Chapter 8
Turing and the History of Computer Music

B. Jack Copeland and Jason Long

Abstract The story of Turing's pioneering work in creating the first computer-generated musical notes in Manchester in 1948–1949 is told, as well as the story of Christopher Strachey's work (later Oxford's first professor of computing), who extended Turing's note-playing routines to create computer-generated melodies. Recordings were made in Turing's Computing Machine Laboratory by the British Broadcasting Corporation (BBC) in 1951: by analyzing Turing's programming manual for the Manchester machine—the first ever written for a stored-program computer—and utilizing retrospective computer analysis of the recordings, a kind of "digital archaeology" is employed in order to reconstruct the Turing-style routines that were used to play the music recorded by the BBC. These techniques have also enabled us to restore the recordings. We establish Turing's leading role in the history of computer music.

8.1 Introduction

One of Turing's contributions to the Digital Age that has largely been overlooked is his pioneering work on transforming the computer into a musical instrument. It's an urban myth of the music world that the first computer-generated musical notes were heard in 1957, at Bell Labs in America.[1] In fact, about nine years earlier, computer-generated notes were heard in Turing's Computing Machine Laboratory at Manchester University.

This chapter analyzes Turing's groundbreaking work at Manchester, and also describes how Christopher Strachey, later Oxford University's first professor of computing, used and extended Turing's note-playing routines to create

The original version of this chapter was revised: This chapter was previously published as non-open access. It has now been changed to open access under a CC BY 4.0 license. The correction to this chapter is available at https://doi.org/10.1007/978-3-319-53280-6_16

[1] See, for example, Chadabe (2001).

B.J. Copeland (✉)
University of Canterbury, Christchurch, New Zealand
e-mail: jack.copeland@canterbury.ac.nz

J. Long
Victoria University of Wellington, Wellington, New Zealand
e-mail: jason.long@ecs.victoria.ac.nz

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
https://doi.org/10.1007/978-3-319-53280-6_8

computer-generated melodies. Computer-generated music was recorded in Turing's Manchester Laboratory by the British Broadcasting Corporation (BBC) in 1951.[2] We outline the techniques of digital archaeology, and the sheer detective work, which led to our reconstruction of the routines that were used to play the music in this recording, and led also to our restoration of the recording. To set the scene, we begin with an overview of the historic Manchester computer and of the characters involved in our story. Then we turn to Turing's own programming manual for the Manchester machine—the first ever written for a stored-program computer. We clarify his notation for the machine's instructions, and trace in detail how his lines of code were actually transformed into computer-generated notes. We describe the computer-assisted search that helped us to reverse engineer the Turing-type melody-playing algorithms used to produce the music in the recording—algorithms that ushered in the field of computer music. Our analyses establish Turing's pioneering role in the history of computer music.

8.2 The Hardware

The world's first electronic all-purpose stored-program computer—the first electronic universal[3] Turing machine—ran its first program in June 1948 (Copeland 2011a, b). Called simply "Baby", this historic machine was tiny, rough and ready, and almost entirely lacking in the facilities needed for serious computing. Programs were entered bit by painstaking bit, using a panel of hand-operated switches to plant each bit in memory. Output was in the form of bright dots on a tiny glass screen. Baby was designed by two brilliant engineers, Freddie Williams and Tom Kilburn, to test their new groundbreaking high-speed electronic memory, the Williams tube. Around an array of three Williams tubes they wired together the simplest stored-program computer they could think of (Fig. 8.1).

In later life Kilburn was at pains to deny that Turing had contributed anything to the Baby (Copeland 2011b). This was highly misleading. For one thing, Turing contributed the fundamental concept, the very idea of a universal machine that stores programs of symbolically-coded instructions in its memory. Summarizing his pre-war work, Turing wrote in 1947:

> Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape. ... [D]igital computing machines ... are in fact practical versions

[2] Part of this recording can be heard at http://www.abc.net.au/classic/content/2014/06/23/4028742.htm. This edition of *Midday with Margaret Throsby* (ABC Radio National, 23 June 2014) is a musical tour through Turing's life and work, on the 102nd anniversary of his birth.

[3] Of course, it was a universal machine with a *finite* memory, a concept introduced by Turing in his "Intelligent Machinery" (1948), where he spoke (p. 422) of "a universal machine with a given storage capacity".

Fig. 8.1 Baby, the first electronic stored-program computer. Baby came to life in June 1948. The proud parents: Tom Kilburn is on the left, Freddie Williams on the right (Courtesy of the University of Manchester School of Computer Science)

> of the universal machine. There is a certain central pool of electronic equipment, and a large memory, [and] the appropriate instructions for the computing process involved are stored in the memory (Turing 1947).

Kilburn had stepped into Turing's world at the end of 1946, when he entered a dingy London lecture room and sat down to listen to Turing explaining how to build a computer.[4] Williams had recently succeeded in storing a single binary digit on the face of a cathode ray tube, proving that his computer memory idea worked in principle; and so, as Williams said, "the point now had been reached where we'd got to find out about computers".[5] They heard that Turing was giving a series of lectures on computer design in London, and it was decided that Kilburn would attend.[6] The lectures ran from December 1946 through to February 1947, and were held in a conference room at the Adelphi Hotel in the Strand.[7] Kilburn was a good pupil, quickly progressing during the lectures from not knowing (as Williams put it) the

[4] The lecture notes are published as "The Turing-Wilkinson Lecture Series (1946-7)" (Turing (1946–7). The series of nine lectures (about half of which were given by Turing's assistant, Jim Wilkinson, most likely from notes prepared by Turing) covered Versions V, VI, and VII of Turing's design for the ACE; see also Copeland (1999).

[5] Williams in interview with Christopher Evans in 1976. ("The Pioneers of Computing: An Oral History of Computing", London: Science Museum. © Board of Trustees of the Science Museum. Transcription by Copeland (1997)).

[6] Bowker and Giordano (1993), p. 19.

[7] See Copeland (2005), pp. 459–464. Womersley's handwritten notes concerning the arrangements for the lectures (Woodger Papers, catalogue reference M15) are in *The Turing Archive for the History of Computing* <http://www.AlanTuring.net/womersley_notes_22nov46>).

"first thing about computers"[8] to the point where he could start designing one himself. Kilburn's initial design (later superseded) for what would eventually be the Manchester computer followed Turing's principles closely, and Kilburn's written reports made extensive use of the terminology Turing had taught him in the Adelphi lectures.[9] When asked where he had got his basic knowledge of the computer from, Kilburn usually said, rather irritably, that he couldn't remember.[10] In a 1993 interview, he commented vaguely "Between early 1945 and early 1947, in that period, somehow or other I knew what a digital computer was", adding "Where I got this knowledge from I've no idea".[11] There is in fact no mystery about where Kilburn got his basic knowledge of the computer from—Turing taught him.

A few weeks after Baby ran its first program, Turing accepted the offer of a job at Manchester University. At last he could get his hands on a universal Turing machine in hardware. Turing improved on the bare-bones facilities, designing an input-output system based on wartime equipment used at Bletchley Park. Williams and Kilburn themselves knew nothing of Bletchley Park and its nine gigantic Colossus computers.[12] The ultra-secret Colossus was the world's first large-scale electronic computer, although it was not all-purpose and did not incorporate Turing's stored-program concept.[13] Turing based his input-output system for the Manchester computer on the same teleprinter tape that ran through Colossus.[14] His tape reader converted the patterns of holes punched across the tape into electrical pulses, and fed these pulses to the computer. The reader incorporated a row of light-sensitive cells which read the holes in the moving tape—exactly the same technology Colossus had used.

As the months passed, a large-scale computer took shape in the Manchester Computing Machine Laboratory (Fig. 8.2). Turing called it the Manchester Electronic Computer Mark I.[15] A broad division of labor developed that saw Kilburn and Williams working on the hardware, and Turing on the software. Williams concentrated his efforts on developing a new form of supplementary memory, a rotating magnetic drum, while Kilburn took the leading role in developing the computer proper. Turing designed the Mark I's programming system, and went on to write the world's first programming manual.[16] The Mark I was operational in April 1949,

[8] Williams in interview with Evans, 1976 (transcription by Copeland (1997)).

[9] For further information see Copeland (2011a, b), and (2012) Chap. 9.

[10] Letter from Brian Napper to Copeland, 16 June 2002.

[11] Bowker and Giordano (1993), p. 19. Copeland is grateful to Napper for drawing this passage to his attention, in correspondence during 2002.

[12] Kilburn: "I didn't know anything about the work at Bletchley". Kilburn in interview with Christopher Evans in 1976; also Williams in interview with Christopher Evans in 1976. ("The Pioneers of Computing: An Oral History of Computing", London: Science Museum. © Board of Trustees of the Science Museum. Transcription by Copeland.)

[13] Copeland et al. (2006), esp. Chap. 9.

[14] For additional detail see Copeland (2011b), pp. 31–32.

[15] Turing, A. M. (c. 1950), p. 85.

[16] Turing (c. 1950).

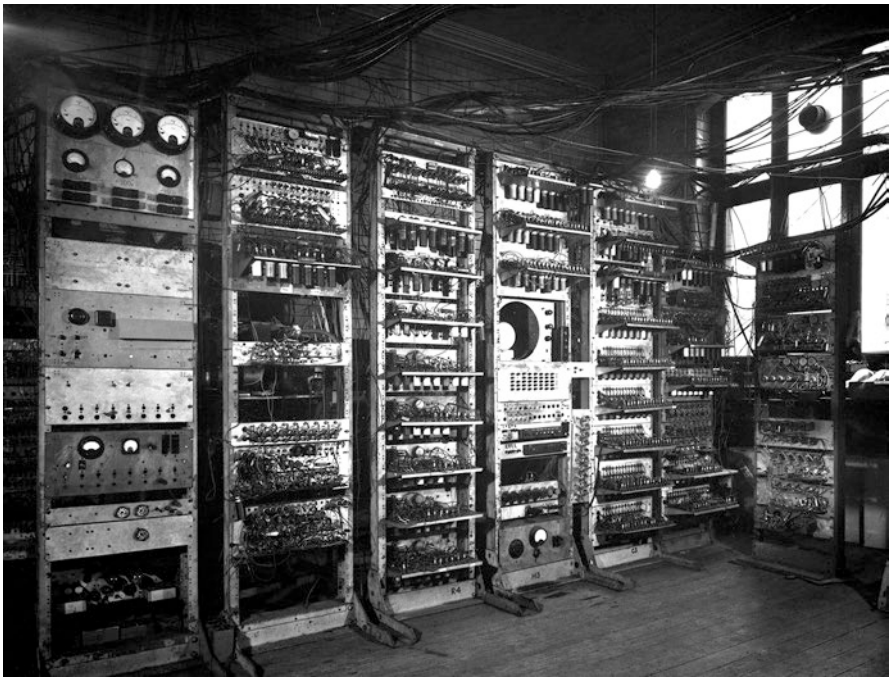

Fig. 8.2 Baby grows into the Mark I (Permission: University of Manchester School of Computer Science)

although additional development continued as the year progressed.[17] Ferranti, a Manchester engineering firm, contracted to build a marketable version of the computer, and the basic designs for the new machine were handed over to Ferranti in July 1949.[18] The first Ferranti computer was installed in Turing's Computing Machine Laboratory in February 1951, a few weeks before the earliest American-built marketable computer became available, the UNIVAC I (Fig. 8.3).[19]

Turing referred to the new machine as the Manchester Electronic Computer Mark II, while others called it the Ferranti Mark I. Turing's nomenclature will be followed here. His programming manual was written in anticipation of the Mark II's arrival, and is titled *Programmers' Handbook for Manchester Electronic Computer Mark II* (c. 1950), but it was the outcome of his programming design work undertaken on the Mark I.[20]

[17] Copeland and Sommaruga (2015), pp. 99–100; Williams and Kilburn (1952).

[18] Williams and Kilburn (1952), p. 59.

[19] The delivery date of the first Ferranti computer is given in a letter from Turing to Woodger, undated, received 12 February 1951 (in the Woodger Papers). A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/turing_woodger_feb51. For details of the UNIVAC see Stern (1979), p. 17; and Stern (1981), p. 149.

[20] See Turing's preface to (c. 1950).

Fig. 8.3 Turing at the console of the Mark II computer (Permission: University of Manchester School of Computer Science)

Turing's *Handbook* contains what is, so far as is known, the earliest written tutorial on how to program an electronic computer to play musical notes.

8.3 Programming Notes

The Manchester computer had a loudspeaker—the "hooter", it was called—that served as an alarm to call the operator when the machine needed attention.[21] With some simple programming, the loudspeaker could be made to emit musical notes.

The computer's "hoot instruction" worked like this. There was an electronic clock in the computer synchronizing all the operations. This clock beat steadily, like a silent metronome, at a rate of thousands of noiseless ticks per second. Executing the hoot instruction a single time caused a sound to be emitted at the loudspeaker, but the sound lasted no longer than a tick, a tiny fraction of a second. Turing described this sound as "something between a tap, a click, and a thump".[22] Executing the hoot instruction over and over again resulted in this brief sound being produced repeatedly, on every fourth tick: tick tick tick *click*, tick tick tick *click*.[23]

[21] There is a circuit diagram of the hooter in Dodd (c. 1953), Diagram 10.

[22] Turing (c. 1950), p. 24.

[23] Dodd (c. 1953), p. 59.

If the clicks are repeated often enough, the human ear no longer hears discrete clicks but a steady note. Turing realized that if the hoot instruction is repeated not simply over and over again, but in different patterns, then the ear hears different musical notes. For example, if the pattern tick tick tick *click*, tick tick tick tick, tick tick tick *click*, tick tick tick tick is repeated, the note of C_5 is heard. (The subscripted number indicates the octave in which the note occurs. Turing described C_5 as middle C, as musicians sometimes do, especially if playing an instrument with a very high register; however, it is more usual to call C_4, which is an octave below C_5, middle C.[24]) Repeating the different pattern tick tick tick *click*, tick tick tick *click*, tick tick tick tick, tick tick tick *click*, tick tick tick *click*, tick tick tick tick produces the note of F_4—and so on. It was a wonderful discovery.

Turing himself seems not to have been particularly interested in programming the machine to play conventional pieces of music. The different musical notes were used as indicators of the computer's internal state—one note for "job finished", others for "error when transferring data from the magnetic drum", "digits overflowing in memory", and so on.[25] Running one of Turing's programs must have been a noisy business, with different musical notes and rhythms of clicks enabling the user to "listen in" (as Turing put it) to what the program was doing. He left it to someone else, though, to program the first complete piece of music.

8.4 God Save the King

One day Christopher Strachey turned up at the Computing Machine Laboratory (Fig. 8.4). Before the war, he had known Turing at King's College, Cambridge. Strachey was soon to emerge as one of Britain's most talented programmers, and he would eventually direct Oxford University's Programming Research Group. When he first strode into the Manchester Computing Machine Laboratory he was a mathematics and physics master at Harrow, one of Britain's foremost schools. Strachey felt drawn to digital computers as soon as he heard about them, in about January 1951, and taking the bull by the horns he wrote to Turing in April.[26] Turing sent a copy of his *Handbook* (c. 1950) and Strachey studied it assiduously.[27] This was "famed in those days for its incomprehensibility", Strachey said.[28] An ardent

[24] By the time of the third edition of Turing's *Programmers' Handbook* (prepared by Tony Brooker in 1953), Turing's "about middle C" had been replaced by "an octave above middle C".

[25] Prinz (1952), section 20. Copeland is grateful to Dani Prinz for supplying a cover sheet that shows the date of this document.

[26] Letter from Strachey to Max Newman, 5 October 1951 (in the Christopher Strachey Papers, Bodleian Library, Oxford, folder A39); letter from Strachey to Michael Woodger, 13 May 1951 (in the Woodger Papers).

[27] Letter from Strachey to Newman, 5 October 1951. Strachey's copy of Turing's *Programmers' Handbook* still exists, signed on the cover "With the compliments of A. M. Turing" (in the Christopher Strachey Papers, folder C40).

[28] Foy (1974), p. 10.

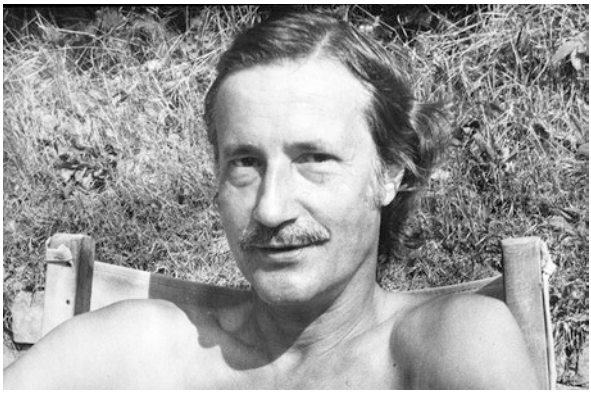

Fig. 8.4 Christopher Strachey sunbathing in the garden of his cottage "The Mud House"; the photo was taken in 1973, two years before his untimely death (Courtesy of the Bodleian Library and Camphill Village Trust)

pianist, he appreciated the potential of Turing's terse directions on how to program musical notes. Strachey first visited the Computing Machine Laboratory in July 1951; Turing decided to drop him in at the deep end and suggested he try writing a program to make the computer check itself.[29] When Strachey left the Laboratory, Turing turned to his friend Robin Gandy and said impishly, "That will keep *him* busy!"[30]

It did keep him busy, during the school summer holidays of 1951.[31] Strachey was a precocious programmer and when he "trotted back to Manchester", he recollected, he had with him twenty or so pages covered in lines of programming code—at that time by far the longest program to be attempted.[32] "Turing came in and gave me a typical high-speed, high-pitched description of how to use the machine", Strachey recounted.[33] Then he was left alone at the computer's console until the following morning.

"I sat in front of this enormous machine", Strachey said, "with four or five rows of twenty switches and things, in a room that felt like the control room of a battleship."[34] It was the first of a lifetime of all-night programming sessions. He worked on debugging his monster program, which he called "Checksheet".[35] The name was a variation on a term Turing had used, in his *Programmers' Handbook*, for a hand method of checking programs. Turing called his method "Check Sheets". The

[29] Letter from Strachey to Newman, 5 October 1951; Robin Gandy in interview with Copeland, October 1995.

[30] Gandy in interview with Copeland, October 1995.

[31] Strachey in Foy (1974), p. 11.

[32] Strachey in Foy (1974), p. 11.

[33] Strachey in Foy (1974), p. 11.

[34] Strachey in Foy (1974), p. 11.

[35] Strachey gave the name of the program in his letter to Newman, 5 October 1951. The Checksheet program itself is in the Christopher Strachey Papers (folder C52).

method was "done on paper with quarter inch squares on which vertical lines are ruled in ink", Turing explained in the *Handbook*.[36]

As well as spending the night struggling to debug Checksheet, Strachey prepared a surprise. He managed to debug and get running another program that he'd brought with him. To the astonishment of onlookers, the computer raucously hooted out the British National Anthem.[37] A budding programmer could hardly have thought of a better way to get attention. A few weeks later, Max Newman, Professor of Mathematics at Manchester and founder of the Computing Machine Laboratory, heard the computer grinding out "God Save the King". Newman quickly wrote a letter to Strachey suggesting he might like a programming job in the Lab.[38]

Manchester's musical computer also caught the attention of the popular press, with headlines like "Electronic brain can sing now".[39] The accompanying article explained that "the world's most powerful brain" was "given a coded version of the score", from which it "constructed the necessary waveform". The BBC sent a recording team together with a radio presenter from *Children's Hour*, known as Auntie, to capture a performance by the computer.[40] As well as "God Save the King", the BBC recorded a version of Glenn Miller's "In the Mood", a reedy and wooden performance of the famous hit. There was also an endearing, if rather brash, rendition of the nursery rhyme "Baa Baa Black Sheep". The Mark II, still full of glitches, managed to crash in the middle of its Glenn Miller party piece. "The machine's obviously not in the mood", Auntie gushed.

The unedited BBC recording of the session conveys a sense of people interacting with something entirely new. "The machine *resented* that", Auntie observed at one point. The idea of a thinking machine, an electronic brain, was in the air at Manchester. Turing merrily fanned the flames. He provocatively told a reporter from *The Times* that he saw no reason why the computer should not "enter any one of the fields normally covered by the human intellect, and eventually compete on equal terms".[41]

Max Newman lectured on the new computer music in 1952, to 250 professional musicians who were attending the annual conference of the Incorporated Society of

---

[36] Turing (c. 1950), p. 12.

[37] Frank Cooper in interview with Chris Burton in 1994; an audio recording of part of the interview is at http://curation.cs.manchester.ac.uk/digital60/www.digital60.org/media/interview_frank_cooper/index-2.html. In the secondary literature it is sometimes said that "God Save the King" was played at the end of Strachey's draughts (checkers) program, but this is not correct (see e.g. Link (2012/2013), p. 23). For further information about Strachey's draughts program, see Copeland (2012), Chap. 9.

[38] Letter from Newman to Strachey, 2 October 1951 (in the Christopher Strachey Papers, folder A39).

[39] See "Electronic Brain Can Sing Now". *The Courier and Advertiser*, 28 February 1952. We are grateful to Diane Proudfoot for finding this article and supplying us with it.

[40] Cooper interviewed by Burton.

[41] Turing quoted in "The Mechanical Brain", *The Times*, 11 June 1949.

Musicians. His lecture was reported in the national press.[42] After explaining that, to make the Manchester computer play melodies, "All you have to do is to send an instruction to the hooter with the frequency of the note you want it to play", Newman described the discovery that the computer could be programmed to *compose* tunes for itself. So far these were, he admitted, "very bad tunes". (Quite possibly the program used Turing's random number generator, a standard hardware component of the Ferranti computers.) According to the *Manchester Guardian*:

> The next step, said Professor Newman, would be to make a machine which could compose *good* tunes, but so far no method of bridging the gap had been devised.[43]

The article continued:

> Professor Newman ended with this note of comfort for the assembled musicians: "All this appears much more alarming and dangerous than it really is. When you see how it is done and how far it is from genuine composition, composers will realise they need not start taking steps to protect themselves against competition from machines."

8.5 Turing's Music Tutorial

Turing's brief tutorial in his *Handbook* was typically compressed and demanding; yet, equally typically, his terse account told readers everything it was necessary to know in order to start writing note-playing programs. Turing called the hoot instruction /V, pronounced "slash vee". The complete tutorial occupied little more than half a page:

> The hooter. When an instruction with function symbol /V is obeyed an impulse is applied to the diaphragm of a loudspeaker. By doing this repeatedly and rhythmically a steady note, rich in harmonics, can be produced. This is used to enable the operator to be called to attend to the machine in some way. The simplest case is where the whole of a job is completed and it is required to clear the electronic stores and start something different. All that is then required is to repeat a cycle of instructions including a hoot, e.g.
>
> ```
> FS  NS/V
> CS  FS/P
> ```
>
> In this case every second instruction will put a pulse into the speaker. These pulses will occur at intervals of 8 beats i.e. 1.92 ms giving a frequency of 521 cycles (about middle C). Or one could use the loop of three instructions

[42] "Very Bad Tunes", *Manchester Guardian*, 4 January 1952. We are grateful to Diane Proudfoot for finding this article and supplying us with it.

[43] "Very Bad Tunes", emphasis added.

```
O@        /V
G@      P@/V
M@      O@/P [see our footnote][44]
```

```
which gives a slightly louder hoot a fifth lower in frequency.
Single pulses applied to the loudspeaker are distinctly audi-
ble as something between a tap, a click, and a thump. This
fact can be turned to good account. By putting hoot instruc-
tions into programmes at suitable points one is enabled to
"listen in" to the progress of the routine. Some indication
of what is going on is given by the rhythm of the clicks that
are heard.[45]
```

In these two loops, one consisting of two instructions and one consisting of three, Turing has used international teleprinter code to abbreviate the instructions. At the level of "machine code", the instructions consist simply of strings of binary digits (bits). Teleprinter code associates keyboard characters with strings of 5 bits; for example, *A* is 11000 and *B* is 10011. Teleprinter code was well known to engineers in that era, and was very familiar to Turing from his wartime work at Bletchley Park on the "Tunny" teleprinter code, used by Hitler and his generals. To Turing, teleprinter code must have seemed a natural choice for abbreviating the Manchester computer's bitcode. This system's main defect, that the abbreviations give no intuitive sense at all of what is being abbreviated, is one reason why his *Handbook* was such heavy going.

Let us unpack what Turing wrote down, in order to clarify its importance in the history of computer music. First, we explain Turing's notationally formidable subroutines, simplifying his notation and spelling out the connection between his sequences of coded instructions and perceived musical sounds. Then we report the results of our computer-assisted analyses of the BBC's 1951 recording of the Manchester computer's music. From these results, we extract our account of how others at Manchester extended Turing's note-playing subroutines, so enabling the Mark II to play its first melodies. We also provide a series of tables that allow the

[44] For ease of exposition, we have replaced Turing's G@/P by O@/P, thereby oversimplifying the behaviour of the /P instruction, but making the loop ostensibly easier to follow. Appearances to the contrary notwithstanding, Turing's G@/P does take execution back to the start of the loop, while our oversimplified version does not do so. For a full explanation of /P, see Prinz (1952), p. 14. Strachey, who marked corrections by hand on his copy of Turing's *Handbook*, altered this loop to:

```
O@   /V
B@   Q@/V
G@   B@/P
```

See also Strachey's typed sheets of errata to Turing's *Handbook* dated 9 July 1951; in the Christopher Strachey Papers, folder C45.

[45] Turing (c. 1950), p. 24. There is a magisterial introduction to programming (what Turing called) the Mark II in Campbell-Kelly (1980).

reader to see not only how these melodies were produced but also, more generally, how the computer itself can be used in historical detective work *about* computers.

In teleprinter code, / is 00000 and V is 01111; thus /V is the teleprinter code abbreviation of the Mark II's 10-digit hoot instruction, 0000001111. Turing's /P is also an instruction; instructions always began with /, or T (00001). The other symbols in Turing's two sample subroutines, NS, P@, FS, CS, O@, G@ and M@, are memory addresses: each pair of symbols abbreviates a 10-digit address in the computer's Williams tube memory.

Instruction /P (unconditional transfer of control) tells the machine to obey next the instruction stored at a location specified via the address immediately to the left of the /. In effect the second line of the first loop sends the machine back to the first line (but see our footnote); and the final line of the second loop again sends the machine back to the first line.[46] The computer will continue to loop until an instruction from elsewhere in the program terminates the loop after *n* repetitions.[47] The programmer selects the number *n*, so determining how long the note is held, as required by the rhythms of the melodies.

Our analysis of the BBC's recording of the Mark II playing "God Save the King", "Baa Baa Black Sheep", and "In the Mood", showed that the durations of the played notes varied between 80 milliseconds and 1100 milliseconds. The analysis also revealed that very short pauses were programmed between each consecutive note, presumably by means of "silent" loops—short loops containing no hoot instruction. The duration of these inter-note pauses is between 40 and 50 milliseconds (exact measurement was difficult because the pauses are blurred over by reverberation from the room housing the computer). The inter-note pauses help to define the beginning of each note, and are essential if a sequence of several notes of the same pitch is played. Without a gap between the individual notes, a single long note would be heard.

The occurrence of NS to the left of /V in Turing's first subroutine can be ignored for the present purposes, and so can the P@ to the left of /V in the second subroutine. These terms create special effects to do with the computer's visual display, and have no role in the production of musical notes. The effect produced by including the address NS in line 1 of the first subroutine is that the information stored at NS momentarily brightens on the monitor display as the machine hoots.[48] This provides a visual prompt to assist the operator. Similarly, the effect of P@ in the second instruction of the three-line subroutine is to cause the information stored at P@ to brighten on the display as the hooter sounds. The two note-playing subroutines can be written more cleanly without these special effects:

[46] See note 44.

[47] Turing explains loop control (by means of a B-tube) in his *Handbook*, pp. 66–67; the B-tube was effectively a register containing *n* and this number was counted down by repeatedly subtracting 1.

[48] Turing (c. 1950), p. 22.

```
FS    /V
CS    FS/P

O@    /V
G@    /V
M@  O@/P
```

Taking clarity a step further, we might replace the teleprinter-coded addresses with simple line-numbers:

```
1   /V
2  1/P

1   /V
2   /V
3  1/P
```

As Sect. 8.3 mentioned, the /V instruction takes four ticks to complete—four *beats* in the Manchester jargon—with the actual hoot occurring on the fourth beat.[49] /P also takes four beats to complete. As Williams and Kilburn put it, the basic rhythm of the Manchester computer was "four beats to the bar".[50] Thus, running through Turing's two-line subroutine once produces: tick tick tick *click*, tick tick tick tick; and looping repeatedly through the subroutine produces the first of the two sequences discussed in Sect. 8.3. Similarly, running through the three-line subroutine once gives: tick tick tick *click*, tick tick tick *click*, tick tick tick tick; and looping repeatedly gives the second sequence in Sect. 8.3.

The precise duration of a single beat was 0.24 milliseconds (ms). The first subroutine produces one click every 8 beats, which is to say every 1.92 ms. Thus, the frequency with which clicks are produced, as the machine loops repeatedly through the subroutine, is (1 ÷ 1.92) clicks per ms—0.52183 clicks per ms, or 521.83 clicks per second. In standard units, the frequency of the clicks is said to be 521.83 Hertz (Hz). This is close to C_5, whose assigned frequency in the 'Equal Tempered' scale is 523.25 Hz. The Equal Tempered scale is the standard scale for keyboard instruments, with adjacent keys playing notes heard as equidistant from one another.[51]

Table 8.1 shows the Equal Tempered frequencies of all the notes occurring in the fragments of the scores of "God Save the King", "In the Mood" and "Baa Baa Black Sheep" that were performed in the BBC recording. Later tables show the actual frequencies that the computer produced.

[49] The Mark II was synchronised by an oscillator with a frequency of 100 kHz. Turing called a single cycle of the oscillator the "digit period". The digit period was 10-microseconds and the duration of a beat was 24 digit periods.

[50] Williams and Kilburn (1952), p. 57.

[51] We follow the A = 440 Hz tuning standard.

Table 8.1 The Equal Tempered frequencies of the notes from those parts of the scores of "God Save the King", "In the Mood" and "Baa Baa Black Sheep" that were performed in the 1951 BBC recording

Note	Frequency (in Hertz)
$F\#_2$	92.5
G_2	98
A_2	110
B_2	123.47
C_3	130.81
$C\#_3$	138.59
D_3	146.83
E_3	164.81
$F\#_3$	185
G_3	196
A_3	220

Later tables show the measured frequencies that the computer produced

By dispensing with any reference to memory addresses, and abstracting from which particular instructions are employed, Turing's note-playing subroutines can be represented very transparently by means of what we call note-loops. The note-loop corresponding to Turing's C_5 routine is:

START – – – H – – – –REPEAT

Each "–" represents a single beat, with "H", the hoot, occurring on the fourth beat of the first bar. Representing this more economically still, the note-loop is simply:

$$<3\mathrm{H},4>.$$

8.6 Exploring the Mark II Notes

Subroutines for playing lower notes require the addition of further instructions, since this has the effect of adding extra blocks of 4 beats between the hoots, so lowering the frequency. Conveniently, one of Turing's instructions, /L, served to waste time: its execution took up 4 beats but it did nothing. (Strictly, the instruction did nothing unless a "dummy stop" switch had been set manually at the control console before the program started, in which case /L caused the machine to pause. We discuss dummy stops in Sect. 8.8.) /L is ideal for creating lower-frequency notes.[52] For

[52] /L's companion dummy-stop instruction /G serves just as well, as do so-called "dummy" instructions such as T£, TM, and TX: these have no effect except to cause a delay of 4 beats (Prinz (1952), p. 20; Turing (c. 1950), p. 58 and Fig. E.).

example, the note-loop <3H, 4, 4> produces a frequency of 347.22 Hz, approximately F_4 (349.23 Hz), a fifth lower than C_5.

<3H, 4, 4> produces the same note as Turing's second example of a loop, which in our notation is <3H, 3H, 4>. Adding the second pulse of sound at the same frequency does not alter the note, but (as Turing said) has the effect of making the note louder. We call note-loops that play the same frequency *equivalent*.

Some further examples of note-loops are <3H, 4, 4, 4, 4, 4, 4, 4>, producing a frequency of 130.21 Hz, fairly close to C_3 (130.81 Hz), and <3H, 4, 4, 4, 4, 4, 4>, producing a frequency of 148.81 Hz, lying between D_3 (146.83 Hz) and D_3 sharp (155.56 Hz). Both these note-loops produce a quiet sound. The same notes are played more loudly if extra hoots are added to form equivalent note-loops, such as <3H, 3H, 3H, 3H, 4, 4, 4, 4> and <3H, 3H, 3H, 3H, 4, 4, 4> respectively. We call a note-loop containing only one hoot the *primary* form, and equivalent note-loops containing more than one hoot *padded* forms of the loop. Padded note-loops typically produce notes with a different timbre or tone-color from the note produced by the loop's primary form. Timbre is manifested by differences in the shape of waveforms of the same frequency. (If a violin and a flute play exactly the same note at exactly the same volume, the sounds are nevertheless instantly recognizable as different, because of their different timbres.)

We built a simulator in order to investigate the effects of padding note-loops, and also to establish that our calculated note-loops really do play the correct notes. Our simulations of the Mark II playing "God Save the King", "In the Mood" and "Baa Baa Black Sheep" can be heard at www.AlanTuring.net/MarkII_music_simulations.mp3. An Atmel ATmega168 microcontroller was used to create a functional computer simulation of the Mark II as a note-playing device. We connected a small loudspeaker directly to one of the digital output pins. Microcontroller programs using pulses and delays reproduced the beat-structure of the Mark II and emulated the effects of the Mark II's music routines. We found that primary note-loops produce relatively thin-sounding notes while their padded equivalents produce somewhat louder, fuller-sounding notes. Over-padding is possible, however. The simulator revealed that including too many hoots adds a high overtone, especially with lower notes containing more beats. Because an uninterrupted sequence of hoot instructions generates the Mark II's highest achievable note of 1041.67 Hz (somewhere in the vicinity of C_6), the result of over-padding a note-loop is that the ear tends to hear not only the intended note but also this maximum note as a high overtone.

The BBC recording indicates that the programmer most likely used padding. If only unpadded loops are used, lower notes are quieter than higher notes, since in a lower note there are longer gaps between the hoots. This is not observed in the recording, and in fact some lower notes are louder than some higher notes. However, because of the poor quality of the recorded material, the analysis described here did not reveal the number of hoots used in each individual note-loop. Our reconstruction of the note-loops in Sect. 8.12 retrieves the primary form of the note-loops only.

Although the normal rhythm of the Manchester computer was 4 beats to the bar, some instructions took 5 beats to execute. Incorporating a suitable 5-beat instruction

Table 8.2 The frequencies that the Manchester Mark II was able to play by means of loops containing 4-beat instructions or mixtures of 4- and 5-beat instructions (down to the lowest frequency in the 1951 BBC recording)

Beats	Frequency (in Hertz)
8	520.83
12	347.22
13	320.51
16	260.42
17	245.10
18	231.48
20	208.33
21	198.41
22	189.39
23	181.16
24	173.61
25	166.67
26	160.26
27	154.32
28	148.81
29	143.68
30	138.89
31	134.41
32	130.21
33	126.26
34	122.55
35	119.05
36	115.74
37	112.61
38	109.65
39	106.84
40	104.17
41	101.63
42	99.20
43	96.90

By increasing the number of beats in a loop still further, the machine will play ever lower notes, until at approximately 20 Hz the human ear begins to perceive a series of individual clicks rather than a note

in note-loops (e.g. Turing's instruction TN[53]) extends the number of playable notes. For example, adding 10 extra beats to either the primary or the padded form of the 148.81 Hz note-loop displayed previously results in a loop that plays 109.65 Hz, very close to A_2 (110 Hz); the primary form is <3H, 4, 4, 4, 4, 4, 4, 5, 5>. The following loop plays the low note F_2 sharp ($F\#_2$): <3H, 4, 4, 4, 4, 5, 5, 5, 5, 5>. This

[53] Turing (c. 1950), Fig. E.

loop produces 92.59 Hz, fractionally higher than the note's Equal Tempered frequency of 92.5 Hz.

In what follows, note-loops are sometimes written in an abbreviated form. For example, <3H, 4×7> replaces <3H, 4, 4, 4, 4, 4, 4, 4> and <3H, 4×4, 5×5> replaces <3H, 4, 4, 4, 4, 5, 5, 5, 5, 5>. Table 8.2 shows the full range of frequencies, down to 96.9 Hz, that the Mark II could produce by means of note-loops containing 4- or 5-beat instructions.

8.7 Hoot-Stop and Radio Stop

As Turing explained in his tutorial, a fundamental use for <3H, 4> was the so-called "hoot-stop". If the two lines of code displayed in the tutorial were placed at the end of a routine (or program, as we would say today), then once the routine finished running, the computer would sound C_5 continuously until the operator intervened. The intervention might take the form of pressing the "KEC" key at the control console—the "clear everything" key—in order to clear out the completed routine, in preparation for running the next.[54] Without the convenient hoot-stop facility, the operator was obliged to remain at the console, watching the indicators, in order to tell whether the routine had stopped running.

A very different solution to effectively the same problem was found in the case of BINAC, an early US computer. Herman Lukoff, one of BINAC's engineers, explained that the technician whose job it was to monitor BINAC through the night, Jack Silver, had to spend all his time "looking at the flashing lights; it was the only way of knowing that the computer was working". One night Silver switched on a radio to alleviate the monotony:

> To Jack's surprise, all kinds of weird noises emanated from the loudspeaker, instead of soothing music. He soon realized that the churning BINAC generated these noises because as soon as it halted, the noises stopped. … He put the computer-generated tones to good use. Jack found that by turning the volume up he was able to walk around the building and yet be immediately aware of any computer stoppage.[55]

BINAC, built in Philadelphia by Presper Eckert, John Mauchly, and their engineers at the Eckert-Mauchly Computer Corporation, was the stored-program successor to the pioneering Eckert-Mauchly ENIAC. Eckert and Mauchly went on to build UNIVAC, one of the earliest electronic digital computers to enter the marketplace.

The first systematic use of the Manchester Mark I's programmable hooter appears to have been to provide the hoot-stop facility.

[54] Dodd (c. 1953), p. 32.

[55] Lukoff (1979), pp. 85–86.

8.8 First Hoots

In a section of his *Handbook* devoted exclusively to the Mark I machine, Turing made it clear that the programmable hooter predated the Mark II machine.[56] /V was Mark II notation; in the Mark I era it was the instruction K (11110) that caused the loudspeaker to sound. The Mark I, which was closed down in the summer of 1950,[57] was a slower machine than the Mark II; the duration of a beat was 0.45 ms, compared with the Mark II's 0.24 ms. This considerably reduced the number of playable notes. For example, lengthening the beat to 0.45 ms causes the frequency of <3H, 4> (the highest-frequency loop) to drop from 523.25 Hz to 277.78 Hz, approximately $C\#_4$.

It is not known precisely when a programmable hooter was first added to the computer. Geoff Tootill's laboratory notebook is one of the few surviving documents relating to the transition from Baby to Mark I.[58] In a notebook entry dated 27 October 1948, Tootill listed the K instruction 11110 among the machine's 32 instructions, but indicated that it was unassigned at this time. Given Turing's focus on the programming side, and the emphasis he placed on the use of the hoot instruction and pause-instructions, which he called "dummy stops", for "testing"—i.e. debugging—new routines, it seems likely that the hooter was incorporated earlier rather than later.[59] The computer was running complex routines by April 1949, in particular a routine that searched for Mersenne primes (primes of the form 2^n-1).[60] Most likely Turing's debugging toolkit of hoots and dummy stops was introduced earlier than this. It also seems likely that the use of the K instruction, and the use of loops to increase the volume of the hooter's native clicks, probably occurred more or less simultaneously. The loud note produced by the loop would have been more useful than the quiet click given by a single instruction. As Dietrich Prinz, a regular user of the Mark I, said, "By programming a simple loop containing this instruction … an *audible* 'hoot' is emitted".[61]

A table in Tootill's notebook dated 28 November 1948, showing the machine's instructions at that time, listed three different dummy stops, N, F, and C. The section of Turing's *Handbook* dealing with the Mark I explained that, during the checking of a program, the dummy stops N, F, and C would be operated in conjunction with the hoot instruction K. By the time of the 28 November table, the K instruction had been assigned: Tootill listed its function as "Stop". However, his table also contains another instruction labeled "Stop" (00010). Since the machine had no need of two ordinary stop-instructions, it seems very likely that K was being used for hoot-stop at this time. When execution of the program reached the point where the hoot-stop

[56] Turing (c. 1950), pp. 87–88.

[57] Williams and Kilburn (1952), p. 59.

[58] Tootill (1948–9), table of the machine's instructions dated 27/10/48.

[59] Turing (c. 1950), pp. 24, 88.

[60] Lavington (1980), p. 37.

[61] Prinz (1952), p. 20 (our italics).

had been inserted, execution would pause and the hooter would play the note of $C\#_4$ (middle C sharp) continuously until the operator intervened. We conclude that the Mark I was playing at least one note in about November 1948.

8.9 Other Early Music

The Manchester Mark I was not the only zeroth-generation electronic stored-program computer to play music. Trevor Pearcey's Sydney-built CSIRAC (pronounced "sigh-rack") had a repertoire that included "Colonel Bogey", "Auld Lang Syne", and "The Girl with the Flaxen Hair", as well as brief extracts from Handel and Chopin. Some of the music routines survived on punched paper tape, but seemingly no audio recordings were preserved. Australian composer Paul Doornbusch has recreated some of the music, using reconstructed CSIRAC hardware and the surviving programs.[62] CSIRAC, still complete and almost in working order, is in Melbourne Museum.

Doornbusch's recordings and the BBC's Manchester recording show that the programmers of both computers ran into the problem of "unplayable notes"—notes that could not be replicated or even closely approximated by means of an available note-loop. An example is the note of D_3, which occurs 5 times in the BBC recording of "God Save the King". This note's Equal Tempered frequency is 146.8 Hz, but the closest that the Mark II can approach is the significantly different note of 148.81 Hz, discussed in Sect. 8.6. To judge from the Doornbusch recordings, $F\#_2$, G_2, $C\#_3$, $F\#_3$, D_4, E_4, F_4, G_4, and A_4 were particularly troublesome for CSIRAC. The Australian and British solutions to the problem of unplayable notes were distinctively different. The Manchester programmers opted to use the nearest playable frequency and tolerated the melody being less in tune (see Sect. 8.14 for a fuller discussion of this technique). CSIRAC's programmers, on the other hand, attempted to mimic the unplayable frequency by rapidly moving back and forth between two playable frequencies that bracketed the note in question. The result was a melody in which tuning-related problems were replaced by timbre-related problems, with the Australian technique producing notes that sound grainy and unnatural.

An embryonic CSIRAC first ran a test program in about November 1949.[63] The computer seems to have been partially operational from late 1950, and in regular operation from about mid 1951. The date when CSIRAC first played musical notes is unrecorded; presumably this was in late 1950 or in 1951. The computer is known to have belted out tunes at the first Australian Conference on Automatic Computing Machines, held at Sydney University in August 1951.[64] A 2008 BBC News article, based on Australian sources, stated that CSIRAC was the first computer to play

[62] Doornbusch (2005). The book includes a CD of recreated music.

[63] McCann and Thorne (2000), p. 2.

[64] McCann and Thorne (2000), p. 3; Doornbusch (2005), pp. 24–25.

music.[65] The only evidence offered for this claim was that CSIRAC's performance at the Sydney Conference allegedly preceded the date of the BBC recording of the Manchester computer (the recording described in Sect. 8.4). However, the date of the BBC recording is in fact unknown; and in any case the Manchester computer's first performance of "God Save the King"—whose precise date is also unknown—would have occurred some days or weeks or even months before the BBC recording was made. Unfortunately, an Australian contribution to *The Oxford Handbook of Computer Music* also states, without evidence, that CSIRAC was "the first computer to play music".[66] CSIRAC, however, was certainly not the first computer to play music.

There were American hoots too. BINAC was playing music before CSIRAC even ran its first test program. BINAC was completed in August 1949 (although it ran a 50-line test program in April of that year).[67] As Lukoff explained, a party was held to celebrate the machine's completion:

> It was held right at the BINAC test area one August evening. In addition to hors d'oeuvres and cocktails, the BINAC crew arranged a spectacular computer show. Someone had discovered that, by programming the right number of cycles, a predictable tone could be produced. So BINAC was outfitted with a loudspeaker attached to the high speed data bus and tunes were played for the first time by program control. The audience was delighted and it never occurred to anyone that the use of a complex digital computer to generate simple tones was ridiculous. … The crowning achievement of the evening came after a long, laborious arithmetic computation; the machine laid an egg! The engineers had programmed the machine to release a hard-boiled egg from its innards.[68]

As far as can be ascertained, therefore, the first melodies to be played by a computer were heard at the Eckert-Mauchly Computer Corporation in the summer of 1949, and very likely individual musical notes were heard a few months earlier at Turing's Computing Machine Laboratory, probably in November 1948. Effectively, the pioneering developments on either side of the Atlantic were roughly contemporaneous, with Australia entering the field a year or two later.

8.10 The BBC Recording

The BBC's website offers an edited digitized version of the original BBC recording of the Manchester Mark II; and there is a full-length version of the same digitalization of the recording on the Manchester University website.[69] Upon pressing play, the listener is greeted by a thick wall of noise—a combination of hissing, humming,

[65] Fildes (2008).

[66] Dean (2009), pp. 558, 584.

[67] Lukoff (1979), p. 84.

[68] Lukoff (1979), p. 86.

[69] Fildes (2008); http://curation.cs.manchester.ac.uk/digital60/www.digital60.org/media/mark_one_digital_music/index.html.

and rhythmically repeating crackles from the original acetate disc. Then a tone not unlike a cello cuts through this cacophony to give a mechanical-sounding rendition of the first two phrases of the National Anthem. The melody, though familiar enough, is somewhat out of tune, with some notes more distinctly out than others. Moreover, some notes are loud relative to their neighbors (most likely the result of padding). At the end of the second phrase, the performance is suddenly cut short by a glitch and nervous laughter.

The engineers restart the routine and this time the machine energetically plays its way through the entire first verse. Then, with scarcely a pause, it follows up with an unbroken performance of the first line of "Baa Baa Black Sheep". For its third number the Mark II attempts "In The Mood", but once again falls victim to an unknown error, causing it to sing out a high pitched beep. The recording team give the computer one more chance to make its way through "In The Mood", and it proceeds admirably until the final line, when it yet again breaks down. Altogether, the recording lasts about 3 minutes.

The frequencies in the recording are higher than the ideal in-tune frequencies that would be heard if the same melodies were played on a modern synthesizer. Because the Mark II offered such a limited palette of notes, deviations such as this are to be expected. In scoring a melody for the Mark II, the musician had to try to ensure that the individual notes were as in tune with one another as possible. Whether or not notes were played at the ideal Equal Tempered frequency would necessarily have been a secondary consideration.

The waveform of the recording looks and sounds very unlike the square pulse-wave produced by a modern synthesizer. The early waveform is much rounder, with a lot less high-frequency energy. Its departure from the ideal square-wave is most likely a result of the Mark II's pioneering digital electronics, together with the long and transformative signal chain between the original sound source and the final digital file. Obtaining clean pulse waves from early digital electronics was a challenge in itself; the pulses from the computer would have resembled trapezoids rather than perfect squares, and so would already be lacking some of the high frequency audio content of modern pulse waves. The hooter, the next element in the chain, may itself have transformed the sound somewhat, depending on the characteristics of the loudspeaker. Then the microphone used to record the performance and the amplifier used to boost the signal, as well as the room in which the performance took place, would all have colored the audio. Finally, the cutting mechanism, which gouged a groove in the rotating acetate disc, was a source of pitch instability and plenty of crackles and noise. Each of these stages provided generous opportunities for the introduction of unwanted noise, artifacts, pitch instability, and spectral alterations.

Two different acetate discs were cut during the recording session. One was taken away by the BBC and presumably used in a broadcast. It is unlikely that this disc survives, but a second disc was given to Manchester engineer Frank Cooper as a souvenir. It contained another recording made at Cooper's request once the main

recording session was over.[70] By that time, Cooper recollected, "the computer was getting a bit sick and didn't want to play for very long". Eventually he donated this 12-inch single-sided acetate disc to the Computer Conservation Society; and subsequently the National Sound Archive, now part of the British Library, made a digital preservation copy of the recording.[71] This further stage of processing may have added its own sonic characteristics.

8.11 The Paradox of the Impossible Notes

Despite the long chain existing between the original music-playing routine and the available recording, we decided to attempt to reverse-engineer the note-loops that the programmer had used. A frequency analysis of the digital recording told us the frequencies of the recorded notes, and where different recorded occurrences of the same note had different frequencies, we were able to gather information about error magnitudes. (The principal software used was iZotope RX for removing noise and artifacts, and Celemony Melodyne for carrying out the frequency analyses and pitch correction.)

Soon, however, we hit a fundamental problem. There were frequencies in the recording that could not possibly have been produced by the Mark II's note-loops—*impossible* notes. An example is the recorded note corresponding to E_3 (which occurs only once in the verse from the National Anthem). The measured frequency of the recorded note is 169.06 Hz, a thumping 4.25 Hz distant from the note's Equal Tempered frequency of 164.81 Hz. 169.06 Hz is an impossible note for the Mark II. It is bracketed by a note-loop of 24 beats, producing a note of 173.61 Hz, and a loop of 25 beats, producing a note of 166.67 Hz (see Table 8.2). Whatever frequency it was that the computer had produced, this was certainly not the frequency we found in the recording on the BBC website.

We decided to move closer to the original recording, assuming that the problem lay in the transformative chain between the source and the digital file we were analyzing; and we obtained from the British Library an identical copy of their archived digital file. The recording we had analysed previously from the BBC website was in fact a digitized version of an *analog* recording that had been made by the National Sound Archive—using a cassette tape and Dolby B noise reduction—at the same time as the digital preservation copy was created.[72] (This cassette recording was subsequently digitized by a technician at Manchester University and put on the Internet as part of Manchester's *Digital 60* celebration of Baby's 60th anniversary.) The digital file as we received it from the British Library did indeed exhibit differences from the previously analysed (cassette-based) recording—but not the differences we had hoped for. The new frequencies we measured were consistently higher

[70] Cooper interviewed by Burton.

[71] British Library Sound Archive reference number H3942.

[72] Information from Chris Burton.

Table 8.3 The measured frequencies of the notes in the National Sound Archive digital copy of the BBC recording, prior to correction

Note	Measured frequency (in Hertz)
$F\#_2$	97.07
G_2	101.78
A_2	113.19
B_2	127.42
C_3	135.74
$C\#_3$	145.90
D_3	156.96
E_3	169.54
$F\#_3$	194.42
G_3	203.56
A_3	241.16

than our previous measurements (indicating that the recording involving the cassette tape ran slower than the British Library's preservation copy). However, the impossible notes were still present, as Table 8.3 shows. E_3 is the most troubling case but the high notes $C\#_3$, D_3, $F\#_3$, and G_3 all stand out as being quite far off the playable frequencies shown in Table 8.2.

Although the software used for our frequency analyses is very reliable, the measured frequencies in Table 8.3 nevertheless have a significant margin of error. This is because of a wobble in the speed of the recording that was most likely introduced by the disc-cutting process. This wobble caused some notes in the recording to bend slightly throughout their duration, and the wobble also resulted in the measured frequency of different occurrences of a note being different from one another. The maximum difference between two readings of the same note from different parts of the recording was one fifth of a semitone (20 cents, in the unit of measurement introduced below). The frequencies shown in Table 8.3 are the midpoints of the range of frequencies measured for the note's various occurrences. However, the margin of error in the measured frequencies created by the wobble (±20 cents) is not large enough to explain the presence of the impossible notes. A more probing analysis was needed.

8.12 Searching for the Right Speed

The difference in speed and consequently in pitch between the two analysed recordings is in fact the clue to the impossible notes. If the speed of the archived recording were itself fast or slow, this could account for the presence of impossible frequencies. For example, if the turntable in the BBC recording van were running slightly too fast as the acetate disc was cut, the frequencies would be shifted systematically

upon playback at the standard speed of 78 rpm.[73] (Achieving speed constancy was a problem with the BBC's standard mobile recording equipment.[74]) Only when we know the true speed S_0 at which the recording should be played can we measure the frequencies actually produced by the computer.

Therefore we set out to determine the true speed of the recording. The basic question was whether some small increase or decrease in the overall speed of the recording would result in frequencies that correlated acceptably with the playable frequencies in Table 8.2. Our initial approach to the question was to nudge the speed up or down a bit and see what frequencies emerged. The result was always the same: some frequencies matched well but other frequencies that had matched at a speed investigated previously now no longer matched. How to get the carpet to fit into all the corners of the room at once? We wrote a command-line program to conduct a brute-force search for the optimum global fit. The program (written in C) incremented the speed of the recording in tiny steps and at each increment calculated the new notes, by appropriately tweaking the notes we had measured. We call the notes calculated at each increment the *calculated* notes, the notes that the computer is actually capable of producing the *playable* notes (shown in Table 8.2), and the notes that we measured from the BBC recording the *measured* notes (shown in Table 8.3).

The exhaustive search commenced at 79.5% of the native speed of the recording, an arbitrary point well below the native speed, and progressed through 80,000 increments to 126% of the native speed. At each step, the program computed the 'distance' between each calculated note and its most closely neighboring playable note. The program averaged these distances to produce Φ_S, the closeness-of-fit parameter for speed S; and then the process repeated, with the program incrementing S by a small fixed amount δ and calculating $\Phi_{S+\delta}$.

Our program expressed the distance between a calculated note and the nearest playable note in musical units called "cents". Cents were also the units of Φ_S. One cent is one-hundredth of the distance between any two adjacent keys on the piano keyboard. To the human ear, the distance between, say, the notes C_4 (261.64 Hz) and C_5 (523.28 Hz), which are one octave apart, is musically the same as the distance between C_5 and C_6 (1046.56 Hz), which are again one octave apart. Working in Hertz, however, these distances are far from the same: the distance between C_4 and C_5 is 261.64 Hz, whereas the distance between C_5 and C_6 is 523.28 Hz. Musically, the distances are the same, but the *frequency* of the notes is doubling. If these distances are expressed in cents, on the other hand, they are indeed identical. Since C_4 and C_5 are 12 notes apart on the keyboard, as are C_5 and C_6, the distance is 1200 cents in each case. In fact, not only distances between notes but also the individual notes themselves can be expressed in cents, e.g. by somewhat arbitrarily taking C_0 (16.35 Hz) to be the zero-point, so that C_1 is 1200 cents, C_2 is 2400 cents, and so on.

The program expressed all values in cents rather than Hertz, a procedure that yields a simpler algorithm, since each increment in speed has a uniform effect on the notes if the units are cents, whereas the formula is more complex if the units are

[73] BBC (1950), p. 26.

[74] BBC (1950), pp. 49, 52.

Table 8.4 The frequencies played by the Mark II once the speed of the 1951 BBC recording is corrected

Note	Frequency (in Hertz)
$F\#_2$	99.21
G_2	104.02
A_2	115.69
B_2	130.23
C_3	138.73
$C\#_3$	149.12
D_3	160.43
E_3	173.29
$F\#_3$	198.71
G_3	208.05
A_3	246.48

Hertz. Moreover, if the notes are expressed in cents, the distance between two notes can be calculated simply by subtracting the smaller value from the larger, whereas matters are more complicated if the notes are expressed in Hertz, as our examples show. A calculated note could be, say, 10 Hz higher than the nearest playable note (100Hz, say), while at a higher speed, a calculated note could be 20 Hz higher than its nearest playable note (say 200 Hz), and yet each of the two notes be equally close to its nearest playable note, since differences in frequency are magnified as the speed becomes higher.

The object of our search, the smallest Φ_S, turned out to be just 37.7 cents, and the associated speed, the calculated true speed S_0, was just 2.2% faster than the native speed of the recording. If the speed of the recording is increased by this small percentage—or, equivalently, if our previous measurements of the notes are corrected by adding 37.7 cents to each measured note—then all the notes lie within 9.67 cents of playable notes, well within the margin of error of ±20 cents. In fact, the situation is even better than this. If A_3 is disregarded, all the notes are within 3.52 cents of playable notes. (There are only four samples of A_3 in the entire recording, all in quick succession at the end of "In the Mood".)

The "impossible" notes have disappeared. They were artifacts, caused by the recording becoming slightly slowed down at some point in the transformative chain—most likely in the cutting of the original acetate disc.

Table 8.4 shows the measured notes after adjustment for the calculated true speed. Having discovered the true frequencies, with a margin of error of only a few cents, we were able to conclude that the routines used to play the versions of "God Save the King", "Baa Baa Black Sheep" and "In the Mood" recorded by the BBC must have used note-loops whose primary forms are shown in Tables 8.5 and 8.6. As Sect. 8.6 explained, the programmer(s) probably used padded forms of the primary loops, but our analysis as described here retrieved only the primary forms.

Table 8.5 The primary note-loops used to play the version of "God Save the King" recorded by the BBC, as indicated by our frequency analysis

Note	Beats	Primary Note-Loop
$F\#_2$	42	<3H, 4×7, 5×2>
G_2	40	<3H, 4×9>
A_2	36	<3H, 4×3, 5×4>
B_2	32	<3H, 4×7>
C_3	30	<3H, 4×4, 5×2>
D_3	26	<3H, 4×3, 5×2>
E_3	24	<3H, 4×5>

The programmer may have used padded forms of the primary loops, as Sect. 8.6 explained, but the quality of the recorded material is too poor for the present analysis to retrieve information about the way the padding was done

Table 8.6 The primary note-loops for additional notes in the melodies "Baa Baa Black Sheep" and "In the Mood"

Note	Beats	Primary Note-Loop
$C\#_3$	28	<3H, 4×6>
$F\#_3$	21	<3H, 4×3, 5>
G_3	20	<3H, 4×4>
A_3	17	<3H, 4×2, 5>

8.13 The Question of Authorship

The question of the authorship of the three routines that played the melodies recorded by the BBC is open. In the wake of Strachey's tour de force a number of people in the computing lab started writing music programs: Cooper related that "everybody got interested—engineers started writing music programs, programmers were writing music programs".[75] Nothing about the BBC recording settles the question of authorship: even the routine that played the National Anthem in the recording may have been a retouched version of Strachey's original.

However, it can at least be said that the programmer(s) of the routines for "Baa Baa Black Sheep" and "In the Mood" used the same key signature as the programmer of "God Save the King"; and also used the same primary loops as those selected for "God Save the King": new loops were introduced only for notes that do not occur in the Anthem. This was so even though some alternative primary loops were available, and in fact it is arguable that some of these choices would have produced frequencies that sounded more in tune (see Sect. 8.14).

We expect that a more refined analysis currently underway will reveal information about the use of padding in the recorded melodies: this may help to clarify aspects of the authorship issue.

[75] Cooper in interview with Burton.

8.14 Tuning Puzzles

There are puzzles about some of the frequencies selected by the programmer(s) and so we end with a discussion of tuning.

While there are always differences between the Equal Tempered frequency of a note and the note actually played by the Mark II, these differences do not produce a general out-of-tune-ness in the performance. Certainly if the computer were playing together with some other instrument that was tuned in accordance with the Equal Tempered scale, the computer would sound conspicuously out of tune. However, with the computer playing solo, what matters more than the absolute frequencies of the notes is the extent to which the performed notes are in tune with one another, and in fact the relative tuning of the notes is quite acceptable, actually closer to the key of G# major than G major, the usual key of the National Anthem. In programming the National Anthem, the programmer seems to have picked frequencies with a view to their overall relationships, rather than trying to hit the Equal Tempered frequencies as closely as possible.

Nevertheless, some notes in the recording are sufficiently out of tune to make anyone with a musical ear cringe. Examples are the sequence of occurrences of D_3 at notes 17 to 20 of "God Save the King" and the final occurrence of D_3 at note 36. These notes would sound more in tune if a 27-beat loop <3H, 4×2, 5×3> had been used instead of the 26-beat loop selected by the programmer <3H, 4×3, 5×2>—as Fig. 8.5 indicates: the "difference line" at the foot of the graph would be flatter if the

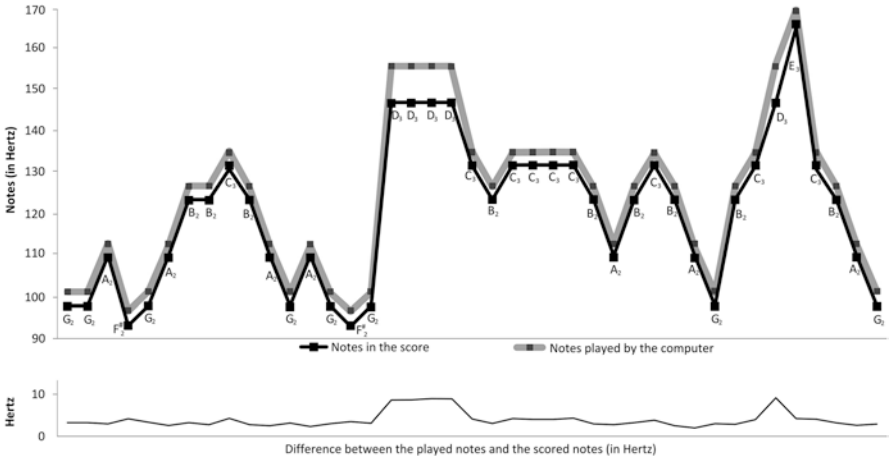

Fig. 8.5 Comparing the notes in the National Anthem's score with the notes actually played by the computer in the BBC recording. The line at the foot of the graph, the difference line, shows the displacement between the other two lines. The straighter the difference line, the more in tune with one another the notes sound. Here, the line shows that the performance sounds most out of tune in the middle and again almost at the end. In both places, the culprit is the computer's rendition of the note D_3, which the programmer pitched a little too high

27-beat loop were used. In fact, the preferable 27-beat loop even produces a frequency closer to the note's Equal Tempered frequency of 146.83 Hz (154.32 Hz as opposed to the 26-beat loop's frequency of 160.26 Hz). Why the programmer chose the excessively high 26-beat loop is puzzling.

Another puzzling example of poor tuning, this time from "In the Mood" is A_3, a note that does not feature in the other two melodies. Substituting an 18-beat loop in place of the 17-beat loop used by the programmer brings an improvement in relative tuning, and again produces a frequency closer to the note's Equal Tempered frequency of 220 Hz (231.48 Hz as opposed to 245.1 Hz in the case of the 17-beat loop). Once again, an excessively high frequency has been selected, to the detriment of the relative in-tune-ness of the notes.

8.15 Conclusion: Restoring the Recording

To put our findings to some practical archival use, we restored the British Library's recording. We increased the speed of the recording to match the original and used pitch-correction software to remove the effects of the wobble. We also filtered out extraneous noise from the recording.

Nobody had heard the true sound of the computer since the early Ferrantis were scrapped more than half a century ago. A German researcher David Link attempted to recreate the sound, by programming his emulation of the Mark II.[76] But an emulation is far from being the real thing, and without the original physical components, including of course the hooter, an emulation cannot recapture the actual sound. But now, thanks to an improbable meeting—in New Zealand—of the 1951 recording and modern analytical techniques, we really can listen to Turing's Mark II.

Our restoration is now in the British Library (reference number H3942) and can be heard at www.AlanTuring.net/historic_music_restored.mp3.

Acknowledgments With thanks to Chris Burton for assistance and advice. Copeland is grateful to the following institutions for supporting this research: University of Canterbury, New Zealand; University of Queensland, Australia; Federal Institute of Technology (ETH), Zurich, Switzerland; and Det Informationsvidenskabelige Akademi, Copenhagen University, Denmark.

References

BBC. 1950. *BBC Recording Training Manual*. London: British Broadcasting Corporation.

Bowker, G., and R. Giordano. 1993. Interview with Tom Kilburn. *IEEE Annals of the History of Computing* 15: 17–32.

[76] Link, D. 'Software Archaeology: On the Resurrection of Programs for the Mark 1, 1948–58', 2015, vimeo.com/116346967.

Campbell-Kelly, M. 1980. Programming the Mark I: Early Programming Activity at the University of Manchester. *IEEE Annals of the History of Computing* 2: 130–168.
Chadabe, J. 2001. The Electronic Century, Part III: Computers and Analog Synthesizers. *Electronic Musician.* www.emusician.com/tutorials/electronic_century3.
Copeland, B.J. 1999. The Turing-Wilkinson Lecture Series on the Automatic Computing Engine. In *Machine Intelligence*, ed. K. Furukawa, D. Michie, and S. Muggleton, vol. 15, 381–444. Oxford: Oxford University Press.
———. 2005. Introduction to 'The Turing-Wilkinson Lecture Series (1946–7)', (Turing (1946–7), in Copeland et al. (2005).
———. 2011a. The Manchester Computer: A Revised History. Part I: The Memory. *IEEE Annals of the History of Computing* 33: 4–21.
———. 2011b. The Manchester Computer: A Revised History. Part II: The Baby Machine. *IEEE Annals of the History of Computing* 33: 22–37.
———. 2012. *Turing, Pioneer of the Information Age*. New York: Oxford University Press.
Copeland, B.J. et al. 2005. *Alan Turing's Automatic Computing Engine*. New York: Oxford University Press.
———. 2006. *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*. New York: Oxford University Press.
Copeland, B.J., and G. Sommaruga 2015. The Stored-Program Universal Computer: Did Zuse Anticipate Turing and von Neumann?, in Sommaruga and Strahm (eds.) (2015).
Dean, R.T. (ed.). 2009. *The Oxford Handbook of Computer Music*. New York: Oxford University Press.
Dodd, K.N. c.1953. *The Ferranti Electronic Computer*. Armament Research Establishment report 10/53.
Doornbusch, P. 2005. *The Music of CSIRAC: Australia's First Computer Music*. Melbourne: Common Ground, 2005.
Fildes, J. 2008. 'Oldest' Computer Music Unveiled, *BBC News | Technology*, 17 June 2008, http://news.bbc.co.uk/2/hi/technology/7458479.stm.
Foy, N. 1974. The Word Games of the Night Bird (Interview with Christopher Strachey). *Computing Europe*, 15 August 1974, 10–11.
Lavington, S.H. 1980. *Early British Computers: The Story of Vintage Computers and the People Who Built Them*. Manchester: Manchester University Press.
Link, D. 2012/2013. Programming ENTER. Christopher Strachey's Draughts Program. *Resurrection* 60: 23–31.
Lukoff, H. 1979. *From Dits to Bits: A Personal History of the Electronic Computer*. Portland: Robotics Press.
McCann, D., and P. Thorne. 2000. *The Last of the First. CSIRAC: Australia's First Computer*. Melbourne: Melbourne University Press.
Prinz, D.G. 1952. *Introduction to Programming on the Manchester Electronic Digital Computer*, Moston, Manchester: Ferranti Ltd., 28 March 1952. A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/prinz.
Sommaruga, G., and T. Strahm (eds.). 2015. *Turing's Revolution: The Impact of His Ideas About Computability*. Basel: Birkhäuser/Springer.
Stern, N. 1979. The BINAC: A Case Study in the History of Technology. *Annals of the History of Computing* 1: 9–20.
———. 1981. *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*. Bedford, MA: Digital.
Tootill, G.C. 1948–9. Digital Computer—Notes on Design & Operation. National Archive for the History of Computing, University of Manchester.
Turing, A.M. 1946–7. *The Turing-Wilkinson Lecture Series (1946–7)*, in Copeland et al. (2005), pp. 464–527.
———. 1947. Lecture on the Automatic Computing Engine, in Turing and Copeland (2004), pp. 378–394.

———. 1948. Intelligent Machinery, in Turing and Copeland (2004), pp. 410–432.
———. c.1950. Turing, A. M. *Programmers' Handbook for Manchester Electronic Computer Mark II*, Computing Machine Laboratory, University of Manchester, no date, circa 1950. A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/programmers_handbook.
Turing, A.M., and B.J. Copeland. 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford: Oxford University Press.
Williams, F.C., and T. Kilburn. 1952. The University of Manchester Computing Machine. In *Review of Electronic Digital Computers: Joint AIEE-IRE Computer Conference*, 57–61. New York: American Institute of Electrical Engineers.