Textual Buchla:

Re-Interfacing the Buchla 250e

Hugh Clery Ward

21316627

Department of Computer Science and Information Systems

Faculty of Science and Engineering

University of Limerick

Submitted to the University of Limerick for the degree of

B.Sc. in Music Media & Performance Technology academic year 2024/25

1. Supervisor: Jürgen Simpson

University of Limerick
Ireland

2. Supervisor: Dr. Robin Parmar

University of Limerick
Ireland

Abstract

This paper documents a practice-based research and design process, resulting in the development of a Live-Coding device, for the sequencing and organisation of electronic music, which foregrounds the conceptual metaphors embedded within the Buchla 250e Dual Arbitrary Function Generator. After contextualising of the fields of algorithmic music and Live-Coding, and detailing the history surrounding the Buchla 250e, the development in Max-MSP of several performance patches centred around Buchla's sequencing concepts is discussed over three sections. Finally, a performance patch, featuring a grid based textual interface for the efficient control of a device based on the 250e is introduced, and integrated into a performance practice which re-contextualises sequencing concepts developed by Buchla with new aesthetic materials.

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The thesis work was produced under the supervision of Mr. Jürgen Simpson and Dr. Robin Parmar at University of Limerick.

Hugh Clery Ward
Limerick, 2024

AI Declaration

I herewith declare that I have not used artificial intelligence to produce my project and/or report.
I further declare that I have discussed this use of artificial intelligence with my supervisor and received permission to use it.

Hugh Clery Ward
Limerick, 2024

Ethics Declaration

I herewith declare that my project does not involve human participants in any way and that I therefore was not required to submit an ethics application.

Hugh Clery Ward
Limerick, 2024

Acknowledgments

I would like to thank my supervisor Jürgen Simpson for his invaluable guidance and expertise throughout this research and extend my gratitude to my second reader and course director Robin Parmar.

Thanks also to: Rosemarie Clery, Colm Ward, Doireann Kennedy, Paddy Delaney, Matthew Casey, Luke Foley and all others whose company, assistance, and friendship throughout the duration of this research is greatly appreciated.

Table of Contents

1 Introduction

1.1 Overview

This research aims to develop a textually based control and programming interface for live performance of algorithmic music in Max-MSP. The research will present an analysis of the functions of various hardware sequencing devices, primarily the Buchla 250e *Dual Arbitrary Function Generator*, investigate its functionality, and question its efficiency and versatility in live contexts. Following this the research will investigate the porting of these devices and their conceptual metaphors from the physical hardware domain to a computer based and textually controlled one. This research will result in the iterative development of a performance patch which presents several variations and extensions of the Buchla 250e, utilising text as the primary user interface, and taking inspiration from other notable sequencers and Live-Coding paradigms. A resulting device should successfully convey the functionality of the 250e and be capable of programming a diverse range of musical algorithms as implemented by several notable composers.

1.2 Motivation

Initial motivation towards this research stems from a personal musical practice, primarily employing Max-MSP as a general-purpose tool, for synthesis, sketching of ideas and journalling. In centring a creative practice on this key tool, certain processes which Max encourages, or approaches to synthesis and sequencing, which are particularly effective in Max, have become canonical processes in this practice. However, after prolonged exposure to the recently installed Buchla 200e system in the University of Limerick EMS[1], the author of this work was quite struck by the system, in how esoteric but functional it was, and was interested in incorporating some of the 200e's features into max patches.

An earlier university project, which was influential to this work, was centred on the construction of a device in Max which implemented some of the 250e's core features, primarily the nested loop functionality and with an alternative approach to gate or trigger sequencing. The problem of how to interface the device remained largely unsolved by the project's end. Relying on skeuomorphic designs to mirror the physical interface seemed unwise when programming the physical module with its knob and encoder-based inputs had already proved nontrivial. After the project's completion, and while beginning to experiment with Live Coding languages, a textual approach to the 250e's interface began to seem appealing.

Aside from this technical motivation towards this research, reasoning for engaging in this work is also conceptually grounded: Mark Fell in the opening chapter of his book *Structure and Synthesis* (Fell 2022) presents a perspective on tools, grounded in the traditions of steelworkers, but which apply nicely to artistic practice, and particularly to that of the

[1] University of Limerick Electronic Music Studio: http://www.dmarc.ie/ems.html

computer musician. Fell describes how a steelworker, on graduating from an apprenticeship to the shop floor, would first have to craft a set of their own tools, necessary for whatever job they would fulfil. While Fell provides a list of practical reasons why this process is useful, he also notes a perhaps more significant and philosophical reason. In creating one's own tools, maintaining them, and continuing to use them, their influence on the user/maker is significant: The tools, when embedded into practice, inform and influence "how one understands, reasons, and thinks." (Fell 2022) This philosophy is embedded into Fell's practice and seems an enticing alternative to a purely theoretically informed practice.

Magnusson and McClean, while explaining the diversity of musical pattern languages that exist and their affordances, state that by developing a set of pattern manipulation functions "a coherent vocabulary is built up: one that will influence the music or style [of] the system." (Magnusson and McClean 2018). Here we see a similar process, but in a context much closer to this paper, the influence of tool on user, and in the case of someone developing a pattern language, tool on toolmaker. This thought process is particularly relevant to this research and is central to its motivation. By developing a language for the interfacing of the Buchla 250e's functions, an opportunity arises to foreground this device, its musical heritage, and develop a number of works which are both shaped by and respond to it.

By taking the Buchla 250e as a starting point, implementing its key functions and conceptual metaphors in Max-MSP, and designing a textual interface for control of this sequencer and other key performance elements, a useful and efficient interface for live performance and improvisation may be achieved. Through the development of this research, the design informed thinking and conceptual framework of the 250e may be moulded to a device, firmly integrated in the performance practice this work centres itself on.

1.3 Objectives

This research, through an iterative design process centred on live performance practice, aims to create a performance system based on a solely textual interface, which utilizes the conceptual metaphors and sequencing approaches found in the Buchla 250e. In producing this system, the goals of this work can be divided into several objectives as detailed below:

In the following chapter of this paper, relevant context to the field will be introduced, and several studies referenced, which will justify this projects approach, and inform the analysis of several contemporary tools. Contextualising the field in this way will ensure more informed design choices be made and outline the niche within the live coding field which this project occupies.

Analysis of three contemporary live coding tools will be carried out following this contextualisation and will also be extended to the 250e as well as the more contemporary RYK-M185 sequencer. This analysis will inform the design of the following system and will also serve as a process through which to evaluate the resulting system.

Both during the analysis phase, and to a more focused degree afterwards, prototyping of the new textual system will be conducted in Max-MSP. Early protypes will focus on the various implementation approaches for textual systems in Max, and later phases will focus more specifically on the design and syntax of this system. A continued cycle of iterative prototype, performance, and re-evaluation will allow for improvements in the system, and iterations on the design.

The development of prototypes during this research will be punctuated by several performance opportunities, which aim to foreground and evaluate the efficacy of the integration of the 250e's conceptual framework within this new context. This context, or more specifically, the pre-existing musical practice this research is grounded in, aesthetically is concerned with the organisation and synthesis and live performance of experimental and algorithmic music, taking cues from left field dance and noise music traditions.

In conducting this research, and developing several experimental devices, incorporating the conceptual framework leveraged by the 250e, this research more directly is addressing the question: How can the conceptual metaphors of the Buchla 250e be integrated into efficient textual interfaces, centred around musical performance practice?

2 Context & Literature Review

2.1 Introduction

This chapter aims to provide the essential context and reference contemporary literature most relevant to this research. With the goal in mind of creating a textual interface for the manipulation of Bucha 250e style sequencing in a live performance context, this section is divided into three key topics: Algorithmic Music & Sequencing, Live Coding, and an Aesthetic Context useful for the contextualisation of this work. Each section introduces the concepts necessary for the understanding of the field and includes literature with the aim of justifying this work's approach.

2.2 Algorithmic Music & Sequencing

In conducting this research, an indispensable resource has been the Oxford Handbook of Algorithmic Music (McLean and Dean 2018). While many of its chapters are important in the context of this work, the introductory chapter outlines the history of algorithmic music and provides a useful discussion about its contemporary issues and future directions. Importantly in this section McClean and Dean establish a very broad definition of algorithmic music as a "field of activity, defined by the urge to explore and/or extend musical thinking through formalized abstractions."(p.6) Following this definition, a discussion of early algorithmic music is preceded by questioning the very idea of a definitive 'canon' of algorithmic music, noting that algorithmic music from Africa, and from female composers are both under researched fields. Alongside this opening chapter, Collins later chapter (2018) provides a more in-depth history of early algorithmic approaches to music as discussed below.

2.2.1 A Brief Chronology

Collins' (2018) handbook chapter outlines the early history of algorithmic music and begins by acknowledging musical algorithms dating as far back as the 11th century. However, the chapter provides a much more detailed description of later algorithmic approaches, focusing on later 18th century process-based approaches[2] taken by Hayes (Hiller and Isaacson 1979, 52) and others, as well as the 17th century description of an options-based music composition system theorised by Athanasius Kirche. Collins continues with descriptions of Stochastic and sonification approaches employed by 20th century composers, which will be discussed later, However the following section on Musical Automation is particularly relevant within this research's context:

Collins describes the rich history of musical automata, referencing several 18th century devices and alludes to the influence of programmable looms on 20th century computing, and all that came after it. Collins further describes later musical devices, reliant on "drum-roll" sequencing which followed the development of the mechanical clock, and the "well established"(Collins 2018) tradition of automata building described by Cohen (2012). Notably Collins recites Bumgardner's (2013) description of "Winkel's mechanical Componium" from 1821 which used two rotating drums and random walks to play variations on a theme. It is also noted that Winkel's metronome, mass produced by Mälzel (Riley 2009) "brought home the mechanization of musical process" (Collins 2018, p88) and came to be mass produced, unlike Mälzels Componium, and at a far greater scale than the Player Pianos which followed in 1890. Collins concludes this section by observing that it was not until Conlon Nancarrow (Gann 1995) and his mid-century player piano studies, that the new capabilities of these machines to play far beyond human capability, were taken advantage of.

Finishing this section with Nancarrow, Collins brings us to the 20th century where algorithmic musical processes were integrated within new computer technologies and employed by several notable composers. From this point, Mclean and Dean's chapter provides useful details on 20th century algorithmic composers, while also highlighting distinctions between algorithmic approaches, contrasting Stockhausen's text based instructional approach with Xenakis's more pragmatic and detailed scores. Furthering their discussion of 20th century algorithmic music, the minimalist works of Reich, Glass, and Riley are highlighted, providing Reich's phasing technique as an example of a "manually composed" (in the case of clapping music and piano phase) application of an algorithm.

[2] Hayes's splattering of paint on a musical score to determine notes, as suggested in his 1751 pamphlet 'The Art of Composing Music by a Method Entirely New, Suited to the Meanest Capacity'

While this chronology excludes many important developments and figures involved with algorithmic music, it is easy to see the influence of this context on popular composition tools today. Be it the player-piano inspired piano rolls found in many daws today, or the dice-game like follow action probability functions found in Ableton (Hein 2021 224). Notably Collins (2018 89) mentions Kippen and Bel's research into algorithmic modelling of north Indian Music(1992) resulting in the Bol Processor Project whose syntax was influential in the development of Tidal Cycles (McClean 2014, McClean and Wiggins 2010). Collins also references Godfried Toussaint's paper (2013) and his discovery of the Euclidian algorithm's ability to synthesize several prominent world rhythms. Both Toussaint and Kippen and Bell's research into algorithmic rhythms have been highly influential on the development of hardware and software for electronic music performance[3].

In a later chapter, Magnusson and McClean (2018) explain the absence of pattern from much of the early electronic music produced with tape techniques, noting the difficulty of creating tape loops meant patterns were largely absent from tape music. What changed this of course was the introduction of voltage-controlled sequencers which allowed for the sequencing of repetitive patterns, one of the first of which was Donald Buchla's 123 Sequential Voltage Source (Buchla 1991 11). Buchla's sequencer, along with Moog's 960 Sequential Controller (Jenkins 2021) were the first manifestation of algorithmic composition approaches in electronic music hardware and were the first in a rich lineage of sequencer devices, whose role in the music of the late 20th century is difficult to overstate (Jenkins 2021). As Magnusson and McClean state later in their chapter:

> *"In modern musical software we typically find systems that derive their design metaphors —both in terms of interface and interaction design—from past traditions, such as the musical score, piano rolls, and the hardware sequencer. Musical patterning in such software is therefore still often subject to the hardwired mechanisms of historical physical equipment."* (Magnusson and McClean 2018 286)

[3] Bol Processors influence on Tidal Cycles syntax, and Various Eurorack and software implementations of the Euclidian rhythms algorithm

2.2.2 A Taxonomy of Sequencers

Taking the above history into account, the influence of early sequencers, as well as pre-computer approaches to algorithmic composition, on contemporary musical software is quite clear. However, as Jenkins (2021) notes, there are many categories of sequencer. Duignan, Noble and Biddle's "Taxonomy of Sequencer User-Interfaces" (2005) is particularly useful for this. This taxonomy classifies sequencing tools by distinguishing between the following characteristics.

Medium - Textual / Graphical

Abstraction Level - Predetermined / Custom

Linearisation Stage - Eager / Delayed

Event Ordering - Control / Data

Applicability - Special / General

For the scope of this work, the main characteristics of interest are *medium*, *abstraction level* and *event ordering*. With all the sequencing paradigms to be investigated being “general purpose applicable” and employing a “delayed linearisation” stage (as opposed to the fixed eager linearisation of timeline paradigms). While distinction between textual and graphical approaches is self-explanatory, it is also at the core of this work. However, the remaining two characteristics, are more complex:

Event ordering distinguishes between systems which rely on ordered events (which can include conditional functions and loops) and data-flow systems which “require the user to determine the final sequence in terms of data flowing through a computational system” and can be used to implement rule-based systems. (Duignan, Noble and Biddle 2015).

Abstraction level within this taxonomy references the “process of using classification, grouping, and hierarchy to hide and reuse details” to avoid users having to deal with low level concepts. A differentiation is drawn between *predetermined* and *Custom* abstractions, where predetermined abstraction is found mostly on domain specific devices such as drum sequencers. Most devices and approaches to be discussed here will employ custom abstractions, as is the nature of modular synthesis and live coding, but with a highly varying degree of abstraction depth. This depth of abstraction is important to consider within this research; Duignan Noble and Biddle (2015) note: *“a well designed and implemented set of customisable higher-level abstractions can add considerably to the power of a music application”*

While these characteristics are quite high level and perhaps more general than this work requires, they provide a useful starting point for the analysis of the various sequencing approaches to be described in this section. Considering these common characteristics of sequencers, as well as a set of music centred design heuristics (Bullock 2018) to be described later, a more robust approach towards the design of a text-based approach to sequencing may be achieved.

2.3 Live Coding

In creating a text based, typed interface for live algorithmic music performance, this research places itself firmly within the field of Live Coding. While this is a broad field and the merit of strictly defining such broad fields is to be questioned, Blackwell *et.al* in the introduction to *Live Coding: a user's manual* state:

> "*Broadly speaking,* (Live coding) *describes the improvisatory real-time composition of predominantly computer-generated audiovisual material, in which the writing of code itself (or other executable instructions) is presented as a live event for an audience.*" (Blackwell *et al.* 2022 3)

Live coding exists in the context of a variety of alternative programming practices (Bergström and Blackwell 2016) and is supported by TOPLAP (Terrestrial Organisation for the Pragmatics of Live Art Programming) who provide a draft manifesto (Toplap 2024) adhered to by many practitioners of live coding. The manifesto lists several recognitions and acknowledgements, but notably demands, among other things, "access to the performers mind", to be shown the screen of the performer, and states a preference for a level of abstraction within code: *"The program is to be transcended - Artificial language is the way"*

Blackwell *et al.* liken the live coding musician to "an improvising composer" (2022 5) more than to a traditional performing musician. Live coders while showing their screens and displaying their code - the very source material of their compositions, are foregrounding the algorithms and functions structuring the performance. Through manipulation of code over the performance, live coding can be seen as *thinking in public*, a popular characterisation of live coding practice.

In Live coding fields today a nearly endless array of approaches to sequencing are available to musicians. However, a programming language encourages the user to enter into a discourse with the *model world* the language has been formulated around (Rohrhuber *et al.* 2005) and thus, requires some conceptual framework around which to base itself. By centring this research on a hardware sequencer, and acknowledging the constraints it puts on users, it is hoped the resulting tool will successfully transfer the conceptual metaphors of the device to a new context, taking many cues from contemporary Live-Coding approaches, and deliver a robust framework within which, the user can be flexible and expressive.

2.3.1 Tensions and Techniques

In a later chapter of the Oxford Handbook of Algorithmic Music, Roberts and Wakefield (2018) in *Tensions and Techniques in Live Coding Performance* list a key set of five "tensions" used as tools to conduct an analysis of a range of live coding practices. This chapter serves a concise summary of many of the key elements involved in live coding performance, and provides a useful context to the reader, by understanding this field through five tensions, a framework through which to analyse several live coding tools later in this paper is established.

The first tension and one which applies to many performance practices is the balancing of risk and search for security and stability in improvised performances. Clowney and Rawlins (2014) note that risk is a part of all performance, however Roberts and Wakefield outline the heightened consequences of errors in live coding. Following this they note the allure of failure, in performances where risk is involved and explain the heightened sense of liveness which can be experienced by the audience, in performances acknowledging the error prone nature, or instability of code. It is also acknowledged that many performers, integrate this into performances, both expecting and welcoming mistakes, allowing them to influence their compositions. Practice, collaborative performance, and previewing, are listed as three popular techniques used to mitigate error in live coding performances.

A second tension outlined in this chapter is Legibility. Here both performer and audience perspectives are considered when debating the merit of showing audiences code or live visualisations, and designing languages intended to be legible by both parties, while remaining powerful. Freeman and Van Troyer consider the design paradox of designing audience legible live coding languages, which balance the efficiency and conciseness of syntax with the audiences "difficulty understanding text that is not sufficiently verbose"(Freeman and Van Troyer 2011). Roberts and Wakefield conclude with support for visualisation in live coding languages emphasizing the increased audience engagement it can achieve.

A third tension, one which exists conceptually at the core of this research, is abstraction – the supressing of "lower-level complexities in order to establish higher-level interaction". Roberts and Wakefield consider the range of abstraction levels preferred by live coders, where some coders require low level control of systems for Digital Signal Processing (DSP) manipulation, others avoid programming functions at this level entirely and rely on high level abstractions provided by languages to perform with minimal code. Importantly, abstraction level does not just dictate the style of programming but are highly influential on the cognitive process of the performer. Roberts and Wakefield quote Magnusson, who describes a language and the abstractions it provides as "a scaffold for externalising musical thinking" (Magnusson 2014). Following this they detail some of the "domain-specific mini-languages" which exist within languages such as SuperCollider and serve to embed "mental models of musical structure and process into abstractions at the level of programming languages themselves". Abstractions in live coding languages are often "grounded in metaphor" and several examples of this are provided in the context of McClean's *Tidal* and Magnussons *ixi lang*. These metaphors, and the abstractions which they are embedded within, are key concern of this research. This is emphasised by Collins and McClean, who note that abstractions substantially influence what kinds of ideas can be expressed, and what discourse can ensue.

The penultimate tension introduced by Roberts and Wakefield is *Flow and Pace* in which they consider the level of cognitive load undertaken by the performer and the pacing of ideas expressed in a performance. This tension is summarised as *control vs pace* and can be navigated with pre-prepared material, and high-level abstractions. The authors present a variety of individual approaches highlighting each coders unique balance between their control over a function's parameters, and the rate at which they can express ideas. The concept of flow introduced here will be revisited later and is an important concern in live coding performance.

The final tension introduced in this chapter is *time*. Here the role of past, present, and future is questioned within the context of live coding, and an alternative understanding of this time division as a "spectra of extended presents" is proposed. In this tension it is appreciated that live coding puts the performer in a unique position as opposed to a traditional instrumentalist, where the immediacy of acoustic instruments is lost but replaced with the ability to affect simultaneous change across multiple processes, and create processes with very little work, which extend far beyond the time domain traditional instruments operate within.

In appreciating each of these tensions, one gains an understanding of the elements of live coding, many of which are relevant here. The importance of the third tension, abstraction, is apparent. In porting the Buchla 250e to a textual system, this work aims to create a set of abstractions for the functions of this device and successfully embed the same conceptual metaphors in a new context.

2.3.2 Textual Interfaces for Musical algorithms

Another similarly relevant chapter in the Oxford Handbook of Algorithmic Music, *Designing Interfaces for Musical Algorithms* (Bullock 2018), presents a variety of approaches to interface design of algorithmic tools, introduces several design fields, and the new conceptual framework of "music centred design", and concludes with a series of case studies which present the different approaches to interface design. Importantly this chapter includes a case study on textual interfaces for musical algorithms. Bullock notes the potential when designing new interfaces for the control of algorithms, for a mismatch between the affordances of the interface, and the capabilities of the algorithm. The text-based case study section introduces several studies which provide useful context for this paper:

Nash and Blackwell (2014) introduce textual interfaces within the category of "digital music notations" as "virtuosity enabled" environments and provide a set of design heuristics for virtuosity which build on pre-established usability heuristics, while taking into account nonvisual feedback experienced in music performance. While it may not be possible to strictly adhere to these heuristics at all times, they provide useful guidelines for interface design:

> H1: Support learning, memorization, and prediction (or "recall rather than recognition")
>
> H2: Support rapid feedback cycles and responsiveness
>
> H3: Minimize musical (domain) abstractions and metaphors
>
> H4: Support consistent output and focused, modeless input
>
> H5: Support informal interaction and secondary notation
>
> (Nash and Blackwell 2014)

This study builds on Nash and Blackwell's survey of tracker users (2012) which suggests limitations of linear timeline paradigms in terms of liveness, and contrasts this with tracker software, which were found to be more effective and demonstrated *"how rapid edit-audition feedback cycles can be used to improve the liveness of notation-mediated interaction, facilitated by the development of motor skills using the keyboard"*. While this research emphasises the potential advantages of textual interfaces, Bullock (2018) notes that Nash's

heuristics can be criticised in that they “don’t necessarily generalise to other algorithmic approaches” outside of the domain of trackers. The solution proposed by Nash is the selective application of heuristics, only when appropriate and when considering the requirements of an algorithm or device, also considering music centred design concepts.

2.3.3 Contemporary Tools

For the purposes of this study, an analysis of three quite varied, but successful live coding paradigms will be carried out, each tool is introduced here. Each of these approaches to live coding involve their own unique syntax and conceptual paradigms. in providing the context surrounding these various live coding systems, it is hoped a greater understanding of the field can be achieved, while also introducing a number of approaches to pattern generation and programming syntax.

Tidal Cycles

Tidal Cycles is a Live coding environment originally introduced as a “pattern language for music improvisation” built in the Haskell programming language (McLean and Wiggins 2010). Tidal is a highly flexible and powerful language popular amongst many live coders[4] and is notable for the way it handles time, dividing time into cycles, where, by default a given pattern regardless of its length will play at the correct speed to take up one cycle. Tidal provides a syntax which allows for the quick typing of both complex polyrhythmic patterns, as well as more simple rhythms (McLean 2014), the power and speed of this language has led to its popularity amongst live coders, although Collins and McLean (2014) note that it prioritizes this speed over ease of learning. Tidal also provides several transformation functions, which are useful for the quick manipulation of patterns. Transformation functions like these exist in a number of live coding languages and often relate to some of the thirteen elemental pattern transformations listed by Spiegel (1981). Tidal’s patterns present an alternative way of sequencing events to most hardware approaches, but are important to acknowledge here, for their speed, flexibility, and terse syntax.

Orca

Orca is an esoteric two-dimensional live coding language developed by Hundredrabbits and aimed at generative composition. The Orca environment presents the user with a two-dimensional (2D) grid and relies on single character *operators* which serve as abstractions for specific functions, using the surrounding grid positions of an operator for inputs and outputs. Orca is notable for its extremely terse syntax, which may make it appealing to non-English speakers (Messina *et al.* 2021) and also its use of the base 36 system for storing numeric values. As well as a two-dimensional grid, Orca divides time into a grid of sixteen frames per bar, where every event triggered aligns with this grid. While Orca is undoubtably esoteric and encourages users to approach sequencing and algorithms in alternate ways due to the 2D programming paradigm, its’ extremely terse syntax and fixed time grid can be limiting, when

[4] See the many practice expositions featuring tidal in Blackwell *et.al* 2022

compared to more verbose languages such as Tidal (Linvega 2020). While Orca can be seen as somewhat of an outlier in the Field of live coding, its esoteric design is resonant of some of Buchla's alternate approaches to sequencing and synthesis concepts.

Teletype

The Monome Teletype is a compact computer for the manipulation of control voltages within Eurorack setups. Teletype employs a "simple esoteric language which aims at staying small and approachable" and is an attempt by Monome to "blur the music-programming barrier." (Crabtree 2024 464) The syntax employed by the teletype is terse and very efficient, while still offering the user a vast array of functions and the capability to create algorithms and scripts for sequencing and control of other modules.

While this work seeks to incorporate some of the conceptual metaphors of modular synthesis in a live coding environment, the Teletype does the opposite and incorporates the flexibility and functionality of code into modular synthesis setups (Crabtree 2024 460). Despite these seemingly opposed goals however, many features of the Teletype are influential in this research, and its ability to act as a single, textual controller for a modular synth style setup is alluring.

2.4 Aesthetic & Practical Context

This work conforms to a practice-based research methodology and takes cues from several contemporary works which have informed the aesthetic context this research is grounded in. The work of Mark Fell, already mentioned in the introduction to this paper, has had a notable influence on this practice. Particularly *Works in sound and pattern synthesis* (Fell 2013) presents a framework with which one can consider the creative thought, "being and action" involved in a practice as constituted within the technical environment as opposed to being distinct from it. This viewpoint accurately describes parts of this creative practice, which is primarily focused on performance with, and design of experimental devices in Max-MSP where an emphasis is often placed on exploring stochastic or generative process.

Aesthetically (and conceptually), the sonic outputs of performance devices, implemented later in this research, take inspiration and stylistic cues from several of Fell's works[5] alongside contemporary electronic works produced by Jessica Ekomane and Astrid Sonne or bass centric dance musics building on the Dubstep genre. Previous research and musical performances relating to this practice had also been concerned primarily with noise music, and the implementation of feedback systems, emulating techniques used by Toshimaru Nakamura, and taking inspiration from other artists such as Coil or Fennesz. The influence of the aforementioned artists is important to note here and has significantly shaped the practice, within which this research takes place.

[5] Also, Fell's collaborations with Mat Steel as *SND*

2.4.1 Max-MSP

Practical elements of this project, excluding contextual research carried out with live coding applications, are carried out in Max-MSP. Max allows for a wide range of approached to the implementation of devices for organising and sequencing events and dividing time, but the approach predominantly employed within this practice is *phasor based*. This approach utilises MSP objects and works by carrying out operations on a base phasor, which is representative of one bar or cycle. Points along the phasor can be assigned as triggers, and the phasor can be infinitely subdivided or operated on, while remaining in phase with the original clock.

Using a phasor-based approach allows for a finer resolution of time than working with midi or max-event triggers and allows the use of gen~ to create sequencing and time manipulation devices from low-level DSP components as will be illustrated later in this research.

Aside from sequencing approaches, max is also used in this research as a synthesis and sampling environment, where sequencing devices can be tightly integrated with sonic outputs, avoiding any reliance on external standards such as MDI, or communication protocols such as OSC. All sonic outputs employed by the performances related to this research process, were devices constructed in Max-MSP.

2.5 Conclusion

This research is an investigation into the 250e and its functions, and an attempt to create a new set of abstractions within a textual system, within which are embedded, the conceptual metaphors of the 250e. Taking Nash and Blackwell's (2014) Music Centric Design heuristics, and Duignan *et.al* 's (2005) sequencer taxonomy, a more informed analysis of the 250e sequencer can be carried out, as well as an evaluation of the three contemporary live coding tools detailed in this section.

The context detailed in this section is not exhaustive however, particularly the algorithmic music section excludes a great number of 20th century composers and approaches to algorithmic composition, a slightly more detailed account of this period is available in Magnusson and McClean's chapter in the Oxford Handbook of Algorithmic Music (2018). Similarly, a great deal of information relating to algorithmic music from non-western backgrounds has not been mentioned, see for example Matthews *Algorithmic Thinking and Central Javanese Gamelan* (2018).

3 The Buchla 250e

This chapter further contextualises this research, Introducing the device at the core of this work: the Buchla 250e, placing it within the lineage of important devices which led to its development, and outlying the importance of the individual and company responsible for it and many other notable devices. Following this, a critical analysis of the 250e is presented, concluding with the primary questions this research is concerned with.

3.1 Donald Buchla

In the first chapter of *Modular Synthesis,* Gordon (2024) presents a concise summary of the history of Donald Buchla's company and his personal philosophy in relation to musical instruments and their potential. Gordon describes the emergence of Buchla's ethos within the countercultural scene operating in California during the 60s and 70s and illustrates how the west vs east coast synthesis dialectic began. Gordon continues to explain that Buchla was keen to avoid the term "synthesiser" when labelling his instruments, and through dissection of a poem submitted to Synthesis Magazine, by Buchla and MacDermed (1971), illustrates that Buchla's ethos towards musical instruments posed the combination of ""hardware computer" and human user as a "biocomputer", which function together as a "systematic circuit"" (Gordon 2024 referencing Buchla and MacDermed 1971).

Pinch and Trocco (2002, 52) describe Buchla's approach as "anticommercial" in comparison to instruments designed by Bob Moog, who began making and selling instruments at the same time as Buchla. Gordon (2024, 73) continues to differentiate between the west and east coast approach by outlining Buchla's desire to produce instruments which enabled "the psychedelic discovery of possible new pathways for sonic and electronic signals." In focusing on this approach to designing modules, Buchla and Moog's approaches grew further apart as the synthesizer market grew.

While Buchla's designs were esoteric and less marketable (Pinch and Trocco 2004 52, referencing Moog 1975), their flexibility was appealing to "a certain type of composer" as noted by Vladamir Ussachevsky in the same chapter by Pinch and Trocco, who also note Suzanne Ciani's preference for Buchla's systems over Moog's due to the increased depth of control available, taking the three dimensional sequencing[6] ability of the *Multiple Arbitrary Function Generator* (MARF) as an example. However, despite their strong appeal within certain communities, original Buchla systems were never mass produced and were prohibitively expensive for many musicians, meaning many systems remained only in academic contexts. Thus, the influence of Buchla's design ethos was somewhat limited during his most experimental periods and before the introduction of the 200e series in 2004 as well as the more affordably priced *Music Easel* by Buchla Electronic Music Instruments in 2012 (Gordon 2024). Alongside the *Music Easel* system, Eurorack versions of Buchla 200 modules

[6] A feature also available on the 249e and 250e *External Input* section.

produced by Tiptop Audio (‘Buchla & Tiptop Audio’ 2024) have made incorporating Buchla’s designs into modular setups, far more accessible.

Buchla's approach to sequencing has developed following a clear lineage spanning a number of influential devices, beginning with one of the first sequencers (Buchla 2008), the *123 Sequential Voltage Source* (Buchla 1991), to the stored voltage section on the electronic music box, MARF to the 250e and 251e. Each of these devices present a development on Buchla's approach to sequencing and are milestones along the iterative design process of Buchla's influential function generator devices. However, it is important to note that none of these device’s names include the word sequencer, this is further evidence of Buchla’s esoteric design process. The more complex of the aforementioned modules are referred to as function generators which highlights these devices versatility. The MARF in particular was regarded as a useful and powerful tool (Resident Advisor 2024) which "offered more subtle controls" (Pinch and Trocco 2002) and functions as both an envelope generator, LFO, and sequencer (Multiple Arbitrary Function Generator n.d.)

3.2 The *Dual Arbitrary Function Generator*

The Buchla 250e Dual Arbitrary function generator is a 16-stage Control Voltage (CV) function generator and is part of the Buchla 200e system, released in 2004. the 250e can be used for a variety of applications, including CV and pulse sequencing, as well as envelope generation and even audio rate or low frequency oscillation. The 250e occupies an important place in many 200e systems due to its versatility and deep functionality and is recommended by Buchla in many 200e configurations (Buchla 2025).

Each of the 16 stages can store two CV values, a time value, two pulse values, and additionally sample and store voltages from four external inputs, which can in turn control the CV or time value of a stage. Each stage operates according to one of several advance modes which determine how the sequence should advance, either by depending on timing values, or by relying on CV sent to the *start/adv* input. Importantly, each stage can also store a loop value, which allows one stage to jump to another, enabling the nested loop behaviours the 250e is known for. CV and time values can be quantized and scaled to several preset ranges and interpolation is also available to enable smooth transitions between values. Every setting on the 250e can be modified for every stage, allowing for the implementation of complex sequences, and stochastic patterns, relying on external inputs and other 200e devices.

Aside from the unique combination of functions the 250e offers, the user interface of the unit is also visually distinct. Control voltages and timings can be edited using the mandala like circular array of input knobs which are surrounded by various compartmentalised control sections, responsible for the editing of each stage’s parameters. The array of 16 CV encoders serves to program both CV outputs, the editing of which can be toggled between using a circular button, one of 19 on the interface. Editing of *jump to* and *loop* values takes place in the centre of the interface and relies on a single encoder and button to toggle between the two values, both of which are displayed on a small LCD. Beside this is the important *edit* button,

which enables the editing of any non-encoder-controlled parameter of the device, such as the advance mode, or quantisation of a stage.

3.2.1 The 249e and 251e

The view of the 250e interface as slightly clunky and awkward to program may be reinforced by an earlier now obsolete module, the 249e Dual Arbitrary Function Generator. The 249e employed the same nested loops approach to sequencing as the 250e while also touting a higher stage count, a second voice, and additional probability functions, in the same footprint and with less UI, lacking the 250e's significant array of knobs. Only a limited number of 249e modules were made before it's discontinuation (Buchla 2006) and while it functioned as a very capable sequencer, its interface challenged users and proved to be a lot of work to program, as one reviewer for Sound-on-Sound Magazine stated:

> *"The possibilities were enormous, but I'm not sure whether it was worth it, because the amount of work involved was horrendous"* (Reid 2006)

While the 250e made significant improvements to the interface of its predecessor[7] (Clark 2016 00:01:03), programming and modifying nested loops and sequences on the fly remains somewhat cumbersome and is an action rarely carried out in a performance context. After the discontinuation of the 249e, the 251e quad arbitrary function generator was also released, which utilises the familiar nested loop system of the 250e and its predecessor. While each of these sequencers offer the user a great deal of utility and function well within the context of the 200e system, this research proposes that the various permutations of this approach to sequencing highlight the difficulties in designing a physical interface for programming sequencers of this style. While preset managers[8] propose a solution to the difficulty of programming loop structures on the 250e[9], this may not be a satisfactory solution for users wishing to improvise, as many computer musicians and live coders do.

The iterative process seen here leading to the 250e is inherent to Buchla's design ethos (Gordon 2024, 72) and has clearly led to improvements in Buchla's modules. When reflecting on Buchla's history of experimentation with function generators and voltage sources, it is clear his idea of what these devices were, never remained static. While unfortunately Buchla is no longer alive today, the experimental and iterative design ethos behind his devices raises the question: what further iterations on the 250e's functions and design, may offer the user a more fluid experience?

---

[7] See edit mode, button press and hold while stage selecting technique.
[8] Such as the 225e or 206e, one of which are recommended by Buchla for all 200e systems.
[9] Preset storage also features on 250e inspired modules such as the frap tools USTA.

3.2.2 The 250e's importance

While this research will argue that the interface of 250e can be improved upon, especially for usage in live contexts, it is undoubtably a powerful and influential device. Buchla himself in a 1991 interview outlines why he believed his devices, and the approaches to music they encapsulated, would retain their relevance:

> "*My approach has always been to take knobs and say, "this is what I want this knob to do, what the technology behind it is, whatever is most efficient at this time." I don't care, whether it's transistors or ICs or infrared or whatever. The function comes first and then the technology. By using that approach I believe I've built instruments that are not easily technologically obsoleted*" (Buchla 1991)

In this statement, Buchla differentiates his design process from that of his more commercially successful competitors and explains his designs as being led by a concept, as opposed to what was thought to be technically possible. What Buchla is saying here, well before the beginning of the so called "Analog revival" (Pinch & Trocco 2002, 317) rings true today, where Buchla 200e systems (and tiptop 200 modules) are still in production, despite being based on designs in some cases over 40 years old, and which follow a lineage of hardware beginning in the 1960's. The strong conceptual grounding of Buchla's designs has ensured their continued relevance in the field of modular synthesis but also serves as a justification for the transferring of these concepts into new contexts such as live coding.

3.2.3 Contemporary Relevance

One of the most outspoken artists on Buchla's designs is Suzanne Ciani, who became known for her work with the 200 system in the 70s (Ciani 1976). Ciani is a strong proponent of the MARF (Ciani 2024) and uses a clone of this device alongside modern 200e modules in her performance setup, her contemporary usage of the 251e and MARF together are well documented.
The lineage between the 248 MARF, the 249e and finally the 250e is quite clear, many functions introduced on the MARF are also present on these two later devices. However, expanding the functionality of the MARF while reducing the device footprint has resulted in a more densely packed interface, which may represent design compromises inherent in the 200e system which stand out of place amongst Buchla's earlier designs. Speaking to this point, Ciani is critical of the 251e, and notes the difficulty of programming and manipulating its patterns in live performance (Ciani 2016). Despite criticisms of the 200e system, Ciani's performance setup, largely reliant on the MARF, foregrounds the value and flexibility of the functions this device offers, and by extension, the relevance of the conceptual metaphors embedded within the 250e. While this device may be shortchanged by its interface the unique utility and function it provides musicians, make for an engaging and rewarding experience.

While contemporary Buchla users such as Ciani, or Floating Points (Shepherd 2019) highlight the relevance of Buchla's design ethos, the influences of his designs are evident in the devices of the Italian company *Frap Tools* who's documentation (Fabbri 2024) contains several references to historical Buchla designs. Their USTA sequencer, while not a clone of the 250e takes clear design cues from Buchla's device, notably the circular array of 16 knobs, central

control section and nested loop functionality while implementing many other features in a more modern interface. While the USTA offers higher level structuring tools unavailable on the 250e, the conceptual framework behind these two devices is undeniably similar and the two devices even share some nomenclature[10]. The influence of Buchla's designs on contemporary Eurorack devices such as those produced by Frap Tools, evidence the continued relevance of esoteric approaches to sequencing and synthesis such as those encapsulated within the 250e and 200e system. In many ways USTA accomplishes in the context of Eurorack what this research proposes to investigate in the context of laptop performance: further experimentation and iteration on Buchla's designs, developed for a new demographic and context.

3.3 Critical Analysis

Although this work is not grounded in a background of UI or UX research, personal experience with the 200e system, as well as evidence gathered online, evidence a legitimate critique of several functional and interface components of the 250e. While the importance, influence, and deep functionality of the 250e is clear, this device, along with the 200e system as a whole is not free of design flaws, usability errors or awkward implementations of feature or interface; The 250e, much like the rest of the 200e series, is “feature packed” (Reid 2005), however in presenting the user with such deep functionality, this “extremely complex” (Buchla- 250e n.d.) module may challenge many users.

> *“The 250*(e)*, I considered to be to some degree non-perform-able”* (Ciani 2024)

Further evidence of user frustration with the interfaces of the 250e, 249e and 251e is not difficult to find on online forums and reviews. These devices outline a timeline of the development and expansion on Buchla’s concept of what a function generator is but in doing so, also serve as an illustration of the difficulties and intricacies of sequencer design itself. Below are outlined the key criticisms of the 250e, central to this research:

3.3.1 Interface

From personal experience with the 250e, difficulties with the interface centre mostly around the ability to view only one stage's settings at any one time. This lack of higher-level visual feedback necessitates the investigation of various stages’ parameters in order to understand the motions of the 250e in a given configuration. While feedback is delivered through lights on the device, it can be difficult to gain an understanding of the devices parameters, particularly at higher clock rates. This issue is compounded by the multi-layered programming approach used on the main CV encoders, which do not reliably represent the CV data they are used to encode once the user switches to a different layer. Often after multiple edits, the state of the 16 encoders do not fully represent either array of 16 values, adding an uncomfortable element of guesswork into live manipulations of CV data.

[10] Reference to *stages* as opposed to the more common *step* terminology

Alongside this complaint, a second most irritating factor while programming the 250e is the cycling through of preset values while in edit mode, particularly the voltage and time ranges and advance modes. This process of cycling through values can only be done in a forward direction and slows down programming. When approached with haste, this process often results in an incorrectly entered value, further interfering with the performance and frustrating the user.

While engaging with the external input functionality on the device, this lack of feedback, and the reliance interface on buttons creates further difficulty. The 250e's separation of stage mode and stage value, while flexible and useful, adds a layer of complexity when programming CV or time channels. To achieve a given configuration for a stage, multiple parts of the interface are brought into play, forcing the user either to work sequentially or to use both hands to select the input or time mode and manipulate the value knob. The 250e does offer a useful convenience where the mode button can be held and the selected mode applied to multiple stages by sweeping the edit knob through the targeted positions. This convenience lends itself to a prescriptive method of programming as opposed to an experimental one, where the user has a patch pre-conceived in their head, and programming is simply the process of translating that pre-conception to the device.

This method of programming evokes what Fell (2013) refers to as a *cartesian world view* in relation to practice, which draws a distinction between creative thought and technical implementation. Fell argues that this distinction in the context of many creative processes is undesirable and inaccurately represents many creative practices. Understanding the mode editing convenience the 250e offers according to Fell's categorisation, it is apparent that it may not be entirely conductive to experimental programming.

In summary, the primary shortcomings of the Buchla 250e appear not relate to the functions the device offers the user, but the interface and abstractions through which they are programmed. The mode and range settings the device relies on are core to its flexibility, but prompt questions as to how they could be programmed or implemented differently, allowing for a more streamlined programming experience. In addition to these, the led-based feedback mechanisms employed by the interface of the 250e often result in confusion, or the interrogation of device settings to rationalise the motions of the playhead.

This apparent shortcomings of the 250e, the intricacies of sequencer design, and the difficulties of iterative design process may be more easily illustrated through comparison with another notable modular device:

3.3.2 Iteration and the RYK-M185

The 250e is a device capable of replicating the functions of many more popular contemporary sequencers. The influential RYK-M185 sequencer, for example relies on variable step length and a variety of gate modes, all with physical sliders and switches per-step, to create a device which rewards the user with tactile interface and offers a "brilliantly engaging way to play with a melody" or pattern (Vincent 2020). While the 250e can replicate the patterns produced by the RYK-M185, the experience of playing and manipulating Buchla's device does not provide the user with a similarly engaging experience. In 2012 Intelligel released the Metropolis, their version of the 185, which was replaced by the highly successful[11] Metropolix in 2021. By expanding on the functionality of the original RYK-M design, Intelligel engages in an iterative design process similar to that of Buchla's. By contrasting the development of Intelligel's sequencer and Buchla's post-MARF function generators, the difficulty of increasing the density of functions while retaining a usable and playable interface can be appreciated:

Intelligel's Metropolix sequencer offers the user a density of functionality at least on par with that of the 250e, allowing the Metropolix to serve as an essential component in many Eurorack systems. However, in aggressively expanding on the functionality of the M-185, Intelligel's sequencers managed to retain the inherent playability of the device they were based on by retaining the core parts of the original interface and relying on more complex interaction models, only for higher level functions less frequently used in performance settings. This is an approach Buchla did not take when developing 200e function generators, abandoning the large format, left to right structure, and distinct control sections of the MARF, for a new circular sequencing paradigm which lacks the structure and hierarchy of its predecessor and relies on more complex interaction models for many crucial functions.

While making this comparison, it must be noted that the Metropolix was released more than a decade and a half after the 200e system and takes advantage of many technologies not available in 2004. The usability and live performance shortcomings of the 250e may be excused to some extent by the ambitious goals of the device, especially in 2004, but regardless of this, the apparent flaws of the device provide valuable insight into the potential pitfalls of designing highly flexible, feature rich sequencer devices. Analysis of the 250e in this context, provides useful design guidelines for the textual system this research is investigating.

[11] Both Metropolis and Metropolix rank in Modular Grid's *Popular 100* list

3.4 Conclusion

The 250e is documented as being a highly flexible albeit esoteric device for function generation, or CV sequencing. While this device has been successfully employed by many artists, and its flexibility is celebrated, it's worth does not come from its function alone, but also from the context it exists in. The 250e, and the Buchla 200e systems which it usually operates within, draw on a lineage of heritage hardware and the design ethos which lies behind it. Don Buchla, and his devices, despite being made in limited quantities, came to be hugely influential, and would come to define one of the two initial schools of thought around synthesis in America. Buchla's constant experimentation resulted in devices and approaches to synthesis which were unique in the markets they existed in and so came to be influential on future designs of synthesis hardware and software which also sought to be experimental. For example, some of Buchla's earliest systems were commissioned by Morton Subotnick for the San Francisco tape music centre (Buchla 1991), a place of undeniable cultural and historical importance in the history and development of electronic music. Buchla's original system here went on to be used by many important and influential composers.

However, as the criticisms in this chapter have illustrated, the 250e is not without flaw. It is clear, in the experience of this author, that interactions with the interface of the device often necessitate convoluted manipulations of controls, accompanied by a dissatisfying lack of visual and tactile feedback. By taking the criticisms levelled here into account, the function and utility of the sequencing concepts embedded in the 250e may be re-encapsulated into a device and interface which fits more seamlessly into the creative practice this research is grounded in. While this research process does not propose a UI/UX informed solution to issues inherent to the 250e, the iterative and practice-based process proposed here echoes parts of Buchla's own methodology and aims to extend the lineage of experimentation and contemporary practice associated with his devices into a new context.

In responding to the issued raised in this chapter and providing further context surrounding the 250e, a more clearly defined response to the research question can be formulated.

4 Re-Interfacing the Buchla 250e:

4.1 Introduction

This chapter will discuss the development of this research (in response to the analysis of the 250e and context outlined above) in the context of the performance-based arts practice it is rooted in. The design process will be introduced and linked to practice, following this the development will be split into three distinct sections, which represent the chronological process through which the research was actualised. Finally, the resulting devices, findings and overarching research process will be critically discussed.

4.1.1 A Practice Based Design Approach

This research, grounded in the personal musical practice detailed above, took advantage of multiple performance opportunities available during the period it was conducted. These performances, organised by the CSIS faculty and members of the DMARC research centre at the University of Limerick, provided a unique opportunity to engage in an iterative development process, where prototyping was followed by test performances and the resulting experience and feedback integrated into the design of the following prototype. This process involved four performances spanning from November 2024 to April 2025 which were crucial to the development and advancement of this research.

It is important to acknowledge the practice this research is rooted in is primarily concerned with live performance and improvisation. While the Buchla 250e is a powerful device, it is rarely seen in live performance context, due in part to its cost and size, but perhaps also as a result of the difficult to program interface, and confusing user experience. While the re-implementation of the Buchla 250e carried out here is not grounded in a UI/UX approach, by developing this research through multiple performances, it is hoped the resulting device has been shaped by and integrated within this performance practice and perhaps may lend itself more suitably to the performance of live improvised electronic music. A successful device will function within the parameters of this practice and is not necessarily a general-purpose tool, useful for all Max-MSP users.

As this research is heavily focused on the technical integration of Buchla's concepts into a pre-existing practice and aesthetic context, this work did not propose the development of a new set of synthesis or sampling tools to accompany the new device. Instead, the following performances relied largely on pre-existing devices, which had been leveraged before as part of earlier performances and had been proven to function well in the context of this practice. While these devices could be modified, relatively little new aesthetic material was introduced.

4.2 Development 1: *The REPL*

Development for this work began with an older event-based [12] implementation of the 250e's nested loop function and a motivation to move from a frustrating *multislider* based programming interface to a purely textual approach. This early event-based and textual approach implemented a simple Read Evaluate Print Loop (REPL) interface and utilised the *coll* object to store the arrays of data used by the sequencer. Three values *JUMP*, *LENGTH* and *LOOPS* were extracted from the coll when a given stage was polled by the user and were then printed and modifiable. A terse syntax was employed where a stages data could be requested using a slash and then programmed using *J*, *L* and *N* characters.[13]

This simple setup mirrored the 250e in only allowing one stages' data to be viewed or edited at a time but improved on the earlier slider-based interface by allowing for the input of precise integer values and quick navigation through stages with the keyboard.

While this interface only allowed for manipulation of the most basic parameters of the 250e's loop structure, it served as a proof of concept for further development of the work and research into textual performance interfaces. The REPL interface implemented here was reliant on the user's knowledge of the underlying system but provided a faster way to navigate and manipulate data. As well as this, the REPL offered a system which could be scaled to offer control over an entire patch, which although outside the scope of this research, was an attractive concept.[14]

Following proof of concept, further experiments with REPL for live performance were carried out. A patch suitable for performance necessitated the development of a more flexible system, utilising the Max *dictionary* object, which stores all the systems' data and scales easily to facilitate control of multiple devices in a patch. This prototype stage was isolated from the conceptual focus on Buchla's function generator paradigms and was concerned solely with experimentation with various types of syntax for the control of a range of devices and sequencers. These outputs included a simple variable stage length three channel drum sequencer and an emulation of the RYK-M185 sequencer[15], alongside a generative breakbeat chopping device and various audio effects.

[12] As opposed to the phasor-based approaches to be explored later in this research.
[13] For example `/5` may return the values `J4 L1 N1` (Jump 4 Length 1 NumberOfLoops 1) which could then be edited and re-submitted to the REPL.
[14] As is evidenced by the inclusion of a built in REPL system in the release of Max 9.
[15] Based on Philip Meyers YouTube tutorial implementation.

4.2.1 REPL Performance

It was with this patch that the first performance opportunity of this work arose; A short 10-minute performance in a local venue, in collaboration with a classmate responsible for live coded visuals. This performance was affiliated with a Live-Coding module completed in concurrence with this research and so cannot be considered a direct component of this work. However, the experience and feedback derived from this performance, reliant on text as the sole interface, were influential on the development of the later textual system associated with this work. This performance served as a further proof of the validity of textual approaches to performance in Max and allowed for a useful critical analysis of the REPL system in use at this point.

The resulting performance leveraged the simplified M185 device sequencing an FM-bass voice alongside a simplistic variable step length drum sequencer to create a polyrhythmic interplay between drum and bass sounds. Alongside these elements, which served as the core of the performance, several semi generative, parameterised devices provided both harmonic FM accompaniment through a semi random just-intonation based recursive chord sequencer and additional percussive elements, through a glitchy drum break chopping device. Each of these devices were controlled primarily by the REPL, through which changes to sequencer controls, or voice parameters could be made sequentially, although multiple commands could be executed simultaneously using an apostrophe to delineate commands.

4.2.2 Analysis

While this performance was a success, the experience prompted several questions relating to the REPL, which provided only one line of feedback per user request. i.e. only one array of data could be polled at a time: in the case of the M-185 sequencer `GATE`, `PULSE` or `PITCH` arrays. This system relied on the user to recall every command from memory and carry with them a strong knowledge of the underlying syntax and functioning of the specific REPL implemented here. While this system functioned well with the sequencing and synthesis devices it relied on and resulted in a successful performance, it did not communicate any of the constraints inherent within itself to the user, relying on knowledge built through practice and the construction and design of the system itself.

Although multiple instructions could be submitted to the REPL simultaneously, they could only be submitted blind, without first polling the dictionary to see the array or data that was about to be changed. It was clear, although this performance had been successful, that the single data point editing system would not scale well for full control of all the 250e's features, and somewhat ironically mirrored the slow to program single data-point loop editing interface present on the 250e.

4.3 Development 2: *Buchla In gen~*

4.3.1 A robust model of the 250e

Following experimentation with prototype REPL systems, the immediate focus of this work became the development of a more robust and accurate emulation of the 250e, using a phasor-based approach as opposed to the earlier event-based system. This device was constructed within the gen~ environment embedded in Max-MSP, which allows for single sample feedback loops, essential in allowing the device to cycle at audio rates, as the 250e can. This device, unlike earlier iterations, sought to implement all the 250e's functionality, and would serve as the building block for experimentation with additional functions and modified behaviours in later stages of the research. Note at this point, the device developed in gen~ will be referred to as the *b250*, as distinct from the physical Buchla 250e module, referred to as the *250e.*

Practically, this involved a re-implementation of the nested loop functionality alongside two data and two trigger channels, external input sampling for data and time channels as-well as four advance modes and the stage addressing section of the 250e. While unlikely that all these features would be concurrently utilised in a patch, their presence in the 250e is key to the functionality and flexibility that allows the device to play a core role in many 200e systems. Omitted from the gen patch are the quantising and voltage range functions. While these are essential controls on the 250e, in the context of a max patch, their implementation outside of the b250 device may allow for greater flexibility and a more condensed UI.

As gen~ is a DSP environment, it cannot deal with traditional lists and instead relies on buffers, which reference external buffer~ objects outside the gen~ patcher. Each of the per-stage parameters of the 250e were assigned a buffer which could be populated with values outside of gen~ by the REPL. In addition to this, the global parameters such as external-in-hold and stage-address-mode were assigned an address in a parameters buffer. While the efficiencies of code could allow a much larger stage count then that of the 250e, or even the 50 stages of the 251e, a stage count of 16 was used throughout this research for simplicity.

This gen~ device, would allow for later experimentation with multiple voices, additional data channels, probability functions, and scalable interpolation between values, and could serve as a test bench, enabling further experimentation and expansion on the feature set of the 250e.

4.3.2 New REPL

This following stage of the work involved the development of a fully featured performance system, featuring the b250 device and REPL interface. Alongside the already complete b250 module, a more substantial REPL interface was to be implemented here which would assume control over all the data and parameters of the function generator. As well as this the REPL would be responsible for control over other devices in the patch, which may function as external inputs into the b250 or respond to the devices’ outputs.

Features & Syntax

The redesigned interface sought to provide more context to the user and enable new data editing efficiencies, without impeding on the flexibility of the device. This involved a re-design of the REPL, enabling the display and interpretation of all a devices parameters' simultaneously, and allowed the interface to display multiple devices in the textedit, parsing them sequentially.

The result of this was a simple plain text representation of a devices' data and parameters, where each line presented the name of an array, followed by its values, separated by a space and closing the array with a semicolon. Devices were delineated by the name of the device followed by a colon, where every following value was a child of the above device until the next device was called. This syntax was essentially a terser representation of the max dictionary syntax and was parsed from textedit to the dictionary and then back to the textedit using a combination of dictionary, list, string and regular regexp[16] objects.

On pushing data to the b250, a buffer corresponding to each parameter was filled with the appropriate data, automatically repeating, or wrapping values where arrays contained less than 16 items,[17] allowing the user to quickly assign a value to all stages and to easily create pseudo-polymetric repeating patterns within the data. This utility of the data wrapping was further enhanced by the addition of comments, where all values after `//` would be ignored before the data wrap. In addition to this, a given value could be duplicated across several stages by using `!` syntax[18]. These features, alongside copy paste functions were the main editing efficiencies implemented at this stage of the work.

4.3.3 gen~ Matrix

As an important additional element to this performance patch and in an attempt to replicate the flexibility of the 200e environment, the b250 device, alongside a series of modulation sources[19] were nested in a parent gen~ patcher, which featured a signal routing matrix, theoretically allowing for the live reconfiguration of the patch without the audio interruptions associated with MSP patching. A matrix system such as this is an attempt to convey the flexibility of traditional live coding systems, where the routing and configuration of a program can be changed on the fly, similar to live patch changes on a Buchla or Eurorack system. Conveniently the matrix also enables the amplification, attenuation and offsetting of control signals, an essential part of patching.

This two-dimensional matrix featured inputs and outputs to the external patchers and an internal connection to the inputs and outputs of each nested device. Programming of the matrix was handled from outside the patch using a second textual interface. This second interface was not a REPL system but implemented a simple syntax with which the user could link the input of a given device to any number of device outputs which could be multiplied or offset by constant values.

---

[16] Max's built in regular expression object based on PERL-flavour regex
[17] For example, the array `1 2 3 4 5` would become `1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1`
[18] For example, `5.4!3` would become `5.4 5.4 5.4`
[19] Namely several flexible LFO's, filtered noise sources and random walk devices

Implementing this matrix in gen~ would allow multiple devices to be connected to each other and importantly would allow multiple sequencers or modulation sources to be configured in feedback loops with only single sample latency, enabling the configuration of complex generative systems without compromising timing accuracy. However, the matrix gen~ patcher, in accumulating 64*64 inputs and outputs, was the most CPU intensive element in the patch by a significant margin and caused concerns in relation to the reliability of the patch. CPU overhead was worsened by the gain smoothing function, which acted to remove clicks caused when matrix connections were made, interpolating from one value to another 4096 times per sample. To combat this, gain smoothing was downsampled by a factor of 30, while the matrix continued to operate at the native sample rate, to avoid any issues with single sample impulse triggers.

By introducing a second textual control system for the matrix, the interface of the patch now consisted of left and right halves, the left for organising the routing of signals through the patch and the right for control over the devices in the patch and the data which they were operating on. On a conceptual level this separation references Buchla's own patching philosophy where sound and structure are separated by different patch cables and voltage ranges (Buchla 1971). Here two slightly different programming syntaxes are employed, allowing both halves of the patch to remain uncluttered and presenting a clearer illustration of the structure of the patch.

4.3.4 Performance

The second performance of this development cycle and first featuring the b250 device with accompanying textual interface, employed a more esoteric range of output devices than the first, and aimed to showcase the strengths and flexibility of Buchla’s function generator concepts. Alongside the b250 device encapsulated in the gen~ matrix, this patch employed a single FM synthesis voice alongside two sampling devices one of which was configured for granular sampling, and the other with two voices for more conventional chopping and triggering of various sample material. A key element of the patch were the modulation sources embedded in the gen~ matrix. These included several multi-function LFO’s, noise sources, and random walk devices, the parameters of which, including waveshape, scale, and rate, could be controlled via the REPL alongside the data arrays of the b250 and other patcher devices.

Although the matrix allowed for any configuration of the b250 device with its accompanying modulators, for this performance it was programmed in quite a conventional way, whereby the main transport clock was fed to the b250, and modulation sources assigned to the external inputs. Additionally, the main clock, with an offset applied, was patched to the start/adv input, allowing the b250 to synchronize with the main clock when a stage was programmed to advance-mode three. Trigger outputs from the b250 device were assigned via the matrix to trigger any combination of sampler or FM voice, allowing for rhythmic and melodic sequencing. Meanwhile slowly drawn-out granular textures, or short unquantized rhythmic loops provided accompaniment, via the second sampler.

The themes of granular sample manipulation juxtaposed with basic FM synthesis, paired with a sparse rhythmic accompaniment, were to be foregrounded in this performance, but had been explored previously in other performances related to this practice. Techniques and ideas established in earlier works were being moulded here into a new performance placing the 250e at its core and investigating the integration of its functionality into an existing practice and aesthetic context.

4.3.5 Analysis

This performance, relying on the new REPL interface and with several modulation devices embedded in the gen~ matrix, was a more convoluted and tumultuous experience than the first of this research. This was due in part to the timescale of development leading up to the performance event, taking place only a number of weeks into the integration of the b250 device and its interface. The gen~ matrix which the stability of this performance relied on, also was not well tested at the time of this performance, and minimal iteration or testing of the left-hand interface that controlled it had been completed.

This short performance successfully built up several unusual sequences and manipulations of sample material but failed to combine any themes or ideas to create a convincing progression or presentation of concepts, generally failing to combine sonic elements in the patch to any great effect. Alongside this, programming actions and buffer pushes were accompanied by harsh clicking sounds, likely a result of an error in the matrix buffer programmer.

Generally, the experience of interacting with the REPL and matrix programmers of this patch was an unpleasant one. The REPL in particular, was slow to parse the data of the multiple devices within the patch[20] and warranted an excessive amount of arrow key presses to navigate through data. The experience of programming the b250, while it often resulted in unexpected patterns due to the data wrapping feature, felt mostly unintentional, accidental, or exploratory. While this is not objectively bad, in the context of this research, this experience indicates the REPL interface here, did not provide any great improvements on the original interface of the 250e. Ideally, an improved interface would allow for more intentional programming, without hindering the exploration or discovery of unintentional programming results experienced here.

The gen~ matrix also created difficulty in this performance and represented a significant technical overhead during the development of this patch. While it did allow for the convenient manipulation of input and output scales and offsets, the complexity and susceptibility to error of its programming resulted in a reluctance to engage with any great manipulation of the matrix, leaving it instead in quite a simple configuration. However, not engaging with the signal flow or patching of a setup is not necessarily such a terrible loss, many performances relying on Buchla or Eurorack setups employ relatively static patches, where live patching is rare. In avoiding manipulation of the matrix, this performance, in a struggle to avoid any disastrous errors, drew into question the utility of the matrix itself. While the separation between patching and programming here allowed both interfaces to retain some simplicity, the implementation of the matrix and its interface was not a success.

[20] This problem was likely worsened by the high CPU overhead caused by the matrix patcher.

4.3.6 Refinement

Till this point in the development of this work, prototyping had primarily been concerned with removing layers of abstraction from the interface of the 250e and the early prototypes which had used its interface as a blueprint. Removing abstraction was a response to Nash and Blackwell's (2014) third *design heuristic for virtuosity*[21] and enabled the exploration of a several approaches to programming the b250's data arrays. The most recent implementation of REPL interface was successful in this respect and enabled several efficiencies such as the data wrap, commenting, and duplication functions, as well as simple copy paste operations.

While these functions provided a valuable utility and allowed faster programming of loop functions than with the physical device, the plaintext representation used here often resulted in quite unclear data structures. This issue became more problematic when using longer float values, which made it difficult to discern which values in the arrays belonged to the same stage. The confusion caused by this, in combination with the lack of feedback within the syntax to indicate which step was currently active, resulted a programming experience that often yielded unexpected and interesting results, but which rarely felt intentional precise or controlled. In conclusion, this prototyping stage illustrated the great potential of a textual approach to interfacing with the 250e but left much room for improvement with respect to the interface and programming experience of the device.

[21] *"Minimize musical (domain) abstractions and metaphors"*

4.4 Development 3: *The Grid*

The final prototype iteration developed during this research represented a substantial change in approach and narrowing of scope compared to earlier developments. The difficulties encountered thus far in the project had highlighted issues relating to the interface and usefulness of the device in live contexts. While early prototypes had enabled significant efficiencies over existing interfaces of the 250e by removing abstractions, this final phase of practice-based research proposed a device, which more appropriately represented the constraints inherent to the 250e and allowed for a new set of abstractions to be constructed over the functions of the device. At this point of the research, an initially vague research question was becoming more defined.

Focusing on the re-introduction of abstractions necessitated a narrowing of the scope established in earlier phases of development. The various REPL interfaces' ability to control all the devices present in a patch, both synthesis and structure related, offered a great convenience leveraged in early performances, but also necessitated multiple compromises, which did not benefit the user experience of programming the b250. Alongside this, the matrix patching functionality introduced in the previous development stage was excluded here to encourage experimentation with the integration of the b250 device in a variety of fixed patching configurations, tailored to specific performances.

This stage of development involved two key sections which were followed by a final series of performances. Firstly, the development of a more efficient and structured interface, which enabled newly introduced programming efficiencies alongside the presentation of visual feedback to the user. Secondly, an exploration of the functions of the 250e, proposing the expansion and inclusion of additional features, present on the 250e's sibling devices, the 249e and 251e.

4.4.1 The Grid Interface

After significant frustrations with the lack of uniformity and structure of the plaintext REPL interface, transitioning to a grid-based approach offered a useful toolkit for presenting the data utilised by the 250e in a clear way, where values maintained vertical alignment, clearly defining the structure of stages (columns) and arrays (rows). Movement towards this approach was prompted in part by the ridged grid interface of the *ORCA* Live-Coding language, which utilises a limited set of characters and a base 36 data system. The clearly communicated and firm constraints present in the orca environment, while initially challenging, result in an environment in which musical results can be reliably achieved without relying on the well-established timeline conventions used by many DAWs and sequencers. While the grid based textual system to be implemented here would not carry the same generality of scope as the ORCA language, it would take advantage of the structure and efficiency of a 2D grid system, nonetheless.

Jit.cellblock

The jit.cellblock object, part of Max's *Jitter* subcomponent, provided a useful and flexible grid system with which to build this interface. The cellblock component, in its default configuration appears and functions much like a traditional spreadsheet but can be styled and programmed to act in several ways. Most importantly for use in this project, the cellblock can be linked dynamically to a coll object, displaying the data contained in the coll in its native form and avoiding any parsing of data from dictionaries to UI objects and back. In handling this data more efficiently, the jit.cellblock solved many performance issues present within the REPL system of the previous prototype. This simplification in the data structure represented a move away from REPL systems for data handling, where the new system used a series of colls which can be linked dynamically to the cellblock to store input data. Parsing was only necessary when data was pushed to the b250 device and had to be re-formatted for insertion into buffers. Visual feedback could then be implemented through separate functions, responding to values input by the user and the position of their cursor in the grid, as well as the position of the b250's playhead.

For this interface, three separate cellblock objects were employed, one to provide context, as to which device was being edited, a second to provide a single row of eight static (read-sequencer global) settings such as the clock multiplication and the external input hold function, and the third to program the main settings and arrays of the device.

Visual feedback was achieved primarily with JavaScript (JS), embedded in max using the *script* object. JS was used to take advantage of the efficiencies of typed code, when applying styles across grids and provided three types of feedback in this prototype. Firstly, the currently active stage, as communicated by the b250 device is highlighted, enabling the user to visualise the pattern of the playhead moving though the loop structure. Secondly, while programming loops, the destination stage as dictated on the loop row is highlighted a secondary colour, allowing the user to easily determine the length of a loop in stages and ascertain whether they have selected the correct stage without relying on the motion of the playhead. Finally, the background colour of the interface could be changed, when different input sections were selected, colouring the background of the main cellblock a darker shade when the static settings cellblock was being programmed and visa-versa.

Syntax efficiencies

As well as introducing the cellblock interfaces, this revision introduced an additional set of textual efficiencies which simplified the programming of external input functionality for the time and data channels. Till this point, time and data arrays were accompanied by a time-mode and data-mode array, which dictated whether a stage would use the internally stored or externally sampled values. This array structure mirrored the configuration of the internal buffers in the b250 device; However, it resulted in a programming process which necessitated multiple movements through the data to enable or disable external sampling while programming. In this iteration the data and time mode arrays were removed, and programming of this functionality was integrated in-line with the data or time arrays, by including an `E` prefix before the number of the input to be sampled. This simplification of the programming process allowed for a far more efficient programming experience and removed three rows of data from the interface, improving the accessibility of all other device data.

Finally, additional functionality was added to streamline the programming of the loop row, which usually involved entering a value in both the loop and loops rows. This workflow was not removed, so both rows were still present on the interface, but a new function was added to the loop row where both the loops and loop row could be programmed simultaneously from the loop row by entering two values separated by a dash character [22]. Additionally, entering the dash character by itself would set the stage to automatically loop on itself, or if the dash was followed by a value, it would loop on itself using the second value in the loops column.[23] As loop programming was an important part of the b250's programming, these improvements offered significant speed increases and a greater level of convenience to the user.

4.4.2 Additional features

By including additional features, from within the canon of Buchla's approaches to function generator devices, this prototype would take advantage of the flexibility of Max and of the new textual interface developed within it to present a wider range of features, prototyped within the context of the specific sonic devices which would be used in combination with the b250 in the final performances.

Multiple Voices

The 250e unlike the MARF only provides a single play-head which can move through the data, meaning it is limited to monophonic playback. Adding a second, or potentially even further voices, potentially offered a significant increase in utility of the b250 device for use in multichannel or polyphonic configurations and could be achieved without great difficulty using MSP's *MC* objects. As a proof of concept of this feature, the b250 was configured with two voices, acting on the same dataset, but with independently scalable clocks and output sections. For usage in stereo configurations, this could result in polymetric patterns across the stereo field, and when stage times are short enough to enter the audible range, even harmonic intervals between both channels.

With the addition of multiple voices, a new control *monoVoice* was added to the static settings bar, which toggled the output of the second voice between the second independent voice, and a duplicate of the first voice. This allowed for control over the width or complexity of multi-voice sequences depending on the configuration of the device.

Alongside the multi-voice functionality, and exclusively for the final performance of this research, a third data channel was added, enabling further control of the sampling device, where the three data channels were assigned to sample length, position, and pitch respectively.

---

[22] For example, `5-3` would result in looping to step five and moving the three down to the loops row, repeating the loop three times.

[23] For example, on stage 4, `-3` in the loop row would repeat stage four three times.

Probability

The possibilities of multi voice sequencing on the b250 were further enhanced by a feature present on the 249e the *jump certainty* control (BEMI 2017, p.36) or loop probability in non-Buchla syntax was included in the b250 device as *Skip Probability* and programmed in the loops row using similar syntax to the external input configuration on the time and data channels.[24] This function allowed for further generative techniques to be explored using the b250 and meant when the device was configured in stereo, two layers of deviating but similar patterns could be created, rarely repeating the exact same pattern.

Array Functions

An additional two functions were added to the interface late in the development of this stage, `RAND` and `DUPLICATE`, used to fill arrays with strings of randomly generated or duplicate values. The duplication function could be accessed by typing `D` followed by any value and typing a hash symbol in any cell to the right of the duplicate command. This would fill every cell between and including the `D` and `#` cells with the value which succeeded `D`. Similarly, the random function took two arguments *range* and *offset* which succeeded the character `R` and would then fill every cell up to and including the `#` with a random value. Random values would be rounded to integers or left as floats depending on the format of the input range value, and are then added to the offset value before insertion into the coll. These two functions allowed for arrays to be quickly filled with values, enabling quick and efficient data changes during performance.

Simplified Buchla 251

Finally, as an accompanying element to the b250, a functionally pared down version of the Buchla 251e *Quad Sequential Voltage Source* was developed. This *b251* device was essentially cut down version of the b250, which lacked advance modes, probability functions and multiple trigger or data channels, but which had two separate datasets, allowing for two totally separate rhythmic and melodic lines to be sequenced. This device was implemented to allow experimentation with techniques employed by Ciani, where the external inputs of a MARF are used together with a 251.

4.4.3 Final performances

The final two performances of the development cycle used a similar but reduced group of sonic outputs to earlier performances. The second to last performance, a showcase of the early results of this research as part of the 4th year CSIS demo day, utilised the b250 device, alongside the simplified two voice b251 device and a simple three channel drum sequencer developed during early research into the REPL.

[24] For example, `P.3` in the loops row would mean a 30% chance the loop will be skipped, and the sequence will progress as normal.

The device at the core of the performance, the b250, was configured in stereo, controlling a simple sampling device. This configuration made use of the ability of the b250 device to cycle through stages at audible frequencies, enabling the granular manipulation of sample material when configured with a sampler and a simple envelope to remove clicks. This setup becomes more useful when using phasor or LFOs as external inputs to the b250. In this configuration, the flexibility of the b250 is foregrounded, where through a singular interface a sample can be stretched into an expanded progression of timbre, chopped rhythmically, or any sequenced into any combination of the two. Providing harmonic accompaniment to the b250's sample manipulations was the simplified b251 device, which was configured in control of two FM synth voices and also fed into the external inputs of the b250 device.

This performance aimed to showcase the utility of the b250 device in acting as a central control for a patch featuring several generative outputs and conceptually employed a more structured approach than the initial b250 performance. The final performance of this research process was an essentially an iteration on the same concept and refined further the granular process and aesthetic foregrounded in this practice.

Iteration

The final patch abandoned the b251 device and instead focused on the manipulation of the b250, paying particular attention to the interplay between its two voices. A set of LFO's with improved graphical user interface (GUI) were developed and deployed in this patch, serving alongside the new textual grid interface as the main points of interaction of the patch. To avoid complexity, additional lesser-used controls for FM voice control and event probability simply used native max GUI objects to avoid further patch complexity.

As opposed to using an isolated sequencer for rhythmic accompaniment, an approach employed for earlier performances, rhythmic elements in this patch were linked to the b250 where a kick drum was triggered based on the stage changes of the device and passed through a probability function, adding variance to the rhythm. This sample was triggered by each voice of the b250 separately, allowing for further variation of rhythm across the stereo field. This kick sound was complimented by an 808-rim sound, triggered probabilistically based on the b250 clock and a sparse Euclidean pattern.

This final patch also introduced a simple chord generator, which rounds a user assigned number of random pitches to multiples of a user selected value. The two parameters this patch provides create a simple but effective way to transition from harmonic to discordant sounds. The FM voices these pitches were assigned to, also were subject to manipulation via the 250, where the second trigger was patched to an envelope controlling the FM depth of a voice.

In summary, the final performance patch of this research, unified the set of output devices, consolidating control over these devices to the b250, configured in a setup which made better use of its outputs, and allowed for deeper control over device parameters.

4.4.4 Analysis

The penultimate performance of this work served as a valuable opportunity to present and proof the implementation of techniques and approaches, further refined in the final performance of the series. These two performances introduced a UI which drastically improved the experience of performing with the b250 by improving accessibility to key features. The ridged data structure, feedback, and programming efficiencies enabled through the grid interface combine to create a device which introduces a useful new set of abstractions over the functions of the Buchla 250e and result in a drastically improved performance experience, in comparison to earlier devices.

In particular the final performance patch more effectively utilized the functionality and flexibility of the b250, avoiding reliance on additional sequencing approaches such as the b251 model, or any other simple trigger sequencers. This was aided by a more flexible and useful set of LFO's and the addition of multiple probability-based devices which operated on the step and trigger outputs of the device. The consolidated approach used here, unified sonic outputs via the controls of the b250 and allowed for a more coherent presentation and progression between the conceptual approaches utilised throughout the performance.

The final performance, due to an improved static settings section, was the first to make use of the *external input hold* function of the b250, which when external inputs were sampling a noise signal, allowed repetitive patterns to be sampled from the noise. The new external input syntax also allowed external values to be sampled with far less effort and enabled the clear patterning of external and static values resulting in quite interesting manipulations of granular samples. This, as well as the probability syntax combined with the duplicate and random functions, allowed the b250 to be quickly reconfigured to create new patterns and sounds, without any great delay. The new interface, and its efficiencies, were key to the success of this performance. Overall, the final performance made more interesting use of the single b250 sequencer and combined this with a set of easily manipulable LFO's to foreground a hybrid approach to granular sampling, with several compelling accompaniments, aided by a reduced but more coherent sonic palette of sample material and synthesis devices.

4.5 Discussion

While a brief analysis of each development stage has already been provided, a summarising discussion is given below:

The iterative approach employed throughout this phase of the research resulted in a number of devices and performance patches, specifically moulded to a given performance context and the musical practice this research is grounded in. The resulting device, developed in the third and final phase of this process succeeded in introducing several improvements to the experience of performing with the 250e and was key to two successful performances, which foregrounded the b250 device as being a highly flexible sequencing device, enabling a number of esoteric performance techniques which integrated well into a pre-established aesthetic context.

Central to the success of this device was the grid interface, introduced in the final phase of development, which abandoned the circular paradigm of the 250e reverting to the successful horizontal approach used by the MARF. The more ridged textual approach used here provided significant usability improvements on earlier REPL interfaces, and resulted in a more efficient, stable, and powerful interface, with which one could perform deliberately and expressively. Among the most significant improvements enabled by this interface, were the new syntaxes for accessing external inputs and probability functions, creating a new layer of abstraction over these functions, as well as the loop programming efficiencies established by the dash syntax.

While the final stage of development, yielded an excellent result, this stage would not have been possible without the findings of the second, far less successful implementation of the b250 device and interface. This phase built on earlier successful REPL performances and experiments but highlighted the difficulty of integrating a complex and specific conceptual framework such as that of the 250e, within a more minimal environment. The tumultuous performance experience resultant of this stage provided feedback and findings, necessary to move on to the final stage, where the scope of the interface was narrowed, and a set of requirements for a more useful interface became clear.

In essence the iterative process although perhaps slightly inefficient, was a success and resulted in a useful performance device, as well as a deeper understanding of Buchla's function generator framework, as applied to a new aesthetic context.

5 Conclusion

In response to the question *How can the conceptual metaphors of the Buchla 250e be integrated into efficient textual interfaces, centred around musical performance practice?* This work has presented several approaches, each an iteration as part of a design process grounded in personal practice and centred on the adaptation of a performance device to the specific needs of a performer. This work has been developed with an awareness of the context surrounding Buchla's devices and placing an emphasis on the progression of Buchla's esoteric concepts from heritage devices such as the MARF to the more recent device this research focuses on.

Through a process of development, performance and iteration, the devices developed in response to this research question highlighted both successful and unsuccessful approaches to encapsulating the conceptual metaphors of the 250e in new contexts. It was found that the more minimal implementations of REPL systems struggled to efficiently represent the abstractions and feedback mechanisms necessitated by the complex framework embedded in the 250e. Considering this, a more structured grid-based approach, divergent from typical live-coding approaches, offered a far more intuitive experience to the user, and resulted in a more efficient system. The two implementations of this, served as a baseline for successful performance, and subsequent experimentation and iteration on the functionality of the 250e while integrating neatly into the performance practice this research is contextualised in.

While a successful integration was achieved over the course of this research, the less successful approaches taken should not be discounted from future use in alternative contexts or practices. A research methodology involving an informed HCI analysis, may provide more concrete categorisations and answers to this research question. However, the research methodology here has resulted in a powerful and flexible device, suited to several essential functions necessary in performance.

Alongside the device resulting from this research, several contemporary uses of Buchla's function generator concepts have been explored, which exist outside of the usual aesthetic context enabled by the 200e system. The usefulness of the 250e's functions in the context of granular sampling for example, was an unexpected discovery, and an approach not facilitated by the 200e system, which lacks any sampling capabilities. This research process has enriched the performance practice of the author and provided an enhanced toolbox of esoteric approaches to performance, informed by Buchla's historic design choices.

This research in foregrounding the flaws of modern Buchla devices such as the 250e, 249e and 251e, emphasizes the successful design and functionality of the Buchla 248 MARF. Future research focusing particularly on this device, and the features which made it such a success, may serve as a starting point for further integration of Buchla concepts in contemporary contexts, particularly that of live coding. While in the hardware realm, Buchla's devices are celebrated, compared to other manufacturers, fewer software emulations of notable Buchla modules exist, in comparison to other notable manufacturers.

In conclusion, this research process, despite its middle phases' unsuccessful approach, has resulted in a highly functional device, and a broader knowledge and appreciation of Donald Buchla's esoteric concepts, and their utility and relevance in contemporary performance practice. The four performance opportunities which punctuated the various phases of technical development provided a unique opportunity to practice and foreground the techniques afforded by Buchla's device and have resulted in the integration of these techniques alongside the b250 device into this musical practice.

6 Appendices

6.1 Appendix 1: b250 gen~ patcher & interface

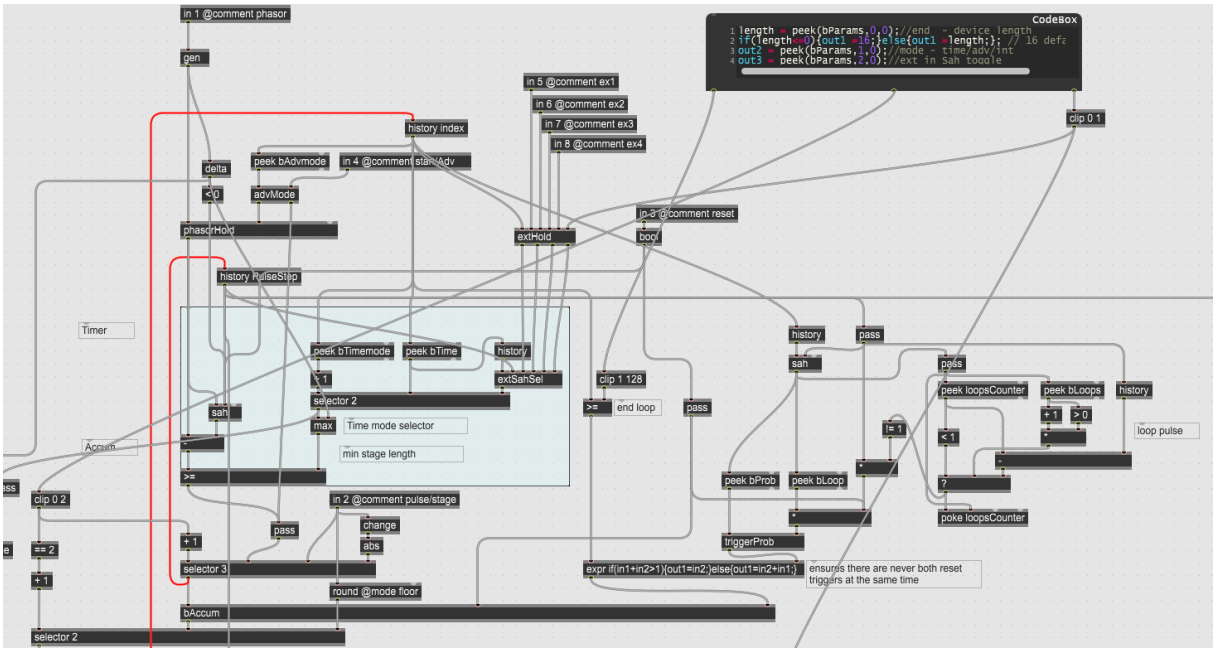

Figure 1. Main section of b250 gen~ patcher, excluding lower output sections

6.1.1 Breakdown of b250 patch

The patcher at the core of this work relies on the accumulation of an external phasor signal, (with negative delta values removed) to generate an infinite ramp, which is sampled and held at the start a stage, then compared to the ramp value to determine measure the passing of time, relative to the delta of the input phasor. When a stage has reached its assigned time, a pulse is sent to an accumulator, whose value (stage index) is sent through a history object (single sample delay) to the top of the patch to allow the next stage times and modes to be determined.

Loop functions are implemented on the right-hand side of the patch where a stages' loop value can be sent to the accumulator, setting the index to that stage. This only happens when a stages' loops counter has not been decremented to 0 or when the loops value was set to 0, allowing the loop to occur indefinitely. Alternately, the accumulator will reset when the stage has passed or reached the maximum limit (16 by default).

A less cluttered illustration of the b250's core process can be seen in the more simplistic b251 patcher:

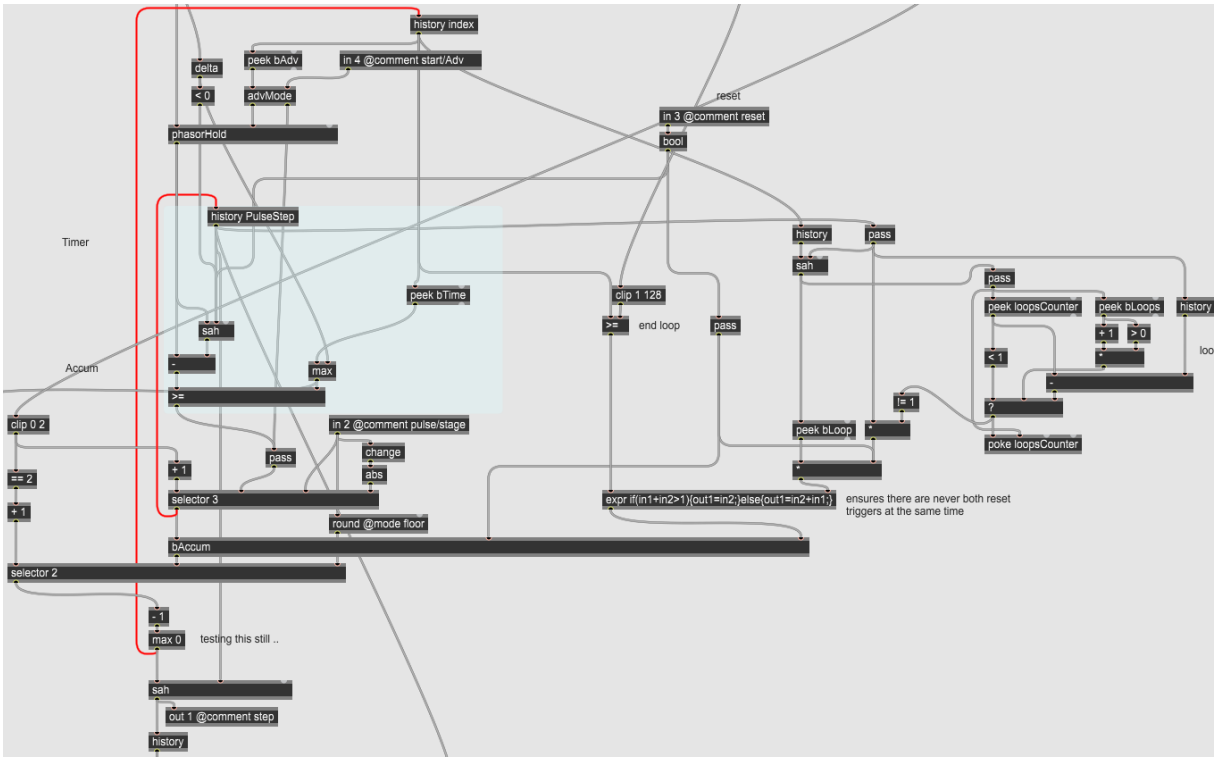

Figure 2: b251 gen~ patcher

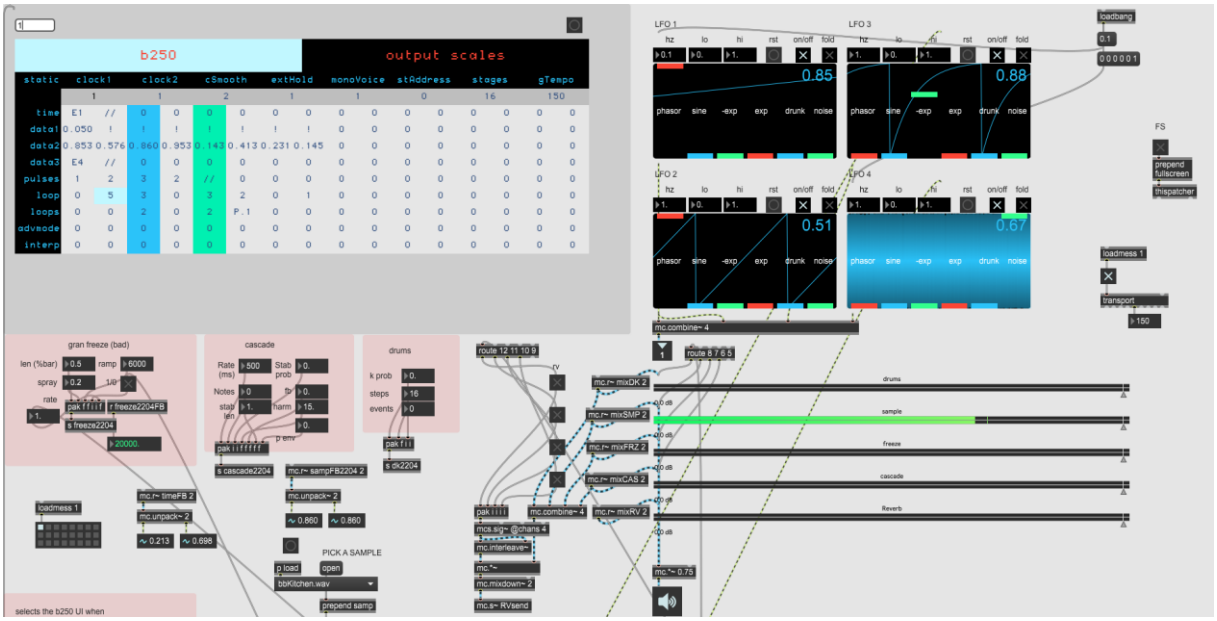

Figure 3: Final performance patch UI

6.1.2 Instructions for the grid interface and b250

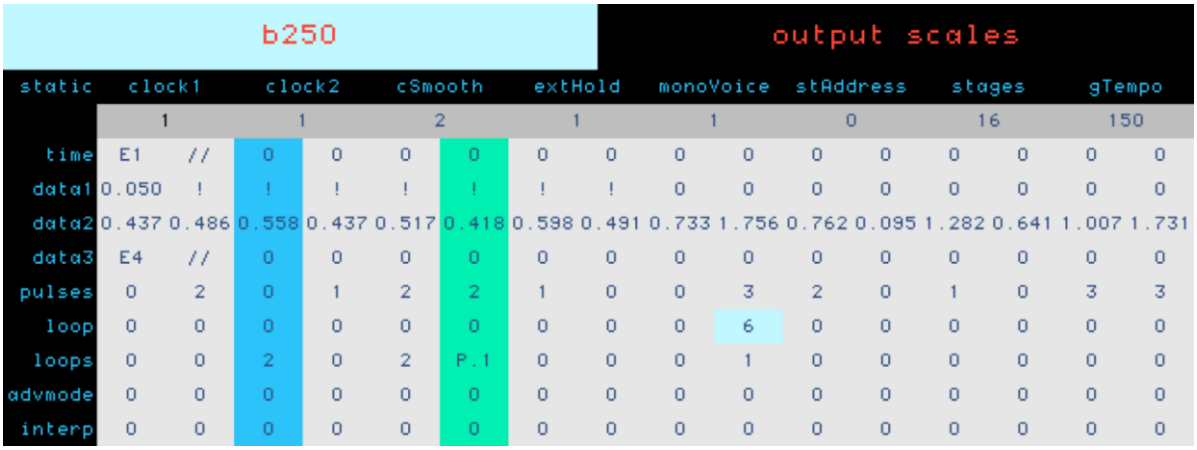

Figure 4: b250 grid interface

The grid interface at the core of the final performance patch presented by this research, can be navigated using the keyboard only, using the arrow keys to navigate through cells, where the space bar will also move horizontally to the next cell and where access to the static sections is toggled using the insert key. Additionally, the tab key is used to access the output scales tab or any other tabs the interface is configured with, such as the b251 interface in earlier versions.

When ctrl + enter is pressed, the interfaces' values will be pushed to the b250 buffers. The device needs to be assigned time values to work, otherwise it will loop through all stages at the sample rate, when some time values are populated and others left blank, the blank stages will be skipped but may cause a sample of delay. Best practice is to set the final stage to loop back to an earlier one, or change the static stages value, when using less than 16 stages, to ensure perfect timing.

On the loop row a value of 0 will have no effect, a value of 1-16 will send the playhead to that stage once the timer has run out. Use the loops value below a loop to determine how many times the loop will execute before moving onto the next step.

To access the external input function for any data or time channel, use the E followed by an integer 1-4 depending on the input to sample. Similarly, to use the probability function on the loops row, use P followed by a float value 0.-1. Which determines the probability of skipping the loop to the above step.

Advance mode values of 0 will allow the sequence to progress as normal, values of 1 will pause the timer when start/adv input is high, conversely values of 2 will pause the timer when start/adv input is low. Finally, mode 3 stops the sequence and advances to the next stage when start/adv input goes from low to high.

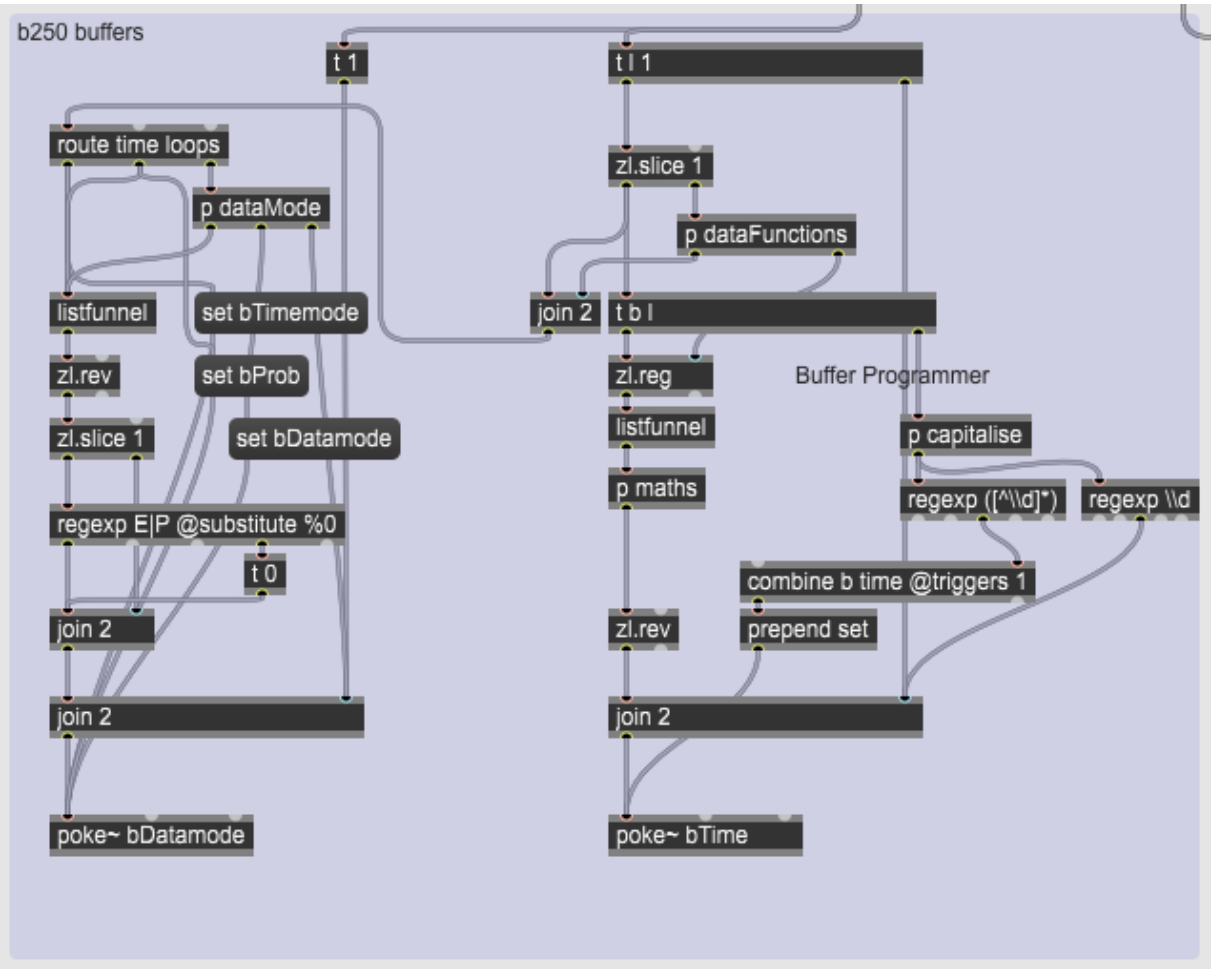

Figure 5: b250 buffer programming section

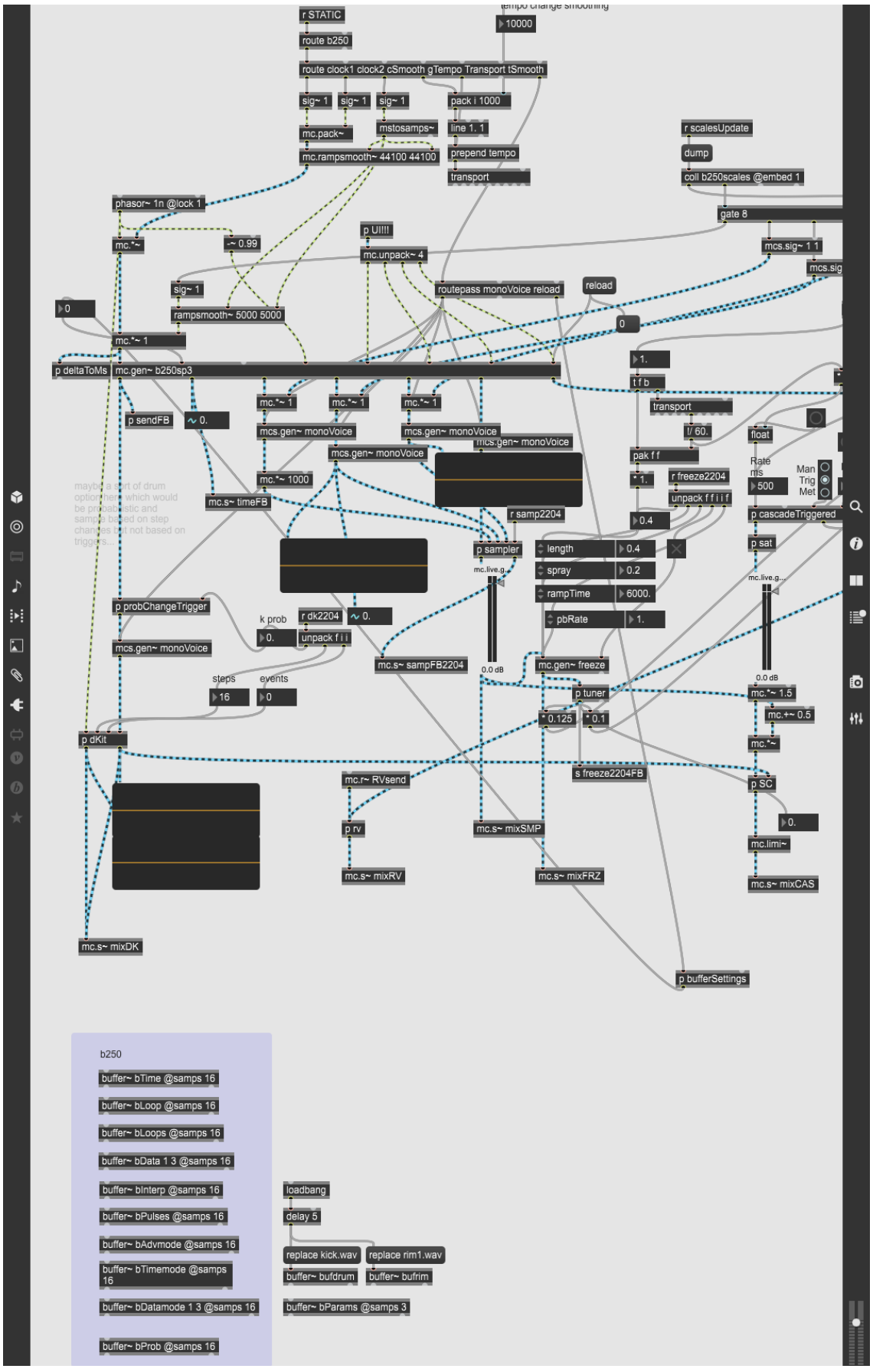

Figure 6: Macro patcher containing b250 and outputs.

7 Bibliography

Bergström, I. and Blackwell, A.F. (2016) '(PDF) The Practices of Programming', in *ResearchGate*, available: https://doi.org/10.1109/VLHCC.2016.7739684.

Blackwell, A., Cocker, E., Cox, G., McLean, A., and Magnusson, T. (2022) *Live Coding: A User's Manual*, Software studies, Cambridge, Massachusetts London: The MIT Press.

Buchla – 250e (n.d.) schneidersladen.de, available: https://schneidersladen.de/en/buchla-250e-arbitrary-function-generator [accessed 14 Dec 2024].

Buchla (2025) Systems Design – Recommended Configurations [online], available: https://buchla.com/systems-design-recommended-configurations/ [accessed 17 Apr 2025].

Buchla & Tiptop Audio [online] (2024) *Buchla & Tiptop Audio*, available: https://tiptopaudio.com/buchla/ [accessed 22 Dec 2024].

Buchla, D. (2008). '"Don Buchla." Interview by David Bernstein and Maggi Payne. In Bernstein, D. eds. The San Francisco Tape Music Center : 1960s Counterculture and the Avant-Garde,. Berkeley and Los Angeles: University of California Press. 163-177

Buchla, D. (1991) 'Transcription of Buchla Tape, Interview by Steina Valsuka and Woody Valsuka', available: https://www.vasulka.org/archive/RightsIntrvwInstitMediaPolicies/IntrvwInstitKaldron/61/BuchlaTranscription.pdf.

Buchla, D. (1971) From the Archives [online], *The Buchla Archives*, available: https://www.buchlaarchives.com/blog/from-the-archives-1 [accessed 10 Feb 2025].

Buchla, E (2006) Matrixsynth https://www.matrixsynth.com/2006/10/buchla-phases-out-259e-249e.html

Bullock, J. (2018) 'Designing Interfaces for Musical Algorithms' in McClean, A. and Dean, R.T., (Eds.) The Oxford Handbook of Algorithmic Music, Oxford handbooks, New York, NY: Oxford University Press.

Bumgardner, J. (2013) 'Variations of the Componium'. Unpublished draft, available: https://jbum.com/papers/componium_variations.pdf [accessed 18 Dec 2024]

Ciani, S. (1976) *Report to National Endowment Re: Composer Grant*, available: https://web.uvic.ca/~aschloss/course_mat/MU307/MU307%20Labs/Lab3_BUCHLA/Suzanne%20Ciani/Ciani_BuchlaCookbook.pdf [accessed 27 Apr 2025].

Ciani, S. (2016) 'Interview with Suzanne Ciani - Gearspace', available: https://gearspace.com/board/electronic-music-interviews/1374857-interview-suzanne-ciani.html [accessed 27 Apr 2025].

Ciani, S. (2024) 'Suzanne Ciani| SOS Podcast: Space, Time and Buchla', available: https://www.soundonsound.com/people/suzanne-ciani-sos-podcast [accessed 17 Apr 2025].

Clarke, A. (2016) 'Buchla 250e Dual Arbitrary Function Generator Tutorial & Demo' [video],

available: https://youtu.be/XDR-BJTnRvg?si=UXhZvYnsisdDl9PV [accessed 21 Dec 24].

Clowney, D. and Rawlins, R. (2014) ‘Pushing the Limits: Risk and Accomplishment in Musical Performance’, *Contemporary Aesthetics*, 12, available: http://hdl.handle.net/2027/spo.7523862.0012.015.

Cohen, A. M. (2012) ‘Science and Technology’. In A. F. Kinney eds. The Oxford Handbook of Shakespeare, 702–718. Oxford: Oxford University Press.

Collins, N. (2018) ‘Origins of Algorithmic Thinking in Music’ in McClean, A. and Dean, R.T., (Eds.) The Oxford Handbook of Algorithmic Music, Oxford handbooks, New York, NY: Oxford University Press.

Collins, N. and McLean, A. (2014) ‘Algorave: Live Performance of Algorithmic Electronic Dance Music’.

Crabtree, B. (2024) ‘The Mycelia of Does-Nothing Objects’ in Teboul, E.J., Kitzmann, A. and Engström, E., eds., Modular Synthesis: Patching Machines and People, London: Focal Press.

Duignan, M., Noble, J., and Biddle, R. (2005) ‘A Taxonomy of Sequencer User-Interfaces’, in *Proceedings of the International Computer Music Conference. 2005.*, Presented at the International Computer Music Conference 2005, Barcelona, available: http://hdl.handle.net/2027/spo.bbp2372.2005.158.

Fabbri, S. (2024) ‘MANUALONE’, available: https://frap.tools/docs/MANUALONE.pdf [accessed 12 Apr 2025].

Fell, M. (2013) *Works in Sound and Pattern Synthesis ~ Folio of Works.*, University of Surrey, available: https://openresearch.surrey.ac.uk/esploro/outputs/doctoral/Works-in-Sound-and-Pattern-Synthesis/99516858802346 [accessed 9 Oct 2024].

Fell, M. (2022) *Structure and Synthesis, The Anatomy of Practice*, Urbanomic.
Freeman, J. and Van Troyer, A. (2011) ‘Collaborative Textual Improvisation in a Laptop Ensemble’, *Computer Music Journal*, 35(2), 8–21, available: https://www.jstor.org/stable/41241719 [accessed 20 Dec 2024].

Freeman, J. and Van Troyer, A. (2011) ‘Collaborative Textual Improvisation in a Laptop Ensemble’, Computer Music Journal, 35(2), 8–21, available: https://www.jstor.org/stable/41241719 [accessed 20 Dec 2024].

Gann, K. (1995) ‘The Music of Conlon Nancarrow.’ Cambridge: Cambridge University Press,.

Gordon, T. (2024) 'The Buchla Music Easel' in Teboul, E., Kitzmann, A. and Engström, E., eds., Modular Synthesis: Patching Machines and People, London: Focal Press.

Hein, E. (2021) Ableton Live 11, from Journal of the American Musicological Society 74 (1): 214–225. available : https://online.ucpress.edu/jams/article/74/1/214/116877/Ableton-Live-11

Hiller, L. A., and Isaacson, L. M. (1979) Experimental Music: Composition with an Electronic Computer. 2nd ed. Westport, CT: Greenwood.

Jenkins, J. (2021) ‘The History of Sequencers’, *InSync-Sweetwater*, available: https://www.sweetwater.com/insync/the-history-of-sequencers/ [accessed 23 Dec 2024].

Kippen, J. and Bel, B. (1992) ‘Modelling music with grammars: formal language representation in the Bol Processor’.

Linvega, D.L. (2020) ‘45 • Devine Lu Linvega • Orca’, available: https://futureofcoding.org/episodes/045.html [accessed 22 Dec 2024].

Magnusson, T. and McClean, A. (2018) ‘Performing with Patterns of Time’ in McClean, A. and Dean, R.T., (Eds.) The Oxford Handbook of Algorithmic Music, Oxford handbooks, New York, NY: Oxford University Press.

Magnusson, T. (2014) ‘ixi lang’. in Collaboration and Learning through Live Coding. Dagstuhl Seminar 13382, Dagstuhl Reports 3, no. 9: 150.

Matthews, C. (2018) ‘Algorithmic Thinking and Central Javanese Gamelan’ in McClean, A. and Dean, R.T., (Eds.) The Oxford Handbook of Algorithmic Music, Oxford handbooks, New York, NY: Oxford University Press.

McLean, A. (2014) ‘Making programming languages to dance to: live coding with tidal’, in *Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling & Design*, Presented at the ICFP’14: ACM SIGPLAN International Conference on Functional Programming, Gothenburg Sweden: ACM, 63–70, available: https://doi.org/10.1145/2633638.2633647.

McLean, A. and Dean, R.T. (Eds.) (2018) *The Oxford Handbook of Algorithmic Music*, Oxford handbooks, New York, NY: Oxford University Press.

McLean, A. and Wiggins, G. (2010) *Tidal - Pattern Language for Live Coding of Music* [online], available: https://doi.org/10.5281/zenodo.849841.

Messina, M., Aliel, L., Gómez Mejía, C.M., Filho, M.C., and Soares de Melo, M.T. (2021) ‘Live Coding on Orca, the Geopolitics of the English Language and the Limits of Creative Semantic Anchoring: A Preliminary Hypothesis’, in *Proceedings of the 11th Workshop on Ubiquitous Music (UbiMus 2021)*, Proceedings of the 11th Workshop on Ubiquitous Music (UbiMus 2021), Matosinhos, Portugal: g-ubimus, 57–61, available: https://hal.science/hal-03368436 [accessed 22 Dec 2024].

Moog, B (1975) Bernie Hutchins’s interview with Bob Moog, Electronotes 7, no. 50 (1975): 9

Multiple Arbitrary Function Generator (n.d) buchla.com, available: https://web.archive.org/web/20111005142023/http://buchla.com/historical/b200/ [accessed 9th Dec 2024]

Nash, C. and Blackwell, A. (2012) *Liveness and Flow in Notation Use* [online], *Proceedings of the International Conference on New Interfaces for Musical Expression*, available: https://doi.org/10.5281/zenodo.1180547.

Nash, C. and Blackwell, A. (2014) ‘Flow of creative interaction with digital music notations’, available: https://doi.org/10.1093/oxfordhb/9780199797226.013.023.

Pinch, T. and Trocco, F. (2002) *Analog Days: The Invention and Impact of the Moog Synthesizer* [online], Cambridge, UNITED STATES: Harvard University Press, available: http://ebookcentral.proquest.com/lib/univlime-ebooks/detail.action?docID=3300076 [accessed 9 Dec 2024].

Resident Advisor (2024) 'Hierarchical Control of Sequences in the MARF | The Art of Production: Suzanne Ciani', The Art of Production [video,] available: https://youtu.be/EvA4xenIrZo?si=Z9pVPPpssgnXoiBL [accessed 22 Dec 2024]

Reid, G. (2005) Buchla 200e: Part 1, Sound on Sound, available: https://www.soundonsound.com/reviews/buchla-200e-part-1 [accessed 14 Dec 2024].

Reid, G. (2006) Buchla 200e: Part 2, Sound on Sound, available: https://www.soundonsound.com/reviews/buchla-200e-part-2 [accessed 14 Dec 2024].

Riley, T. (2009) 'Composing for the Machine'. European Romantic Review 20, no. 3 : 367379.

Roberts, C. and Wakefield, G. (2018) 'Tensions and Techniques in Live Coding Performance' in McClean, A. and Dean, R.T., (Eds.) The Oxford Handbook of Algorithmic Music, Oxford handbooks, New York, NY: Oxford University Press.

Rohrhuber, J., de Campo, A., and Wieser, R. (2005) 'ALGORITHMS TODAY: NOTES ON LANGUAGE DESIGN FOR JUST IN TIME PROGRAMMING', *Conference: Proceedings of International Computer Music Conference Barcelona*.

Shepherd, S. (2019) 'The Art Of Production: Floating Points', available: https://ra.co/features/3548 [accessed 27 Apr 2025].

Spiegel, L. (1981) 'Manipulations of Musical Patterns', in ResearchGate, available: https://www.researchgate.net/publication/266316606_Manipulations_of_Musical_Patterns [accessed 2 Dec 2024].

Toplap (2024) ManifestoDraft [online], *Toplap*, available: https://toplap.org/wiki/ManifestoDraft [accessed 29 Oct 2024].

Toussaint, G. (2013) *The Geometry ofMusical Rhythm*.

Vincent, R. (2020) RYK Modular M185 Sequencer [online], *Sound On Sound*, available: https://www.soundonsound.com/reviews/ryk-modular-m185-sequencer [accessed 18 Apr 2025].