

FAQ - FPMC-TAXI175

This product uses the AMD TAXI chipset (AM7968 and AM7969) which was discontinued at the end of 1999. Will this module still be available?

We invested in inventory of the AM7968 and AM7969, so this module will be available only until that inventory is used up.

Have you done performance benchmarks for throughput from either HotLinks or Taxi cards to RAID?

We have done performance benchmarks in VxWorks / PowerPC environments to JBOD and RAID storage on Motorola and Force VMEbus carriers and easily sustained 33 MB/s. We have also done benchmarks on VME/RACEway systems using our own PowerRACE carrier cards and sustained 66 MB/s from dual HOTLinks into a JBOD/RAID FibreChannel interface.

How much power does an FPMC-TAXI card typically draw?

The typical power consumption for the TAXI-175-C Copper board is 3.45 W. The typical consumption for the TAXI-175-F Fiber board is 5.07 W. The typical consumption for the TAXI-125-C Copper board is 3.42 W. The typical consumption of the TAXI-125-F Fiber board is 5.07 W.

The power consumption analysis was based on typical power consumption figures for each of the components on the boards. In cases where the typical power consumption for a certain part was not given by the data sheets, an estimate of half of the maximum power drawn was used.

Can I use the FPMC-TAXI175 with '10B' encoding to understand my (10 data bits+ 4 data bits/4 Tag bits/2 unused bits) data?

The TAXIchip supports both 4B/5B and 5B/6B encoding. In 8 bit mode, it takes 8 data bits + 4 control bits and generates a pair of 5B codes (i.e. two 4B-to-5B conversions). Essentially, there are 256 possible data values and 15 control values encoded into a 10 bit token. This isn't actually the same encoding as 8B/10B, as it is two separate 4B/5B encodings.

In 9 bit mode, the TAXIchip uses one 4B/5B encoding and one 5B/6B encoding. This encodes 512 possible data values and 7 possible control values into an 11 bit token.

In 10 bit mode, the TAXIchip uses two 5B/6B encodings. This encodes 1024 possible data values and 3 possible control values into a 12 bit token.

Our hardware supports 8 and 9 bit modes, as we have a 9 bit data path between our FPGA logic and the TAXI chip (we use a 9 bit FIFO). In 8 bit mode, we can support 256 data values and 15 control values. In 9 bit mode, since we have no control-vs-data bit, we only support 512 data values, no control values. We do not support 10 bit data mode, since we don't have a 10 bit data path.

How can I figure out the "actual" frequency being programmed by the clock synthesizer on a FastLink Product?

Both modules use a PLL-based clock synthesizer to generate arbitrary output frequencies under software control. The PLL synthesizer can generally match the desired frequency within 1000 ppm, depending on the relationship between the reference frequency and the desired output.

If you have installed TEK's software drivers for the module, you can simply attempt to configure the module for the desired frequency and the driver will respond with the "actual" frequency being programmed.

If you have not yet installed TEK's software drivers for the module, you may use the Cypress BitCalc software (available at <http://www.cypress.com/design/progprods/clock/clocks.html>) to check for a specific frequency.

The "actual" frequency reported by either TEK's software or Cypress's BitCalc includes all effects due to the programmable options in the PLL but assumes that the reference frequency is precisely accurate. Errors in the reference frequency will create proportional errors in the output frequency.

For most of TEK's products, the reference frequency is an oscillator frequency (33.3333 MHz) frequency divided by two. The PCI bus clock can also be selected as a reference frequency. Note that some PCI/PMC platforms use 33.000 MHz and others use 33.333 MHz; the TEK drivers assume that the PCI bus frequency is 33.333 MHz.

For example, if the desired output frequency is 6.35 MHz, TEK's software will generate a PLL program control word which will have an output of 6.349206 MHz (-125 ppm), assuming a local bus clock of 33.333333 MHz and a reference clock of 16.666667 MHz.

How can I convert a .TTF file to a .H header file?

We provide a utility to perform this function.

If an FPMC module has an Altera FPGA, how is the programming accomplished (is special hardware/cabling required to download code)?

For all of our FPGA-based FPMC modules (and IP modules for that matter), the FPGA is downloaded using the Altera "passive serial" download capability. The user software has access to a control register which drives nCONFIG, DATA0 and DCLK to the FPGA and monitors the nSTATUS and CONF_DONE. This allows FPGA download to be implemented with no special hardware or cables.

Can the FPGA code be "boot" resident or does it need to be downloaded to the FPGA before every use?

Our PMC and IP modules require an FPGA download for every power-up or reset.

Our software uses the Altera TTF output and generates a C language include file which can then be built into the application program. To minimize program space, we typically use a simple run-length compression algorithm on the bitstream and expand it when downloading to the FPGA.