Phase 4 Documentation

Change made	**Location**	**Explanation/Justification**
Phase 3 changes	semantic.ssl	🎓 Updated PackageDefinition to include *oSymbolStkChooseKind* 🎓 Updated VariableDeclarations to not expressly emit tLiteralInteger and also added *oValuePushSymbol, oEmitValue,* and *oValuePop* to properly create the tLiteralAddress compound token
Input tokens	Input tokens in coder.ssl	🎓 Copied t-code output tokens from “semantic.ssl” into “coder.ssl” 🎓 Removed “tLiteralChar” and “tLiteralString” from input 🎓 Added “twofiftysix = 256” to integer type, and “string = 3” to data kind. 🎓 Removed old trap codes and replaced with new trap codes for string operations. 🎓 Copied “coder.def” contents into “coder.pt”
Mixed declarations and statements	“Block” and “Statement” rules in coder.ssl	🎓 Moved options from “Statement” rule to “Block” rule to allow a mix of statements and declarations to be handled 🎓 Changed “Statement” rule to call Block rule (rather than removing and refactoring) to allow declarations where statements can be called
Initial Values	“Block” rule in coder.ssl	🎓 Added an alternative for “tInitialValue” in the “Block” rule 🥂 Call “OperandPushExpression” rule to accept the initial value expression 🥂 Expect “tInitEnd” and “tLiteralAddress” tokens after pushing the expression 🥂 Use *oOperandPushVariable* and *oOperandSwap* to prepare the variable and value on the stack as expected for the subsequent “OperandAssign...PopPop” rule calls, which handle initial values of Integers, Addresses, Chars (Strings), and Booleans
Choose statement else clauses	“EmitDefaultCaseAbort” in coder.ssl	🎓 Added the else clause for a choose statement since it’s a part of Like and not PT 🎓 Added a choice option to check for “tCaseElse” then call the “Statement” rule and emit a *oEmitCaseMergeBranch* 🎓 If “tCaseElse” doesn’t exist then use the original call to case abort trap.

String constants	“OperandForceIntoTemp” and “OperandForceToStack” in coder.ssl and coder.pt	🎓 In “OperandForceIntoTemp”, changed handling of string length operands to call “OperandForceAddressIntoTemp”. 🎓 In “OperandForceToStack”, changed handling of string length operands to call “OperandForceAddressIntoTemp”. 🎓 Added “stringSize = 256” to machine size options in “coder.pt”. 🎓 Changde “assert19” error to return a message for “tStringData”. 🎓 In “AcceptInputToken”, changed handling of “tStringData” to read a trailing null character in “coder.pt”
String variables	“OperandPushVariable”, “OperandPushExpressionAssignPopPop“, and “OperandAssignCharPopPop” in coder.ssl	🎓 Changed “tFetchChar” case to “OperandPushVariable” to set operand’s length to string from byte data kinds. 🎓 Added “tSkipString” case to “OperandPushExpressionAssignPopPop” to allow assigning string values to variables. 🎓 Changed “OperandAssignCharPopPop” to call the new “trAssignString” trap.
String arrays	“OperandCheckedSubscriptNonManifestCharPop” and “OperandUncheckedSubscriptNonManifestCharPop“ in coder.ssl and coder.pt	🎓 Added code to scale each subscript by string size (256) in “OperandCheckedSubscriptNonManifestCharPop” and “OperandUncheckedSubscriptNonManifestCharPop“. 🎓 Added a case for string length operands in “OperandFoldManifestSubscript“ in “coder.pt” to scale the address subscript by stringSize.
String operations	“OperandPushExpressionAssignPopPop” and “OperandPushExpression” in coder.ssl	🎓 Added new rules to “coder.ssl”: 🥂 “OperandConcatenatePop” accepts 2 string inputs and returns a string output in the result register. 🥂 “OperandRepeatStringPop” accepts an integer and a string inputs and returns a string output in the result register. 🥂 “OperandSubstringPopPop” accepts 2 integers and a string inputs and returns a string output in the result register. 🥂 “OperandLength” accepts a string input and returns an integer output in the result register. 🥂 “OperandStringEqualPop” accepts 2 string inputs and returns a boolean output in the result register.