# MPC5674F Flash Programmer using Tigard Board

**Author:** mobilemutex

**Date:** 2026-01-15

## Overview

This Python-based flash programmer allows you to erase and program the internal flash memory of the NXP MPC5674F microcontroller using a Tigard board (FT2232H-based) for JTAG communication.

The tool implements the following protocols:

| Protocol | Description |
|---|---|
| JTAG | IEEE 1149.1 Test Access Port for physical communication |
| OnCE | On-Chip Emulation for debug control of the e200z7 core |
| Nexus | IEEE-ISTO 5001 for memory read/write access |

## Hardware Requirements

- **Tigard Board** (or any FT2232H-based JTAG adapter)
- **MPC5674F Target Board** with accessible JTAG header
- **USB Cable** for connecting Tigard to your computer

### JTAG Pin Connections

Connect the Tigard JTAG pins to your MPC5674F target as follows:

| Tigard Pin | MPC5674F Pin | Description |
|---|---|---|
| TCK | TCK | Test Clock |
| TDI | TDI | Test Data In |
| TDO | TDO | Test Data Out |

| | | |
|---|---|---|
| TMS | TMS | Test Mode Select |
| GND | GND | Ground |
| VREF | VDD | Target voltage reference (3.3V) |

**Note:** The MPC5674F uses 3.3V logic levels. Ensure your Tigard is configured for 3.3V operation.

## Software Requirements

- Python 3.8 or later
- pyftdi library

### Installation

```bash
pip install pyftdi
```

On Linux, you may need to configure udev rules for FTDI devices:

```bash
# Create udev rule for FTDI devices
sudo tee /etc/udev/rules.d/99-ftdi.rules << EOF
SUBSYSTEM=="usb", ATTR{idVendor}=="0403", ATTR{idProduct}=="6010", MODE="0666"
EOF

# Reload udev rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

## Usage

### Basic Commands

```bash
# Erase flash and program a binary file
python mpc5674f_flasher.py firmware.bin --erase
```

```
# Program without erasing (for incremental updates)
python mpc5674f_flasher.py firmware.bin

# Program and verify
python mpc5674f_flasher.py firmware.bin --erase --verify

# Read a single register
python mpc5674f_flasher.py --read 0xC3F88000

# Dump memory region
python mpc5674f_flasher.py --dump 0x00000000 256
```

## Command Line Options

| Option | Description |
|---|---|
| `file` | Binary file to program to flash |
| `--erase` | Erase flash before programming |
| `--verify` | Verify flash contents after programming |
| `--read ADDR` | Read a single 32-bit register (hex address) |
| `--dump ADDR SIZE` | Dump memory region (hex address, size in bytes) |
| `--url URL` | FTDI device URL (default: `ftdi://0x0403:0x6010/1`) |

## FTDI Device URL Format

The default URL `ftdi://0x0403:0x6010/1` specifies:

- Vendor ID: 0x0403 (FTDI)
- Product ID: 0x6010 (FT2232H)
- Interface: 1 (first MPSSE channel)

If you have multiple FTDI devices, you can specify the serial number:

Bash

```
python mpc5674f_flasher.py firmware.bin --url
"ftdi://0x0403:0x6010:FT123456/1"
```

## MPC5674F Memory Map

The tool is configured for the following memory regions:

| Region | Address Range | Size | Description |
|---|---|---|---|
| Flash Array | 0x0000_0000 - 0x003F_FFFF | 4 MB | Internal flash memory |
| Internal SRAM | 0x4000_0000 - 0x4003_FFFF | 256 KB | Internal SRAM |
| Flash_A Registers | 0xC3F8_8000 - 0xC3F8_BFFF | 16 KB | Flash controller A |
| Flash_B Registers | 0xC3F8_C000 - 0xC3F8_FFFF | 16 KB | Flash controller B |

## Flash Programming Sequence

The tool follows the standard MPC5674F flash programming sequence:

1. **Enter Debug Mode** - Halt the CPU using OnCE protocol
2. **Unlock Flash Blocks** - Write password to LMLR/HLR registers (if locked)
3. **Select Blocks** - Configure LSR/MSR/HSR for target blocks
4. **Erase** (if requested):
    - Set ERS bit in MCR
    - Write interlock to flash array
    - Set EHV bit to start erase
    - Poll DONE bit for completion
    - Check PEG bit for success
5. **Program**:
    - Set PGM bit in MCR
    - Write data to target address (interlock)
    - Set EHV bit to start programming
    - Poll DONE bit for completion
    - Check PEG bit for success

6. **Verify** (if requested) - Read back and compare

## Limitations and Known Issues

1. **Flash Block Unlocking**: The tool assumes flash blocks are unlocked. If your device has locked blocks, you may need to modify the code to write the correct password to LMLR/HLR registers.
2. **Programming Speed**: The current implementation programs one double-word (8 bytes) at a time, which is slow for large files. Future versions could implement burst programming.
3. **Flash_B Support**: Currently only Flash_A is fully supported. Flash_B uses the same protocol but at a different base address (0xC3F8_C000).
4. **Error Recovery**: If programming fails mid-way, the flash may be in an inconsistent state. Always erase before retrying.

## Troubleshooting

### "Failed to enter debug mode"

- Check JTAG connections
- Verify target is powered
- Ensure VREF is connected to target VDD
- Try reducing TCK frequency (edit `set_frequency()` call)

### "Failed to read MCR"

- The CPU may not be halted properly
- Check that OnCE TAP is enabled
- Verify the target is an MPC5674F (not a different variant)

### "Erase/Program failed (PEG not set)"

- Flash blocks may be locked
- Target voltage may be insufficient for flash operations
- Flash may be damaged

### USB Permission Errors (Linux)

- Ensure udev rules are configured (see Installation)

- Try running with `sudo` (not recommended for regular use)

## References

1. **AN4365**: Qorivva MPC56xx Flash Programming Through Nexus/JTAG
2. **MPC5674F Reference Manual** (Rev. 7)
3. **e200z7 Core Reference Manual**
4. **IEEE 1149.1**: JTAG Standard
5. **IEEE-ISTO 5001**: Nexus Standard

## License

This tool is provided as-is for educational and development purposes. Use at your own risk.