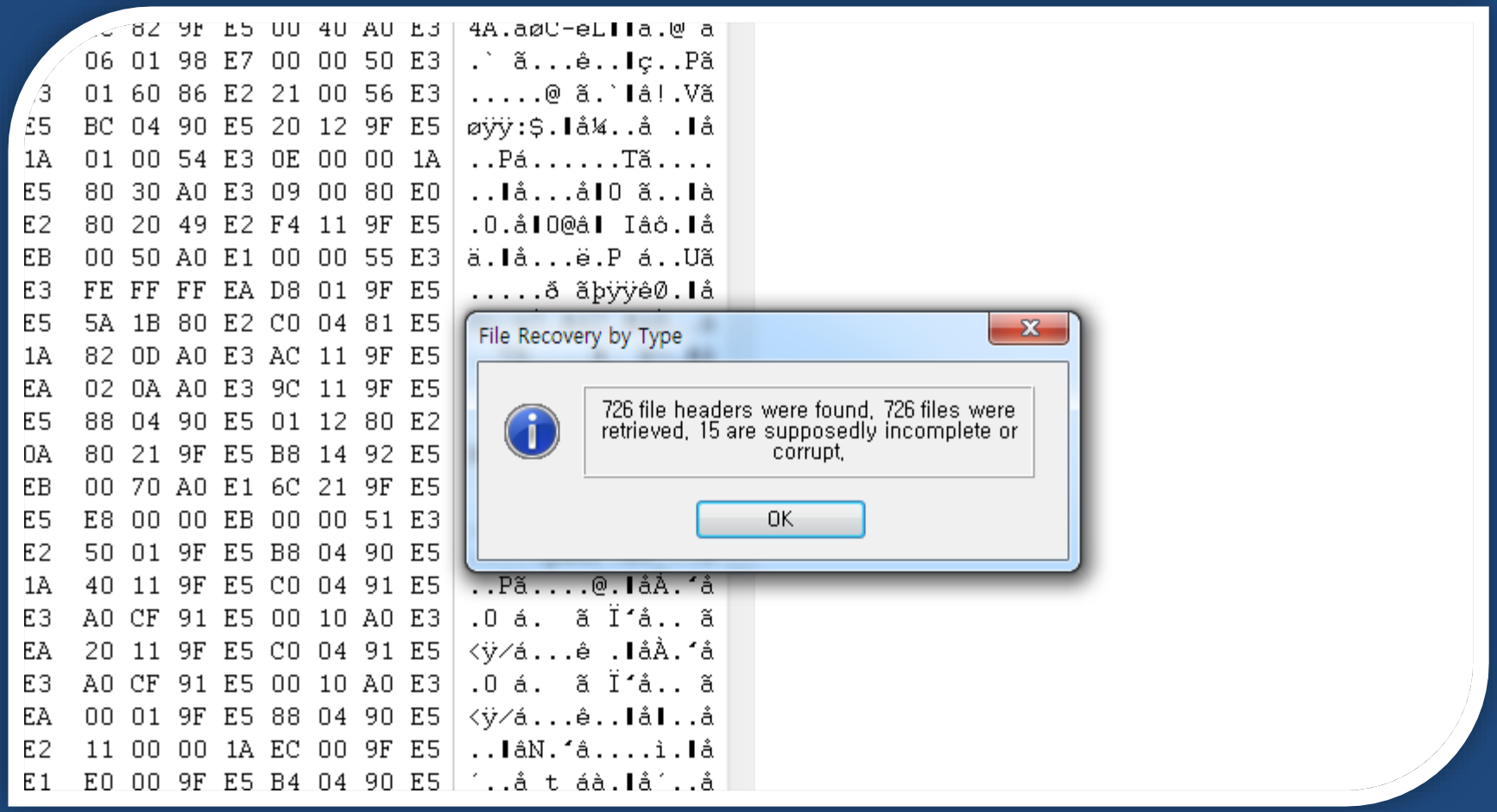
펌웨어 바이너리 내의 파일 추출



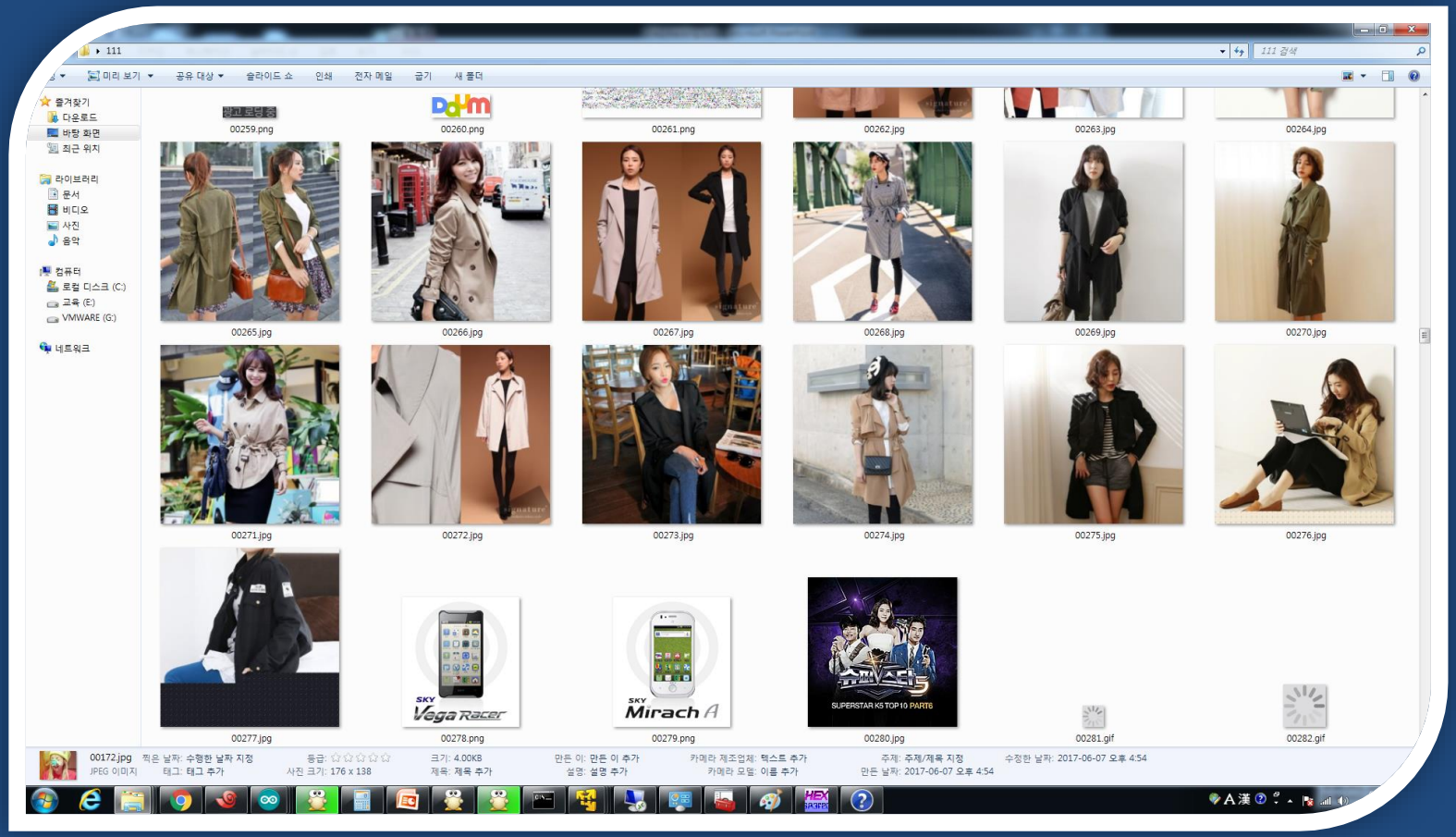
펌웨어 바이너리 내의 파일 추출



펌웨어 바이너리 내의 파일 추출

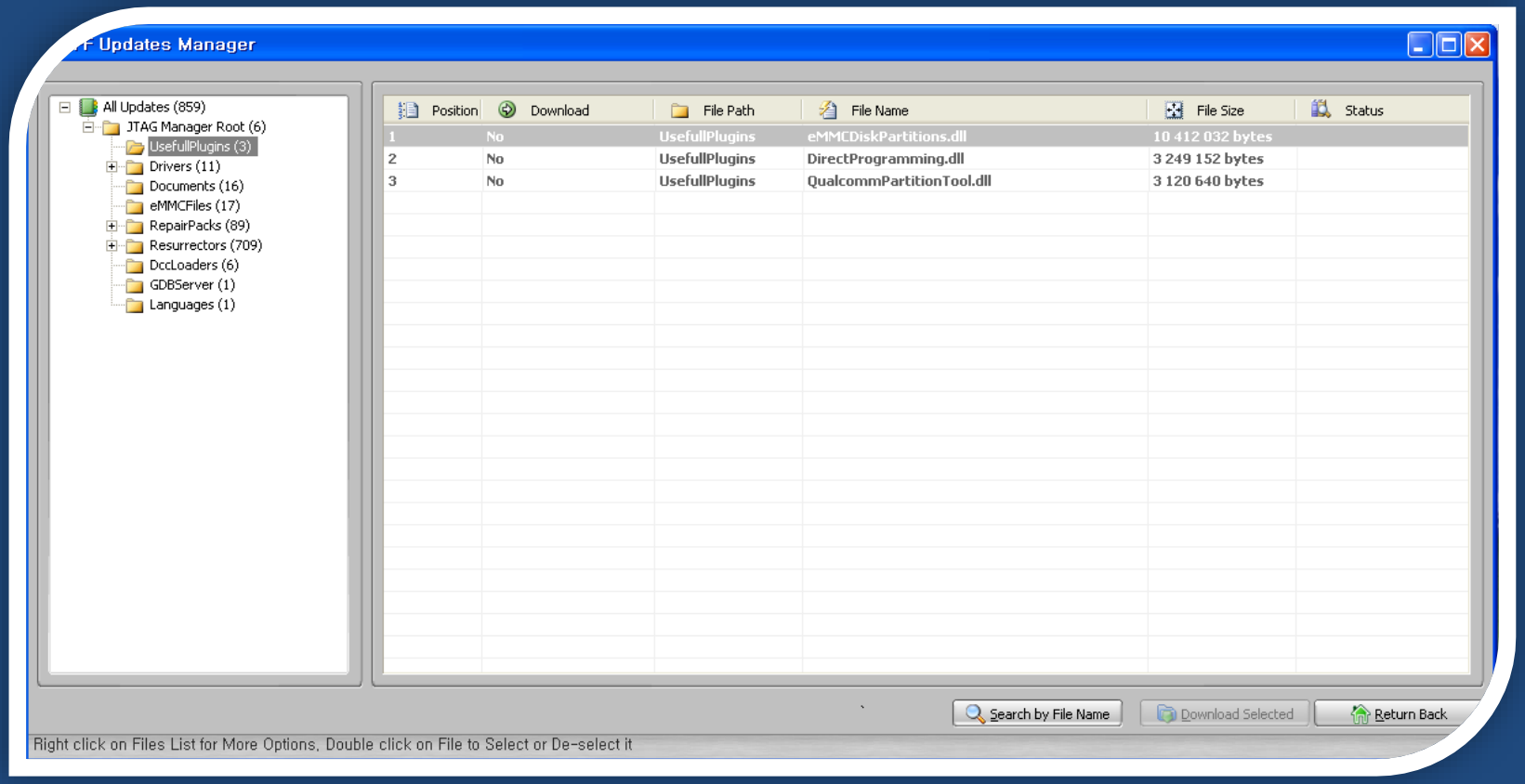


펌웨어 바이너리 내의 파일 추출

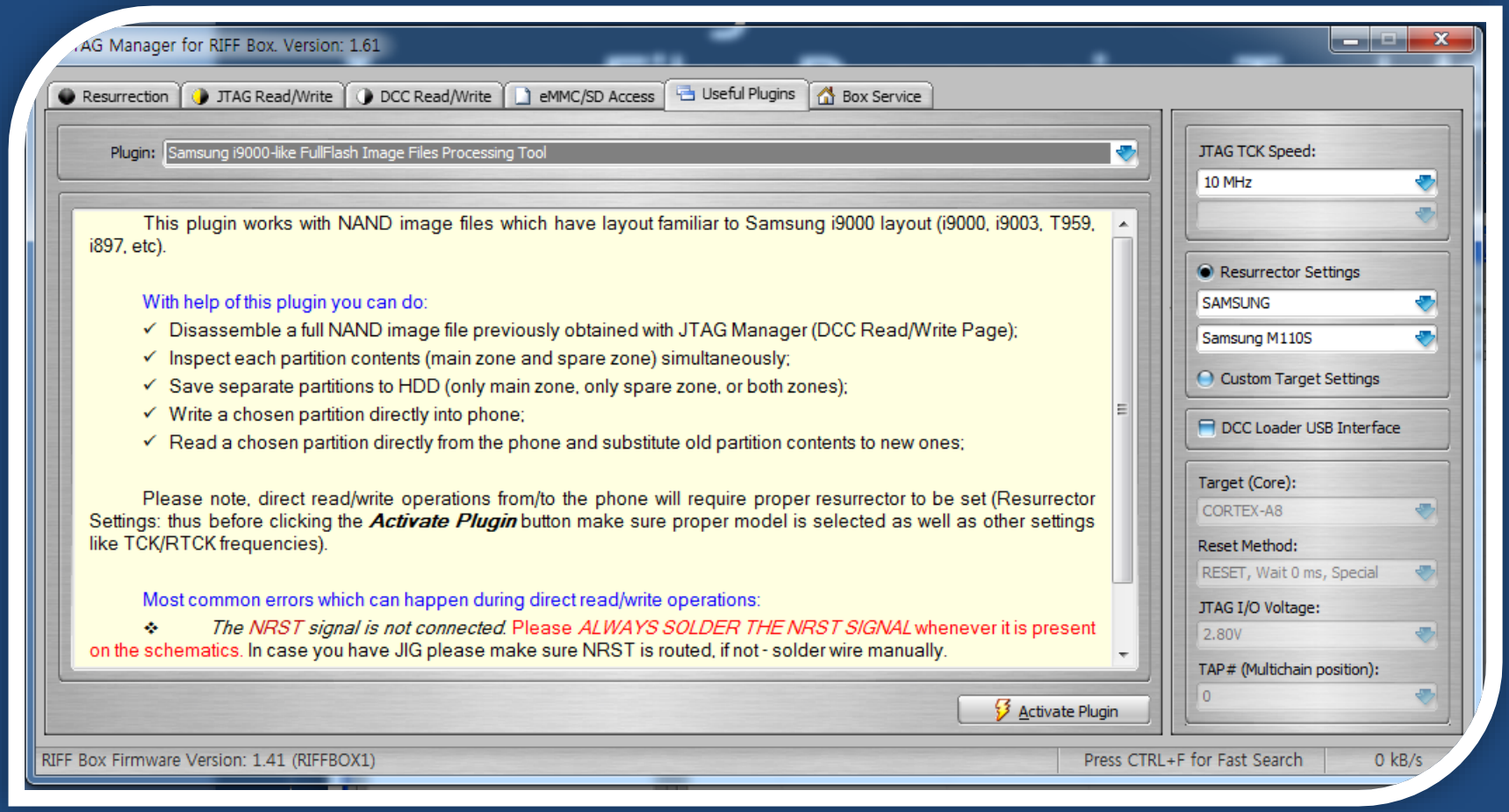


JTAG을 이용한 갤럭시 S1 Debugging 예제

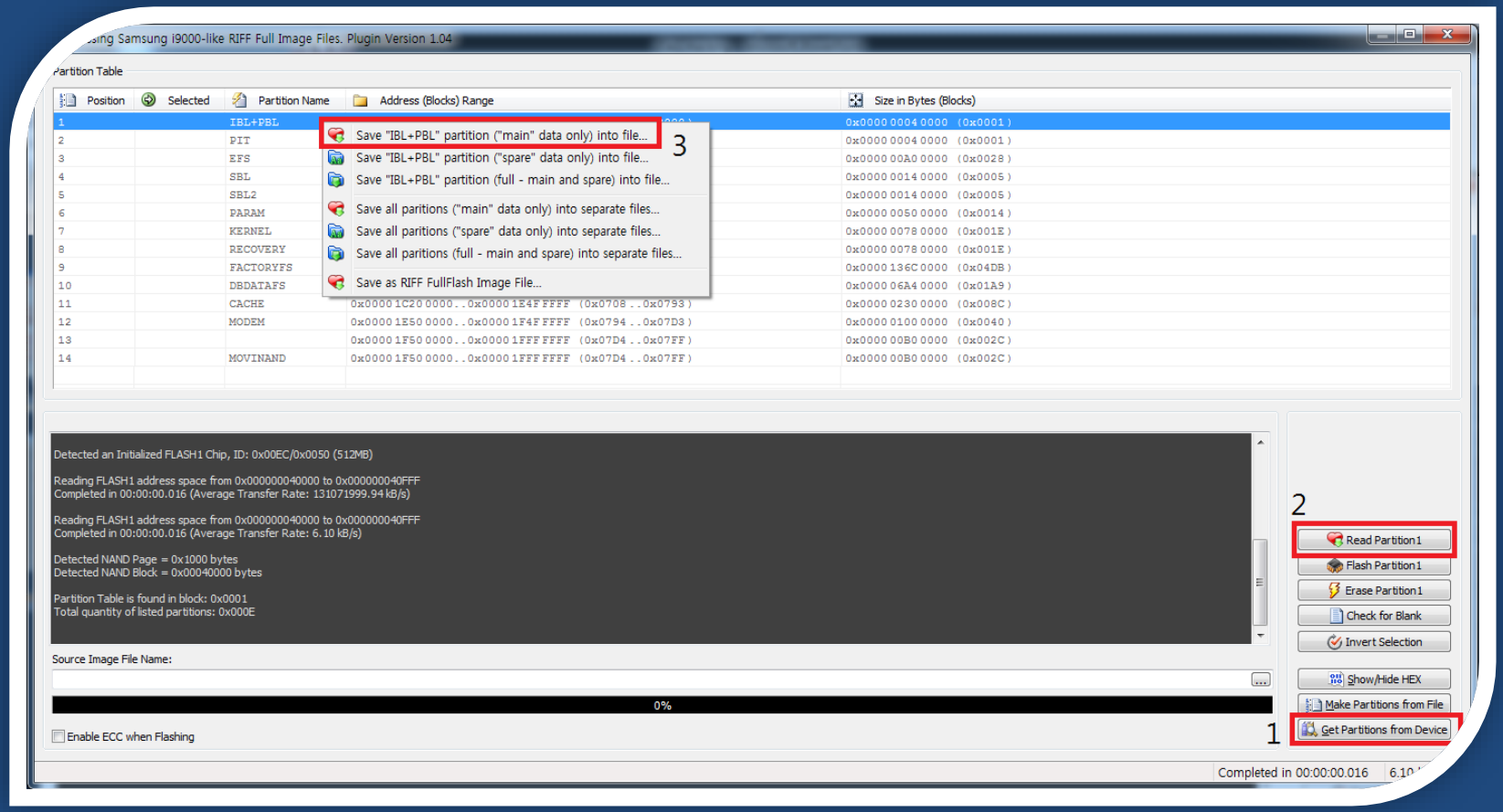
UserfulPlugins -> eMMCDiskPartitions 설치



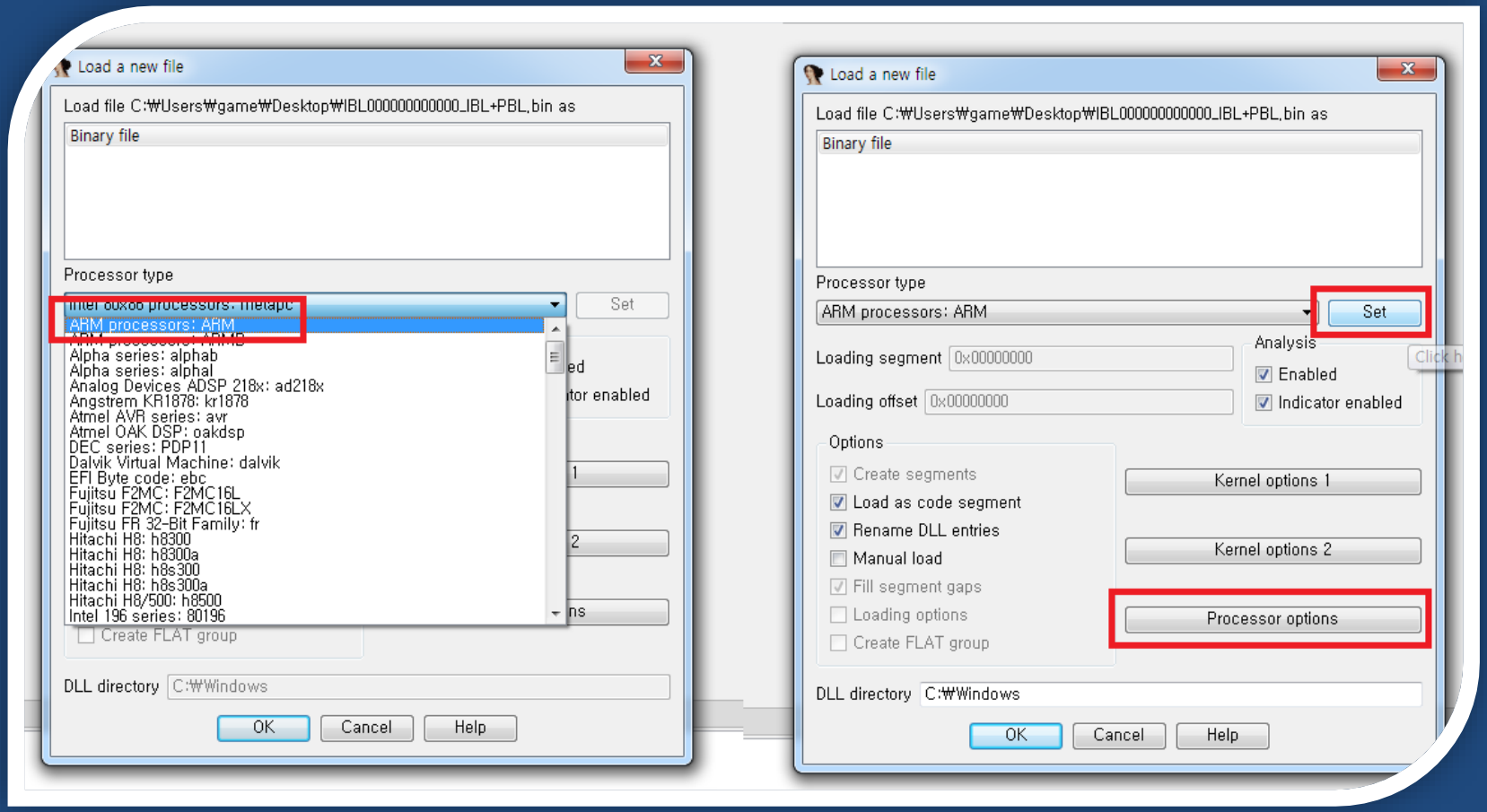
Useful Plugins 탭 -> Activate Plugin



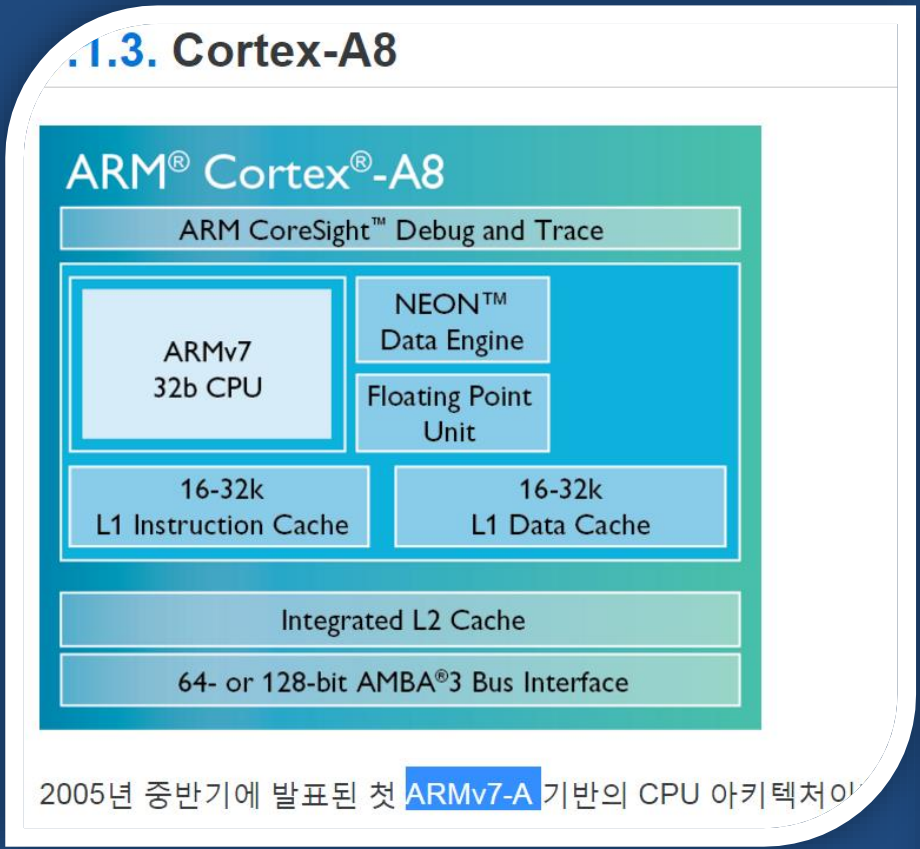
Boot Loader 덤프

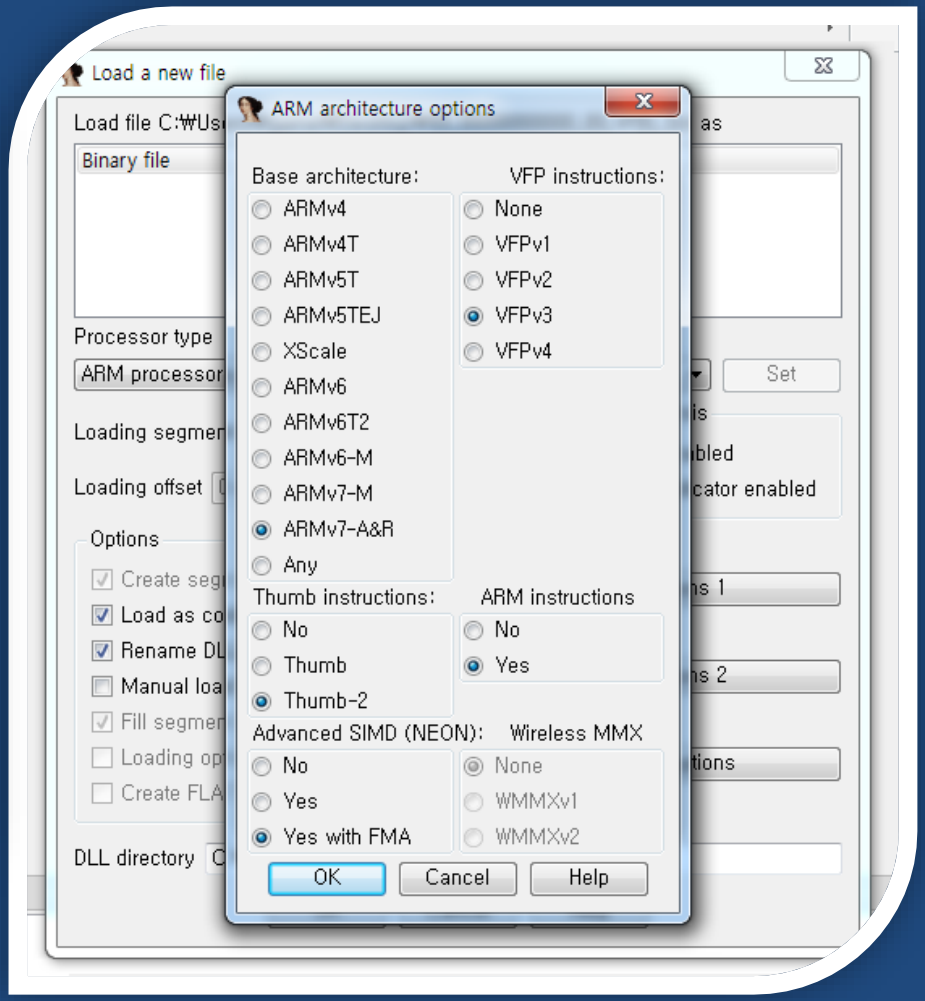


Binary 로딩

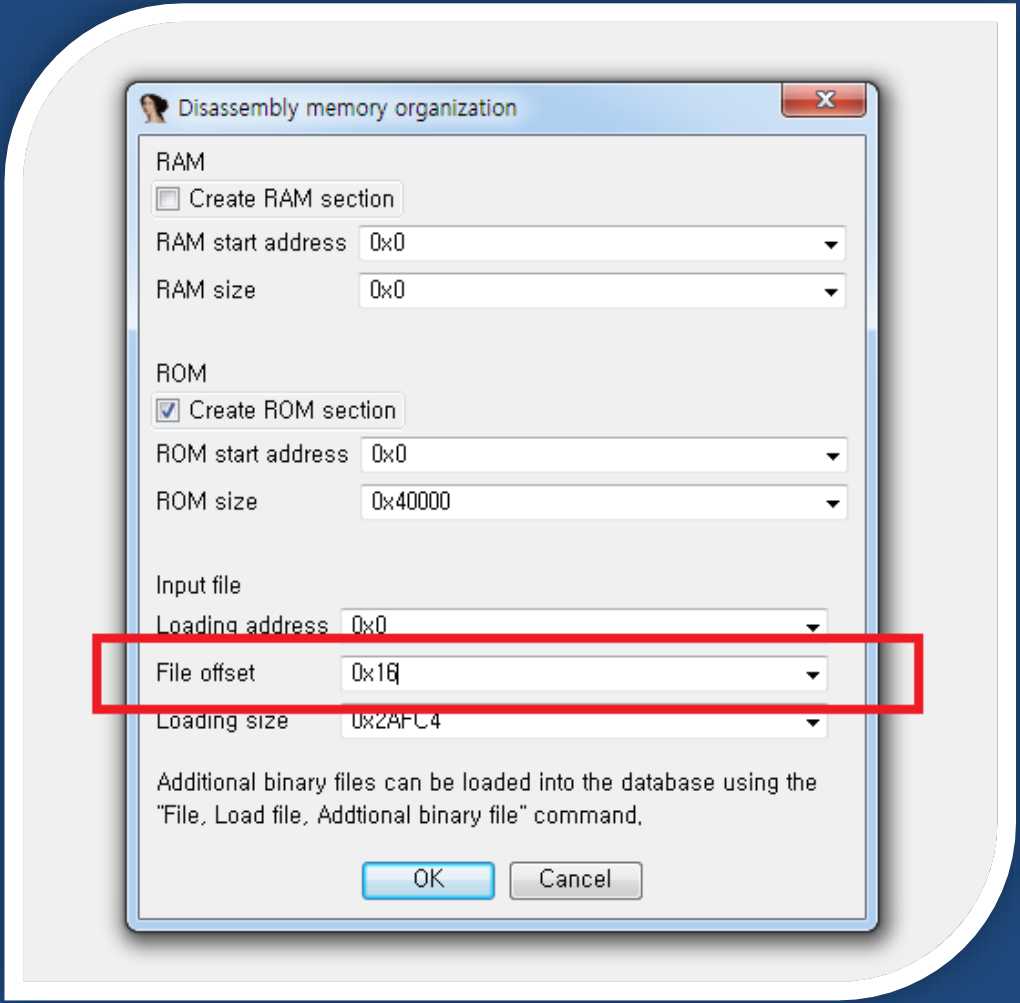

Load a new file
Load file C:₩Users₩game₩Desktop₩IBL000000000000_IBL+PBL.bin as
Binary file
Processor type
Intel 80x86 processors: metapc
ARM processors: ARM
Alpha series: alphab
Alpha series: alphal
Analog Devices ADSP 218x: ad218x
Angstrem KR1878: kr1878
Atmel AVR series: avr
Atmel OAK DSP: oakdsp
DEC series: PDP11
Dalvik Virtual Machine: dalvik
EFI Byte code: ebc
Fujitsu F2MC: F2MC16L
Fujitsu F2MC: F2MC16LX
Fujitsu FR 32-Bit Family: fr
Hitachi H8: h8300
Hitachi H8: h8300a
Hitachi H8: h8s300
Hitachi H8: h8s300a
Hitachi H8/500: h8500
Intel 196 series: 80196
Set
Create FLAT group
DLL directory
C:₩Windows
OK
Cancel
Help
Load a new file
Load file C:₩Users₩game₩Desktop₩IBL000000000000_IBL+PBL.bin as
Binary file
Processor type
ARM processors: ARM
Set
Loading segment
0x00000000
Loading offset
0x00000000
Analysis
Enabled
Indicator enabled
Options
Create segments
Load as code segment
Rename DLL entries
Manual load
Fill segment gaps
Loading options
Create FLAT group
Kernel options 1
Kernel options 2
Processor options
DLL directory
C:₩Windows
OK
Cancel
Help

Binary 로딩

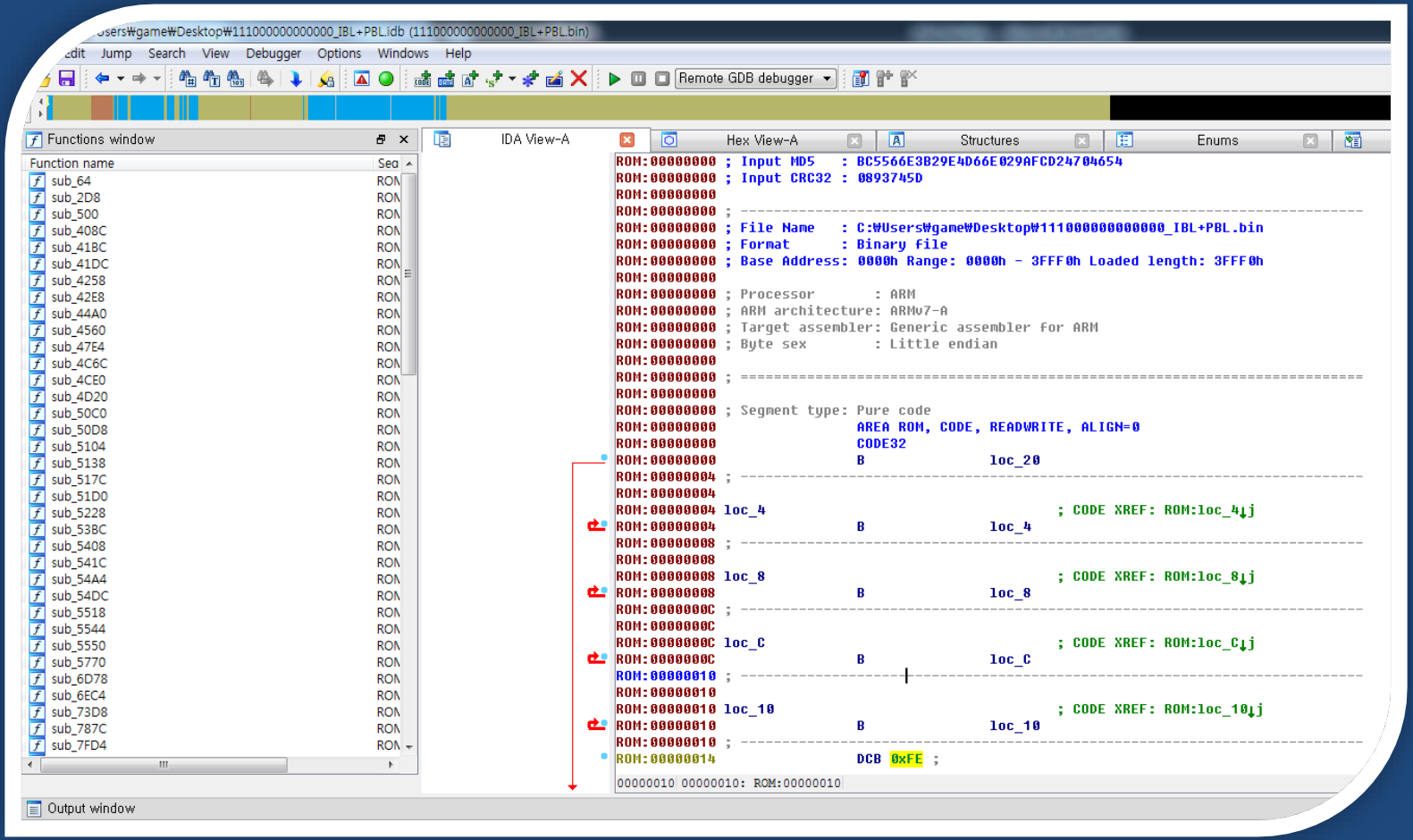

.1.3. Cortex-A8
ARM® Cortex®-A8
ARM CoreSight™ Debug and Trace
ARMv7
32b CPU
NEON™
Data Engine
Floating Point
Unit
16-32k
L1 Instruction Cache
16-32k
L1 Data Cache
Integrated L2 Cache
64- or 128-bit AMBA®3 Bus Interface
2005년 중반기에 발표된 첫 ARMv7-A 기반의 CPU 아키텍처이


Load a new file
Load file C:₩Us
Binary file
Processor type
ARM processor
Loading segmen
Loading offset
Options
Create seg
Load as co
Rename DL
Manual loa
Fill segmen
Loading op
Create FLA
DLL directory
Set
ARM architecture options
Base architecture:
ARMv4
ARMv4T
ARMv5T
ARMv5TEJ
XScale
ARMv6
ARMv6T2
ARMv6-M
ARMv7-M
ARMv7-A&R
Any
VFP instructions:
None
VFPv1
VFPv2
VFPv3
VFPv4
Thumb instructions:
No
Thumb
Thumb-2
ARM instructions
No
Yes
Advanced SIMD (NEON):
No
Yes
Yes with FMA
Wireless MMX
None
WMMXv1
WMMXv2
OK
Cancel
Help

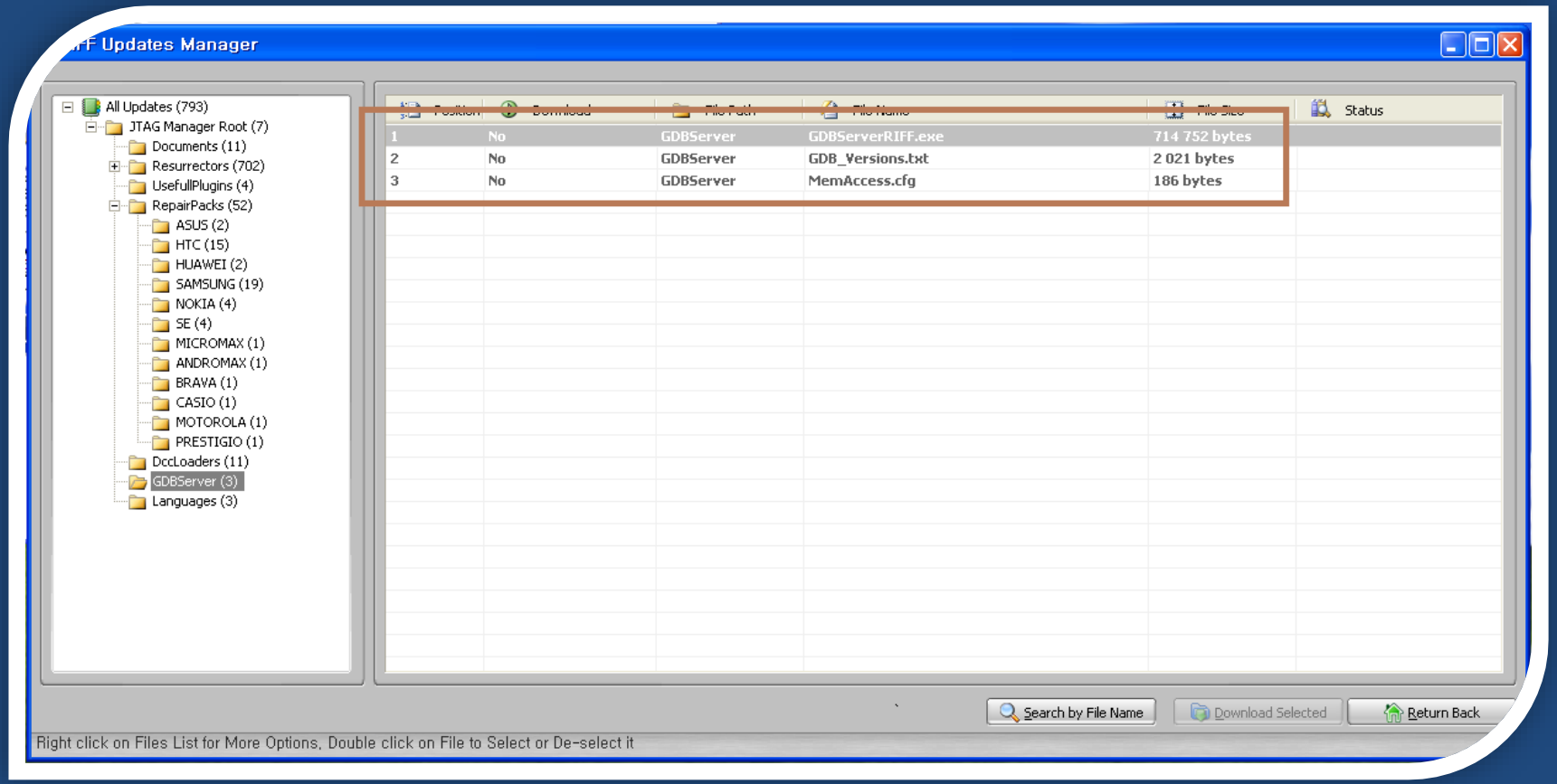
Header 영역 SKIP



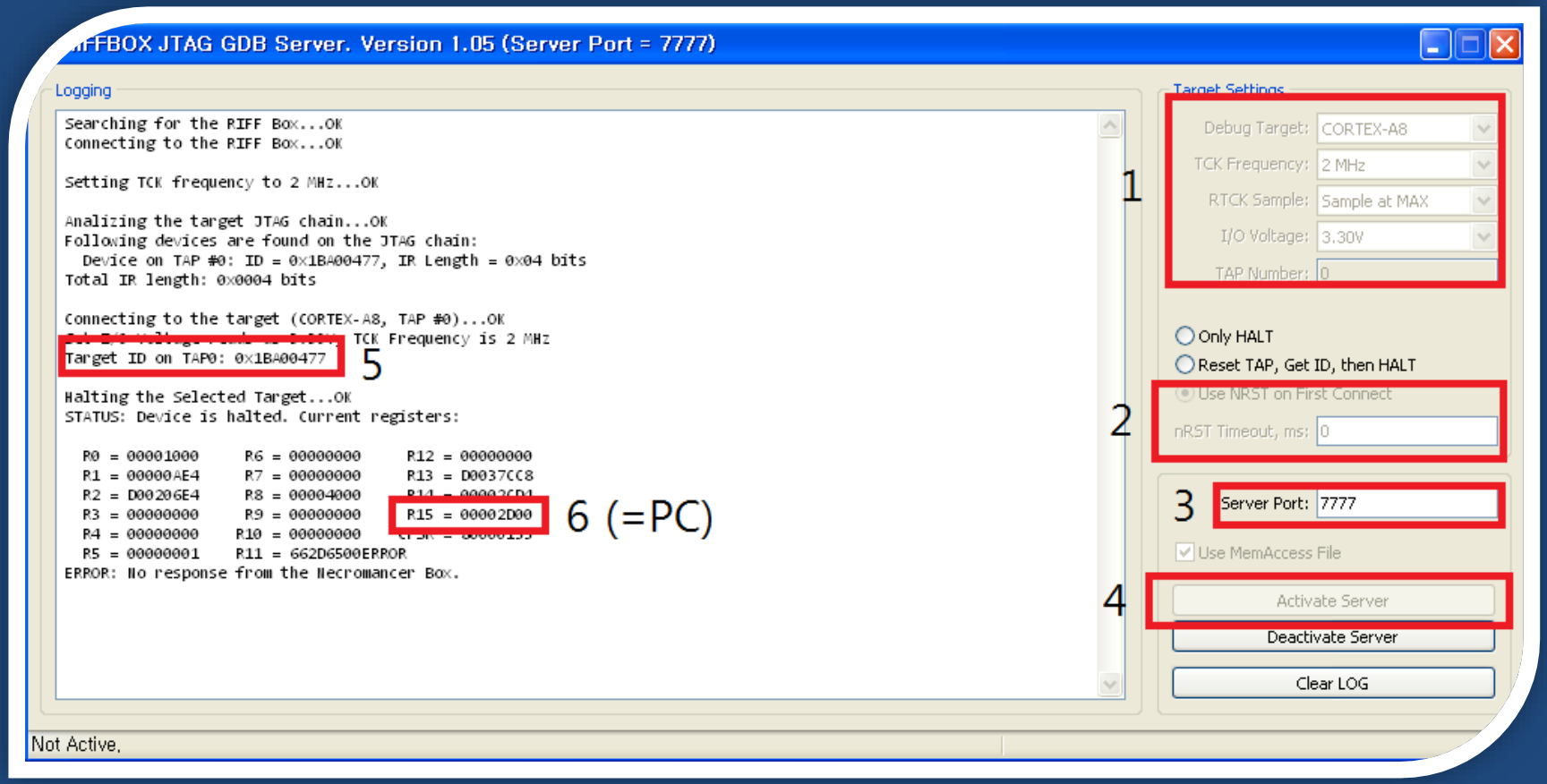
Binary 로딩


Jump Search View Debugger Options Windows Help
Remote GDB debugger
Functions window
Function name
sub_64
sub_2D8
sub_500
sub_408C
sub_41BC
sub_41DC
sub_4258
sub_42E8
sub_44A0
sub_4560
sub_47E4
sub_4C6C
sub_4CE0
sub_4D20
sub_50C0
sub_50D8
sub_5104
sub_5138
sub_517C
sub_51D0
sub_5228
sub_53BC
sub_5408
sub_541C
sub_54A4
sub_54DC
sub_5518
sub_5544
sub_5550
sub_5770
sub_6D78
sub_6EC4
sub_73D8
sub_787C
sub_7FD4
IDA View-A
Hex View-A
Structures
Enums
ROM:00000000 ; Input MD5 : BC5566E3B29E4D66E029AFCD24704654
ROM:00000000 ; Input CRC32 : 0893745D
ROM:00000000 ; File Name : C:\Users\game\Desktop\111000000000000_IBL+PBL.bin
ROM:00000000 ; Format : Binary file
ROM:00000000 ; Base Address: 0000h Range: 0000h - 3FFF0h Loaded length: 3FFF0h
ROM:00000000 ; Processor : ARM
ROM:00000000 ; ARM architecture: ARMv7-A
ROM:00000000 ; Target assembler: Generic assembler for ARM
ROM:00000000 ; Byte sex : Little endian
ROM:00000000 ; Segment type: Pure code
ROM:00000000 AREA ROM, CODE, READWRITE, ALIGN=0
ROM:00000000 CODE32
ROM:00000000 B loc_20
ROM:00000004 loc_4 ; CODE XREF: ROM:loc_4↓j
ROM:00000004 B loc_4
ROM:00000008 loc_8 ; CODE XREF: ROM:loc_8↓j
ROM:00000008 B loc_8
ROM:0000000C loc_C ; CODE XREF: ROM:loc_C↓j
ROM:0000000C B loc_C
ROM:00000010 loc_10 ; CODE XREF: ROM:loc_10↓j
ROM:00000010 B loc_10
ROM:00000014 DCB 0xFE ;
00000010 00000010: ROM:00000010
Output window

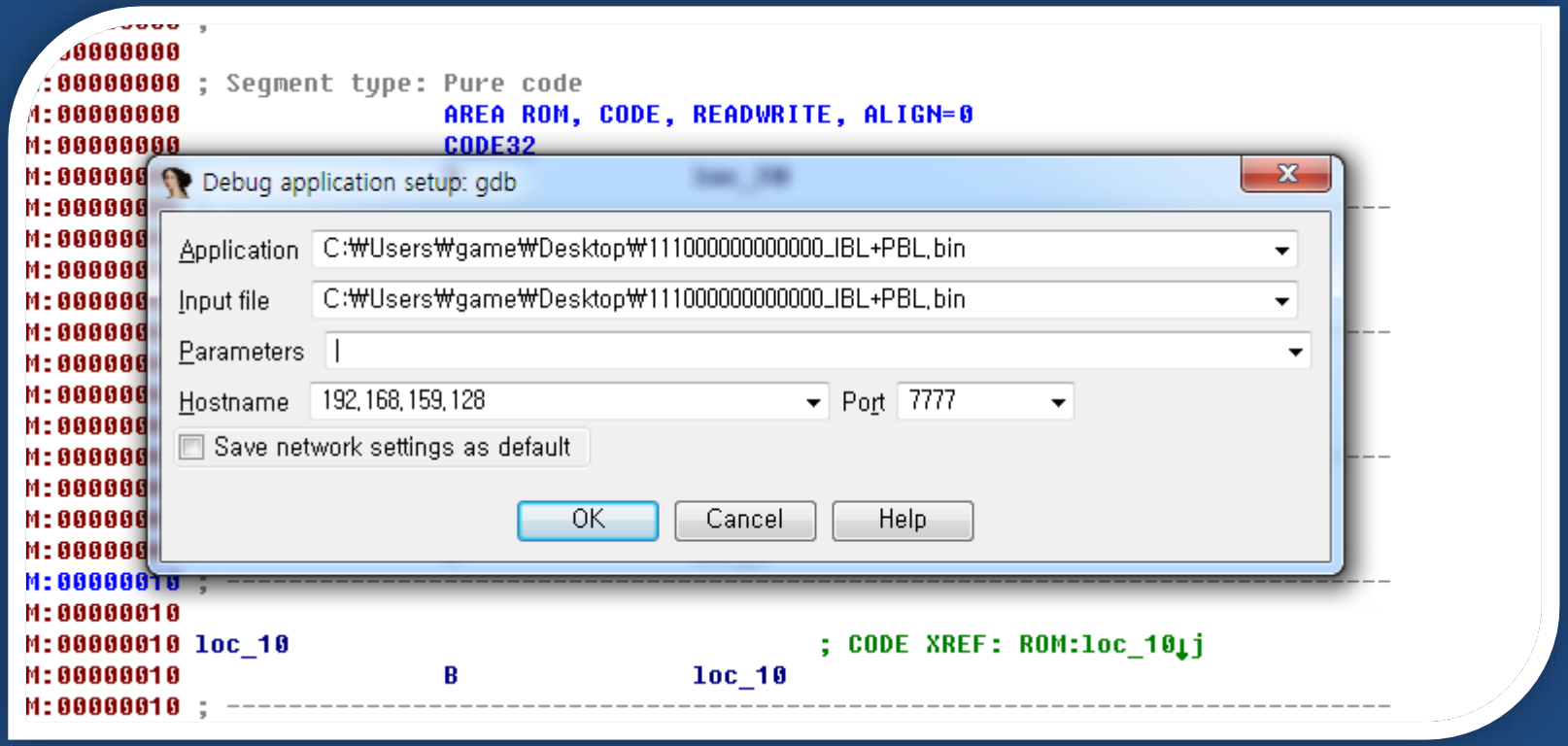
GDBServer 설치



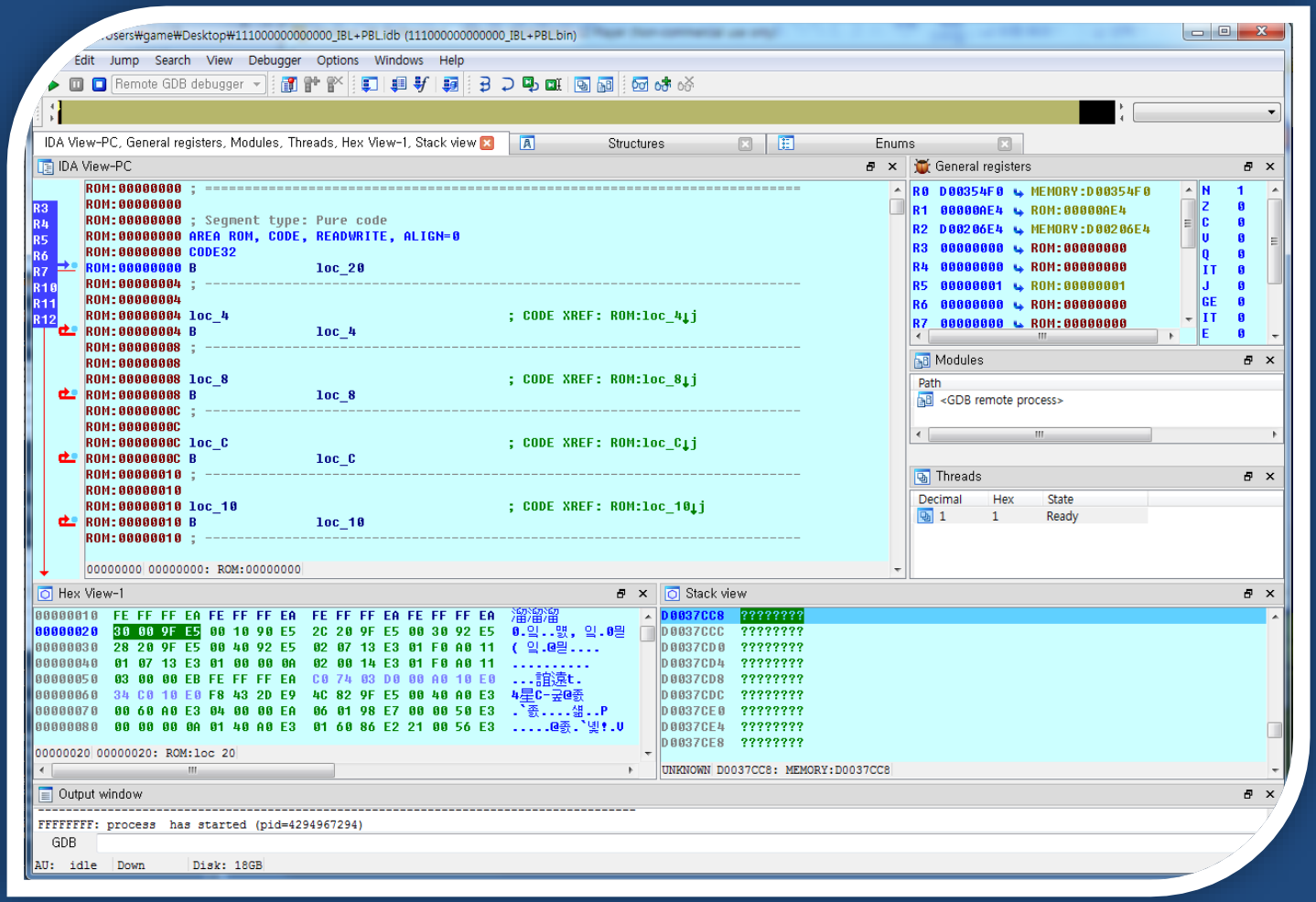
GDBServerRIFF.exe 실행



Remote debugger 세팅

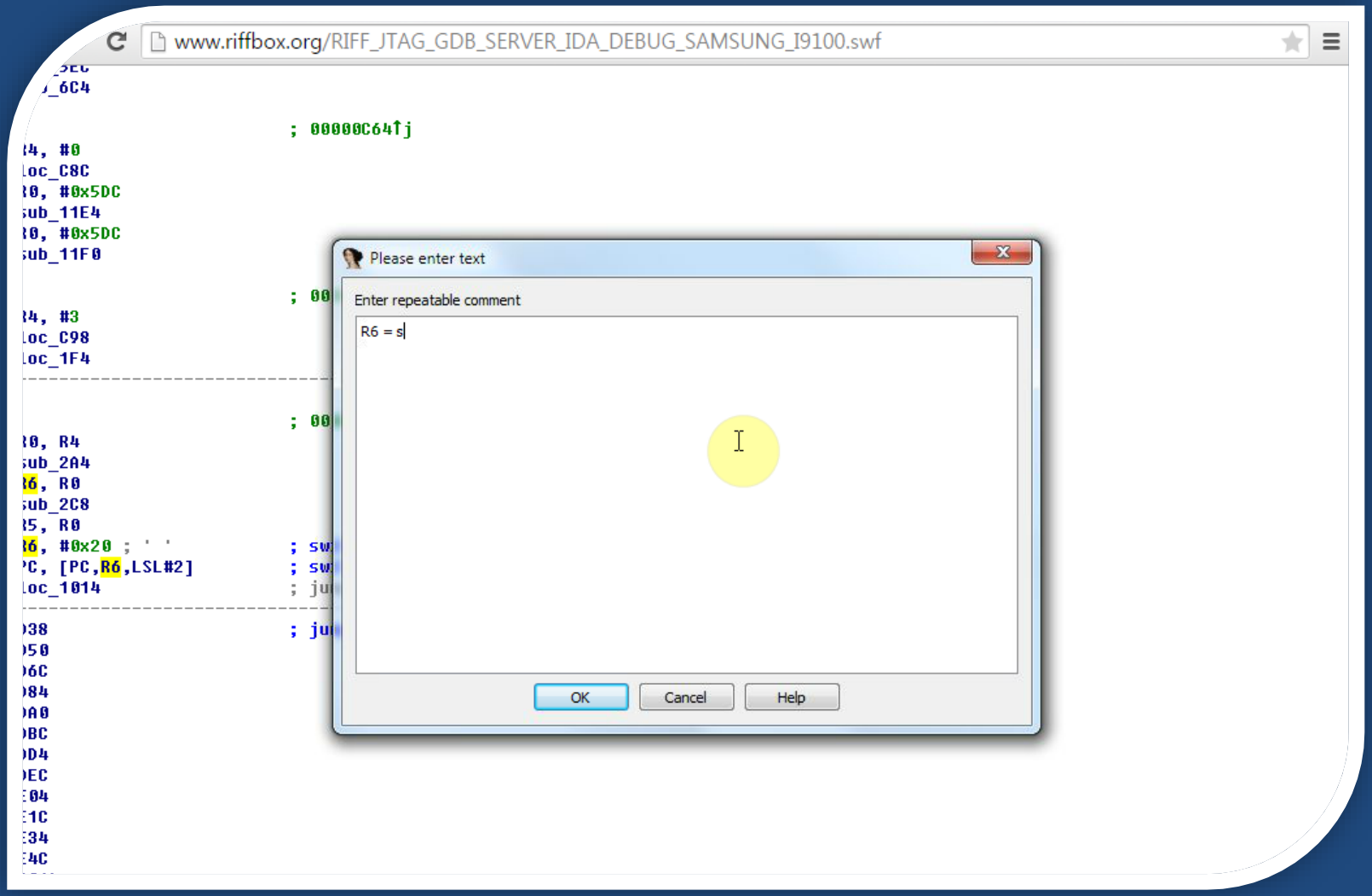


Debugger -> Start process



JTAG 동적디버깅 참고 영상

http://www.riffbox.org/RIFF_JTAG_GDB_SERVER_IDA_DEBUG_SAMSUNG_I9100.swf

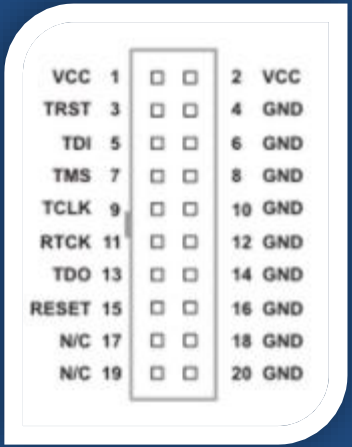


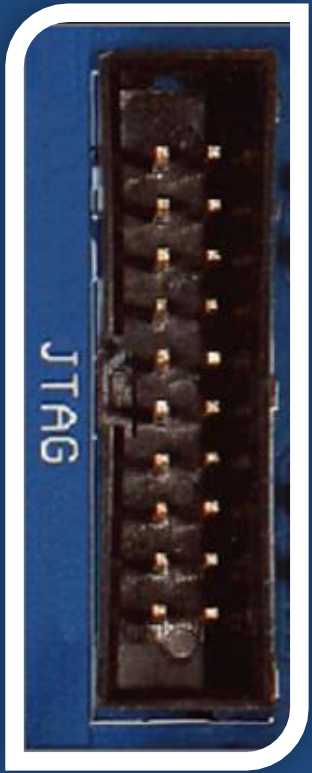
JTAG Pin 찾기

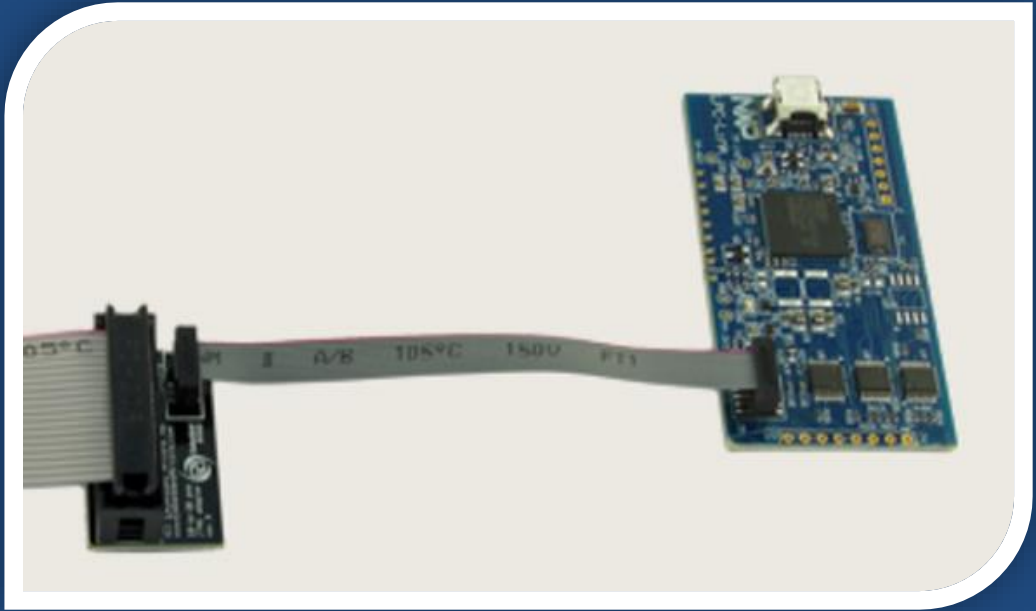
JTAG 핀을 찾는 방법들

- PCB에 JTAG이라고 인쇄된 식자 확인
- TDO, TDI, TMS, TCK 식자 확인
- CPU JTAG 핀들과의 연결 추적
- JTAG Pin Scanning

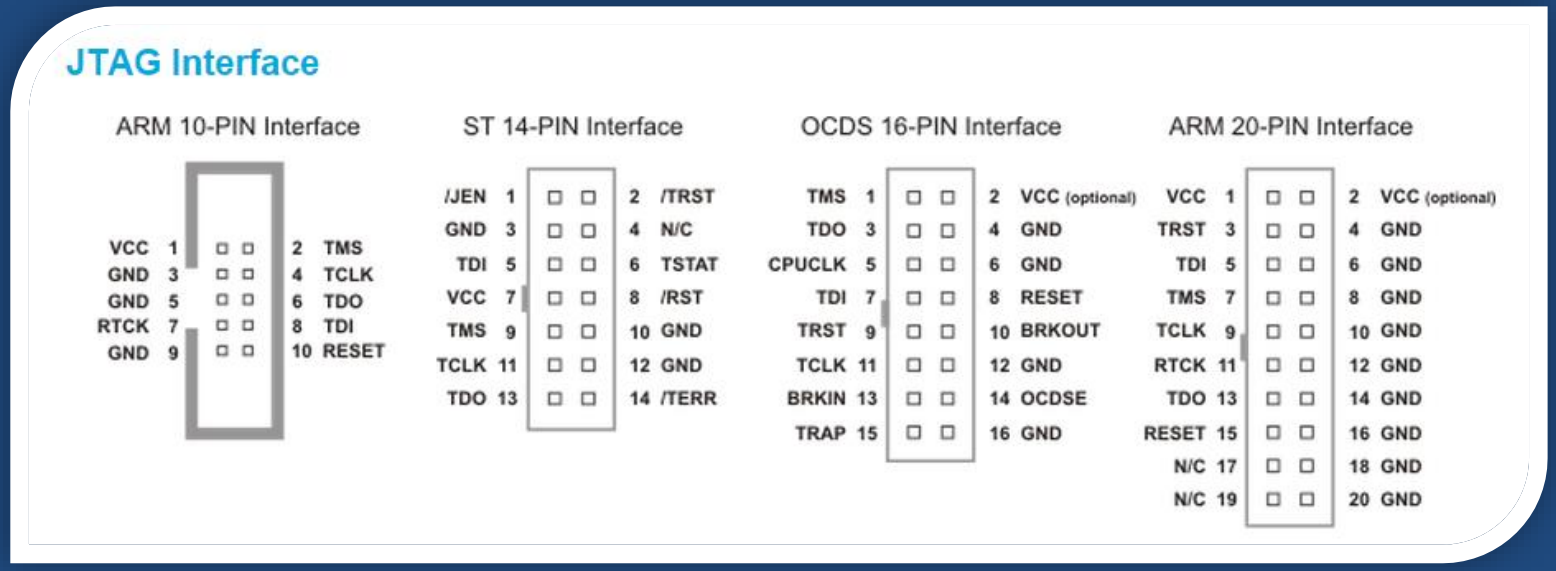
JTAG 찾기 예제


VCC 1
TRST 3
TDI 5
TMS 7
TCLK 9
RTCK 11
TDO 13
RESET 15
N/C 17
N/C 19
2 VCC
4 GND
6 GND
8 GND
10 GND
12 GND
14 GND
16 GND
18 GND
20 GND

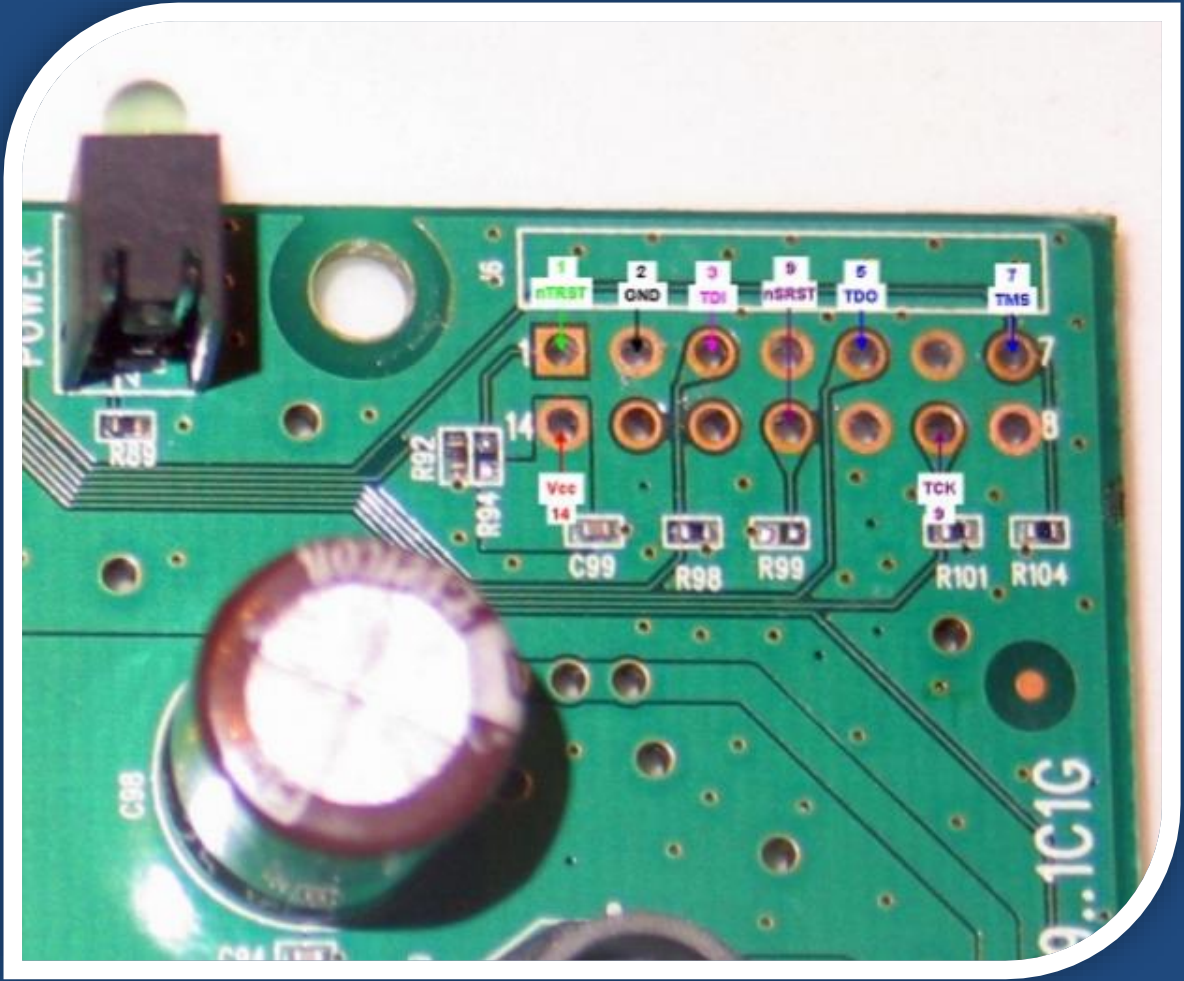

JTAG



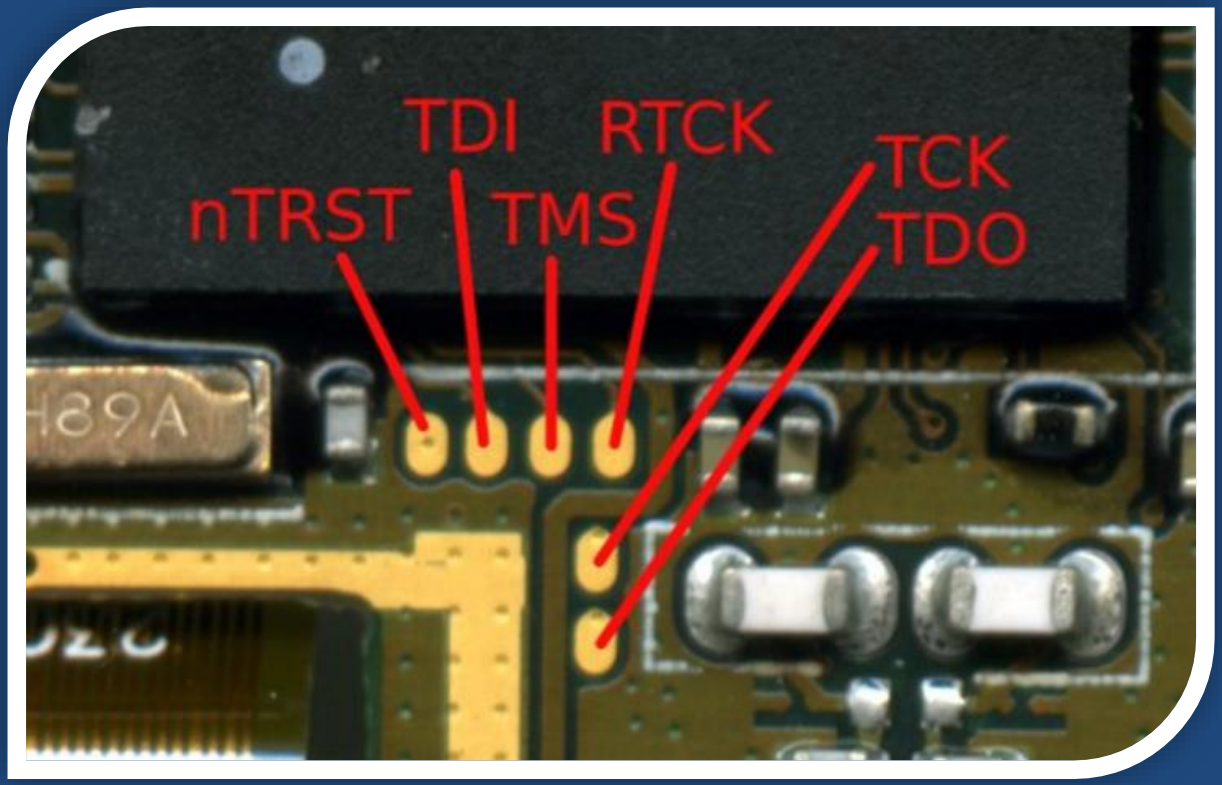
다양한 JTAG 인터페이스들



JTAG 찾기 예제

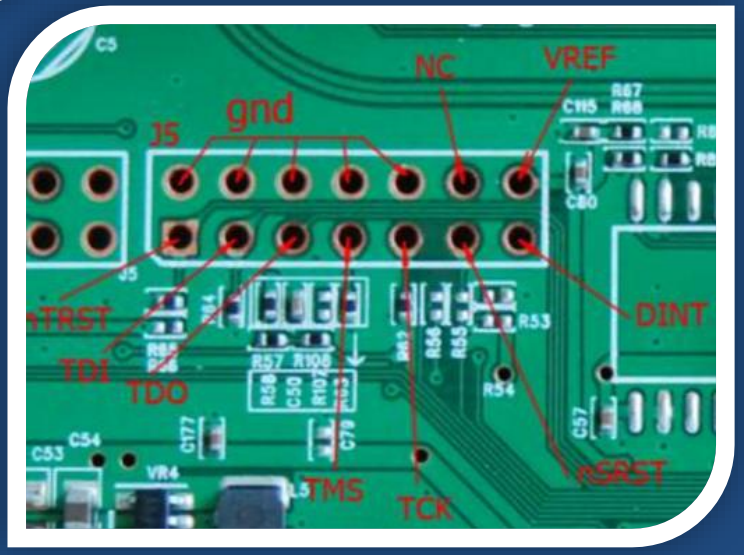

1
nTRST
2
GND
3
TDI
9
nSRST
5
TDO
7
TMS
1
14
7
8
Vcc
14
TCK
9
R89
R92
R94
C99
R98
R99
R101
R104
C98
J5

JTAG 찾기 예제



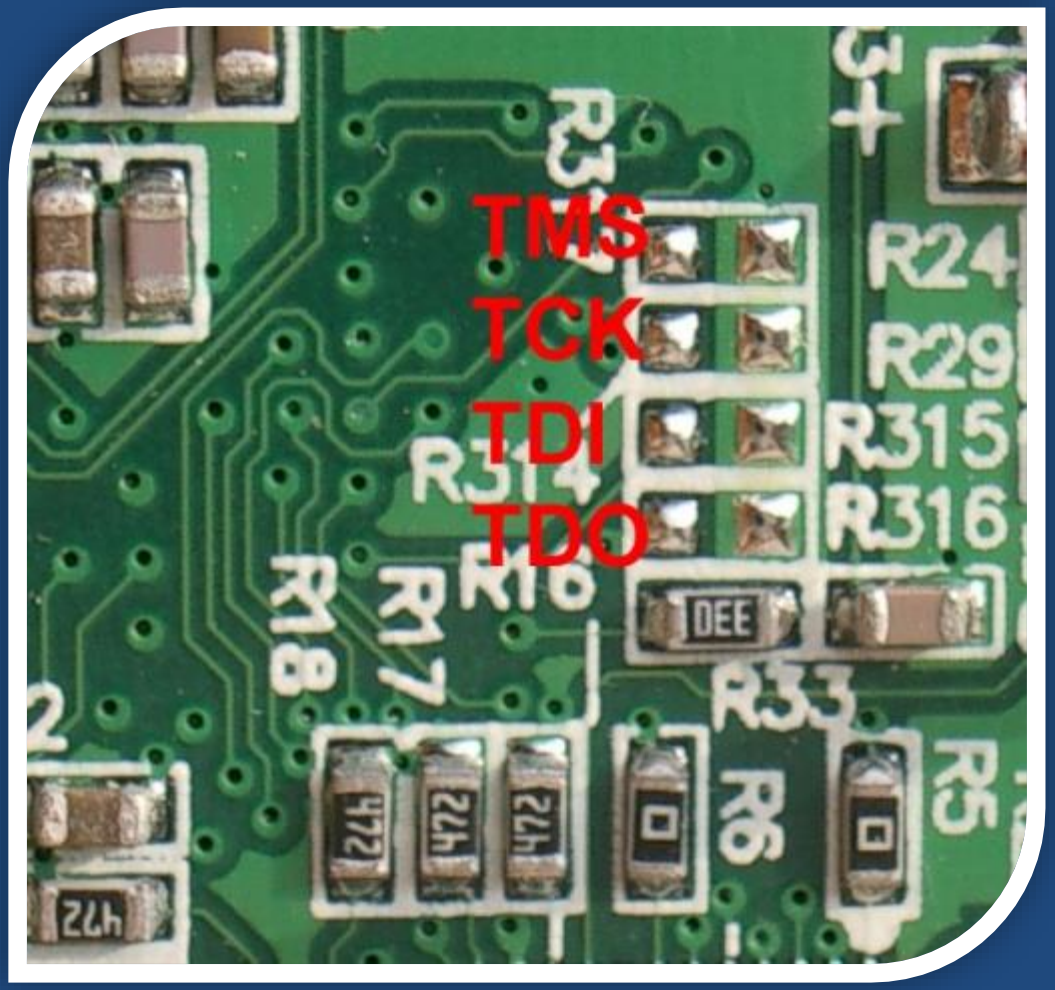
JTAG 찾기 예제

J-TAG
0V
RN201
TCK
TDI
TDO
TMS
TRST
Max 232
RX
TX
0V
+V
J301
J303
C253
R311
C306
R314
D301
R313
R312

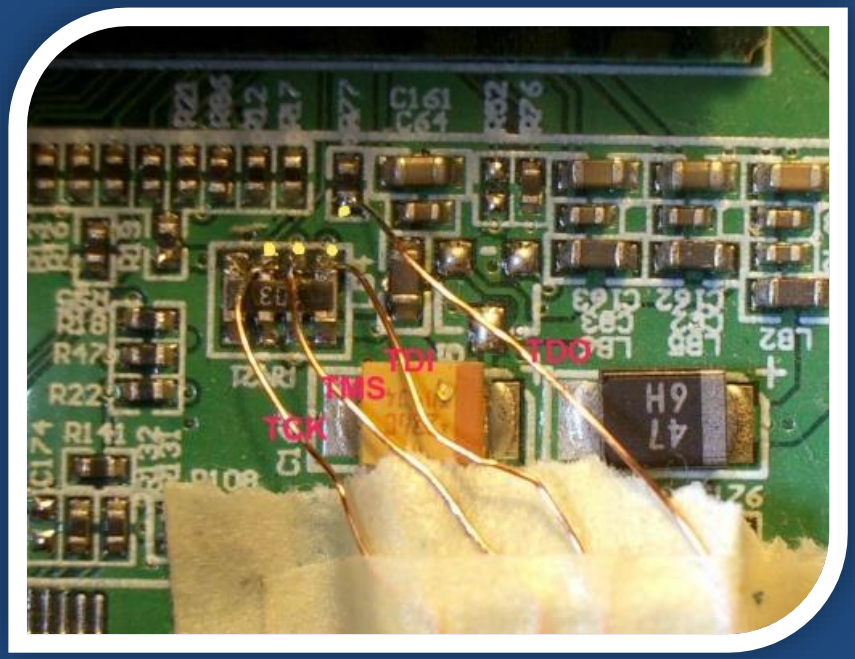

NC
VREF
gnd
J5
nTRST
TDI
TDO
TMS
TCK
DINT
nSRST

JTAG 찾기 예제

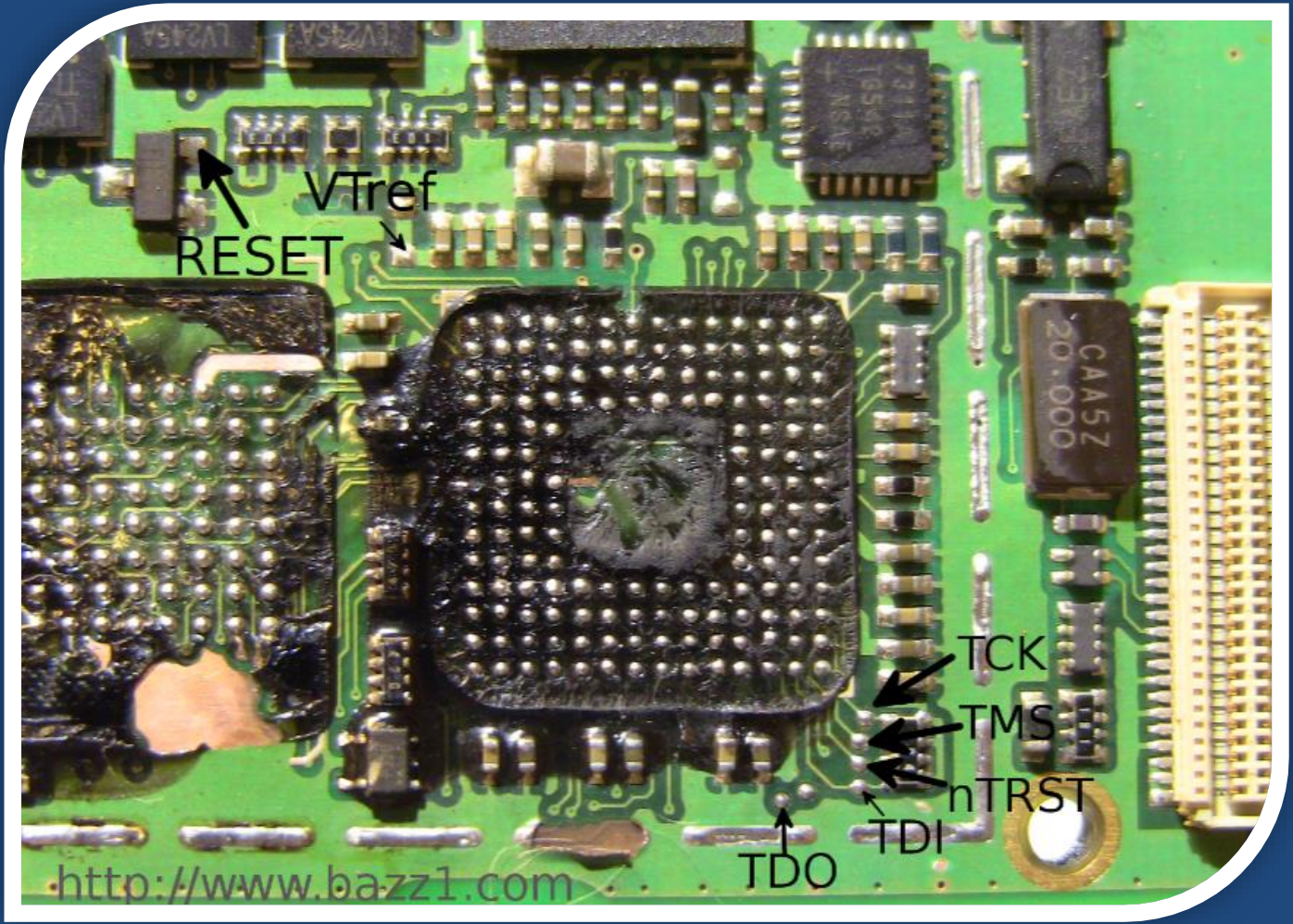
Siemens
SE505 v2
JTAG (J7)
TDO
TDI
TCK
TMS
GND
LED Front


TMS
TCK
TDI
TDO
R24
R29
R315
R316
R314
R16
R17
R18
R33
DEE

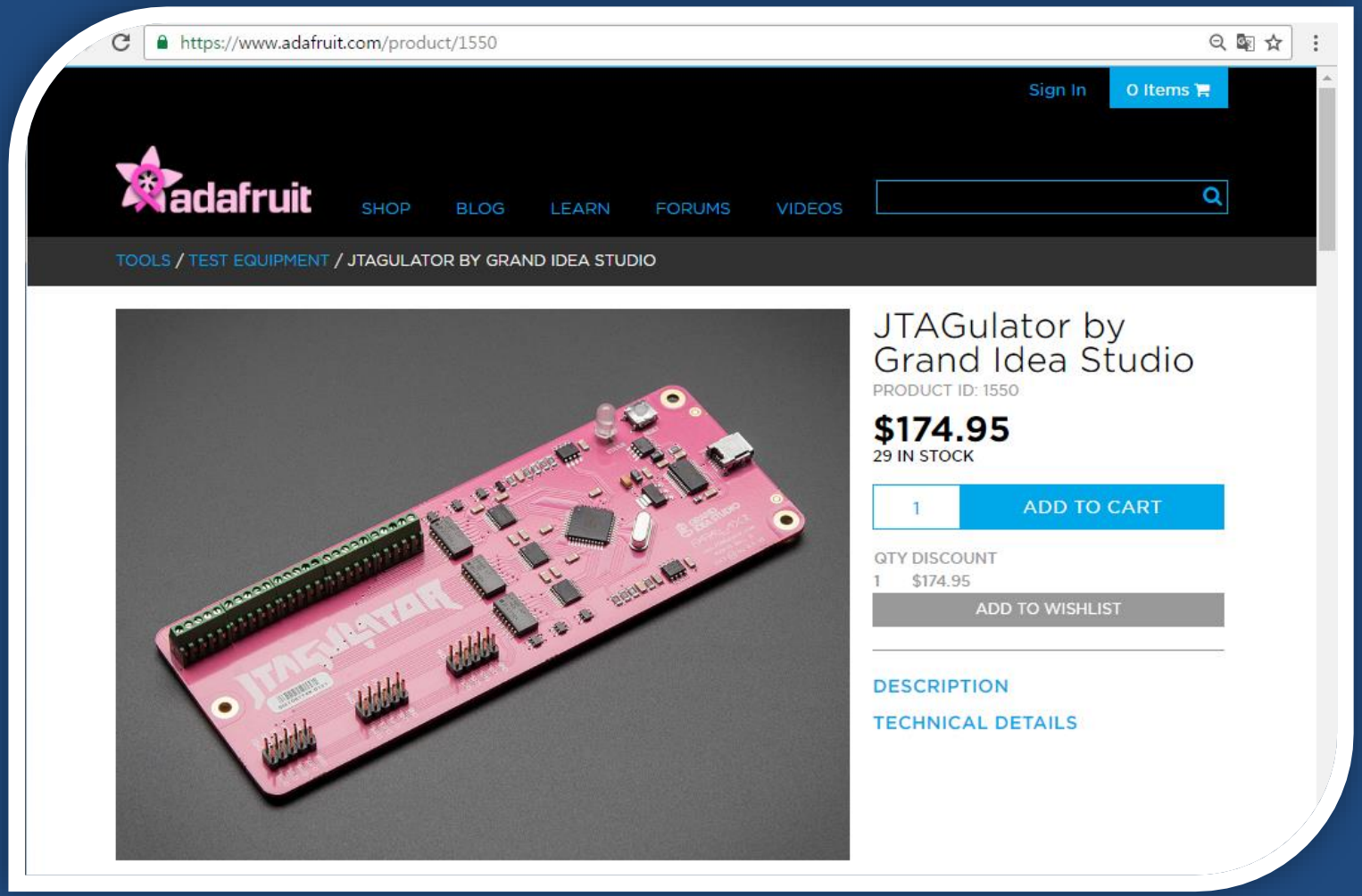
JTAG 찾기 예제



JTAG 찾기 예제

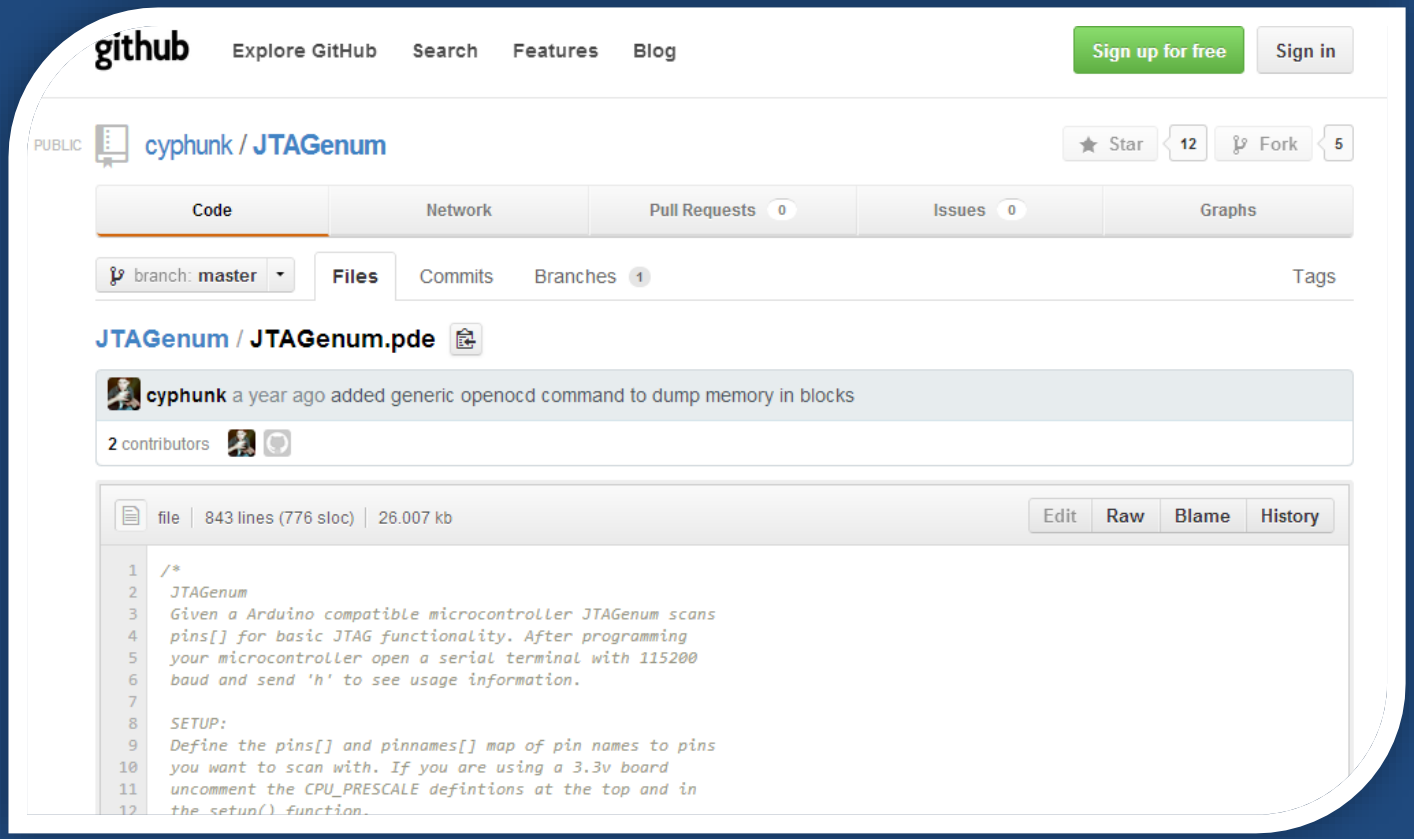


JTAG Pin Scanner

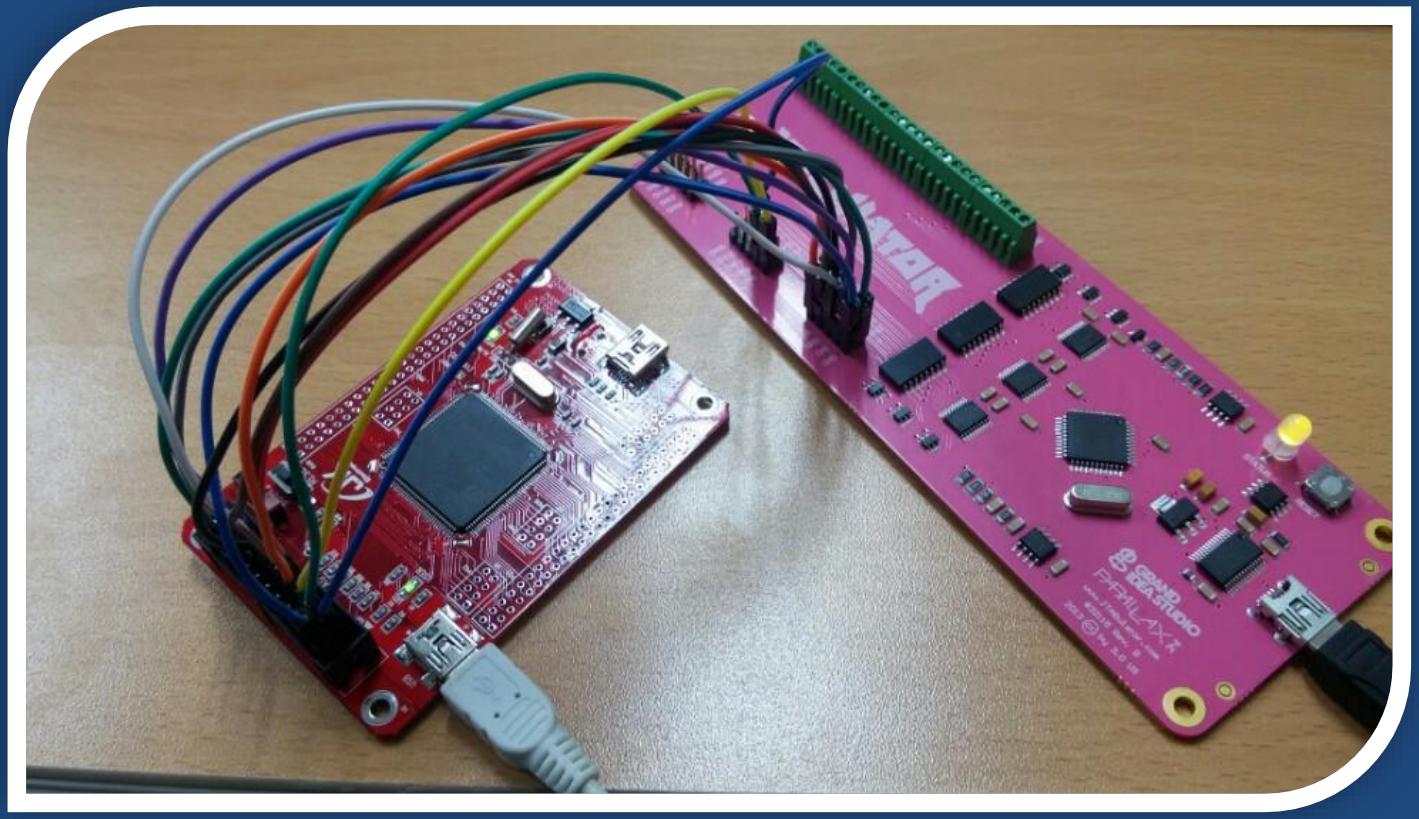


JTAG Pin Scanner

- https://github.com/cyphunk/JTAGenum/blob/master/JTAGenum.pde



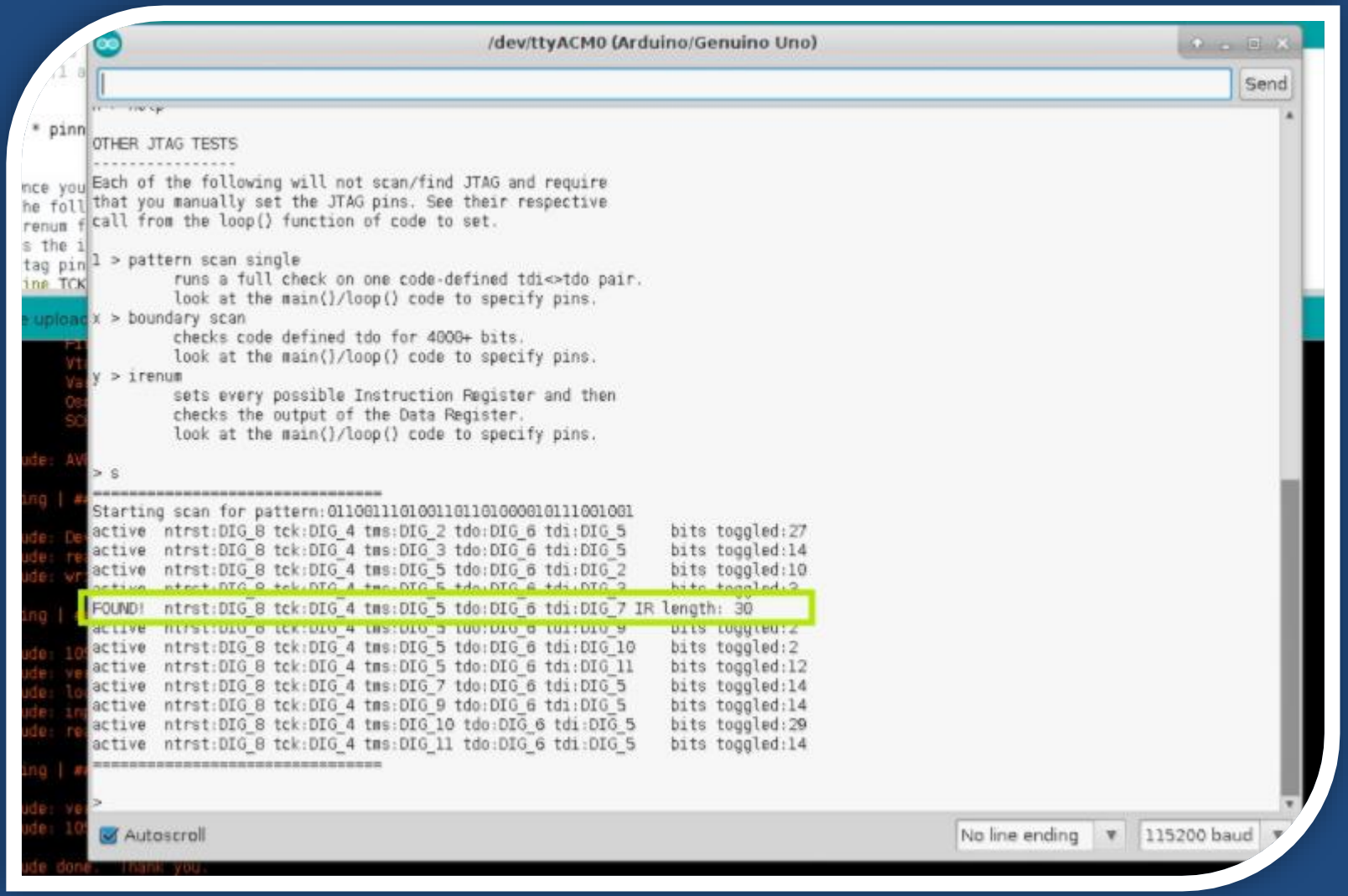
JTAGulator 사용 예제



JTAGulator 사용 예제

```
.
:I
Enter number of channels to use (3 - 24): 11
Ensure connections are on CH10..CH0.
Possible permutations: 990
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort.......
TDI: N/A
TDO: 4
TCK: 6
TMS: 7
TRST#: 9
......
IDCODE scan complete!
:b
Enter number of channels to use (4 - 24): 11
Ensure connections are on CH10..CH0.
Possible permutations: 7920
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort.................................
.......................................................................
.......................................................................
.......................................................................
.......................................................................
TDI: 8
TDO: 4
TCK: 6
TMS: 7
TRST#: 9
Number of devices detected: 2
.......................................................................
```

JTAGEnum 사용 예제



결 론

- JTAG은 개발자는 물론 해커에게 매우 유용한 디버깅 시스템이다.
- JTAG을 이용하면 Flash Memory 덤프 및 Real-time debugging이 가능해 진다.
- JTAG 연결에 성공하면, 대상 기기의 내부를 장악한 것과 다름 없다.

Q/A

감사합니다.