LTspice .asc 回路図ファイルフォーマットの詳細解析：構文、高度な応用、そして未公開機能

第1章: .ascファイルの解剖学：基本概念

LTspiceにおける回路設計とシミュレーションの中核をなす.ascファイルは、一見すると単純なテキストファイルですが、その内部にはグラフィカルな回路図を再現し、シミュレーションを実行するための厳格なルールと豊富な情報が格納されています。このセクションでは、.ascファイルの基本的な役割、構造、そして主要なキーワードの構文について解説します。

1.1. LTspiceエコシステムにおける.ascファイルの役割

.ascファイルは、LTspiceの回路図エディタで作成されるグラフィカルな回路情報を保存するためのファイルです 。これは純粋なSPICEネットリスト（.netや.cirファイルなど）とは根本的に異なります [2]。

.ascファイルの主な役割は、コンポーネントのシンボル、配線、ラベル、その他のグラフィカル要素の配置情報を保持することです [5]。

シミュレーションを実行する際、LTspiceは以下のプロセスを経ます：

1. ユーザーが作成した.ascファイルを解析する。
2. .ascファイル内のグラフィカル情報（コンポーネントの種類、接続関係、値など）から、シミュレーションエンジンが解釈できるSPICEネットリスト（.netファイル）を内部的に生成する [7]。
3. 生成された.netファイルをシミュレーションエンジンに渡し、計算を実行する。

このワークフローは、.ascファイルが持つ二重の性質を浮き彫りにします。それは、GUIが回路図を「描画」するための命令セットであると同時に、シミュレーションエンジンが回路を「理解」するためのソースコードでもあるという点です。この二重性こそが、.ascファイル内にWIREや

SYMBOLの座標情報と、SYMATTR Valueのような論理的な値が共存している理由です。この点を理解することは、.ascファイルが単なるネットリストではないという基本的ながら重要な概念を把握する上で不可欠です。

1.2. ファイル構造と重要なフォーマットルール

.ascファイルはプレーンテキスト形式ですが、その構造は厳格に定められています。手動編集や外部ツールでの生成を行う際には、以下のルールを遵守する必要があります。

- 行ベースの構文: ファイルは、キーワードで始まる単一行のステートメントの連続で構成されます。各ステートメントは、必ず改行コード（CR/LF）で終了しなければなりません [10]。Webページなどからテキストをコピー＆ペーストした際に改行が失われ、ファイル全体が一行になってしまうと、LTspiceはファイルを正しく解釈できません [10]。
- 座標系: 回路図上のすべてのグラフィカル要素は、整数ベースの単純なX-Y座標系によって配置されます。これらの座標は、各キーワードのパラメータとして指定されます [4]。
- ファイルエンコーディングの罠: 「プレーンテキスト」という言葉は誤解を招きやすい側面があります。特にクロスプラットフォームでの作業において、ファイルエンコーディングは深刻な問題を引き起こす可能性があります。多くのフォーラム投稿で、テキストエディタで.ascファイルを編集した後にファイルが「壊れた」という報告がなされています [12]。その根本原因は、LTspice for Windowsが生成するus-asciiエンコーディングのファイルと、LTspice for Macが生成するUTF-16 LEエンコーディングのファイルとの間の非互換性にあります [12]。これらの異なるエンコーディングのファイルを混在させると、LTspiceは構文を解析できなくなります。したがって、.ascファイルを直接編集する際は、エンコーディングに注意を払うか、互換性のあるテキストエディタを使用することが極めて重要です。

1.3. 回路図オブジェクトの基本キーワード

.ascファイルは、回路の視覚的および電気的構造を定義する一連のキーワードによって構成されています。以下に、最も基本的なキーワードとその構文を示します [11]。

- **SHEET <width> <height> <number>**: 回路図シートのサイズや番号を定義します。
- **WIRE <x0> <y0> <x1> <y1>**: 2点間にワイヤ（配線）セグメントを定義します。
- **FLAG <x> <y> <net_name>**: 指定した座標にネットラベルを配置します。<net_name>に0を指定すると、グローバルなグラウンドノードとして扱われます [6]。

- **IOPIN <x> <y> <type>**: 階層回路ブロックのI/Oポートを定義します。<type>にはIn, Out, BiDir、または空文字列を指定できます [11]。
- **TEXT <x> <y> <justification> <font_size>!<text_content>**: 回路図上に任意のテキストを配置します。特に重要なのは、SPICEディレクティブ（ドットコマンド）がこのTEXTキーワードを用いて記述される点です。その場合、テキストの先頭に!または.が付きます [5]。
- **WINDOW <num> <x> <y> <justification> <font_size>**: 参照番号や値など、コンポーネントに付随するデフォルトテキストの表示位置やスタイルを、コンポーネントの原点を基準に定義します [11]。
- **SYMBOL <symbol_name> <x> <y> <rotation/mirror>**: コンポーネントを配置するための中心的なステートメントです。使用するシンボルファイル名（例: res, voltage）、配置座標、向き（回転・反転）を指定します。
- **SYMATTR <attribute_name> <value>**: SYMBOLステートメントの直後に記述され、そのコンポーネントの属性を定義します。最も一般的な属性は、インスタンス名（参照番号）を指定するInstName（例: R1）と、値を指定するValue（例: 1k）です [10]。このキーワードがコンポーネントのカスタマイズの鍵となります。

表1: 主要な.ascファイルキーワードとパラメータ

キーワード	構文 (パラメータ)	パラメータの説明	使用例
SHEET	<width> <height> <number>	width, height: シートの幅と高さ。number: シート番号。	SHEET 1 880 680
WIRE	<x0> <y0> <x1> <y1>	(x0, y0): 始点座標。(x1, y1): 終点座標。	WIRE 224 96 128 96
FLAG	<x> <y> <net_name>	(x, y): ラベルの配置座標。net_name: ネット名。	FLAG 288 192 OUT
SYMBOL	<name> <x> <y> <rot/mirror>	name: シンボル名。(x, y): 配置座標。rot/mirror: 回転/反転コード。	SYMBOL res 208 80 R0
SYMATTR	<name> <value>	name: 属性名。value: 属性値。	SYMATTR InstName R1

TEXT	<x> <y> <just> <size>!<content>	(x, y): テキストの配置座標。just: 行揃え。size: フォントサイズ。content: テキスト内容。	TEXT 120 376 Left 2!.tran 10mS

表2: SYMBOLキーワードの回転/反転コード

コード	説明
R0	回転なし (0度)
R90	90度回転
R180	180度回転
R270	270度回転
M0	左右反転後、0度回転
M90	左右反転後、90度回転
M180	左右反転後、180度回転
M270	左右反転後、270度回転

出典: [11]

第2章: シミュレーション制御と数値ルールの埋め込み

.ascファイルは単なるグラフィカルなレイアウト情報だけでなく、シミュレーションの挙動を制御するためのコマンドや、コンポーネントの値を定義するための数値ルールも内包しています。このセクションでは、これらのシミュレーション固有の要素が.ascファイル内でどのように記述されるかを解説します。

2.1. TEXTキーワードによるSPICEディレクティブ

SPICEのすべての「ドットコマンド」（.tran, .acなど）は、.ascファイル内ではTEXTオブジェクトとして実装されます [5]。これらのテキストは、シミュレーション時にLTspiceによって解釈される特別な命令であり、通常、先頭に

.または!が付けられます [6]。

一般的な解析コマンドの記述例は以下の通りです。

- 過渡解析 **(.tran)**: TEXT <x> <y> <just> <size>!.tran <stop_time>
  - 例: TEXT 120 376 Left 2!.tran 10mS [6]
- **AC**解析 **(.ac)**: TEXT <x> <y> <just> <size>!.ac {LIN,OCT,DEC} <points> <f_start> <f_stop>
  - 例: !.ac dec 100 1 1Meg [1]
- **DC**スイープ解析 **(.dc)**: TEXT <x> <y> <just> <size>!.dc <src_name> <start> <stop> <increment>
  - 例: !.dc V1 -5 5 0.1 [1]
- 動作点解析 **(.op)**: TEXT <x> <y> <just> <size>!.op [1]
- 初期条件設定 **(.ic)**: TEXT <x> <y> <just> <size>!.ic V(node)=value
  - 例: TEXT 8 72 Left 2!.ic v(OUT)=0V [6]

表**3: .asc**ファイルにおける一般的な**SPICE**ディレクティブ

ディレクティブ	.ascファイル内での構文例	目的
.tran	TEXT 10 10 Left 2!.tran 1ms	過渡解析を実行する。
.ac	TEXT 10 20 Left 2!.ac oct 20 10 1Meg	AC小信号周波数応答解析を実行する。
.dc	TEXT 10 30 Left 2!.dc Vin 0 5 0.1	DCスイープ解析を実行する。
.op	TEXT 10 40 Left 2!.op	DC動作点を計算する。
.ic	TEXT 10 50 Left 2!.ic V(out)=0	ノードの初期電圧条件を設定する。

.step	TEXT 10 60 Left 2!.step param R list 1k 2k 5k	パラメータを変化させながら繰り返しシミュレーションを実行する。
.param	TEXT 10 70 Left 2!.param Rload=1k	回路内で使用するパラメータを定義する。
.model	TEXT 10 80 Left 2!.model MyDiode D(Is=1p)	デバイスモデルを定義する。
.include	TEXT 10 90 Left 2!.include mymodels.lib	外部のモデルライブラリファイルを読み込む。

2.2. パラメトリック解析: .PARAM, .STEP, および関数

LTspiceでは、コンポーネントの値をパラメータ化し、その値をスイープさせることで、効率的な回路評価が可能です。

- **.PARAM**: TEXTオブジェクトとして.PARAM <parameter_name> = <value>と記述し、回路内で使用できるパラメータを定義します [3]。
- パラメータの参照: 定義したパラメータは、コンポーネントのValue属性内で波括弧 {} を使って参照します。例えば、R1の値をパラメータRvalにする場合、SYMATTR Value {Rval} と記述します [3]。
- **.STEP**: .STEP PARAM <name> LIST <v1> <v2>... のようにリスト形式で値を指定するか、.STEP PARAM <name> <start> <stop> <increment> のように範囲を指定して、パラメータを自動的に変化させながらシミュレーションを複数回実行します [3]。
- **.FUNC**: .FUNC <name>(args) {expression} 構文を用いて、ユーザー定義の関数を作成することも可能です [16]。

2.3. 数値と単位の表記ルール

LTspiceの数値パーサーは、一般的な工学表記法に従いますが、いくつかの特有のルールと注意点が存在します。これらを理解しないと、予期せぬシミュレーションエラーの原因となります。

- スケールファクタ: 数値の後ろに特定の文字を付加することで、桁数（スケール）を指定できます。これらのファクタは大文字と小文字を区別しません [1]。
    - T or t: テラ (1012)

    - G or g: ギガ (109)
    - MEG or meg: メガ (106)
    - K or k: キロ (103)
    - M or m: ミリ (10−3)
    - U or u: マイクロ (10−6) (LTspiceはuをμに置き換えて表示します)
    - N or n: ナノ (10−9)
    - P or p: ピコ (10−12)
    - F or f: フェムト (10−15)
- 表記上の重要な注意点:
    - **M vs MEG**: LTspiceのパーサーはケースインセンシティブであるため、Mとmは両方とも「ミリ」(10−3)として解釈されます。これは初心者が陥りやすい最も一般的な間違いの一つです。「メガ」(106)を意図する場合は、必ずMEG（またはmeg）を使用しなければなりません [1]。例えば、1メガオームの抵抗は
      1Mではなく1MEGと記述します。
    - **F vs Farad**: 同様に、F（またはf）は「フェムト」(10−15)として解釈され、「ファラッド」ではありません。1ファラッドのキャパシタの値を指定する場合は、単位を付けずに単に1と入力します [1]。
      1Fと入力すると、1フェムトファラッドとして扱われてしまいます。
    - **mil**: インチの1000分の1（25.4×10−6 メートル）を表すmilも使用可能です [6]。

これらのルールは直感的ではないため、特に手動で.ascファイルを編集したり、ネットリストを記述したりする際には細心の注意が必要です。

第3章: 高度な実装と例外的なケース

このセクションでは、ユーザーの要求の中心である、より複雑で非自明な実装例について掘り下げます。カスタムデバイスの統合、階層的な回路設計、振る舞い（ビヘイビア）電源の活用、そしてバス配線を持つデバイスの表現方法について詳述します。

3.1. サブサーキットによるカスタムデバイス (.SUBCKT)

LTspiceの標準ライブラリにないデバイスを使用する場合、サブサーキット（.SUBCKT）を利用してカスタムコンポーネントを定義するのが一般的です。

3.1.1. カスタムコンポーネントのエコシステム

カスタムコンポーネントをシームレスに利用するためには、通常、以下の3種類のファイルが連携して機能します [17]。

1. **.asc**ファイル: カスタムコンポーネントをインスタンス化して使用する、メインの回路図ファイル。
2. **.asy**ファイル: 回路図エディタ上でコンポーネントをグラフィカルに表現するためのシンボルファイル。
3. **.lib / .sub / .mod**ファイル: .SUBCKT....ENDSブロックを用いて、コンポーネントの電気的な振る舞いを定義するモデルファイル [19]。

3.1.2. 連携のメカニズム: SYMATTR属性

シンボル（.asy）とモデル（.lib）を結びつけるのは、SYMATTRキーワードで定義される属性群です。これらの属性は通常.asyファイル内で設定されますが、.ascファイル内で上書きすることも可能です。

- **Prefix X**: この属性は、LTspiceに対して、このコンポーネントが組み込みのプリミティブ素子ではなく、サブサーキットの呼び出しであることを示す、極めて重要なフラグです [21]。
- **ModelFile**: サブサーキット定義が含まれるライブラリファイル名（例: opamps.lib）を指定します。LTspiceは指定されたファイルをライブラリ検索パスから探します [22]。
- **Value (**または **SpiceModel)**: ModelFile内で使用する.SUBCKTの名前（例: LM741）を指定します。この名前は、.SUBCKT定義の名称と完全に一致する必要があります [22]。

これらの属性間の関係は、カスタムコンポーネントを機能させるための鍵であり、しばしば初心者がつまずくポイントです。この連携を理解することで、外部モデルの導入が格段に容易になります。

表**4:** カスタムコンポーネント連携のための**SYMATTR**属性

属性	目的	使用例	文脈と注意点
Prefix	コンポーネントの種類を定義する。	X	Xはサブサーキットを示す。Rは抵抗、Cはコンデンサなど。
Value	シンボルに表示される値であり、多くの場合サブサーキット名としても機能する。	LM741	SpiceModel属性が指定されていない場合、この値がサブサーキット名として使われる。
SpiceModel	使用するサブサーキットモデル名を明示的に指定する。	LM741.sub	Value属性とは独立してモデル名を指定できる。
ModelFile	サブサーキット定義を含む外部ファイル名を指定する。	national.lib	指定されたファイルはLTspiceのライブラリパスに存在する必要がある。

出典: [22]

3.1.3. .ascファイル内での表現例

以下は、カスタムオペアンプをインスタンス化する.ascファイルの記述例です。

```
SYMBOL my_opamp 128 128 R0
SYMATTR InstName XU1
SYMATTR SpiceModel LM741
SYMATTR ModelFile opamps.lib
```

この例では、my_opamp.asyというシンボルを使用し、インスタンス名をXU1としています。そして、opamps.libファイル内のLM741という名前のサブサーキットモデルを呼び出しています。

3.2. 階層回路: 回路図をコンポーネントとしてインスタンス化する

複雑な回路を管理するための強力な手法として、ある.ascファイル（子回路）全体を、別の.ascファイル（親回路）内で一つのコンポーネントとして扱う「階層化」があります [28]。

この手法の鍵は、標準的なサブサーキットとの連携方法の違いにあります。標準サブサーキットがSYMATTR ModelFileでモデルファイルを明示的に指定するのに対し、階層ブロックは暗黙的な命名規則によってリンクされます。

1. 子回路の作成: 子回路となる.ascファイルを作成し、外部との接続点となるネットにIOPINキーワードを用いてポート（In, Out, BiDir）を定義します [11]。
2. シンボルの作成: 子回路の.ascファイルと同じベース名を持つ.asyシンボルファイルを作成します。このシンボルには、子回路のIOPINで定義したネット名と一致する名前のピンを配置する必要があります [25]。
3. リンクの確立: 親回路の.ascファイルからこのシンボルを配置すると、LTspiceは同じディレクトリ内に同じベース名を持つ.ascファイル（子回路）が存在するかを自動的に検索し、リンクします。この際、シンボルのSYMATTR属性（Prefix, SpiceModel, ModelFile）はすべて空欄にしておく必要があります [25]。

この「設定より規約」に基づくアプローチは、関連ファイルを同じフォルダにまとめるだけで機能するため非常に便利ですが、この暗黙のルールを知らないと正しく動作させることができません。

3.3. 振る舞い（ビヘイビア）電源**:** 任意かつ関数的な制御

振る舞い電源は、電圧や電流を数式や他の回路の変数に依存して定義できる、非常に強力なコンポーネントです。これにより、複雑な伝達関数や非線形なデバイスを簡単にモデル化できます。

3.3.1. 構文と使用法

LTspiceには、任意の振る舞い電圧源（bv）と電流源（bi）が用意されています [31]。これらのコンポーネントの核心は、数式文字列を値として持つ

Value属性にあります [32]。

- 電圧源: SYMATTR Value V=<expression>
- 電流源: SYMATTR Value I=<expression>

3.3.2. 式の作成

<expression>には、ノード電圧V(node_name)、ノード間電圧V(node1, node2)、素子電流I(R1)、トランジスタのベース電流Ib(Q1)など、回路内の様々な量を使用できます [33]。さらに、多数の組み込み関数を利用して複雑な振る舞いを記述できます。

表5: 振る舞い電源の式で使用できる主要な関数

関数	構文	説明
if	if(x, y, z)	もし x > 0.5 なら y、さもなければ z を返す。
sin	sin(x)	x（ラジアン）の正弦を返す。
cos	cos(x)	x（ラジアン）の余弦を返す。
pow	pow(x, y)	x の y 乗を返す。
pwr	pwr(x, y)	abs(x) の y 乗を返す。
abs	abs(x)	x の絶対値を返す。
sqrt	sqrt(x)	x の平方根を返す。
log10	log10(x)	x の常用対数（底が10）を返す。
limit	limit(x, y, z)	x を y と z の間に制限する。
table	table(x, a, b, c, d,...)	x の値に基づき、(a, b), (c, d),... のペアで定義されるルックアップテーブルから値を補間する。
rand	rand(x)	x の整数値に依存する0から1の

		間の乱数を生成する。

出典: [7]

3.3.3. 高度な例: 条件分岐ロジック

if関数（または等価な三項演算子 condition? true_val : false_val）を使用することで、条件に応じた出力が可能です [34]。

- 電圧リミッタの例: V=if(V(in)>5, 5, if(V(in)<-5, -5, V(in)))
  - この式は、入力電圧V(in)が+5Vを超えたら出力を5Vに、-5Vを下回ったら-5Vにクランプし、それ以外の場合は入力電圧をそのまま出力します。

3.3.4. 例外的な例: 未公開の振る舞い抵抗

LTspiceには公式には文書化されていませんが、非常に強力な「振る舞い抵抗」を定義する機能が存在します。これはR=<expression>という構文で実現されます [32]。

- 電圧制御スイッチの例: R=if(V(control)>2.5, 0.1, 1G)
  - この式は、制御電圧V(control)が2.5Vより大きい場合は抵抗値を0.1Ω（ON状態）に、そうでない場合は1GΩ（OFF状態）に変化させます。
  - この未公開機能は、ユーザーが求める「例外的で難しい」記述の好例であり、単純なコンポーネントで複雑なスイッチング動作をモデル化する際に絶大な効果を発揮します。

3.4. 複数インターフェースを持つデバイス: バスとアレイ表記

デジタル回路や多チャンネルのアナログ回路では、複数の信号線をまとめてバスとして扱うと便利です。.ascファイルでは、バスは単一のキーワードではなく、複数のグラフィカル要素の組み合わせとして表現されます。

1. バスの定義: WIREでバスの幹となる太線を描画し、FLAGやIOPINでbus[0:7]のようにバス名と範囲を定義します [35]。

2. バス・タップ: BUSTAP <x1> <y1> <x0> <y0> キーワードは、個別のワイヤがバスの幹に接続する分岐点（タップ）のグラフィックを定義します [11]。
3. 分岐配線: タップから個別のWIREを引き出し、bus, busのようにネットラベルを付けます [35]。

3.4.1. 高度な例: コンポーネント・アレイ

バス表記は、コンポーネントの配列を効率的に作成するための強力なテクニックに応用できます [37]。例えば、集中定数モデルの伝送線路を多数の抵抗とインダクタのペアで作成する場合、一つ一つの素子を手で配置するのは非現実的です。

代わりに、以下のように記述できます。

- 抵抗シンボルを1つ配置し、そのInstName属性をR[1:256]、Value属性を10とする。
- インダクタシンボルを1つ配置し、そのInstName属性をL[1:256]、Value属性を1uとする。
- これらのコンポーネントを、in[1:256]とout[1:256]のようなバス配線の間に接続する。

これにより、LTspiceは自動的に256個の抵抗とインダクタのインスタンスを生成し、それぞれ対応するバスラインに接続します。これは非常に高度で非自明なテクニックであり、反復的な回路構造を扱う際に絶大な生産性向上をもたらします。

第4章: 専門家による洞察: 未公開機能とバージョン依存性

.ascファイルは非常に強力で柔軟なフォーマットですが、その利用には注意が必要です。特に、公式に文書化されていない機能の利用や、LTspiceのバージョンアップに伴う変更は、回路資産の互換性に影響を与える可能性があります。

4.1. 未公開機能の危険性

振る舞い抵抗（R=...）やサンプル＆ホールド素子 [38] のような未公開機能は、シミュレーションに大きな力をもたらしますが、それらはLTspiceの将来のバージョンで予告なく変更されたり、機

能しなくなったりするリスクを伴います [39]。

実際に、過去のバージョンアップでは、巧妙だが非公式な機能が動作しなくなった例が報告されています。

- サブサーキットのノード定義に数式を使用する機能 [39]。
- temp={temp}という記述でグローバルな温度パラメータをサブサーキットに渡す機能 [39]。
- モデル名に?や!のような特殊文字を使用する機能 [39]。

これらの事実が示唆するのは、.ascフォーマットがXMLやJSONのような静的な公式規格ではなく、LTspiceアプリケーションの進化と共に変化する「生きた」フォーマットであるということです。したがって、.ascファイルを解析・生成するツールや、未公開機能に依存した回路モデルを利用する際には、LTspiceのバージョンアップに際して互換性が失われる可能性があることを常に念頭に置く必要があります。これは、プロフェッショナルな環境でLTspiceを使用する上で極めて重要な戦略的アドバイスです。

4.2. 相互運用性と.ascフォーマット

KiCadやEasyEDAといった他のEDAツールが.ascファイルのインポートを試みる際、多くの課題に直面します [41]。最大の課題は、

.ascファイルが単体では機能しないことが多い点です。シミュレーションを完全に再現するには、関連する.asyシンボルファイルや.libモデルファイルが不可欠ですが、これらが提供されないケースが頻繁にあります [42]。

この事実は、第3.1節で述べたコンポーネントの「エコシステム」の重要性を裏付けています。回路図を他者と共有する際には、.ascファイルだけでなく、依存するすべてのカスタムシンボルとモデルファイルを一緒に提供することが、移植性を確保する上で不可欠です。

結論

本レポートでは、LTspiceの.asc回路図ファイルフォーマットについて、その基本構造から高度な応用例、そしてバージョン依存性といった専門的な側面に至るまで、詳細な分析を行いました。

.ascファイルは、回路のグラフィカルなレイアウトを記述する、人間が判読可能なテキストファイルです。その直接編集機能は高い柔軟性を提供しますが、構文、ファイルエンコーディングの微妙な違い、そしてバージョン間の非互換性といった点について深い理解が求められます。

特に、以下の高度な機能を習得することは、複雑なシミュレーションを効率的に行う上で大きなアドバンテージとなります。

- カスタムデバイス: .asyファイルと.libファイルを連携させ、SYMATTR属性を正しく設定することで、あらゆるデバイスを回路図に統合できます。
- 階層回路: 回路をブロック化し、命名規則に基づいてリンクさせることで、大規模な設計の可読性と再利用性を劇的に向上させます。
- 振る舞い電源: 数式や条件分岐を用いて任意の電圧・電流源を定義することで、物理的なコンポーネントではモデル化が難しい複雑な振る舞いをシミュレートできます。
- バスとアレイ: バス表記とコンポーネント・アレイのテクニックを組み合わせることで、反復的な構造を持つ回路を極めて効率的に記述できます。

一方で、.ascフォーマットはLTspiceアプリケーションと密接に結びついた「生きた」フォーマットであり、特に未公開機能への依存は、将来のバージョンアップによって互換性が失われるリスクを内包しています。したがって、安定したワークフローを維持するためには、公式にサポートされている機能を主軸に据えつつ、関連するすべてのファイル（.asc, .asy, .lib）を一つのエコシステムとして管理する意識が不可欠です。このフォーマットの能力と限界を正確に理解することが、LTspiceを最大限に活用するための鍵となります。

引用文献

1. Getting Started with LTspice - FAQs/Docs - EngineerZone - Analog Devices, 6月 18, 2025にアクセス、https://ez.analog.com/design-tools-and-calculators/ltspice/a/faqs-docs/c/getting-started-with-ltspice
2. B. Circuit description. - LTwiki-Wiki for LTspice, 6月 18, 2025にアクセス、https://ltwiki.org/files/LTspiceHelp.chm/html/CircuitDescription.htm
3. LTSpice Tutorial - part 1 - PedalPCB Community Forum, 6月 18, 2025にアクセス、https://forum.pedalpcb.com/threads/ltspice-tutorial-part-1.6872/
4. Open or Convert an LTspice Schematic? - QSPICE - Qorvo Tech Forum, 6月 18, 2025にアクセス、https://forum.qorvo.com/t/open-or-convert-an-ltspice-schematic/14156
5. asc LTspice support? · Issue #234 · drahnr/oregano - GitHub, 6月 18, 2025にアクセス、https://github.com/drahnr/oregano/issues/234
6. LTspice - Wikipedia, 6月 18, 2025にアクセス、https://en.wikipedia.org/wiki/LTspice
7. Introduction to LTSpice - Electrical and Computer Engineering (ECE), 6月 18, 2025にアクセス、http://www.ece.mcgill.ca/~grober4/SPICE/SPICE_Decks/1st_Edition_LTSPICE/chapter1/Chapter%201%20Introduction%20web%20version%20LTSpice.html

8. LTspice IV - Course Websites, 6月 18, 2025にアクセス、https://courses.grainger.illinois.edu/ece464/fa2019/ltspice/ltspice_help.pdf
9. LTspice: How to convert .asc file to netlist? - Electrical Engineering Stack Exchange, 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/106854/ltspice-how-to-convert-asc-file-to-netlist
10. Need help regarding opening .asc file in LTspice. - All About Circuits Forum, 6月 18, 2025にアクセス、https://forum.allaboutcircuits.com/threads/need-help-regarding-opening-asc-file-in-ltspice.186020/
11. ASCII File Formats - Details - LTwiki-Wiki for LTspice, 6月 18, 2025にアクセス、https://ltwiki.org/files/adventures_with_analog/SchBuilder/ascii_file_format.pdf
12. Formatting .op Data Labels in LTspice for Mac - EEVblog, 6月 18, 2025にアクセス、https://www.eevblog.com/forum/projects/formatting-op-data-labels-in-ltspice-for-mac/
13. Editing .asc Files in MacOS - Q&A - LTspice - EngineerZone - Analog Devices, 6月 18, 2025にアクセス、https://ez.analog.com/design-tools-and-calculators/ltspice/f/q-a/568853/editing-asc-files-in-macos
14. LTSpice/LTS0001 - Examples REV A/Educational/NonLinearTransformer.asc at master - GitHub, 6月 18, 2025にアクセス、https://github.com/Ribster/LTSpice/blob/master/LTS0001%20-%20Examples%20REV%20A/Educational/NonLinearTransformer.asc
15. LTSPICE Decks For Microelectronic Circuits, 1st Edition - McGill University, 6月 18, 2025にアクセス、http://www.ece.mcgill.ca/~grober4/SPICE/SPICE_Decks/LTspicedecks_ed1_index.html
16. Introduction to LTspice SPICE Netlists LTspice - MIT, 6月 18, 2025にアクセス、https://web.mit.edu/6.101/www/s2020/handouts/LTSpiceIntro_4.pdf
17. Add permanent component in LTSpice from asc and asy - Electronics Stack Exchange, 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/138356/add-permanent-component-in-ltspice-from-asc-and-asy
18. Components Library and Circuits - LTwiki-Wiki for LTspice, 6月 18, 2025にアクセス、https://ltwiki.org/?title=Components_Library_and_Circuits
19. How to Use LTspice Models - ROHM Semiconductor, 6月 18, 2025にアクセス、https://fscdn.rohm.com/en/products/databook/applinote/common/how_to_use_ltspice_models_an-e.pdf
20. How to convert a LTspice schematic file (.asc) to a .lib file? - EngineerZone - Analog Devices, 6月 18, 2025にアクセス、https://ez.analog.com/design-tools-and-calculators/ltspice/f/q-a/594672/how-to-convert-a-ltspice-schematic-file-asc-to-a-lib-file
21. BSS129_test.asc - LTwiki-Wiki for LTspice, 6月 18, 2025にアクセス、https://ltwiki.org/files/LTspiceIV/examples/LtSpicePlus/Discretos/Mos&Fet/BSS129_test.asc

22. simulation - How to make LTSpice sub-circuits available globally ..., 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/246406/how-to-make-ltspice-sub-circuits-available-globally
23. How to Use LTspice as a Schematic Capture Tool for FreePCB, 6月 18, 2025にアクセス、https://www.andyc.diy-audio-engineering.org/ltspice-freepcb/index.html
24. LT Spice model file question | Electronics Forum (Circuits, Projects and Microcontrollers), 6月 18, 2025にアクセス、https://www.electro-tech-online.com/threads/lt-spice-model-file-question.147057/
25. Understanding LTspice "SolarCell" model - EEVblog, 6月 18, 2025にアクセス、https://www.eevblog.com/forum/projects/understanding-ltspice-solarcell-model/
26. How do you get the ASCII model (.mod?) from a .asy (if possible) in LTSpice?, 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/727793/how-do-you-get-the-ascii-model-mod-from-a-asy-if-possible-in-ltspice
27. Where is the text for the symbol stored in LTspice? - Electrical Engineering Stack Exchange, 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/107011/where-is-the-text-for-the-symbol-stored-in-ltspice
28. LTspice-Circuit hierarchy(blocked) - Spiceman, 6月 18, 2025にアクセス、https://spiceman.net/ltspice-hierarchical-circuit-block/
29. Hierarchical Schematics and Automatic Creation of a Schematic Symbol, 6月 18, 2025にアクセス、https://ez.analog.com/cfs-file/__key/communityserver-discussions-components-files/330/LTspice-_2D00_-Hierarchical-Schematics.pdf
30. LTspice Subcircuit Tutorial | PDF | Spice - Scribd, 6月 18, 2025にアクセス、https://www.scribd.com/document/102353048/LTspice-subcircuit-tutorial
31. LTspice-Types of Voltage and Current Sources - Spiceman, 6月 18, 2025にアクセス、https://spiceman.net/ltspice-voltage-current-source/
32. Resistance is Futile - EngineerZone Spotlight - EZ Blogs ..., 6月 18, 2025にアクセス、https://ez.analog.com/ez-blogs/b/engineerzone-spotlight/posts/resistance-is-futile
33. Behavioral Sources, Parameters and Expression Evaluation - IEEE, 6月 18, 2025にアクセス、https://www.ieee.li/pdf/viewgraphs/ltspice_behavioral_sources_parameters_and_expression_evaluation.pdf
34. ltspice - If-Then-Else in a Behavioral Voltage Source - Electrical ..., 6月 18, 2025にアクセス、https://electronics.stackexchange.com/questions/530078/if-then-else-in-a-behavioral-voltage-source
35. LTspice-How to Operate Commands for a Schematic | Spiceman, 6月 18, 2025にアクセス、https://spiceman.net/ltspice-command-schematic/
36. EAGLE-LTSpice-port-to-Linux-and-Mac/ltspice.ulp at master - GitHub, 6月 18,

2025にアクセス、
https://github.com/cadsoftcomputer/EAGLE-LTSpice-port-to-Linux-and-Mac/blob/master/ltspice.ulp
37. LTSpice begginer - EEVblog, 6月 18, 2025にアクセス、
https://www.eevblog.com/forum/beginners/ltspice-begginer/
38. Use LTspice to simulate mixed continuous and sampled systems - EDN Asia, 6月 18, 2025にアクセス、
https://www.ednasia.com/use-ltspice-to-simulate-mixed-continuous-and-sampled-systems/
39. LTspice 24.1.4 incompatibilities (with LTspice 24.0.12 and earlier) - Q&A - EngineerZone, 6月 18, 2025にアクセス、
https://ez.analog.com/design-tools-and-calculators/ltspice/f/q-a/593147/ltspice-24-1-4-incompatibilities-with-ltspice-24-0-12-and-earlier
40. LT1013, LT1014 models still buggy - Q&A - LTspice - EngineerZone, 6月 18, 2025にアクセス、
https://ez.analog.com/design-tools-and-calculators/ltspice/f/q-a/594569/lt1013-lt1014-models-still-buggy
41. Import LTspice - EasyEDA Pro User Guide, 6月 18, 2025にアクセス、
https://prodocs.easyeda.com/en/import-export/import-ltspice/
42. Import LTspice schematic - KiCad.info Forums, 6月 18, 2025にアクセス、
https://forum.kicad.info/t/import-ltspice-schematic/52882
43. [question] Do you plan to support .asc files? · Issue #113 · ra3xdh/qucs_s - GitHub, 6月 18, 2025にアクセス、https://github.com/ra3xdh/qucs_s/issues/113