How 2 Debug?

...

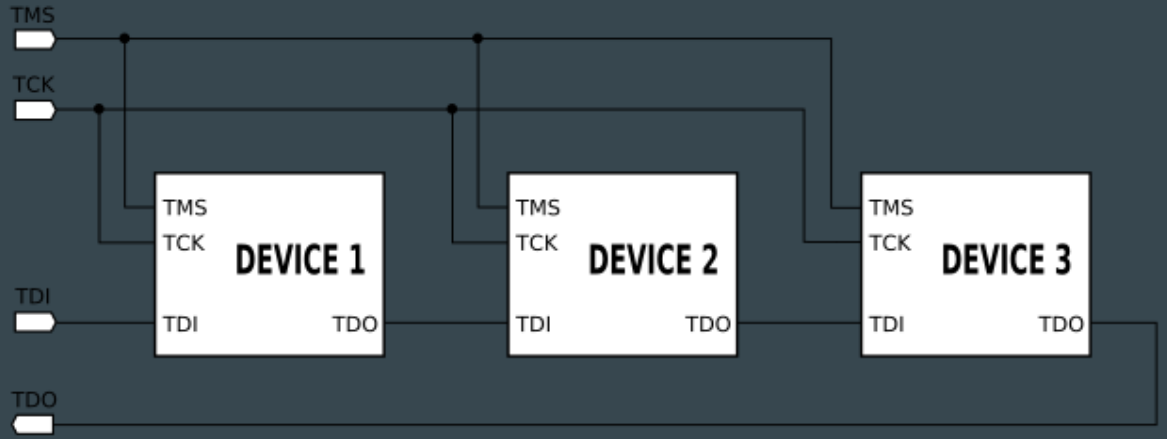
A Dive into Debugging Hardware

Some History - 1980

- PCBs were getting more advanced
  - multi-layer circuit boards
  - ICs w/ BGA and similar mounting technologies were becoming standard
- Connections between ICs couldn't be probed
- Boards would fail due to bad solder connections going unnoticed

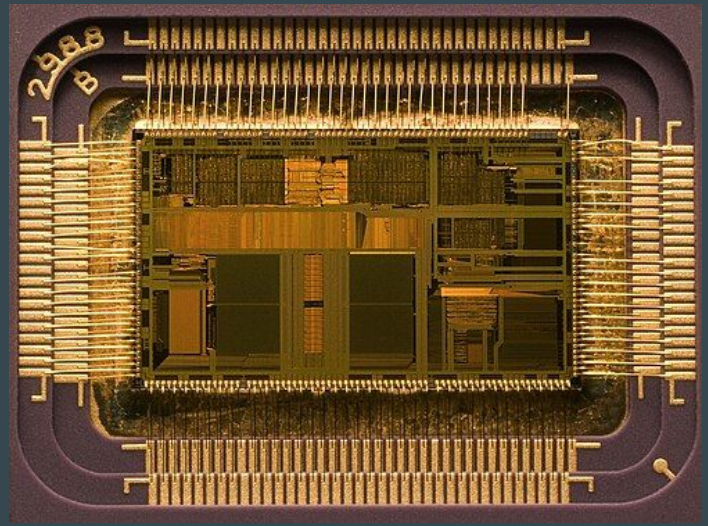
Some History - 1985

- Group was formed to create testing standard
  - Created Joint Test Action Group (JTAG)
  - Used to interact with pins in an IC (boundary scan)
    - Can test connectivity without physical access
  - Very simple state machine
  - Pretty specific function



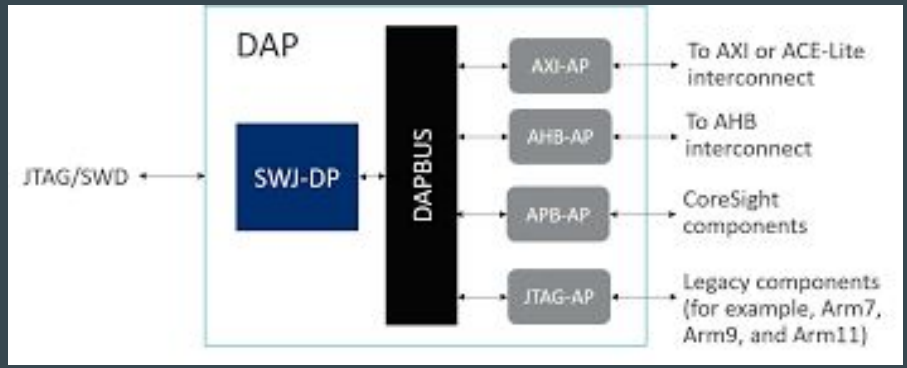
Some History - 1990

- Became official standard
  - IEEE Std. 1149.1-1990
- Intel released first CPU with JTAG (80486)
  - Lead to widespread adoption
- 1994: Boundary scan description language was added
  - Standardized testing pins on different IC packages



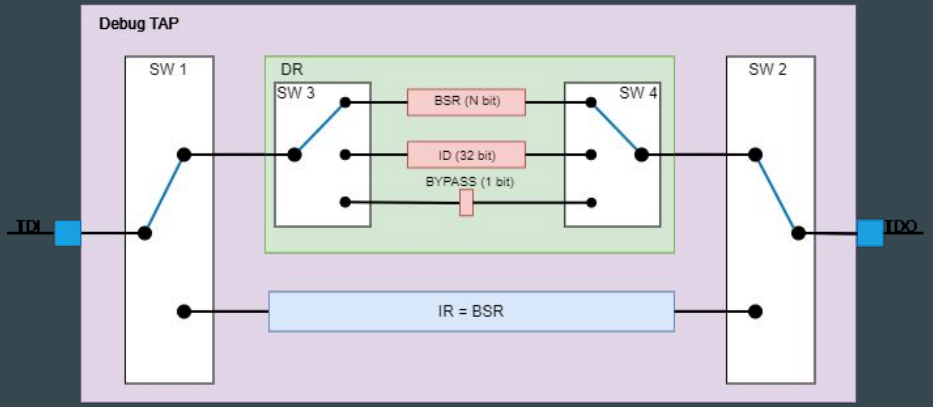
Some History - Now

- JTAG has become a ubiquitous protocol found on many IC's
- Many variations
  - Serial Wire Debug (SWD)
  - Compact JTAG (cJTAG)
- Used to access submodules
  - Provides simple way to get access to core functions at first instruction
- Many extensions that add functionality
  - ARM's CoreSight: processor core debug
  - Nexus: Vendor independent debugging interface
  - Uploading firmware



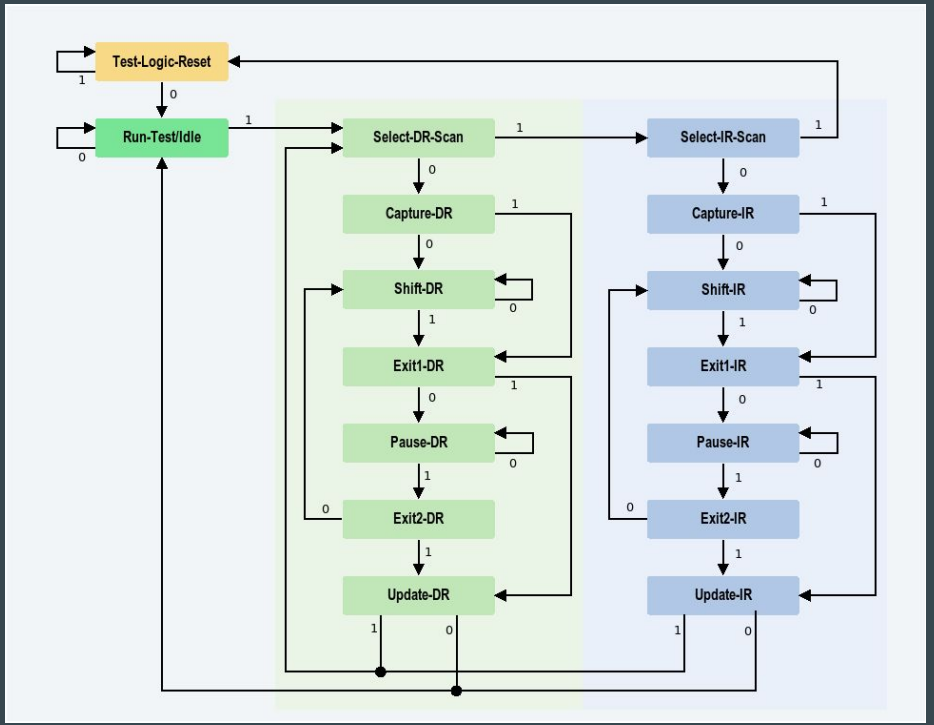
JTAG

- Test Access Point(TAP)
  - Finite State Machine
  - 2 Registers
    - IR: Controls the “instruction”/register that DR uses
    - DR: Used to give/take data
  - Very simple primitive to write to many registers
  - Required registers
    - ID (IDCODE)
    - Boundary Scan Register (BSR)
      - SAMPLE
      - PRELOAD
      - EXTEST
      - *INTEST
    - Bypass Register (BR)



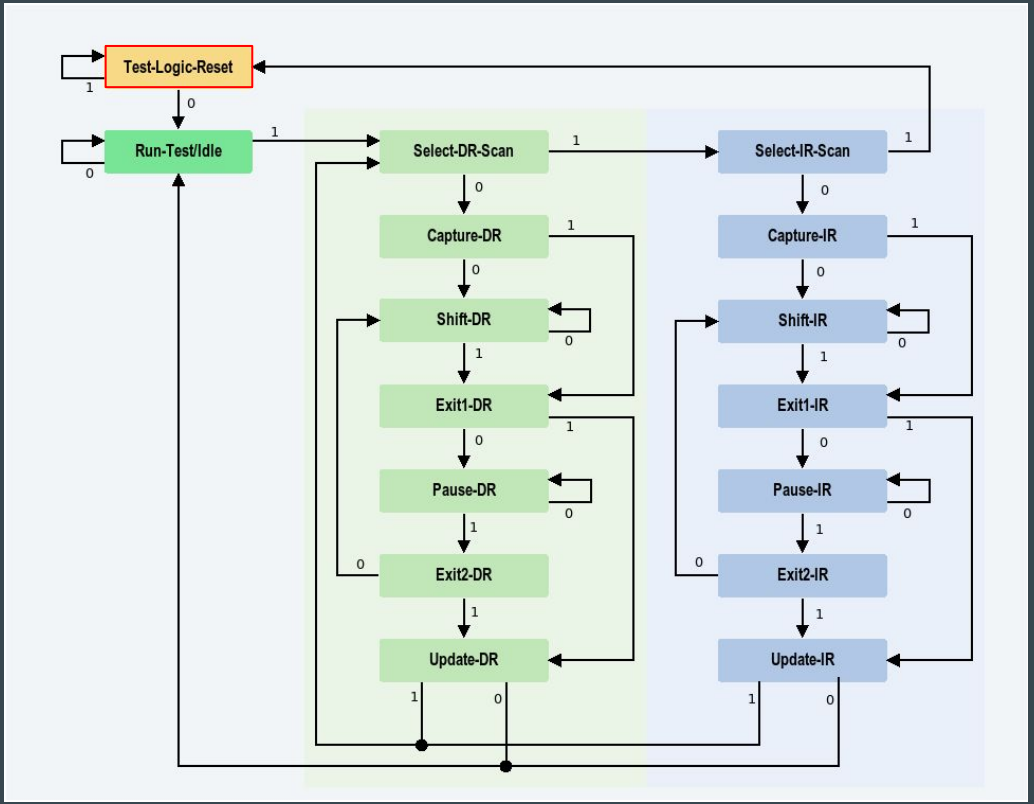
JTAG State Machine

- Controlled via TMS and TCK pins
- Can always reach reset by setting TMS high 5 times
    - Can be used to synchronize multiple state machines in JTAG chain



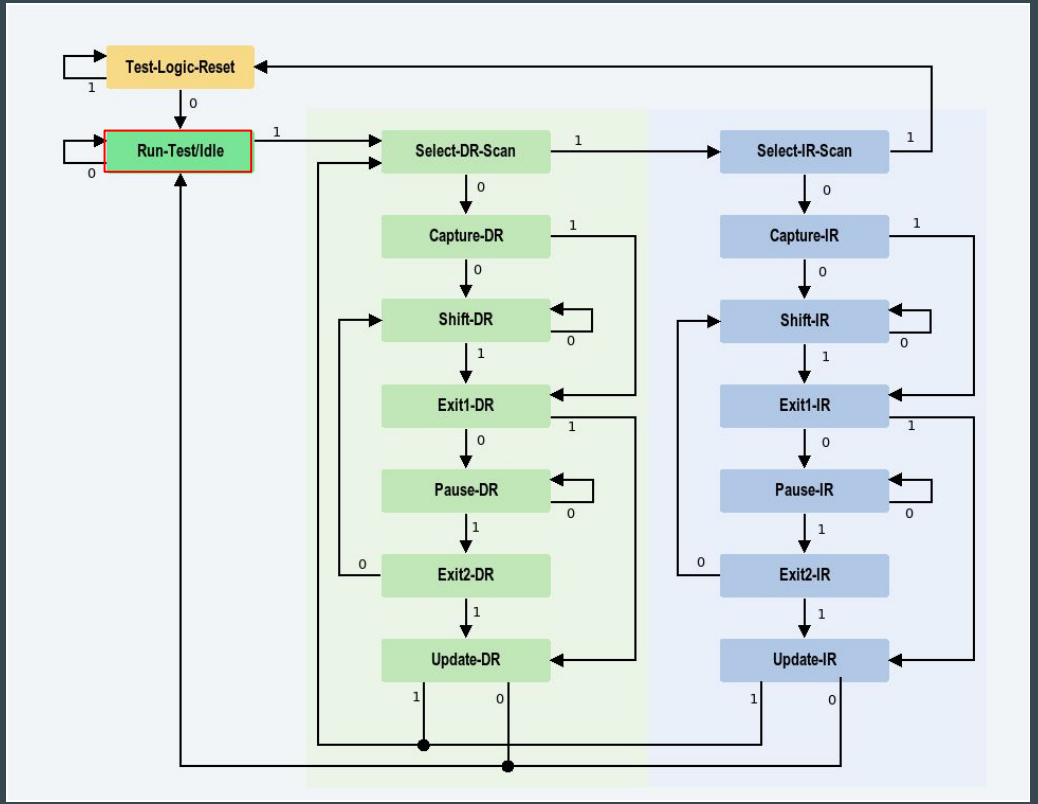
Example: Setting all pins high

Initial reset state



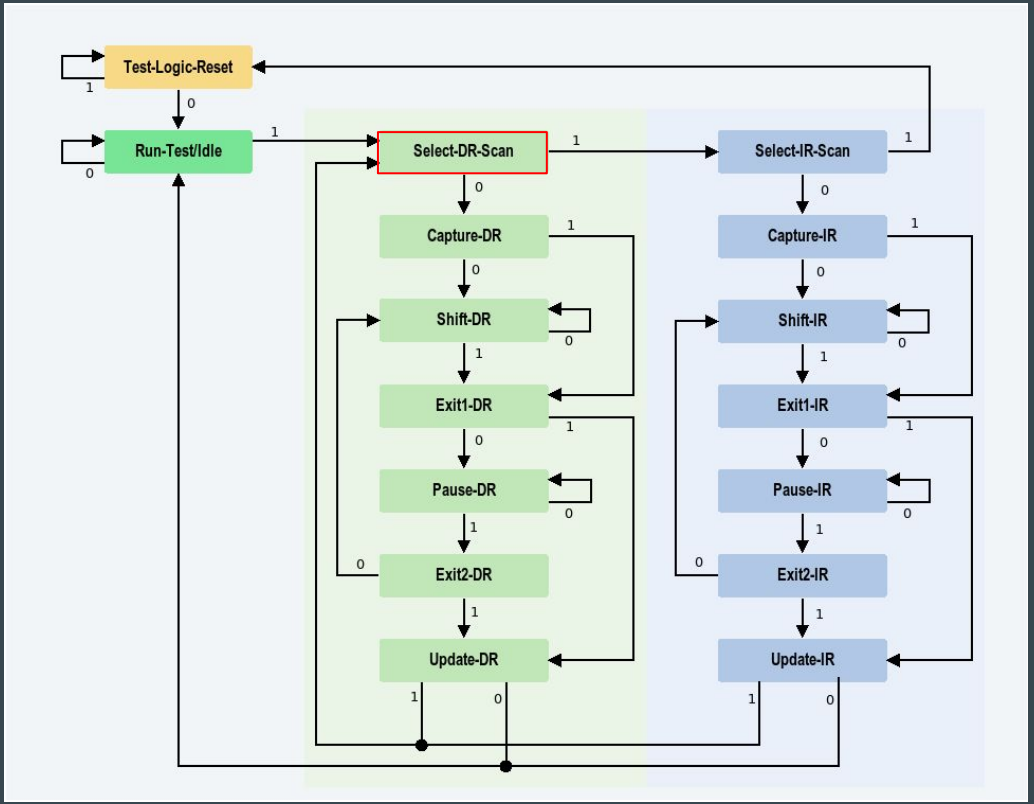
Example: Setting all pins high

TMS 1 → Idle



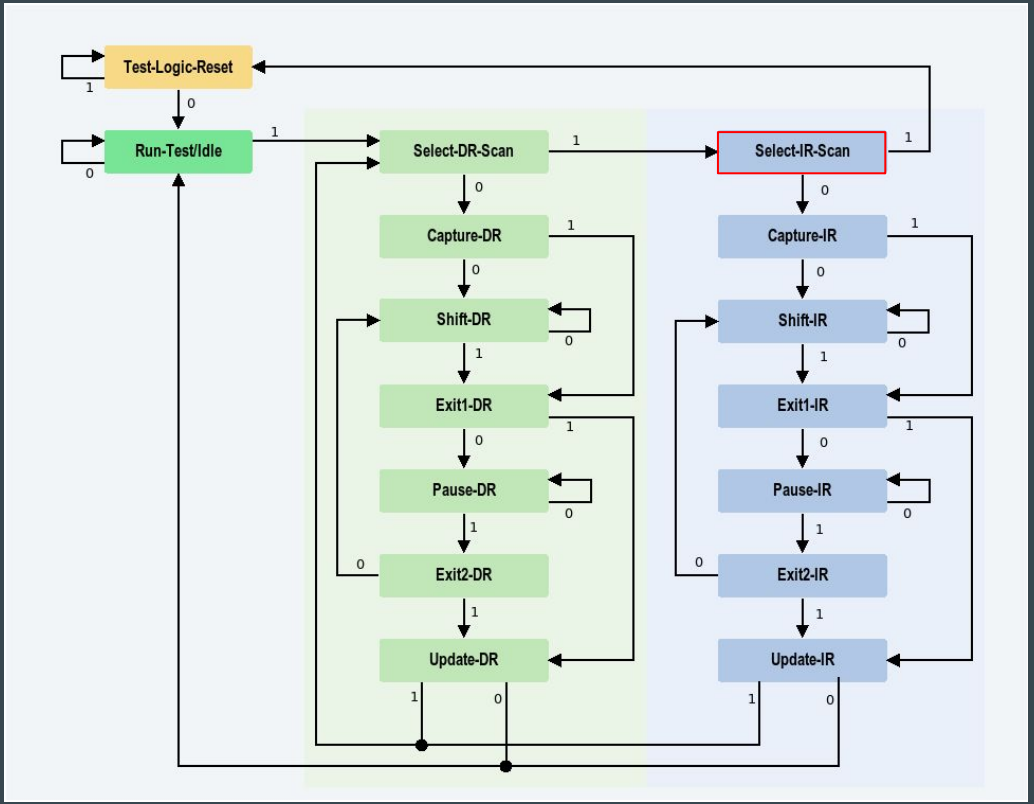
Example: Setting all pins high

TMS 1 → Select DR



Example: Setting all pins high

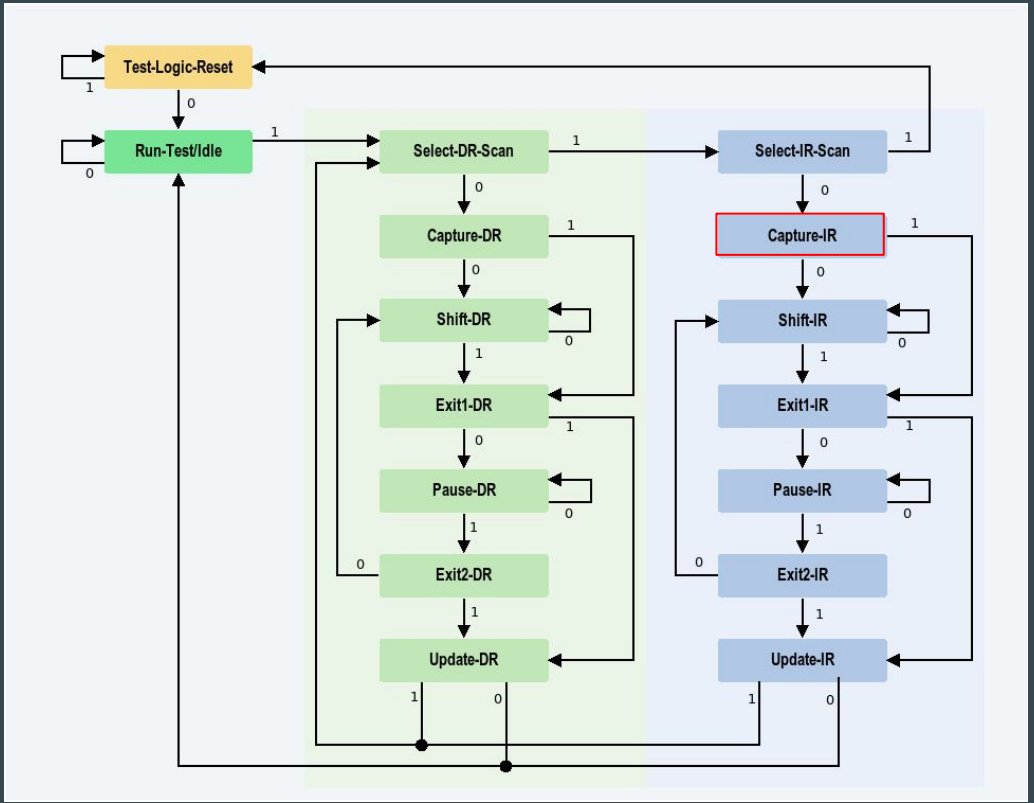
TMS 1 → Select IR



Example: Setting all pins high

TMS 0 → Capture IR

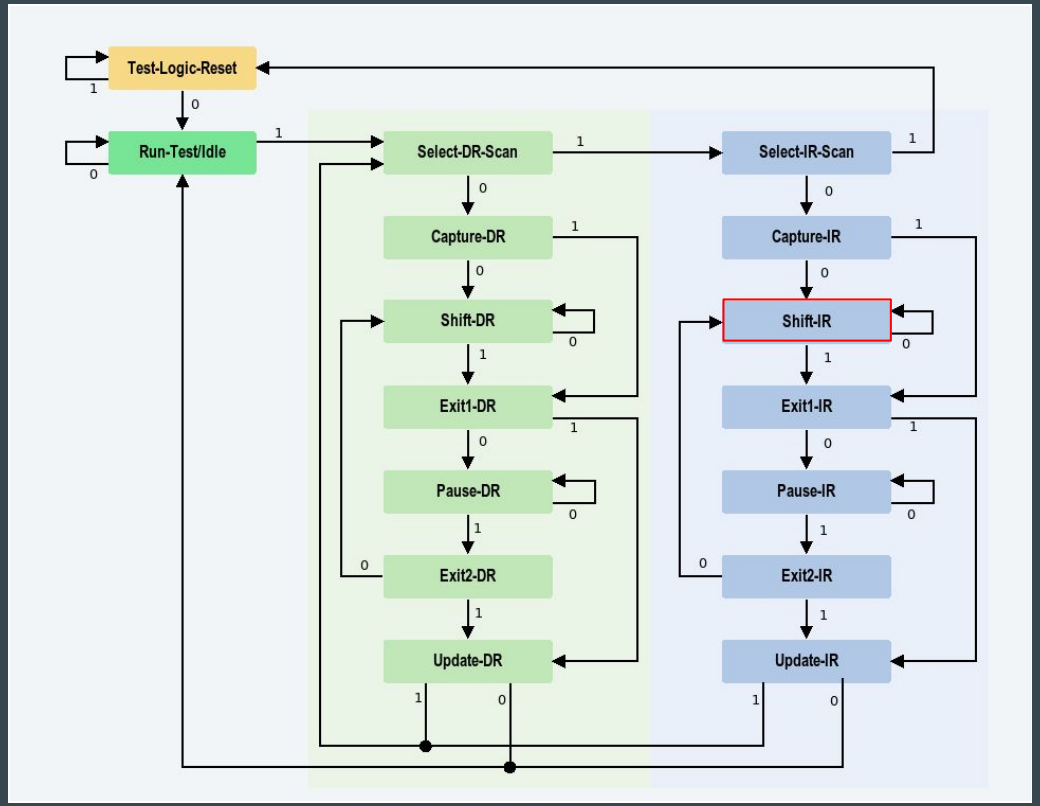
- Shift register stores prev IR



Example: Setting all pins high

TMS 0 → Shift IR

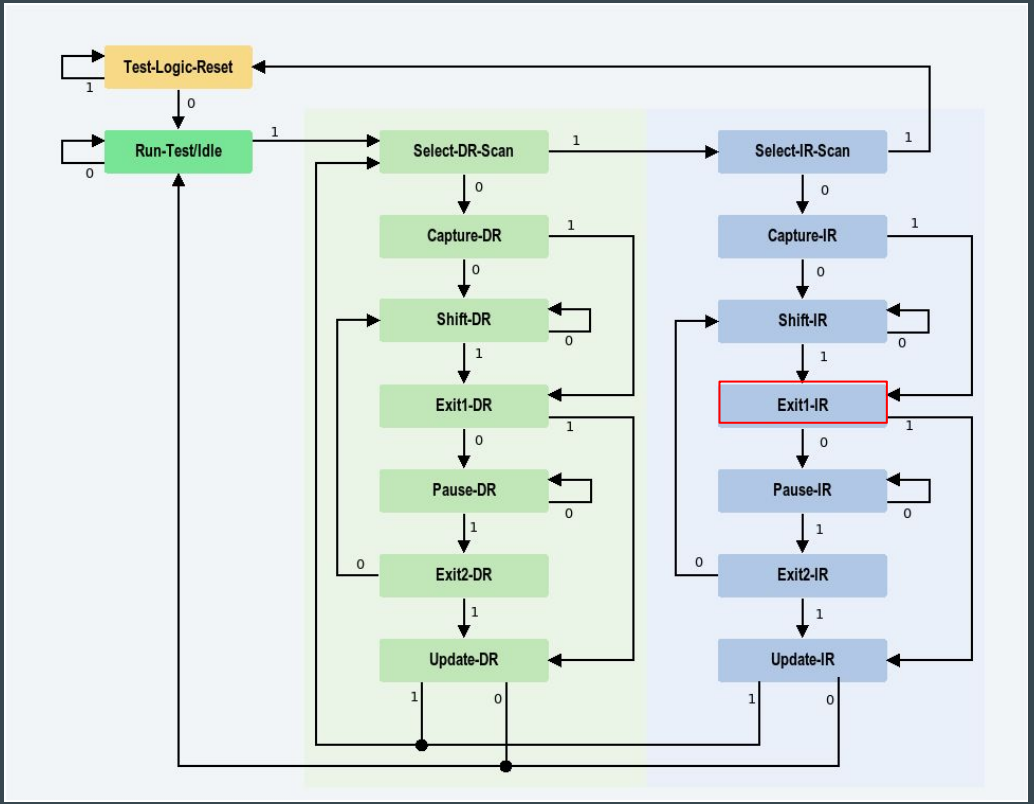
- Shift values in/out of shift register
  - Get prev IR
  - Store new IR (PRELOAD)



Example: Setting all pins high

TMS 1 → Exit 1 IR

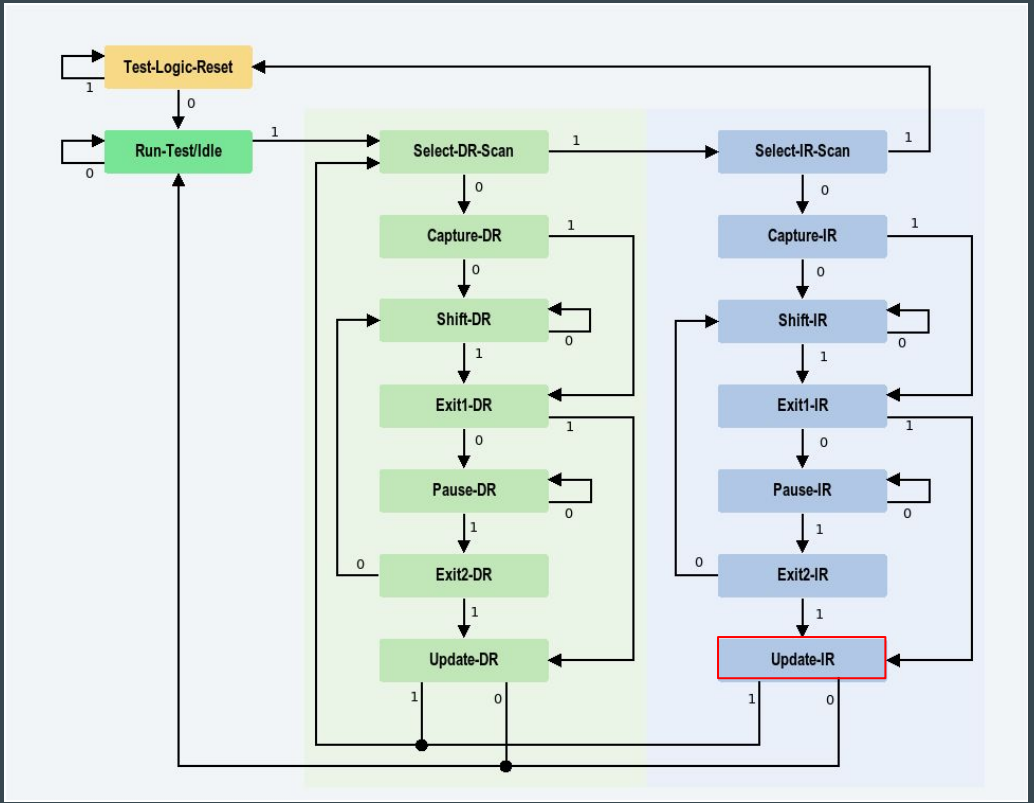
- Shift register contains PRELOAD instruction value



Example: Setting all pins high

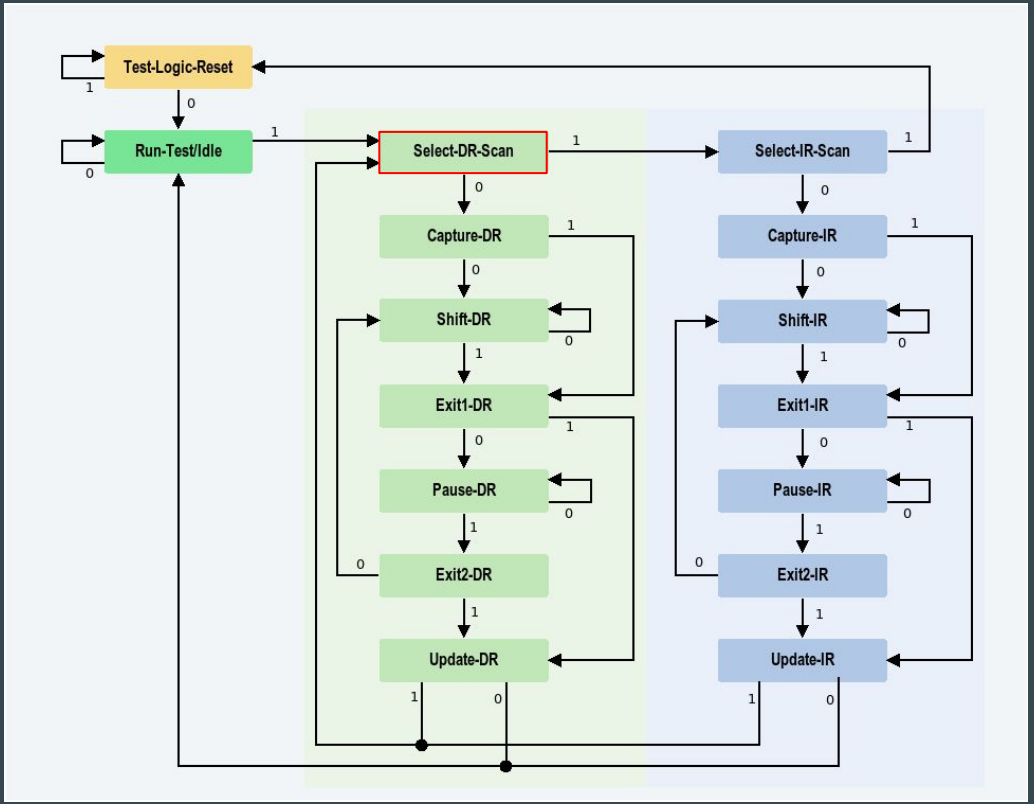
TMS 1 → Update IR

- Store shift register in IR



Example: Setting all pins high

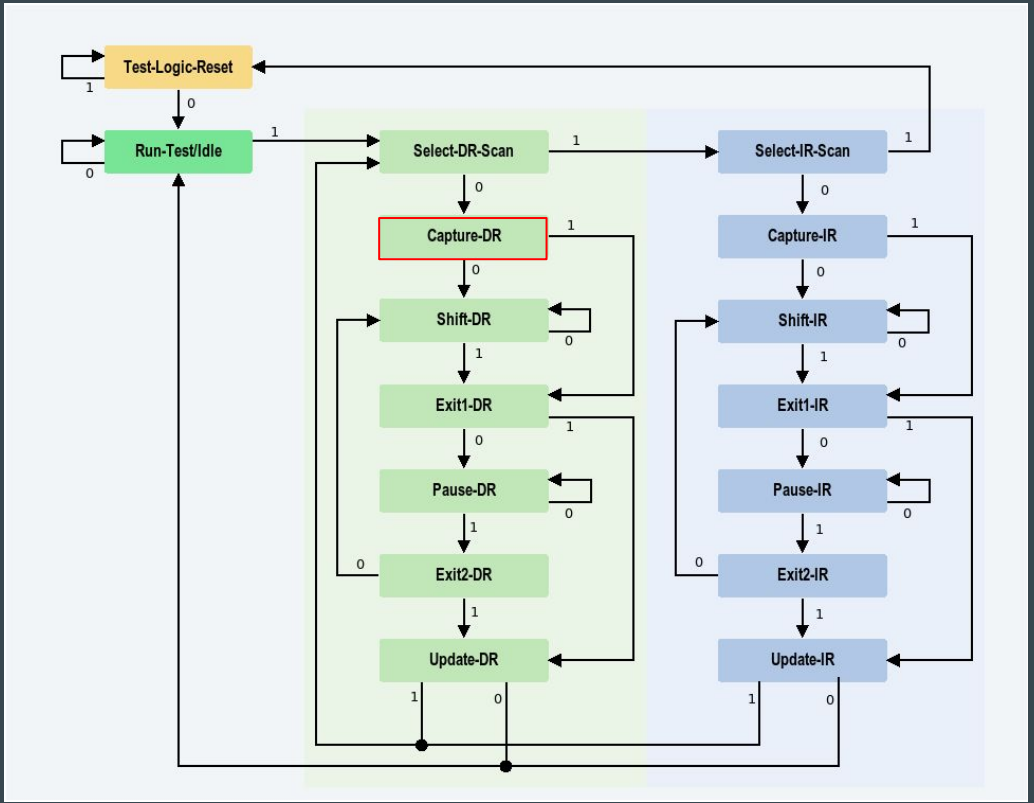
TMS 1 → Select DR



Example: Setting all pins high

TMS 0 → Capture DR

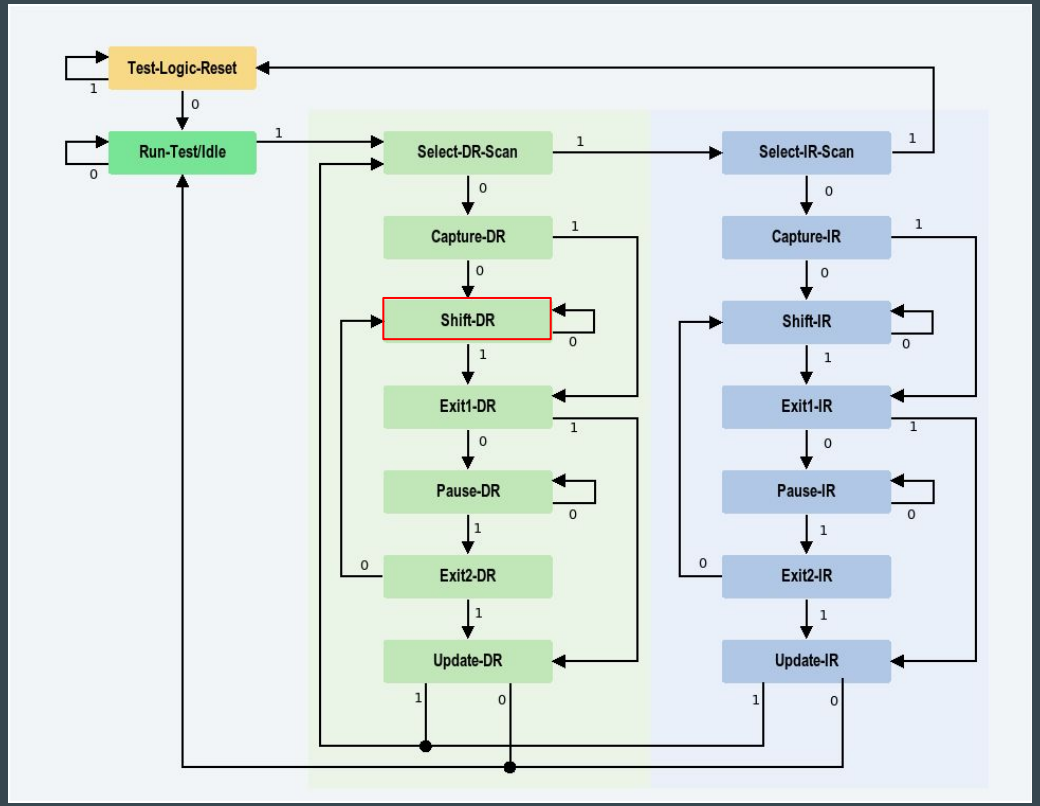
- Shift register stores prev DR



Example: Setting all pins high

TMS 0 → Shift DR

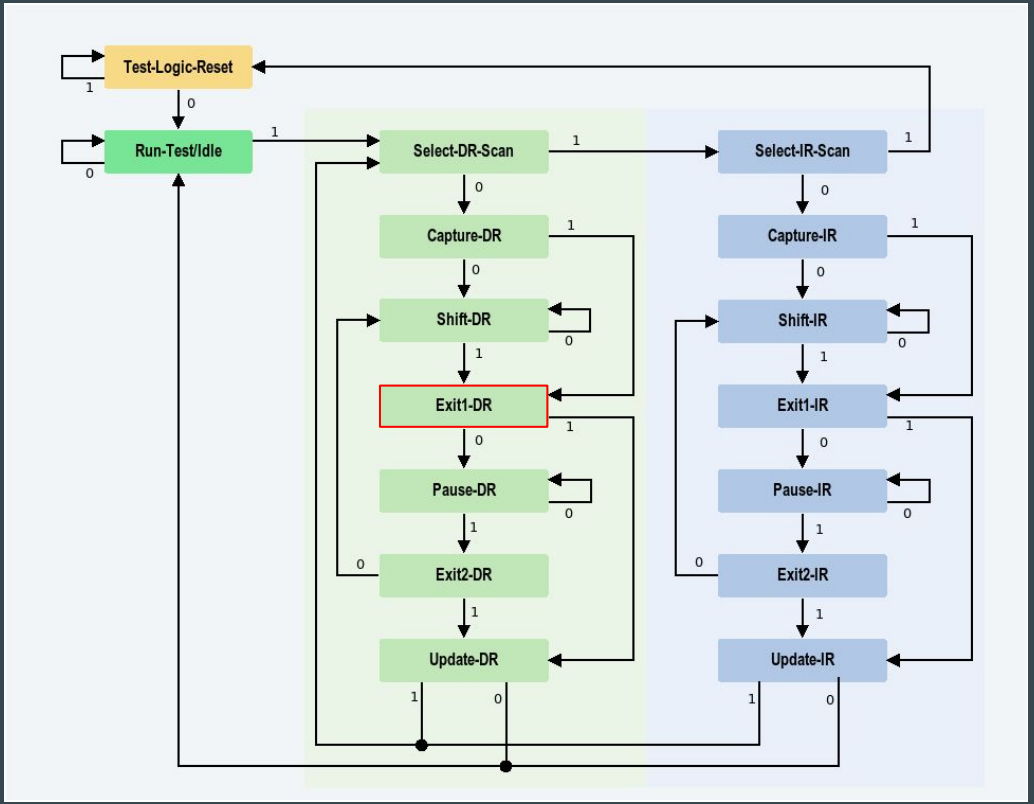
- Shift values in/out of shift register
  - Get prev DR (data out)
  - Store new DR (data in)



Example: Setting all pins high

TMS 1 → Exit 1 DR

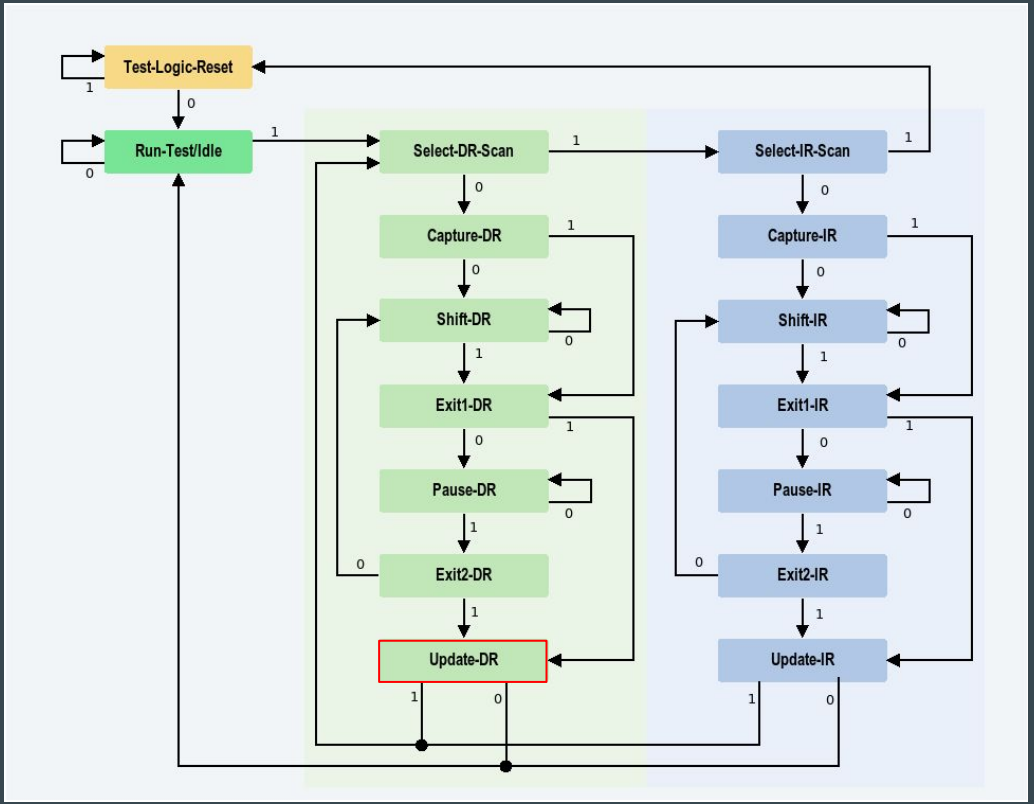
- Shift register contains all 1 (pin high)



Example: Setting all pins high

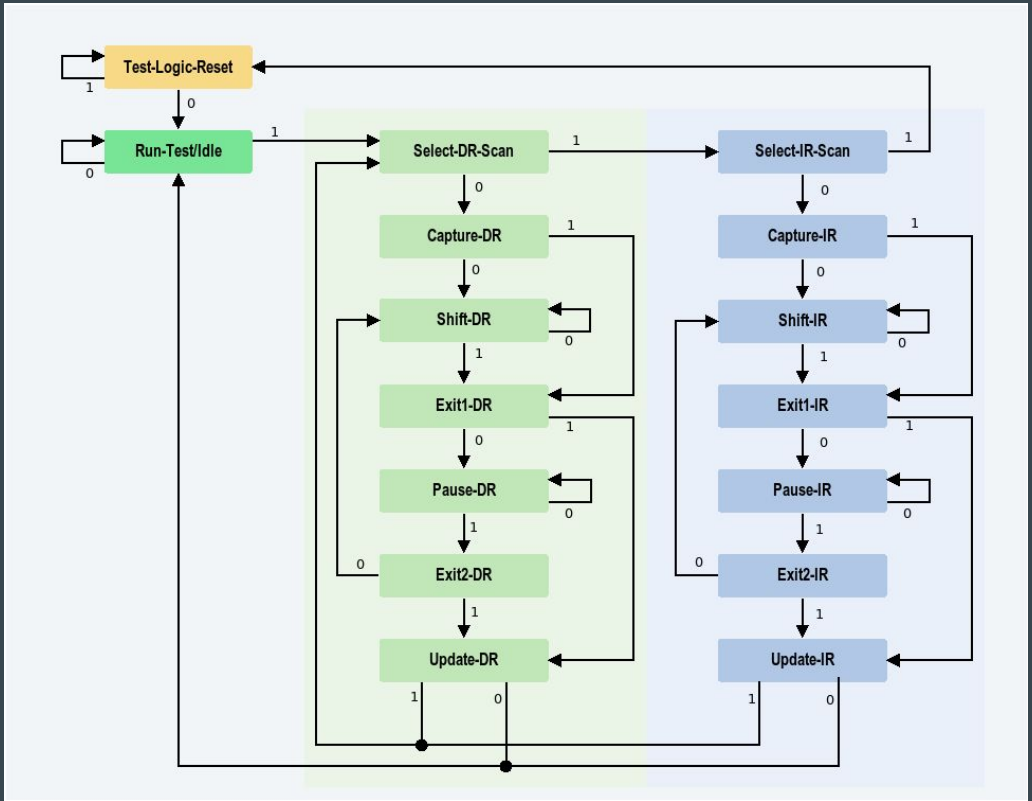
TMS 1 → Update DR

- Store shift register into DR



Example: Setting all pins high

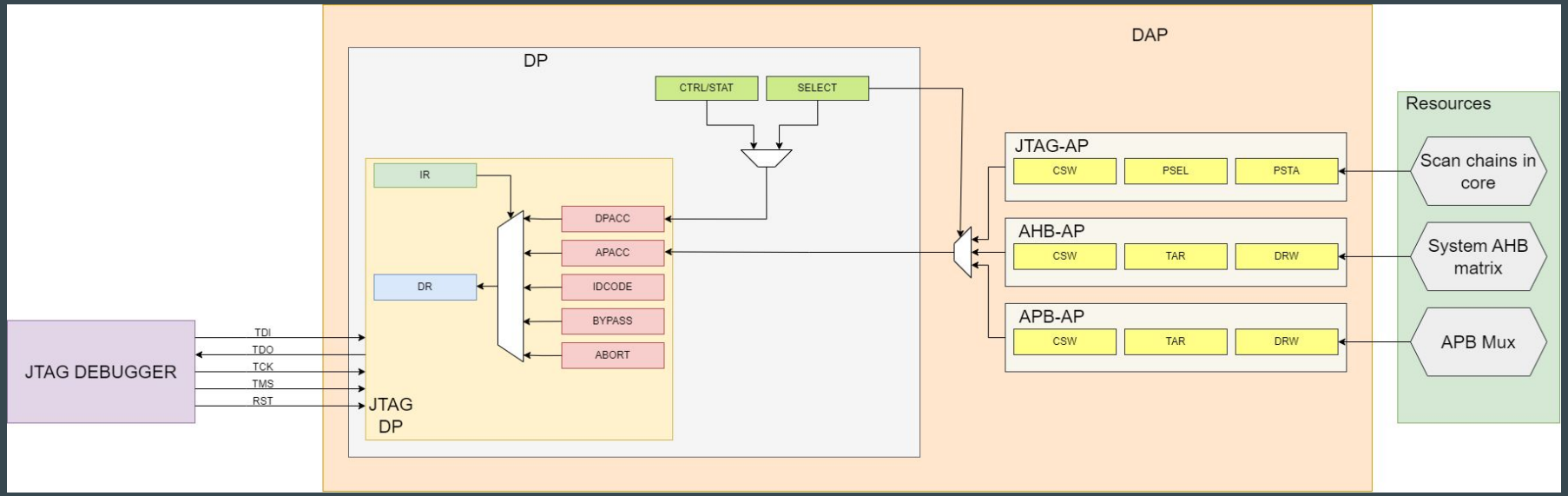
We also need to set IR to EXTEST



Debugging with JTAG (ARM)

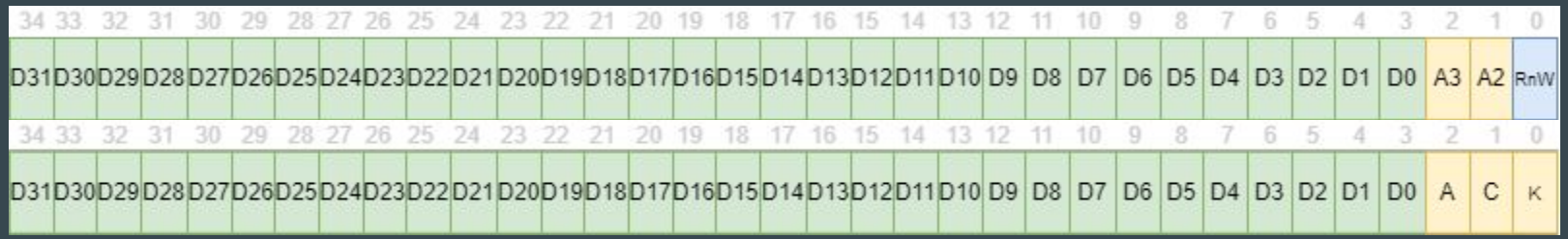
Terminology - Debug Port (DP), Access Port (AP), Debug Access Port (DAP)

Everything is registers!



Debugging with JTAG (ARM) - Special JTAG instructions

- Each instruction corresponds to 35 bit DR
- 0b1000 (ABORT): Interrupt current AP transaction and give control back to debugger (extreme!)
- 0b1010 (DPACC): Debug port access register
    - Used to control CTRL/STAT and SELECT registers
- 0b1011 (APACC): Access port access register
    - Used to read and write to the selected register in the access port



R/W of DPACC/APACC

Debugging with JTAG (ARM) - CTRL/STAT

Control and status information about the DP

- [31] RO CSYSPWRUPACK System power-up acknowledge
- [30] R/W CSYSPWRUPREQ System power-up request
- [29] RO CDBGPWRUPACK Debug power-up acknowledge
- [28] R/W CDBGPWRUPREQ Debug power-up request
- [27] RO CDBGRSTACK Debug reset acknowledge
- [26] R/W CDBGRSTREQ Debug reset request
- [25:24] - - Reserved, RAZ/SBZP
- [23:12] R/W TRNCNT Transaction counter
- [11:8] R/W MASKLANE Indicates the bytes to be masked in pushed compare and pushed verify operations
- [7] RO WDATAERR Set if a Write Data Error occurs
- [6] RO READOK This bit is set to 1 if the response to the previous AP or RDBUFF read was OK
- [5] RO STICKYERR Set if an error is returned by an AP transaction
- [4] RO STICKYCMP This bit is set to 1 when a match occurs on a pushed compare or a pushed verify operation
- [3:2] R/W TRNMODE Transfer mode for AP operations: 00 = Normal operation, 01 = Pushed verify operation, 10 = Pushed compare operation, 11 = Reserved.
- [1] RO STICKYORUN Overrun error flag
- [0] R/W ORUNDETECT This bit is set to 1 to enable overrun detection

Debugging with JTAG (ARM) - SELECT

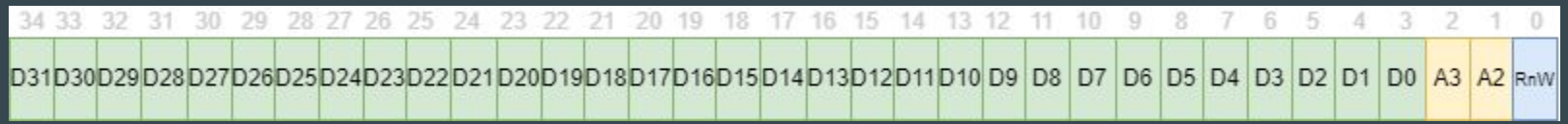
Select AP to interact with

[31:24] APSEL Selects the current AP (AHB-AP, APB-AP, JTAG-AP)

[23:8] - Reserved

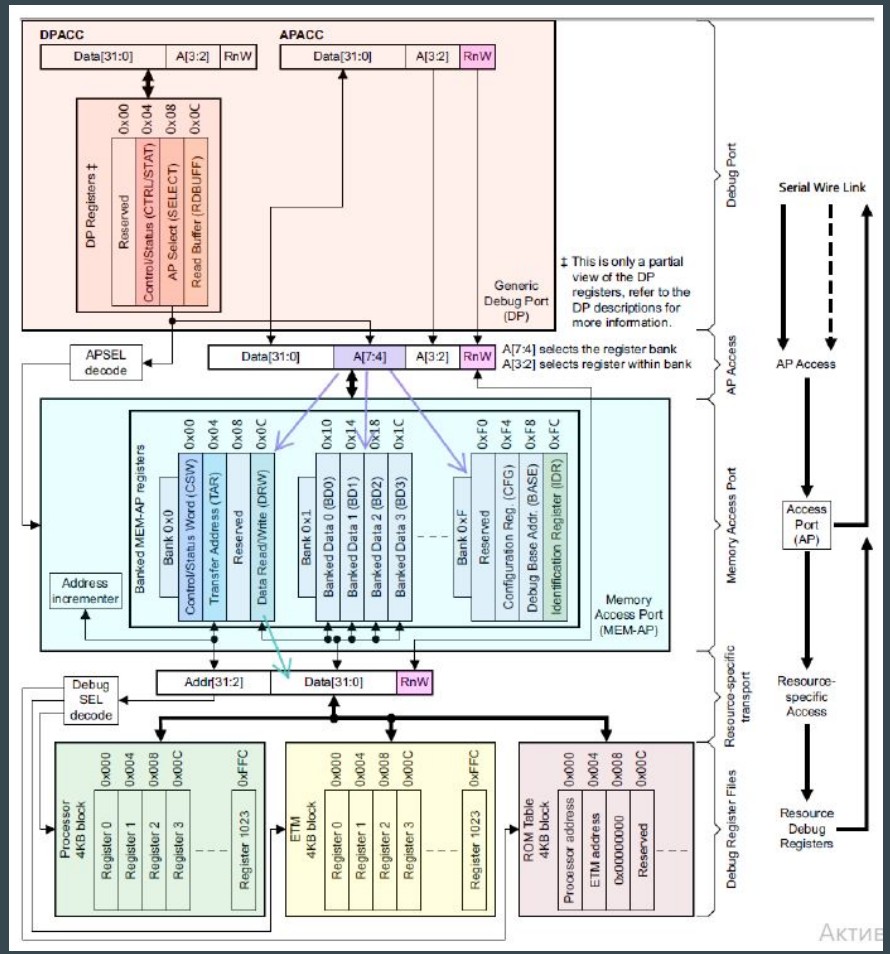
[7:4] APBANKSEL Select the active four-word register bank on the current AP

[3:0] - Reserved



Debugging with JTAG (ARM) - Memory AP

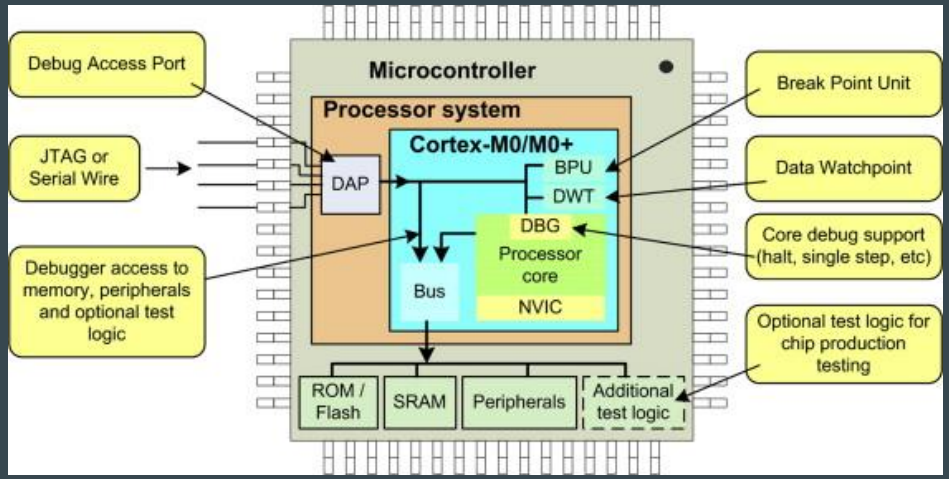
MEM-AP register	Address	Register bank	Offset (A[3:2])
Control/Status Word	0x00	0x00	0b00
Transfer Address Reg	0x04	0x00	0b01
Data R/W	0x0C	0x00	0b11
Banked Data 0	0x10	0x01	0b00
Banked Data 1	0x14	0x01	0b01
Banked Data 2	0x18	0x01	0b10
Banked Data 3	0x1C	0x01	0b11
ConFiguration Reg	0xF4	0x0F	0b01
BASE	0xF8	0x0F	0b10
IDentification Reg	0xFC	0x0F	0b11


DPACC
Data[31:0]
A[3:2]
RnW
APACC
Data[31:0]
A[3:2]
RnW
DP Registers ‡
0x00
0x04
0x08
0x0C
Reserved
Control/Status (CTRL/STAT)
AP Select (SELECT)
Read Buffer (RDBUFF)
Generic Debug Port (DP)
‡ This is only a partial view of the DP registers, refer to the DP descriptions for more information.
Debug Port
Serial Wire Link
APSEL decode
Data[31:0]
A[7:4]
A[3:2]
RnW
A[7:4] selects the register bank
A[3:2] selects register within bank
AP Access
AP Access
Banked MEM-AP registers
Bank 0x0
Control/Status Word (CSW)
Transfer Address (TAR)
Reserved
Data Read/Write (DRW)
Bank 0x1
Banked Data 0 (BD0)
Banked Data 1 (BD1)
Banked Data 2 (BD2)
Banked Data 3 (BD3)
Bank 0xF
Reserved
Configuration Reg. (CFG)
Debug Base Addr. (BASE)
Identification Register (IDR)
Address incrementer
Memory Access Port (MEM-AP)
Memory Access Port
Access Port (AP)
Addr[31:2]
Data[31:0]
RnW
Resource-specific transport
Debug SEL decode
Resource-specific Access
Processor 4KB block
ETM 4KB block
ROM Table 4KB block
Register 0
Register 1
Register 2
Register 3
Register 1023
Processor address
ETM address
0x00000000
Reserved
Debug Register Files
Resource Debug Registers

Debugging with JTAG (ARM) - CoreSight

BASE Register

- Usually points to ROM table
  - Used by CoreSight to store information about debug components
  - Can point to more ROM tables contains more debug information
  - Debugger can find all debugging capabilities by traversing ROM tables
  - Flexible debug components with various features
    - Flash breakpoint (FPB)
    - Embedded Trace Macrocell (ETM)
    - Embedded Cross Trigger (ECT)

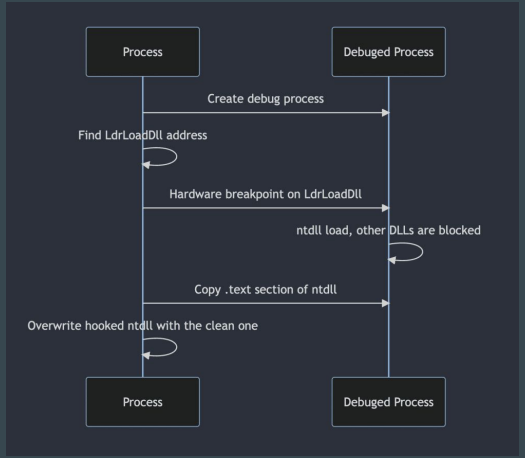


Hardware Debugging Advantages - Developer

- Faster and more controlled debugging
  - Hardware breakpoints are faster and allow for masking
  - Traces get real time data without slowing down system
    - Embedded Trace Macrocell: All instructions
    - Program Trace Macrocell: All changes in program flow
  - Watchpoints
- Access to specific buses and sub-components
- Lightweight

Hardware Debugging Advantages - Attacker

- Runs below any software
  - No software protections can stop it*
  - Can get around software protections
    - EDR Evasion with Hardware Breakpoints: Blindside Technique
- Impossible to deactivate unless specific hardware security measures are implemented
  - Extra cost to implement

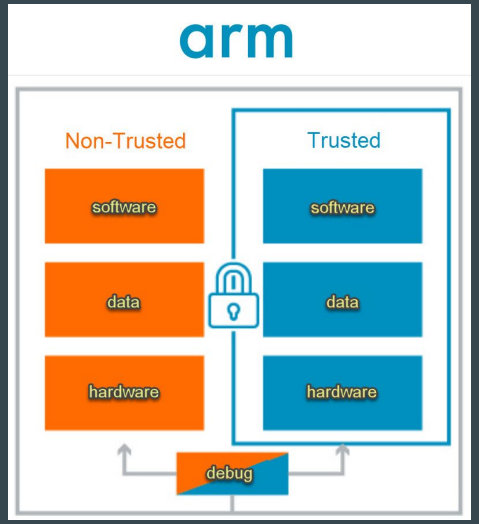


Hardware Debugging Disadvantages

- Requires specific hardware
  - Expensive
  - If not implemented, no debugging :(
- Requires physical access
  - Out of luck if JTAG/SWD pins are not exposed
- Limited abilities
  - Ex: Can only set ~16 watchpoints vs ∞ software breakpoints

Security

- E-fuses
  - Can be checked at boot to enable bootloader/software debug access
  - Can use voltage glitching to trick processors while booting
- Trustzone
  - Create trusted zones and protections that limit the effectiveness of SWD
- Secure Enclave / Coprocessor
  - Authentication is done in another processors which generates/relays SWD signals



Practical Example

It didn’t work :(

NECCDC Applications open soon!

If you're interested in blue teaming, this is a competition where you defend a simulated corporate network against industry professional hackers!

- Fill out our interest form!
- Application materials will be released next week.

Interest form

Join us for MinutemanCTF!

Where?

LGRC A112

When?

Oct 17 @ 6:00 PM

Want a challenge?

Want to see industry and professor panels?

Come play MinutemanCTF!

Questions?

How do I learn more?
How can I get involved?
When are you guys available?

Come Up & Ask!

Resources Posted in Discord

Newsletter

Discord

Twitter

Website