LambLisp vs. Scheme R5RS and R7RS Specifications - compatibility matrix

Feature status color codes:	🟢	Supported
	○	Road Map Phase 1
	◌	Road Map Phase 2
	🔴	Unsupported
	⚫	Unspecified
	▲	Source license required

	Scheme R5RS	Scheme R7RS	Lamb Lisp
Embedded Systems Adaptations			
Arduino-compatible API (Wire, WiFi, analog & digital I/O, …)	🔴	🔴	🟢
Stop-the-world adaptive garbage collector	⚫	⚫	🟢
Incremental adaptive garbage collector	⚫	⚫	△
High performance optimized type hierarchy	⚫	⚫	🟢
Integrated high speed hash tables	🔴	🔴	🟢
Class and object system	🔴	🔴	○
High speed integer and float instructions	🔴	🔴	🟢
Bitwise operators & | ^ (i.e., bitwise AND OR XOR)	🔴	🔴	🟢
Timers to support asynchronous operation	🔴	🔴	🟢
Common interface for all native procedures	⚫	⚫	🟢
Links with existing C++ hardware drivers	⚫	⚫	🟢
Logging facility	🟢	🔴	🟢
Comprehensive interface to operating system	⚫	⚫	🟢
Concurrent Lisp VM instances	⚫	⚫	🟢
Scheme language features			
Proper tail recursion, lexical scoping, “duck” typing, REPL	🟢	🟢	🟢
Datum labels	🔴	🟢	○
#u8 data type	🔴	🟢	🟢
Type predicates			
boolean? char? number? symbol? pair? vector?	🟢	🟢	🟢
procedure? string? port? eof-object?	🟢	🟢	🟢
null? bytevector?	🔴	🟢	🟢
Procedures			
define lambda	🟢	🟢	🟢
nlambda	🔴	🔴	🟢

Conditionals			
if else cond case and or not	🟢	🟢	🟢
when unless	🔴	🟢	🟢
cond-expand case-lambda	🔴	🟢	🔴
Assignments, Binding, and Syntax Definition			
set! define let let* letrec =>	🟢	🟢	🟢
define-syntax let-syntax letrec-syntax	🟢	🟢	○
syntax-rules	🟢	🟢	○
syntax-error	🔴	🟢	○
let-values let*values define-values	🔴	🟢	○
include	🔴	🟢	○
include-ci	🔴	🟢	🔴
nlambda	🔴	🔴	🟢
Evaluation and quotation			
quote quasiquote unquote unquote-splicing	🟢	🟢	🟢
Reader macros ' ` , ,@	🟢	🟢	🟢
begin do “named let”	🟢	🟢	🟢
Delayed Evaluation			
delay force	🟢	🟢	○
delay-force promise? make-promise	🔴	🟢	○
Dynamic bindings			
make-parameter	🔴	🟢	🔴
parameterize	🔴	🟢	🔴
Libraries and Importing			
import only except prefix rename define library	🔴	🟢	🔴
Records			
define-record-type	🔴	🟢	○
Equivalence Predicates			
eq? eqv? equal?	🟢	🟢	🟢
Numeric types			
integer real	🟢	🟢	🟢
complex	🔴	🟢	◌
rational	🔴	🟢	○
Numeric Operations			
number? complex? real? rational? integer?	🟢	🟢	🟢
exact? inexact? exact-integer? finite? infinite?	🔴	🟢	🟢
nan? zero? positive? negative? odd? even?	🟢	🟢	🟢
abs max min + - * / < <= = >= >	🟢	🟢	🟢
quotient remainder modulo	🟢	🟢	🟢

floor ceiling truncate round	🔴	🟢	○
floor/ floor-quotient floor-remainder	🔴	🟢	○
truncate/ truncate-quotient truncate-remainder	🔴	🟢	○
numerator denominator gcd lcd	🟢	🟢	○
abs expt log square sqrt	🟢	🟢	🟢
sin cos tan asin acos atan	🔴	🟢	🟢
exact-integer-sqrt	🟢	🟢	○
real-part imag-part magnitude angle	🟢	🟢	◌
make-rectangular make-polar	🟢	🟢	◌
number->string string->number	🟢	🟢	○
Pairs and Lists			
pair? cons car cdr set-car! set-cdr! list?	🟢	🟢	🟢
null?	🔴	🟢	🟢
atom?	🔴	🔴	🟢
make-list list	🔴	🟢	○
caar .. cddr (all combinations of car & cdr)	🟢	🟢	🟢
caaaar .. cddddr	🟢	🟢	🔴
append a reverse list-tail list-ref list-set! list-copy!	🟢	🟢	🟢
reverse!	🔴	🔴	🟢
memq memv member assq assv assoc	🟢	🟢	🟢
vector->alist alist->vector	🔴	🔴	🟢
Symbols			
symbol? symbol=? symbol->string string->symbol	🟢	🟢	🟢
Characters			
char? char=? char<? char>? char<=? char>=?	🟢	🟢	🟢
char-alphabetic? char-numeric? char-whitespace?	🟢	🟢	◌
char-uppercase? char-lowercase?	🟢	🟢	○
char→integer integer→char	🟢	🟢	🟢
Case-independent char-ci-* functions	🟢	🟢	🔴
char-upcase char-downcase	🟢	🟢	🟢
char-foldcase	🔴	🟢	🔴
digit-value	🔴	🟢	○
Strings			
string? make-string string string-length string-ref string-set!	🟢	🟢	🟢
string<? string <=? string=? string>=? string>?	🟢	🟢	🟢
Case-independent string-ci functions	🟢	🟢	🔴
substring string-append string->list list→string	🟢	🟢	🟢
string-copy string-fill!	🟢	🟢	🟢
string-copy!	🔴	🟢	🟢

string-foldcase	🔴	🟢	🔴
Vectors			
vector? make-vector vactor-length vector-ref vector-set!	🟢	🟢	🟢
vector-fill! vector->list list->vector	🟢	🟢	🟢
vector->string string->vector vector-append	🔴	🟢	○
vector->alist alist->vector	🔴	🔴	🟢
vector-copy	🔴	🟢	🟢
Bytevectors			
bytevector? make-bytevector bytevector bytevector-length	🔴	🟢	🟢
bytevector-u8-ref bytevector-u8-set!	🔴	🟢	🟢
bytevector-copy bytevector-copy! bytevector-append	🔴	🟢	🟢
utf8->string string->utf8	🔴	🟢	🟢
Control Features			
procedure? apply map for-each	🟢	🟢	🟢
string-map vector-map	🔴	🟢	○
string-for-each vector-for-each	🔴	🟢	○
force delay	🟢	🟢	○
call-with-current-continuation	🟢	🟢	🔴
values call-with-values	🟢	🟢	🔴
dynamic-wind	🟢	🟢	🔴
Exception handling			
guard raise	🔴	🟢	○
with-exception-handler	🔴	🟢	○
raise-continuable	🔴	🟢	○
error error-object? error-object-message error-object-irritants	🔴	🟢	○
read-error? file-error?	🔴	🟢	○
Environments and Evaluation			
environment	🔴	🟢	🟢
eval scheme-report-environment	🟢	🟢	🟢
null-environment interaction-environment	🟢	🟢	🟢
Ports			
call-with-input-file call-with-output-file	🟢	🟢	◌
call-with-port	🔴	🟢	◌
port? input-port? output-port?	🟢	🟢	🟢
textual-port? binary-port?	🔴	🟢	🟢
input-port-open? output-port-open?	🔴	🟢	🟢
current-input-port current-output-port close-port	🟢	🟢	🟢
current-error-port	🔴	🟢	🟢
with-input-from-file with-output-to-file	🟢	🟢	○

open-input-file open-output-file	🟢	🟢	🟢
open-input-binary-file open-output-binary-file	🔴	🟢	🟢
close-input-port close-output-port	🔴	🟢	🟢
open-input-string	🔴	🟢	🟢
open-output-string get-output-string	🔴	🟢	🟢
open-input-bytevector	🔴	🟢	🟢
open-output-bytevector	🔴	🟢	🟢
Input and Output			
read read-char peek-char eof-object? char-ready?	🟢	🟢	🟢
read-string read-line	🔴	🟢	🟢
read-u8 peek-u8 u8-ready? read-bytevector read-bytevector!	🔴	🟢	🟢
write display newline	🟢	🟢	🟢
write-shared write-simple	🔴	🟢	○
write-char-string write-u8 write-bytevector flush-output-port	🔴	🟢	○
System Interface			
load	🟢	🟢	🟢
file-exists? delete-file	🔴	🟢	○
command-line	🔴	🟢	○
exit emergency-exit	🔴	🟢	🔴
get-environment-variables	🔴	🟢	○
current-second current-jiffy jiffies-per-second	🟢	🟢	🟢
features	🔴	🟢	○