# Colossus and Programmability

Thomas Haigh, University of Wisconsin—Milwaukee & Siegen University (thomas.haigh@gmail.com) and Mark Priestley, Independent Scholar (m.priestley@gmail.com)

This is a preprint. Please quote or cite the final published version, which is Thomas Haigh & Mark Priestley, "Colossus and Programmability," *IEEE Annals of the History of Computing* 40:4 (Oct-Dec, 2018): 5-27

*Abstract: We analyze the capabilities of the Colossus codebreaking devices, built in 1943-1945 under the direction of Tommy Flowers of the UK General Post Office. Colossus is often described as a programmable computer, a misconception we trace to old battles about the "first computer" and to former secrecy about its actual capabilities. In fact Colossus was not called a computer at the time, and does not meet later definitions because it carried out no mathematical operation other than counting. Colossus automatically executed a program, i.e. performed series of discrete operations, but it was not programmable because that program could not be fundamentally modified by its users. Instead of the old focus on allocating "firsts", we argue for new perspectives on Colossus focused on its use and impact, its relationship to other early machines, and its place in the history of digital communications engineering.*

The original Colossus was built by the British General Post Office at the end of 1943, under the direction of telecommunications engineer Tommy Flowers, to assist in British attacks on certain German codes. Although it was recently honored on a Royal Mail stamp as "world's first electronic digital computer," its place within the history of computing remains ambiguous. Colossus is also often said to have been a "programmable electronic computer," again typically in order to claim it as the first such computer. Despite its fame, accounts of what Colossus was and what it could do are contradictory and no clear description of its control capabilities has been published. This is in large part the result of residual secrecy that clung to Colossus during the early 1980s when the nascent history of computing community hashed out the allocation of "firsts" among early machines before moving on to more productive matters.

In this paper we present a clear and concise account of the architecture and control capabilities of Colossus, grounded in primary sources declassified in recent decades. We are particularly concerned with the concept of programmability, which has never been properly defined in the history of computing literature. To establish broad and historically grounded definitions of "program" and "programmability", we looked at the contemporaneous emergence of the idea of "programming" in the ENIAC project and at the everyday use of "program" in other contexts. This led us to a sense of "program" as a series of operations carried out by a computer, which is applicable to Colossus even though these operations were embodied in its structure rather than being encoded symbolically as instructions. For this reason, the program of operations executed by the machine could not be fundamentally changed by users. We argue that it was not, therefore, programmable. We also challenge the idea of Colossus as a computer, which we suggest has more to do with a need to justify its importance in the context the discourse of the 1970s than with its actual capabilities. Despite making extensive use of digital electronics Colossus fits neither 1940s definitions of the term nor more modern ones. Statements made by Flowers himself, and some of those who worked with him, made more nuanced characterizations, for example as an "electronic processor."

Our conclusion that Colossus was neither programmable nor a computer, reached rather to our own surprise, does not diminishes its historical importance.[1] Rather, the special pleading previously undertaken to shoehorn Colossus into the role of programmable computer reflects the concerns of the history of computing community in its early days, and indeed the limitations of "history of computing" as an analytical frame. Colossus fits within that frame only to the extent to which it was a computer. Today, as our reliance on digital communications grows and computers vanish from view into digital devices and data centers we are better able to appreciate this remarkable machine on its own terms.

## Situating Colossus

Dozens of unique electronic and mechanical computers were built during the 1940s. A handful, such as the Harvard Mark 1, ENIAC, EDSAC, and the UNIVAC 1, have clear and prominent places in the history of computing. They consistently appear in overview histories, such as Martin Campbell-Kelly and Willian Aspray's *Computer* and Walter Isaacson's *The Innovators*, in television documentary series, and in comprehensive museum exhibitions such as those at the Computer History Museum and the Heinz Nixdorf Museums Forum. Each is remembered as the "first" machine to reach one or another historical milestone, agreed upon after a long and messy battle to anoint the "first computer."

The historical place of Colossus is less clear. Most other pioneering computers were publicized during their operational lifetimes. ENIAC, for example, was announced to the world with a front page story in the *New York Times* and installed in a showpiece facility where it was frequently displayed for visitors.[2] The Colossus machines were designed in secret, deployed as a vital part of one of the war's most militarily sensitive operations, and kept confidential for decades afterwards. From the 1940s to the 1970s, as teams of lawyers gathered records concerning other early machines and subjected their designers to repeated rounds of deposition and testimony, those responsible for Colossus remained quiet about the machine's capabilities and even its existence.

Word of Colossus began to spread in the 1970s after Brian Randell, a computer scientist with an interest in the early history of electronic computing, gathered testimony from veterans of the project and persuaded the UK government to acknowledge its existence. In 1976 he shocked the computer pioneers at a seminal computer history meeting at Los Alamos National Laboratory with news that "a series of programmable electronic digital computers was built in Britain during World War II, the first being operational in 1943." He characterized Colossus as "a special-purpose program-controlled electronic digital computer" that could "most aptly be compared" to ENIAC in its flexibility and programming method.[3]

[1] We previously endorsed the idea of Colossus as a digital electronic computer based on claims in the secondary literature, but had not been able to form a very clear impression of what it actually did or how it was used. Cf. Thomas Haigh, Mark Priestley, and Crispin Rope, *ENIAC In Action: Making and Remaking the Modern Computer* (Cambridge, MA: MIT Press, 2016). Realizing this ignorance inspired the research described here.

[2] Ibid.

[3] A revised version was published in the seminal volume Brian Randell, "The Colossus", in *A History of Computing in the Twentieth Century*, ed. N Metropolis, J Howlett, and Gian-Carlo Rota (New York: Academic Press, 1980):47-92.

Particularly in Britain, the high public profile of Colossus comes in large part from its connection with the work of Bletchley Park, which has become one of the most celebrated facets of the war effort. Colossus even makes a brief appearance in *Cryptonomicon*, Neal Stephenson's hugely popular novel of cryptography and the wartime origins of information technology. For this reason, Colossus is more often considered as part of the history of codebreaking than of the history of computing or of telecommunications. It was recently honored with a postage stamp (figure 1).

Within the history of computing, Colossus is celebrated by a small community of enthusiasts, some of whom view it as the most important of all the early computers. For example, Jack Copeland has claimed that if the Colossus machines had been preserved as "the heart of a scientific research facility" then "the Internet—and even the personal computer—might have been developed a decade or more earlier."[4] Fans of Colossus, like those of other early computers, sometimes strike a partisan tone.[5] Copeland, for example, attributes to Flowers a remark that "Colossus was 'much more of a computer than ENIAC'" which was "just a 'number cruncher.'"[6] Tony Sale, who devoted his retirement to the construction of a new Colossus to replace those disassembled at the end of the war, was motivated by a sense that "for far too long the Americans have got away with the myth that the ENIAC was the first large-scale electronic digital calculator in the world." Following the completion of the reconstruction, he claimed, "There has been a stunned silence from across the water."[7]

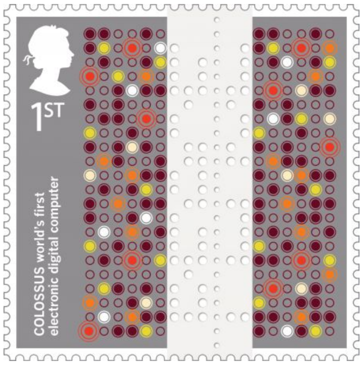

*Figure 1: The Colossus commemorative stamp, issued by the Royal Mail in 2015, calls it the "world's first electronic digital computer."*

It is far from clear that the silence noted by Sale is the result of stunned acquiescence to his claims. Colossus has been marooned between the praise of boosters, who believe it to have a singularly important place in the history of computing, and the scholarly authors of overview histories of computing, who have politely ignored these claims. In *Computer*, Campbell-Kelly

[4] B Jack Copeland, *Turing: Pioneer of the Information Age* (New York, NY: Oxford University Press, 2013), 119.

[5] This is particularly noticeable with fans of computer pioneer John Atanasoff, such as Alice Burks, *Who Invented the Computer: The Legal Battle that Changed Computing* (New York, NY: Prometheus Books, 2003).

[6] Jack Copeland, "Colossus and the Rise of the Modern Computer", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):101-115

[7] Tony Sale, "The Colossus Rebuild Project", Codes and Ciphers, n.d., accessed July 4, 2018, http://www.codesandciphers.org.uk/lorenz/rebuild.htm

and Aspray note merely that Bletchley Park's work on mechanical devices to attack the Enigma code "was followed by an electronic machine, the Colossus, in 1943," as a result of which several people who would later work on computer projects were exposed to electronic technologies.[8] The other standard scholarly history of computing, Paul Ceruzzi's *A History of Modern Computing* doesn't mention Colossus at all.[9]

The current situation, then, is one in which Colossus is largely ignored by historians shaping broader narratives on the emergence of modern computing but lavishly, if sometimes shrilly, praised by its fans and increasingly embraced by the British public as a symbol of national greatness. This reflects an enduring vagueness about what Colossus actually did. Randell dug up an impressive amount of information, but without access to original documents his account was unavoidably speculative. Randell's revelations led to considerable interest in Tommy Flowers, who gave several public talks and interviews in the 1970s and published his own technical article on Colossus and its history in 1983. Yet even Flowers was working from memory, and his description turned out to have several significant historical and technical inaccuracies.[10] More recent investigations have provided masses of detail but not a clear and concise overall description of the architecture and capabilities of Colossus separate from the codebreaking applications for which it was designed, or of the methods by which it was prepared for use.[11] To bridge this gap we have gone back to the primary sources to clarify what Colossus did and to probe its similarities and differences with early electronic computers.

## Was Colossus a Computer?

There are two senses in which a historical object might be considered to be a computer. One is if people at the time called it a computer, even if definitions have subsequently shifted. That would apply, for example, to the human computers in Philadelphia calculating firing tables for the US Army in 1942, or the analog torpedo computer used in submarine warfare.[12] Historians call  a term used in original sources an "actors' category" and would typically look at when people first started using it, and how its meanings were contested and changed over time.

[8] Martin Campbell-Kelly and William Aspray, *Computer: A History of the Information Machine* (New York, NY: Basic Books, 1996), 99-100. The more recent third edition retains the same text (p.82).

[9] Paul E Ceruzzi, *A History of Modern Computing* (Cambridge, MA: MIT Press, 1998). Ceruzzi gives Colossus about a page in his more recent *Computing: A Concise History*. He asks why Colossus is not "more heralded," and concludes that its focus on textual rather than numerical operations had combined with the long prevalent secrecy to marginalize it.

[10] Thomas H Flowers, "The Design of Colossus," *Annals of the History of Computing* 5, no. 3 (Jul-Sep 1983):239-252. For example, Flowers remembered the electronic buffer introduced on Colossus 2 as buffering the outputs of five channels of simulated cipher wheel output, but later analysis showed that it buffered only one channel. A detailed analysis of this episode can be found in James A. Reeds, Whitfield Diffie, and J V Field, *Breaking Teleprinter Ciphers at Bletchley Park. An Edition of General Report on Tunny With Emphasis on Statistical Methods (1945)* (Piscataway, NJ: IEEE Press/Wiley, 2015), 606-608.

[11] Colossus has been explored in Jack Copeland, *Colossus: The First Electronic Computer* (New York: Oxford University Press, 2006), Paul Gannon, *Colossus: Bletchley Park's Greatest Secret* (London, UK: Atlantic Books, 2006), Harvey G Cragon, *From Fish to Colossus* (Dallas, Texas: Cragon Books, 2003) and Reeds, Diffie, and Field, *Breaking Teleprinter Ciphers*.

[12] David A Mindell, *Between Human and Machine: Feedback, Control, and Computing Before Cybernetics* (Baltimore: Johns Hopkins University Press, 2002).

The second sense in which we could interpret the term is to ask whether an object, though not necessarily called a computer at the time, would meet a later definition of computer. For example, Babbage's analytical engine would, if constructed, have had comparable capabilities to some of the computers of the 1940s. Historians refer to this a term used in this way as an "analysts' category." This requires us to endorse a particular definition of the term, or formulate a new one, and apply it consistently across time regardless of actual historical usage.

Colossus was not a computer in the first sense. Only many years after the Colossus machines were shut down did anyone begin to call them computers or even calculators. This sets Colossus apart from most of the pioneering electronic computers of the 1940s, which were usually named either as computers (the "C" in ENIAC stood for "Computer" as did the "C" in EDVAC) or calculators (the "C" in machines such as EDSAC and IBM's SSEC). This is because the new machines replaced the labor of humans, whose job title was "computers."[13] They were often known as "automatic computers," just as machinery that could direct the flying of a plane was called the "automatic pilot" because it carried out some of the tasks of a human pilot.

Neither was Colossus a computer in the second sense, as it does not meet any plausible subsequent definition of a computer. Computers carried out lengthy mathematical tasks, often involving thousands of individual mathematical operations. This drew attention to their ability to move from one operation to the next without human intervention. George Stibitz built a series of pioneering tape-controlled computers at Bell Labs during the 1940s. His 1945 definition captures the contemporary understanding of a computer as something able to perform automatically a sequence of operations ("some or any of" multiplications, divisions, additions, and subtractions), storing the intermediate results from earlier operations so they could be further manipulated by later ones.[14] More complex operations, such as square roots, logarithms, and trig functions, were handled in some early machines with special hardware and in others by specifying the appropriate sequence of elementary operations. Many of these machines were designed with

[13] While we know of no evidence that Colossus was ever called a "computer" during the 1940s, some British codebreakers were known as "Computor Clerks," or sometimes simply as "computors." According to Michael Smith, "This curious title had nothing to do with electronic computers… but was an echo of an old War Office covername for cryptanalysts – Signal Computor." Michael Smith, *The Secrets of Station X: How the Bletchley Park Codebreakers Helped Win the War* (Biteback Publishing, 2011). A 1940 memo noted that "The air ministry has already instituted the use of the word 'Computor' in all cases in lieu of 'cryptographer.'" (Illegible) to Denniston, July 14, 1940, HW 14/6, TNA. Another document noted that secrets sometimes had to be discussed over the telephone, and mandated substitutions which "bear no relation to the real nature of the intelligence." It specified "special computer" as a substitute for cryptographer, and "special computation" as a substitute for cryptography. "Draft," July 28, 1940, HW 14/6, TNA. This reminds us of other deliberately misleading wartime cover names, like the "tank" in World War 1, or the "tube alloys" codename for British atomic research. However, a subtly different pattern of usage occurred inside Bletchley Park itself, where such subterfuge was unnecessary. Later documents suggest that itself the term was most commonly used for workers carrying out more routine tasks, particularly those focused on counting and tallying operations. For example, within the Newmanry where Colossus was used, the term "computer" was applied not to the male cryptographers who oversaw the attack but to the "Wrens who enter, flag, and converge rectangles" (a laborious process of tabulating and adding data). Reeds, Diffie, and Field, *Breaking Teleprinter Ciphers, sections 31B & 71.* Thus the term "computer," while mandated for broad use outside Bletchley Park, may have sometimes been used within its walls in a sense paralleling the scientific use of human, usually female, "computers" discussed in David Alan Grier, *When Computers Were Human* (Princeton, NJ: Princeton University Press, 2006). Given the prevalence of scientists and mathematicians in senior roles within this part of Bletchley Park the adoption of this terminology is unsurprising.

[14] Stibitz, 1945 AMP report "Relay Computers." ML27-b3 in his papers at Dartmouth College. p.2

table making in mind and made it easy to compute a result for one set of parameters after another by constantly repeating the same processes.

Colossus, unlike these other machines, was not built to carry out numerical computations and we know of no evidence that it was used to carry out calculations, or could have been usefully applied to them.[15] Flowers did not see Colossus as a computer and was never particularly interested in computing, despite the popular misconception that he was keen to work with computers after the war but was somehow thwarted. His career trajectory confirms that electronic telephone exchanges, not computers, were his passion.[16] Flowers eventually left the Post Office when promised he could develop his exchange technology elsewhere – a conspicuous contrast with his decision to remain there after work stalled on the ACE computer he was supposed to be building for the National Physical Laboratory shortly after the war.[17]

Flowers himself was hesitant to call Colossus a computer. When, in 1977, he gave one of his first public talks about Colossus, after news of the machine had begun to reach the public, he related that “it is now said that during the Second World War I was responsible for the production of the world’s first electronic digital computer,” yet cautioned that “if so, that was an accident incidental to the solution of a problem.” [18] Flowers complained that telecommunications companies had focused on using general-purpose computers to control the switches connecting together telephone lines. He had spent his career urging that electronic “processors” be used to replace electromagnetic switches, rather than control them, so that the “structure” of the exchange would itself become electronic. Flowers noted that claims for Colossus as the first computer came “to the surprise of those concerned who thought of it as just another processor” or “a new-fangled processor,” seeing it as continuous with his work before and after the war on electronic telephone exchanges.

Perceptions of what people did and why it mattered change over time, even in their own minds. Flowers showed remarkable restraint in continuing to nuance his language decades after others won him recognition as an inventor of the computer. The closest he came to claiming Colossus as a computer seems to have been this 1983 passage: “Colossus had features now

[15] Jack Good, a veteran of Colossus practice at Bletchley Park, later claimed that, if appropriately configured, Colossus could almost have carried out a multiplication but that this would not have been possible in practice because of constraints on what could be accomplished in a processing cycle. (We haven’t been able to figure out what he had in mind, but then we aren’t as clever as him). This claim has been offered as proof of the flexibility of Colossus, which in a sense it does attest to: a device designed without any attention to numerical computations could almost, but not quite, have multiplied thanks to the flexibility with which logical conditions could be combined. Yet it also proves the very real differences between Colossus and devices designed for scientific computation. Multiplications were vital to computations, and a device that could not multiply would not, by the standard of the 1940s, be termed a “computer “or “calculator.” See Irving John (Jack) Good, "From Hut 8 to the Newmanry", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):204-222.

[16] Flowers’ career is examined Thomas Haigh, "Thomas Harold ("Tommy") Flowers: Designer of the Colossus Codebreaking Machines," *IEEE Annals of the History of Computing* 40, no. 1 (January-March 2018):72-78

[17] B Jack Copeland, ed., *Alan Turing's Automatic Computing Engine: The Master Codebreaker's Stuggle to Build the Modern Computer* (New York: Oxford University Press, 2005)

[18] Thomas H Flowers, *Electronic Computers and Telephone Exchanges (NPL Report DNACS 24/80)* (Teddington, UK: National Physical Laboratory, 1980)

associated with digital computers—semi-permanent and temporary data storage, arithmetic and logic units including branching logic, and variable programming—that may justify its being regarded as the first digital computer." In the rest of that paper Flowers consistently calls Colossus a "machine" rather than a "computer."[19]

Flowers stuck with this position to the end of his life, even as he heard and saw others claim Colossus as the first computer. Even his posthumously published chapter "Colossus" opened with the sentence "Machines such as counters, computers, and Colossus process information." This positioned Colossus as related to both counters and computers, but not as itself a computer. In the rest of the article he uses the word "machine" rather than "computer" when talking about Colossus.[20] In another posthumous publication, "D-Day at Bletchley Park," Flowers consistently used phrases such as "electronic machine" and "processor" to describe Colossus and attributed to others the idea that Colossus was a computer:

> [A]cademics interested in the history of computing have recognized that Colossus was the world's first electronic computer. It was not designed as a computer: computers had not yet been invented. It resembled a modern computer about as much as George Stephenson's Rocket locomotive of 1829 resembled the Royal Scot and other steam locomotives of the twentieth century. The basic technology used in a modern computer—data storage and retrieval, ultrafast processing, variable programming, the printing out of the results of the processing, and so forth—were [sic.] all anticipated by Colossus, some of it by as much as ten years.[21]

Even here Flowers remains reluctant to call Colossus a computer or himself the inventor of the computer, though he does, properly, claim credit for the development of many of the digital electronic techniques later used to build computers.[22]

## The Colossus Program

The term "program" can be treated either as an actors' category or an analysts' category. We know of no evidence that the word "program" was applied during the mid-1940s to any part of Colossus, or to anything it did. It was not an actor's category at Bletchley Park or Dollis Hill, though on the other side of the Atlantic the term was being adopted by members of the ENIAC group at the University of Pennsylvania. So we use program here as an analysts' category, imposing our own definition. However we attempt to do this in a way that is sympathetic to 1940s usage, not just with regard to automatic computers but also in other areas.

[19] Flowers, "The Design of Colossus", 252. His assertion that Colossus incorporated "branching logic" is questionable, and the intended meaning of "variable programming" is unclear.

[20] Thomas H Flowers, "Colossus", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):91-100.

[21] Thomas H Flowers, "D-Day at Bletchley Park", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):78-83.

[22] While we could find no article or talk by Flowers in which he directly calls Colossus a computer, Jack Copeland has made prominent use of quotations from unpublished interviews in which Flowers appears less ambivalent.

To begin with a conclusion: Colossus did execute a program. We can describe that program using a flow chart (Figure 4). Many years later, Harry Fensom, a senior member of the team that designed Colossus, reconstructed from memory the series of human and automatic actions it took to guide Colossus through a typical run. As he mentioned, “One panel of Colossus contained the so-called ‘master control.’ This acted as a program sequencer, guiding the run through all its steps, from switch-on, to print-out, and then on to the end of the run. Flowers designed the routine, or program, carried out by the master control, using a timing diagram and logic diagrams that had almost a modern flavour…”[23]

Before proceeding, we need so say a little bit about what Colossus did. It was designed to attack the teleprinter encryption produced by a German device, known as the Lorenz SZ40 by the Germans and codenamed “Tunny” by the British. This held twelve rotating code wheels, each studded with a different number of configurable pins. With each new character of the message some of the wheels turned to their next positions. The encryption process is shown in Figure 2.

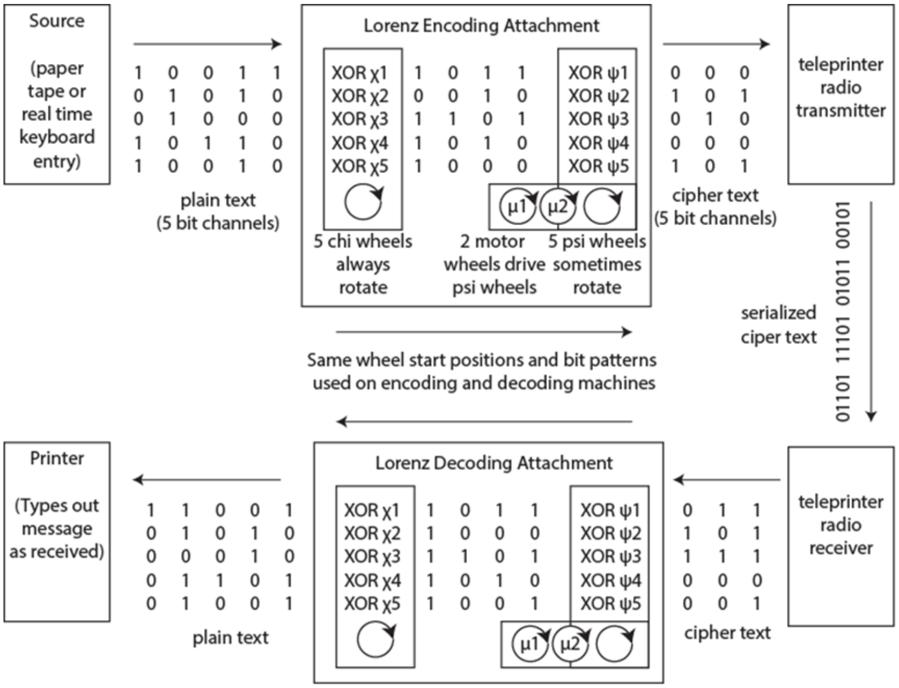

*Figure 2: Logical representation of the action of the Lorenz machine, dubbed “Tunny” by the British. The gap of four characters shown between the chi and psi wheels is to symbolize the idea of two logically independent transformations applied to a five channel bitstream, rather than a representation of the actual inner working of the machine.*

It was designed to dramatically speed one particular part of the code-breaking process, shown on Figure 3 below as “chi wheel setting.” This was the most time-consuming of all the tasks to accomplish with manual methods. Decrypting the message required knowledge of the bit patterns set on each wheel and the position to which each wheel should be turned at the

[23] Harry Fensom, "How Colossus Was Built and Operated -- One Of Its Engineers Reveals Its Secrets", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):297-306.

beginning of the message. Unencrypted text is made of words and, as any Scrabble player knows, the distribution of letters in natural language words is highly irregular. In a well encrypted bitstream, however, all codes are equally likely. A simple statistical test on decrypted text could easily identify the correct set of wheel start positions, assuming the pin settings were already known. But someone who sat down to try decrypting every possible combination of start positions for the twelve wheels would still be working when the war finished, and indeed when the earth was swallowed up by the sun. Colossus was designed to exploit a subtle flaw in the design of the Lorenz equipment used by the Germans. Because of this flaw, it was possible to obtain statistical evidence of correct wheel settings by looking only at correspondences between two of the chi wheels and two of the five bit channels that composed the intercepted message. This method, devised by mathematician Bill Tutte, involved comparing the deltas (changes) between successive bits generated by the code wheels and read from the intercepted message tape. Setting the first two chi wheels involved 1,271 trial decryptions of the message with all possible start positions. In the best case, two more runs (or in the case of the later versions of Colossus, three runs carried out in parallel) would identify the start positions of the other three wheels.[24]

Making thousands of complete trial decryptions for each message remained almost as impractical for cryptographers without electronic assistance. With Colossus, however, they could get the job done in less than an hour. The message was read again and again from a loop of paper tape rotated at up to five thousand characters a second, while the machine automatically tallied similarities between the deltas generated by the code wheels and read from the tape. Once the message tape completed a revolution it reset the counters, incremented the start position of one of the code wheels, and started again. To speed operation, Colossus used electronics to simulate the revolution of the code wheels, rather than the physical cogs of the real Lorenz machine.

[24] A variation on this description occurs somewhere in almost every one of the previous descriptions of Colossus we have already cited. Our own attempt at a somewhat more complete description came in Thomas Haigh, "Colossal Genius: Tutte, Flowers, and a Bad Imitation of Turing," *Communications of the ACM* 60, no. 1 (January 2017):29-35.

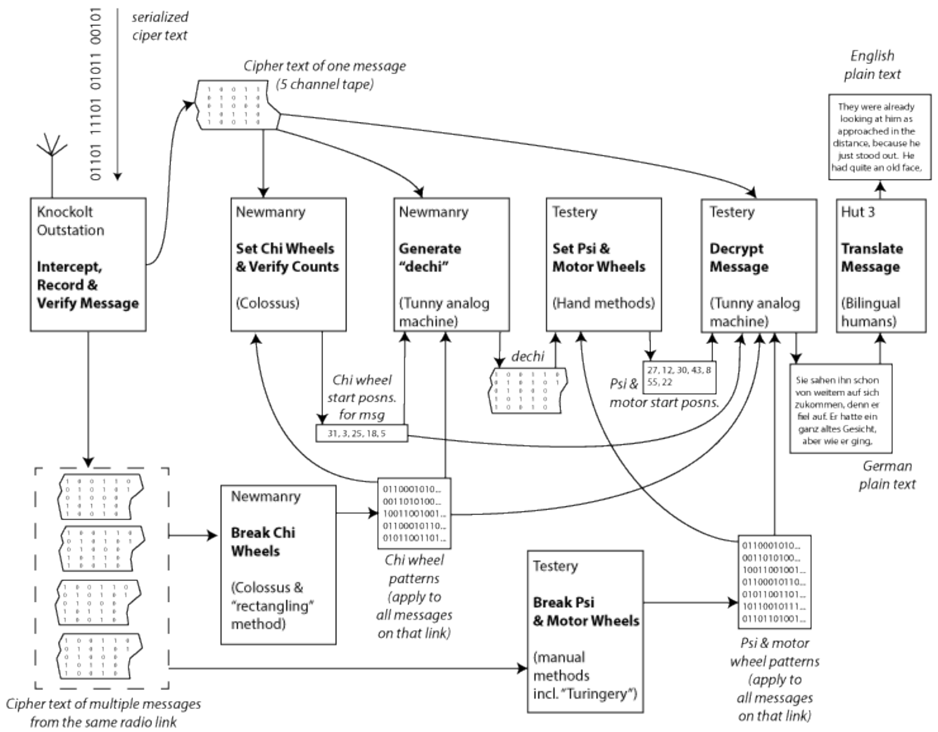
01101 11101 01011 00101
serialized ciper text
Cipher text of one message (5 channel tape)
English plain text
They were already looking at him as approached in the distance, because he just stood out. He had quite an old face.
Knockolt Outstation
Intercept, Record & Verify Message
Newmanry
Set Chi Wheels & Verify Counts
(Colossus)
Newmanry
Generate "dechi"
(Tunny analog machine)
Testery
Set Psi & Motor Wheels
(Hand methods)
Testery
Decrypt Message
(Tunny analog machine)
Hut 3
Translate Message
(Bilingual humans)
dechi
Chi wheel start posns. for msg
31, 3, 25, 18, 5
Psi & motor start posns.
27, 12, 30, 43, 8 55, 22
Sie sahen ihn schon von weitem auf sich zukommen, denn er fiel auf. Er hatte ein ganz altes Gesicht, aber wie er ging.
German plain text
Newmanry
Break Chi Wheels
(Colossus & "rectangling" method)
0110001010... 0011010100... 10011001001... 01100010110... 01011001101...
Chi wheel patterns (apply to all messages on that link)
Testery
Break Psi & Motor Wheels
(manual methods incl. "Turingery")
0110001010... 0011010100... 10011001001... 01100010110... 01011001101... 10110010111... 01101101001...
Psi & motor wheel patterns (apply to all messages on that link)
Cipher text of multiple messages from the same radio link

*Figure 3: The overall work flow needed to produce decrypted Tunny messages. This shows a typical configuration as for late-1944, in which Colossus was to set and verify the start position of chi wheels for each message and to initially "break" the same wheels by determining appropriate bit patterns. Techniques were identified to use Colossus machines to the psi wheels but because of the limited supply of machine time this was usually done manually.*

Colossus switched between operations based on the interaction of its control circuits with the contents of the message tape. A special code punched at the end of the message triggered control signals to reset its counters and, if a predefined threshold had been reached, to print the code wheel settings being evaluated and the counts obtained. Each message was followed by a blank sequence in the tape, which gave Colossus time to increment the uniselectors holding the code wheel settings currently being scored. These settings were then used to fix the positions of the electronic code wheels, so that when the tape spun round again to the special character that marked the beginning of the message, the machine was ready to evaluate the message against the next possible combination of wheel settings.

Fensom documented thirteen manual actions to get the machine ready – loading a message tape, configuring the plug board with the appropriate logical inputs, setting the wheel start positions, and so on. These were followed by twenty automatic steps, its program, such as resetting wheel positions, waiting for the message start signal, and comparing the counts to the thresholds selected by the operator. The sequence included inner and outer loops. The inner loop was followed each time a character was read from the message. The outer loop repeated each time the entire message had been read to reset the totals and increment the wheel start position.[25] Building on Fensom's description, modified by other sources and extended to include the additional capabilities of the second and subsequent Colossus machine, we prepared the flow chart below (figure 4) to visualize the program carried out by Colossus.

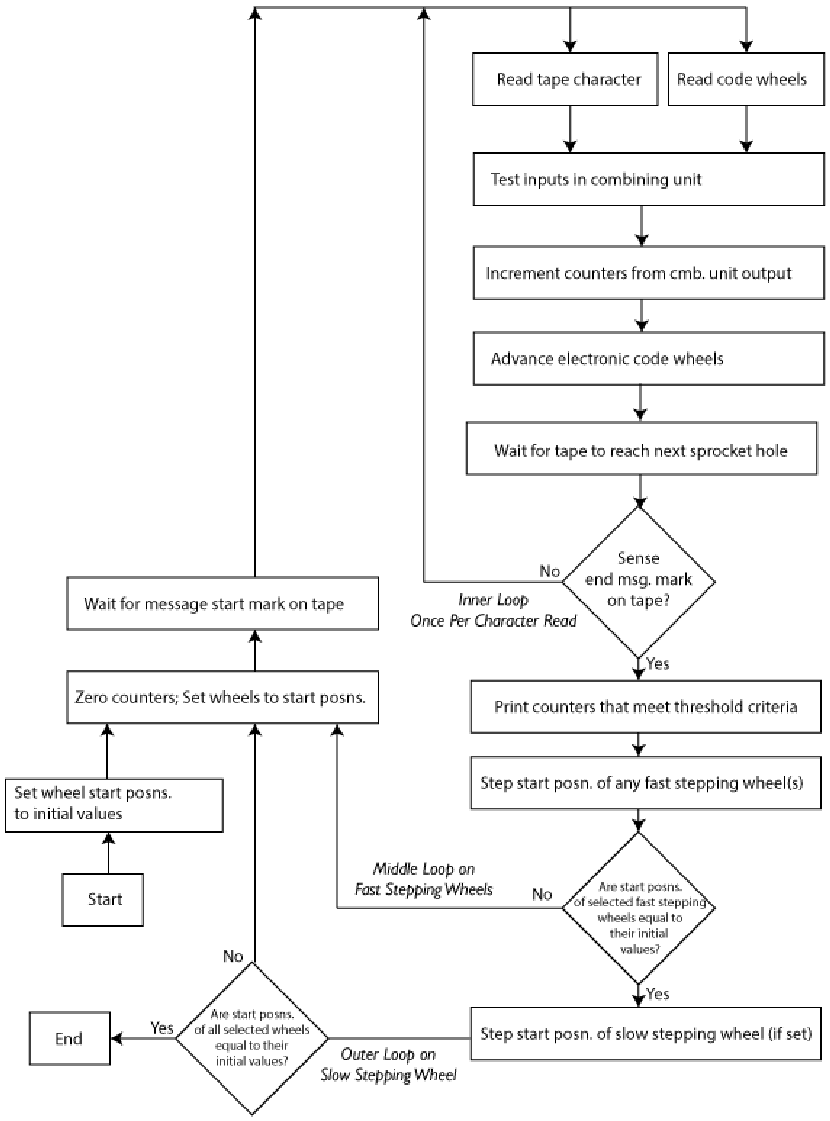
Read tape character
Read code wheels
Test inputs in combining unit
Increment counters from cmb. unit output
Advance electronic code wheels
Wait for tape to reach next sprocket hole
Sense end msg. mark on tape?
No
Yes
Inner Loop
Once Per Character Read
Wait for message start mark on tape
Zero counters; Set wheels to start posns.
Set wheel start posns. to initial values
Start
Print counters that meet threshold criteria
Step start posn. of any fast stepping wheel(s)
Are start posns. of selected fast stepping wheels equal to their initial values?
No
Yes
Middle Loop on
Fast Stepping Wheels
Step start posn. of slow stepping wheel (if set)
Are start posns. of all selected wheels equal to their initial values?
No
Yes
End
Outer Loop on
Slow Stepping Wheel

*Figure 4: The program of operations performed by Colossus.*

For the job we mentioned earlier, setting the first two chi wheels, Colossus would be configured to generate the difference between two consecutive message bits on two of the five channels of the tape and compare this to the signals coming from two simulated code wheels. In

[25] Fensom described the first version of Colossus. One of the features added in second and subsequent machines was the ability to set wheels to step either "fast" or "slow." Reflecting this, our diagram breaks his outer loop into middle and outer loops – once the fast stepping wheel (or wheels) had been tried in every possible start position (middle loop) Colossus would increment the start position of the slow stepping wheel.

that case, the inner loop would cycle each time the message tape finished, the middle loop would increment the start position for one of the code wheels being tested, and the outer loop would increment the position for the other code wheel. When both code wheels had returned to their start positions a light would illuminate to tell the operators that the job was finished.

### What Is a Program, Then?

How can we argue that Colossus was not a computer, but nevertheless carried out a program? In discussion of computing these things are often treated as inseparable: something is a computer because it can store and run a program, and a program is a series of instructions for a computer. We are using a more general definition of program: a series of discrete operations carried out over time. We should stress that Colossus did not store an encoded program in any single part of its apparatus, or read it from a medium. Rather, its machinery embodied a single, largely fixed program of operations. In this sense, a program is something enacted. Our definition fits not just computer programs but also other common uses current in the 1940s. It is also sympathetic to the original meaning of "program" in computing, which originated as a simple extension of its everyday meaning.

To begin with the more general meanings of program, current in the 1940s and today. A concert program, for example, specifies a sequence of musical works to be performed by an orchestra on a particular evening. A television network programmer is responsible for choosing and sequencing shows (also called "television programs") to produce a schedule. A washing machine fills with water, soaks, agitates, empties, rinses, and spins under the control of its "programmer." In each of these cases the program consists of actions to be performed in a particular order. We believe that the application of the word to computers began as a simple extension of this ordinary sense, and only later evolved into a distinct meaning of its own. This is a contrast to some previous work on the topic, which has made rather more convoluted arguments.[26]

The defining characteristic of the automatic computing machine was its ability to carry out one mathematical operation after another without human intervention, which fit naturally with the established idea of a program as a sequence of actions. The first groups attempting to build automatic computers did not use the term program to describe these sequences, though it would be entirely reasonable for a historian to apply the term as an analysts' category. Charles Babbage followed standard mathematical terminology in calling the discrete actions carried out by his planned engine Analytical Engine "operations." It would have been reasonable and consistent with contemporary English for Babbage to say that his machine would carry out a program of operations, though as far as we know he never used that phrase. Most of the first generation of automatic computers, built during the early 1940s, relied on paper tape to control their operation. The patterns punched onto the control tape of the Harvard Mark 1 computer, built by IBM for Howard Aiken's Computing Laboratory, were called "codes," and its staff

[26] David Alan Grier, "The ENIAC, the Verb "to program" and the Emergence of Digital Computers," *IEEE Annals of the History of Computing* 18, no. 1 (January 1996):51-55 David Alan Grier, "Programming and Planning," *IEEE Annals of the History of Computing* 33, no. 1 (Jan-Mar 2011):86-88. We share Grier's general sense that the idea of "programming" was introduced to computing in the ENIAC project, but are not convinced by his suggestions adoption of the term reflected a desire to "establish the economic importance of automatic computing" or that the alternative term "planning" reflected a specific commitment to Taylorist conceptions of production engineering.

talked about "coding" rather than programming. The word "sequence" was often used to describe the content of a particular strip of paper tape, usually a subroutine. The ability to automatically perform sequences of operations was central to the new machines, something recognized in the titles IBM gave to this machine (the Automatic Sequence Controlled Calculator) and to its more ambitious successor (the Selective Sequence Electronic Calculator). A phrase like "Program Controlled Calculator" would have also fit with contemporary usage.

The earliest application of the idea of a program to an automatic computer was to describe the control mechanisms of ENIAC. ENIAC used vacuum tubes rather than electromechanical relays for its arithmetic and memory circuits. These could switch thousands of times faster than the relays used in earlier computers. To exploit this speed its designers fully automated its control. Rather than read control sequences from tape, its operations were sequenced by a network of wires carrying "program pulses" between different parts of the machine. Their arrival of a pulse at a particular input terminal triggered whatever action had previously been set using the unit's control switches and knobs. These controls "programmed" the operations carried out by that unit's circuits. As well as discussing "program pulses," a June 1944 progress report described two ENIAC accumulator units as being "automatically programmed to receive the multiplier and multiplicand" when a program pulse triggered the multiplier unit to which they were attached. ENIAC's most complex unit, the "master programmer" controlled the overall computation, counting off loops and branching between different sequences when it was time to move on to a new task.

Interestingly, the terms "programming" and "program" were not originally used to describe ENIAC's closest analogs to their modern senses: the act of configuring the machine to carry out a particular problem and the resulting configuration of wires and switches. Instead these were called, respectively, "setting up" ENIAC and a "set-up." By late 1945, however, the ENIAC team was beginning to use "programming" in something much closer to its modern meaning. The new meaning of program seems to have been connected to the new approach to automatic control formulated for EDVAC, the follow-on to ENIAC commissioned in the summer of 1944. John von Neumann's celebrated "First Draft of a Report on the EDVAC," circulated within the ENIAC team in April 1945, combined the established approach of controlling a computation by reading a sequence of coded instructions with the novel idea of storing these instructions in a large, addressable memory using the same mechanisms employed to store and manipulate data.

Von Neumann himself followed the Harvard group by calling these instructions "code" and the process of producing them "coding," but others adapted the Moore School's existing vocabulary of "programming" to the new approach, altering its meaning in the process. A letter from one of the project's leaders noted that "the EDVAC will contain a large number of units capable of remembering programming instructions," to be copied from tape "before the actual program is started."[27] Similar terminology was soon applied to ENIAC: a report described the practices used in "planning a set-up for the ENIAC" as "programming techniques."[28] Rather than

[27] H. Goldstine to H. Curry, 3 October 1945, in the collection "ENIAC Patent Trial Collection" in the University of Pennsylvania archives.

[28] J Persper Eckert et al., *Description of the ENIAC and Comments on Electronic Digital Machines. AMP Report 171.2R. Distributed by the Applied Mathematics Panel, National Defense Research Committee, November 30* (Philadelphia, PA: Moore School of Electrical Engineering, 1945).

describe the action of the control circuits responsible for triggering operations at the correct time, the verb "to program" now referenced the work of the humans devising sequences of operations.

Since then the idea of a program in computing has continued to develop a specialized set of meanings, particularly in the discourse around the so-called "stored program concept."[29] Here we are using it in its more general sense. Even this broader sense is useful in distinguishing the capabilities of Colossus from those of other contemporary machines which did not sequence distinct operations over time. For example, in analog computers, such as differential analyzers, each part of the machine carried out the same operation throughout the course of the computation. There was no sense in which the machine executed a program of operations, or in which one part of the machine was "programming" another to stop what it was doing and to start something different.

## Defining Programmability

The history of computing community has generally observed a distinction between "general purpose" computers that can be reprogrammed by their users to do different kinds of tasks and "special purpose" computers executing a "fixed program." This distinction was already commonplace within the computing field by the 1950s. Early special purpose computers were built for tasks such as missile guidance or toll collection.

Colossus is, to our knowledge, unique in being widely characterized as both "special purpose" and "programmable," a startling formulation derived from Randell's early work. He called Colossus both "programmable" and "program controlled," apparently interchangeably, but also described it as "special-purpose." The legacy of this characterization endures, for example, on the Wikipedia page for Colossus which describes it as a "special-purpose electronic digital programmable computer." In his history of the relationship between computers and cryptography, Paul Gannon reshuffled the adjectives slightly: "Colossus can be defined as an, (sic.) electronic, binary/logic-processing, programmable, specific-purpose machine."[30] Yet to the best of our knowledge, neither Randell nor any other scholar has attempted to define specifically what "programmable" means as a historical term, or exactly what degree of configurability would qualify a device like Colossus as programmable but not as general purpose.[31]

[29] The "stored program concept" has recently been examined at length in Thomas Haigh, Mark Priestley, and Crispin Rope, "Reconsidering the Stored Program Concept," *IEEE Annals of the History of Computing* 36, no. 1 (Jan-Mar 2014):4-17, Haigh, Priestley, and Rope, *ENIAC In Action: Making and Remaking the Modern Computer* and Doron Swade, "Inventing the User: EDSAC in Context," *The Computer Journal* 54, no. 1 (2011):143-147.

[30] Gannon, *Colossus*, 435.

[31] Randell points out to us that he had, however, tried previously to rigorously define programs and programmability in his non-historical work with J.J. Horning on computational processes, in dialog with Dijkstra's contemporaneous work on structured programming. They describe a "general-purpose computer" as "a processor with the outstanding characteristic that program instructions, as well as program status, are represented by values of its (changeable) state variables." They then introduce "the term *programmable processor* to describe an interpreter with the property that some significant variables of the underlying process are instruction variables, and are not observable at the higher level." It is not obvious how to map the concepts of instructions or variables onto the capabilities of Colossus. A program is defined by reference to programmability: "Given a programmable processor, it becomes convenient to define a process by means of a program. A *program* for a particular processor is a set of initial values for its program status and instruction variables." J J Horning and Brian Randell, "Process Structuring," *ACM Computing Surveys* 5, no. 1 (1973):5-30.

We may gain clue from a parallel discussion underway in the mid-1970s. By the 1960s as anything described as a computer was understood to be programmable, and hence people usually just talked about computers rather than "programmable computers." The word "programmable" gained new currency in the 1970s following the introduction of powerful electronic calculators, where users could specify and store sequences of operations to be carried out automatically. Were these computers? The concept of a "programmable calculator" was introduced to describe a class of portable, personal machines that could be programmed by their users but that was more limited than true computers.[32] For example, according to a 1976 report "Calculators and the Computer Science Curriculum" cheaper calculators were not "programmable by the user" even though "they do contain stored programs and can execute these programs" for example when a user pushes the square root button. Thus "there seems to be a clear difference between these calculators and what most computer scientists commonly think of as computers." In contrast, "programmable calculators" had "sufficient memory to store a series of key strokes (that is, a sequence of machine language instructions) and then to execute the program." In a revealing nod to computer history the author continued, "at the higher price levels (but well under $1,000) such machines approach the ENIAC in capability, and will soon exceed it."[33] That was the same year in which Randell originally described Colossus as "programmable," and it seems reasonable to suppose that he had this discourse in mind when invoking the idea that a machine without the full capabilities of a general purpose computer could still be programmable. (At the time Randell was writing the documents that would have made clear whether Colossus actually had capabilities similar to a programmable calculator were still classified).

Having failed to find a useful and relevant definition of "programmable" in the existing literature, we will instead attempt to create our own. Our definition of a program as a series of operations carried out over time helps us separate the concepts of program and programmability. Whereas "program" has a long history in many different contexts, "programmable" appeared only after the spread of the electronic computer and can thus be applied to Colossus only as an analysts' category. *The Oxford English Dictionary* shows no usage prior to 1953, in which year it documents the appearance of two distinct but related meanings: "Capable of being scheduled in accordance with a programme of events" and "Of an apparatus, operation, etc.: capable of being programmed." Our definition of programmable is fairly similar. A program is a sequence of operations performed over time. Programming is the act of specifying those actions, and a device is programmable to the extent that its users can program it.

The simplicity of that definition conceals an important point: invoking the action of users means that a device will be programmable under some circumstances but not others. Any device that performs a program was designed and built by a group of people. Those same people could have chosen to design the machine to perform a different program, and with sufficient resources and motivation could presumably have taken it apart and rebuild it to do something different. For example, the claim of programmability is at the heart of the case traditionally made for the importance of Colossus to the history of computing, distinguishing it from the slightly earlier "Atanasoff Berry Computer." The ABC was likewise driven by the rotation of a physical

[32] This was more of a practical and marketing distinction than a theoretical one, as some programmable calculators had Turing complete programming languages.

[33] David Moursund, "Calculators and the Computer Science Curriculumn," *ACM SIGCSE Bulletin* 8, no. 2 (1976):21-23.

medium, in this case a rotating capacitor drum memory rather than the paper tape used with Colossus. It carried out a sequence of mathematical operations, performing some steps on a conditional basis depending on the results obtained. ABC and Colossus have traditionally both been granted "firsts" as follows: the ABC was a special purpose fixed program digital electronic computer whereas Colossus, built a few years later, was also an electronic digital computer but was in addition programmable.

The ABC is a famous example of a fixed program machine. Its design embodied a single program, which could not be fundamentally changed except by redesigning and rebuilding it. That had the advantage, for its intended users, that the steps needed to solve systems of linear algebraic equations were built into the machine and did not need to be specified. In contrast ENIAC, a general-purpose computer, had no built-in program. When used as originally intended, and documented in its instruction manual, users had to go through a long and elaborate process of deciding on the appropriate series of operations and figuring out the appropriate network of wires and switch settings to trigger those operations. Its program changed entirely from one job to another.

Later computers, modelled after the 1945 design for the EDVAC, essentially combined these approaches by adopting what we have called the "modern code paradigm."[34] On the most fundamental level they ran a single fixed program, like the ABC. But this program was, to borrow a term that became popular a little later, an interpreter. It fetched, decoded, and executed instructions held as data in an addressable memory. That meant that, like ENIAC, the computers could carry out whatever series of operations their users specified.[35] As we have described elsewhere, in 1948 ENIAC itself was reconfigured for this new programming paradigm, with a fixed program wired on its controls and user programs coded as numerical sequences on its function tables. Later systems added further levels of fixed programs underling the changeable program – microcode, operating systems, interpreters, BIOS code, and so on. But as long as their users can specify some new programs we would consider those aspects of the system programmable.

A device that is programmable by one users with one tools, will not necessarily be programmable by other users with other tools. For example, when the iPhone was first released its operating system was deliberately locked down to prevent the creation or installation of any programs other than those Apple had created for it. For users outside Apple's own development teams it was not programmable. Some users wrote new code to replace parts of the standard operating system to "jailbrake" it, allowing the installation of other programs. Soon Apple itself created its "App Store" and tools to allow users outside the company to create their own programs for the iPhone, which quickly became a crucial feature. Most users still do not create or modify its programs, but the machine is programmable for those with the skills, motivation, and tools to do so.

We should also distinguish between creating programs and choosing between pre-existing programs. For example, by our definition a player piano follows a program when it

[34] Haigh, Priestley, and Rope, "Reconsidering the Stored Program Concept".

[35] At this level of abstraction, one might depict Turing's "universal machine" as doing something similar, as it could carry out the same work as any other machine without requiring modification to its control table if provided with the appropriate sequence of symbols on tape.

plays notes from a music roll. If its user had a machine able to punch notes onto a blank roll it would be programmable. Most users did not. They could select between programs by changing rolls, but not alter them. Likewise, Atari's 1970s home videogame console, the VCS, was a simple unit that ran code from ROM cartridges. Ordinary users purchased, selected, and ran programs but did not create them. For someone with the appropriate development system and the ability to burn programs to ROM the VCS was programmable.

Other systems provided the ability to choose between built-in programs, but unlike the player piano or the Atari VCS were not designed to let users swap in additional programs. An example is the washing machine. As we mentioned above, an automatic washing machine incorporates a "programming unit" to control the sequence of operations it carries out. Washing machine users choose between several different programs by turning a control dial (figure 5) to the desired starting point, and can also push buttons to set parameters, such as "light load," that modify the washing operations. Yet it would seem very unusual to talk about "programming" a washing machine.[36] The conventional term is that a user "selects" a program cycle. Likewise, in the computing context one would not usually call the act of choosing a predefined program and triggering its execution "programming."[37]

[36] Google currently finds 94 uses of the exact phrase "programmable washing machine" which suggests that it is rarely used.

[37] Such a usage might resonate with the original 1944 ENIAC sense of programming, in which directing the operation of a piece of machine was programming it, but the concept of "programmability" appeared only after our modern sense of a computer program was well established. Thus the idea of "programmability" has historically been applied only to a specific, narrower and later sense of "program" and, as the *OED* suggests, means that a user can set up a schedule of events.

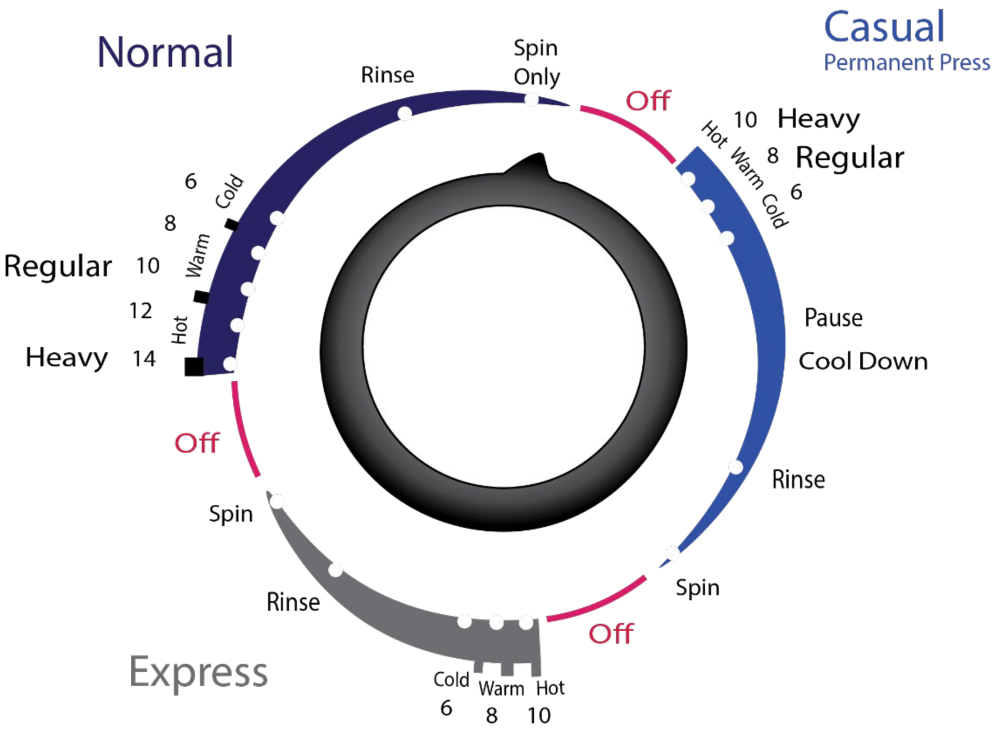

*Figure 5: On this typical washing machine dial, users select one of three predefined programs, such as the normal program, which runs as follows: hot, warm, cold, rinse, spin, off. Users can also skip to a particular point within each program, for example starting the normal wash program at the beginning for heavily soiled clothes or omitting the hot wash operation for regular soilage.*

We conclude that the concept of "programmability" as applied to a device requires not only that the device carries out a sequence of distinct operations over time, i.e. that it follows a program, but also that it allows a given user to define new sequences of operations. We see programmability as a relational rather than absolute property. One user with one tool will be able to program aspects of the machine's behavior that other users, with other tools, cannot program. So in asking whether Colossus was programmable, we must be a little more specific. We suggest that the definition of programmability should be inclusive, to define a machine as programmable if *any* actual group of users with any set of existing tools could program its overall behavior, excluding only the special case of engineers rebuilding it. If the cryptanalysts and operators at Bletchley Park were able to set up Colossus to carry out new sequences of operations then Colossus was programmable. If changing the program carried out by Colossus (and diagrammed by us as figure 4 above) would have required calling Flowers and his team to take it apart then Colossus was not programmable. As in the case of the washing machine and the ABC, allowing users to select between pre-defined sequences of actions, skip steps, or apply parameters to alter the behavior of particular steps does not constitute programmability. On the other hand, because we have a broad sense of program our definition of programmability is not encumbered with requirements for Turing completeness, conditional branching, and so on. By our definition a player piano is programmable by someone with a tape punching machine, and so were relay computers like the Harvard Mark I.

### Parts and Controls of Colossus

To characterize Colossus as "programmable", then, is to suggest not only that it followed a program but also that users could fundamentally change the program of operations it performed without rebuilding its hardware. Colossus was indeed highly configurable, so that its users applied it to many different tasks – a tribute to the foresight of Flowers' team and its close liaison with Newman and his group at Bletchley Park. But did these configuration capabilities change the program itself, or just set parameters for a fixed program? To answer those questions, we must retreat temporarily from historiography to materiality, to talk a little bit about what Colossus did and the extent to which its program could be changed by expert users.

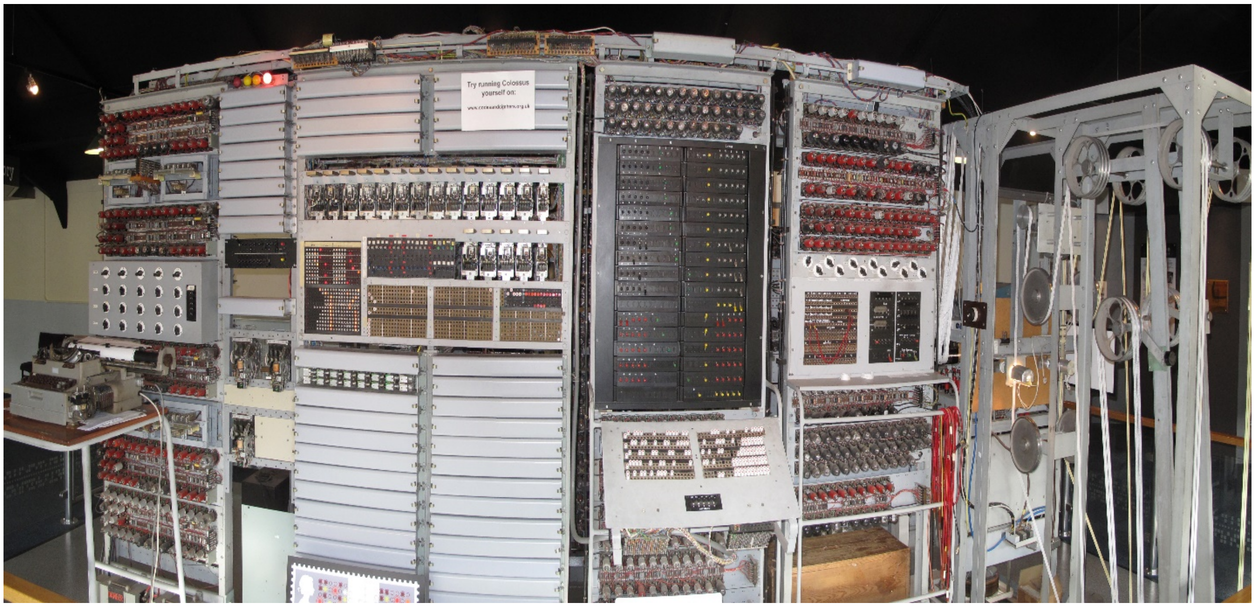

*Figure 6: The reconstructed Colossus 2 at the National Museum of Computing. By TedColes - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=38648141.*

Thanks to the declassification of archival records and the work done by Tony Sale and his team in rebuilding a Colossus at the UK's National Museum of Computing (figure 6) we are now able to document exactly what it could and could not do. The overall structure of the machines and details of electronics and circuitry are not well described in the surviving primary sources available to us, and here we have relied on the work of rebuild team. Their depictions in the earlier secondary literature are not always accurate. The detailed functionality and intended use of most of the machines' controls, however, are described in the 1945 "General Report on Tunny" particularly in Chapter 53.[38] We have also made use of the substantial number of printouts from Colossus runs reproduced in that report and others dating from 1945; these have enabled us to confirm many details of the machines' functionality independently of the reconstruction. In combination, these sources allow a more complete and accurate description of the functionality and use of the machines than might be expected given the secrecy that

[38] This report was recently published as Reeds, Diffie, and Field, *Breaking Teleprinter Ciphers*.

surrounded them. We present a summary of these findings here, but readers who want more detail on the analysis and behind them should read our full technical report.[39]

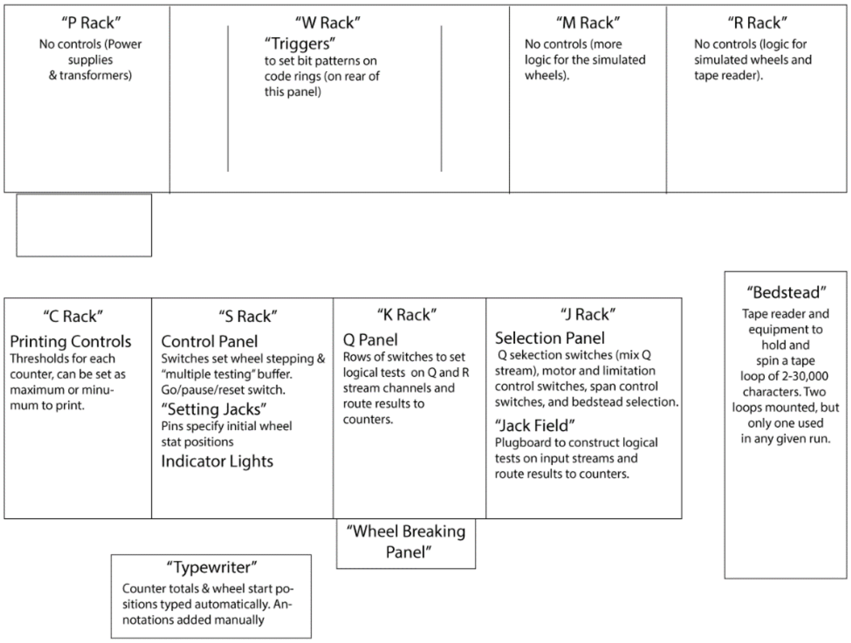
"P Rack"
No controls (Power supplies & transformers)
"W Rack"
"Triggers"
to set bit patterns on code rings (on rear of this panel)
"M Rack"
No controls (more logic for the simulated wheels).
"R Rack"
No controls (logic for simulated wheels and tape reader).
"C Rack"
Printing Controls
Thresholds for each counter, can be set as maximum or minu-mum to print.
"S Rack"
Control Panel
Switches set wheel stepping & "multiple testing" buffer. Go/pause/reset switch.
"Setting Jacks"
Pins specify initial wheel stat positions
Indicator Lights
"K Rack"
Q Panel
Rows of switches to set logical tests on Q and R stream channels and route results to counters.
"J Rack"
Selection Panel
Q sekection switches (mix Q stream), motor and limitation control switches, span control switches, and bedstead selection.
"Jack Field"
Plugboard to construct logical tests on input streams and route results to counters.
"Bedstead"
Tape reader and equipment to hold and spin a tape loop of 2-30,000 characters. Two loops mounted, but only one used in any given run.
"Wheel Breaking Panel"
"Typewriter"
Counter totals & wheel start po-sitions typed automatically. An-notations added manually

*Figure 7: Physical layout of Colossus, showing its main user-configurable controls.*

Many of the racks (figure 7) combined several distinct functions, so we find it useful to conceptualize Colossus as being composed of three main subunits (figure 8). Its forerunner, Heath Robinson, was literally designed and manufactured in three separate parts, interconnected for the first time at Bletchley Park. These were the tape unit, the combining unit, and the counter. Colossus followed the same pattern, though the subunits were less physically distinct.

[39] The report has the working title "Colossus Configuration and Capabilities" [update: now "Colossus: The Missing Manual"] and is expected to appear in the Media of Cooperation working paper series of Siegen University during 2019.

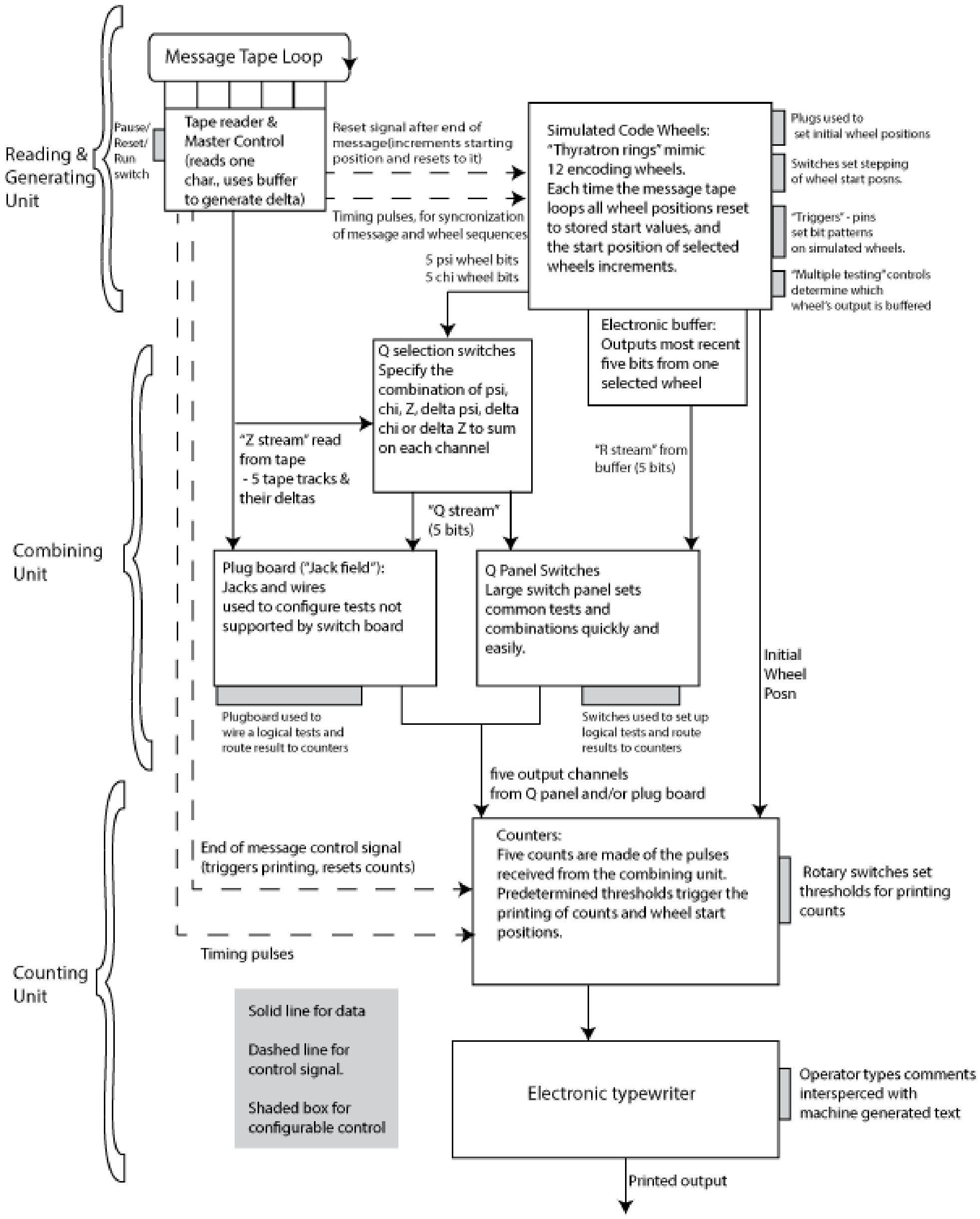
Reading & Generating Unit
Message Tape Loop
Pause/ Reset/ Run switch
Tape reader & Master Control (reads one char., uses buffer to generate delta)
Reset signal after end of message(increments starting position and resets to it)
Timing pulses, for syncronization of message and wheel sequences
5 psi wheel bits
5 chi wheel bits
Simulated Code Wheels: "Thyratron rings" mimic 12 encoding wheels. Each time the message tape loops all wheel positions reset to stored start values, and the start position of selected wheels increments.
Plugs used to set initial wheel positions
Switches set stepping of wheel start posns.
"Triggers" - pins set bit patterns on simulated wheels.
"Multiple testing" controls determine which wheel's output is buffered
Electronic buffer: Outputs most recent five bits from one selected wheel
Combining Unit
Q selection switches Specify the combination of psi, chi, Z, delta psi, delta chi or delta Z to sum on each channel
"Z stream" read from tape - 5 tape tracks & their deltas
"Q stream" (5 bits)
"R stream" from buffer (5 bits)
Plug board ("Jack field"): Jacks and wires used to configure tests not supported by switch board
Q Panel Switches Large switch panel sets common tests and combinations quickly and easily.
Initial Wheel Posn
Plugboard used to wire a logical tests and route result to counters
Switches used to set up logical tests and route results to counters
five output channels from Q panel and/or plug board
Counting Unit
End of message control signal (triggers printing, resets counts)
Timing pulses
Counters: Five counts are made of the pulses received from the combining unit. Predetermined thresholds trigger the printing of counts and wheel start positions.
Rotary switches set thresholds for printing counts
Solid line for data
Dashed line for control signal.
Shaded box for configurable control
Electronic typewriter
Operator types comments interspersed with machine generated text
Printed output

*Figure 8: Logical architecture of Colossus, broken down into its three main function units and showing data and control flows between them as well as the main configurable settings available to control them.*

The reading and generating unit consisted of a tape drive (often used to mount encrypted text) and a set of twelve electronic ring counters that simulated physical code wheels in the Lorenz cipher machines used by the Germans. Physically, this functional unit included "bedstead" tape equipment and reader, and lots of electronics for the simulated wheels in racks W, M, and R at the back of the machine. For convenience, most of the controls for these circuits were mounted on the front of Colossus on the S rack and included:

- "Setting jacks" to set the initial wheel start positions;
- Stepping controls, to govern the stepping of these initial positions each time the message was restarted;

- Pins to set the bit patterns on the simulated wheels (known as "triggers");
- "Multiple testing" controls to determine which code wheel's output was buffered so that up to five consecutive wheel positions could be evaluated simultaneously; and
- Span controls, used to count only part of the input tape when producing totals.

The combining unit combined selected channels from the various tape and wheel bitstreams. The results were further combined by using them as inputs to use-configurable networks of logic gates. The equipment for these functions centered on a plugboard in Rack J and a large switch panel in Rack Q (the "Q Panel"), both used to define the logical tests applied to inputs and specify the counters to which the results should be sent. User-configurable controls included:

- "Q selection switches" to determine which of the many possible inputs were routed onwards to the Q panel and whether to supply it with raw values or "deltas" between successive bits;
- Wires set on the plug board ("jack field") to run selected inputs through particular combinations of logic gates and route them to particular counters; and
- Keys set on the Q panel to run selected inputs through particular combinations of logic gates.

The counting unit included the electronically controlled typewriter used for printing and, in the C Rack, the electronics that drove it and buffered output so that Colossus could continue to work while the results of the previous set of counts were output. Only one aspect of this was configurable:

- Rotary switches to set the threshold above which the contents of a counter, and corresponding initial wheel positions, should be printed.

### What Could the Controls Configure

These various controls did give Colossus considerable flexibility. It was designed with wheel setting in mind but codebreakers found many other ways to use Colossus for different parts of the coding job. For example, one way to determine whether decryption settings were correct was to count the number of times each letter appeared in a message. Colossus could be set to look only at the characters on the tape (ignoring the simulated cipher wheels entirely) and to count the frequency with which they appeared. A run of that kind would finish after a single pass through the inner loop in our flow diagram, having immediately met the termination conditions for the middle and outer loops. In our technical report we discuss several common Colossus configurations and exactly how the machine would be set up for each.

Those describing Colossus as programmable have generally rested their case on the combining unit controls. For example, according to Benjamin Wells "tightly refined codebreaking algorithms were implemented in plug-wiring and switches. But the crucial story is that the same machine supported many different algorithms via flexible programming."[40]

[40] Benjamin Wells, "The PC-User's Guide to Colossus", in *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, ed. Jack Copeland (New York: Oxford University Press, 2006):116-140.

We will therefore describe these controls in a little more detail. Plugboard cables and, on later models, switches were used to select inputs and run them through logic gates to generate pulses for the counters. Users had a great deal of flexibility in configuring its circuits to run inputs from the message tape and electronic code wheels though different logic gates to combine them in different ways for different purposes. Signals from the electronic simulated code wheels and the message tape were available as inputs. The message tape appeared as five separate binary channels, as did each of the two main sets of code wheels. The Colossus machines also provided inputs representing the differences between two consecutive character positions and, in later models, five consecutive positions of one of the electronic code wheels (the "R stream" on our diagram). The results of these logical combinations could be fed into different electronic counters, the contents of which would be printed or not printed after the message was fully read according to thresholds set by the user. After each reading of the message tape the initial code wheel settings would be stepped to a different combination according to switches configured by the user.

Operators plugged wires or set switches to combine these inputs. This let them specify up to five truth tables, making particular logical connections between the input pulses and the output fed to each counter. Colossus included circuits to implement a variety of Boolean logic operations, including XOR and NOT. By running pulses through several of these circuits other logical conditions could be specified.[41]

The network of logic gates configured on Colossus's combining unit by its user had the effect of implementing up to five logical truth tables, each table specifying which combinations of inputs should trigger an output pulse for a counter. The logic circuits then reset, and processed the next combination of inputs. This was analogous to the function of a traditional tabulating machine, in that both processed a stream of records (characters from paper tape for Colossus, punched cards for the tabulating machine) and incremented the appropriate counter whenever the input data satisfied a specified condition. Colossus allowed for rather more complex logical conditions than traditional tabulating machines, but it similarly transformed and counted streams of input data. The only conditional action that Colossus users could specify via the logic circuits of the combining unit was to either increment or not increment each counter in response to a particular combination of input received from the tape and code wheels. Data pulses output from the combining unit were routed only to counters, and the contents of the counters went only to the printer.

Computer scientists use automata theory to distinguish between the fundamental capabilities of different kinds of automatic devices. The most advanced, including programmable computers from ENIAC onward, are equated with Turing Machines. Push down automata are less powerful than Turing machines, and finite state automata are less powerful than push down automata. As their name suggests, even finite state automata preserve state information from each time step to the next. In contrast, the "combining unit" of Colossus did not preserve state information. The technical term for this kind of system is "combinational logic"[42] (which is

[41] Reeds, Diffie, and Field, *Breaking Teleprinter Ciphers*, 323.

[42] Gerard A Maley and John Earle, *The Logic Design of Transistor Digital Computers* (Englewood Cliffs, NJ: Prentice Hall, 1963).

echoed in the term “combining unit” applied by Flowers and his team to the corresponding part of Heath Robinson) or “time-independent logic.”[43]

The logic performed one complex step repeatedly, but did not sequence operations or maintain any state information from one input character to the next. Counter totals were retained from one character of the tape to the next, and the uniselectors held current wheel start positions from one entire cycle of the tape to the next. In an abstract sense both held state information, and thus constituted a kind of memory, but this data could not be fed into the combining unit as inputs. Counter values were not available as inputs for the configurable logic in the combining unit, eliminating the possibility of using counters as control flags. We do not believe that the ability to reconfigure combinational logic to implement an arbitrary truth table is evidence of “programmability” because it does not truly change the sequence of operations performed by the machine. It merely provides a logic expression evaluated in one of those steps.

It has sometimes been claimed that Colossus possessed the capability for conditional branching. Colossus certainly included circuits that would trigger an action only under particular conditions – for example resetting the machine when the end of message code was read from the tape but not when ordinary data was read. During the reset process at the end of each loop of the message tape, dedicated circuits would compare the counter values against a user selected threshold to determine whether the results of that cycle should be printed. If that was “conditional branching” then so is any control circuit, even an adder that carries digits from one position to the next, or a thermostat that turns on a heater once the temperature falls below a user-specified level.

Colossus was not programmable, according to our definition, because the basic sequence of operations performed, the program, could not be changed by the users. That judgement involves answering a difficult question: what kinds of modification to the program executed by a machine are sufficient to make that machine “programmable” as opposed to “configurable”? What is a new program and what is just a parameter? We have shown that the Colossus controls could not alter the basic sequence of operations shown in our flow diagram, though with certain switch settings some steps would be skipped. To us, that kind of configuration seems more like setting a parameter than defining a new program.

That actual programming capabilities of Colossus turn out to be much more limited than those claimed by its boosters. We stridently dispute the suggestion of S. Barry Cooper that Colossus “may well be Turing complete” (i.e. could handle any problem solvable by any computer if given sufficient time and storage space).[44] Wells’ claim that different “algorithms” could be set up is similarly misleading. An algorithm expresses a computational procedure as a step-by-step series of operations carried out over time. Colossus could carry out only one algorithm. Fensom characterized it as carrying a program designed by Flowers, rather than running many different programs written by its users.[45] We think he was right.

[43] C J Savant, Martin S Roden, and Gordon Lee Carpenter, *Electronic Design: Circuits and Systems* (Benjamin/Cummings, 1991).

[44] S Barry Cooper, "The Machine as Data: A Computational View of Emergence and Definability," *Synthese* 192, no. 7 (July 2015):1955–1988.

[45] Flowers himself, in 1983, described Colossus as following a “master control program” which was fixed by the control unit rather than defined by the operator. However, he then muddied this distinction by calling

### Was Colossus Binary?

Having described the architecture of Colossus we can clarify a final point. Colossus is often described in the secondary literature as a binary machine, sometimes with the implication that this made it closer to modern computers, and hence more advanced, than computers using decimal number representations. Paul Gannon made this explicit: because the "modern computer can be defined as an electronic, binary/logic-processing, conditional-programming, stored-program-control, general-purpose machine" and Colossus, unlike ENIAC, was also a "binary/logic-processing, programmable" machine "Colossus was closer to the modern concept of the computer than ENIAC was in some significant ways."[46] Echoing this, in *The Innovators*, Walter Isaacson noted that "well before ENIAC… the British code breakers had built a fully electronic and digital (indeed binary) computer." ENIAC, in contrast, "was not like a modern computer" to the extent that it used a decimal system.[47]

In popular discourse of this kind, binary is often associated with the idea of representing information as a series of electrical pulses, conceptualized as 0s and 1s. Colossus certainly did that – its users spoke of "impulses," which they represented with the "dots" and "crosses" terminology of teleprinter engineers rather than the 1 and 0 of abstract logic. But all digital electronic machines transmit pulses, whether they use binary or not. ENIAC, for example, transmitted the decimal digit four as four pulses.

"Decimal" and "binary" properly refer to the number bases used by computers when representing and manipulating numbers: 10 for decimal and 2 for binary. The difference between binary and decimal computers came in three main areas: the format used to store numbers in memory, the format used to transmit them, and the design of the arithmetic circuits used to manipulate them. It is hard to define Colossus as either binary or decimal, because most of the electronic pulses manipulated by Colossus did not represent numbers. It had no hardware to interpret input patterns as binary numbers, or to carry out any arithmetic other than incrementing a counter. A computer reading a five-channel paper tape would interpret the five channels as a five-digit binary number, assigning a different value to a 1 depending on its position. For example, 10011 might be interpreted by its hardware as the number 19. In contrast, Colossus treated the five channels as synchronized but independent bitstreams. It performed logical transformations, rather than arithmetic, on the data it read.

Three parts of Colossus counted, each in a different way but none used a binary representation. 1) The units that emulated the encoding wheels of the Lorenz machines were a set of electronic ring counters of different lengths (and thus neither binary nor decimal). 2) The electronic counters tallied pulses output from the combining unit. These used decimal arithmetic, but stored each digit as a bit pattern using a biquinary representation. 3) The electromechanical

Colossus configurations set up with jacks and switches "programs" and the person configuring Colossus a "cryptanalyst-programmer."

[46] Gannon, *Colossus*, circa 435.

[47] Walter Isaacson, *The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution* (New York: Simon and Schuster, 2014), 75 & 79. This connection of binary arithmetic with progress is rather dubious. Decimal arithmetic has some advantages for floating point and remained a feature of many later computer architectures.

uniselectors that held the current start positions for the code wheels were decimal counters, advancing with each cycle of the message tape.[48]

### Conclusions: Colossus and the Early Digital

Computers were far more visible than other forms of digital electronics during the 1970s and 1980s, and inventing the computer a much more visible honor than inventing the digital signal processor. For decades, from the late-1940s to the 1990s, specialists and the public shared a fairly clear sense of what "the computer" was. It consisted of a box holding a processor and memory, to which were connected storage devices such as disks and tapes, and input and output devices such as keyboards and printers. Over time the boxes shrank and got cheaper, and they spread from data processing centers into schools, offices, and homes. Making a case for the historical importance of Colossus meant shoehorning it into this tradition, as Brian Randell did when he won it a prominent place at the seminal 1976 Los Alamos meeting where the early agenda for the history of computing field was set.

Early computing innovation is still often discussed from the viewpoint of logic, stressing mathematical and theoretical insights behind the development of computer architecture.[49] The progression from one early computer to the next has been depicted as a series of abstract architectural innovations to be checked off and annotated with dates and machines – first conditional branch, first stored program, and so on. In turn, these innovations have sometimes been represented as mere practical instantiations of the work of Alan Turing on the mathematical logic of computation.[50]

This obsession with firsts has limited our understanding of all early machines, by reducing each to a date and a couple of approved adjectives.[51] Even ENIAC, with a long and productive life, has been remembered only as a single historical point in 1945. During this research we realized that the tradition has imposed even more serious limitations on general understanding of Colossus. Within it, the implicit reason to care about Colossus has been that it was the first programmable electronic computer. We don't believe it was either of those things. Colossus certainly followed a program, in that it sequenced different operations over time. But contrary to previous claims, it was not programmable, because this sequence could not be changed in any fundamental way. We think that Flowers' own, carefully nuanced, characterization of Colossus as an "electronic processor" rather than a "computer" is essentially correct. Colossus did not have any capabilities for numerical operations. This sets it aside from every other machine celebrated in the history of computing.

[48] Biquinary representation, as used in the Colossus counters, took advantage of vacuum tube circuits that had not just two stable positions but five. Using a two-position counter and a five-position counter for each decimal digit allowed for ten possible combinations,

[49] The classic contribution of this kind is Martin Davis, *Engines of Logic: Mathematicians and the Origin of the Computer* (New York, NY: Norton, 2001).

[50] This assumption is critiqued in Thomas Haigh, "Actually, Turing Did Not Invent the Computer," *Communications of the ACM* 57, no. 1 (January 2014):36-41.

[51] Michael R Williams, "A Preview of Things to Come: Some Remarks on the First Generation of Computers", in *The First Computers: History and Architectures*, ed. Raúl Rojas and Ulf Hashagen (Cambridge, MA: MIT Press, 2000):1-16

As Flowers proudly noted, Colossus certainly had many elements in common with early computers. But its architecture and purpose were fundamentally different. In his public statements on Colossus, Flowers situated Colossus within the context of early work on digital communication. The recent proliferation of digital devices and the vanishing of "the computer" as a distinct and coherent thing makes us better able to appreciate Colossus, as Flowers did himself, more as a contribution to the development of digital communication than to computing. Colossus was a digital electronic device able to generate bitstreams electronically, combine those bitstreams with others read from paper tape, apply logical transformations, and count the results.

Claims for the historical importance of Colossus can and should be about more than the exact string of adjectives to insert between "first" and "computer." This challenges us to articulate its importance in more productive terms, for example by its achievements in use. From the traditional historiographic perspective, the only thing separating Colossus from the Atanasoff Berry Computer is the insertion of the word "programmable" between "first" and "electronic computer." That distinction doesn't hold up: both machines could be configured by setting parameters, but not by defining new sequences of operations. As with Colossus, the basic sequence of operations the ABC performed was fixed, in this case to solve systems of linear equations, but its behavior would change based on the parameters supplied when the machine was configured by setting switches and rewiring plug boards.[52]

Looking at the use of the two machines, rather than their claims to "firstness," is a much more revealing way of separating them. The ABC incorporated many novel features, but its homebrewed input/output system (using sparks to burn paper) never worked reliably enough for it to tackle the problems it had been built to solve. (It managed some useful work on much simpler statistical problems, but nothing that would justify the creation of such an elaborate machine). Colossus, in contrast, proved spectacularly effective when applied as intended. It facilitated the reading by Allied commanders of some of Nazi Germany's highest-level military communications, including messages written by Hitler himself. It has been credited with a crucial role in revealing German troop positions and plans in Normandy before and after the D-day landings. It's often claimed that breaking Enigma, or sometimes even the personal contributions of Alan Turing, shortened the war by two years. [53] We find that unconvincing, as did Max Hastings and John Keegan in overview histories of the contributions of intelligence

[52] Alice R Burks and Arthur W Burks, *The First Electronic Computer: The Atanasoff Story* (Ann Arbor, MI: University of Michigan Press, 1989).

[53] The claim is often made that the Second World War was shortened by two years either by (a) Alan Turing's personal contributions to codebreaking, (b) breaking Enigma, or (c) Colossus. The original source of the idea seems to be the introduction to E F Hinsley and Alan Stripp, eds., *Code Breakers: The Inside Story of Bletchley Park* (New York: Oxford University Press, 1993) where it is made for Ultra intelligence, i.e. the Bletchley Park operation as a whole. Specifically, the authors suggest that without the benefit of Ultra, the battle of the Atlantic and the campaign in North Africa would both have taken significantly longer to win, with the result that the D-Day landings could not have been attempted in 1944. They also believe that the landings would then have failed or been impossible in 1945 without Ultra intelligence, in part because of attacks by V weapons and new weapons that in our timeline Germany was not able to bring into operation. Thus D-Day would not take place until 1946, lengthening the war by two years. This rather Britain-centric view seems to us to neglect the overall trend of the conflict. In particular, the rapid progress of Soviet troops westward during the first half of 1944 suggests that the Red Army would have reached Berlin well before mid-1947 with or without the D-Day landings. We also note that the atomic weapons used in Japan in 1945 could have been deployed against Germany if the war had lasted even a few months longer.

work.[54] Still, if access to Tunny intercepts shortened the war by even a month, that would make Colossus one of the best investments in history. The ABC spent the war abandoned in a basement.

Flowers's partnership with the codebreakers at Bletchley Park predated his work on Colossus, and he had enough sense of the operational context to optimize the usefulness and versatility of his machines. Not only did he lead a project that quickly built a reliable machine from exotic parts, but he also built exactly the machine that was needed by Britain's codebreakers to deliver intelligence to Allied leaders. In doing so it made an appreciable contribution to the course of world history. Less than a year went by between the request from Newman for Dollis Hill to work on a machine to tackle Tunny and the delivery of the first Colossus to Bletchley Park, and for much of that time Flowers was focused on other machines.

Colossus also stood out for its reliability in comparison other similarly complex electronic digital machines of the 1940s and early 1950s. All the others all seem to have spent a year or so between being finished and being reliable enough to carry out useful work. For example, after being moved to Aberdeen Proving Ground at the start of 1947 ENIAC relapsed into unreliability, and for more than a year struggled to carry out any useful work. As Colin Burke's recently declassified work has shown, Colossus was one of many wartime designs for complex devices, sometimes called "rapid analytical machines" intended to speed codebreaking. No other machine of comparable technical ambition was ready in time to assist with the war, and many of them never worked reliably enough to be useful.[55] Colossus, in contrast, was handling a full load of production work by March 1944, less than two months after being delivered to Bletchley Park.[56]

This reminds us of the enormous importance of professional engineering work in distinguishing the relatively small number of successful projects from the greater number of failures. Flowers and his colleagues were part of a telecommunications engineering institution. They could draw on experience, internal engineering talent, and existing relationships with component suppliers. Colossus made extensive use of electronics for counters and logic. Flowers insisted that these technologies were not unproven to him, even if the rest of the world was skeptical, because of the pre-war work he had been doing on electronic telephone switching.[57]

Colossus challenges us to look beyond the traditional reading of the history of computing in the 1940s as a series of incremental steps leading to the "modern computer." Colossus is an exemplary artifact of what we like to call the "early digital" because of, not despite, its fundamental lack of architectural resemblance to a computer.[58] Our analysis underlines the need for a historiography of the early digital that is concerned with use as much as invention and for a history of computing fully integrated with broader historical analyses, such as social, business,

[54] Max Hastings, *The Secret War* (New York, NY: Harper Collins, 2016).

[55] Colin B Burke, *It Wasn't All Magic: The Early Struggle to Automate Cryptanalysis, 1930s-1960s* (National Security Agency, 2002). We expand on this point in another paper, working title "Colossus in Context," currently under review with *Technology & Culture*.

[56] de Grey to Radley, March 29, 1944, HW 62/6, TNA.

[57] Flowers, "The Design of Colossus".

[58] Thomas Haigh, ed., *Exploring the Early Digital* (Springer, 2019).

labor, and military history. If there has been such a thing as a digital revolution then it involved much more than just computers.

**Acknowledgements:** This project was generously supported by by Mrs. L.D. Rope's Second Charitable Settlement. We extend our thanks to Crispin Rope and to all those at Lucy House. Also to those who commented on drafts prior to submission including Brian Randell, Martin Campbell-Kelly, Quinn DuPont, and others at the 2016 SIGCIS workshop and the National Museum of Computing. We appreciate the helpful suggestions and careful analysis of anonymous reviewers 2 and 3, which strengthened the paper significantly, and the editorial guidance of David Hemmendinger.