

FAQ - FPMC-LVI and FPMC-LVO

How does the data get transferred over the wire?

The LVI modules use National Semiconductor LVDS interfaces which serialize the parallel data. Specifically, the LVI-2x28 uses the National DS90CR286 which converts a 28 bit parallel TTL interface to four LVDS signals and a clock. Each LVDS signal provides 7 bits of serial data for each clock, resulting in 4 x 7 bits of data total for each clock edge. The clock rate can be up to 66 MHz at the link level; the total throughput is limited by the 32-bit PCI bus to approx 120 MB/s.

The LVI-3x36 uses the National DS90CR484 which converts a 48 bit parallel TTL interface into eight LVDS signals and a clock. This driver still uses a 7x bit rate, but each LVDS signal consists of 6 data bits and a running disparity bit. This results in 48 bits total (8 signals x 6 bits/signal). Our LVI module only implements 36 bits of the 48 possible bits, so we only use six of the possible eight signals for data, plus the clock. The 'CR484 supports clocks up to 112 MHz; again, the total throughput is limited by the 32-bit PCI bus to approx 120 MB/s and our module isn't specified beyond 66 MHz.

Are the National DS90CR285/286 Channel Link devices interoperable with the National DS90CR211/212 devices?

The LVI-2x28 and LVO-2x28 use the National DS90CR285/286 Channel Link devices, which implement 28 data bits using four 7x serial streams. National also makes a device called the DS90CR211/212 which implements a very similar 21 data bit interface using three 7x serial streams. These interfaces are nominally interoperable, with some constraints.

The DS90CR211/212 has a clock frequency range from 20 to 40 MHz. The DS90CR285/286 has a clock frequency range from 20 to 66 MHz. To be interoperable, the clock must be between 20 and 40 MHz.

The DS90CR285/286 multiplexes the 28 bits into the four streams in a somewhat haphazard mapping. Bits 7,6,4,3,2,1,0 are in serial channel 0, bits 18,15,14,13,12,9,8 are in serial channel 1, bits 26,25,24,22,21,20,19 are in serial channel 2, and bits 23,17,16,11,10,5,27 are in serial channel 3. The DS90CR211/212, on the other hand, has a simpler mapping: bits 6-0 are in serial channel 0, bits 13-7 are in serial channel 1, and bits 20-14 are in serial channel 2.

Therefore, if the three serial channels of a DS90CR211/212 are connected to three of the four channels of a DS90CR285/286, the bits at the DS90CR285/286 interface are scrambled. Since these bits are all processed by customizable FPGA logic on our LVI-2x28 and LVO-2x28, the bits can easily be reshuffled back to an appropriate 21 bit word. We do not currently have FPGA support for this.

The bottom line is that the LVO-2x28 can drive an interface to a DS90CR212, and the LVI-2x28 can receive an interface from a DS90CR211, with FPGA modifications to shuffle the bits. For the purposes of checking out an interface in non-real-time, a software "shuffling" could be used with the standard FPGA.

Do you plan any LVDS modules using the DS90CR211/212 interfaces?

At this time we have no plans to build LVDS modules using the DS90CR211/212 interfaces.

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.