B. BNF for Joule Syntax

This chapter presents, in Backus-Naur form, a grammar for the Joule language forms and expression syntax. Lexical conventions will appear in a later version of this Appendix.

B.1. BNF Conventions

In the BNFs in this appendix, the following conventions apply:

- Italicized names indicate terminals. The terminals are not presented in this Appendix. See **Section 4.1:** *Lexical Conventions* for an informal presentation.
- Verticals ("|") are used to separate alternative components that may be used in the same place.
- A question mark ("?") following a component means exactly zero or one instance of the component is allowed.
- An asterisk ("*") following a component means zero or more instances of the component are allowed.
- A plus sign ("+") following a component means one or more instances of the component are allowed.
- Braces ("{ }") are used to indicate grouped components, to which one of the preceding allowance indicators applies as a unit. `{fee fie}*` means zero or more instances of the series `fee fie` are allowed.
- A component followed by some delimiter `foo` and an asterisk means that zero or more instances of the component may be present, separated by `foo`. For example, "`{bar},*`" means that any number of `bar` components may be present, separated by commas.
- A component followed by some delimiter `foo` and a plus sign means that one or more instances of the component may be present, separated by `foo`.
- A production name for which multiple definitions are given means that any one of the definitions may be used where that token appears.
- The indentation describes the indentation rules that were generally used throughout this manual, but has no semantic significance.

B.2. Forms

Production	Production Definition
block	{form}*
form	• simpleExpr {opExpr},+ {**then** opExpr}?
	Define {param | param = opExpr},* block **endDefine**
	ForAll param ⇒ param block **endForAll**
	ForOne param ⇒ param param block **endForOne**
	Handler opExpr block **endHandler**
	HandlerTap opExpr block **endHandlerTap**
	Keeper opExpr block **endKeeper**
	Signal opExpr
	If opExpr block {**orIf** opExpr block}* {**elseIf** opExpr block {**orIf** opExpr block}* }* {**else** block}? **endIf**
	Switch opExpr {**case** pattern {**or** pattern}* block}* {**otherwise** param block}? **endSwitch**
	Type param {**super** *Identifier*}? {**op** {pattern}**or**+ block {**to** *Identifier* {opExpr},+ block}*}* **endType**
	Server param {method}? {var}* ops {facet}* **endServer**
var	**var** {param | param = opExpr},* block

Production	Production Definition
ops	{**implements** *Identifier*}? {**op** method}* {**otherwise** param block}
method	{pattern}**or**+ block {change block}*
change	**to** *Identifier* {opExpr},+ | **set** {*Identifier* = opExpr},+
facet	**facet** param ops

B.3. Expressions

opExpr	simpleExpr | simpleExpr *Operator* opExpr
simpleExpr	*Identifier* | *Literal* | *Quasiliteral* | tuple | '(' nestExpr ')'
nestExpr	simpleExpr | simpleExpr opExpr
tuple	{*Operator* | *Label* } {opExpr}*
param	*Identifier*
pattern	tuple | *Quasiliteral*