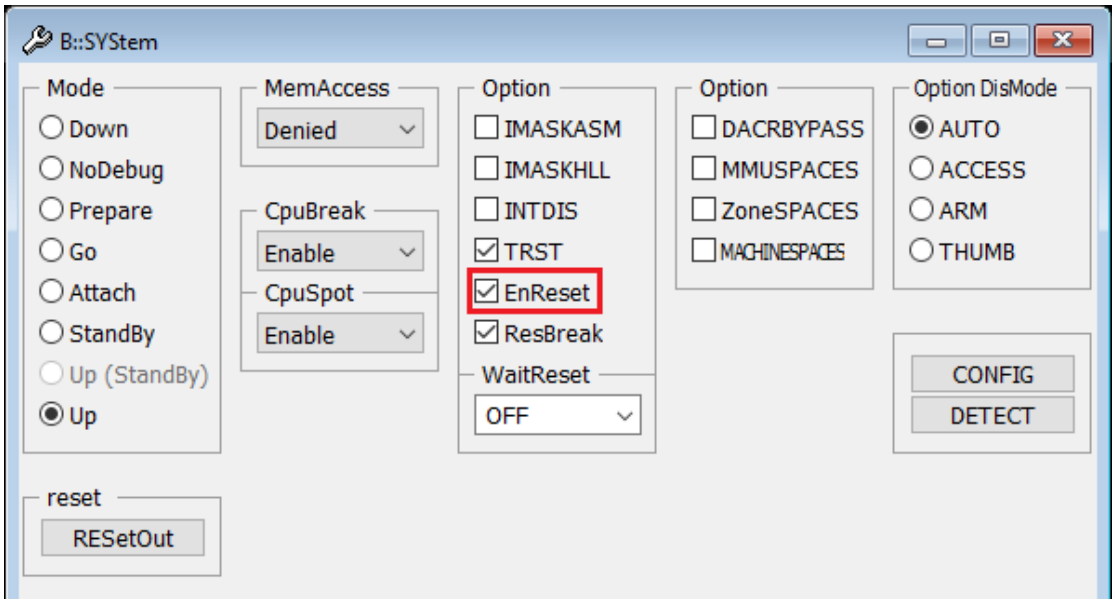
[Arm] Target resets and boots correctly after pushing the reset button / after a power-on reset, but not after a SYStem.Up+Go

2026-04-03 - Comments (2) - Arm

The debugger uses the JTAG reset pin (**nSRST**) available on the debug connector (e.g., pin 15 on the 20-pin connector or pin 10 on the 10-pin MIPI connector).

The `SYStem.Up` command only asserts this reset line if the option `SYStem.Option.EnReset` is set to `ON`.

Please check if this option is enabled in the **SYStem** window (menu CPU > System Settings...):



Please also check whether the debugger's reset line is properly connected to the chip reset (i.e., the same reset source as the reset button).

It is also possible that the reset line, which is pulled low during `SYStem.Up` by default, does not reset the entire SoC. In such cases, `SYStem.Up` cannot fully reset the device. As a result, the processor may not start from a clean reset context, which can lead to boot issues. In addition, the reset button and the nSRST pin may be connected to different points on the board. Therefore, check the board schematics carefully. Some designs include reset logic that treats the reset button and nSRST differently. As a test, you can disconnect nSRST and reroute it to the reset button signal to ensure that the debugger triggers the same reset behavior.

Please also check whether a SoC-specific reset mechanism via registers is available (i.e., a core reset triggered by writing to a dedicated register). If so, this mechanism can be used together with the command `SYStem.Option.RESetRegister`.

If the reset cannot be achieved with a single register write (e.g., if multiple registers must be configured), the reset sequence should be implemented in a PRACTICE script. For example, the script can access the reset registers via AHB/AXI in `SYStem.Mode Prepare`. After performing the reset, the debugger can connect using `SYStem.Mode Attach`. Note that `Attach` does not guarantee that the CPU is stopped at the reset vector. To

ensure this, you can place an endless loop at the reset vector before releasing the CPU from reset. This can also be done in *Prepare* mode if the CPU memory is accessible via AHB/AXI. In this case, use the `Data.Assemble` command as shown in the template script below:

```
// This example outlines a rough scheme how a CPU could be reset via a custom
// register sequence and held at the reset location if the CPU starts from a
// writable RAM location, e.g. SRAM

SYStem.CPU <cpu>
SYStem.Mode Prepare

Data.Set [EAHB | EAXI]:<register1> % <value>   // Bring core(s) into reset
Data.Set ...                                   // Additional settings, e.g. start
address after reset

Data.Assemble [EAHB | EAXI]:<start_addr> b $-0x0  // Assemble endless loop

Data.Set [EAHB | EAXI]:<register_n> % <value>  // Release core(s) from reset
Data.Set ...

CORE.ASSIGN 1. 2. 3. ...                       // Assign as many cores as released
SYStem.Mode Attach                             // Attach to running cores
Break.direct
```

Moreover, the following template script allows catching the CPU at the reset vector (Armv8 only):

```
// This example outlines a rough scheme how a CPU could be reset via a custom
// register sequence and caught at the reset vector with a reset catch mechanism.
// This is useful if the core starts from a non-writable location (e.g. ROM),
// where an endless loop cannot be assembled.

SYStem.CPU <cpu>
CORE.ASSIGN 1. 2. 3. ...                       // Assign as many cores as released

SYStem.Mode Prepare

Data.Set [EAHB | EAXI]:<register1> % <value>  // Bring core(s) into reset
Data.Set ...                                   // Additional settings, e.g. start
address after reset

// Enable reset catch (example for Armv8; '/CORE' not needed for single-core setups)
Data.Set e:(COREBASE()+0x24) %Long 0x2 /CORE 1
Data.Set e:(COREBASE()+0x24) %Long 0x2 /CORE 2
...
Data.Set e:(COREBASE()+0x24) %Long 0x2 /CORE n

Data.Set [EAHB | EAXI]:<register_n> % <value> // Release core(s) from reset
Data.Set ...

SYStem.Mode Attach                             // Attach to stopped cores
```

Comments (2)

Comments (2)

YA **Yasmin Amer**
1 year ago
I have a question related to the reset topic, I am trying to reset through the sw (the sw directly access the reset register to perform the reset) but then the reset has happened and the sw rerun again without clicking start from the debugger (it is not the expected behavior), is this an issue related to a missing debugger configuration?
Oussema Koubaa

1 year ago

Hello, To ensure a proper support, please open a new ticket: "https://support.lauterbach.com/new-ticket" provide more details about the issue (target, scripts used, etc.), include relevant screenshots, and generate a system information report about your TRACE32 configuration by selecting the TRACE32 menu 'Help' > 'Support' > 'System Information...', click 'Save to File' and send the resulting text file as an attachment to your e-mail.