Programming Pequeno

- with peqFlash

September 2024

Table of Contents

1. Programming Interface

Pequeno subsystem consists of Instruction and Data memories (IMEM and DMEM) which can be programmed by binary code generated by pqr5asm assembler. UART is the programming interface used by Pequeno subsystem to upload the program from an external host. The host can program the Pequeno core via UART at a pre-defined baud rate and boot the CPU. The baud rate should be the same as the Loader baud rate (macro: `BAUDRATE`) configured during the core generation.

2. peqFlash

peqFlash is the SW flasher tool used to program binary to the Pequeno core from a Host system via UART.

- ✓ Compatible with target device: Pequeno v1.0 subsystem with Loader.
- ✓ Compatible with Loader's UART specifications: 8-bit data, 1 start/stop bit, no parity.
- ✓ Compatible with the binary file format of pqr5asm.
- ✓ Supports on-the-fly programming and reboot.
- ✓ Supports relocation of program binary to different base address (text segment).

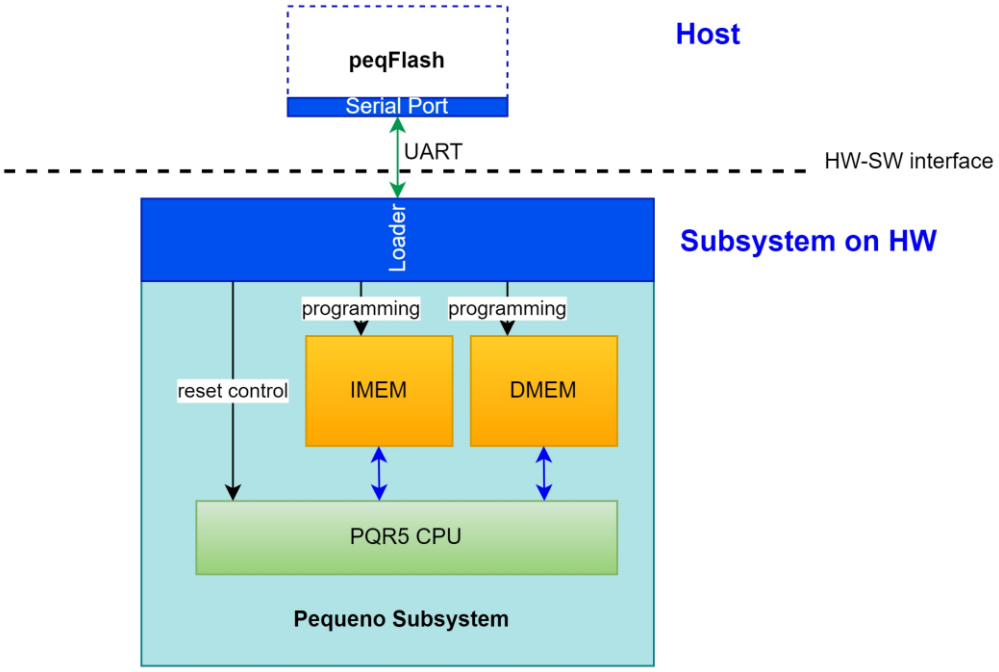

Figure 2.1: peqFlash in action

3. Command line arguments

Following arguments are supported while invoking peqflash.py from command line.

Flag	Argument	Description & usage
`-serport`	Serial COM/TTY port	Serial port to which the target is connected. Eg: `-serport COM3`
`-baud`	Baud rate	Baud rate of the serial communication in bps. Eg: `-baud 115200`
`-imembin`	Instruction binary file	Instruction binary file path relative to the directory from which the script is invoked. Eg: `-imembin "./sample_imem.bin"`
`-dmembin`	Data binary file	Data binary file path relative to the directory from which the script is invoked. Eg: `-dmembin "./sample_dmem.bin"`
`-reloc`	Relocation address	The base address to which the program binary should be relocated to (text segment). The address should be 4-byte aligned. The input binary code is assumed to be relocatable binary. The PC on resetting the core is assumed to be the same as this address. Eg: `-reloc 0x00000004` `-reloc 4`
`-cleanimem`		Flag to clean instruction memory with NOP instructions before writing the binary/reboot.
`-rebootonly`		Flag to override binary files and reboot with existing binary.

Table 3.1: Command line arguments

4. Programming flow

Once the tool is invoked successfully with necessary command line arguments, following steps are performed to program the binary to the core.

1. Verify device signature.
2. Validate the input binary files (skipped if `-rebootonly`).
3. Perform IMEM clean (only if `-cleanimem`).
4. Flash instruction and data memory with binary (skipped if `-rebootonly`).
5. The CPU boots if there are no errors reported by peqFlash or Loader on board. Otherwise, the CPU is held in reset for safe state.

5. Troubleshooting

peqFlash is a robust tool which handles and reports errors, if any, during the programming process. This section provides guidance on diagnosing and resolving common errors encountered while using the peqFlash tool to upload the program. It covers typical error messages and scenarios the user may face during the programming process.

Error	Cause	Steps to resolve
`Incomplete/No response/communication link timeout`	peqFlash didn't receive the SUCCESS/FAILURE response from the core to the commands sent by the flasher. This could lead to timeout.	This could be a random unexpected error. Check all the connections, and retry uploading the binary. If this error happened during the flashing step, it is recommended to reset the system and run the tool with `-cleanimem`.
`Error opening serial port/serial port connection lost`	Serial port is inaccessible.	Check the permissions by the OS on the COM port, check all the connections, and retry uploading the binary. If this error happened during the flashing step, it is recommended to reset the system and run the tool with `-cleanimem`.
`Device signature verification failure`	Invalid target	Ensure the target is Pequeno v1.0 subsystem with Loader.
`Error opening binary file`	Invalid binary file	The binary file may be non-existing or invalid file type. Check the binary file path, and verify the file type is .bin.
`Binary file validation failure`	Invalid/corrupted binary file	Re-generate the binary files and retry uploading the binary.
`IMEM clean request failed/Reboot failed`	Random	This could be a random unexpected error. Check all the connections, and retry uploading the binary with `cleanimem`.
`Flashing failed with ERROR code = <>`	0xEC = Invalid command 0xED = Programming error 0xEE = Post-amble error	These are the class of errors that occur during flashing the binary to the core. 1. Invalid command: This could be a random unexpected error. Check all the connections, and retry uploading the binary. It is recommended to reset the system and run the tool with `-cleanimem`. 2. Programming error: This error happens when the tool fails to write the complete binary code to the instruction/data memory due to memory overflow. Ensure that the memory has enough size to accommodate the full binary starting from the base address of text/data segments.

		3. Post-amble error: This error happens when Loader doesn't receive post-amble. This could be a random unexpected error during serial communication. Verify the binary files and retry uploading the binary.
`Timeout/UART error (reported by Loader on board)`	This is a FATAL ERROR due to Loader internal timeout or UART frame error.	Reset the system and run the tool with `-cleanimem`. Check all the connections, and retry uploading the binary.
`Invalid command error`	Invalid command received by Loader	This could be a random unexpected error. Check all the connections, and retry uploading the binary. If this error happened during the flashing step, it is recommended to reset the system and run the tool with `-cleanimem`.

Table 5.1: Troubleshooting errors

Tool dependencies:

peqFlash is python based tool. The host system should support at least python 3.9 to run this tool.

Revision History

The following table shows the revision history of this document.

Date	Version	Revision
Sept-2024	1.0	• Initial version