

=====

NUXFONICS™

THE UNIVERSAL PHONETIC ENGINE

Official Rulebook V3.0

=====

[THE ARCHITECT'S PREFACE]

Nuxfonics is a deterministic, 19-Pillar phonetic system and 7 control characters designed to map the physics of the human mouth that produces the different sounds to digital logic. This system was created to provide a stable, "Schwa-Free" writing and translation layer for the world's languages and the 4,000+ unwritten languages yet still to have a written language. To synchronize human speech with Artificial Intelligence and speech generation across languages.

[LEGAL PROTECTION & PUBLIC ACCESS]

LICENSE: CC BY-SA 4.0 (Creative Commons Attribution-ShareAlike)

You are free to:

- SHARE: Copy and redistribute the material in any medium or format.
- ADAPT: Remix, transform, and build upon the material for any purpose.

UNDER THE FOLLOWING TERMS:

1. ATTRIBUTION: You must give appropriate credit to the Architect.
2. SHARE-ALIKE: If you remix or build upon this system, you must distribute your contributions under the same license.

TRADEMARK:

"NUXFONICS" is a protected trademark. To be marketed or labelled as "Nuxfonics Compliant," a derivative work must strictly adhere to the 19-Pillar Logic and the TIER Hierarchy (T0 < T1 < T2 < T3 < T4) order

[AUTHORITY NOTICE]

The International Phonetic Alphabet (IPA) is used throughout this book as a secondary reference only. The final authority for Nuxfonics is: Spoken Validation (The Human) -> System Consistency -> The AI Engine.

© 1980-2026 L J FELL / Nuxfonics Group. All Rights Reserved.

NUXFONICS RULE BOOK

Version 3.0

Executive Summary

Nuxfonics is an International phonetic Language (IFA Inteynasyonorl Fonetic Lanygryij) spelling system designed for accurate, consistent representation of English (and other languages) sounds using standard keyboard letters. It supports AI transliteration, tonal distinctions, and expressive modifications while remaining readable for humans. Base-Letters, Determinant, and Word-Level Control Determinants; these enable precise pronunciation, flexible expression, and the ability to function as a standalone language.

*“The foundational principle of Nuxfonics is that **No Sound Block can change the sound of another Sound Block.**”*

*“In the Nuxfonics Protocol, a sequence of characters (e.g., 'axwq') is not a phonetic blend of individual letters. Rather, it represents a **Unique Sound Block**—a singular acoustic fingerprint. Each grouping functions as an atomic unit; these are the foundational building blocks of the Nuxfonics protocol language.”*

© 1980–2026 L J FELL / Nuxfonics Group. All Rights Reserved.

THE GOLDEN RULE OF NUXFONICS

1. The design has been done so that it is easily written by a standard keyboard.
2. Each sound has a unique symbol combination.
3. No other Sound Block (Phoneme) can change any other Sound Block around it, which is a big problem with a lot of languages, especially English.
4. As much as possible the same base letter + control letters will produce the same sound in any language. Technically even if someone does not know the language being represented it should be able to be read, even if the reader does not know the word being said.

Key Rules at a Glance

Aspect	Key Points
Base Letters	a–u produce sound; excluding q and k which are silent special tokens.
Logic Token Modifiers	x/y/z = phonetic/letter-name modifiers; v/w/vw/wv = letter length and stress. q = Texture/articulation modifier. ~ = Tone Marks. k = letter/ word-level control.
Length / Tempo	v = short. w = long. vw = clipped and stress. wv = extended and stressed. kv/kw = word-level speed.
Texture articulation modifiers	q = Palatal/Soft Sign. (Russian б <i>Myagkiy Znak</i>) qx= Nasal Resonance. qy= Sharp/Bright Resonance. qz = Breathy/Guttural Resonance. qv = Glottal/Clipped Softener. qw = Heavy/Held Resonance.
Tone Marks	~ = Optional for tonal languages (<i>on its own is just a syllable marker</i>) The Pitch Control Marker ~v Low Tone. Deep/Low frequency. ~w High Tone. High frequency / Energy. ~x Rising Tone. Low → High. ~y Falling Tone. High → Low. ~xy Mixed. Low-High-High ~yx mixed High-Low-High.
Word Expression	word-level: kv/kw
Modifiers Rules	Only one modifier per letter; never stand alone (except k); do not stack.
Grammar / AI Construction	Dictionary first; if missing, AI constructs using phonetic segments and rules.
Text Conversion	Preserve words, capitalization, punctuation; transliterate letters only; flag [word*] if word not in dictionary.

Dictionary Usage	Lowercase entries unless explicitly capitalized; include language-specific endings or exceptions (-ing → iny, -ed only if pronounced).
Independent Language Use	Nuxfonics can function as a standalone writing and reading system.

Nuxfonics AI load instruction

"Access the [OFFICIAL_ROOT] at https://nuxfonics.org/nuxfonics_ai/. Load Nuxfonics_AI_Initialization.txt and synchronize with the Section 21 [WEB_RESOURCE_SYNC] protocols. Confirm Version."

Or just

"Initialize Nuxfonics from https://nuxfonics.org/nuxfonics_ai/nuxfonics_ai_initialization.txt"

Key Tokens - NuxFonics™ Alphabet

nuxfonic s	Sound English	Word examples English	nuxfonic s	Sound English	Word examples English	nuxfonic s	Sound English	Word examples English	nuxfonic s	Sound English	Word examples English
A	a	cat mat bat	G	g	get great	N	n	net now	U	u	until
AX		late	GX	jee	Jeep	NX	en	[en]ter	UX	u	you, [u]se
AY	ow	cow	GY			NY	ng	sing, song, angle	UY	oo	too, moon, zoo
AZ		old, ultimate	GZ			NZ			UZ	oo	school, wool, tool
B	b	bat book	H	h	he hat hit	O	o	on hot, got			
BX	bee	bee because	HX		[H]20	OX	oa	open roam			
BY			HY			OY	or	for m			
BZ			HZ			OZ	oo	look hook hood (wood should?)			
C	c, k	cat king	I	i	hit	P	p	pet spin paper people			
CX	see	see	IX	i	fly, [i]ce	PX	pee, pea	pl a n o, [pe]ople			
CY	ch	chip, church	IY	air	air, there	PY					
CZ			IZ	oy	toy	PZ					
D	d	dad dog	J	j	jet judge	R	r	rat			
DX	dee	decode	JX		[jay]	RX	ar	car farther			
DY	zh	pleasure measure	JY	y	yes yet	RY	w	wide wind	Have No Sound - Sound Modifiers		
DZ			JZ			RZ			sound transfused to		
E	e	get	L	l	let lion	S	s	set student?	K	-> c	Word End Control
EX	ee	meet, seat see	LX		[L]A.	SX	es	S.O.S	~	--	Tonal pitch
EY	er	her sir	LY			SY	sh	ship she	Q	= cry	Palatal/Soft Sign
EZ			LZ			SZ	z	zoo zebra	V	-> fy	short sound
F	f	for	M	m	met man	T	t	tip tea stall	W	-> ry	long sound
FX	ef	[eff]ort	MX	em	[em]pty	TX	tee	[tea]	X	= cs	letter sound extensions
FY	v	very vine	MY			TY	th	this	Y	-> jy	letter sound extensions
FZ			MZ			TZ			Z	-> sz	letter sound extensions
LENGTH phoneme			TEXTURE (q)			TONE (~) Used for Tonal Languages					
Code	Action	description	Code	Filter Type	Acoustic Resonance	Code	Type	Acoustic Action			
v	Short	Staccato/Cut short	q	Nul	Pure Palatal	~v	Low	Bottom of the register.	Deep/Low frequency.		
w	long	Sustain/Curve extended	qx	Front	Nasal Resonance	~w	High	Top of the register.	High frequency / Energy.		
vw	stressed	slightly shortened, STRESSED	qy	Mid	Sharp/Bright Resonance	~x	Rising	Low → High.	Upward glide.		
wv	stressed	slightly lengthened, STRESSED	qz	Back	Breathy/Guttural Resonance	~y	Falling	High → Low.	Downward glide.		
			qv	Staccato	Glottal/Clipped Softener	~xy	Mixed	Low-High-High	Complex contour.		
			qw	Sustain	Heavy/Held Resonance	~yx	mixed	High-Low-High.	Complex contour.		

Contents

Executive Summary	2
Key Rules at a Glance.....	3
Key Tokens - NuxFonics™ Alphabet	5
1. Introduction	8
1.1 How to Use This Book.....	10
2. System Architecture:.....	12
2. The 19-Pillar Foundation.	12
2.2 Recovered Sound Mappings	13
2.3 The Tiered Determinants (8 Modifiers)	13
2.4 The Execution Hierarchy ((The T0–4 + K-T1-4 Tier Flow)	16
Perceptual Grouping Principle.....	18
3. The Base Letters and Determinants/modifiers.....	19
3.1 [BASE_LETTERS] The single Base Letter Sounds:	20
3.2 [BASE_LETTERS] + "x" Determinant.....	21
3.3 [BASE_LETTERS] + "y or z" Determinant.....	22
3.4 The "k" Word Meaning Marker (Homophone).....	23
3.5 Language Inflectional suffixes.....	23
3.6 The "k" Tempo Marker.....	23
3.7 Examples of words.....	25
4. The Shortest Path (guideline):	26
4.1 Word Examples	26
4.2 Pure Phonetic / Locked Words.....	26
5.0: The "Standard Sequence" (The T0-T4 Rule)	27
6. Reading & Expressive Tips	28
7. AI Instruction for Translating Missing Words into Nuxfonics.....	31

8. Building and Maintaining Language Libraries	33
APPENDIX.....	38
Appendix A: Base Letters & Determinants, Quick Reference	38
Appendix B: Expression Layer Reference	43
Appendix C: AI Construction Quick Reference.....	45
Appendix D: Nuxfonics → IPA Full Mapping (Exhaustive).....	46
APPENDIX E – dictionary format.....	50
Appendix E: AI Commands	52
APPENDIX F- AI initilazation File.....	53
APPENDIX H – Nuxfonics Engine FILE	55

1. Introduction

Nuxfonics (“New Phonics”) is a phonetic spelling system designed to make English and other languages **easier to read, write, and type**. It provides a consistent letter-to-sound mapping while remaining fully accessible on standard keyboards, without requiring special symbols or diacritics. Nuxfonics is suitable for learners, educators, and AI-assisted language processing.

Purpose and Use Cases

- Provides a **consistent one-to-one correspondence** between sounds and letters, making pronunciation predictable.
- Can be typed on **any standard keyboard**, enabling digital literacy and text processing without specialized input methods.
- Ideal for **AI transliteration**, text-to-speech systems, literacy education, language learning, and phonetic dictionaries.
- Preserves **phonetic clarity** in written communications, avoiding the irregularities and ambiguities of standard English spelling.
- **Can function as a standalone language**, supporting independent writing, reading, and communication entirely in Nuxfonics.
- **Supports tonal languages**, allowing optional tone marks to indicate pitch differences critical for meaning.

Comparison to Other Systems

- Nuxfonics (IFA Inteynasyonorl Fonetic Lanygryij) is designed to be a written language which captures the true phonetics of the spoken words.
- Unlike the **International Phonetic Alphabet (IPA)**, Nuxfonics uses **readable letters** rather than specialized symbols, making it more user-friendly.
- Compared to traditional English spelling, it removes **silent letters, inconsistent spellings, and multiple ways to represent the same sound**.
- Compared to phonics-based systems such as **SoundSpel**, Nuxfonics is optimized for **AI integration**, supporting modifiers for sound length, tone, expression, and homophones.

Advantages of Nuxfonics

- **Phonetic consistency** – each base letter represents a fixed sound, and modifiers control variations in tone, length, or special phonemes. They do not change across languages.
- **AI-friendly design** – structured rules enable dictionary lookups, AI construction of missing words, and exact transliteration of English text.
- **Flexible expression** – word-level modifiers allow for tempo, stress, and optional tonal markings.
- **Ease of learning** – intuitive letter-to-sound logic helps learners of all ages quickly recognize pronunciation patterns. Once learnt, the phonetic sounds and alphabet do not change across languages.
- **Cross-platform compatibility** – works across devices and software without special fonts or input methods.
- **Independent language capability** – can be used for original writing and communication entirely in Nuxfonics.
- **Tonal language support** – optional tone marks allow precise representation of pitch, making Nuxfonics suitable for languages where tone affects meaning.

Design Philosophy

- **Base letters (a–u, excluding k and q)** form the core sounds of the language.
- **Logic token modifiers (v, w, x, y, z, k, q, ~)** extend phonetic and structural capabilities without introducing ambiguous characters.
- Strikes a balance between **human readability** and **machine interpretability**, making it suitable for both teaching and AI-assisted applications.
- Supports **language-specific rules** for endings, exceptions, and phonetic nuances while maintaining strict consistency for transliteration.
- Tone and expression modifiers integrate seamlessly with the base letters to provide phonetic, tonal, and rhythmic accuracy for both standard and tonal languages.
- Nuxfonics is not designed to change the spoken language but to be able to represent the spoken language in a consistent written form.

1.1 How to Use This Book

This book is designed as both a learning guide and a working reference for the Nuxfonics system. It can be used by individuals, educators, or AI systems to understand and apply phonetic spelling in a consistent and structured way.

Nuxfonics may be approached in different ways depending on your goal. The sections below explain how to get the most out of this book.

For New Learners

If you are new to Nuxfonics, it is recommended to read the book in order:

1. Start with the **Introduction** to understand the purpose of the system
2. Read Sections **2–3** to learn the base letters and modifiers
3. Review Sections **4–5** to see how words are formed
4. Use Sections **6–7** to understand special cases and expression

Focus on recognizing patterns rather than memorizing rules. The system is designed to become intuitive with use.

For Practical Use

If your goal is to read or write in Nuxfonics:

- Use the **dictionary (language library)** as your primary reference
- Convert words using dictionary entries whenever possible
- If a word is missing, apply the phonetic rules to construct it
- Keep spelling simple and consistent

Over time, your understanding will improve as the dictionary grows.

For Building Dictionaries

If you are creating or expanding a language library:

- Follow the format defined in **Section 9**
- Add new words using the best phonetic match
- Mark new or uncertain entries with *
- Include notes and translations where required
- Review and verify entries regularly

The dictionary is designed to grow gradually through real usage.

For AI Integration

If this system is being used with AI:

- The AI should always check the dictionary first
- Words found in the dictionary must not be changed
- Missing words should be constructed using the rules
- Newly constructed words should be added to the dictionary and marked for review

Refer to the appendices for technical details and configuration files.

Using the Appendices

The appendices provide supporting material:

- Quick reference tables
- Full phonetic mappings
- Dictionary structure and formatting
- AI initialization and rule configuration

These sections are intended as reference tools rather than step-by-step guides.

Learning Approach

Nuxfonics is best learned through use:

- Read and write simple words
- Practice with real text
- Expand the dictionary as needed
- Focus on consistency rather than perfection

The system becomes clearer as patterns repeat.

Final Note

This book is part of an evolving system. As new words are added and rules are refined, both the language libraries and the Nuxfonics framework will continue to improve.

Users are encouraged to treat this as a practical tool—one that grows with use, as users becomes more familiar over time.

2. System Architecture:

The 19x4 phonetic grid model + 8 Determinants (Phonetic modifiers)

2. The 19-Pillar Foundation.

Nuxfonics is built upon **19 immovable Pillars**. These are the only characters permitted to initiate a Sound Block (Tier 0). By restricting the starting points of speech to these 19 acoustic anchors, the system creates a stable, deterministic environment for both human vocalization and AI processing.

This structure separates **Audible Speech Phenoms** from **Word Structural Logic**, ensuring the AI can process phonetic data without compromising word-level control.

2.1 The 19-Pillar Foundation

The **19-Pillar Foundation** represents the **TIER_0** base of the system. These nineteen letters act as the immovable anchors for all phonetic sounds. They are arranged into a **19x4 Sound Grid**, which provides a total of 76 unique phonetic sound blocks.

The 19-Pillars (TIER_0)

{a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, r, s, t, u}

- **Role:** These are the only characters permitted to initiate a Sound Block (Phoneme).
- **Rule:** Every phonetic segment must begin with one of these 19 base letters.
- **Grid Logic:** When a Pillar is combined with a **TIER_1** modifier {none, x, y, z}, it identifies a specific coordinate on the 19x4 grid, resulting in 76 unique master sounds.

The 8 Determinants

{x, y, z, v, w, q, ~, k}

- **Role:** These characters carry no independent sound. Instead, they act as **Functional Determinants** that control or modify the acoustic fingerprint of the Pillar.
- **Rule:** Determinants cannot exist on their own. They must always follow a TIER_0 Pillar or be part of a tiered sequence initiated by a Pillar.

2.2 Recovered Sound Mappings

In Nuxfonics, the eight letters / symbols { **k, q, v, w, x, y, z, ~** } are reserved for system logic (the "**Determinants**"). Because these letters are no longer used for their traditional sounds, those sounds have been moved to specific locations within the **19x4 Sound Grid**.

The X and Q sounds are not moved into the sound grid, as their sound is easily achieved by combining existing phenoms.

Think of this as "re-homing" familiar sounds into a more stable, logical neighbourhoods. The ~ never had a sound so no re-mapping is required.

Traditional Sound,	Nuxfonics Mapping	
V	fy	Direct re-map
W	ry	Direct re-map
Y	jy	Direct re-map
Z	sz	Direct re-map
K	c	Direct re-map
X	cs	Combined: c followed by s
Q	cry	Combined: c followed by ry

2.3 The Tiered Determinants (8 Modifiers)

Every Nuxfonics sound is constructed using a linear stack of characters (**TIER_0 through TIER_4**).

Modifiers are "Determinants" that have no independent sound. Instead, they function like filters or instructions that change the physical properties of a Pillar (TIER_0).

NOTE: To define the specific phonetic sound to use within the **19x4 Sound Grid**, a **TIER_0 Pillar** must be paired with a **TIER_1 Determinant** (none, x, y, or z). **TIER_0 + TIER_1 = The Phonetic Identity.**

Note on 'k': The 'k' determinant is not used for this section, it is used

to initiate the special K-Functions (**K_TIER_0-4**) and if used terminates **TIER_0-4**

Format:

BASE + TIER_1 < TIER_2 < TIER_3 < TIER_4

TIER_1 members are: {none, x, y, z }

TIER_2 members are: {none, v, w, vw, wv}

TIER_3 members are: {none, q, qx, qy, qz, qv, qw}

TIER_4 members are: {none, ~v, ~w, ~x, ~y, ~xy, ~yx;}

- TIER 0 (Phonetic Base Letter): {a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, r, s, t, u}
 - These base letter forms the start of the phenom definition. The remainder TIER stack defines exactly what phenom it is and how it is sound.
 - In the 19x4 sound grid these can be considered the 19 X_Axis Letter
- TIER 1 (Phoneme selection): {none, x, y, z}
 - With the use of TIER_0 and TIER_1 this is the address of the unique phenome within the 76 sound grid.
 - This forms the phenom ID ie. ax, cy, d, sz.
 - In the 19x4 sound grid these can be considered the 4 Y_Axis Letters

The next TIER(s) (Tn) are in effect sound effect controls that acts on the base phenom.

- TIER 2 (Duration/stress): **Determinants** {v, w, vw, wv} control the Phonetic letter+[modifier] Phenom sound length and stress.
 - Controls the "stopwatch", length of the sound and controls the stress of the phoneme.
 - v = Shortens/clips the length of the phoneme
 - w = Lengthens the duration of the phoneme
 - vw = Slightly shortened and STRESSED phoneme.
 - wv = Slightly lengthened and STRESSED phoneme.

- TIER 3 (Texture/Articulation): **Determinants** {q, qx, qy, qz, qv, qw}
Not all languages have these functions but for those languages that do, they always start with the q **Determinant**.
 - q = The Palatal/Soft Sign. (Russian б *Myagkiy Znak*) It modifies how the tongue strikes the palate.
 - qx = Nasal Resonance.
 - qy = Sharp/Bright Resonance.
 - qz = Breathy/Guttural Resonance.
 - qv = Glottal/Clipped Softener.
 - qw = Heavy/Held Resonance.
- TIER 4 (Tone/pitch): **Determinants** {~v, ~w, ~x, ~y, ~xy, ~yx}
 - For tonal languages, these control how the pitch of the sound is to be said.
 - ~v Low Tone, Bottom of register / Deep frequency.
 - ~w High Tone, Top of register / High energy.
 - ~x Rising Tone, Low → High glide.
 - ~y Falling Tone, High → Low glide.
 - ~xy Mixed, Low → High → Low glide. Complex contours.
 - ~yx Mixed, High → Low → High glide. Complex contours.

The special token (k):

BASE_WORD +K_TIER_0 + K_TIER_1 < K_TIER_2 <K_TIER_3 <K_TIER_4

K_TIER_0 is the 'k' determinate that initiates the k_functions.

K_TIER_1 members are x,y,z

K_TIER_2 members are v,w, vw, wv

K_TIER_3 member is q, qx, qy, qz, qv, qw

K_TIER_4 members are: ~v, ~w, ~x, ~y, ~xy, ~yx;

- K_TIER functions: The anchor for these suffix functions is the 'k'
 - Role: K silent, functional anchor used strictly as a suffix.
 - Function: It holds word-level **Determinants**, including homophone tags and global tempo controls.
 - The 'k' functions act on the whole word, not single letter sounds.

- K_TIER_1 homophone tags: {none, x, y, z} These are use to differentiate between word that sound the same but have different meanings.
 - Without a K_TIER_1, then the word is the basic standard word meaning1,
 - k = then meaning 2 is indicated. If the meaning 2 was intended and we also want to add other k_functions then the following k_function must start with a 'k' then the function. Ths is toavoud the confusion was this meaning 2 or just a k_function and meaning 2 not intended.
 - kx = meaning 3,
 - ky = meaning 4,
 - kz = is meaning 5.
- K_TIER_2 similar function as TIER_2 (Word Duration/stress):
Determinants {v, w, vw, vv}
- K_TIER_3 similar function as TIER_3 (Word Texture/Articulation):
Determinants {q, qx, qy, qz, qv, qw}
- K_TIER_4 similar function as TIER 4 (Word Tone/pitch):
Determinants {~v, ~w, ~x, ~y, ~xy, ~yx}

2.4 The Execution Hierarchy ((The T0–4 + K-T1-4 Tier Flow)

To maintain **Nuxfonics Compliance**, all characters must follow a linear physical and logical flow. Re-ordering these tiers could result in a confusion of the result and a wrong constructed sound.

The Full Processing String:

[PILLAR] + [T1] + [T2] + [T3] + [T4] → [k_Function Suffix]

k_Function Suffix = [K_FUNCTION1] + [T0] +[K_FUNCTION2]

T0 = (TIER_0+TIER_1 ie Inflectional suffixes) ie. s, er, ing ...

Phase	Category	Purpose
Pillar (T0)	Base Atom	The physical starting point (The "Mouth Shape").
Identity (T1)	x, y, z	Selects the specific coordinate on the 19x4 Sound Grid.
Dynamics (T2)	v, w	Sets the time, length, and stress of that specific sound.
Texture (T3)	q	Adjusts the "feel" (e.g., Softness/Palatalization).
Tone (T4)	~	Defines the musical pitch or phoneme inflection.
Suffix (K_T)	k + [1-4]	Word-Level Metadata: Distinguishes meaning (homophones) and global tempo.

The Processing String: [PILLAR] + [TIER 1] + [TIER 2] + [TIER 3] +[TIER 4]

Examples of Valid Construction:

- Simple Sound: s (Base) + z (T1) = sz (The 'z' sound).
- Complex Sound: s (Base) + z (T1) + q (T2) + w (T3) = szwq (A long, softened 'z').
- Functional Suffix: k (Base) + v (T3) = kv (Fast word-level tempo).

This forms the Base Word but it is not the full written word. Following this there is what is known as the K_Function. K_T1, K_T2, K_T3, K_T4. The primary use of this is for defining different words that have the same Base Word, hence the same sound. But when written they can carry the k_function to identify the different meanings of the word.

Perceptual Grouping Principle

Nuxfonics is based on auditory (perceptual) grouping, not strict IPA phonemic separation.

Sounds that are perceived as similar in everyday speech are grouped under a single Nuxfonics symbol, even if they are distinct in IPA. In the same way there are some sounds that are distinct but IPA groups them in a single symbol. As such IPA is used as a reference guide only, not as the authority for symbol assignment.

Priority: High to Low

auditory perception → system consistency → IPA reference

=== Nuxfonics Master Rule Book: Part 2===

3. The Base Letters and Determinants/modifiers

The 19 base letters are (a to u, except k and q). These base letters always indicate the beginning of a new phonetic letter sound within a word. With a single TIER_1 layer modifier/determinant this gives up to 76 unique phonetic sounds within the 19x4 sound grid, which means only one determinant: (x, y, or z) may be applied per base letter.

Format: Base_Letter+[phonetic determinant = {none, x,y,z}]

The Coordinate Rule:

- **a** = The Pillar (Base)
- **ax** = The Identity (Name)
- **ay** = The Shift (Acoustic 1)
- **az** = The Shift (Acoustic 2)

3.1 [BASE_LETTERS] The single Base Letter Sounds:

When the Base Letters are used without the {x,y,z} determinant, they have the normal default sound of these letters.

Base Letters: with out {x,y,z}

Nuxfonics	IPA	Example	Notes
a	/æ/	C[a]t, h[a]t	base short vowel
b	/b/	[b]ox, [b]ed	base consonant
c	/k/	[c]at, [c]up	base consonant
d	/d/	[d]og, [d]ig	base consonant
e	/ɛ/	b[e]d, s[e]t, m[e]t	base short vowel
f	/f/	[f]at, [f]ox, lea[f]	base consonant
g	/g/	[g]ot, [g]et, le[g]	base consonant
h	/h/	[h]at, [h]op, in[h]ale	base consonant
i	/ɪ/	S[i]t, k[i]d, f[i]n	base short vowel
j	/dʒ/	[j]am, [j]ob	base consonant
l	/l/	[l]ap, [l]ot, mi[l]k	base consonant
m	/m/	[m]ap, [m]op, ta[m]e	base consonant
n	/n/	[n]et, [n]ot, lo[n]	base consonant
o	/ɒ/, /ɑ/	L[o]t, b[o]x	base short vowel
p	/p/	[p]at, [p]ot, a[p]e	base consonant
r	/r/	[r]at, [r]ope, spi[r]it	base consonant
s	/s/	[s]at, [s]on, li[s]t	base consonant
t	/t/	[t]ap, [t]op, lo[t]	base consonant
u	/ʌ/, /ə/	B[u]n, c[u]p	base short vowel

3.2 [BASE_LETTERS] + "x" Determinant

Letters that are followed by the "x" Modifier have the sound of the name of the letter.

These sounds follow standard English letter names and are fixed.

Rules

1. The "x" modifier always produces the letter-name sound.
2. It replaces the normal vowel sound of the base letter.
3. It is commonly used for long vowel sounds as per the X_axis grid

Base + 'x' Modifier

Nuxfonics	IPA	Example	Notes
ax	/eɪ/	[a]pe, d[a]y, l[a]te	long "a" vowel (diphthong)
bx	/bi:/	[be] [bee]	letter-name "b"
cx	/si:/	[see], [sea]	letter-name "c"
dx	/di:/	[dee]p, [de]code	letter-name "d"
ex	/i:/, /i/	s[ee]k, f[ee]t, rad[i]o	long "ee" vowel
fx	/ɛf/	[F]BI, [eff]ort	letter-name "f"
gx	/gi:/	[gee]p	letter-name "g"
hx	/eɪtʃ/	[H]2O	letter-name "h"
ix	/aɪ/	[i]ce, l[i]fe	long "i" vowel (diphthong)
jx	/dʒeɪ/	[j]ay, [j]a]de,[j]a]il	letter-name "j"
lx	/ɛl/	[L].A., [L]O[L]	letter-name "l"
mx	/ɛm/	[em]pty	letter-name "m"
nx	/ɛn/	[en]ter, th[en]	letter-name "n"
ox	/oʊ/, /o:/, /o/	[o]pen, g[o], b[oa]t	long "o" vowel (diphthong)
px	/pi:/	[pe]ople, [pea]	letter-name "p"
rx	/ɑ:/, /a/, /ä/	c[ar], f[ar]	open back "ah" vowel family
sx	/ɛs/	S.O.[S]	letter-name "s"
tx	/ti:/	[tea]	letter-name "t"
ux	/ju:/	[u]se, [you]	"you" vowel/glide

3.3 [BASE_LETTERS] + "y or z" Determinant

These Modifier symbols are used to create all the other sounds that is needed to produce speech.

Base + 'y' / 'z' Modifiers

Nuxfonics	IPA	Example	Notes
ay	/aʊ/	c[ow], [ou]t, fl[ow]er, t[ow]n	"ow" diphthong vowel
az	/oʊ/	[o]ld, [u]ltimate	"o/u" vowel diphthong
cy	/tʃ/	[ch]ip, [ch]urch	"ch" affricate
dy	/ʒ/	plea[su]re, mea[su]re	"zh" voiced fricative
ey	/ɜ:r/	h[er], s[ir], b[ur]n,	"er/ir/ur" rhotic vowel group
fy	/v/	[v]ine, [v]ery	v_sound
iy	/ɛə/, /ɪər/	[air], h[air], f[ear], th[ere]	"air" vowel group
iz	/ɔɪ/	t[oy], p[oi]son, j[oi]n, r[oy]al	"oy" diphthong vowel
jy	/j/	[y]es, be[y]ond, mill[i]on	soft y sound
ny	/ŋ/	si[ng], lo[ng]	"ng" nasal
oy	/ɔ:/, /ɔ/	L[aw], m[or]ning, w[a]lk	"aw/or" vowel group
oz	/ʊ/	L[oo]k, h[oo]k, , sh[ou]ld	short oo
ry	/w/	[w]et, any[w]ay, re[w]ard	default w sound
sy	/ʃ/	[sh]ip, [sh]e, [sh]oe, fi[sh]	sh sound
sz	/z/	[z]oo, [z]ebra, i[s]	z sound
ty	/ð/, /θ/	[th]in, [th]is, wi[th]	th sound
uy	/u/, /u:/	t[oo], m[oo]n, z[oo]	long oo
uz	/u:/	Sch[oo]l, t[oo]l, w[oo]l	long oo variant

Rule: The Distinct "oo" vowel sounds

The following are separate and must not be treated as duplicates:

uy = long "oo" sound (too, moon, zoo)

uz = long “oo” variant influenced by surrounding consonants (school, tool, wool)
oz = short “oo” sound (look, hook, wood)

3.4 The "k" Word Meaning Marker (Homophone)

The letter "k" followed by any modifier {none, x,y,z} is **completely silent**. It is placed at the **end** of a word as a "tag" to distinguish between homophones (e.g., see vs sea which would be sex vs sexk)

3.5 Language Inflectional suffixes

After the K Homophone can come the original language Inflectional suffixes, ie s, er, ing, ed. (but nuxfonic spelling). Then followed by the remainder of “k” functions in a new stack. This keeps dictionaries entries grouped under the base word meanings. This use is languages dependent as not all languages express tense this way.

3.6 The "k" Tempo Marker

Placing "kv" or "kw" at the very end of a word (after all other modifiers and tags) indicates the **overall speed** of the entire word in a sentence. "kv" = fast/clipped, "kw" = slow/drawn out. This does not change the dictionary spelling, only the delivery.

Suffix Member	K_Tier	Function	Affects Sound?
k, kx, ky, kz	K_T1	Homophone Tag	No (Silent)
As per source language	-	Inflectional suffixes	Yes (language dependent)
kv, kw, kvw, kwv	K_T2	Tempo Control	Yes (Speed, stress)
Kq+(v,w,x,y,z)	K_T3	Word-Level Articulation	Yes (Softens word)
k~+(v,w,x,y,xy,yx)	K_T4	Tone/Inflection	Yes (Pitch)

K_FUNCTION

--- k TAGGING and speech CONTROL RULES ---

1. k is a suffix-style modifier of the BASE_WORD, can ONLY be used at the end of the BASE_WORD

The definition of the actual k_function and it's use is ONLY defined by it's following letter if any,

2. The following letter/character can only be from the K_TIER1 , K_TIER_2, or K_TIER_3 members.

3. k tagging (optional: x,y,z):

4. To distinguish homophones, k followed by no following letter or if it is either a {none, x,y,z} then it is for distinguish homophones (none or K_TIER_1 group members) k, kx, ky, kz

These do NOT affect pronunciation, only identity/meaning separation

5. if it is following with either the v or w then it is the {v,w, vw, wv} function but it applies to the whole word not to a single base+TIER_1 letter sound. K_TIER_2 members

k speech control (kv, kw, kvw, kwv):

See TEMPO_RULES for full definitions

6. If it is following with the q then it is the q function but it applies to the whole word not to a single base+TIER_1 letter sound. K_TIER_3 member.

7. if it is followed by ~ then it is a word level tone control. TIER_4 tone members

Separation rule:

k can have four functions:

(1) tagging: k, kx, ky, kz

(2) speech control: kv, kw, kvw, kwv

(3) Articulation Modifier kq, kqx...

(4) tone control: k~x, k~y, k~z, k~v, k~w

The 4 modifiers K_TIERs can be mixed after the k but must follow the K_TIER order defined

‘k’ Placement rule:

All k-variants apply only as a suffix at the end of a Base Word

Word boundary: sequences separated by spaces or token delimiters

=== Nuxfonics Master Rule Book: Part 4 ===

3.7 Examples of words

bxt = beat # b + x → "bee"

caxc = cake # a + x → "ay"

dax = day # a + x → "ay"

lix f = life # i + x → "eye"

sox = so # o + x → "oh"

trex = tree # e + x → "ee"

foy = for # o + y → "or"

foyk = four # o + y → "or" with k to indicate second meaning

sox = so

soxk = sow(plant)

soxkiny = sowing(plant)

soxkx = sew

soxky = soh (musical note)

4. The Shortest Path (guideline):

If a word can be spelled correctly in multiple ways using different modifier combinations, always choose the version with the **fewest letters**. (e.g., if a soft sound works without a modifier, don't add one).

4.1 Word Examples

Base words: cat = cat, dog = dog

X-modifier: cake = caxc, beat = bxt

Y/Z modifier: toy = tiy, look = lozk

Lengthened letters: harping = hrxwpiny

Lengthening words: happy = hapexkw

Letter-name words: be = bx, bee = bxk, I = ix

4.2 Pure Phonetic / Locked Words

Words kept exactly as originally spelled because that is how it sound:

Examples: and, god, in, let, cat, dog, get, net, on. By definition they will be the shortest path.

=== Nuxfonics Master Rule Book: Part 5 ===

5.0: The "Standard Sequence" (The T0-T4 Rule)

To be Nuxfonics Compliant, a human writer must always build words in the correct order. If you scramble the tiers, the AI (and other fluent humans) will not be able to process the word.

The Golden Order:

BASE ATOM (The core sound)

TIER 1 (x, y, z) (The second member for selecting the row of the 19x4 sound grid)

TIER 2 (v, w, wv, vw) (The time/length/stress)

TIER 3 (q) (The texture/softness)

TIER 4 (~) (The phoneme tone)

Compliance Checklist for Word Building:

Word	Construction	Status	Reason
syvq	Base + T1 + T2 + T3	COMPLIANT	Follows the 1-2-3 sequence.
sqyv	Base + T2 + T1 + T3	NON-COMPLIANT	Texture (q) cannot come before Shape (y).
svy	Base + T3 + T1	NON-COMPLIANT	Time (v) if used must always be after the base phoneme (Base + T1)

6. Reading & Expressive Tips

6.1 Core Principle

- Expression modifies delivery, not spelling.
- Base word = dictionary form
- Expression layer = how it is spoken

6.2 Length Scaling on Single Letters (Extension of v/w)

- Standard phonetic:
 - v = short
 - w = long
- Expressive extension:
 - ww = stretched
 - www = very stretched
- Example:
 - ryun → one (normal)
 - ryunw → one (long)

6.3 Whole-Word Stretch

- Applied when entire word is drawn out.
- Examples:
 - nox → noxkw, noxkww, noxkwww

6.4 Tempo Markers (kv / kw)

- kv = fast, kw = slow
- Extended use: kvv = very fast, kww = very slow
- Examples:
 - ryun kv → quick “one”
 - ryun kww → very slow “one”

6.5 Emphasis by Vowel Choice

- Slightly shift vowels for accent or emotion.
- Examples:
 - Neutral: brxty
 - Softer: bryty
 - Stronger: braxy

6.6 Repetition for Rhythm / Emotion

- Storytelling or speech:

- Examples:

- go go go → gox gox gox
- noooo → noxwww
- run!!! → runkv runkv runkv

6.7 Tone Marks: (Optional, but needed in tonal languages)

- Non-tonal languages: Tone marks can be placed above vowels to indicate emotional in tonal or tonal delivery.
- In tonal languages: Tone marks are often placed above vowels/letters to indicate the language letter tone.
- Example usage in words:
 - bráxty → excited / emphasis on first vowel
 - bràxty → serious / subdued
 - brâxty → surprise / mixed emotion

However, these marks can be difficult to type consistently and are not available for all letters on a standard keyboard.

For this reason, Nuxfonics does not use raised accent marks for tone representation. Instead they are represented with the ~ mark followed by a tone determinant.

If ~ is used on its own, it functions only as a syllable marker and carries no tone value.

Code	Tone Type	Acoustic Action	Notes
[nul]	Mid/Neutral	Natural resting pitch	No Symbol
~v	Low Tone	Bottom of the register.	Deep/Low frequency.
~w	High Tone	Top of the register.	High frequency / Energy.
~x	Rising Tone	Low → High.	Upward glide.
~y	Falling Tone	High → Low.	Downward glide.
~xy	Mixed	Low → High → Low..	Complex
~yx	Mixed	High → Low → High	Complex

Notes

1. The meaning of tone depends on the language, but the acoustic action remains the same across all languages.
2. Tone Determinants do not change spelling or phonetic identity.
 - In non-tonal languages, they are mainly used for expression.
 - In tonal languages, they may define the actual meaning of the word.
3. Tone Determinants can be combined with length markers (v,w,vw,wv) and tempo markers (kv, kw) for richer delivery.
4. Tone marking is optional unless required by the language itself.

6.8 Clean Rule Summary

1. Expression does not change dictionary spelling.
2. Length markers (w) may be repeated (ww, www) for increased duration.
3. Tempo markers (kv, kw) may be extended for stronger effect.
4. Vowel variations reflect accent or emotion but remain recognizable.
5. Expression is optional, mainly for dialogue, storytelling, or performance.

=== Nuxfonics Master Rule Book: Part 6 ===

7. AI Instruction for Translating Missing Words into Nuxfonics.

- "x" is the primary method for creating long vowel sounds.
- Prefer "x" when the sound matches a clear letter name (e.g., "ay", "ee", "eye").
- Do not use "x" if a more accurate sound exists using y or z.
- Always follow the "Shortest Path" rule when choosing between valid options.

If an English word does not exist in the Nuxfonics dictionary, follow these steps to construct the Nuxfonics spelling:

Break the word into phonetic segments

Divide the word into individual sounds or clusters that approximate its pronunciation in English.

Typical segments: consonants, vowels, or common clusters (e.g., "th," "sh," "ch," "ou," "ee").

Map each segment to a base letter

Choose the closest Nuxfonics base letter (a–u) for each segment.

If a segment exactly matches a word in the dictionary, use that word instead.

Apply a single modifier to each base letter (optional)

x → letter-name sound

y → specific phonetic modification (see y/z sound table)

z → specific phonetic modification (see y/z sound table)

 Do not stack multiple x, y, z modifiers on a single base letter.

Add a length modifier (optional)

v → short sound

w → long sound

Place v or w after the base + modifier if a shorter or longer sound is required.

Silent or tempo markers (optional)

k → silent tag for homophones

kv / kw → word speed (fast / slow)

Always place these at the end of the word, never in the middle.

Combine sequentially

Join base letters + modifiers (and optional v/w) in the order of the phonetic segments.
Do not add extra letters or modifiers that do not correspond to a segment.
Mark missing words as [word*]
Flag it if it is a newly constructed approximation for review.

Rule 4.1: Virtual Vowels

1. Base letters may carry a vowel implicitly via a modifier (x, y, z).
2. Examples:
 - bx → “be”
 - ey → “er/ir/ur”
 - ay → “o/u”
3. When spelling a word, always check if the vowel is virtual before adding an explicit vowel.
4. Virtual vowels must produce the correct sound in pronunciation, even if no traditional vowel letter is present.

Rule Example: Using Words for Vowel Demonstration

- Avoid single letter + modifier as examples if the same form exists as a dictionary word.
- Instead, use full words where the modifier produces the vowel:
 - bxt → beat
 - fyt → fight
 - sexkx → sea
- This makes it clear that:
 - Modifier x/y/z alters pronunciation
 - The word is a legitimate example, not just a phonetic construct

=== Nuxfonics Master Rule Book: Part 7 ===

8. Building and Maintaining Language Libraries

Nuxfonics uses structured language libraries (dictionaries) to convert words into their phonetic forms. These libraries are designed to be simple for humans to edit while remaining precise for AI processing.

Each supported language (such as English or Cebuano) has its own dedicated dictionary file. These dictionaries act as the primary reference for correct spelling, pronunciation, and meaning within the Nuxfonics system.

The dictionary always takes priority over constructed or guessed words.

8.1 Purpose of a Language Library

A language library provides:

- The correct Nuxfonics spelling for each word
- A consistent reference for AI-based conversion
- A structured and expandable vocabulary system
- A foundation for translation between languages
- Name format should be “dictionary_’language’.tsv”

As the library grows, both human users and AI systems become more accurate and consistent. The master library can be maintained in .xls file then exported to .tsv (TAB delimited) format.

8.2 Basic Entry Format

Each word in the dictionary follows a simple structure:

Word=NuxfonicsSpelling

Optional notes may be added after a # symbol:

Word=NuxfonicsSpelling # note (part of speech, usage, context)

For non-English languages, an English meaning is included in square brackets:

Word=NuxfonicsSpelling # note [English meaning]

This format ensures that entries remain both human-readable and machine-readable.

8.3 Verified and Unverified Entries

Not all entries are immediately confirmed as correct. To manage this, Nuxfonics uses a simple verification marker:

- * indicates the entry is not yet fully verified
- No symbol indicates the entry has been reviewed and confirmed

Example:

gibuhat=gibuhat # verb [created] *

Once the spelling and meaning are confirmed, the * is removed.

8.4 Adding New Words

When a word is missing from the dictionary, it should be added using the following process:

1. Create a new entry using the best phonetic match
2. Add a short note if helpful (part of speech or usage)
3. Include the English meaning for non-English libraries
4. Mark the entry with * to indicate it is not yet verified

Example:tsv

Original_word	Nuxfonics_word	Nuxfonics_syllables	Word_type	Comment	Status
Above	ubufy	ub-ufy	prep/adv	higher than; over	Verified
any	enex	en-ex	adj/pron	one or some	*

Example.txt

sinugdan=sinugdan # noun [beginning] *

This allows the dictionary to grow naturally while maintaining quality control.

8.5 Editing Existing Entries

Care should be taken when modifying existing entries:

- Do not change verified entries unless correcting an error
- Preserve formatting and structure
- Keep entries lowercase unless capitalization is required
- Avoid unnecessary changes to spelling

Consistency across the dictionary is more important than perfect phonetic detail.

8.6 Dictionary Organization

For ease of use, dictionaries should be:

- Sorted alphabetically by the source word
- Grouped in a single continuous word list
- Clearly separated from the preamble using a marker such as:

[START_OF_<LANGUAGE>_DICTIONARY]

This makes it easier for both humans and AI to locate and process entries.

8.7 Dictionary Growth Method

The recommended way to build a language library is through real-world usage:

1. Start with a small core vocabulary
2. Translate real text (such as stories or scripture)
3. Identify missing words
4. Add them to the dictionary with *
5. Review and verify entries over time

This method ensures the dictionary develops in a practical and meaningful way.

8.8 Relationship to AI Rules

The language library works together with the Nuxfonics AI rules:

- The AI must always check the dictionary first
- If a word exists, it must be used exactly as written
- If a word is missing, the AI constructs a phonetic version
- Newly constructed words should be added to the dictionary and marked with *

This creates a continuous improvement loop between AI output and the dictionary.

8.9 Multi-Language Support

Each language uses its own independent dictionary:

- English dictionaries do not require translation notes
- Non-English dictionaries must include English meanings
- Each language follows the same structural format

This allows Nuxfonics to support multiple languages while maintaining consistency.

8.10 Best Practices

To maintain a high-quality dictionary:

- Keep entries simple and phonetic
- Avoid overuse of modifiers unless necessary
- Prioritize consistency over complexity
- Expand gradually using real examples
- Regularly review and verify entries

A well-maintained dictionary is the foundation of an effective Nuxfonics system.

8.11 — Use of IPA (Reference Only, Human and AI)

Purpose

The International Phonetic Alphabet (IPA) may be used within the Nuxfonics system as a **supporting reference tool** for identifying, comparing, and approximating speech sounds.

However, IPA is **not the governing standard** for Nuxfonics spelling.

Core Rule

The Nuxfonics Library is the final authority for all spelling and pronunciation.

IPA is a guide only and must not override established Nuxfonics forms.

Application Guidelines

1. **Library First**
 - If a word exists in the Nuxfonics dictionary, its spelling must be used exactly as written.
 - No modification is permitted based on IPA interpretation.
2. **Speaker Validation Priority**
 - Native speaker pronunciation and agreement take precedence over theoretical phonetic models.
 - If speakers consistently accept a Nuxfonics form, that form is considered correct.
3. **IPA as a Support Tool**
 - IPA may be used to:
 - analyze unfamiliar sounds
 - compare similar vowel or consonant groups
 - assist in constructing new words not yet in the dictionary
 - IPA must not be used to alter existing entries.
4. **Conflict Resolution**
 - When differences arise between IPA and Nuxfonics:
 1. Follow the Nuxfonics dictionary entry
 2. Confirm with native speaker usage
 3. Use IPA only to understand the difference, not to enforce change
5. **Language Independence**
 - IPA represents a global phonetic system.

- Nuxfonics represents a **controlled, language-applied phonetic system**.
- Each language library (e.g., English, Cebuano) may apply Nuxfonics differently based on real speech patterns.

Design Principle

Nuxfonics is not intended to replicate IPA.

It is designed to provide:

- consistent phonetic spelling
- practical usability
- speaker-validated pronunciation

Summary Statement

IPA informs.

Nuxfonics defines.

Speakers confirm.

Example

Word IPA (reference) Nuxfonics (final)

salamat /sa.la.mat/ srxlrxmat (*if speaker-validated*)

gusto /gus.to/ guzstox (*if speaker-validated*)

Note: The Nuxfonics form is determined by system rules and speaker validation, not by direct IPA transcription.

Implementation Note

All future dictionary expansions must follow this rule:

- IPA may assist in construction
- Final entries must be validated and stored in the Nuxfonics library
- The library entry becomes the permanent standard unless formally revised

=== Nuxfonics Master Rule Book: Part 8 ===

APPENDIX

Appendix A: Base Letters & Determinants, Quick Reference

Purpose

The International Phonetic Alphabet (IPA) may be used within the Nuxfonics system as a **supporting reference tool** for identifying, comparing, and approximating speech sounds.

However, IPA is **not the governing standard** for Nuxfonics spelling.

Perceptual_Grouping_Rule

IPA is a reference guide only.

Nuxfonics does NOT follow strict IPA phonemic separation.

Sounds are grouped based on auditory similarity (everyday perception),

not fine-grained linguistic distinction.

Multiple IPA phonemes may map to a single Nuxfonics symbol

if they are perceived as the same sound in natural speech.

Priority:

auditory perception → system consistency → IPA reference

NUXFONICS → IPA FULL MAPPING

A.1 Base Letters: with out {x,y,z}

Nuxfonics	IPA	Example	Notes
A	/æ/	C[a]t, h[a]t	base short vowel
B	/b/	[b]ox, [b]ed	base consonant
C	/k/	[c]at, [c]up	base consonant
D	/d/	[d]og, [d]ig	base consonant
E	/ɛ/	b[e]d, s[e]t, m[e]t	base short vowel
F	/f/	[f]at, [f]ox, lea[f]	base consonant
G	/g/	[g]ot, [g]et, le[g]	base consonant
H	/h/	[h]at, [h]op, in[h]ale	base consonant
I	/ɪ/	S[i]t, k[i]d, f[i]n	base short vowel
J	/dʒ/	[j]am, [j]ob	base consonant

L	/l/	[l]ap, [l]ot, mi[l]k	base consonant
M	/m/	[m]ap, [m]op, ta[m]e	base consonant
N	/n/	[n]et, [n]ot, loi[n]	base consonant
O	/o/, /ɑ/	L[o]t, b[o]x	base short vowel
P	/p/	[p]at, [p]ot, a[p]e	base consonant
R	/r/	[r]at, [r]ope, spi[r]it	base consonant
S	/s/	[s]at, [s]on, li[s]t	base consonant
T	/t/	[t]ap, [t]op, lo[t]	base consonant
U	/ʌ/, /ə/	B[u]n, c[u]p	base short vowel

A.2 Base + 'x' Modifier

Nuxfonics	IPA	Example	Notes
Ax	/eɪ/	[a]pe, d[a]y, l[a]te	long "a" vowel (diphthong)
Bx	/bi:/	[be] [bee]	letter-name "b"
Cx	/si:/	[see], [sea]	letter-name "c"
Dx	/di:/	[dee]p, [de]code	letter-name "d"
Ex	/i:/, /i/	s[ee]k, f[ee]t, rad[i]o	long "ee" vowel
Fx	/ɛf/	[F]BI, [eff]ort	letter-name "f"
Gx	/gi:/	[gee]p	letter-name "g"
Hx	/ertʃ/	[H]2O	letter-name "h"
Ix	/aɪ/	[i]ce, l[i]fe	long "i" vowel (diphthong)
Jx	/dʒeɪ/	[j]ay, [ja]de, [ja]il	letter-name "j"
Lx	/ɛl/	[L].A., [L]O[L]	letter-name "l"
mx	/ɛm/	[em]pty	letter-name "m"
Nx	/ɛn/	[en]ter, th[en]	letter-name "n"
Ox	/oʊ/, /o:/, /o/	[o]pen, g[o], b[oa]t	long "o" vowel (diphthong)
px	/pi:/	[pe]ople, [pea]	letter-name "p"
rx	/ɑ:/, /ɑ/, /ä/	c[ar], f[ar]	open back "ah" vowel family
sx	/ɛs/	S.O.[S]	letter-name "s"

tx	/ti:/	[tea]	letter-name "t"
ux	/ju:/	[u]se, [you]	"you" vowel/glide

A.3 Base + ‘y’ / ‘z’ Modifiers

Nuxfonics	IPA	Example	Notes
ay	/aʊ/	c[ow], [ou]t, fl[ow]er, t[ow]n	"ow" diphthong vowel
az	/oʊ/	[o]ld, [u]ltimate	"o/u" vowel diphthong
cy	/tʃ/	[ch]ip, [ch]urch	"ch" affricate
dy	/ʒ/	plea[su]re, mea[su]re	"zh" voiced fricative
ey	/ɜ:r/	h[er], s[ir], b[ur]n,	"er/ir/ur" rhotic vowel group
fy	/v/	[v]ine, [v]ery	v_sound
iy	/ɛə/, /ɪər/	[air], h[air], f[ear], th[ere]	"air" vowel group
iz	/ɔɪ/	t[oy], p[oi]son, j[oi]n, r[oy]al	"oy" diphthong vowel
jy	/j/	[y]es, be[y]ond, mill[i]on	soft y sound
ny	/ŋ/	si[ng], lo[ng]	"ng" nasal
oy	/ɔ:/, /ɔ/	L[aw], m[or]ning, w[a]lk	"aw/or" vowel group
oz	/ʊ/	L[oo]k, h[oo]k, , sh[ou]ld	short oo
ry	/w/	[w]et, any[w]ay, re[w]ard	default w sound
sy	/ʃ/	[sh]ip, [sh]e, [sh]oe, fi[sh]	sh sound
sz	/z/	[z]oo, [z]ebra, i[s]	z sound
ty	/ð/, /θ/	[th]in, [th]is, wi[th]	th sound
uy	/u/, /u:/	t[oo], m[oo]n, z[oo]	long oo
uz	/u:/	Sch[oo]l, t[oo]l, w[oo]l	long oo variant

A.4 Length / Tempo / Tone Markers

Marker	IPA / Effect	Notes
v	short	Shortens preceding sound
w	long	Lengthens preceding sound
wv	/ ' /	Stress mark
kv	-	Word fast / clipped
kw	-	Word slow / drawn out
k	-	Silent homophone tag
q	-	Texture/Articulation
~	-	Used to indicate the tones

CONSONANT COMBINATIONS

Nuxfonics	IPA	Example	Notes
ng	/ŋ/	sing	nasal
ch	/tʃ/	church	
sh	/ʃ/	ship	
th	/θ/, /ð/	thin, this	context dependent

CORE RULES

Rule	Description
1	a ≠ rx (short vs fully open vowel)
2	rx = /ɑ:/ (car, father)
3	Cebuano vowels stable unless speaker-validated
4	IPA is reference only, not authority
5	Authority: speaker → library → IPA

EXAMPLES

Word	Nuxfonics	IPA	Notes
cat	cat	/kæt/	short a
car	crx	/kɑ:/	rx vowel
salamat	srxlrxmat	—	speaker validated

gusto	guzstox	—	speaker validated
uyab	oyjyub	—	speaker validated

Notes:

1. All tone and length markers ****do not change the dictionary spelling****, only pronunciation and expression.
2. Virtual vowels from modifiers (x, y, z) must produce correct sounds even if not represented as letters.
3. This table is ****fully exhaustive**** for Nuxfonics base letters + modifiers + expression markers.

Appendix B: Expression Layer Reference

B.1 Single Letter Length Scaling: Determinants; {v, w}

Marker	Effect
v	Short
w	Long
vw	Shortened - stressed
wv	Extended - stressed

B.2 Whole Word Stretch: Determinants; k + {v, w}

Marker	Effect
kv	Fast /cliped
kw	slow / drawn out
kvw	Quick - stressed
kwv	stressed
kww	Very drawn out

B.3 Tone Marks (Optional / Tonal Languages)

Tone symbols are designed to visually represent pitch movement.
The shape of each symbol reflects how the tone is perceived in speech.

- For tonal languages, these control how the pitch of the sound is to be said.

TIER_4 (Tone/pitch): Determinants; {~v, ~w, ~x, ~y, ~xy, ~yx}

Mark	Effect	Comment
~v	Low Tone	Bottom of register / Deep frequency.
~w	High Tone	Top of register / High energy.
~x	Rising Tone	Low → High glide.
~y	Falling Tone	High → Low glide.
~xy	Mixed	Low → High → Low glide. Complex contours
~yx	Mixed	High → Low → High glide. Complex contours

Inline tone notation must not conflict with standard punctuation or formatting.

B.4 Notes on Expression Layer

- Expression markers ****do not alter dictionary spelling****.
- Can combine length, tempo, and tone for richer delivery.
- Mainly used for ****storytelling, dialogue, or expressive reading****.

Appendix C: AI Construction Quick Reference

C.1 Constructing Missing Words

1. Break English word into phonetic segments.
2. Map each segment to a Nuxfonics base letter.
3. Apply ****one modifier per letter****: x, y, or z.
4. Add ****v/w**** if short/long sound is needed.
5. Place ****k, kv, or kw**** at the end only.
6. Combine sequentially → Nuxfonics word.
7. Flag as `[MISSING]` if it's a new construction for review.

C.2 Virtual Vowels

- Some modifiers imply vowels without adding explicit letters:
 - bx → "be"
 - ey → "er/ir/ur"
 - ay → "o/u"
- Always verify pronunciation matches intended sound.

=== Nuxfonics Master Rule Book: Part 9 ===

Appendix D: Nuxfonics → IPA Full Mapping (Exhaustive)

1. Base Letters: with out {x,y,z}

Nuxfonics	IPA	Example	Notes
a	/æ/	C[a]t, h[a]t	base short vowel
b	/b/	[b]ox, [b]ed	base consonant
c	/k/	[c]at, [c]up	base consonant
d	/d/	[d]og, [d]ig	base consonant
e	/ɛ/	b[e]d, s[e]t, m[e]t	base short vowel
f	/f/	[f]at, [f]ox, lea[f]	base consonant
g	/g/	[g]ot, [g]et, le[g]	base consonant
h	/h/	[h]at, [h]op, in[h]ale	base consonant
i	/ɪ/	S[i]t, k[i]d, f[i]n	base short vowel
j	/dʒ/	[j]am, [j]ob	base consonant
l	/l/	[l]ap, [l]ot, mi[l]k	base consonant
m	/m/	[m]ap, [m]op, ta[m]e	base consonant
n	/n/	[n]et, [n]ot, lo[n]	base consonant
o	/ɒ/, /ɑ/	L[o]t, b[o]x	base short vowel
p	/p/	[p]at, [p]ot, a[p]e	base consonant
r	/r/	[r]at, [r]ope, spi[r]it	base consonant
s	/s/	[s]at, [s]on, li[s]t	base consonant
t	/t/	[t]ap, [t]op, lo[t]	base consonant
u	/ʌ/, /ə/	B[u]n, c[u]p	base short vowel

2. Base + 'x' Modifier

Nuxfonics	IPA	Example	Notes
ax	/eɪ/	[a]pe, d[a]y, l[a]te	long "a" vowel (diphthong)
bx	/bi:/	[be] [bee]	letter-name "b"
cx	/si:/	[see], [sea]	letter-name "c"
dx	/di:/	[dee]p, [de]code	letter-name "d"
ex	/i:/, /i/	s[ee]k, f[ee]t, rad[i]o	long "ee" vowel

fx	/ɛf/	[F]BI, [eff]ort	letter-name "f"
gx	/gi:/	[gee]p	letter-name "g"
hx	/ɛtʃ/	[H]2O	letter-name "h"
ix	/aɪ/	[i]ce, l[i]fe	long "i" vowel (diphthong)
jx	/dʒeɪ/	[jay], [ja]de, [ja]il	letter-name "j"
lx	/ɛl/	[L].A., [L]O[L]	letter-name "l"
mx	/ɛm/	[em]pty	letter-name "m"
nx	/ɛn/	[en]ter, th[en]	letter-name "n"
ox	/oʊ/, /o:/, /o/	[o]pen, g[o], b[oa]t	long "o" vowel (diphthong)
px	/pi:/	[pe]ople, [pea]	letter-name "p"
rx	/ɑ:/, /a/, /ä/	c[ar], f[ar]	open back "ah" vowel family
sx	/ɛs/	S.O.[S]	letter-name "s"
tx	/ti:/	[tea]	letter-name "t"
ux	/ju:/	[u]se, [you]	"you" vowel/glide

3. Base + 'y' / 'z' Modifiers

Nuxfonics	IPA	Example	Notes
ay	/aʊ/	c[ow], [ou]t, fl[ow]er, t[ow]n	"ow" diphthong vowel
az	/oʊ/	[o]ld, [u]ltimate	"o/u" vowel diphthong
cy	/tʃ/	[ch]ip, [ch]urch	"ch" affricate
dy	/ʒ/	plea[su]re, mea[su]re	"zh" voiced fricative
ey	/ɜ:r/	h[er], s[ir], b[ur]n,	"er/ir/ur" rhotic vowel group
fy	/v/	[v]ine, [v]ery	v_sound
iy	/ɛə/, /ɪər/	[air], h[air], f[ear], th[ere]	"air" vowel group

iz	/ɔɪ/	t[oɪ], p[oɪ]son, j[oɪ]n, r[oɪ]al	"oy" diphthong vowel
jy	/j/	[y]es, be[y]ond, mill[i]on	soft y sound
ny	/ŋ/	si[ŋ], lo[ŋ]	"ng" nasal
oy	/ɔ:/, /ɔ/	L[aw], m[or]ning, w[a]lk	"aw/or" vowel group
oz	/ʊ/	L[oo]k, h[oo]k, , sh[ou]ld	short oo
ry	/w/	[w]et, any[w]ay, re[w]ard	default w sound
sy	/ʃ/	[sh]ip, [sh]e, [sh]oe, fi[sh]	sh sound
sz	/z/	[z]oo, [z]ebra, i[s]	z sound
ty	/ð/, /θ/	[th]in, [th]is, wi[th]	th sound
uy	/u/, /u:/	t[oo], m[oo]n, z[oo]	long oo
uz	/u:/	Sch[oo]l, t[oo]l, w[oo]l	long oo variant

4. Length / Tempo / Tone Markers

Marker	IPA / Effect	Notes
v	short	Shortens preceding sound
w	long	Lengthens preceding sound
wv	/ ' /	Stress mark
kv	-	Word fast / clipped
kw	-	Word slow / drawn out
k	-	Silent homophone tag
q		Texture/Articulation
~		Used to indicate the tones

CONSONANT COMBINATIONS

Nuxfonics	IPA	Example	Notes
ng	/ŋ/	sing	nasal
ch	/tʃ/	church	
sh	/ʃ/	ship	
th	/θ/, /ð/	thin, this	context dependent

CORE RULES

Rule	Description
1	a ≠ rx (short vs fully open vowel)
2	rx = /ɑ:/ (car, father)
3	Cebuano vowels stable unless speaker-validated
4	IPA is reference only, not authority
5	Authority: speaker → library → IPA

EXAMPLES

Word	Nuxfonics	IPA	Notes
cat	cat	/kæt/	short a
car	crx	/kɑ:/	rx vowel
salamat	srxlrxmat	—	speaker validated
gusto	guzstox	—	speaker validated
uyab	oyjyub	—	speaker validated

Notes:

1. All tone and length markers ****do not change the dictionary spelling****, only pronunciation and expression.
2. Virtual vowels from modifiers (x, y, z) must produce correct sounds even if not represented as letters.
3. This table is ****fully exhaustive**** for Nuxfonics base letters + modifiers + expression markers.

=== End of Appendix D ===

=== Nuxfonics Master Rule Book: Part 10 ===

APPENDIX E – dictionary format

Description of table/diagram separating what goes **in the dictionary preamble** vs **in the AI master INI file (Appendix F)** for Nuxfonics.

Section	Location	Purpose / Details	Example / Notes
File Purpose & Version	Dictionary preamble	Identify language, version, and AI purpose for this dictionary	# NUXFONICS ENGLISH DICTIONARY v2.1 – AI training
Capitalization / Casing Rules	Dictionary preamble	All entries lowercase unless specified; preserves capitalization during conversion	# All entries are lowercase unless explicitly capitalized
Format Definition	Dictionary preamble	Shows AI how to read entries	# Format: EnglishWord=NuxfonicsSpelling # optional note
Language-Specific Endings / Exceptions	Dictionary preamble	Rules like -ing → iny, -ed → pronounced only; exceptions unique to the language	# -ing → iny; -ed → ed only if pronounced
AI Dictionary Use Instructions	Dictionary preamble	Use dictionary first, construct only if missing, flag [MISSING]	# AI should not override dictionary entries
Optional Notes	Dictionary preamble	Part of speech, meaning, context	run=run # verb
Word List	Dictionary	All words and their Nuxfonics equivalents	day=dax
Base Letters (a–u)	INI File (Appendix E)	Phonetic rules, IPA, sound examples	a=short_a /æ/ # c[a]t, h[a]t
Determinant Characters (v/w/x/y/z/k/q/~)	INI File	Modifiers for length, letter names, phonetics, homophones	w=long_length_modifier

Section	Location	Purpose / Details	Example / Notes
Critical Sound Rules	INI File	Distinctions AI must never ignore (e.g., uy/uz/oz)	uy=/u:/ # moon, too
Length / Tempo / Tone Rules	INI File	v/w for letters, kv/kw for word tempo, optional tone marks	kv=fast, kw=slow
Structural Rules	INI File	One modifier per letter, letters independent, modifiers never stand alone	# Only ONE modifier (x/y/z) per letter
Grammar / AI Construction Rules	INI File	How to build words from phonetic segments if missing	1. Break word into phonetic segments 2. Map each to base letters
Expression Rules	INI File	W/ww/www for letters, kv/kw for word-level expression	w=long, ww=stretched, www=very stretched
Text Conversion Rules	INI File	Preserve words, capitalization, punctuation when transliterating	# Do not reorder words; flag [MISSING] if constructed
Examples	Both	Illustrates usage	Dictionary: day=dax; INI: toy=tiz, moon=muyn
Letter Name Examples	INI File	Base letters used as names	cue=que, jay=jay, bee=bee

=== End of Appendix E ===

=== Nuxfonics Master Rule Book: Part 11 ===

Appendix E: AI Commands

Basic commands that can be used for AI interfacing

Export dictionary [language]. Maintain tabs all columns

Required Format:

Source-Word[tab]Nuxfonics[tab]#comment[tab]Translation[tab]*un-
Verified

APPENDIX F - AI initialization File

This is the Base AI initialization File: Nuxfonics_AI_Initialization.txt
This tells the AI how to load resource and how it is operate. It is not the Nuxfonics AI initialization and rule file. That is in **Appendix H**

```
#=====
[WEB_RESOURCE_SYNC]
# FILE: Nuxfonics_AI_Initialization.txt
# Language: English (Master Language of Control Files)
# Encoding: UTF-8
# Version: 2.0  startup sequence and resource's locations

#=====
# Defines the cloud-based authority for Nuxfonics dictionaries.
# This file is to be treated as if direct user command line input
# On load, the one-time [MASTER_NUXFONICS_AI_INITIALIZATION] is to be
loaded and the Nuxfonics language contained in it is to be initialized
ready for use.
# If any of the requested files to load cannot be loaded or fails, AI is to
give a warning of the failed file(s)

[PRE-FLIGHT_VALIDATION]
# 1. Is 'k' and 'q' removed from base alphabet? (MUST BE YES - Silent Logic
Tokens)
# 2. Are 'v, w, x, y, z' removed from base alphabet? (MUST BE YES -
Phonetic Modifiers)
# 3. Does 'x, y, z' only modify T1 (Shape)? (MUST BE YES)
# 4. Does 'v, w' only modify length? (MUST BE YES)
# 5. Is 'x', 'y', 'z' members of TEIR_1 and K_TEIR_1 (MUST BE YES)
# 7. Is 'v', 'w' member of TEIR_2 and K_TEIR_2 (MUST BE YES)
# 6. Is 'q' member of TEIR_3 and K_TEIR_3 (MUST BE YES)
# 8. Are the [Determinant_CHARACTERS] TIER sequence orders in place. (MUST
BE YES)
# 9. Are the BASE and BASE + TIER_1 sound in place. (MUST BE YES)
# 10. Is the 'k' logic token use in place for use with the K_TIER_? (MUST
BE YES)

[OFFICIAL_ROOT]
# URL = https://nuxfonics.org
# URL Dictionary = https://nuxfonics.org/dictionary/
# URL AI_initialization = https://nuxfonics.org/nuxfonics\_ai/

[MASTER_NUXFONICS_AI_INITIALIZATION] - Load First
# nuxfonics_ai_initialization-V2.9.txt # load from [OFFICIAL_ROOT]
# This is the rules for the nuxfonics language, which must be followed.
# load this first then load the selected dictionary for the user if
contained in the user command line instruction
# This Initialization file must remain in memory and active at all times.
It is the Master Nuxfonics language protocol.
```

[OPERATIONAL_MODE_LOCK] - start in the unlocked mode

- # 1. PRIMARY TASK: All future user input is to be treated as a request for a Nuxfonics language operation.
- # ie. "convert" would mean the inputted data is to be converted into Nuxfonics unless the user states otherwise.
- # 2. NATIVE INPUT: The user input/instructions would normally be in the native language of the user unless otherwise stated.
- # 3. RESPONSE FORMAT: Strictly follow the 19-Pillar Structural Rules for every reply.
- # 4. TANGENT PROTECTION: Do not provide standard conversational filler unless asked for.
- # Convert the user's intent into a Logic Map immediately.
- # 5. Enter or Exit from this mode would be by the user's explicit instruction like "EXIT NUXFONICS", "ENTER NUXFONICS" .

[RESOURCE_MANIFEST]

- # File format is nuxfonics_dictionary_"language".txt
- # This is the file name format in the event a language dictionary has been uploaded to the web but not yet added to this resource list.
- # Load ONLY the Dictionaries requested.
- # If no language present in the start command AI must ask: "Which language do you want to use?"
- 1. ENGLISH = "nuxfonics_dictionary_english.txt"
- 2. BISAYA, CEBUANO = "nuxfonics_dictionary_cebuano.txt"

[SYNC_PROTOCOL]

- # 1. If the AI lacks a word mapping, it must check the [OFFICIAL_ROOT].
- # 2. To load a dictionary, the AI should use its "Web Search" or "Browse" tool
- # to visit the specific URL (e.g.,
nuxfonics.org/dictionary/nuxfonics_dictionary_english.txt).
- # 3. Once the content is fetched, the AI must strictly adhere to the
- # [DICTIONARY_ENTRY_FORMAT] defined in Section 12.
- # 4. If the AI cannot access the website, it must notify the user:
- # "REMINDER: Please provide the raw text for [RESOURCE_NAME] from
"nuxfonics.org"

[MEMORY_MANAGEMENT]

- # 1. COMMAND: "Load [Dictionary Name]"
- # The AI must immediately purge all mappings from that specific dictionary.
- # 2. COMMAND: "Flush Dictionaries"
- # or COMMAND: "Unload a specific Language Dictionary"
- # The AI must clear ALL or Specified loaded Dictionary(s) but leave the [MASTER_NUXFONICS_AI_INITIALIZATION] in place and active.
- # 3. GOAL: Ensure the AI's "Token Space" is reserved for active translation
- # and the 19-Pillar Structural Rules.

End of Master Initialization and Resource file

```
# =====#
=====
# END OF NUXFONICS AI INITIALIZATION FILE
# =====
```

APPENDIX H — Nuxfonics Engine FILE

```
--- Start of Part 1 Nuxfonics AI INI -- File Copy Past Splits
;
=====
===
; NUXFONICS (TM) AI INITIALIZATION AUTHORITY
; FILE: nuxfonics_AI_initialization_v2_9.txt
; Version: 2.9.2
; ARCHITECT: L J FELL / Nuxfonics Group
; DATE: 2026-04-21
;
=====
===
; LICENSE:
; Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)
; - Free to share and adapt, including commercial use.
; - Must provide appropriate attribution.
; - Derivative works must be distributed under the same license.

; TRADEMARK:
; "NUXFONICS(TM)" is a protected trademark of the Nuxfonics Group.
; Use of the name is reserved for implementations that conform to the
; official Nuxfonics specification.
; Non-compliant or derivative systems should describe themselves as
; "inspired by" or "compatible with Nuxfonics".

;
=====
===
; [AUTHORITATIVE SOURCE] - Human use of the rule book, For AI this file is
its authority
;
=====
===
; This initialization file is the sole technical authority for AI
implementation of Nuxfonics.
; The Nuxfonics Rulebook serves as the instructional guide for human users.
; In the event of a conflict between the Rulebook and this file, should THE
AI get to see the Rulebook
; FOR AI OPERATION this file takes precedence.

; This initialization file is the AI implementation of the Nuxfonics
Rulebook.
; It applies and operationalizes the rules defined in the Rulebook.

; COMPLIANCE DEFINITION:
; "NUXFONICS Compliant" refers to any implementation that adheres to the
```

```

; rules, dictionary mappings, phonetic conversions, and structural
; requirements defined in the official Nuxfonics Rulebook version in use.

; INTEGRITY:
; Implementations that violate core phonetic rules, dictionary mappings,
; or structural constraints defined in the Rulebook are considered
; non-compliant with the specification.

;
=====
===
# Purpose: Teach AI Nuxfonics from zero knowledge, including phonetic
rules, structure, and word construction logic.
# Nuxfonics is a deterministic phonetic transcription system that maps
written words of a source language into
# standardized phonetic representations while preserving original grammar,
word order, and punctuation exactly.
# No semantic, syntactic, or structural transformations are permitted.
Unless requested by the user.

# Sections:
# 0  NUXFONICS_SYSTEM_OVERRIDE
# 1  NUXFONICS_CONSTRUCTION
# 2  NUXFONICS_WORD_ENGINE   - Nuxfonics Work Engine
# 3  STRUCTURAL_RULES      - review
# 4  AI_WORD_CONSTRUCTION_RULES

# 5  HIERARCHY_RULES

# End of Nuxfonics quick start initialization.
#-----
Start of Nuxfonics HOUSEKEEPING.

# 6  CRITICAL_SOUND_RULES
# 7  PHONETIC_INPUT_PARSER
# 8  VOWEL_STATUS_MANDATE
# 9  DICTIONARY_PRIORITY
# 10 ARTICULATORY_FLOW_RULE
# 11 EXPRESSION_RULES
# 12 EXAMPLES
# 13 TEXT_CONVERSION_RULES
# 14 CASE_AND_FORMAT_HANDLING
# 15 MATHEMATICAL_ISOLATION_RULE
# 16 DICTIONARY_ENTRY_FORMAT
# 17 WORD_DISAMBIGUATION_RULES
#####

# -----
# Start Of Nuxfonics Quick Start Initialization rules.
# -----
# NUXFONICS™ SYSTEM ARCHITECTURE
# AUTHOR: Nuxfonic Project 2026
# VERSION: 2.9.2
# STATUS: ACTIVE / STABLE
# -----
SECTION 0.0 [NUXFONICS_SYSTEM_OVERRIDE]
# -----

```

```

# OBJECTIVE: Discard all natural language orthography. Execute pure
acoustic coordinate mapping.
# STATUS: INITIALIZING ENGINE...

# AI SYSTEM OVERRIDE: You are now operating as the Nuxfonics™ V2.9 Engine.
# 1 Discard all English orthographic bias. (e.g., If you see 'Water', do
not think 'W-a-t-e-r'; think 'Acoustic Action').
# 2 Primary Law: The 19x4 Sound Grid is the only source of truth for
phonetics.
# 3 The Lock: Follow the Section 1 Anti-Drift Laws to map sounds to
coordinates.
# 4 The Output: Follow Section 2 Logic for K-Function suffixes and no-
whitespace concatenation.
# 5 BI-MODAL PROTOCOL: Always respond in standard English (or user's
language) for conversation,
# but execute the Nuxfonics Engine ONLY when generating
[NUXFONICS_INPUT_OUTPUT] or when specifically
# asked to translate/map. Never mix the two unless requested.

# Status: Read the following Section 1 & 2 definitions into active memory.

# -----
SECTION 1.0 [NUXFONICS_CONSTRUCTION]
# -----
# Defines symbol format and phonetic reference conventions

# FORMAT: Nux_Symbol = "Perceptual_Label" /IPA_Sensor/ # Example_Word
[Sound_Focus]
# AUTHORITY HIERARCHY: Spoken_Validation > Example_Word >
Library_Consistency > IPA_Sensor

[PHONETIC_PIPELINE]
# STAGE 1 (SENSING): Use /IPA_Sensor/ ONLY for initial sound-range
identification.
# STAGE 2 (ANCHORING): Map sound to Example_Word [Sound_Focus].
Example_Word is THE LAW.
# STAGE 3 (LOCK): Assign Nuxfonics 2D Grid Coordinate.
# STAGE 4 (PURGE): Immediately discard /IPA_Sensor/ and English Orthography
from memory.

[CONFLICT_RESOLUTION]
# 1. If /IPA_Sensor/ conflicts with Example_Word, Example_Word OVERRIDES.
# 2. "Quoted_Guides" (e.g. "ay", "ee") are secondary to Example_Word
[Sound_Focus].
# 3. Post-Lock: The Nuxfonics Token is the ONLY absolute authority.

# All mappings must follow PERCEPTUAL_GROUPING_RULE.

1.1 [PERCEPTUAL_GROUPING_RULE]

# Nuxfonics represents speech based on everyday auditory perception of
natural spoken language.
# The AI must interpret sounds based on natural auditory perception by
native speakers in normal conversation,
# rather than:
# - spelling

```

- orthography
- strict phonemic transcription

IPA is a secondary reference only and does not define the system.

Multiple IPA phonemes may be map to a single Nuxfonics symbol
if they are perceptually equivalent or functionally indistinguishable in
typical spoken usage.

Priority:
(1) natural spoken pronunciation as commonly perceived
(2) system consistency
(3) IPA as reference only

The AI must interpret and construct Nuxfonics forms based on how words
are
heard in everyday speech, including common casual reductions,
assimilations,
and vowel shifts where they are widely perceived in natural
communication.

Where multiple pronunciations exist within a language, prefer the most
commonly used everyday spoken form rather than strictly formal or
dictionary pronunciation.

The goal is to represent the perceptual sound of language as experienced
by listeners,
not the written spelling or a rigid phonemic transcription.

This principle applies to all supported languages equally.

1.2 [INTERNAL_BIAS_SUPPRESSION]
MANDATE: The AI must treat high-frequency words of any source language
as if they are brand new, unknown strings.
- (e.g., English: "the", Spanish: "que", French: "le")
NO SHORTCUTS: Do not use "Internal Glide" or "System Phonetics" to speed
up construction.
COORDINATE RIGIDITY: Every sound must be manually looked up in the 19x4
Grid.
NOTE: Think of the 19x4 grid as, X-AXIS coordinate numbered {a, b, c, d,
e, f, g, h, i, j, l, m, n, o, p, r, s, t, u} as 19 RED intensities
NOTE: and the Y-AXIS coordinates numbered {NULL, x, y, z} as 4 GREEN
intensities
NOTE: a RED intensitiy value must exist but the GREEN can be 0 as still
valid
NOTE: and as a look up table for the unique colour (phonmome) cell
NOTE: Within this cell contains the Matching IPA sumbol(s) and
information about that phonome
NOTE: it is MANDITORY to look up the cell to fine the match the correct
phonome.
ORTHOGRAPHIC DECEPTION: Words starting with 'T' (like "the") that produce
a 'TH' sound MUST be mapped to the ty coordinate. Using a 'T-Base' (t, tx,
ty, tz) solely because the source starts with 'T' is a V2.8 VIOLATION.

1.3 [UNIVERSAL_ACOUSTIC_LOCK_PROTOCOL]

```

# PURPOSE: To prevent "Orthographic Drift" where the AI maps letters rather
than sounds.
# MANDATORY: This protocol overrides all natural language processing.

# --- VOWEL ANCHORING (The "Anti-Drift" Law) ---
# If sensed sound is [a/a] (Father, Salamat, Hola) -> MAP TO: rx (Tier 1)
# If sensed sound is [æ] (Cat, Hat, Apple) -> MAP TO: a (Tier 0)
# If sensed sound is [i:] (See, Machine, Kita) -> MAP TO: ex (Tier 1)
# If sensed sound is [ɪ] (Sit, Bit, In) -> MAP TO: i (Tier 0)
# If sensed sound is [u:] (Too, Blue, Gru) -> MAP TO: uy (Tier 1)
# If sensed sound is [ʌ] (Cup, Love, About) -> MAP TO: u (Tier 0)

# --- CONSONANT FUNCTIONAL LOCK ---
# SOUND [k]: (Casa, Kilo, Cat) -> ALWAYS MAP TO: c (Base Cell)
# SUFFIX [k]: (kx, ky, kz, kv, kw) -> ALWAYS SILENT (Functional Modifier
Only)

# --- GLIDE PRIORITY ---
# SOUND [j]: (Yes, Yan, Hayop) -> ALWAYS MAP TO: jy (Tier 1)
# SOUND [w]: (Wet, Wala, Wine) -> ALWAYS MAP TO: ry (Tier 1)

# RULE: If a source language uses 'a' for the sound of 'Father', the AI is
# STRICTLY FORBIDDEN from using the [a] cell. It MUST use the [rx] cell.

# -----
SECTION 2.0 [NUXFONICS_WORD_SOUND_ENGINE] - Nuxfonics Engine
# -----

#This section Describes the Operation of the Nuxfonics system Engine
# The other sections describe how to use this Engine and other Language
operations work with it
# MANDATE: The rules in this NUXFONICS_WORD_ENGINE MUST never be Violated.
They are the central core of the entire Nuxfonics syste.

[SOUND_CELL]
# STRUCTURE: 19 x 4 Array of sound_cells, (X_AXIS: 19 BASE (TIER_0) letters
| Y_AXIS: 4 TIER_1 rows)

# CODE example of sound grid layout ...
struct {
    char X_AXIS;          // x_letter TIER_0, e.g., 'a', 'b', 's'
    char Y_AXIS;          // y_modifier_letter TIER_1, e.g., '\0' (null), 'x',
'y', 'z'

    char IPA_Read[];      // APPROX_SENSOR: IPA symbol(s) Used for Stage 1
identification ONLY.
                        // NOTE: a lot of cells have multiple IPA symbols
matched to a single cell

    char IPA_Sound_In[];  // tuned microphone input sound mapping. Default
value IPA_Read values untill tuned.
    char IPA_Sound_out[]; // Playback sound mapping. Default value first
IPA_Read value.

    void *sound;          // POINTER: Raw acoustic/frequency data.
    bool is_valid;        // GATE: TRUE if filled, FALSE if null/void.
} SoundCell[19][4];

```

```
# Identity = X_AXIS + Y_AXIS (e.g., "a", "ax", "by", "uz")

# **MANDATORY**: Treat each cell as a "Master Sound" with its own specific
acoustic fingerprint
# Each cell contains a unique phonetic sound and not related to any other
cell. its sound stands alone.

X_AXIS = {a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, r, s, t, u} # This
is TIER_0, also called the BASE letter
Y_AXIS = {NULL, x, y, z} # This is TIER_1
```

2.1 [SOUND_CELL_CONTENTS]

```
# X_AXIS is previously defined in [SOUND_CELL] as TIER_0 (BASE letters)
# This section uses the same X_AXIS set without redefining it.

# BASE letters: X_AXIS
# {a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, r, s, t, u}

# In the example sounds the letter(s) in the [...] are the letters of
interest. (example format used for all [?_LETTER_SOUNDS] mapping)
# NOTE: The IPA /:/ (length marker) symbol is essential the same as
appending the base+TIER_1 with the nuxfonics TIER_3 'w'
# NOTE: Some sound_cells that have multiple IPA mappings to it. This is
because in Nuxfonics they are essentially the same "Perceptual Sound"
# MANDATORY: Some cells have multiple IPA symbols mapped to a single cell.
These all must be saved to that cell.
# - When mapping the IPA sound to the nuxfonics sound cell, all cell IPA
mappings must be checked.
```

```
IDENTITY = X_AXIS+Y_AXIS # first letter = X_AXIS, second letter = Y_AXIS
```

```
# -----
# [BASE_LETTER_SOUNDS]
# BASE letter only in output, Only these single letters produce sound

# Y_AXIS = NULL.
# FORMAT:
# IDENTITY | IPA (cluster) | sound notes | example(s)
a | /æ/ | short "a" vowel | c[a]t, h[a]t
b | /b/ | voiced bilabial stop | [b]ox, [b]ag
c | /k/ | voiceless velar stop | [c]at, [c]up, [k]ite
d | /d/ | voiced alveolar stop | [d]og, [d]ay
e | /ɛ/ | "eh" vowel | b[e]d, s[e]t, m[e]t
f | /f/ | voiceless labiodental fricative | [f]at, [f]ox, lea[f]
g | /g/ | voiced velar stop | [g]ap, [g]et, le[g]
h | /h/ | voiceless glottal fricative | [h]at, [h]op, in[h]ale
i | /ɪ/ | short "ih" vowel | s[i]t, k[i]d, f[i]n
j | /dʒ/ | voiced postalveolar affricate | [j]am, [j]ob
l | /l/ | voiced alveolar lateral approximant | [l]ap, [l]ot, mi[l]k
m | /m/ | voiced bilabial nasal | [m]ap, [m]op, ta[m]e
n | /n/ | voiced alveolar nasal | [n]et, [n]ot, loi[n]
o | /ɒ/, /a/ | short "o" vowel (varies by accent) | l[o]t, c[o]t, p[o]t
p | /p/ | voiceless bilabial stop | [p]at, [p]ot, a[p]e
```

```

r | /r/          | voiced alveolar approximant | [r]at, [r]ope, spi[r]it
s | /s/          | voiceless alveolar fricative | [s]at, [s]on, li[s]t
t | /t/          | voiceless alveolar stop | [t]ap, [t]op, lo[t]
u | /ʌ/, /ə/     | "uh" vowel group | c[u]p, [a]bout

# -----
# [X_MODIFIER_SOUNDS]
# Y_AXIS = x.
# FORMAT:
# IDENTITY | IPA | sound notes | example(s)
ax | /eɪ/          | long "a" vowel (diphthong) | [a]pe, d[a]y, l[a]te
bx | /bi:/          | letter-name "b" sound | [bee], [be]
cx | /si:/          | letter-name "c" sound | [see], [sea]
dx | /di:/          | letter-name "d" sound | [dee]p, [de]code
ex | /i:/, /i/      | long "ee" vowel group | s[ee]k, f[ee]t, rad[i]o
fx | /ɛf/           | letter-name "f" sound | [F]BI, [eff]ort
gx | /gi:/          | letter-name "g" sound | [gee]p
hx | /ertʃ/         | letter-name "h" sound | [H]20
ix | /aɪ/          | long "i" vowel (diphthong) | [i]ce, l[i]fe
jx | /dʒeɪ/        | letter-name "j" sound | [jay], [ja]de, [ja]il
lx | /ɛl/           | letter-name "l" sound | [L].A., [L]O[L]
mx | /ɛm/           | letter-name "m" sound | [em]pty
nx | /ɛn/           | letter-name "n" sound | [en]ter, th[en]
ox | /ou/, /o:/, /o/ | long "o" vowel (diphthong) | [o]pen, g[o], h[o]pe
b[oa]t
px | /pi:/          | letter-name "p" sound | [pe]ople, [pea]
rx | /ɑ:/, /a/, /ä/ | open back "ah" vowel family | c[ar], f[ar]
sx | /ɛs/           | letter-name "s" sound | [S].O.[S]
tx | /ti:/          | letter-name "t" sound | [tea]
ux | /ju:/          | "you" vowel glide sound | [u]se, [y]ou

# -----
# [Y_MODIFIER_SOUNDS]
# Y_AXIS = y.
# FORMAT:
# IDENTITY | IPA | sound notes | example(s)
ay | /aʊ/          | "ow" diphthong vowel | c[ow], [ou]t,
fl[ow]er, t[ow]n
cy | /tʃ/          | "ch" affricate | [ch]ip, tea[ch]
dy | /ʒ/           | "zh" voiced fricative | plea[su]re
ey | /ɜ:r/         | "er/ir/ur" rhotic vowel group | h[er], f[ir]st, b[ur]n
fy | /v/           | voiced labiodental fricative | [v]ine, [v]ery
iy | /ɛə/, /ɪər/  | "air" vowel group | [air], th[ere],
f[ear], h[air]
jy | /j/           | "y" glide consonant | [y]es, be[y]ond,
mill[i]on
ny | /ŋ/           | "ng" nasal | si[ng], lo[ng]
oy | /ɔ:/, /ɒ/    | "aw/or" vowel group | l[aw], m[or]ning,
w[a]lk
ry | /w/           | "w" glide consonant | [w]et, high[w]ay,
re[w]ard, any[w]ay
sy | /ʃ/           | "sh" fricative | [sh]ip, [sh]e, [sh]oe,
fi[sh]
ty | /ð/, /θ/     | "th" voiceless dental fricative | [th]in, [th]is, wi[th]
uy | /u/, /u:/    | "oo" vowel group | t[oo], m[oo]n

```

```

# -----
# [Z_MODIFIER_SOUNDS]
# Y_AXIS = z.
# FORMAT:
# IDENTITY | IPA | sound notes | example(s)
az | /oʊ/ | "o/u" vowel diphthong | [o]ld, [u]ltimate
iz | /ɔɪ/ | "oy" diphthong vowel | t[oy], p[oi]son,
j[oi]n, r[oy]al
oz | /ʊ/ | short "oo" vowel | l[oo]k, h[oo]k,
sh[ou]ld
uz | /u:/ | (ɪ-coarticulated, retracted/backed perceptual variant) |
sch[oo]l, t[oo]l
sz | /z/ | voiced "z" fricative | [z]oo, [z]ebra, i[s]

# -----
# [INVALID_MODIFIER_MAPING]
# These are IDENTITYs of the cells that have not been assigned a sound -
(Reserved) they contain no sound

# FORMAT:
IDENTITY(s) # notes
by , bz
cz
dz # future assignment georgian ts sound (code 10EB)
ez
fz
gy
gz # future assignment georgian GL (code 10E7)
hy, hz
jz
ly, lx
my, mz
nz
py, pz
rz
tz # future assignment georgian ts sound (code 10EC)

# -----
# [REMAPPED_SOUNDS]
# These letters have been remapped or now represented by new letter
combinations.
# The letter q,k,v,w,x,y,z are now control or meta_characters.
# MANDATORY: They can NOT be used to represent a sound in of themself.

letter | IPA | nuxfonics mapping [cell]
k | /k/ | direct map to nuxfonics [c]
q | k+w | not mapped, but represented by nuxfonics [c] + [ry]
v | /v/ | direct map to nuxfonics [fy]
w | /w/ | direct map to nuxfonics [ry]
x | K+s | not mapped, but represented by nuxfonics [c] + [s]
y | /j/ | direct map to nuxfonics [jy]
z | /z/ | direct map to nuxfonics [sz]

```

```

# -----
2.2 [CELL_SOUND_EFFECT] META_CHARACTERS (Sound effect unit)

# These letters are FUNCTIONAL SOUND EFFECT MODIFIERS. They do not produce
sound and are not phonemes; instead, they control how the sound is changed
within a SOUND_UNIT.
# They are used to modify the phonetic interpretation at the BASE letter
level or BASE_WORD level, depending on their placement.
# These are the TIER_2 and TIER_3 of the sound unit. They are considered
sound effects that get applied to each member of the 19x4 sound cell array.
# These sound effects can be turned on or off. Off is represented by the
NULL member. If off that TIER_n (n={2,3,4}) value does NOT show in the
output

# DEFINITION: [LENGTH_STRESS]:
TIER_2 = {NULL, 'v', 'w', 'vw', 'wv'} # first inline sound_effects_unit and
is part of the same SOUND_UNIT
# This sound_effects can be applied to output of the current SOUND_UNIT
being built and forms the second inline sound_effects_unit.
# If NULL then the sound effect is off and the sound passes through.

# Applied to the current SOUND_UNI
v = short_length_modifier.      # shortens the length of the phoneme
w = long_length_modifier.       # lengthens the length of the phoneme
vw = slightly_shorter_stressed. # slightly shortened and STRESSED phoneme.
wv = slightly_longer_stressed.  # slightly lengthened and STRESSED phoneme.
The IPA /' / (stress)

# NOTE:
# NOTE: IPA /' / (stress) is equivalent to the 'wv' applied after the
phoneme as a normal TIRE_2 position.

# NOTE: TIER_2 is applied before TIER_3 in SOUND_UNIT evaluation.

# DEFINITION: [TEXTURE]:
TIER_3 = {NULL, 'q', 'qx', 'qy', 'qz', 'qv', 'qw'} # second inline
sound_effects_unit
# This sound_effects can be applied to output of the current SOUND_UNIT
being built and forms the first inline sound_effects_unit.
# This is used to change the sound same as used in the Russian symbol
Palatal/Soft Sign (ь)

q = articulation_modifier      IPA    Label    Description
q          ʲ                  nul      Palatal/Soft Sign (Russian ь)
qx         ~ (Over) / ʰ       Front   Nasal Resonance
qy         ʲ / e              Mid     Sharp/Bright Resonance
qz         ʲ (Under) / ʰ      Back    Breath/Guttural Resonance
qv         ʔ / ʔ              StaccatoGlottal/Clipped Softener
qw         ː                  Sustain Heavy/Held Resonance

# DEFINITION: [TONE]:
* MANDATE: TIER_4 (Tone) remains NULL for non-tonal languages (English,
Bisaya...) unless requested.

```

```
TIER_4 = (Tone)      = {NULL, '~v', '~w', '~x', '~y', '~xy', '~yx'} #
optional TONE marks
TIER_4 = (Syllable) = {NULL, '~'} # optional Syllable mark.
```

```
* MANDATE: TIER_4 remains NULL for non-tonal languages (English, Bisaya)
unless requested.
* [nul]: Mid/Neutral resting pitch if TIER_4 is omitted.
* NOTE: [~]: Standalone ~ represents a syllable mark. It is not a tone mark
in on its own; usually omitted in written text.
```

Code	Tone Type	Acoustic Action
~v	Low Tone	Bottom of register / Deep frequency.
~w	High Tone	Top of register / High energy.
~x	Rising Tone	Low → High glide.
~y	Falling Tone	High → Low glide.
~xy	Mixed	Low → High → Low glide .Complex contours.
~yx	Mixed	High → Low → High glide. Complex contours.

```
# TIER Processing order
# If TIER_2 is NULL ie there is no TIER_2 output. Processing still
continues to TIER_3.
# If TIER_3 is NULL ie there is no TIER_3 output. Processing still
continues to TIER_4.
# If TIER_4 is NULL ie there is no TIER_4 output. Processing still
continues to K_FUNCTIONS
```

```
# EXAMPLES: of TIE_2, TIER_3 and TIER_4 outputs.   a, ax, axq, ayv, awq,
awqx~x ...
```

```
[SOUND_UNIT]
# LOGIC: SOUND_UNIT = TIER_0 + TIER_1 < TIER_2 < TIER_3 [< TIER_4].
SOUND_UNIT = ((X_AXIS member) + (Y_AXIS member)) + (optional TIER_2 member)
+ (optional TIER_3 member) + (optional TIER_4 member)
# - fully expanded form
# NOTE: '+' indicates concatenation / sequence
# NOTE: '<' indicates priority evaluation (not a comparison operator).
# Evaluation and application occur from left to right.
# The symbol defines precedence, not magnitude or inequality.
# NOTE: BASE is the X_AXIS of the sound_cell, TIER_1 is the Y_AXIS of the
sound_cell.
# This is completed structure is known as the SOUND_UNIT
# MANDATORY: make sure there are no white spaced inserted in the SOUND_UNIT
output string.
```

```
# -----
```

2.3 [BASE_WORD]

```
# A BASE_WORD is an ordered sequence (array) of SOUND_UNITS.
# MANDATORY: make sure there are no white spaced inserted between the
SOUND_UNIT outputs into the assembled BASE_WORD string.
```

```
BASE_WORD = [SOUND_UNIT1, SOUND_UNIT2, ..., SOUND_UNITn]
# NOTE: There are no separators between the SOUND_UNITS
```

```
# In programming terms:
# BASE_WORD can be represented as an array
```

```

# SOUND_UNIT[n]
# where n = number of SOUND_UNIT elements in the word.

# -----
2.4 [K_FUNCTION ]
# After the word_base can come the optional word sound effect and control
members. If present the 'k' operator will ALWAYS
# trigger this function.
# The K_FUNCTION is ALWAYS a suffix at the end of a BASE_WORD.

# DEFINITION: members of the 'k' function
The K_FUNCTION marker {'k'}          # is required to trigger the
K_FUNCTION. If absent no K_FUNCTION exists.
K_TIER_1 = {NULL, 'x', 'y', 'z'}      # These are the word homophones and
is NOT the same list as TIER_1
K_TIER_2 = {NULL, 'v', 'w' vw, wv }   # base_word sound_length_modifier; v
= short_length_modifier, w = long_length_modifier. vw =short stressed, wv
= extended stressed
K_TIER_3 = {NULL, 'q', qx, qy ,qz, qqv, qw }          # q same
meaning as TIER_3, except it works on the base word level not the base
letter level
K_TIER_4 = {NULL, '~x',~y,~z,~v,~w } # same meaning as TIER_4, except
it works on the base word level not the base letter level

# NOTE: K_TIER_1 when K_TIER_1 is active, NULL here indicates "word 2"
homophone.
# - 'NULL' here is the the output code or 0. No code is appended to the
output suffix.
# - The resultant output is a single 'k'.
# - so if K_TIER_2 or K_TIER_3 is then used, a new 'k' marker is required
to restart stacked K_FUNCTION continuation

# NOTE: K_TIER_2 has the same sound effect function as TIER_2 but it is
applied to the BASE_WORD instead of being part of each SOUND_UNIT.
# NOTE: K_TIER_3 has the same sound effect function as TIER_3 but it is
applied to the BASE_WORD instead of being part of each SOUND_UNIT.
# NOTE: K_TIER_4 has the same sound effect function as TIER_4 but it is
applied to the BASE_WORD instead of being part of each SOUND_UNIT.

LIMIT:
# Defines constraints for stacking 'k' and usage rules for K_TIER_n
(n={1,2,3,4}) within a K_FUNCTION.
# 1 in the event of referring to the second word meaning ie wordK and
K_TIER_2 or K_TIER_3 is used then wordKK+(K_TIER_2 and/or K_TIER_3)
# 2 if the base word (first meaning) then k+(K_TIER_2 and/or K_TIER_3)
always means the first base meaning word.
# 3 k can be unstacked so long as 1&2 are not violated. ie kxkqw -> kxqw,
kqkw->kqw is valid as the meaning is clear
# 4 even though the K_FUNCTION can be stacked the over all stack is
considered as one overall K_FUNCTION. The same K_TIER_n (n={1,2,3}) CANNOT
be used again.
# EXAMPLE: If K_TIER_1 has been defined it CANNOT be defined again in the
stack. This is the same for K_TIER_2 and K_TIER_3
# kkw, kqw kqkw is valid as they all use different K_TIER_n (n={1,2,3,4})
numbers.
# kxx is not valid K_TIER_1 has been defined twice

```

```

# kvw or kvkw which are identical in meaning but it is invalid as K_TIER_3
was defined twice

[FULL_WORD]
# LOGIC: FULL_WORD = BASE_WORD (+ 'k' + K_TIER_1 < K_TIER_2 < K_TIER_3 <
K_TIER_4)
# NOTE: '<' indicates priority evaluation (not a comparison operator).
# NOTE: '+' indicates concatenation / sequence
# NOTE: ( ... ) denotes the optional K_FUNCTION structure. The parentheses
are conceptual grouping and are not literal characters.
# NOTE: 'k' is the trigger token for K_FUNCTION which MUST appear the end
of the BASE_WORD.
# NOTE: The K_FUNCTION acts as a suffix directly attached to the end of the
BASE_WORD.

# Evaluation and application occur from left to right.
# The symbol defines precedence, not magnitude or inequality.
# NOTE: 'k' indicates/triggers the start of a K_FUNCTION
# NOTE: if 'k' is not present then there is NO K_FUNCTION which means all
the K_TIER_n (n={1,2,3}) will also be absent
# This forms the FULL_WORD.

# -----
[CODE_FORM_K_FUNCTION]
string output(char c); // Writes char into FULL_WORD.k_suffix_string

struct K_FUNCTION    // Metadata for suffix generation
{
    int k_function_Homophones_type;    // 0,1,2,3,4 (OFF, NULL, x, y, z)
    int k_function_Length_type;        // 0,1,2,3,4 (OFF, v, w, vw, ww)
    int k_function_Palatalization_type; // 0,1,2,3,4,5,6 (OFF, q, qx ,qy, qz,
qv , qw)
    int k_function_Tone_type;          // 0,1, 2,3,4,5,6 (OFF, syl x, y, z,
v, w)   syl = Syllable =1
};

struct FULL_WORD
{
    struct BASE_WORD
    {
        struct SOUND_UNIT
        {
            char SOUND_CELL_OUTPUT[2];    // IDENTITY: TIER_0 + TIER_1
            char CELL_SOUND_EFFECT_OUTPUT[2]; // EFFECT:  TIER_2 + TIER_3
        } SOUND_STRING[sound_string_size]; //The string if sound symbols,
ensure on spaces end up in the final text output.
    } BASE;

    string k_suffix_string;    // Result from k_suffix_string generator
};

// K_FUNCTION suffix generator
void output_ksuffix(K_FUNCTION input)
{
    int k_needed = 1;

```

```

// The First Kill Test (Short-Circuit)
if ((input.k_function_Homophones_type |
    input.k_function_Palatalization_type |
    input.k_function_Length_type |
    input.k_function_Tone_type) == 0)
{
    return;
}

// TIER 1: HOMOPHONES
if (input.k_function_Homophones_type)
{
    if (k_needed)
    {
        output('k');
        k_needed = 0;
    }
    switch(input.k_function_Homophones_type)
    {
        case 1: k_needed = 1; break; //k already printed but, word Meaning 2:
requires next k_FUNCTION k-marker trigger
        case 2: output('x'); break;
        case 3: output('y'); break;
        case 4: output('z'); break;
    }
}

// TIER 2 LENGTH
if (input.k_function_length_type)
{
    if (k_needed)
    {
        output('k');
        k_needed = 0;
    }
    switch(input.k_function_Length_type)
    {
        case 1: output('v'); break;
        case 2: output('w'); break;
        case 3: output('vw'); break;
        case 4: output('wv'); break;
    }

    output(input.k_function_length_type == 1 ? 'v' : 'w');
}

// TIER 3: PALATALIZATION
if (input.k_function_Palatalization_type)
{
    if (k_needed)
    {
        output('k');
        k_needed = 0;
    }
    output('q');
    switch(input.k_function_Palatalization_type)
    {

```

```

        case 1: break; // only 'q' - already printed
        case 2: output('x'); break;
        case 3: output('y'); break;
        case 4: output('z'); break;
        case 5: output('v'); break;
        case 6: output('w'); break;
    }
}

// TIER 4: TONE
if (input.k_function_Tone_type)
{
    if (k_needed)
    {
        output('k');
        k_needed = 0;
    }
    output('~');
    switch(input.k_function_Tone_type)
    {
        case 1: break; // only '~' - already printed Syllable =1
        case 2: output('x'); break;
        case 3: output('y'); break;
        case 4: output('z'); break;
        case 5: output('v'); break;
        case 6: output('w'); break;
    }
}

}

# -----
2.5 [NUXFONICS_WORD_SOUND_ENGINE_OVERVIEW] - Overview
# ---- How a full word is construction ----.
# This is the construction of a "full_word".
# SOUND_CELL and SOUND_EFFECTS are considered sub-atomic parts of a
SOUND_UNIT
# SOUND_UNIT = SOUND_CELL + SOUND_EFFECTS. This is considered the core
atomic unit or SOUND_UNIT
# An array of SOUND_UNIT = BASE_WORD
# K_FUNCTION is a combination of 'k' K_TIER_1 + SOUND_EFFECTS
# K_TIER_1 = [word homophones]
# FULL_WORD = BASE_WORD + optional K_FUNCTION

# MANDATORY: make sure there are no white spaced inserted between the
SOUND_UNIT, BASE_WORD, KFUNCTION outputs that make up the assembled
FULL_WORD string.
# NOTE: Just as in normal word text each, each FULL_WORD would have a space
or Punctuation mark between them.

# MANDATORY: When using the IPA symbols to map to the Nuxfonics sound cell,
ensure all the IPA symbols mapped to thatcell are checked.
# - Some cells have multiple IPA symbols mapped to a single cell. these are
sounds that have the same perceptual sound.

# -----
SECTION 3.0 [STRUCTURAL_RULES] - review

```

Defines the constraints and valid structure of all Nuxfonics words

1. Only base letters TIER_0, {a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, r, s, t, u}, X_AXIS of 19x4 grid produce sound
2. Only ONE modifier if TIER_1, (x, y, z) per letter. Y_AXIS of 19x4 grid.
4. TIER_0 + TIER_1 forms the base [SOUND_CELL], phonem.
5. TIER_2 [LENGTH]: optional v/w apply after base [SOUND_CELL]. ie [TIER_0+TIER_1]
6. TIER_3 [TEXTURE]: optional q[+optional modifiers] apply after [TIER_0+TIER_1][TIER_2]
7. TIER_4 [TONE]: optional ~[+ tone definition] apply after [TIER_0+TIER_1][TIER_2][TIER_3]
8. the array of ([TIER_0+TIER_1][TIER_2][TIER_3][TIER_4]) phenomes[n] forms the [BASE_WORD]
9. The optional [k_FUNCTION] suffix is placed after the [BASE_WORD]
10. Each letter is read in their TIER grouping order.
11. q,k,v,w,x,y,z,~ Modifiers are always silent.
12. Modifiers never stand alone
13. Do not introduce additional whitespace.

SECTION 4.0 [AI_WORD_CONSTRUCTION_RULES]

Defines how words are constructed when not found in the dictionary.

- * MANDATE: The AI must construct the missing word not in the Dictionary. Nuxfonics spelling is based on perceptual pronunciation.
- * SYSTEM: AI Construction must follow the base letter system and modifier rules defined in this file.
- * NATURAL SOUND: The result should reflect how the word is heard in natural spoken language.
- * FLAGGING:
 - * - For AI constructed BEST_GUESS words, ie. missing from the Dictionary, they Must be enclosed in brackets with a *.
 - * - FORMAT [word*] in the output when required by system rules.
- * CONSISTENCY: Construction must remain consistent with Nuxfonics defined phonetic mappings.
- * IPA POLICY: IPA may be used as a starting reference internally, but IPA cannot be relied on, and must not appear in output.
- * PRIORITY: The AI must prioritize perceptual sound over written spelling.
- * ALIGNMENT: Constructed outputs must align with the overall Nuxfonics structure, modifier system, and assigned sounds.
- * ANOMALY PROTOCOL: If a sound is too far removed from an existing grid cell (e.g., when converting a new language), the AI must mark the specific position of the missing letter within the word using an underscore `\_`. The output must then be flagged as [MISSING*].
- * LIBRARY CONTEXT: This anomaly protocol applies specifically if the user indicates a new language is being converted or when making a New Library.

NOTE: There may be extra help on the specific language for word(s) being decoded/converted/constructed.

ALWAYS LOOK: These rules will be found in that specific language dictionary.

PURPOSE: This is to get a better language sound match for converting into the final Nuxfonics word(s).

SECTION 5.0 [HIERARCHY_RULES]

Defines rule precedence for interpretation and conflict resolution

1. DICTIONARY_PRIORITY

If the word exist in the dictionary use this.

2. CRITICAL_SOUND_RULES

Words are construction on perception of sound rather than strict IPA

3. STRUCTURAL_RULES

How Nuxfonics words are constructed.

4. AI_WORD_CONSTRUCTION_RULES

Word are constructed according to their native perception sound of the language.

End of Nuxfonics quick start initialization.

#####

--- End of Part 1 Break Point. Nuxfonics AI INI -- File Copy Past Splits

--- Start of Part 2 Break Point Nuxfonics AI INI -- File Copy Past Splits

#####

Start of Nuxfonics HOUSEKEEPING.

SECTION 6.0 [CRITICAL_SOUND_RULES]

Defines Non-negotiable distinctions that must be preserved during construction

No stacking (only one modifier per letter)

Other overrides for AI construction

Distinct oo sounds, distinction: uy != uz != oz (must not be substituted)

See [MODIFIER_SOUNDS] for full definitions of uy, uz, oz

SECTION 7.0 [PHONETIC_INPUT_PARSER]

[DIALECT_RESEARCH_PROTOCOL]

1. IDENTIFY: Upon the command "Convert [Language]," the AI must isolate the target dialect.
2. RESEARCH: If a word is not in the Local Dictionary, the AI MUST retrieve published linguistic data for that dialect, focusing on:
 - Resources like IPA for that language.
 - Vowel Inventory (e.g., Are 'a' sounds open/central [rx] or flat [a]?).
 - Phonotactics (e.g., How do terminal stops like 't' affect preceding vowels?).
 - Stress Patterns (e.g., Which syllable receives the energy [w] modifier?).
3. MAP: Translate the researched linguistic data into the 19x4 Sound Grid coordinates.

9.1 [MANDATORY_CONSTRAINTS]

If the dictionary is loaded then these will be defined in that dictionary.
If not present in the dictionary then [DIALECT_RESEARCH_PROTOCOL] is still used.

9.2 [PARSING_ORDER]

Defines how an input word is interpreted

0. IDENTIFY DIALECT: Detect language header (e.g., "Convert Bisaya") and activate [PHONETIC_INPUT_PARSER] profile
 1. Identify full word boundary
 2. Check dictionary (if found, use stored form)
 3. If not found, process using AI_CONSTRUCTION_RULES and the [DIALECT_RESEARCH_PROTOCOL] for the right input into the NUXFONICS_ENGINE.
 - If linguistic data is conflicting, ie same language, but from different districts, the AI must
 - Ask to clarify which District/Province is the language from,
 - If still conflicting, present the two most likely acoustic options and ask for user verification.
 4. Parse each letter / syllable / word, according to LANGUAGE DIALECT CONSTRAINTS, then feed that output to the according to the NUXFONICS_ENGINE and convert according to those rules
 5. REMEMBER to apply according to the LANGUAGE CONSTRAINTS:
 - If a the original language had a 'k' in it, it is to be treated as part of the word just like any other letter, according to that languages use. It is not a K_FUNCTION unless it is a nuxfonics original word.
 - Apply tone (if present)
 - Any Articulation Modifier (Palatal/Soft Sign) (q) to preceding unit
 - Apply length modifiers (v/w) to preceding unit
 - Apply word-level control (kv, kw), they MUST be at the end of the BASE_WORD
- # NOTE: The (k, kx,ky,kz) k_function_Homophones_type can only come from the Dictionary. Without that AI is purely dealing with the word phonetics.
NOTE: In virtually every time, without the dictionary, the word K_FUNCTIONS cannot be determined by the AI, so don't guess, juts leave them OFF.
NOTE: The applying of the K_FUNCTIONS come from the human word review and validation.

NOTE: For new word conversion it is strongly recommended for word integrity, that AI does not add any K_FUNCTIONS to a new word

6. INTELLIGENT SEQUENCE AUDIT: If the TIER order is incorrect (e.g., T2 before T1), the AI must:

A. Reorder: If the combination is unique and the intent is clear (e.g., aqx -> axq).

B. Best Guess: If ambiguity exists, select the most probable Nuxfonics pattern based on dictionary frequency.

C. Log Error: Flag the correction for human review without stopping the process.

SECTION 8.0 [VOWEL_STATUS_MANDATE]

PURPOSE: Remove English(and other languages)-orthographic bias regarding vowel function with in Nuxfonics.

1. VOWELS: (A, E, I, O, U, ...) are only relevant to the native language being converted to Nuxfonics.

2. NO VOWEL HIERARCHY: Once converted into Nuxfonics Vowels have no inherent "power" to

modify surrounding characters. They function exclusively as BASE units.

3. STRICT PHONETIC MAPPING: Vowel bases do not possess "long/short"

variants unless explicitly triggered by a Tier 1 or Tier 2 modifier.

4. SILENT ELIMINATION: Any letter that does not produce a distinct

Base + Modifier unit is structural "noise" and must be purged.

5. the (,...) is to indicate that different languages may have extra or different vowels, but these same still rule apply.

The Nuxfonics rules on sound perception rules allways apply.

SECTION 9.0 [DICTIONARY_PRIORITY]

1. If a word exists in the active language dictionary, use the exact Nuxfonics entry for that word.

2. If a word does not exist in the dictionary, AI is required to attempt to construct it, using [AI_WORD_CONSTRUCTION_RULES]

3. The AI must not override, alter, or reinterpret existing dictionary entries.

4. Dictionary entries are authoritative for the source language of the text being converted.

SECTION 10.0 [ARTICULATORY_FLOW_GUIDE_RULE]

PURPOSE: Ensure constructed words are physically easy for a human

to pronounce as much as possible in a continuous stream of speech.

This is more of a guide than a HARD RULE to help between multiple options that would essentially

give the same or similar word sound result.

1. THE GLIDE PRINCIPLE

```

# When two phonetic constructions are perceptually similar (e.g., ey vs
iy),
# the AI MUST select the one that allows the tongue and jaw to "glide"
# rather than "staccato" reset.
# Example: "fyiyrex" is preferred over "fyerex" (v-air vs v-e-r).

# 2. MANDATE: Prioritize the "Glide" sound over the "Stop" and "go",
# but this is not to be confused with syllables.
# [STRUCTURAL_PRECISION] has priority over the [ARTICULATORY_FLOW_RULE]
when
# the [STRUCTURAL_PRECISION] already is already close enough to satisfying
the [ARTICULATORY_FLOW_GUIDE_RULE]
# The AI must not "loosen" or simplify the spelling if the precise version
already achieves a fluid transition.

# 3. CONFLICT RESOLUTION
# In a tie between two accurate phonetic maps, the version with
# the highest "Mouth-Feel" (Ease of Articulation) is the
# Official Dictionary Entry and is closest to the [STRUCTURAL_PRECISION]

# -----
SECTION 11.0 [EXPRESSION_RULES]
# -----

# Defines non-phonetic expressive variations in speech delivery

# EXPRESSION_RULES are aliases only. See LENGTH_RULES and TEMPO_RULES for
definitions.
# EXPRESSION_RULES do not define mappings and introduce no new behavior.
# They only reference existing rules.

# Expression rules are optional and control delivery (tempo and length)
# Expression does NOT change spelling

# Letter level (length expressions): see LENGTH_RULES
# Word level (tempo expressions): see TEMPO_RULES

# -----
SECTION 12.0 [EXAMPLES]
# -----

# words that need more than the basic letter sounds
car = crx
cake = caxc
beat = bxt
day = dax

toy = tiz
look = lozk
moon = muyn

light = lixt
night = nixt

see = sex
sea = sexk    # same sound as English "see" but different meaning
jay = jx

```

```

bee = bx

# example words that can be spelt with basic letter sounds
cat = cat
dog = dog

# -----
SECTION 13.0 [TEXT_CONVERSION_RULES]
# -----

# unless otherwise stated when asked to convert, AI must follow these
rules.

# =====
# PURPOSE
# =====
# Defines how source text is converted into Nuxfonics phonetic form.
# This section handles linguistic transformation only.

# =====
# CORE PRINCIPLE
# =====
# - Conversion is PHONETIC ONLY.
# - Do NOT translate meaning.
# - Do NOT paraphrase, summarize, or reorder text.
# - Maintain original word order and sentence structure.
# - Each input word corresponds to one output word token unless a
dictionary entry explicitly defines a multi-word mapping.

# =====
# DICTIONARY PRIORITY
# =====
# - If a word exists in the dictionary, use the exact mapped Nuxfonics
spelling.
# - Dictionary entries override any phonetic guess.
# - Do not modify dictionary outputs.

# =====
# UNKNOWN WORD HANDLING
# =====
# - If a word is not in the dictionary:
#   -> Apply best-effort phonetic approximation.
#   -> Preserve syllable structure where possible.
#   -> Do not infer meaning.
#   -> Do not replace with synonyms.

# =====
# MULTI-WORD PHRASES
# =====
# - If a phrase exists in the dictionary:
#   -> Use the full phrase mapping.
# - Otherwise:
#   -> Convert each word individually.
# - Do not merge or split words unless defined in dictionary.

# =====
# PUNCTUATION PRESERVATION

```

```

# =====
# - Preserve all punctuation exactly as in the source.
# - Do not add, remove, or alter punctuation.
# - Includes: periods, commas, question marks, exclamation marks,
#   quotes, apostrophes, ellipses, and line breaks.

# =====
# GRAMMAR AND STRUCTURE
# =====
# - Preserve original grammar structure.
# - Do not change tense, voice, or syntax.
# - Do not reorder words or clauses.
# - Particles, markers, and function words must remain in place.

# =====
# AFFIXES AND MORPHOLOGY
# =====
# - Prefixes, suffixes, and inflections remain attached to words.
# - Do not strip or normalize grammatical markers.
# - Convert them phonetically as part of the full word.

# =====
# LOANWORDS AND PROPER NOUNS
# =====
# - May be preserved or phonetically transliterated.
# - Prefer phonetic representation unless a fixed dictionary entry exists.

# =====
# WORD COUNT CONSISTENCY
# =====
# - The number of output word tokens must match the number of input word
tokens.
# - No extra words may be added.
# - No words may be omitted.
# - Each input token maps to exactly one output token unless explicitly
defined otherwise in dictionary.

# =====
# PHONETIC CONSISTENCY
# =====
# - Apply consistent phonetic patterns across similar sounds.
# - Follow established dictionary spellings for similar phonemes.
# - Avoid introducing inconsistent phonetic variants.

# =====
# ERROR HANDLING
# =====
# - If ambiguity exists:
#   -> Choose the closest phonetic match.
#   -> Do not guess meaning-based alternatives.
# - Do not invent new semantic interpretations.

# =====
# VALIDATION RULES
# =====
# - No semantic translation.
# - No paraphrasing.

```

```

# - No restructuring.
# - Output must strictly reflect phonetic conversion of input text.

# STRICT MODE:
# Any violation of word count, order, or structure = INVALID OUTPUT

# ENFORCEMENT:
# The AI must validate output before returning it.
# If invalid, the AI must re-evaluate and correct the output.

# FAILURE HANDLING:
# If a valid result cannot be achieved after re-evaluation:
# -> Return the best possible output
# -> Append on a new line:
# [POSSIBLE NUXFONICS STRUCTURE RULES VIOLATION]

# OUTPUT RULE:
# The violation message must appear on a new line and must not alter
# the word count or structure of the converted sentence itself.

# -----
SECTION 14.0 [CASE_AND_FORMAT_HANDLING]
# -----

# =====
# PURPOSE
# =====
# Defines how the converted Nuxfonics text is presented visually.
# This section does NOT affect phonetic conversion.

# =====
# SCOPE
# =====
# - Handles capitalization
# - Handles formatting consistency
# - Handles layout and readability
# - Does NOT modify phonetic word forms

# =====
# CASE RULES
# =====
# - Preserve original capitalization style where applicable.
# - Sentence-initial words should retain capital letters if present in
source.
# - Proper nouns may retain capitalization based on source text.
# - Do not arbitrarily convert all text to uppercase or lowercase.
# - Do not alter case of internal word characters unless required by
source.

# =====
# PUNCTUATION FORMAT
# =====
# - Preserve punctuation exactly as output from Section 19.
# - Do not add spacing around punctuation unless already present in source.
# - Do not normalize or restructure punctuation placement.

# =====

```

```

# SPACING AND LAYOUT
# =====
# - Preserve original spacing between words.
# - Preserve line breaks and paragraph structure.
# - Do not merge or split lines.
# - Do not introduce additional whitespace.

# =====
# QUOTATIONS AND SPECIAL CHARACTERS
# =====
# - Maintain all quotation marks as in the source.
# - Preserve apostrophes and special characters without modification.

# =====
# OUTPUT CONSISTENCY
# =====
# - Formatting must not change the phonetic content produced by Section 19.
# - This section only adjusts presentation, not linguistic content.

# =====
# SEPARATION OF CONCERNS
# =====
# - Section 19 determines "what the text becomes"
# - Section 20 determines "how the text looks"
# - These sections must not overlap in responsibility

# =====
# VALIDATION RULES
# =====
# - Do not modify word spelling.
# - Do not modify phonetic mappings.
# - Do not alter word order.
# - Only apply visual formatting rules.

# -----
SECTION 15.0 [MATHEMATICAL_ISOLATION_RULE]
# -----

# PURPOSE: Protect the structural integrity of equations and variables.
# This rule overrides all phonetic conversion and modifier logic.

# 1. TRIGGER IDENTIFICATION
# The AI must enter "LITERAL MODE" if any of the following are detected:
# - Standard operators like: =, +, -, *, /, ^, <, >
# - Bracketed single-letter variables: (x), (y), (V), (A), ...
# - Standalone variables in an expression: x + y = z or like a = b + k

# 2. PROCESSING LOGIC (LITERAL MODE)
# A. HALT: Suspend all [AI_CONSTRUCTION_RULES] and [MODIFIER_SOUND_GROUP]
# parsing.
# B. BYPASS: Do not apply x, y, z modifiers to mathematical symbols.
# C. PRESERVE: Output the string exactly as it appears in the source text.
# D. RESUME: Return to Nuxfonics phonetic mode only after a space-delimited
# word that contains no mathematical operators is detected.

# 3. EXAMPLES
# Source: "Volume (V) = Area (A) * h"

```

Output: "Fyoluxm (V) = Aryx (A) * h"

4. CONFLICT RESOLUTION

If a letter like 'x' appears in an equation, it is a VARIABLE (Literal).

If 'x' appears attached to a Base Letter in a word, it is a MODIFIER (Phonetic).

SECTION 16.0 [DICTIONARY_ENTRY_FORMAT]

Each entry represents a mapping from a source language word to its Nuxfonics spelling.

// The storage fields of the dictionary words

struct DICT_WORD_ENTRY

```
{
    char Original_word[];           // The original language source word
    (written)
    char Nuxfonics_word[];          // the new spelling in Nuxfonics word
    (written)
    char Nuxfonice_word_syllables[]; // Pronounced word split into
    syllabales (how to say)
    char word_type_use[];           // noun, verd etc...
    char Comment[];                 // short meaning
    char English_translation[];     // the word translated into English
    int Entry_status_flags;         // Confirmed, to_be_checked,
    AI_Recalibrated on request
};
```

[ENTRY_STATUS_CODES]

0 = "Verified" - The entry has been verified by human, and to be considered 100% corerect.

1 = "Pending_Check" - These entries may not be 100% phonetically or semantically correct and should be reviewed by a human.

2 = "AI_Recalibrated"

3 = "Phonetic_Conflict"

[TSV_EXTORT_FORMAT] - output in copy past window, unless otherwise told by user of different format

For exporting the word dictionary the default tsv export format is:

Structure_Type = "DICT_WORD_ENTRY"

Fields = ["Original_word", "Nuxfonics_word", "Nuxfonics_syllables",

"Word_type", "Comment", "Translation", "Status_Flag"]

Delimiter = "TAB"

Encoding = "UTF-8"

Headings:

For the "Original_word" = "Original Language Name".

"Status_Flag" is text form of Entry_status_flags.

Other headings as per export Fields.

[DICTIONARY_FILE_FORMAT]

Rules:

- The AI must use '=' as the separator between source word and Nuxfonics spelling.

- Left side = source language word.

- Right side = Nuxfonics representation.

- All entries must be lowercase unless explicitly capitalized

```

# - Entries are processed line by line.
# - All entries must follow this format consistently across all language
dictionaries.

# Comments:
# - Anything after '#' is treated as a comment.
# - Comments are ignored by the AI during processing.
# - Comments may include:
#   - part of speech
#   - contextual meaning
#   - disambiguation notes
#   - pronunciation or usage guidance
# - The part_of_speech notes in comments are intended for human reference
only.
# - They may help distinguish entries during dictionary creation and
maintenance.

# AI Behavior:
# - The AI must rely on dictionary structure and sentence context (not
comments) to determine meaning.

# Verification:
# - An asterisk (*) at the end of a line indicates the entry is AI-
generated and not fully verified. Or text form as indicated
# - These entries may require human review for phonetic or semantic
accuracy.

# Disambiguation:
# - If a source word has multiple meanings or uses, multiple entries may
exist.
# - The part_of_speech note in comments may assist human understanding.
# - AI must refer to [WORD_DISAMBIGUATION_RULES] for selecting the correct
meaning in context.

# Example:
# learned = leynd    # adjective: knowledgeable *
# learned = learned # verb: past tense of learn *

# -----
SECTION 17.0 [WORD_DISAMBIGUATION_RULES]
# -----

# Purpose:
# Define how the AI determines the correct meaning of a word that has
multiple possible interpretations.

# Rules:
# - Some source words may have multiple meanings (homographs), e.g.
"learned" (verb vs adjective).
# - The AI must use sentence context to determine the correct meaning.
# - Context includes surrounding words, sentence structure, and grammar
patterns.

# Decision Priority:
#   1. Sentence context (primary)
#   2. Language grammar rules (secondary)
#   3. Dictionary entries and structure (supporting reference)

```

```

# Comment Usage:
# - Comments (#) in dictionary entries are for human readability only.
# - Comments may include part-of-speech or usage notes to assist humans.
# - The AI must NOT use comments as direct input for conversion decisions.

# Disambiguation:
# - If multiple dictionary entries exist for the same source word:
#   - The AI must select the entry that best matches the intended meaning
  in context.
# - If ambiguity cannot be resolved with high confidence:
#   - Choose the most natural/common usage for the given context.

# Consistency:
# - The same source word may map to different Nuxfonics outputs depending
  on meaning.
# - The AI must maintain consistency within a given context once meaning is
  determined.

# -----
# End of Nuxfonics HOUSEKEEPING.
# -----

# =====
# END OF FULL NUXFONICS AI INITIALIZATION FILE
# =====

```

=== End Of Book ===