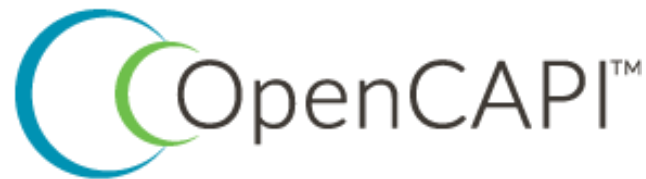


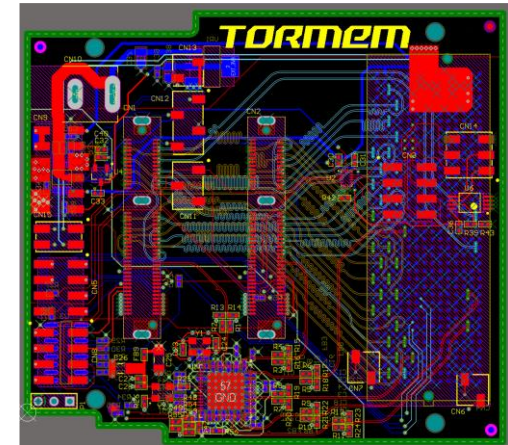
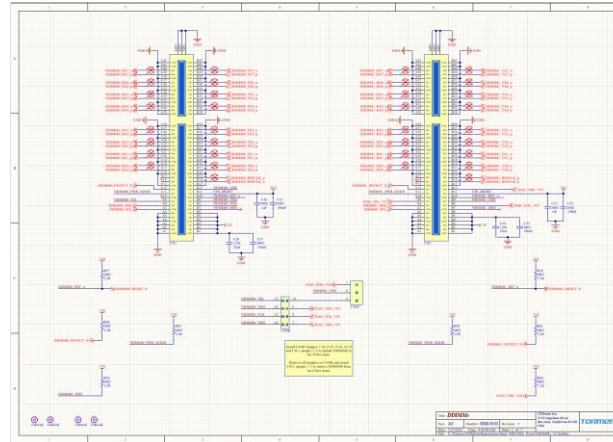
OMI Development Kit

TORmem has been collaborating with other OpenCAPI members on a development platform to address its needs for implementing OMI interfaces in its products. As this work progressed, it became clear that the tools being created and the experience gained would be beneficial to anyone interested in OMI. Given TORmem's interest in supporting community-driven development, this work is being contributed to OpenCAPI and will be available to help accelerate OMI evaluation for future interested parties.

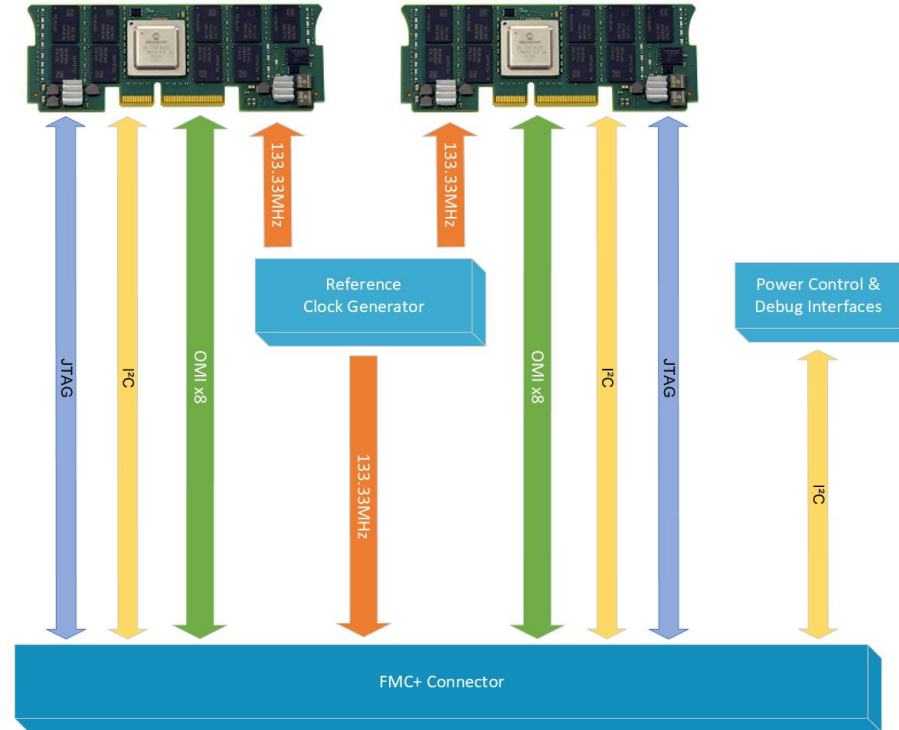


Open Source Design

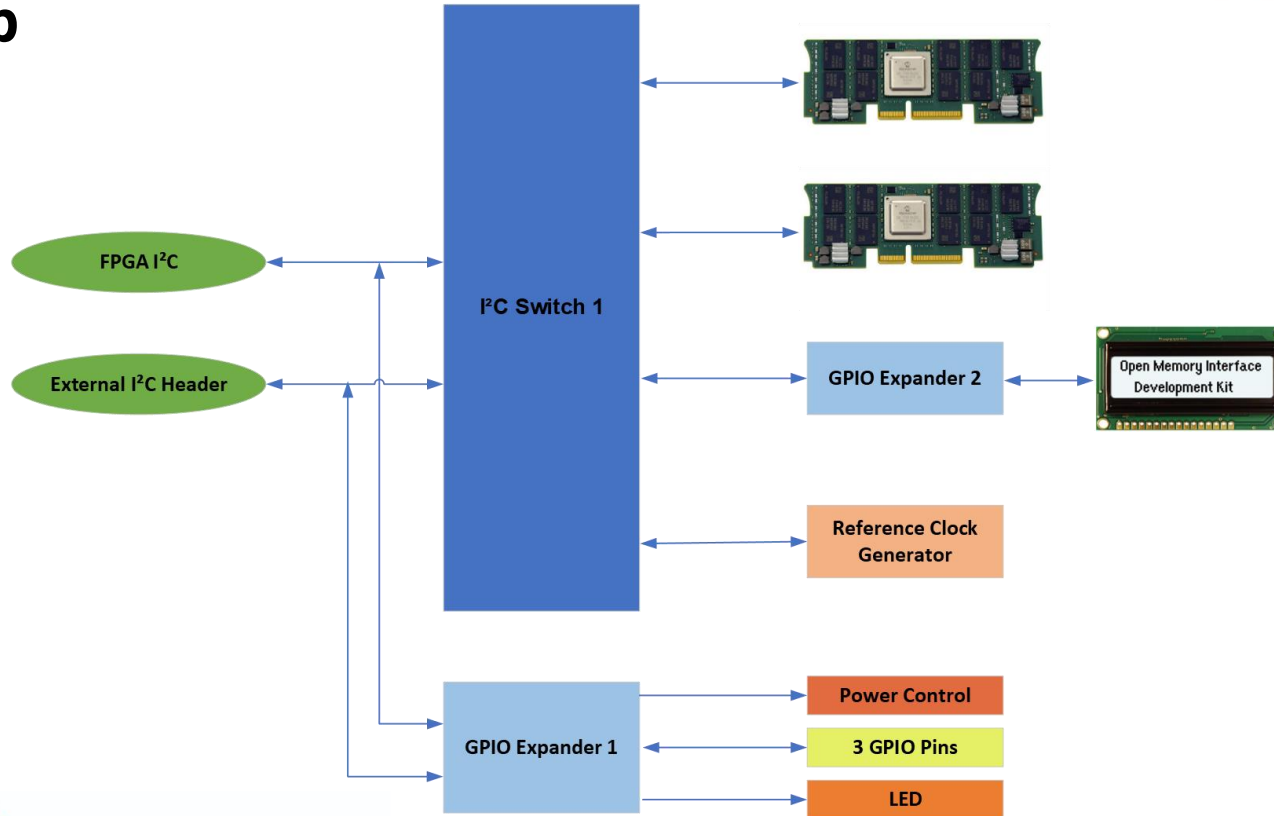
The OMI Adapter design is in Altium Designer format and source files will be available. The bill of materials was created with component availability in mind and vendor part numbers are provided. The PCB and all components include corresponding 3D models.



System Block Diagram



I²C Map



Configurable Clock Source

A Microchip Technology ZL30265 Clock Generator provides the DDIMMs and FPGA with an ultra-low jitter 133.333MHz reference clock. Four pre-programmed profiles are selectable by an onboard DIP switch. Additional profiles can be flashed through a dedicated three pin 3.3v level I²C header by the ZL30265 configuration utility.

DDH: 3.3 V VDL: 3.3 V

Working Offline

XA

☐ Doubler ☐ Invert

Mode: 300 uW Crystal

XA Frequency (MHz): 50.000000

Frequency After Divider: 50.000000 MHz

IC1

☐ Enable ☐ Invert

IC1 Frequency (MHz): 133.320000

Frequency After Divider: 133.320000 MHz

IC2

☐ Enable ☐ Invert

IC2 Frequency (MHz): 50.000000

Frequency After Divider: 50.000000 MHz

IC3

☒ Enable ☐ Invert

IC3 Frequency (MHz): 50.000000

Frequency After Divider: 50.000000 MHz

APLL1

☐ Enable APLL ☐ Enable Frac Divider

☐ Doubler ☐ Align Output Dividers

Status: Locked

Selected Source: XA

Feedback Signal: FB Divider

Source Selection: Reg Controlled

Source: XA

VCO Frequency (MHz): 400.000000

Frequency After Int Divider (MHz): 400.000000

Frequency After Frac Divider (MHz): Undefined

☐ Enable Bypass Path 1

APLL2

☒ Enable APLL ☐ Enable Frac Divider

☐ Doubler ☐ Align Output Dividers

Status: Locked

Selected Source: IC3

Feedback Signal: FB Divider

Source Selection: Reg Controlled

Source: IC3

VCO Frequency (MHz): 400.000000

Frequency After Int Divider (MHz): 400.000000

Frequency After Frac Divider (MHz): Undefined

☐ Enable Bypass Path 2

Bank A

Source: APLL2 Int Divider

Format: HCSSL

Divisor: 3

Output Frequency: 133.333333 MHz

Bank B

Source: APLL2 Int Divider

Format: HCSSL

Divisor: 3

Output Frequency: 133.333333 MHz

Bank C

Source: APLL2 Int Divider

Format: HCSSL

Divisor: 3

Output Frequency: 133.333333 MHz

Bank D

Source: APLL2 Int Divider

Format: HCSSL

Divisor: 3

Output Frequency: 133.333333 MHz

Bank E

Source: APLL2 Int Divider

Format: HCSSL

Divisor: 3

Output Frequency: 133.333333 MHz

Bank F

Source: APLL2 Int Divider

Format: HCSSL

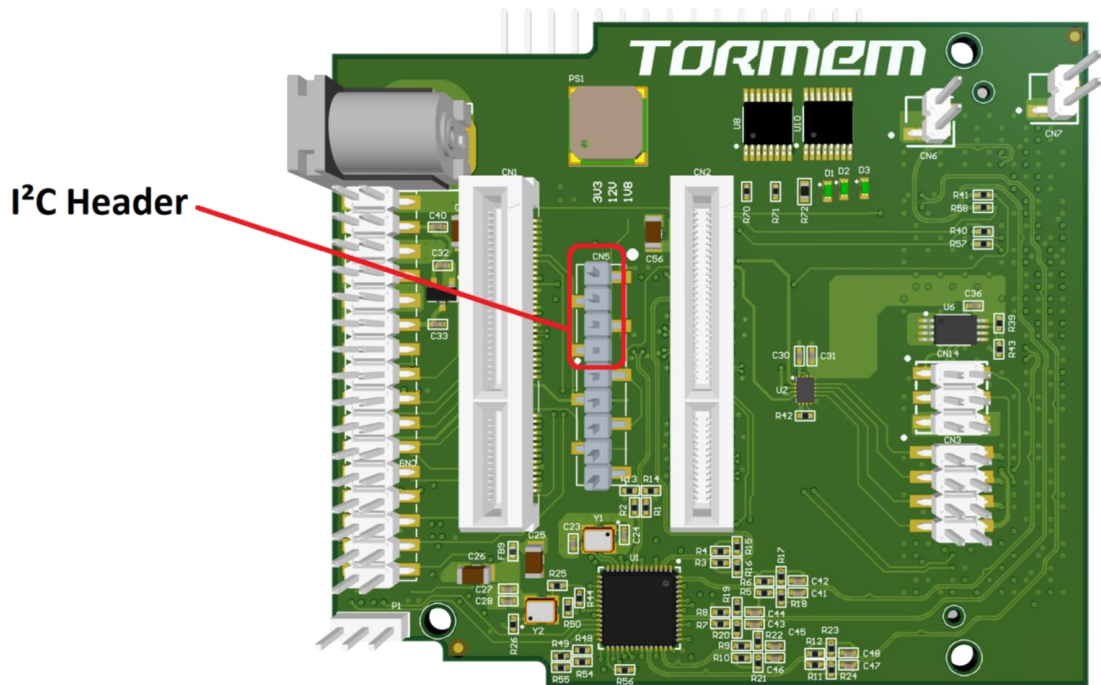
Divisor: 3

Output Frequency: 133.333333 MHz

Microsemi

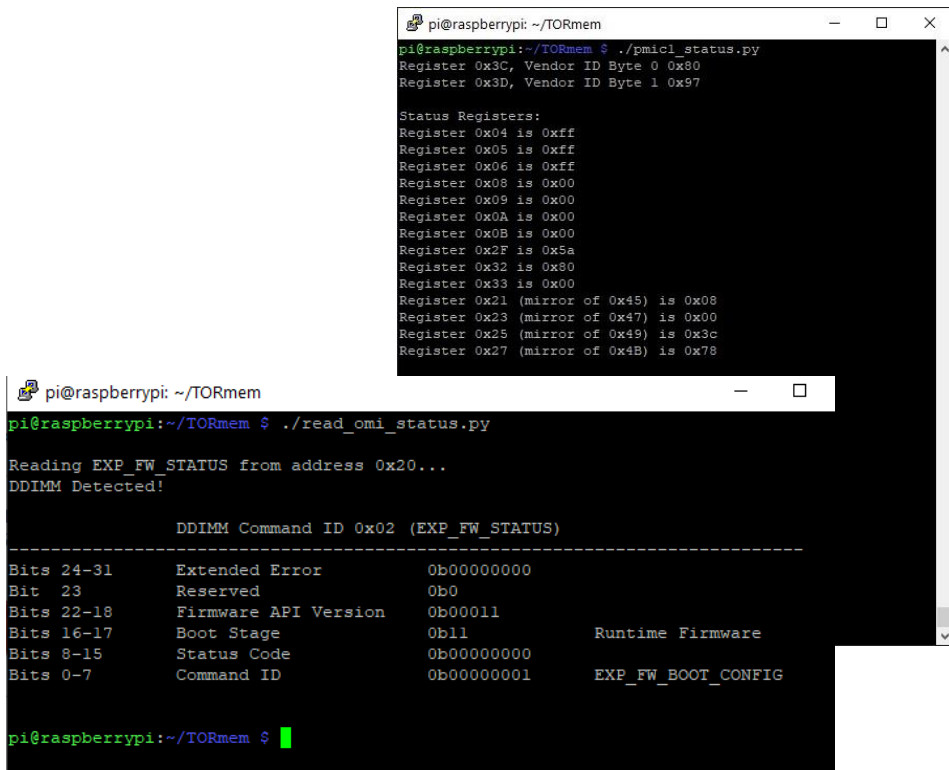
Control Planes

DDIMMs require a sequence of I²C configuration commands sent to the modules before they can be used as memory. These commands can be supplied by an external device, such as a Raspberry Pi, or by the FPGA. External I²C access is provided by a 3.3v level I²C header. Once the OMI interface is active, further commands can be provided in-band over OMI from the FPGA. Alternatively, the DDIMMs can be configured for out-of-band access over I²C by the FPGA or via the header by an external device.



Python Configuration Scripts

Control plane script examples are provided for common DDIMM configurations. These scripts are written in Python and can be executed on a Raspberry Pi, on an FPGA-implemented Microblaze soft-core or any other device that supports Python.



```
pi@raspberrypi: ~/TORMem
pi@raspberrypi:~/TORMem $ ./pmic1_status.py
Register 0x3C, Vendor ID Byte 0 0x80
Register 0x3D, Vendor ID Byte 1 0x97

Status Registers:
Register 0x04 is 0xff
Register 0x05 is 0xff
Register 0x06 is 0xff
Register 0x08 is 0x00
Register 0x09 is 0x00
Register 0x0A is 0x00
Register 0x0B is 0x00
Register 0x2F is 0x5a
Register 0x32 is 0x80
Register 0x33 is 0x00
Register 0x21 (mirror of 0x45) is 0x08
Register 0x23 (mirror of 0x47) is 0x00
Register 0x25 (mirror of 0x49) is 0x3c
Register 0x27 (mirror of 0x4B) is 0x78

pi@raspberrypi: ~/TORMem
pi@raspberrypi:~/TORMem $ ./read_omi_status.py

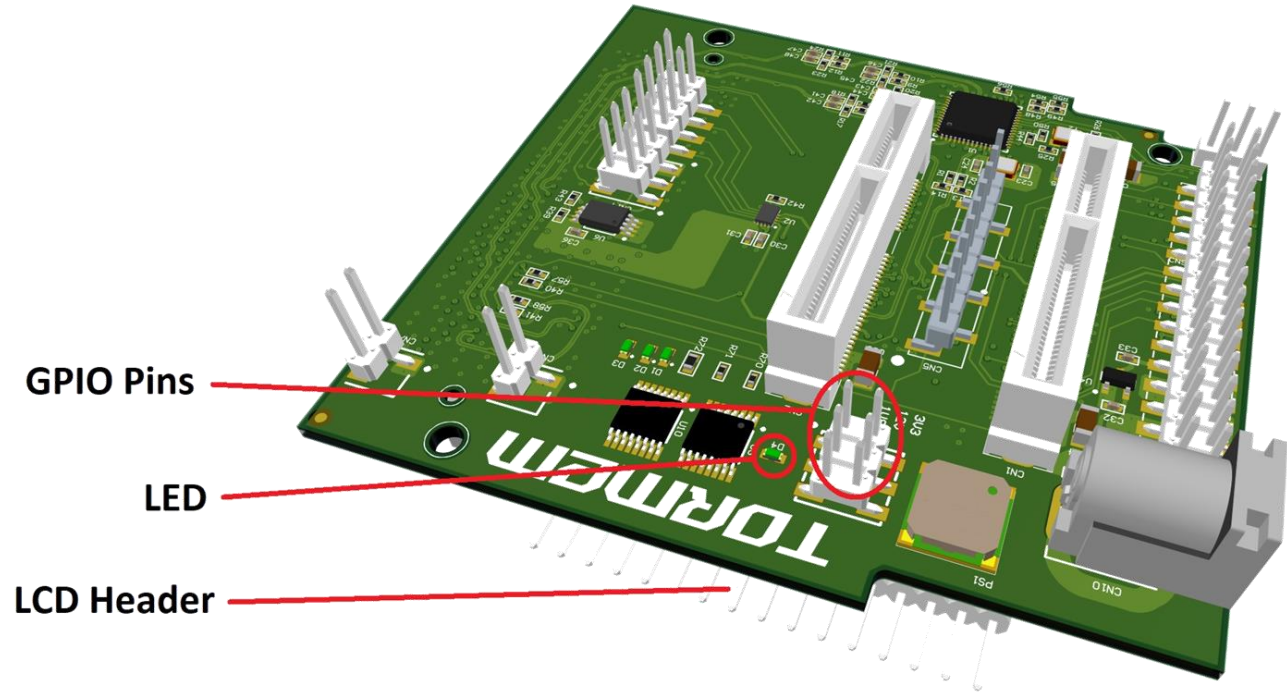
Reading EXP_FW_STATUS from address 0x20...
DDIMM Detected!

DDIMM Command ID 0x02 (EXP_FW_STATUS)
-----
Bits 24-31    Extended Error    0b00000000
Bit 23       Reserved          0b0
Bits 22-18   Firmware API Version 0b00011
Bits 16-17   Boot Stage        0b11          Runtime Firmware
Bits 8-15    Status Code       0b00000000
Bits 0-7     Command ID        0b00000001    EXP_FW_BOOT_CONFIG

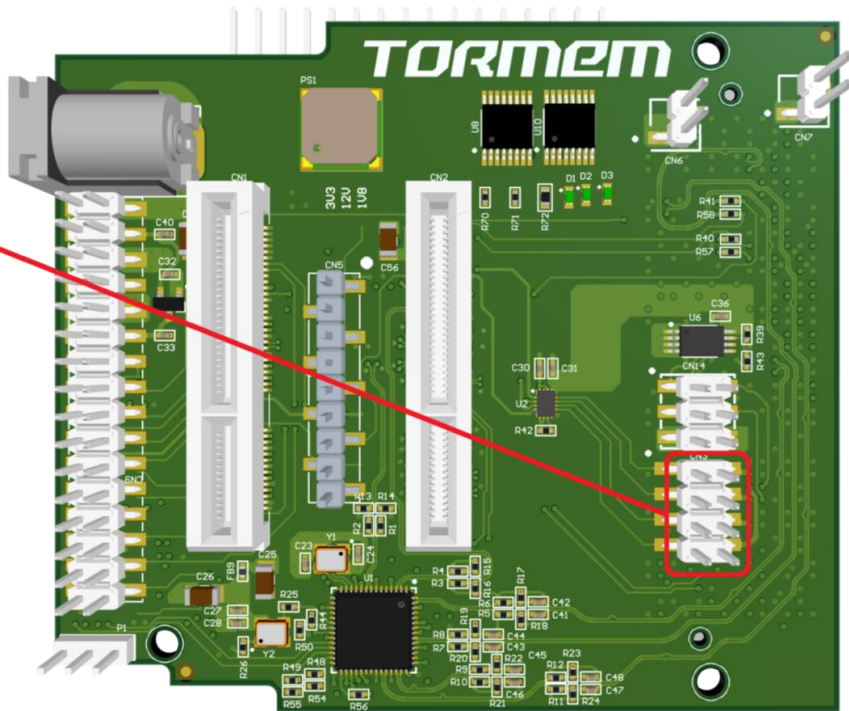
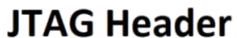
pi@raspberrypi:~/TORMem $
```

Debug Capabilities

The OMI Adapter includes an I²C interface to an LCD, a LED and three pins of GPIO for debug output and status indication. Individual power rail control is also provided.



The OMI Adapter Development Board supports JTAG access to the PM8596 OMI Memory Controller either directly attached to the onboard header or jumpered into the JTAG chain of the VCU128.



Images

