Mark Priestley; Thomas Haigh

Colossus: The Missing Manual

2019

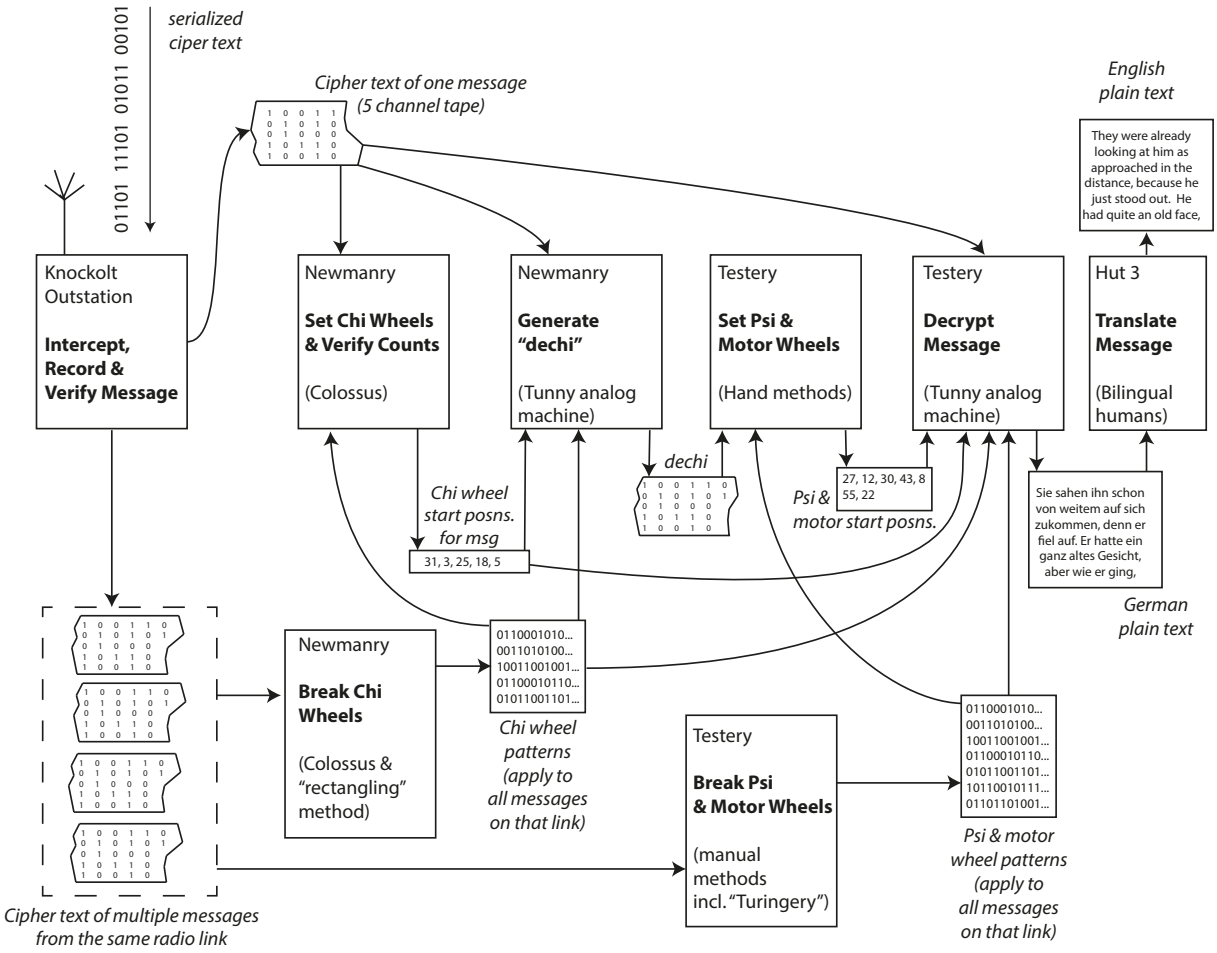
https://doi.org/10.25969/mediarep/13804

Veröffentlichungsversion / published version
Working Paper

Empfohlene Zitierung / Suggested Citation:
Priestley, Mark; Haigh, Thomas: *Colossus: The Missing Manual*. Siegen: Universität Siegen: SFB 1187 Medien der Kooperation 2019 (SFB 1187 Medien der Kooperation – Working Paper Series 10). DOI: https://doi.org/10.25969/mediarep/13804.

Colossus: The Missing Manual

Mark Priestley *Research Fellow, The National Museum of Computing — Bletchley Park, UK*
Thomas Haigh *University of Wisconsin—Milwaukee & Siegen University*



WORKING PAPER SERIES | NO. 10 | OCTOBER 2019

Collaborative Research Center 1187 Media of Cooperation
Sonderforschungsbereich 1187 Medien der Kooperation

Working Paper Series
Collaborative Research Center 1187 Media of Cooperation

Print-ISSN 2567–2509
Online-ISSN 2567–2517
URN urn:nbn:de:hbz:467-15072

Publication of the series is funded by the German Research Foundation (DFG).

This Working Paper Series is edited by the Collaborative Research Center Media of Cooperation and serves as a platform to circulate work in progress or preprints in order to encourage the exchange of ideas. Please contact the authors if you have any questions or comments. Copyright remains with the authors.

The cover image shows Colossus in use as part of a broader system of collaboration, including many teams of humans and machines and the media passed between them.

The Working Papers are accessible via the website http://wp-series.mediacoop.uni-siegen.de or can be ordered in print by sending an email to: workingpaperseries@sfb1187.uni-siegen.de

Universität Siegen, SFB 1187 Medien der Kooperation
Herrengarten 3, 57072 Siegen, Germany
www.sfb1187.uni-siegen.de
workingpaperseries@sfb1187.uni-siegen.de

Like many modern electronic devices, Colossus did not come with a printed manual. We provide the missing information needed to get the most out of it.

Colossus: The Missing Manual

Mark Priestley *Research Fellow, The National Museum of Computing — Bletchley Park, UK* — m.priestley@gmail.com
Thomas Haigh *University of Wisconsin — Milwaukee & Siegen University* — thomas.haigh@gmail.com

Abstract There has until now been no comprehensive, convenient, and reliable description of the actual capabilities of the Colossus codebreaking machines used at Bletchley Park during World War II, the way they were used, and the jobs they were applied to. This gap in the literature has led to a lack of understanding of the machines' functionality and hence to exaggerated claims about their capabilities. In this report we remove the Colossi as far as possible from their cryptanalytical context and consider them simply as computational devices. We give an architectural description of the whole family of related machines, including the initial model known as "Heath Robinson", and a functional description of the major capabilities of the second and final Colossus design. We include detailed examples of how the machines would have been set up to perform a range of typical tasks, and conclude with a discussion of their versatility, examining in particular the question of how useful they would have been once the war had ended. We present several examples of actual Colossus configurations and the historical output they produced, illustrating the cooperation between figures typed automatically by Colossus and text and annotations added by the human operator.

Keywords Colossus, Fish, Bletchley Park, cryptography, Robinson

Introduction

A great deal has been written about the Colossus machines, but descriptions of their architecture and functioning are usually embedded in detailed histories of code breaking techniques and the work of Bletchley Park. Although Colossus is usually called a computer, and sometimes claimed as "the first computer" (or, more precisely, as "the first programmable electronic computer") it has not been clearly described from the viewpoint of computer architecture. Claims that Colossus had a crucial and often overlooked place in the history of computing hinge on the idea that, if not decommissioned at the end of the war and kept secret for decades thereafter, the machines could have been applied to many computing tasks beyond the specific codebreaking work for which they were designed. This is a counterfactual discussion, but the validity of such claims depends on the extent to which Colossus had programming mechanisms that could be reconfigured to tackle scientific computing work. However, these programming mechanisms have not been comprehensively described.

Most descriptions of Colossus begin with a mass of detail about code breaking techniques and work at Bletchley Park, focusing more on the details of codebreaking practice and the quirks of the Lorenz machines Colossus was designed to attack than on the capabilities of the Colossus machines themselves. Our original intention was to reverse the usual structure of Colossus discussion by presenting as little information as possible about codebreaking practice. Instead, we aimed to produce the "missing manual" for Colossus, explaining its capabilities and programming mechanisms in general terms.[1]

1 This title is inspired by the "missing manual" series of books written by David Pogue and others in the early

As our research progressed we discovered that there was a good reason for the previous tendency to embed descriptions of Colossus deep within discussion of codebreaking techniques: Colossus was not programmable after all, and could not have usefully been applied to tasks other than attacking the specific cipher for which it was designed without considerable hardware modification. So we had to go much deeper than we originally anticipated into the details of codebreaking to make its capabilities and architecture fully comprehensible. Nevertheless, we have attempted to honor our original objective as far as possible. We begin with an abstract look at the Colossus family machines as information processing devices, describing Colossus as a bitstream processor before explaining, to the minimum extent necessary, its relationship to codebreaking practice and offering a range of codebreaking configurations as examples of how the Colossi were actually used. In further publications we plan to delve more deeply into that practice by exploring the full range of surviving Colossus output and associated materials.

1 The Colossus Family as Bitstream Processors

The details of how Bletchley Park managed to retrieve key sequences from encrypted messages, and so read Fish traffic, are fascinating but exceedingly complex. For the purposes of this report, it is enough to know that the machines in the Colossus family were originally designed to automate a particular operation within this complicated procedure. This involved combining bitstreams derived from a coded message with other bitstreams derived from simulated code wheels. At a functional level, then, we can view the Colossus family machines as bitstream processors.[2]

Considered as information processing devices the overall structure of the Colossus family machines was simple: they took bitstreams as input and produced counts as output. The relationship between the bitstreams and the counts could be configured with a large and diverse assortment of controls, which combined and transformed the inputs.

Most Colossus family machines had two input streams[3]; each input stream consisted of 5 bitstreams which we refer to as the "channels" of the input stream. (In contemporary discussions, the five channels were called "impulses.") The 5 channels allowed the encoding of the 32 characters in the conventional teleprinter alphabet used by the Germans but, as we discuss below, the machines treated the channels as independent streams.

At least one of the input streams was read at high speed from a paper tape loop. The source of the second stream was the key difference between the Robinson branch of the family and the Colossus branch. Robinsons read both streams optically from paper tape. Colossi generated the second stream electronically, based on bit patterns set up on control boards. Most Colossus family machines printed out potentially relevant counts using a typewriter mechanism.

1.1 A Note on Terminology

It is not quite accurate to talk about the reading or combining units of Colossus family machines manipulating characters or numbers. They processed each channel as an independent series of bits, rather than interpreting bits from different channels as characters or binary coded numbers.

Colossus read five channels from its paper tape, but it never combined these as the five constituent bits of a single number. In contrast, computers clump bits together in fixed-length chunks (such as 8-bits, 16-bits, and so on) to represent numbers. Each bit is conventionally represented as 1 or 0, but together 8 bits represent any number between 0 and 255. The value of each bit depends on its position in the number being encoded, just as, in decimal number representation, the value of the number 5 differs according to whether it is placed in the tens column or the millions column. That number, in turn, might encode a letter or the color of an individual pixel. In contrast, the Colossi treated the bitstreams as logically separate data sources, and most jobs took as input no more than two of the five bitstreams that collectively encoded a single character in conventional teleprinter applications.

The machines detected holes and the absence of holes on punched tapes, and represented these two possibilities in various ways: the presence and absence of a pulse, for example, or the phase of an electrical signal. Individual bits were called "characters," and rather than representing the two possible values of each "character" as 0 and 1, or as true and

2000s, once hardware and software producers stopped including printed manuals with their wares. Colossus, predating the long era of computer manuals, likewise lacked a comprehensive and clearly presented reference guide.

2 There is a precedent for this terminology in the captions that were added to the photographs of Colossus that GCHQ released in 1976, and in the 1973 history of Colossus prepared by GCHQ. The UK National Archives (hereafter "TNA"), FO 850/234 ("Annotated photographs of the COLOSSUS Electronic Digital Computer") and HW 25/24 (D. C. Horwood, "A Technical Description of Colossus 1").

3 The exceptions here were some of the later Robinson machines, delivered in 1944, which could combine bits read from four tape drives.

false, the users of Colossus referred to them as "dot" and "cross," terms whose exact significance varied with context. Colossus could be set up to count either value, so did not inherently favor one interpretation over another.

Likewise, although the codebreakers used the conventional teleprinter alphabet to represent five-channel bit patterns read from tape, primarily to represent distribution counts obtained for verification purposes, this was nothing more than an external notation: Colossus included no specific capabilities to manipulate characters.

As our aim here is to reconstruct the capabilities of Colossus as a computer, an irreducibly ahistorical project, we decided to adopt computing terminology by using "bit" for "character," "channel" for "impulse," 1 for dot, and 0 for cross.

Even calling these signals "bits," a term that originated as a contraction of "binary digit" is a little misleading, however, as it might suggest that the Colossus family represented numbers in a base 2 system. It's worth emphasizing that this was not the case. Although the Bletchley Park mathematicians recognized the similarity between the most important operation on these bits and what they termed "modulo 2 addition," the bits themselves were semantically neutral representations of the two possible states of hole/no hole, or various other physical distinctions. The only place where numbers were stored was in the counters and in the uniselectors holding code wheel start positions, and here all the machines used decimal (base 10) representations.

1.2 Heath Robinson: The Original Colossus Family Architecture

In this report, references to "Colossus family machines" include Robinson series machines as well as the ten Colossus machines themselves.[4] There were many variants of Colossus and Robinson, but we focus here on the two best-documented: the Heath Robinson prototype that proved the workability of the Colossus family in the summer of 1943 and Colossus 2, the template for the later Colossus machines, which went into action about a year later.

Heath Robinson was the very first machine, and it set the stage for all subsequent developments. Its basic architecture is preserved throughout, and many of the modifications in later machines were made in response to experience gained in using Heath Robinson.

Heath Robinson defined the basic architecture of the Colossus family. It consisted of three main units. The *tape reader* was designed and built at the UK General Post Office's research establishment at Dollis Hill under the direction of Tommy Flowers. *Counters* were built at the Telecommunications Research Establishment (TRE) in Malvern under the direction of Charles Wynn Williams, a scientist and pioneer of electronic counter technology. Finally, the *combining unit* was a collaboration, designed by TRE but built at Dollis Hill. Once completed, these three units were delivered to Bletchley Park where the complete machine was put together and tested.

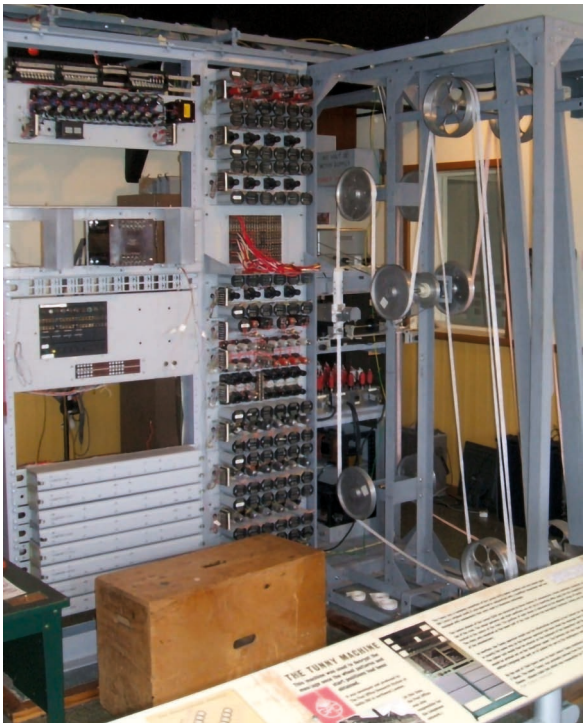


Fig. 1: The Heath Robinson reconstruction at the National Museum of Computing in 2015. Photograph: Mark Priestley.

These three units are clearly visible in the Heath Robinson reconstruction at the UK's National Museum of Computing (see Figure 1). The tape reader on the right senses bits punched onto the input tapes, and runs them into the combining unit in the center. This was often called the "valve rack," after the British term for vacuum tubes. The selected inputs are combined according to the circuits wired on the plug board, and the results of the combination are routed into the counters in the left-most panel.

Logical connections between the units were fairly simple, as shown in the diagram below.

Readers: The Colossus family machines read five-channel paper tape, of the kind used with teleprinters. The machines were timed to the input tape(s), so a new processing cycle began each time the tape advanced. In the two-tape Robinsons, this meant that the two high-speed tape readers needed to be

4 At the end of the war, 10 Colossi were in service and an eleventh was in the course of being assembled.

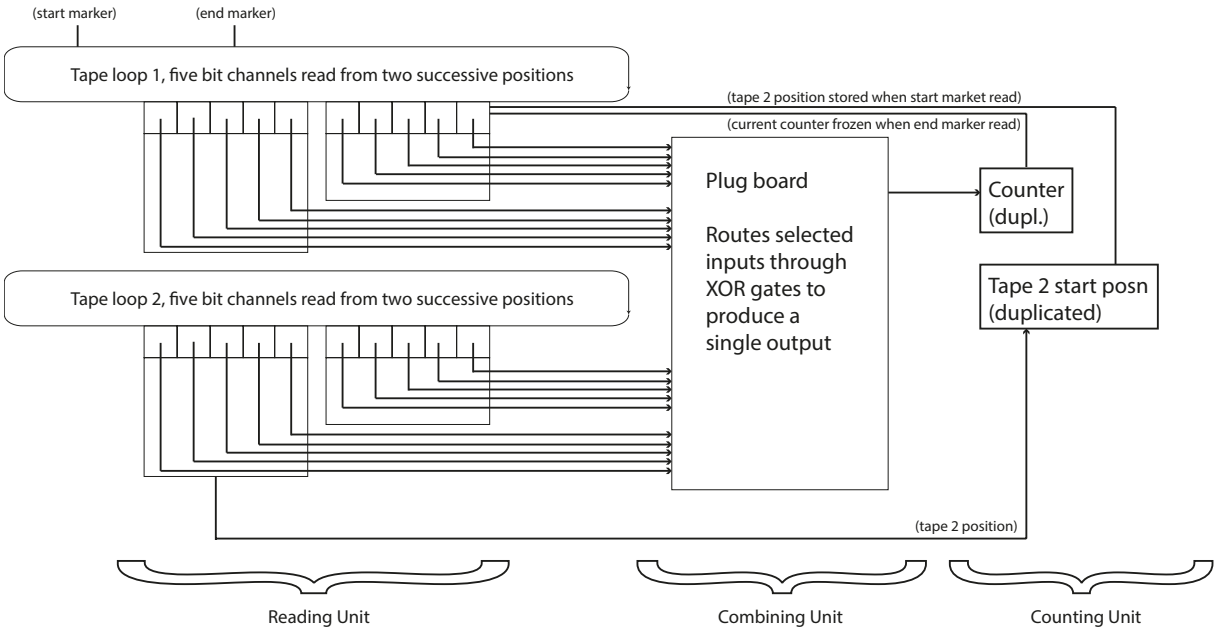


Fig. 2: Data flows within Heath Robinson, emphasizing its separation into three units for reading, combining, and counting.

precisely synchronized, a challenging and problematic task for the engineers. Heath Robinson read two consecutive bit positions from each tape simultaneously, so each reader had two sets of read heads, outputting a total of twenty bits. Specially placed holes on the tape marked the beginning and end of the sequence. The readers also detected these holes, which provided control information telling Heath Robinson when to start and stop processing the data from the bitstreams. These inputs let Heath Robinson track the current and starting position of each tape.

Combining Unit: At each cycle, then, Heath Robinson's readers read four positions from its two five-channel tapes. This delivered a total of 20 bits to the second main architectural component of Heath Robinson, the combining unit. This unit contained some rather idiosyncratic circuitry that allowed selected input bits to be XORed together. Also described at Bletchley Park as modulo-2 addition, this was the basic logical operation required to carry out the machine's principal cryptanalytic task. The design of these circuits also allowed other tasks to be plugged, such as counting the number of holes punched in a particular channel of the paper tape.

Counters: Each Colossus family machine had counters, used to accumulate the results transmitted on the output line(s) of the combining unit. Totals increased when outputs from the combining units fired. Heath Robinson counted just one output from the combining unit, but alternated between two counters. A count completed every time the first tape loop reached a special stop market, indicating the end of the character stream. The total was displayed in one counter during the next revolution of the tape, as the new total accumulated in the other counter. Another pair of counters recorded the corresponding start positions of the second tape when the message on the first tape began again.[5] Operators had to write down totals and start positions of interest quickly, before they were replaced by newer values. Heath Robinson ran its tapes at up to 2,000 characters per second, so a message of 6,000 characters would take only three seconds to read.

Robinson Algorithm: Most of the jobs run on the Colossus family machines involved obtaining counts for different starting positions for the two input streams. Because it took only a few seconds to read the message tape it was not practical to start and stop the machine each time the relative starting positions of the two input bitstreams had to be changed. Instead tape loops cycled continually.

This meant that the tapes had to be of particular lengths, to ensure that each time the first tape completed a revolution the second tape was in a start position that had not yet been evaluated. Heath Robinson operators used relatively prime lengths for the first input tape loops, to ensure that allowing both tapes to cycle continually would generate each possible start position for the second tape. When the two

5 We are not sure whether the counter held the actual value of the start position of the second tape when the first tape last completed a revolution, or a count of some kind from which the start position could be derived.

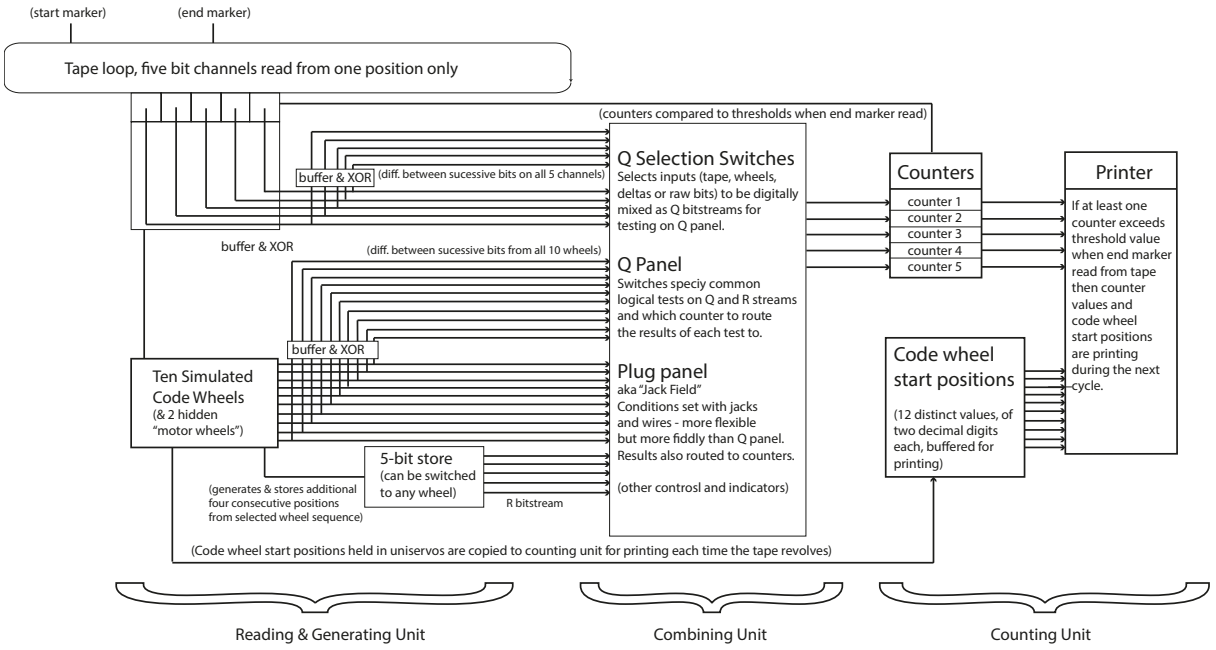


Fig. 3: Data pathways between the three logical units of Colossus (includes the 5-bit buffer introduced with Colossus 2).

tapes started a cycle in the same position relative to each other they had been in at the beginning of the run that indicated that all possible combinations of start positions had been tested. The machine sensed this and set an indicator light, signaling the operator that the run was complete.

1.3 Colossus 2: The Colossus Architecture

Robinson's two tapes gave great flexibility, but also brought problems. The tapes were liable to break or tear under the stress of high-speed rotation, and even when they didn't break, synchronization was a persistent problem. Added to that was the labor of producing the tapes and the difficulty of ensuring that they were error-free. The Colossus machines introduced a number of changes to the Robinson model, but the most significant was the replacement of Robinson's second tape by internal circuitry that replicated the behavior of the Lorenz machines' wheels.

There is a very clear distinction between the first Colossus and all the subsequent ones. The first was later sometimes referred to as a prototype, but the second ("Colossus 2") was the template for all subsequent machines, even though each had modifications and customizations for specific tasks. Colossus 2 is the definitive Colossus architecture, described in the overwhelming majority of the secondary literature and reincarnated in the rebuild at the UK's National Museum of Computing (TNMOC). By contrast, Colossus 1 is rather poorly documented. Later Robinsons and Colossi acquired additional control panels to speed set up, and in some cases additional input and output devices (a reported tape punch for Colossus 6, an extra two input tapes for Super Robinson). These are not well documented, but we know of no evidence that these tweaks changed the architecture or basic control capabilities of the machines. Therefore we focus here on Colossus 2.

Unlike Heath Robinson the Colossus machines were designed and built entirely by the Post Office, as integrated machines rather than an assemblage of separately procured modules. Physically Colossus consisted of many panels integrated into two rows. The Colossus machines were more complicated than Heath Robinson, and used many more electronic components.

Logically, however, the basic architecture of the Robinsons remained intact despite a substantial increase in the complexity of each of the three modules. Most significantly, data flows and control signals between them remained very constrained. The Colossus panels equivalent to Robinson's combining unit, for example, had more inputs, more outputs, and many more ways of establishing connections between them. Yet it remained stateless: no information from the examination of one set of inputs was retained when processing the next bit position from each input channel. The only course of action it could take based on its inputs was to increment one or more counters, but the contents of these counters could not be read back as inputs for the combining unit.

Readers: Like the Robinsons, the Colossus machine read a sequence of inputs from a five-channel "message tape." Instead of reading two consecutive bits from each channel, however, the reading unit was

augmented with five 1-bit buffers each storing the previous bit read from one of the channels. Rather than provide the combining unit with current and previous bits from each channel on the tape, as Heath Robinson did, Colossus used the buffers to provide the current bit and the delta (XOR) between this and the previous bit. The sprocket holes in the message tape were detected, and the resulting pulses used to time the operation of the machine. The tape included start and stop perforations, as before.

Colossus machines generated internally the data that would typically be stored on Robinson's second input tape. They incorporated 12 "thyratron rings" each simulating one of the 12 encoding wheels of the Lorenz machine. These were configured in two ways. Firstly, the pattern of dots and crosses on each wheel was specified on special plugboards. Secondly, the starting position of each wheel was set by a switch. Other controls stored the initial start position and controlled the stepping of each of the simulated wheels. Once Colossus sensed the end of the input stream on its tape it incremented the start position of the simulated code wheels (in accordance with the stepping switches) and reset them, which had the effect of beginning the second input sequence at a different point.

Inputs available to the combining unit represented the output of at least 10 of the simulated code wheels.[6] These signals were used to generate and output deltas between the current and previous bit positions on each wheel. So this part of Colossus passed at least 20 bit channels on to the combining unit, as opposed to the 10 produced by Robinson's wheel tape reader.

Colossus added a five-bit electronic buffer that could be switched to one of the electronically generated bit channels.[7] This allowed users to test five consecutive positions for the corresponding code wheel each time the message tape looped, dramatically speeding up some operations as the buffered code wheel's starting position could then be stepped by five positions, rather than one, each time the tape was read.

Combining Unit: As in Robinson, the resulting bitstreams were delivered to a plugboard where they could be combined in various ways. The range of operations available on a Colossus plugboard was wider than the simple modulo-2 addition (XOR) provided by Robinson, providing "logical addition" (OR) as well as negations and the ability to replicate a bitstream for multiple testing. These could be combined to implement (within the physical limitations of the machine, such as the number of gates of each kind) any desired truth table mapping inputs to output.

In addition to the plugboard, Colossus machines provided a "Q panel" on which some commonly used tests could be set up quickly using switches.[8] A separate set of mixer controls decided what combination of bitstream inputs should be combined for each of the five input channels of the Q panel. It supported two kinds of tests. Firstly, the five bits coming from the mixer could be tested for any desired pattern of dots and crosses. Secondly, any or all of the five bits could be added together, modulo 2, and the resulting sum tested to see if it was a dot or a cross. Conditions could also be set on the bits coming from the buffer.

Counting Unit: Whereas Heath Robinson had only one output from the combining unit, the Colossus machines had five, making it possible to carry out several logical tests simultaneously with the results routed to separate counters. Colossus printed its results using an electromatic typewriter, which was more reliable than the printer added to the production Robinsons. Users set a threshold for each of the counters, above (or optionally below) which its contents would be printed. Once the tape had been fully read, dedicated circuits compared the value stored in each counter to the corresponding threshold value. If the threshold condition was met, then Colossus printed the values stored in each counter, along with the starting positions of each simulated electronic code wheel. A "span control" feature added to later versions of Colossus suppressed counting except within a designated subset of the input stream.

Colossus Algorithm

As with Robinson, many runs on Colossus involved cycling the tape loop and tallying counts repeatedly, each time with different start positions for the second bitstream. The Robinson machines accomplished this with a minimum of control capabilities. As the appropriate sequence for the job had already been generated and punched onto tape, the only real control needed was an indicator that checked, each time the first tape returned to its start position, whether the second tape had also returned to its start position. This would indicate that the job was over.

Colossus generated the second bitstream internally, which made its algorithm rather more com-

6 All Colossus machines output current bits and deltas for the five chi and five psi wheels. Treatment of the two motor wheels seems to have varied as time went by, but later machines had some ability to work directly with them.

7 This buffer was not in the prototype Colossus, but was added for the second and subsequent models and may eventually have been retrofitted to the first model.

8 The prototype Colossus did not contain the full Q panel, but some sources suggest that it had a more limited capability to set up logical tests with switches in addition to its plugboard.

plex. As discussed in the next section, it did this by simulating electronically the movement of twelve code wheels present in the Lorenz coding machine Colossus was designed to target.

In abstract terms, Colossus implemented a set of nested loops. The authors of this report disagree on the question of whether it is most informative to consider the program carried out by Colossus as involving two levels or three levels of nested loops. We agree that Colossus effectively executed an inner loop every time a new character position was read from the tape, and carried out a second level of looping by incrementing code wheel positions every time the message tape was read. We disagree on whether the fact that Colossus could be configured to step some code wheels slowly, advancing them only when fast stepping wheels had returned to their start positions (like the relationship between the units and tens digits in a speedometer), justifies separating out the incrementing of slow and fast wheels as two separate levels of looping. Haigh focuses on the behavior of Colossus when slow and fast stepping are both used, likening it to a classic nested loop. Priestley feels that Colossus's control circuits are more accurately modeled as two nested loops, and that, even though it may be an accurate functional model, the three-loop diagram gives the distinction between slow and fast stepping a prominence that it didn't have in either hardware or usage. Figure 4 reflects Haigh's preference.

The inner loop repeated until the tape loop had been read in its entirety. This consisted, essentially, of a single operation. Once the tape reader signaled that a new character had been read, Colossus advanced the electronic code wheels to obtain the next position of the other five-channel input stream. It then fed the latest set of inputs through the combining unit, which evaluated the logic expression set up there. Colossus then incremented its counters based on the outputs of the combining unit, before settling down to wait for the next character to be read from tape.

The two other loops were concerned with the stepping of the simulated electronic code wheels. Colossus systematically tested the tape input against different start positions in the electronically generated bitstreams. Before a job started the user would select initial start positions for each code wheel. Each time the tape loop was completely read, Colossus reset the code wheels to their current start positions. Before it did this it could increment the current start position of one or more of the code wheels.

A "fast stepping" wheel had its start position incremented each time the tape loop was completely read. This corresponds to the middle loop. While incrementing the wheel's start position Colossus would also compare the final values of the counters to the assigned thresholds. Any values to be printed were copied into a print buffer and the counters were then rest, ready for the next cycle of the tape.

A "slow stepping" wheel had its own start position incremented only when the fast stepping wheel's start position had cycled all the way around and returned to its initial value. This corresponds to the outer loop.

If, after stepping, the current start positions of all fast and slow stepping wheels matched their initial start positions then the job was over. If no wheels were set to step then this would occur after a single cycle of the message tape. If there was a fast stepping wheel set for testing but no slow stepping wheel set for testing then this would occur in a minute or so, which was known as a "short run." If one slow stepping wheel and one fast stepping wheel were both set for testing then the job might take ten or twenty minutes, which was known as a "long run."[9]

In this sense the basic algorithm or program followed by Colossus could not be changed by the user.[10] There were, as we discuss below, many parameters that could be altered: the logical conditions set in the combining unit for counting, the thresholds for printing, the wheel stepping behavior, the span of tape in which counts were made. These all plugged values into specific steps of the algorithm. But none of them altered the basic sequence of operations hardwired into Colossus.

2 Decoding Tunny

So far we have described the Colossus machines in neutral terms, distinct from their applications. However, to describe Colossus in use and make sense of its specific capabilities we do need to say something about the code machine it targeted. Full details, and a historical narrative of the breaking of the Lorenz ciphers, are available in many other sources.

The Colossus family machines were developed as part of Bletchley Park's attack on a family of German ciphers known as "Tunny" or, more generally, "Fish." Encryption was performed automatically, by means of mechanical attachments to the standard teleprinter machines used by the German military command for long-distance communication. Messages could be punched on paper tape or transmitted directly by radio link. The Lorenz attachments encoded messages by generating a sequence of characters known as "key." A message was encrypted by lining it up with a key sequence, combining each character with the corresponding character of key to produce an en-

9 The run finished when all wheels matched their start positions at the beginning of a cycle. Thus a long run could also be produced by setting two wheels to step fast and testing both for termination.

10 We explore this idea more in Thomas Haigh and Mark Priestley, "Colossus and Programmability," *IEEE Annals of the History of Computing* 40, no. 4 (Oct-Dec 2018): 5–17.

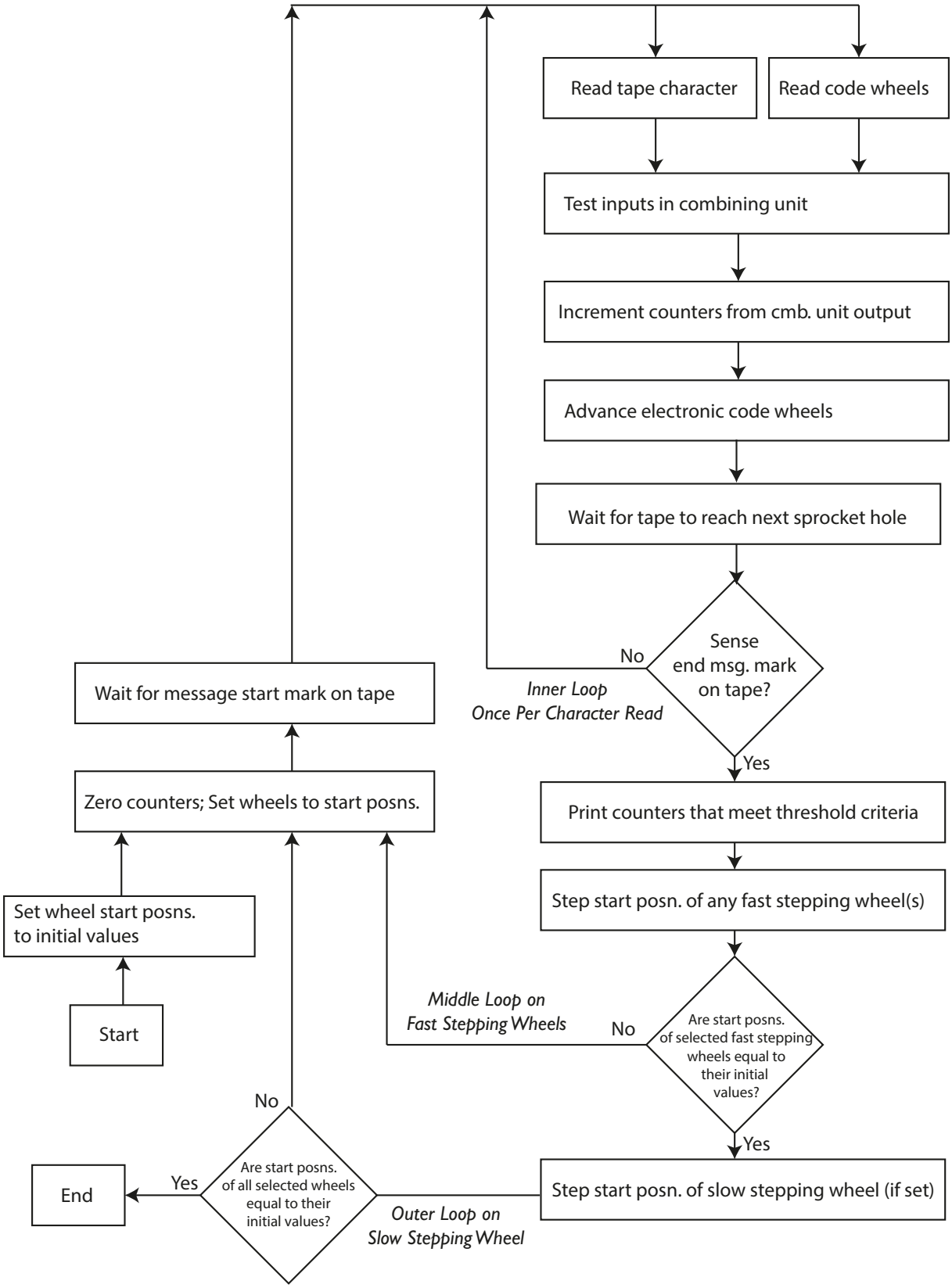


Fig. 4: Flowchart representation of the program carried out by Colossus. This represents the behavior of the machine as a whole as a fixed series of operations and tests. The program was not represented or stored explicitly anywhere in Colossus. Note that setting wheels to step (fast, slow, or not at all) and specifying which wheels to test for loop termination were separate choices.

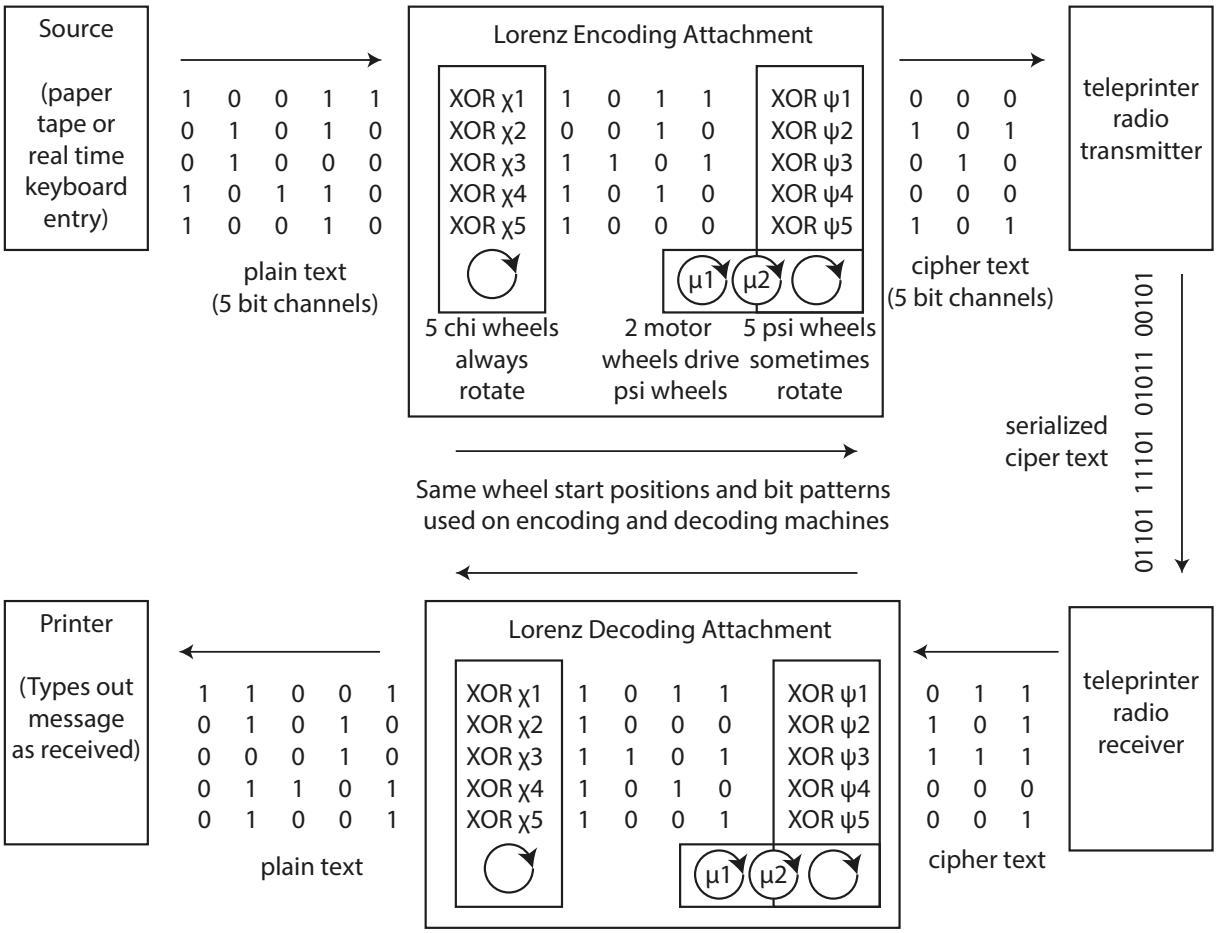


Fig. 5: The schematic operation of the Lorenz cipher machines that Colossus family machines were designed to tackle.

crypted character. Decryption repeated this process: the original message was recovered by combining the encrypted message with the same key sequence.

2.1 Code wheels on the Lorenz machine

Each Lorenz cipher machine held twelve wheels. Two wheels acted via "logical addition" on each of the five channels of the message being encrypted. One set of five wheels, known by the codebeakers as the "chi wheels", rotated together every time a character was processed. The other set, known by the codebreakers as the "psi wheels", also rotated together but did not always move. Whether or not they moved depended on the action of the final two wheels, dubbed the motor wheels.

Heath Robinson could carry out a comparison between any two tapes. Initially, however, it was intended to be used with "wheel tapes" that contained data derived from a partial simulation of the Lorenz machine. The wheel tapes used by the Robinsons were generated by machines that simulated these aspects of the Lorenz machines. The wheel patterns would be set up and tapes punched containing the bits contributed by specific wheels to the key. Perhaps the most important difference between the Robinson and Colossus machines was that the latter included circuitry that performed this simulation internally.

Because the Robinsons read both input streams from tape they could handle a broader range of tasks than Colossus. One could punch any desired set of five bitstreams onto either tape loop, padding as necessary to ensure that their lengths were relatively prime. The machine would then automatically try every combination of start positions for the second stream against the first.

Colossus traded generality for speed and efficiency of setup. As long as one of the input streams represented code wheels from a Lorenz cipher machine, a Colossus could tackle the job much faster than a Robinson. Eliminating the need to synchronize two paper tapes made it possible to read the remaining tape faster – at 5,000 characters per second rather than 2,000. With a Colossus, changes to simulated code wheel bit patterns could be made in an instant by adjusting controls. Using a Robinson any change to the input wheel pattern would require punching a new input tape. Even shifting the attack from one set

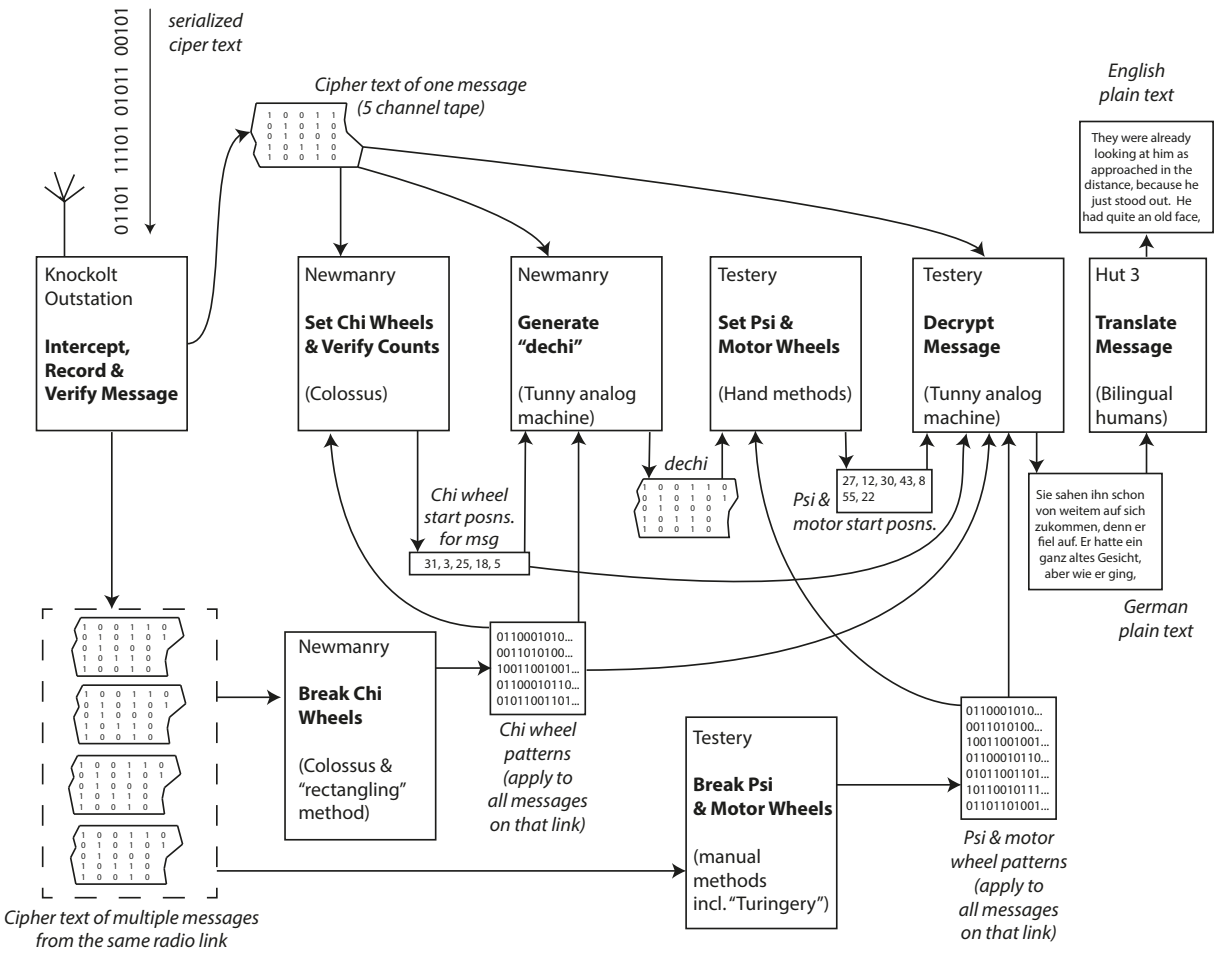


Fig. 6: The overall workflow involved in producing decrypted Tunny messages, as of late 1944. Although Colossus could be applied to several steps in the process, its core applications were in setting and breaking chi wheels.

of code wheels to another, which was necessary at least twice in determining the code wheel settings for each message, would require the loading (and quite possibly the punching) of a new tape.

If you want to read an encrypted Tunny message, you need to know the key. The key is generated by the interaction of the wheels on the Lorenz machine, so what you need to know is the machine settings used to encode the message. There are two aspects to this.

First are the bit patterns set on each of the Lorenz machine's twelve code wheels. These were varied by moving pins. Early in the war they were changed infrequently, but by the end of the war they were changed daily. The same bit patterns were used for all messages sent over a particular radio link. Figuring out the correct bit patterns was known as "wheel breaking." For a long time, wheel breaking was carried out by human codebreakers, though as experience was gained with the Colossus family machines, they were also applied to this task.

Second are the starting positions for each of the ten code wheels. These changed with each message sent. Finding the correct start positions was known as "wheel setting."

Figure 6 shows the overall organization of Tunny-breaking work at Bletchley Park. Colossus, like Heath Robinson, was initially applied to one of these tasks: setting the chi wheels. This diagram represents the typical situation in late-1944. By this point the Colossus family machines were also used to break the chi wheels. Some psi and motor wheel setting was also done with Colossus, but as Colossus time was scarce most of it was still done by manual methods. This distinction between machine and manual methods was also an organizational one: the Colossus family machines were the central concern of a section of Bletchley Park generally known as the Newmanry, while manual codebreaking and the decryption of messages was handled by a separate section: the Testery. Both were named after their leaders.

2.2 Colossus vs Heath Robinson for Wheel Setting

A Colossus family machine would need to make multiple runs to tackle any actual codebreaking job. So the overall throughput of the machines depended not just on the speed at which they could read the

input tape, but also on how much work could be accomplished during each run and on how long the machine sat idle between runs.

The Colossus machines could complete the series of runs needed to set chi wheels for a message much more rapidly than Heath Robinson. There were five main reasons for this, exploration of which reveals the ways in which operator practices and technological capabilities coevolved to boost productivity.

First, the Colossus machines could cycle through the message tape faster, reading it at up to 5,000 characters per second rather than 2,000 for Heath Robinson. Eliminating the second tape also eliminated a common source of error: problems in synchronizing the two input tapes.

Second, later versions of Colossus could apply "multiple testing" techniques to evaluate up to five possible start positions for one of the chi wheels each time the message tape cycled through.[11] Colossus provided five counters, into which runs of this kind could simultaneously accumulate counts for five wheel position combinations. The additional counters speeded some other kinds of run too, for example in a character counting run each counted for a different character.

Third, Colossus eliminated a great deal of work preparing input tapes. Bit patterns for the wheels and stepping of initial start positions (which would depend on which wheels were being broken on a particular run) were set up with switches and plugs. Heath Robinson, in contrast, had relatively few configurable settings as the wheel inputs were read from tape. Every time bit patterns on the wheels changed, operators would have to produce a set of tapes holding the full sequences for different wheel combinations – for example a tape 1,271 characters long with the full cycle for chi-wheels 1 and 2. Newman's description of procedure suggests that the same tapes could be used for long runs and short runs. For example, the short run to set wheel 4 would use the 1066-character tape prepared to give the full sequence for chi wheels 1 and 4. However this would only work if the length of the message tape was a multiple of 41, the length of wheel 1, so that every run aligned the start of the message with the correct (known) start position for wheel 1 but with a different start value for wheel 4. This technique implies a need to duplicate several versions of each message, padded to the lengths needed for different runs, or a need for Heath Robinson operators to be cutting and splicing messages between runs to adjust their length. According to a contemporary report, "In sticking the tapes into loops for Robinson, great care has to be taken to get their length precisely right. So many errors were made at this that an appreciable amount of time was lost. Also tapes had to be recopied frequently because a different length was needed for a new run."[12]

Fourth, Colossus added a great deal of logical flexibility to the combining unit, which combined with the on-the-fly generation of simulated ring sequences allowed for more complex logical comparisons between inputs. Heath Robinson had been designed with simple runs in mind. The basic technique used in runs such as "1+2" involved a cascade of eight "phase shifters," the circuit that XORed a signal with the result of previous operations.

The Colossus machines used a similar sequence of comparisons for the initial run. However, subsequent runs were able to incorporate inputs from more than two channels from the simulated wheels, which yielded extra information and so boosted chances that a clearly significant result would be achieved when the message tape was cycled against the correct setting for the unknown wheel(s). For example, the "4=5/1=2" Colossus run incorporated signals from the known positions for wheels 1 and 2. These were configured not to step their start positions, so that the correct start positions would be used every time the tape cycled. Such a test would have been infeasible with Heath Robinson, as it would require a wheel tape of 760,058 characters in length (1271 x 26 x 23) and a message tape that was a multiple of 1271 characters in length so that it stayed aligned with the correct settings for wheels 1 and 2.

Colossus also added more kinds of logic gate to its plug board. Heath Robinson's XORs were sufficient for runs such as 1+2. A run such as "4=5=/1=2" required other logical operations to test whether four bits were all the same. Carrying out "3+4x/1x2x" demanded a logical AND over three inputs (i. e. two binary AND gates).

Fifth, the Colossus combining unit was designed to speed the setup of common tests. The Colossus machines eliminated the need to wire XOR gates to generate deltas on input channels by providing switches that could be flipped to switch input channels between actual bit values and deltas.[13] Other switch panels let operators set up the comparisons for most runs quickly, without the need to move wires.

11 On the 1+2 run this provided a speedup of only 4.55 rather than five times. The wheel lengths were 41 and 31. Applying the array unit to wheel 1 would evaluate the first 40 start positions in eight revolutions of the tape, but on the ninth would evaluate only one new start position (and duplicate results for the first four). The Colossus would complete the run after 279 revolutions of the tape, rather than the usual 1271. During this time it evaluated 1395 wheel combinations, of which 124 were duplicates. So the maximum possible speedup on this run would be a factor of 4.55.

12 National Archives and Records Administration (USA): NARA-HCC b579. Report on British Attack on Fish, p. 47.

13 Surviving evidence is incomplete and contradictory over exactly how the first Colossus handled the generation of deltas, which operations its plugboard provided, and whether it provided any switches.

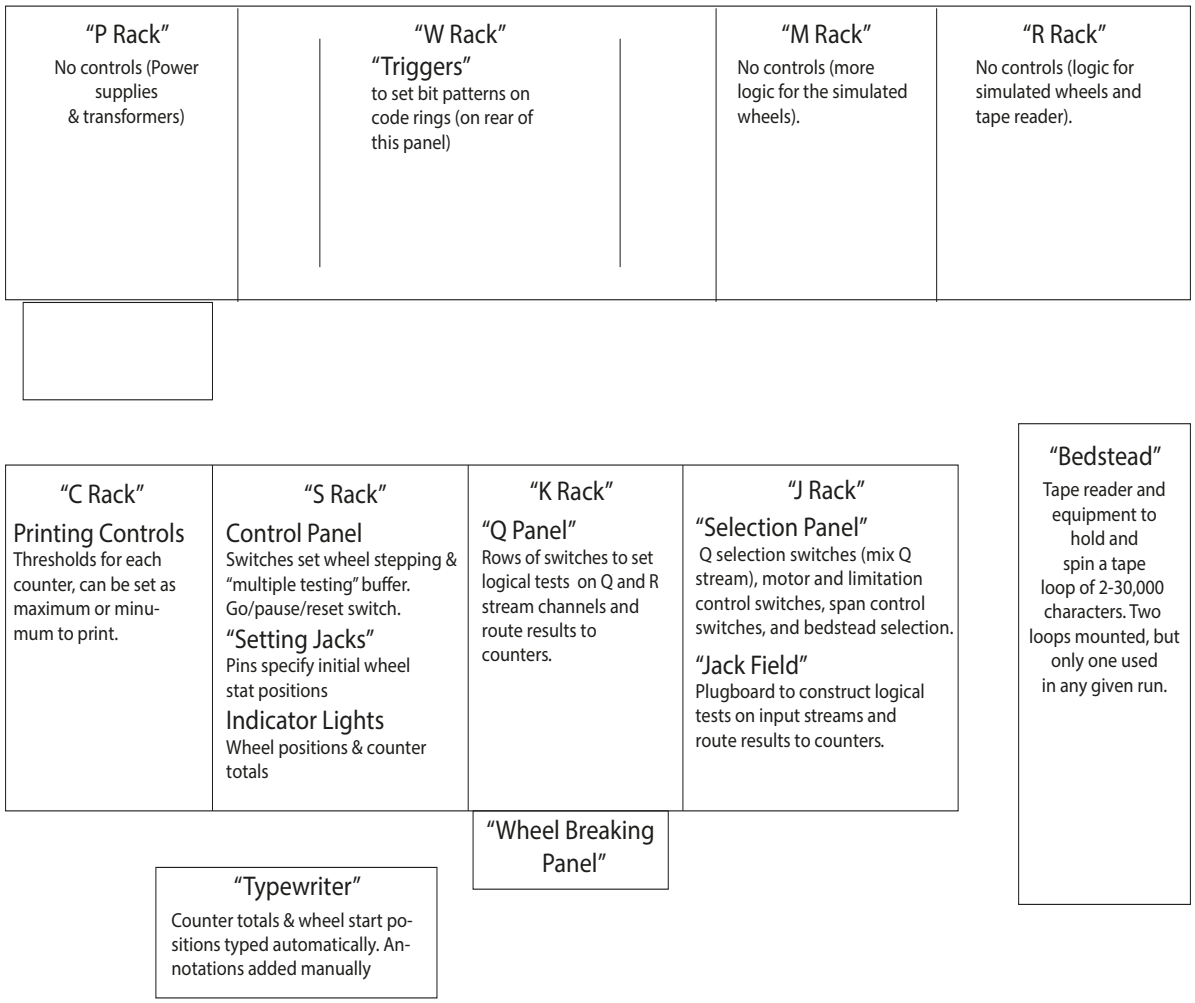


Fig. 8: Physical layout of Colossus. The physical layout diagram focuses on the controls spread over the various racks. The basic three-part architecture introduced with Heath Robinson can still be discerned in the physical layout of Colossus. Essentially the reader unit, now including electronic generation of the code wheel signals, was on the back row (racks W, M and R) plus the duplicate tapes and readers next to the "bedstead." Racks S, K and most of Rack J formed the combining unit. The counters, on Rack C, controlled the typewriter.After Tony Sale.[14]

3 Colossus Controls

3.1 Physical Layout Overview

Despite its name Colossus was fairly compact compared to some other machines designed during the war, such as Bell Laboratories' gigantic codebreaking machine Madame X, or the University of Pennsylvania's ENIAC. Electronic components and switches were spread across eight racks, arranged in two rows. The tape reader relied on an elaborate system of pulleys to handle tapes of different length read at high speed. These were mounted in a large frame known as a "bedstead."

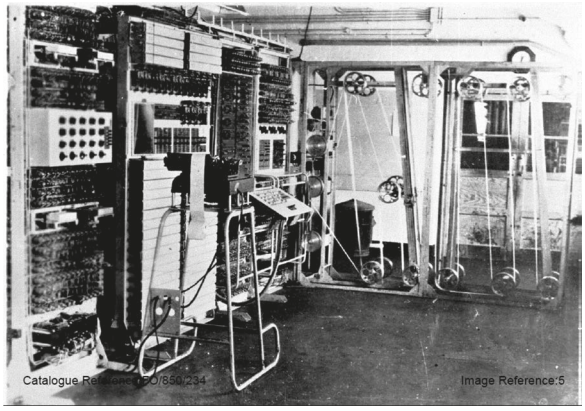


Fig. 7: This photograph of an original Colossus shows, from left to right, Rack C, Rack S, the typewriter (foreground), Rack K (with the wheel breaking panel sloping out from it), Rack J, and the "bedstead" tape system. National Archives FO 850/234 ("Annotated photographs of the COLOSSUS Electronic Digital Computer").

14 Anthony E Sale, "The Rebuilding of Colossus at Bletchley Park," *IEEE Annals of the History of Computing* 27, no. 3 (July-September 2005):61–69, page 66.

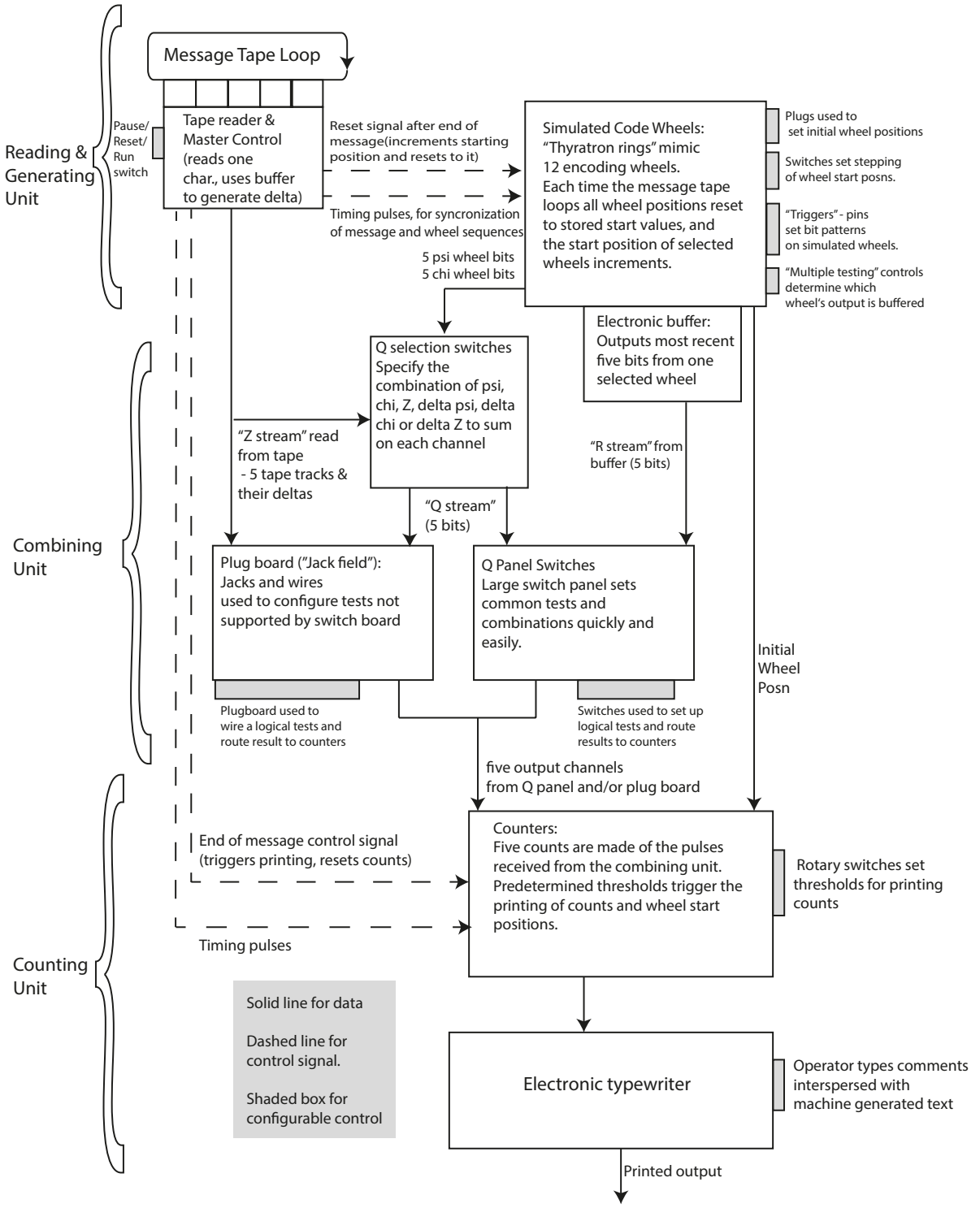


Fig. 9: Logical architecture of Colossus 2, focusing on controls and control mechanisms.

Figure 9 gives an idea of how the simple structure used for Heath Robinson grew more complex with the later Colossus machines. It takes some liberty with the actual positioning of controls, as the three units were no longer so neatly separated. For example, the master control switches are shown here as part of the reader because all control functions were driven by sensors in the tape reader. Likewise, the switches to control the electronic buffer are shown as part of the reader unit because they control the signals being generated. In fact, both the master control switch and the buffer controls were on the same control panel (part of Rack S). In the discussion that follows we group controls according to this logical separation, but note for each the rack on which they are physically located.

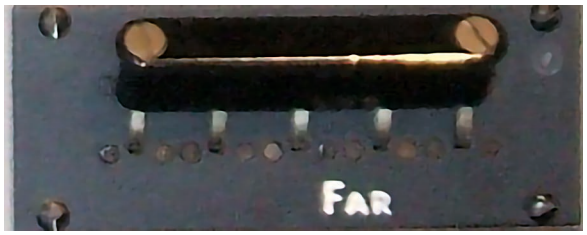


Fig. 10: Bedstead selector switch (TNMOC reconstruction). Photograph: Mark Priestley.

Fig. 12: Span control switches (TNMOC reconstruction). Photograph: Mark Priestley.

Fig. 11: Span counters (TNMOC reconstruction). Photograph: Mark Priestley.

3.2 Reading and Generating Controls

3.2.1 Reading the tape

Each Colossus had two bedsteads, allowing one tape to be mounted while another was being read. A group of five switches (usually ganged together) on the selection panel on rack J allowed the operators to choose which bedstead ("near" or "far") the input came from (see Figure 10).

3.2.2 Span Counters

Spanning allowed operators to analyse only part of the message tape. This was controlled by the span counters" on rack J, shown in Figure 11. These are twelve 10-place switches, arranged in three groups each specifying a four-digit number. The two groups "Start counter" and "End of span" define the interval where the counts are recorded. Outside that interval, Colossus would supress signals to the counters. The final set of switches, labelled "Psi Start," controlled the point from which the psi wheels were motorized. These were normally set to 0000.

Initially the Colossi were built to handle a maximum tape length of 10,000 characters, so four-digit span controls were adequate. Some later machines had "long bedsteads" that could handle tapes up to 30,000 characters long. To enable spanning on these tapes, some "span control" switches were provided on the rack J selection panel, as shown in Figure 12, to provide a "rudimentary 5th decade." The three left-hand switches correspond to the "start counter," "end of span," and "psi start" switches, and added 10,000 or 20,000 to the total set on the regular span switches when thrown up or down.

3.2.3 Defining and Selecting Wheel Patterns (Triggers)

For reasons said to be rooted in a misunderstanding of electronics terminology, the controls used to set bit patterns for the simulated code wheels were known as "triggers." Most Colossus controls were placed on the front side of the front row of racks, but triggers were mostly placed on rack R on the rear row of racks, along with the power supplies and the electronics for the code rings.

Each trigger consisted of a series of sockets. Placing a U-shaped pin in a particular socket caused the simulated code wheel to output a 1 on its bitstream

Fig. 13: One of the "triggers." (TNMOC reconstruction). Photograph: Mark Priestley.

when the appropriate wheel position was reached. Otherwise the wheel output a 0. The photograph in Figure 13 is from the rebuilt Colossus.

Colossus provided several triggers for each code wheel. There were five triggers (named *a, b, c, d, e*) for each of the chi and psi wheels, and seven (*a, b, c, d, e, f, g*) for the μ wheels. The *trigger selection switches* on the selection panel (rack J) determined which trigger would define the patterns for each simulated wheel in a particular run (see Figure 14).[15]

Fig. 14: Trigger selection switches (TNMOC reconstruction). Photograph: Mark Priestley

Different code wheel patterns were used on different German radio links, so being able to store multiple patterns and switch between them would save setup time when switching between messages intercepted on different links. As bit patterns used on a particular link were changed, with increasing frequency as the war went on, the ability to store several patterns would also save time when switching between messages intercepted on different days.

15 The reconstructed Colossus does not have a full set of trigger controls, and the ones that are present are labelled differently from those on the original machines.

Additional switch positions *e'* and *g'* allowed the *e* trigger for the chi and psi code wheels and the *g* trigger for the motor wheels to be used in a special way, as "special" or "doubting" patterns, to be used in addition to an ordinary trigger. If these positions are selected, the impulses from the affected wheel were not used in the normal way (appearing on the Q panel, or motorizing), but appeared instead on the plug board.

The Colossus machines were eventually applied to wheel breaking as well as wheel setting, a task which required the operator to make frequent changes to bit patterns in search of a solution. Moving between the main triggers, on the rear of rack R, and the other controls would slow things down, which is why the "Wheel breaking panel" added to rack K, close to the other controls, included two additional trigger sets. These used different, easier to handle pins and were much more conveniently located. According to the *General Report on Tunny* (GRT 52(h), 314),[16] "Intolerable delays and mistakes during wheel-breaking were caused by the need for setting up pins at the back of Colossus and complaints finally extorted the wheel breaking panel on the front of some machines."

3.2.4 *Setting the Wheel Start Positions*

Once the wheel patterns were defined, the starting positions for the 12 simulated code wheels had to be set. This was done by putting plugs in the setting jacks on rack S, just below the control panel (see Figure 15). There are 12 jack strips, one for each simulated wheel.

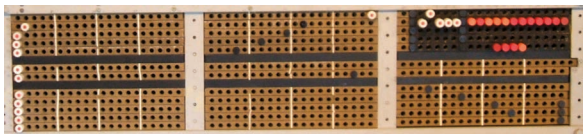


Fig. 15: Setting jacks (TNMOC reconstruction). Photograph: Mark Priestley.

16 We will be repeatedly citing the 1945 "General Report on Tunny" produced by members of the Bletchley Park staff (probably D. Michie, I. J. Good, and G. Timms) shortly after the end of the war. It was declassified decades later, in 2000, and released as TNA: HW 25/4 and HW 25/5. A scan of the original is available online at http://www.alanturing.net/turing_archive/archive/index/tunnyreportindex.html. The report has recently been published in an annotated version as 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). Our citations to "GRT" provide the both the section number, usable with the original report, and a page number for the Reeds et al. reprint.

3.2.5 Controlling Wheel Stepping

Every time the message on the tape finished, Colossus rest the position of each code wheel to a stored value known as its start position, or "setting". These values were held on electro-mechanical uniselectors. Most runs involved reading the message repeatedly to try out different combinations of wheel start positions. So Colossus could advance wheel start positions before reading the message again. This was known as "stepping" a wheel.

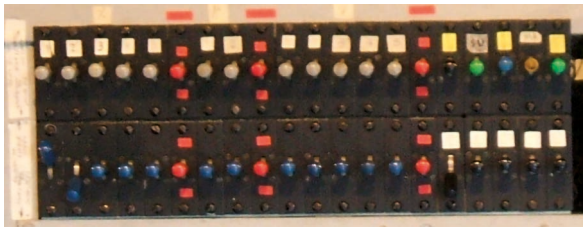


Fig. 16: Stepping was controlled by the stepping switches to the left of the control panel (the grey and blue switches in this picture). (TNMOC reconstruction). Photograph: Mark Priestley.

Many runs involved testing the tape against all the possible combinations of starting positions of two wheels. These were called "long runs." A wheel that stepped forward by one position each time the tape rotated was said to "step fast." In a long run, it was common for one wheel to step fast and the other slow, i. e. to step only when the fast wheel had come back to its original setting. These runs were set up on the blue switches visible in Figure 16: the image shows the first wheel is stepping slowly (switch up) and the second fast (switch down). The run would terminate when all the wheels returned to their original position, when a "repeat light" would be illuminated (it is unclear whether the operator then had to stop the machine manually, or whether it automatically cut out).

A short run only involved testing the positions of one wheel, and in this case the wheel would be set to step fast. Colossus was capable of running several short runs simultaneously, printing the results for up to five wheels. If more than one wheel was set to step fast on the blue switches, the run would finish only when all combinations had been tested. For simultaneously short runs, however, it is not necessary and would be wasteful of time to examine all possible wheel positions. The run can be stopped when the longest wheel reaches its starting position, as by then the shorter wheels will also have been stepped to all possible positions. In this case, the grey switch for the longest wheel would be set down, and the other wheels to be tested up. All those wheels will then step fast, and the repeat lamp will be illuminated when the longest wheel (with its grey switch down) reaches its starting position. (A single short run could be set up by putting either the blue or the grey switch for the wheel down.)

3.2.6 Multiple Testing

Multiple testing allowed Colossus to perform tests on five successive output bits from one of the simulated code wheels. As discussed previously, this was accomplished using a buffer switched to the appropriate code wheel (on later versions of Colossus any of the chi or psi wheels or one of the two motor wheels).[17] This had the effect of evaluating five combinations of wheel start settings simultaneously, with the result that the start position of the code wheel being buffered could be stepped by five positions, rather than one position as usual, each time Colossus reached the end of the message on its input tape.

The red switches interspersed between the setting switches (see Figure 16) controlled multiple testing. The wheel set for "multiple testing" could step fast or slow, but when it did step, it stepped by five positions rather than one.

3.3 Combining and Testing Controls

3.3.1 Mixing the Input Streams

As each bit position of the tape was read, the following inputs were available for testing:

- The five bits read from the five-channel input tape. This was called the Z stream.
- The five bits generated by the simulated chi wheels. This was called the χ stream.
- The five bits generated by the simulated psi wheels. This was called the ψ stream.
- If multiple testing was being used, the five remembered bits from the last five positions of the wheel on multiple test.

Before any tests were carried out, the Z, χ, and ψ streams were combined to form a single five-channel stream known as the Q stream. In principle, each of the five channels of the Q stream could be formed from a different combination of inputs. For each channel, the bit appearing in the Q stream was formed by combining up to three bits. For example, for the channel 1 of the Q stream the three sets of choices were as follows.

- Channel 1 from Z stream, deltas from channel 1 of the Z stream, or nothing.

17 "Multiple testing is provided for all wheels except μ_{61}, χ_5, ψ_5 were added later and have not been fitted to all Colossi. μ_{37} has its own switch: the others are in pairs, each pair sharing a three-way switch, viz. χ_1, χ_2, χ_3, χ_4; ψ_1, ψ_2, ψ_3, ψ_4, χ_5, ψ_5." (GRT 53L(g), 329).

– Channel 1 from χ stream (representing the first chi wheel), deltas from channel 1 of the χ stream, or nothing.
– Channel 1 from ψ stream (representing the first psi wheel), deltas from channel 1 of the ψ stream, or nothing.

These choices were made using the Q selection switches shown in Figure 17. These formed part of the selection panel on the J rack. Each switch could be toggled up to include the bit from the corresponding stream, or down to include the delta. Leaving a switch in the central neutral position meant that the input from that stream was ignored.

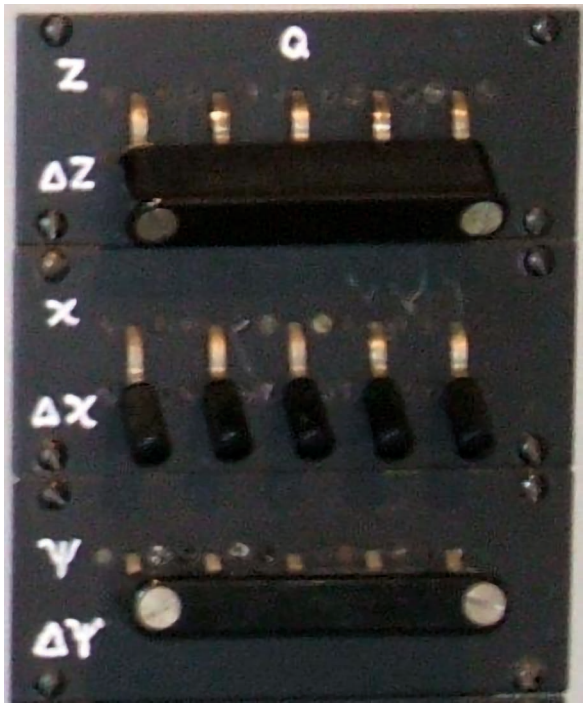


Fig. 17: Q selection switches (TNMOC reconstruction). Photograph: Mark Priestley.

In practice, the switches for channels 1 to 5 were usually moved together, something made easier by a crossbar shown here fitted to the five Z stream controls to set all of them to the delta Z position. In cases like this, the settings could be described by a simple equation. The illustration above shows that each channel of the Q stream is formed by combining the deltas of the corresponding channels of the Z and χ streams. This can be described by the equation

$$Q = \Delta Z + \Delta\chi.$$

The selected bits were combined with XOR operations, often referred to at the time as "addition modulo 2" (hence the + sign in the equation).

We think of this process as akin to the mixing together of different channels accomplished on an audio control board, in a recording studio or live performance, though of course the combination of input streams here took place digitally. This "mixing" took place in two stages. First, the Q stream was formed as described above, and then the bits derived from multiple testing were handled. Multiple testing was supported by a five-bit buffer that held the current and four previous bits generated by a wheel. These bits were not simply added to the Q stream, however; instead, they were combined with the other bits coming into the "mixer" in the way defined by the Q selection switches, giving rise to five new bits R_1, R_2, R_3, R_4, and R_5. Thus if $Q = \Delta Z + \Delta\chi$, as above, and χ_1 was being multiply tested,

$R_1 = \Delta Z_1$ (present) + $\Delta\chi_1$ (present)
$R_2 = \Delta Z_1$ (present) + $\Delta\chi_1$ (1 back)
$R_3 = \Delta Z_1$ (present) + $\Delta\chi_1$ (2 back)
$R_4 = \Delta Z_1$ (present) + $\Delta\chi_1$ (3 back)
$R_5 = \Delta Z_1$ (present) + $\Delta\chi_1$ (4 back)[18]

In other words, R_1 is the same as Q_1, R_2 is the same as the Q_1 that would have been generated at the previous tape position if multiple testing had not been in use, and so on.

3.3.2 Plug Panel

The later Colossi had two functional units which allowed the properties of the Q and R bitstreams to be tested and counted. The first was the jackfield, or plug panel, positioned on rack J (see Figure 18). This allowed great flexibility in the tests that could be set up, but also required them to be set up from scratch. As experience was gained, it appeared that much of the work the Colossi involved the repeated applica-

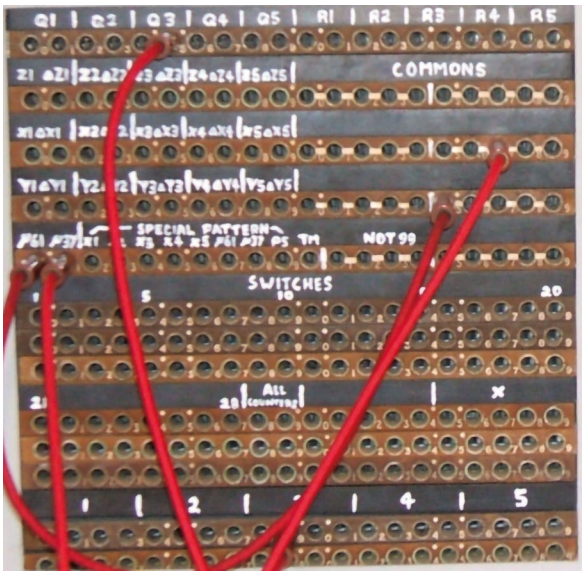


Fig. 18: Plug panel, sometimes called the "jack field" (TNMOC reconstruction). Photograph: Mark Priestley.

18 GRT 53L(c), 328.

tion of a quite small set of tests. From Colossus 2 onwards, the machines were fitted with a special panel that allowed tests of these types to be set up quickly by flipping switches rather than by plugging wires (this panel is described in the next section). The *General Report on Tunny* described some example uses of the plug board, including a particular wheel breaking run "normally done by plugging" (GRT 53K, 326–8).

The plug panel is divided into three broad areas.

1. The top five rows represent the input, i. e. the bitstreams coming from the reading and generating parts of Colossus;
2. the middle six rows (two groups of three) allow bits to be combined;
3. and the final two rows send output bitstreams to the counters.

The first five rows of the plug panel largely contained the outputs from various bitstreams. These included the Q and R streams available on the Q panel, the individual input channels of the message tape, and all wheels (these weren't available on the Q panel) and various special patterns.

The "common" areas in rows 2–5 (six groups of five plugs) allowed bitstreams to be duplicated. The Q_1 bitstream, for example, was available at the two plugs at the top left of the panel. If for a complex test this bitstream had to be used more than twice, one of these plugs could be wired to a common plug, and the bitstream would then be available at the other four plugs in the same field.

The next six rows included the "addition field" (labelled "SWITCHES" in the reconstructed machine shown in the photograph; 28 columns of three plugs), which provided the basic modulo 2 addition (XOR) operation. Multiple tests could be set up simultaneously here. Inputs were plugged into the top two rows, and the results taken from the bottom row. It was possible to sum more than two bitstreams in one operation.

Towards the right of rows 9–11 are the "start units" that allowed "constant" dots or crosses to be added into a test. These could be used to construct other logical combinations. The *General Report on Tunny* describes a combined use of the start and addition fields: "To plug any impulse [bit] to equal a *cross*, add a cross and plug normally [on the addition field]" (GRT 53K(i), 327). Adding a cross to a bit produces its opposite. In effect, this allows the Boolean NOT operator to be plugged.

The bottom two rows (and the "all counters" jacks on rows 9–11) allowed the results of the tests to be sent to the counters. More than one test could be sent to the same counter, providing a basic Boolean OR combination.

3.3.3 Q Panel

Flipping switches on the Q panel, found on rack K, accomplished most of what had been done by routing wires on the combining unit of the first Colossus: select inputs of interest, apply logical operations to them, and route the results to particular counters.[19] Setting switches was quicker than plugging wires, and because the switch panel was designed with common Colossus practices in mind one switch flip often substituted for several wire placements. The historic photograph in Figure 19 shows the scale of the Q panel, while the schematic in Figure 20 shows the layout of its switches. Conditions on specific channels are set on rows of switches like the one shown in Figure 21.

The five black switches on the left correspond to the five input channels of the Q stream emerging from the mixer. Each can be switched up for false (a dot) or down for true (a cross). If the switch is set to

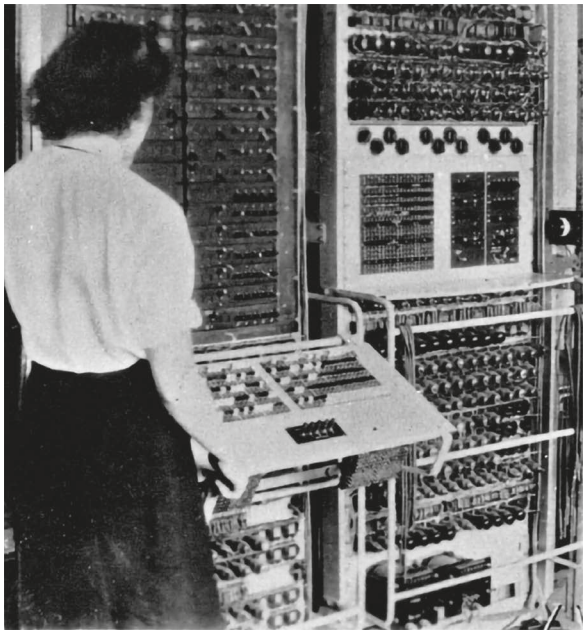


Fig. 19: The Q panel, partially obscured by the head of operator Dorothy Du Boisson, was by far the largest control panel on Colossus. National Archives FO 850/234 ("Annotated photographs of the COLOSSUS Electronic Digital Computer").

19 Evidence here is contradictory. Horwood did not mention any switching mechanism for the first Colossus, but the General Report on Tunny reported that "runs of the form $i + j = \bullet$ [i. e. chi wheel setting runs] could be done by switching except in the fifth counter. Most other runs required plugging, though there was a single set of five dot and cross switches for 'all counters.'" (GRT 52(e), 313). If this is true then the first Colossus had, or later acquired, a switch panel of some more limited kind.

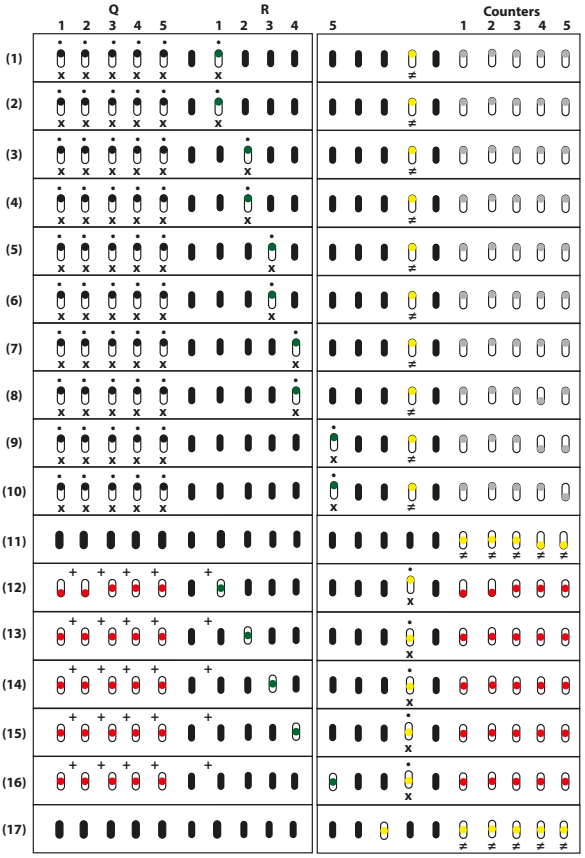
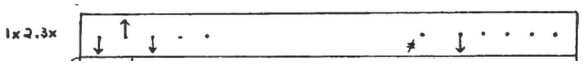


Fig. 20: A schematic depiction of the Q panel. Switch colors are taken from the TNMOC rebuild, though some labels present on the original machines have been added. The row numbers in parenthesis on the left are not present on the rebuild or the original machines but are added here and in the later setup diagrams for comprehensibility.

the middle, neutral, position the corresponding impulse is simply ignored. This example[20]



is therefore equivalent to the Boolean condition:

$$(Q_1{=}\times)\wedge(Q_2{=}\bullet)\wedge(Q_3{=}\times).$$

This could be represented more compactly using modern conventions. In the abstract language of bitstreams, which we use elsewhere in this report, 0 represents a dot (or a logical FALSE) and 1 represents

Fig. 21: The top row of the Q panel (TNMOC reconstruction). Photograph: Mark Priestley.

a cross (or a logical TRUE). If one also adopts those conventions then this is equivalent to

$$Q_1\wedge\neg Q_2\wedge Q_3.$$

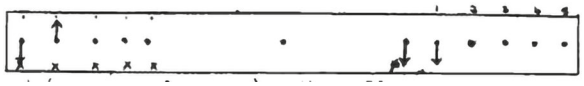
However, we are sticking with the longer representation in this section as it more clearly corresponds to the labelling of the Colossus control panels.

The five switches on the right of the row correspond to the five counters. When the condition set on the row is met the selected counter (or counters) increment. Thus the example above would form a count of the positions in the input bitstream satisfying 1x2•3x to be made in counter 1.

The yellow switch to the left of the counter negates the test set on the Q and R switches. This enables conditions of the form "X or Y" to be set up in the equivalent form "not (not X and not Y)." For example, the test

$$(Q_1{=}\bullet)\vee(Q_2{=}\times)$$

was diagrammed as follows (GRT 53J(d), 323):



These switch settings encode the equivalent Boolean formula

$$\neg((Q_1{=}\times)\wedge(Q_2{=}\bullet))$$

and the matching inputs are counted in counter 1.

If more than one row sends its results to the same counter, the counter is only incremented if both conditions are true. Another logical function was provided by row 11 of the Q panel (see Figure 22).

Fig. 22: The switches on the 11th row of the Q panel (TNMOC reconstruction). Photograph: Mark Priestley.

There are five switches, one for each counter. Depressing a switch reverses the effect of the tests applying to that counter, and so provides a way of negating a whole expression.

These facilities could be combined to test quite complex conditions. For example, one way to set the condition 1=2=4 is to express it as:

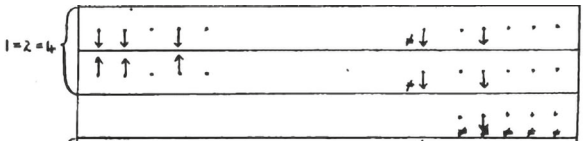
$$\big((Q_1{=}\times)\wedge(Q_2{=}\times)\wedge(Q_4{=}\times)\big)\vee\big((Q_1{=}\bullet)\wedge(Q_2{=}\bullet)\wedge(Q_4{=}\bullet)\big)$$

20 This example is taken from (GRT 53J(f), 325). The authors used a schematic representation of the Q panel, in which arrows pointing up or down were used to indicate the switches that had been set. Switches left in the center position were represented as dots.

and then transform it to the equivalent:

$$\neg\big(\neg((Q_1{=}\times)\wedge(Q_2{=}\times)\wedge(Q_4{=}\times))\wedge\neg((Q_1{=}\bullet)\wedge(Q_2{=}\bullet)\wedge(Q_4{=}\bullet))\big).$$

This can be switched and counted in counter 2 as follows (GRT 53J(f), 325).



At the left of the first row, three switches test for $(Q_1{=}\times)\wedge(Q_2{=}\times)\wedge(Q_4{=}\times)$; at the right of the row this test is negated and sent to counter 2. Similarly, the second row sets up a test for $(Q_1{=}\bullet)\wedge(Q_2{=}\bullet)\wedge(Q_4{=}\bullet)$, negates it and sends it to counter 2. At this point, counter two is set up with the test

$$\neg((Q_1{=}\times)\wedge(Q_2{=}\times)\wedge(Q_4{=}\times))\wedge\neg((Q_1{=}\bullet)\wedge(Q_2{=}\bullet)\wedge(Q_4{=}\bullet)).$$

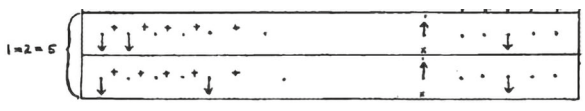
The switch set on the final row negates this whole condition.

The rows 12–16 of the Q panel allowed selected impulses to be XORed, or "added modulo 2", together. Figure 23 shows the last such row (row 16, containing mostly red switches), just above row 17 at the very bottom of the Q panel containing more "not" switches (yellow).

Fig. 23: The bottom two rows of the Q panel (TNMOC reconstruction). Photograph: Mark Priestley.

Flipping down one of the five red switches on the left added the value of the corresponding channel into the sum. The yellow switch to the left of the counter selection switches let the operator choose whether the sum was tested equal to a dot or a cross. The basic 1+2/ run (this notation is explained below) meant $(Q_1{+}Q_2{=}\bullet)$, and could therefore be set up on one of these rows by depressing the Q_1 and Q_2 switches, and setting the yellow switch to the left of the counter switches to •.

A more complex use of multiple rows provides an alternative way of testing whether three input bits are equal (GRT 53J(f), 325).



These two rows encode the condition

$$(Q_1{+}Q_2{=}\bullet)\wedge(Q_1{+}Q_5{=}\bullet).$$

As two bits are equal if and only if they XOR to dot, this is equivalent to

$$(Q_1{=}Q_2)\wedge(Q_1{=}Q_5).$$

Each of the 15 test rows on the Q panel included a switch for each channel in the Q stream. When a multiple testing run was taking place, the input to the Q panel included five bits from the wheel being multiply tested. These could be included in tests using the R switches, which behaved in the same way as the Q switches. An example of switching using the R switches can be found in section 4.3 below.

The five yellow switches to the right of row 17, labelled ≠, negated conditions, just like the similarly labelled switches in row 11. However, the switches in row 17 "negate whole columns, not merely the lower part of the panel; in particular they negate the upper row of 'not' switches" (GRT 53J(e), 324).

3.4 Counting Controls and Output

3.4.1 The Display

Rack S included a small panel of display lights (see Figure 24):

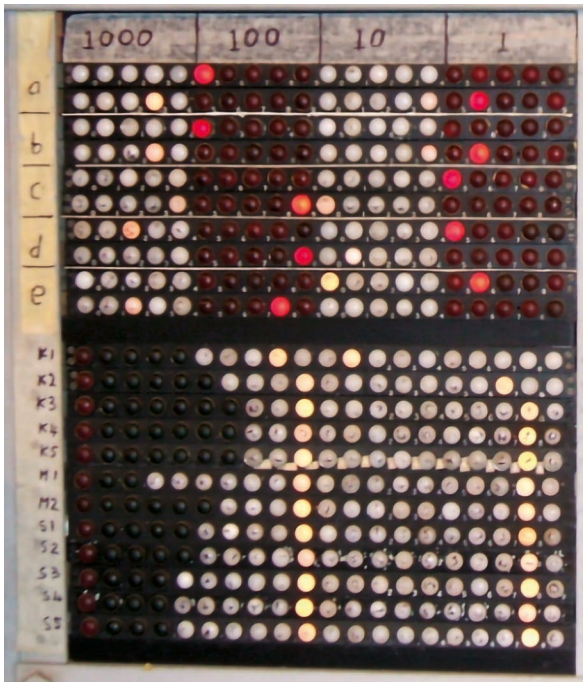


Fig. 24: Display lights on Rack S (TNMOC reconstruction). Photograph: Mark Priestley.

The five upper rows, labelled a, b, c, d, e, give the current contents of the five counters. The use of letters a–e matched the identifiers used on Colossus printouts, though elsewhere the counters were referred to with numbers. One light was illuminated in each block, corresponding to the units, tens, hundreds and

thousands digit of the decimal representation of the counter contents. In Figure 24, the contents of counter e, for example, is 2,198.

The twelve lower rows show the current positions of the simulated wheels. (K was often used as a typeable or more easily written alternative to χ, S for ψ, and M for μ). In each row, one of the rightmost ten lights is illuminated to give a decimal units value. The remaining lights, on the left, give the decimal tens value. Because the wheels were of different lengths the number of lights varies by row. For example, the fifth chi wheel was only 23 bits long, and so needed only three lights to represent the tens digit (0, 1 or 2). The longest wheel, the first motor wheel, was 61 bits long and so needed seven positions (the tens digit could be 0, 1, 2, 3, 4, 5, or 6). Thus in the example the first row ("K1") shows 18, the second row ("K2") shows 02, and the other rows all show 01 – indicating that these wheels had a start position of 1 and had not been stepped.

3.4.2 Threshold and Printer Controls

The basic function of the printer was to print the values stored in the counter after each rotation of the tape. This functionality could be configured in two ways, first by the "set totals" switches, and second by a small jackfield that controlled which counters were printed.

The set totals switches are found on rack C (see Figure 25). The switches change nothing in how or what the counters count, which is a function of the inputs being fed into each counter from the plug board and Q panel. What these switches do is to automatically suppress the printing of counts obtained with particular wheel settings if those counts fall well within the range that would be expected from random variation.

The large, rotary switches specify a five-digit decimal number. The smaller switches specify, for each counter, whether the threshold specified is a minimum value (below which the printing of counts should be suppressed) or a maximum value (above which the printing of counts should be suppressed). These automated much of the work of the human operators of Heath Robinson, who were expected to scan the counters and write down counts and wheel start positions that exceeded a certain level. Operators would need to look up the appropriate thresholds for a particular run, which would be determined according to the length of the message tape being analyzed and the proportion of positive results that would be expected by chance.

A small 5 x 12 jackfield, visible to the top-right of the setting jacks in Figure 15 above, further controlled the printed output. When multiple tests were being carried out simultaneously, there is a mapping between wheels and counters: for example, the counts for different positions of K1 might be recorded in counter A, and those for K3 in counter B. When a count is printed, we only want to see the setting for the wheel associated with that counter. Inserting a plug in the relevant position in the jackfield (5 x 12 = 5 counters x 12 wheels) included the setting in printing output.

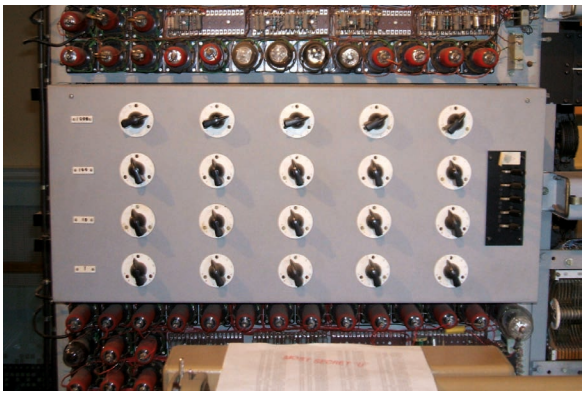


Fig. 25: Threshold controls set the criteria used to selectively suppress printing of totals (TNOC reconstruction). Photograph: Mark Priestley.

3.5 General Run Control

A small set of control switches on rack S controlled a somewhat miscellaneous selection of operations. These are the rightmost switches visible in Figure 16 above. The arrangement and labelling of switches differs between the rebuild and the historic photographs of Colossus, and also varied between the original Colossus machines themselves. The report describes the switches as follows:

- PMH – Print Main Headings. Causes the printer to output two rows listing the settings of each wheel, as shown in the top two rows of Figure 34 below.
- SET – Set wheels. Resets the wheels to their initial positions, as defined by the setting jacks.
- RESET – Reset counters. "Clears all scores in storage: in particular if PCO is in use it allows stepping to be resumed" (GRT 53G(j), 322).
- MAS – Master switch. "Unless this switch is thrown, Colossus can neither count nor step. It is however possible to set wheels and to reset counters" (GRT 53N, 334). Note that when LC (see below) is set, however, it is possible to count.
- REC – Rectangle. Two active positions: "print scores" and "normal" (GRT 53M(f), 333).
- PCO – Printer cut-out. "Prevents Colossus from sending impulses to the printer, so that stepping ceases" (GRT 53G(i), 322).
- L c/o – Lamp cut-out. "Cuts out the 'settings' lamps" on the display (GRT 53G(d), 321).
- LC – Letter Count. "Is for making counts at fixed settings. It stops the machine after printing a batch of scores" (GRT 53G(h), 322).

– KL – Cancel Lights. "Extinguishes the 'settings' lamps [on the display] when all scores in storage have been printed" (GRT 53G(d), 321).

Some of these switches were used to control the starting and ending of runs. Once a message tape had been loaded and the tape reader started, the tape would keep revolving and the switches determined when the recording of counts would begin.

4 Typical Colossus Configurations

The Colossi were principally used to make counts of certain properties of encrypted messages, thus providing the raw data used for the Newmanry's "statistical" attack on Tunny (as opposed to the Testery's "linguistic" methods).

4.1 The Colossus Run Notation

The unit of Colossus work was the "run," defined as an operation "which, when started, the machine can complete without human operation" (GRT 71, 420). Runs were set up by switching and plugging the controls described in the previous section into particular configurations. A full taxonomy of Colossus runs will be given in a later report. Here, we describe how Colossus was configured for some typical runs in order to give a sense of the range of its capabilities and the extent to which it could be "programmed."

The notation used to define different runs followed the following syntax (slightly simplified):

– Simple terms **n** ::= 1 | 2 | 3 | 4 | 5 denote the 5 channels in the input stream.
– Sums **s** ::= **n** + **n** | **s** + **n** denotes the modulo-2 addition of bits from the corresponding channels.
– Terms **t** ::= **n** | **(s)**. Parentheses round sums are sometimes omitted.
– Atomic conditions **a** ::= **t** | **t**· | **t**x. **t** and **t**· are true if **t** is dot, **t**x is true if **t** is cross
– Equality conditions **e** ::= **t** = **t** | **e** = **t** are true if the terms are all equal, i. e. dot or all cross.
– Conditions **c** ::= **a** | **e** | **ca** | **ce** are non-empty sequences of atomic and equality conditions, denoting their conjunction.

Colossus runs were usually specified by a condition containing at most one slash symbol /. This was explained by US liaison officer Albert Small as follows: "The slants have been placed in to show what 'runs' are usually made on Colossus. Thus 4=5/=1=2 means that given 1 and 2, this is a good way to find 4 and 5 simultaneously. But 4=/5=1=2 and 5=/4=1=2 are just as plausible to run under proper circumstances."[21]

21 Albert W. Small, "Special Fish Report" (1 December 1944). NARA-HCC b579, p. 7.

The numbers to the left of the slash are the channels whose starting positions the run is attempting to find, those to the right are those assumed known. A channel is "known" if the starting position of the corresponding wheel is assumed to be known; these positions are recorded on the setting jack panel before the run starts. A short run has one channel number to the left of the slash, a long run has two. Runs mentioned by Small include:

(1p2)·/	4=5/=1=2	4x5x/1x2x	(4p5)·/1x2x	(3p4)x/1x2x
(4p5)·/(1p2)	(3p4)x/2x	(2p5)·/	(4p5)·/	(3p4)x/(1p2)·
5·/2·(1p4)·	(3p4)x/	5=/1=2	4·5·/1·2·	(3p/1)·2·
5·/1·2·	(2p4)·/	3x/1x2·	4=/1=2	(3p/1p2)·
4x/1x2x	(1p3)·/	5x/1x2x	3x/1x2x	(4p5)·/1·2·
4·/1·2·	(1p5)·/	3=/1=2	(1p4)·/	(3p4)x/1·2·
3x/1x2x	3x/1x4·	(3p/2)x1x4x	3·/1·2·5·	3·/1·2·4·5·

Removing the slash from a run specification yields the condition that should be set up for that run.

4.2 Configuration for the 1+2/. Run

This was the basic run used to set the χ_1 and χ_2 wheels for a message where the chi wheel patterns were already known. Derived from Tutte's original insight, it was the task that the original Heath Robinson machine was built to carry out. It is a typical example of what was called a "long run," one which made counts for all possible settings of two wheels. In practice, on Colossus 2 and later machines, this run would always have been carried out using multiple testing. In the hope of making things clearer, we first describe how this run would be set up without multiple testing.

As the length of the χ_1 and χ_2 wheels was 41 and 31, respectively, the message tape would be cycled $41 \times 31 = 1271$ times, with the χ_1 and χ_2 wheels set to a different combination of starting positions on each cycle. On each cycle, the first two bit channels in the message were combined with the corresponding bits of the χ_1 and χ_2 wheels, and the count incremented for every character position in which the resulting bit was 0 ("dot"). The operator would determine the value at which a count gave statistically significant information, and configure Colossus to print the wheel settings and the count at those values only.

The Colossus controls were set up as shown in Table 1, which uses a format similar to that found in (GRT 24B, 116).

As we mentioned earlier, any combination of Colossus settings for combining inputs and counting them could be represented as a truth table with deltas and current values from the tapes and wheels as the only inputs (because the combining hardware had no access to state information retained from one

Tab. 1: Configuration of Colossus controls for the 1+2/. run.

Run type	LC switch not thrown
Spanning	0000 – message end (or limited if a slide detected)
Select wheel patterns	χ_1 = a, χ_2 = a (or other trigger)
Wheel-breaking panel switches	N/A
Wheel setting jacks	χ_1 = 1, χ_2 = 1
Wheel stepping switches	χ_1 fast (lower switch down) to control χ_2 slow (lower switch up)
Q selection switches	Q = $\Delta Z + \Delta\chi$ (or $Z + \chi$ if the tape and triggers were set up with Δs)
Multiple testing	N/A
Q panel	See Figure 26
Jackfield	Not used
Print setting jacks	Include settings for χ_1 and χ_2 on printer A
Set totals	Determined according to message length
Control printing	Printer A set to > (only print counts that exceed set total)

Tab. 2: Truth table showing the connection to counter outputs and logical inputs to the "combining unit" of Colossus.

Inputs[22]				Output
$\Delta\chi_1$	$\Delta\chi_2$	ΔZ_1	ΔZ_2	Counter A
1	1	1	1	1
1	1	1	0	0
1	1	0	1	0
1	1	0	0	1
1	0	1	1	0
1	0	1	0	1
1	0	0	1	1
1	0	0	0	0
0	1	1	1	0
0	1	1	0	1
0	1	0	1	1
0	1	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

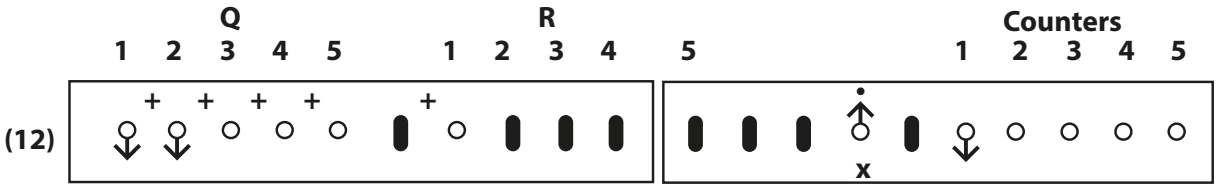


Fig. 26: Q panel configuration for 1+2/. in a hybrid format, imposing the arrows used at the time on a labelled version of the Q panel for easier comprehension. Only row 12 is used for this configuration – all switches on other rows are left unset.

bit position to the next). In this case, the table would be as shown in Table 2.

Note that the counter here is switched to tally dots. Using the convention that a dot is 0, this means the output to the counter is effectively NOT ($\Delta\chi_1$ XOR $\Delta\chi_2$ XOR ΔZ_1 XOR ΔZ_2). We will not show the tables for the other sample configurations as they are substantially larger. Configurations where more than one counter is used would require either a single conventional truth table for each counter, or an unconventional truth table with multiple output columns (one for each counter used).

22 We should clarify that these are the inputs to the conceptual "combining unit" of Colossus, not to the actual Q panel switches. In the physical Colossus the delta inputs would already have been combined. For conceptual purity one could construct a larger table, showing bit values rather than deltas for each input, but we feel that this one is adequate to support our point that any possible configuration of the logical conditions set up on Colossus to increment a counter could be represented as a single truth table.

4.3 Configuration for the 1+2/. Run with Multiple Testing

Multiple testing allows five characters of one wheel to be examined at the same time. To convert the basic run to use multiple testing, the following changes to the basic set-up are required:

1. Define which wheel is going to be on multiple test (often χ_1)
2. Adjust the switching on the Q panel to use the R inputs and all five counters.
3. Set up five counters to record five simultaneous counts.

```
BR 3407 0450/24/2 WD 23/2 COL 7

T 6020

1P2/L R 3106 A 1553 ø 27.8 ST 1624
K1 K2
02 08 E 1631
02 20 E 1625
11 09 A 1633
16 07 A 1629
18 08 D 1629
19 31 C 1625
37 21 E 1637
40 28 B 1655      B 102   3.7ø
40 28 C 1655

SET K1 K2  40 28
```

Fig. 28: Output of a Colossus run 1+2/. with multiple testing (GRT 23D, 84). The row "K1 K2" identifies the two wheels being tested. It and the following 9 lines were printed automatically by Colossus; the remainder was typed by the operator. These rows contain the positions of the two wheels, a letter identifying a counter, and the count itself. "B 102 3.70" was entered by the operator to confirm the correct settings with a measure of statistical significance.

Tab. 3: Configuration of Colossus controls for the 1+2/. run with multiple testing.

Run type	LC switch not thrown
Spanning	0000 – message end (or limited if a slide detected)
Select wheel patterns	χ_1 = a, χ_2 = a (or other trigger)
Wheel-breaking panel switches	N/A
Wheel setting jacks	χ_1 = 1, χ_2 = 1
Wheel stepping switches	χ_1 fast (lower switch down) to control χ_2 slow (lower switch up)
Q selection switches	Q = $\Delta Z + \Delta\chi$
Multiple testing	χ_1 on multiple test
Q panel	See Figure 27. Five rows switching $R_i + Q_2$ = . and sending result to printer A – E
Jackfield	Not used
Print setting jacks	Include settings for χ_1 and χ_2 on all printers
Set totals	Determined according to message length
Control printing	All printers set to > (only print counts that exceed set total)

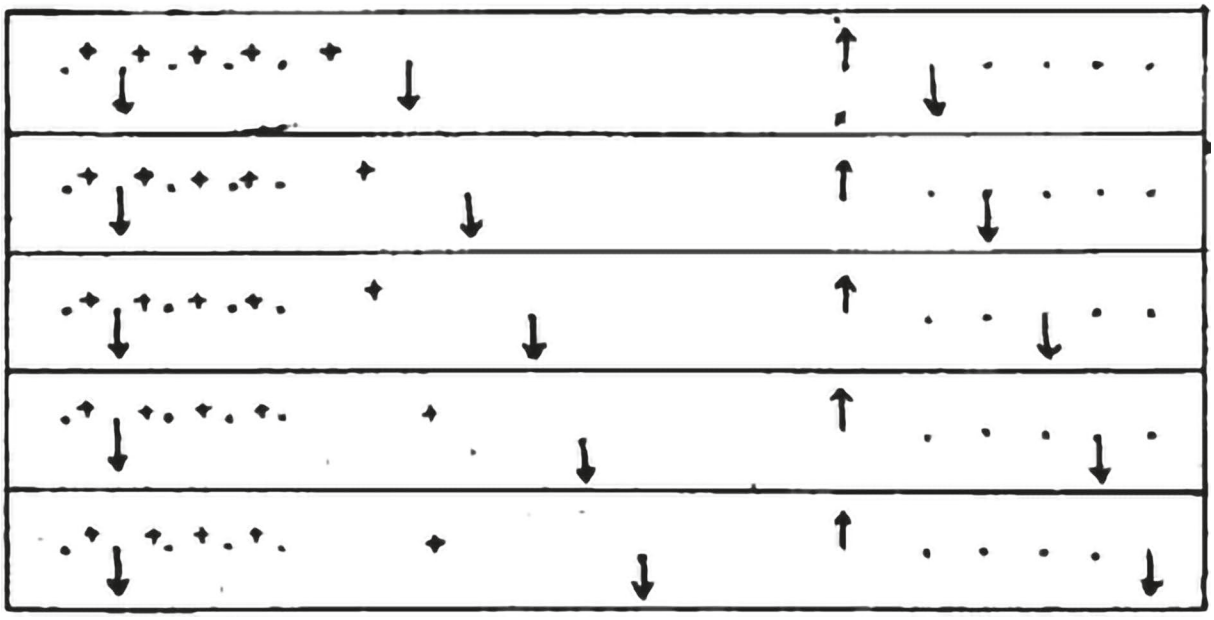


Fig. 27: Q panel configuration for the 1+2/. run, in the original diagram format (GRT 53L(l), 330).

4.4 Setting Up Colossus to Set Chi Wheels 3, 4, 5 With 1 and 2 Known

More than one short run could be set up to run at the same time. For example, if the settings of the χ_1 and χ_2 wheels were known, the other three chi wheels could be tested by running the three runs 4=/1=2, 5=/1=2, and 3=/1=2 simultaneously. These runs were common enough that they were known by the abbreviations C1, C2, and C4, respectively. Table 4 describes the set-up for such a run, and Figure 29 (see p. 26) shows the switching that would be required on the Q panel.

Figure 30 shows actual Colossus output from such a run:[23] The top row gives the settings of the two known wheels, the second row lists the runs being carried out, and the third row gives some message statistics, including the set total, 751. All these rows would have been typed by the operator. When the run started, Colossus printed the names of the stepping wheels ("k3 k4 k5") and each following row prints a count that is higher than the set total. The setting of the appropriate wheel is printed, followed by the printer identifier and the score. The run terminated

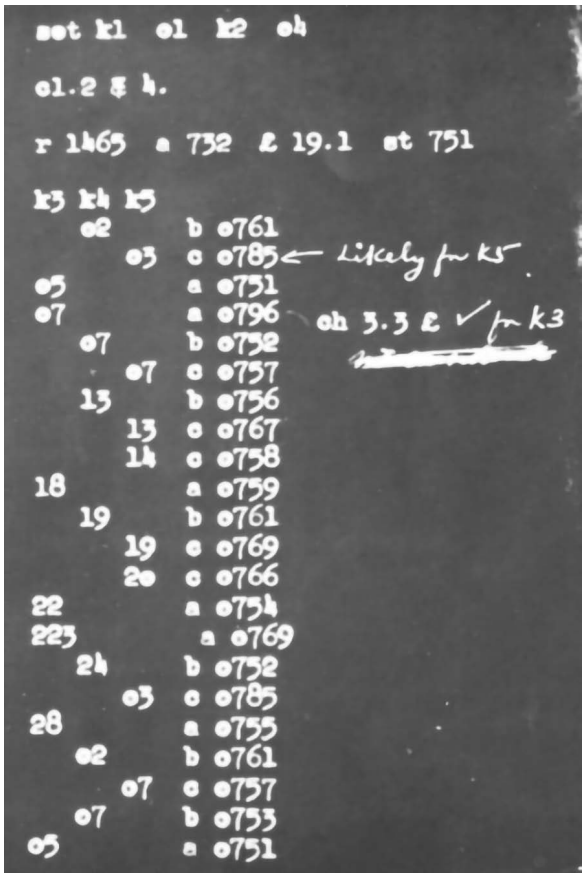


Fig. 30: Output from a Colossus run to set chi wheels 3, 4, 5 with 1 and 2 known.

Tab. 4: Configuration of Colossus controls to set chi wheels 3, 4, 5 with 1 and 2 known.

Run type	LC switch not thrown
Spanning	0000 – message end (or limited if a slide detected)
Select wheel patterns	χ_1 = a, χ_2 = a (or other trigger)
Wheel-breaking panel switches	N/A
Wheel setting jacks	χ_1 = 1, χ_2 = 1
Wheel stepping switches	χ_3 fast (upper switch down), χ_4 and χ_5 fast (upper switches up)
Q selection switches	Q = ΔZ + Δχ
Multiple testing	Not used
Q panel	See Figure 29
Jackfield	Not used
Print setting jacks	Insert plugs so that printer A counts only show settings for χ_3, printer B for χ_4 only, and printer C for χ_5 only.
Set totals	Determined according to message length
Control printing	All printers set to > (only print counts that exceed set total)

when the longest wheel, χ_3 (i. e. k3), returned to its starting position. This meant that some of the scores for χ_4 and χ_5 are printed twice; note that the score for the setting 07 of χ_4 is printed with the values 752 and 753. Presumably this is a Colossus run-time error or a printer glitch (like the repeated "2" in the χ_3 setting 23, printed as "223").

4.5 Setting up Colossus for a Character Counting Run

Frequently Colossus was used simply to count characters on a tape. Counts could be carried for any required subset of the teleprinter alphabet, down to single character counts. A frequent occurrence was to make a count for all 32 letters to check on the settings of the chi wheels. This was done in 8 separate runs, each counting 4 characters. (As there were five counters, it would have been possible to count 32 letters in seven runs. We believe that 8 runs were used because properties of the teleprinter code made the inter-run switching easier if they were done in batches of four.) Table 5 and Figure 31 show the configuration for the first run of a 32-letter count, which made a count for the letters /, 9, H, and T.

23 Small, "Special Fish Report", p 11.

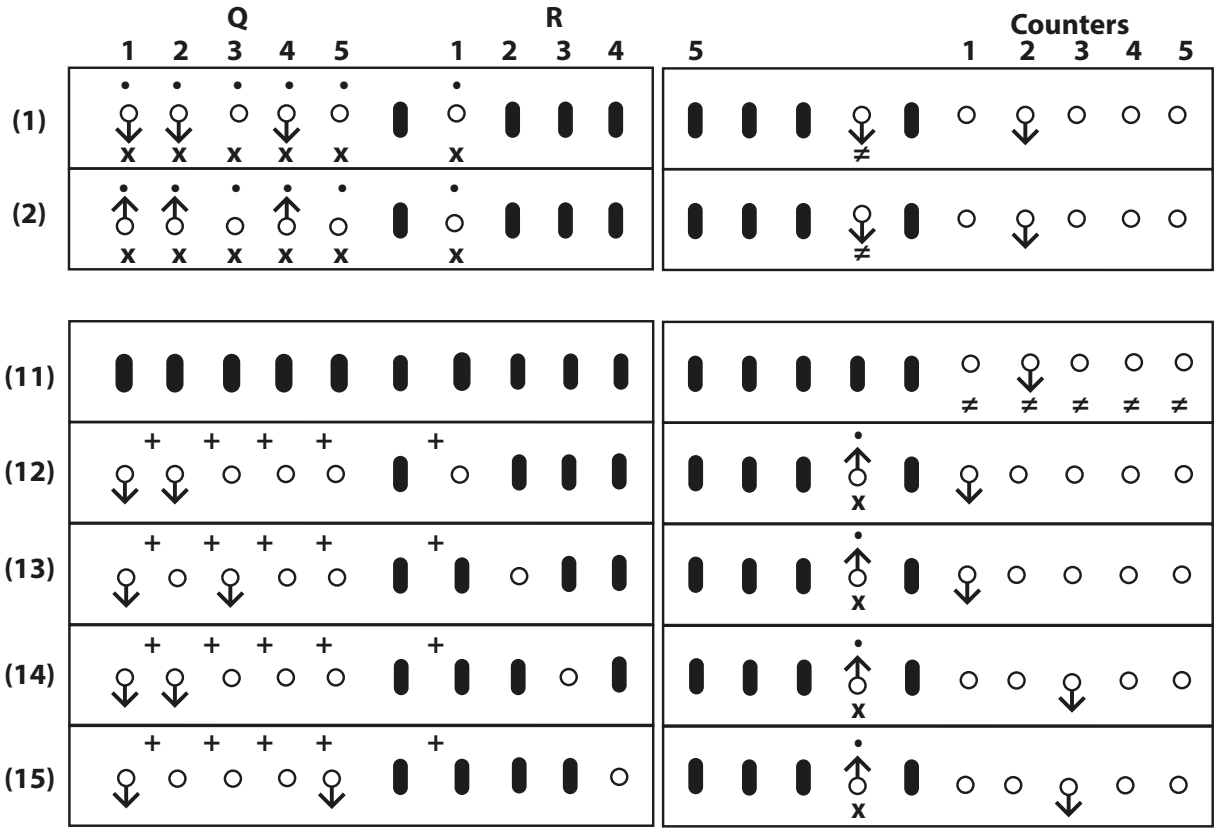


Fig. 29: Q panel configuration to set chi wheels 3, 4, 5 with 1 and 2 known. Rows 1, 2, and 11 implement the test C1. Using a different technique, rows 12 and 13 implement the test C4 and rows 14 and 15 implement the test C2. Both techniques follow (GRT 53J(f), 325).

Tab. 5: Configuration of Colossus controls for a character counting run.

Run type	LC switch thrown
Spanning	0000 – message end (or limited if a slide detected)
Select wheel patterns	All chi wheels set up on selected trigger
Wheel-breaking panel switches	N/A
Wheel setting jacks	All chi wheels set to start at 1
Wheel stepping switches	No stepping
Q selection switches	Q = ΔZ + Δχ
Multiple testing	Not used
Q panel	See Figure 31
Jackfield	Not used
Print setting jacks	Plug so that no settings are printed
Set totals	Set to 0, so all counts printed
Control printing	All printers set to > (only print counts that exceed set total)

Tab. 6: Configuration of Colossus controls for a decoding run.

Run type	LC switch thrown
Spanning	Span from n to n+1 to decode character n
Select wheel patterns	Set up all wheel patterns on selected trigger
Wheel-breaking panel switches	N/A
Wheel setting jacks	All wheels start at position 1
Wheel stepping switches	None set.
Q selection switches	Q = Z + χ + ψ (plaintext)
Multiple testing	Not used
Q panel	See Figure 33
Jackfield	Not used
Print setting jacks	No printout produced
Set totals	N/A
Control printing	N/A

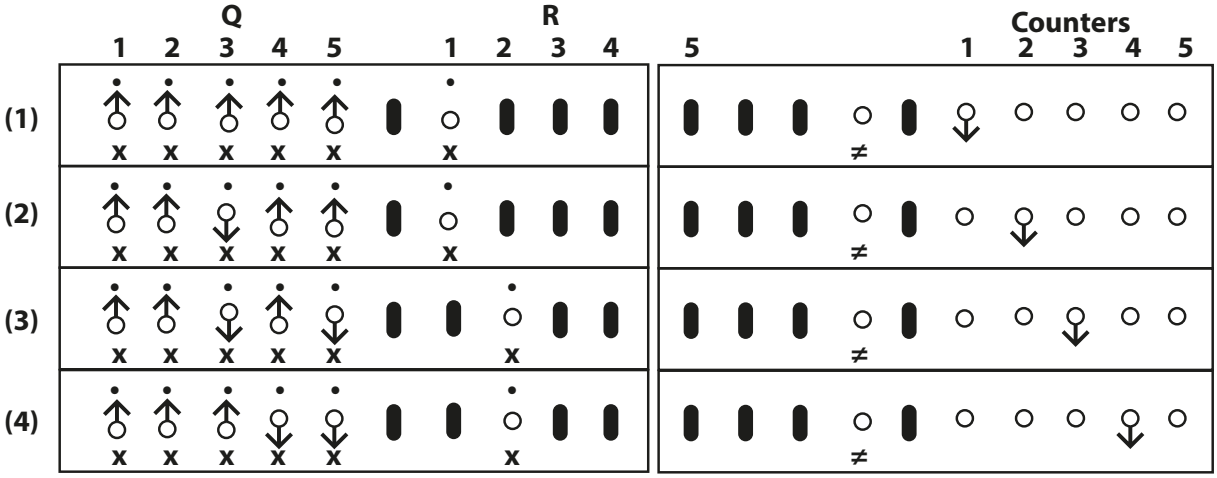


Fig. 31: Configuration of the Q panel for the first run of a Colossus character count.

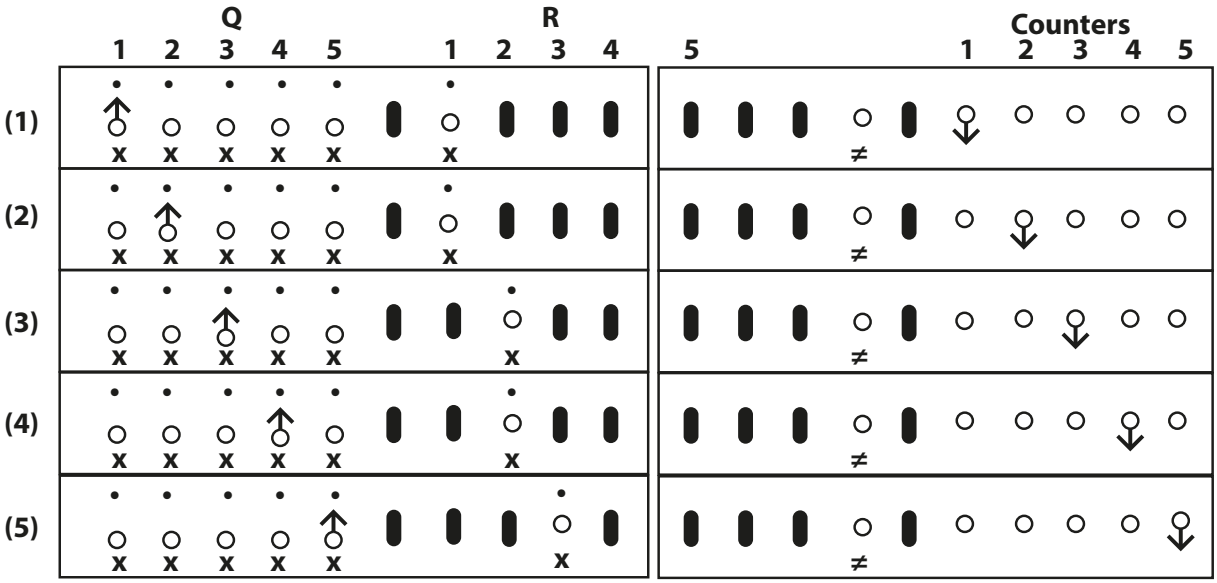


Fig. 33: Configuration of the Q panel for a Colossus decoding run.

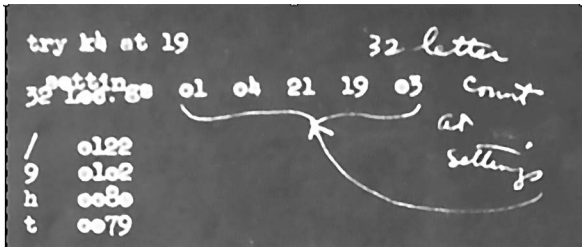


Fig. 32: Output from a Colossus character counting run. Small, "Special Fish Report", p 15.

An example of Colossus output for a run of this kind is shown in Figure 32. The wheel settings were entered by the operator. Colossus only printed the digits in the counts; the letters identifying the counts were probably added later for pedagogic purposes. Apparently the operators knew the letter sequence so well that they often didn't need to write them in.

4.6 Setting up Colossus for a Decoding Run

Colossus is sometimes described as a code-breaking machine, whereas in fact it was a machine that made counts of bitstreams. However, in December 1944 a technique was invented to use Colossus for decoding (GRT 74, 450). It was intolerably slow for general use, as it produced only one letter of plain text per tape cycle and required manual intervention to record the letter. Apparently it could take a Wren 1.5 hours to produce 41 characters of plain text. However, the technique was used as a sanity check on wheel settings before passing the message on to the Testery for manual decryption; often, only the first 9 letters of plain text would be produced. Table 6 and Figure 33 show the configuration details for a decoding run.

Each tape cycle looks at one letter, as defined by the span control. When the machine stops, each

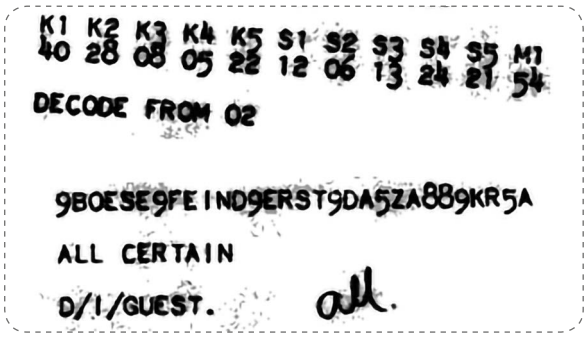


Fig. 34: Output from a Colossus decoding run. The top two rows (clipped here) were generated by the "Print Main Headings" switch on the control panel (see Section 3.5 above). The decoded text, which was typed by the operator, is the line beginning "9BOESE …". (GRT 23D, 88.)

counter will hold 1 if the corresponding bit of the letter is dot, and 0 if it is cross. The operator looks at the display, sees that the counts are 11010, say, and types the corresponding letter (H) on the printer. The resulting printout is illustrated in Figure 34. First, the operator presses the "Print Main Headings" switch, which causes Colossus to print the wheel settings; the following line of decoded characters is entered by the operator.

5 What Else Could Colossus Have Done?

The different kinds of runs we have discussed above give a good sense of the known applications of Colossus.

When we set out to write and research this report our goal was not only to discover what Colossus actually did but also to determine what it could have done if applied to other applications. Its boosters have argued that as the first programmable computer it deserves a much more prominent place in overview histories of computing, but they have not produced any comprehensive or rigorous descriptions of its capabilities as a computer or of the mechanisms used to program it. Their argument is that the destruction of most of the Colossus machines at the end of the war, and the hiding away of the two surviving machines, was a tragic and shortsighted blunder that deprived Britain of a world-leading source of computer power. Had more Colossus machines been spared they could have been put to work on many different tasks.[24]

When we combed the literature for a description of what else Colossus could have done, or how it was programmed, we found a disappointing lack of specifics. When detailed descriptions of Colossus have appeared they are deeply bound up with the minutiae of codebreaking or concerned with the question of how the pieces of Colossus could be rearranged to build an entirely different machine. As we discuss in our paper "Colossus and Programmability"[25] Colossus has a unique position in the historiography of early computing: it is often claimed to have been both programmable and special purpose. This combination of characteristics is rather confusing: a programmable machine can be instructed to do different things, but combining this with "special purpose" suggests that there were significant constraints on what such programs could do.

It seemed to us that a careful description of how Colossus was programmed and what its programs could and couldn't do would help historians to properly integrate Colossus into the history of programming practice and of computer architecture. Yet our inescapable conclusion is that Colossus pioneered many of the digital electronic practices used to build computers but was, from an architectural viewpoint, a different kind of machine entirely. The gap between what it did during the war and what it could have done if put to service in peace time turned out to be much narrower than we had been led to believe.

5.1 The Program Followed by Colossus was Fixed

As we discuss in "Colossus and Programmability," the idea of programmability, or indeed of a program, was never used in original Colossus sources. Historically, the word "program" was applied only to activities in which possible actions were selected and sequenced over time. In the Colossus machines this basic sequences of events was fixed, and could not be changed by its users. We conclude that Colossus followed a program but that, because the overall sequence of actions could not be changed, it was not programmable. Recall our earlier, Heath Robinson-derived separation of Colossus family machines into reading, combining, and counting units. Parameters could be configured for each unit, for example to read raw bits or differences, to combine bits from different channels, or to set the printing threshold for a particular counter. But these parameters did not change the overall sequence of operations performed. For example, a counter value could not influence any Colossus action other than the suppression of printing where threshold settings had not been satisfied. There was no way to feed an accumulated value back into the combining unit, which would make it possible to change logical conditions applied according to information read earlier in an input tape. Without

24 The boldest version of this argument is made in the discussion of Colossus in B Jack Copeland, *Turing: Pioneer of the Information Age* (New York, NY: Oxford University Press, 2013).

25 Haigh and Priestley, "Colossus and Programmability".

this capability, the combining unit had no access to state information from one message character position to the next.

The only settings that modified the overall structure of a run were the wheel stepping controls, since Colossus would continue trying different start positions until all combinations of selected wheels were evaluated. Or rather, it would illuminate a light once all positions had been tried, so that the operator knew the run was over. Choosing to step two wheels rather than one, or a shorter wheel versus a longer one, would change the number of repetitions. In this sense the stepping controls altered the overall sequence of actions performed, but we see this more as akin to changing the "from," "to," and "step" values in a set of nested loops rather than writing a new program. In the same way, using a shorter or longer message tape would alter the number of message characters processed in a single cycle, but could not really be called "reprogramming."

5.2 Colossus's Simulated Code Wheels Were Specific to Fish

This architectural inflexibility was not the only, or even necessarily the most important, limitation on the practical generality of Colossus. Robinson read both of its input streams from tape, meaning that it could compare two arbitrary bitstreams (and, if the sequence on each was appropriately padded, try every possible combination of start positions). So if the appropriate tapes were prepared Robinson could be used in many different cryptanalytic attacks, even if the target code machine had a different architecture. Colossus generated the second bitstream electronically, mirroring the motor controls and wheel lengths of the Lorenz cipher machine.

Colossus could be used to count characteristics from a single input stream, for example counting the occurrence of each of five characters on an input tape. But this was a trivial operation, which came nowhere near justifying its complexity. Applied to any target other than Fish, a Colossus became a Robinson with a single input tape. This was not nearly as useful as a Robinson with the usual two or more input tapes, which explains why codebreakers kept more Robinson machines in service after the war than Colossus machines. Indeed, new kinds of Robinson were ordered by post-War codebreakers. Colossus was well adapted to Fish, but would have been almost useless against any other coding scheme within major hardware modifications.

One could imagine using the code wheels for other applications, as each one stored an arbitrary bit pattern. The psi wheels moved irregularly, but the chi wheels stepped together so could be used to store a short sequence of five-channel character representations and compare these against the bits read from tape. The problem was that each of the twelve wheels had a different length, and these were hard-coded into the Colossus hardware. Colossus would reset all code wheels once the input tape reached its end of message marker, but could not reset individual wheels. So there was no way to repeat a sequence on all five channels. The shortest chi wheel had only 23 bits, after which it would start repeating, but the longest chi wheel was 41 bits long. Bit patterns in each individual channel were repeated, but moved out of sync with each other. That was exactly how Lorenz encryption worked, but in other contexts this feature became a profound limitation.

To be fair, it would probably have been feasible to retrofit each simulated code wheel on Colossus with a switch to set its effective length. Switching the longer wheels to match the length of the shortest wheel in use could repeat a bit pattern across several channels. Even then, however, Colossus would not be able to undertake even a simple operation like "dragging" a short crib through a message as it had no way to apply logical conditions to sequences of more than two characters.

5.3 Could Colossus Be Applied to Mathematics?

As we discuss in "Colossus and Programmability" we are not sure that "computer" is the most illuminating word to apply to Colossus. Certainly Flowers himself preferred other descriptions. Colossus did not support the basic operations of multiplication and division performed by other devices known as "computers" or "calculating machines," did not interpret the signals on its input tape as numbers, and was never applied to any mathematical problems.

In a widely reported aside, Jack Good mentioned that "After the end of the war in Europe it was shown by Timms that multiplication to base 10 was almost possible on Colossus by complicated plugging. I say 'almost' because the calculation could not be completed in the time between clock pulses …. Although there was not much point in doing base-10 multiplication at the time, the capability shows that the machine was in principle more general purpose than its designer intended. Basically this is because ordinary calculations can be expressed in Boolean terms."[26]

We have spent some time and mental energy trying to understand this remark. Colossus could certainly count, which means it could add. In the simplest case, if one represented the number five as a series of five

26 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.

1s on one channel of the tape and followed it by the number 10, represented as a series of 10 1s on the same channel, then the sum of 15 could be accumulated in a counter wired to that channel.

Counters could only be incremented, which means that we do not see any practical way to implement subtraction on Colossus.[27] Neither did Colossus have any way to represent a negative number in a counter. Colossus could combine conditions on several channels, so if the larger number was always placed on one channel (say channel 1) and the smaller number on another (say channel 2), Colossus could be set up to inhibit counting of 1s from channel 1 if a 1 was also present on channel 2. That is a specialized form of subtraction, but it does not seem very useful.

Could the plugboard and control panel somehow be wired to produce a true binary adder, so that a 1 read on channel 1 coded for 1, a 1 read on channel 2 for 2, a 1 read on channel 3 for 4, a 1 read on channel 4 for 8, and a 1 read on channel 5 for 16? This would be impossible for two reasons. It is true that the XOR gates provided in the combining unit have been described as performing binary addition, and it is true that gates of this kind are the building block used to construct adders in computers. But yoking XOR gates together to build a binary adder requires a carry output, so that 1+1=0, carry 1. The Colossus XOR gates provided only a single output, and the carried bit was lost. Second, even if one could build an adder within the combining unit there would be no point, as each counter could only increment (or not increment) by 1 each time a character position on the input tape was processed.

One could simply punch a series of numbers in binary onto the input tape using the format mentioned above and accumulate the total of each channel separately: channel 1 into counter 1, channel 2 into counter 2, and so on. Taking the decimal totals from each counter, manually multiplying each as needed (counter 5 by 16, counter 4 by 8, etc.), and summing the results would give the total of all the numbers on the tape. The result would certainly be faster than adding thousands of numbers by hand, but given the constraint that only numbers between 0 and 31 could be added, and the fact that numbers could more easily be keyed into an adding machine than punched onto paper tape, this method would be of no practical use.

If one could not build a binary adder using the Colossus plug board it seems rather less likely that one could build a binary multiplier, and in any event Good's comments point to a decimal rather than binary multiplier. The circuits of the combining unit had, as mentioned above, no access to the state information stored in the counters and so could not engage in any mathematical process that required storing information from one input bit position to the next. Good does not, in any event, seem to expect this to happen: his insistence that multiplication must be done "in the time between clock pulses" suggests that multiplication was somehow expected to occur within the combining unit in the time it took to read a single character.

We had thought initially that the technique that Good had in mind for multiplication might involve coding the numbers to be multiplied onto the input tape in such a way that they could be combined with an ingenious stream of bits held on the electronic code wheels to simulate multiplication by repeated addition by the time the message tape had been fully read. But this would not fit with his implication that multiplication would (almost) take place within the reading of a single character.

We know of one way that Colossus could have multiplied, though this is probably not the method referenced by Good as it has no constraints from the Colossus clock time. The numbers to be multiplied would have been encoded on the electronic chi code wheels as a series of bits (five 1s for 5 etc.), with the input tape ignored entirely. Colossus could easily count the number of 1s on a wheel, by routing the appropriate input to a counter. Multiplication would take place by setting a logical AND condition so that a counter incremented only when a 1 on wheel 1 occurred at the same time as a 1 on wheel 2. After the wheels had stepped through all possible combinations the product would have been formed in the counter. This method would rely on the ordinary motion of the wheels, not the built-in stepping capabilities used to advance wheel start positions each time the message tape cycled. As no inputs would be taken from the message tape, the same result would be printed each time the message tape cycled until the operator paused Colossus. This method would require a tape whose message length was the product of the length of the two chi wheels being used to store the numbers. For example, if chi wheels 1 and 2 were being used, the tape would have to have exactly 1,271 characters between its start and end markers. This seems like a lot of work to go to in order to multiply a number in the range 0–41 by a number in the range 0–31, but we can see no reason that it would not work.

The larger lesson from this discussion is that Colossus was not useful for computational work. It could count but couldn't subtract. It could multiply two very small numbers using a method that would have required an infeasible amount of hand plugging, and could, reportedly, have almost (but not quite) multiplied using an unknown method.

27 Conceptually, subtraction can be carried out via addition if an appropriate notation is used – as in the arithmetic system of two's complement. But this would hardly be practical with counters that reset on reaching the value of 10,000, necessitating the use of a five-digit ten's complement. For example, subtracting 5 from 10 would involve reading ten inputs of 1 to set the initial value in the counter, then reading 9,995 inputs of 1 to subtract 5.

Acknowledgments

This project was generously supported by Mrs. L. D. Rope's Second Charitable Settlement. We extend our thanks to Crispin Rope and to all those at Lucy House. We are indebted to all those who commented on drafts of this report, and related articles, including Brian Randell, Martin Campbell-Kelly, Quinn DuPont, Jim Reeds, Mark Crowley, and David Hemmendinger. Nicolai Schmitt did a great job with the layout work for this report.