The PowerQUICC II is very flexible and can be configured in several different ways. Some of the configuration options must be available as soon as the processor comes out of power-up and so a special power-up configuration routine is necessary.

The PowerQUICC II is very flexible and has the ability to be configured in many different ways, and some of that configuration must be known at power up. This module will describe how Initial configuration is defined, the core configuration registers then how the Reset and Configuration registers are handled.

This module provides a detailed look at the different ways the PowerQUICC II can be configured for power up/power down sequences and concludes with a brief exercise.

There are 11 pages in this module, along with 11 reference pages, which will take approximately 25 minutes.
This chart shows the various causes of reset and the effect of each. There are basically three different levels of reset, the highest being power-on which resets all logic and software. When the processor comes to life it’s like many of us when we wake up – totally confused!! It must be brought to a state where the hardware and software can operate from a known state. Fortunately, it happens faster than for most of us!

Once up and running, the internal hardware is organized so it no longer needs to be reconfigured but there are two system considerations. One is that a major system fault occurs, in which case all the firmware must be reconfigured. Alternatively, a minor error occurs or the requirement is simply to restart the application, in which case none of the internal configuration should be modified. If all you want to do is restart the application, the general system requirements such as connections, memory organization, clock rates will not change, so the configuration should not change. The memory controller, system protection logic, interrupt controller and parallel IO pins are unaffected by Soft Reset, and System configuration is maintained.

Both Hard Reset and Soft Reset can be asserted externally or internally, they are bi-directional. Hard Reset and Soft Reset are Open-drain and both should have an external pull-up resistor. Neither inputs can be recognized while being driven internally.
• Power-on-Reset must be asserted for at least 16 clocks after 2/3 Vcc to ensure internal logic latches. Then RSTCONF is sampled for master configuration.

• Internally PORESET holds for another 1024 clocks during which time the reset configuration sequence takes place.

• As internal PORESET is negated, MODCK pins are sampled for clock setup.

• After the PLLs lock, the HRESET is held for an additional 512 clocks and SRESET 515 clocks.

• Once the Resets are negated the processor is running, basic configuration is complete, and the core enters reset exception processing.

Reference Manual section 5.1.2/3/4

This timing diagram shows the relationship between the three reset pins and internal functions related to them. Initially, when power is applied to the chip, the power-on reset pin should be asserted, held low. Obviously, the hardware can do nothing until the voltage levels reach a usable level, in this case, two thirds of the operating level. The power-on reset internally asserts the hard reset signal, which in turn asserts the soft reset signal, so at this time all three signals are asserted.

After at least 16 clock pulses, internal logic takes over from the external input and holds the power-on reset level asserted for a further 1,024 clocks and then negates the signal. At that time, the mod-clock pins are sampled by the controller to configure the basic internal clock rates to be used. Notice that the clock configuration is controlled by hardware by providing the required input levels at reset.

Once the clock configuration is known, the internal phase locked loop must stabilize and at that time an internal signal indicates lock. Once the PLL is locked, the hard reset continues to be asserted for a further 512 clocks and is then negated. Soft reset negates three clocks later. Once reset is negated the internal core jumps to it’s reset vector and begins the startup routine.
This diagram indicates the various clock options available. Initially, at power on reset the initial clock configuration is defined by the mod clock pins. The reset configuration can also be modified by the reset configuration value discussed later.

The base clock rate is defined by the input clock that must be present. This drives a phase locked loop, which has a multiplier on board controlled by the reset configuration. Each of the major modules has its own divider stage so that the core, bus, CPM, serial controllers and baud rate generators each have their own independent control, with skew elimination. With the exception of the baud rate generators, all the other modules have two clock sources, one 90 degree out of phase from the other providing better control options.

The system clock mode register contains the divider definitions for the clock and it is set up by the reset configuration process.
When a system is powered, it must be ready to start operating, at least at its basic level. The PowerQUICC II is very flexible and has the ability to be configured in many different ways, and some of that configuration must be known at power up. To enable this, a special power-on reset configuration operates whenever power is applied to the device. This diagram indicates a possible requirement for a system, with three MPC8260s connected, and showing the additional basic connections to get the system running. Some normal memory is required, as well as non volatile storage for the startup routine. Then, of course, there will be other peripherals associated with the application. At start-up, some initial considerations are necessary.

The first consideration is who is in control. When the system first powers up and no software is active it could be complete chaos if all the devices attempted to do their own thing. So, which device is going to control the initial organization?

Once the system is up and running with multiple devices, attempting to access the bus arbitration priorities must be defined.

For access to memory, a memory controller is required. All the 8260s have memory controllers. Are all devices using their own, which could be dangerous, and probably unnecessary? If not, which controller is to be used?

Is the system to be used in single or multi bus mode? In other words, are all accesses on the bus to be controlled by the memory controller or are there other devices in the system to control bus activity?

When the core processor enters boot up code, what bus width is required to access the boot memory?

The PowerQUICC II allows the internal dual-port RAM to be allocated to almost any location in the memory map. This is necessary to allow multiple PowerQUICCs to operate together. Where is the DPR to be located?

What bus width is required for external devices to access the dual-port RAM?

There are two possible regions in memory for the boot memory to be accessed. The processor must know where to find it. Is it in the high or low region?

Who is the bus master? In other words is bus arbitration controlled internally or externally?

What clock frequency is to be used?

Many of the pin functions of the PowerQUICC II are multiplexed and default to a specific function, but some functions must have access to the necessary pins to operate at start-up. What are those pin functions to be?

Most of the other decisions cannot be made until the controlling device has been decided, so that's the first decision made. The device that will be controlling the initial configuration process must have the reset configuration pin asserted, probably simply connected to ground. When power is applied, the controller will perform the configuration process, supplying all the necessary start-up configuration for the other PowerQUICC's.

The controlling device reads its own configuration information from memory and then reads the configuration information for each other device. It then writes it to the appropriate device by asserting that devices reset configuration pin while driving the data on the bus. A more complete description of this process is shown on the next page.
The PowerQUICC II device controlling the power-up configuration is shown at the top of the page with a representation of another two devices shown below and a portion of the memory map starting at address zero shown to the right. The assumption is that the top device has its reset configuration pin asserted, and all devices are connected to the data bus represented by the large gray arrows, and the address and data bus is connected to memory located at this portion of the map. The controlling device has address line zero connected to the reset configuration pin of the first configuration slave, and subsequent address line up to six connected to any other configuration slaves. The memory map is shown to be 64-bits wide because that is the maximum possible data bus width of the PowerQUICC II.

However, one question to consider is how wide is the bus connected to memory? Does the configuration controller know at power-on? To know that, the device would have to be pre-defined for start-up. But, one of the basic principles is that designers should have the maximum flexibility for their systems. Therefore the controller must start by assuming the lowest possibility.

The configuration controller must read the 32-bit configuration word from address zero, but reads it one byte at a time from the only locations known, without doubt, to be available, at eight byte intervals. This seems strange at first until you consider all the possibilities. The memory could be 8, 16, 32, or 64 bits wide. The controller does not know at this time. If it reads those locations, they are guaranteed to be present under all circumstances, and so it always reads the most significant byte lane for this process. The data read from memory is collected internally and distributed to the various internal registers to define the start-up configuration.

The controller reads the next configuration word from memory, collecting each byte internally. When it has all four bytes it drives the complete thirty-two bits onto the data bus and drives address line A, zero low with all others high, thereby asserting the reset configuration pin on the first configuration slave.

The device with the reset configuration pin asserted reads the data on the bus and distributes it internally for its configuration.

The controller then reads the next configuration word and writes it to the next configuration slave in the same way, this time with only A, one driven low.

The controller continues to repeat the process for five more slaves, making a total of eight possible devices configured in this way, including the master configuration controller. There is no way for the master to know how many devices require configuration, so it always performs this for seven slaves. All of this process takes place during the hard reset, and the configuration master has all the logic and controls to perform this function. The address to configuration memory and the reset configuration outputs on A, zero through six are driven as required. By this stage all the possible devices in the system have their respective configuration information to get started.

The process has now got the address to memory down to hex 100, which is the reset vector required for the core processor to access its start-up routine.

This simply shows the memory organization for the configuration data if the memory were 32 bits wide. Obviously, the configuration memory must be loaded with the necessary data before this process can be performed. The only alternative to using this process is if the configuration for all devices is to be the default, when the configuration word value is zero, and the system is to be used in single bus mode as opposed to 60x mode. Then, all reset configuration pins can be tied high and the processors will take the default configuration. To view the hard reset configuration options, click on the configuration word button.
There are three core configuration registers to select the options available for the core processor. These are core processor special purpose registers 1008, 1009 and 1011. They are designated Hardware Implementation Dependent registers 0-2.

HID0 selects machine check exception, parity, power modes, and cache controls.

HID1 selects phase-locked-loop configuration settings.

HID2 selects low power modes and cache way locking.
**Bus Arbitration** (Level Registers)

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPC-ALRH</td>
<td>60x bus Arbitration-level Register (High) - Defines arbitration priority for the bus master.</td>
</tr>
<tr>
<td>PPC-ALRL</td>
<td>60x bus Arbitration-level Register (Low) - Defines arbitration priority for the bus master.</td>
</tr>
<tr>
<td>LCL-ALRH</td>
<td>Local bus Arbitration-level Register (High) - Defines arbitration priority for the local bus master.</td>
</tr>
<tr>
<td>LCL-ALRL</td>
<td>Local bus Arbitration-level Register (Low) - Defines arbitration priority for the local bus master.</td>
</tr>
</tbody>
</table>

The values initialized in these registers will determine which system device will get access to the 60x and local bus at any given time.

There are numerous devices that can control the bus. Without some form of control, the system would not work. These registers provide the ability to select the arbitration priorities of the possibilities allowed, for both the 60x bus and the local bus. To view the registers, click on the one of interest.
Arbitration Configuration Registers

**PPC-ACR**

60x bus Arbitration Configuration Register
Defines arbitration modes and parked master on the 60x bus.

**LCL-ACR**

Local bus Arbitration Configuration Register
Defines arbitration modes and parked master on the local bus.

Two further registers are used for arbitration configuration, the 60x arbitration configuration register and the local bus arbitration configuration register. The 60x bus arbitration configuration register defines the arbitration modes and parked master on the 60x bus. The local bus arbitration configuration register defines arbitration modes and parked master on the local bus. To view these registers, click on the one of interest.
Reset and Configuration Registers

The SIU Module Configuration Register – Selects bus control features including pin configuration and core disable.

The Bus Configuration Register – Selects various attributes of bus control.

The Reset Mode Register – Enables reset due to a checkstop state.

The Reset Status Register – Indicates reset events.

Four further registers are used to handle system configuration and handle reset options. The system interface unit module configuration register selects bus control features including pin configuration, and core disable.

The Bus Configuration Register selects various attributes of bus control.

The Reset Mode Register enables reset due to a Check-stop state.

The Reset Status Register indicates reset events. To view these registers, click on the one of interest.
Completion

- That completes the Reset and Configuration section.
- It's important to remember that there are many other considerations when initializing, such as pin connections and controller functions.
- To test your understanding of this section select the Exercise button.

That completes the Reset and Configuration section. It’s important to remember that there are many other considerations when initializing, such as pin connections and controller functions. To test your understanding of this section select the Exercise button.