Using Design Directives in a Schematic Document

Altium Designer uses directives as instructions to certain parts of the software, in order to achieve the desired outcome. Design directives (to give them their full title) are objects that are placed solely within the confines of schematic sheets. A variety of such directives are on offer, the use of which can be associated to the following three areas:

- Directives associated with compilation of source schematic documents.
- Directives used to pass information defined on a schematic sheet through to the PCB.
- Directives used in the real-time analysis of nodes in an FPGA source schematic.

The following sections take a closer look at these areas and the directives on offer.

More information on a design directive, including placement and editing, is contained within the Schematic Editor, Panel and Object References.

Compilation-related directives

Rome wasn't built in a day. The same applies to the capture phase of a design project. It is highly likely that the design will be 'captured' in stages, with distinct schematic sheets used for Power circuitry, I/O interface, and so on.

A comprehensive check for electrical and drafting errors will undoubtedly be performed upon completion of the capture – once all circuitry is specified and wired. Prior to this final check it is fairly common to want to verify the electrical integrity of the design in stages, as it evolves, ensuring circuit elements are placed and wired correctly as you go.
Compilation of an individual schematic document (or the entire project) at intermittent stages in the capture process will often yield a number of error messages, resulting from circuitry that has yet to be captured, or interface wiring between circuit fragments that is still incomplete. Such messages are of no real value at this time – you already know that sheet entries A through E in sheet symbol B have no corresponding ports below, as you won't be defining that descendent sheet until Thursday morning, in-between a meeting and a final check of Gerbers for another deadline project! Wouldn't it be great if this visual message clutter – this message 'noise' – could be suppressed, leaving only pertinent messages that relate to those elements of the circuit that actually do exist? Altium Designer hears that resounding "yes" and provides the solution through its No ERC and Compiler Mask directives.

**No ERC directive**

*Main article:* [Schematic No ERC](#)

Place a No ERC directive on a node in the circuit to suppress any report warnings and errors that may be generated when compiling the schematic. Use this directive to deliberately prevent error checking of certain parts of a circuit that you know will generate an error – such as unconnected input pins, ports and sheet entries – while checking the rest of the circuit.
Figure 2. Place No ERC directives on unused output pins.

You may have come across the Compiler-related message "Signal x has no load". This is typically the result of an output pin that is either not yet connected to another part of the circuit or is not being used at all. Placing a No ERC directive on such a pin will suppress this message. If you are not using the pin at all, leave the No ERC directive in place, otherwise remove it once you have wired the pin to its intended destination.

How many times have you encountered a warning about a net not having a driving source, only to find that the message can be safely ignored? Perhaps an input pin is fed from a connector, the pin of which is nominally passive and the driving signal only present when an external cable is plugged in? Maybe the net is sourced from a pull-up resistor or switch, again passive in nature? One of the following strategies could be adopted to resolve this warning:

- You could change the electrical characteristic of a source pin on the net. This is a fix rather than suppression, but as it involves a change to a pin's default mode of operation, it could create trouble further down the track. For example, consider wiring changes made to a design, in which the graphical display of pin direction is not enabled. Such changes might result in an output being connected to a pin of a passive device. If the pin of that device has been set electrically as an output (to alleviate previous driving source warnings), then you will have created a connection violation.
- You could set the report mode for the associated violation check – defined on the Error Reporting tab of the Options For Project dialog (Project » Project Options) – to No Report. This disables the check of this particular violation, but you would also not be able to catch any genuine errors elsewhere in the design.
- The third (and arguably best) option is to place a No ERC directive on the net. You are not changing the design in any way, other than to suppress warning message clutter that you know is not a problem.
Figure 3. Place No ERC directives on nets you know will cause 'no driving source' warnings.

See No ERC to learn more about ERC placement and editing.

**Compile Mask directive**

A No ERC directive is great for the suppression of a low number of unconnected pins, ports, or sheet entries, or nets that only contain passive device pins. Perhaps the largest number of these directives would be found on unused pins of an FPGA device, or the pins of a connector. Should you wish to quickly 'blanket' larger regions of design circuitry – including components, net identifiers and wiring interconnects – then you should consider using a Compile Mask directive.

As its name suggests, this directive simply instructs the Compiler to ignore any objects that fall completely within the bounds of the defined mask. Place the mask exactly as you would a note or rectangle object.

Consider the example schematic circuitry in Figure 4, where the wiring to the LCD1 device is not yet complete. Compiling just this schematic (Project » Compile Document) will result in numerous violation messages (also shown), each of which is caused by the incomplete circuitry.
By placing a Compile Mask directive around the incomplete circuitry, these violations will be ignored by the Compiler, while the rest of the circuit on the schematic – which is completely wired – is checked. Notice that objects that are truly masked – those that completely fall within the bounding
rectangle of the mask – will appear grayed-out.

Figure 5. Placing a Compile Mask to remove a section of design from the Compiler's 'eyes'.

See Compile Mask to learn more about Compile Masks.

**PCB-related directives**

As part of its fully-integrated design environment, Altium Designer provides the ability to define PCB requirements prior to laying out the board. This is achieved by adding and specifying parameters to objects placed on the schematic sheet(s).

For certain schematic design objects – such as components, sheet symbols, ports, etc – this simply involves adding the relevant parameter(s) as part of that object's properties. For net objects such as wires and buses, parameters can not be added directly as a property of the wire or bus. Instead, the parameters required to hold the information are specified using dedicated design directives.

The following information can be specified, using directives, which will be transferred to the appropriate PCB-based definitions during design synchronization:

- PCB layout constraints
- Net classes
- Differential pairs

**Why specify such information, pre-layout?**

So just why would you define design rules, net classes and differential pairs on the schematic side? This question is often pondered – after all, each can be easily specified on the PCB side. There are typically two scenarios in which pre-layout definition of such information is desired:

- You may wish to fully define the design requirements on the schematic, such that the schematic becomes the 'master record' for the design. Any amendments to the design would be carried out on the schematic side only and pushed across to the PCB.
- You may have two different people working on the design – one capturing it and the other laying it out on the PCB. The person capturing the design may wish to ensure that the particular constraints they have in mind are indeed used during the layout phase. Rather than a constant stream of emails or phone calls, it is far easier (not to mention more accurate) to define the constraints pre-layout and relay them through the synchronization process.

**Parameter Set directive**

All of the information types listed previously are specified using a Parameter Set directive. As its name suggests, it is simply a container for one or more parameters, which in turn target a required net object within the schematic design.

![Parameter Set](image)

A default Parameter Set directive, one that is devoid of parameters, can be placed (Place » Directives » Parameter Set) and the relevant parameter(s) added to it. However for the three types of information listed, there are predefined Parameter Set directives that can be used (available from the same sub-menu). The following sections take a closer look at using these parameter-based directives.

See [Parameter Set](#) to learn more about Parameter Set placement and editing.

**PCB Layout directive**

Place a PCB Layout directive on a wire or bus to define one or more design constraints targeting the associated net(s).

The parameters defined in this type of directive must be rule-based parameters. A default rule parameter with undefined value is present in a newly-placed PCB Layout directive. A rule parameter is defined from the Choose Design Rule Type dialog, accessible from a parameter's associated properties dialog (Figure 7).
Use the Choose Design Rule Type dialog to choose the rule that you wish to add as a rule parameter to the directive. Double-clicking on a rule type will give you access to the relevant Edit PCB Rule (From Schematic) dialog, from where you can define the constraints for the rule (Figure 8).
Figure 8. Specifying the constraints for a chosen rule.

The entry for the parameter’s Value field will be the rule type chosen, along with the specified constraints. Figure 9 illustrates several defined rule parameters for a PCB Layout directive. When the design is transferred to the PCB, the relevant design rules will be created, based on the information contained within a directive. The word Schematic is used in the name for each generated rule, to distinguish the source of that rule.
Figure 9. Defining multiple constraints for a particular net.

Figure 10 illustrates the resulting rule entries in the PCB Rules and Constraints Editor dialog, for the PCB Layout directive defined in Figure 9.

A PCB Layout directive attached to a wire will generate a design rule with a rule scope of Net (e.g. InNet('LCD_LIGHT')). If the directive is attached to a bus, the generated design rule will
Design constraint information can be specified for other design objects by adding the relevant parameters, configured as rules. For more information, see the section Using Design Directives in a Schematic Document.

**Rule synchronization**

Right, so I can define rule directives on my schematic and transfer them to the PCB as the relevant design rules. Great. But what if a rule gets tweaked on either side? Don't worry, synchronization between a rule directive and its corresponding PCB rule is maintained through use of Unique IDs. When adding design rule parameters to objects on a schematic, a unique ID is given to each. The same IDs are given to the corresponding design rules that are created in the PCB. With this Unique ID, the constraints of a rule can be edited on either the schematic or PCB side and the changes pushed through upon synchronization.

Figure 11 shows an example of adding a PCB Layout directive to a wire in a schematic. In the example, a Width rule has been defined for the wire which, when passed to the PCB design upon synchronization, appears as an additional Width rule. The important thing to notice is that the Unique ID entries for the parameter and the PCB design rule are the same.
Net Class directive

Net Class directives enable you to create user-defined net classes on the schematic. When a PCB is created from the schematic source documents, the information in a Net Class directive is used to create the corresponding net class on the PCB. To make a net a member of a net class, simply attach a directive of this type to the relevant wire or bus and set the directive's ClassName parameter to the
name of the desired class. Figure 12 illustrates use of a Net Class directive to add three individual power nets (POS1_8V, POS3_3V, POS5V) to a new net class with the name PowerNets.

Net Class directives can only be used to create net classes on the PCB side, provided the Generate Net Classes option (for User-Defined Classes) is enabled, on the Class Generation tab of the Options for Project dialog (Project » Project Options).
Component classes can also be defined on the schematic, by adding a ClassName parameter to the required component(s). For more information, see the section Using Design Directives in a Schematic Document.

Differential Pair directive

Figure 13. Defining differential pairs on the schematic.

Use Differential Pair directives to define differential pairs on the schematic. The two nets that constitute a pair must be named with the suffixes _N and _P, and a directive must be attached to each. Figure 13 illustrates the definition of two differential pairs. Each pair consists of two nets – one positive and one negative. A Differential Pair directive has been added to each of the four nets.

So what's under the hood? Double-click on a directive and you'll see that the Parameters dialog contains a single parameter entry, with a name of DifferentialPair and a value of True. That parameter is all that's needed – you don't have to edit a thing. It really is a case of Place N Go!

Figure 14. The parameter for a Differential Pair directive is completely predefined.

The differential pair definitions will be transferred to the PCB during design synchronization, which can be quickly verified using the PCB panel, configured in Differential Pair Editor mode (Figure 15).
Figure 15. Verify creation on the PCB side using the PCB panel.

For more information on Differential Pairs, refer to the application note Interactive and Differential Pair Routing.

**Indirect design directives**

Additional information may be specified on the schematic, which will also be used to create corresponding objects when the design is transferred to the PCB. Such information is not defined by placing a directive object, but by adding (and defining) one or more parameters to the relevant schematic object. In essence, they are parameter-based directives.

**Specifying additional design constraints**

Additional design constraints may be specified on the schematic side, which will translate into PCB design rules with appropriate scope. For example you may wish to define a height constraint for a particular component, or a clearance constraint targeting all objects in the design. The required parameter that defines a constraint is added to the object as a rule, in the same way as parameters added to a PCB Layout directive. The scope of the corresponding PCB design rule, created as a result of the design synchronization process, is determined by the nature of the object to which the parameter (added as a rule) is assigned. Table 1 summarizes the schematic parameter-to-PCB rule scope options that are supported, including those defined by placing PCB Layout directives.

Table 1. Support rule definitions on the schematic side.
Add a Parameter (as a rule) to a... From... For a PCB rule scope of...

Pin
- the Parameters tab of the Pin Properties dialog
- Pad

Port
- the Parameters tab of the Port Properties dialog
- Net

Wire
- the Parameters dialog, after placing a PCB Layout Directive (Parameter Set object) on the wire using the Place » Directives » PCB Layout command
- Net

Bus
- the Parameters dialog, after placing a PCB Layout Directive (Parameter Set object) on the bus using the Place » Directives » PCB Layout command
- Net Class

Harness
- the Parameters dialog, after placing a PCB Layout Directive (Parameter Set object) on the harness using the Place » Directives » PCB Layout command
- Net Class

Blanket
- Place » Directives » PCB Layout command. Include a ClassName parameter to create a net class for all nets covered by the blanket, which will then be used for the rule scope.
- Net Class

Component
- the Parameters region of the Component Properties dialog
- Component

Sheet Symbol
- the Parameters tab of the Sheet Symbol dialog
- Component

Sheet
- the Parameters tab of the Document Options dialog (Design » Document Options )
- All Objects

Specifying component classes

In a similar vein, component classes can be defined on the schematic. Simply add a standard parameter to a component, with a name set to ClassName and a value set to the desired class name. When the design is transferred to the PCB, the defined component classes will only be created provided the Generate Component Classes option (for User-Defined Classes) is enabled, on the Class Generation tab of the Options for Project dialog.

FPGA-related directives

As part of its LiveDesign methodology, Altium Designer provides you with the ability to analyze nets in an FPGA design, while it is running on the target physical device. Where a design net is connected directly to a pin of the physical device, you have the ability to monitor the status of that pin – in real-time. By using a virtual instrument, such as a logic analyzer, you can gain valuable operational information about your design. The ability to 'reach in' and interactively analyze design nets can give you the edge needed to finalize development prior to manufacture, leading to shorter time-to-market and making a good product great.

To aid in this analysis, Altium Designer provides the Probe and Instrument Probe directives.
**Probe directive**

The days of being able to attach fly-lead probes physically to the pins of a chip and analyze states is fast nearing the doors of antiquity. Devices – especially FPGAs – are becoming pin-saturated beasts which, even if you could 'see' the pins, they are undoubtedly too finely pitched to access externally. So how about monitoring pins from within the design – directly from the schematic sheet? It's a vision made real with LiveDesign and a well-placed Probe directive.

![Figure 16. FPGA pin status monitoring using Probe directives.](image)

Place a Probe directive on any net (wire or bus) in the design that connects directly to a pin of the physical FPGA device. Once the design has been programmed into the device, the state of such a pin can be monitored, in real-time, directly on the schematic sheet (Figure 16).

The key to the pin-state display is the directive's ProbeValueDisplay parameter. In fact you could place a Parameter Set directive instead, and as long as this parameter is defined for it, the result would be the same.

For the pin state to be updated in real-time on the schematic, the source FPGA design must be downloaded to the physical FPGA device and the associated JTAG Viewer panel for that device must be open and remain open. The JTAG Viewer panel for a physical device is accessed by clicking the JTAG Viewer Panel button on the associated instrument panel for that device. The latter is loaded into the Instrument Rack – Hard Devices panel upon double-clicking the entry for the device, in the Hard Devices chain of the Devices view. For more information on one of these panels, press F1 with the cursor over the (focused) panel.

**Instrument Probe directive**

Tip:

In addition, when attached to a net that connects directly to an FPGA device pin, the Instrument Probe directive behaves as per the Probe directive. Again, the JTAG Viewer panel must be open.

Probing device pins from the schematic is great, but what about being able to analyze and potentially fault-find further back into the design circuitry? No problem. Simply place an Instrument Probe
directive, which allows you to monitor any point in a design – not just the status of FPGA device pins. Placed on a net object, such as a wire or bus, it instructs the system to connect the associated net directly to the input of a monitoring instrument (e.g. a logic analyzer) – without having to explicitly wire that net up through the design hierarchy to the sheet with the instrument on it.

The key element that turns this directive from being a simple probe into a powerful point-monitoring aid, is its possession of the additional InstrumentProbe parameter. Once the directive is placed at the point of interest, it is this parameter that is used to effectively ‘link’ that point to the monitoring device. Simply enter a meaningful name for the probe point – such as the name of the associated net or the particular signal being monitored – then connect a wire to the required input of the monitoring instrument and attach a net label to the wire, the name of which is the same name you have defined for the InstrumentProbe parameter (Figure 17).

![Diagram](https://via.placeholder.com/150)

**Figure 17.** Monitoring select points in a design without excessive wiring overhead.

Measurement information obtained through the directive's ProbeValueDisplay parameter is, like the Probe directive, displayed in real-time, over the Hard Devices JTAG chain. In contrast, measurement information obtained through the directive's InstrumentProbe parameter is ONLY displayed after compilation, over the Soft Devices JTAG chain.

**A Word about probing buses**

When an Instrument Probe directive is attached to a bus, the entire bus is taken up to the top-level sheet, irrespective of the name you assign to the InstrumentProbe parameter. When you add a net label to the input for the monitoring instrument, you must define the bus width required. Consider, for example, having attached an Instrument Probe directive to a bus with identifier Port1_Out[7..0] on a lower-level sheet. The value for the InstrumentProbe parameter could be simply set to Port1_Out. The entire bus will be connected up to the sheet with the monitoring device (e.g. a configurable LAX). Should you wish to wire up the whole bus as an input signal to the device, you would place a bus to the required input and add a net label of Port1_Out[7..0]. If you only wanted a particular signal or range of signals from the bus, you can simply define the width required in the attached net label (e.g. Port1_Out[4..2]).