Recovering Software for the Whirlwind Computer

Author: Guy Fedorkow
Affiliation: Independent Scholar
Email: fedorkow@mit.edu

Version 06d, Dec 30, 2020

Abstract

The late 1940s were seminal years in the development of electronic digital computing machines, as researchers began to realize the enormous potential held by these machines. One of these new generation of computing devices was the vacuum-tube Whirlwind computer, designed in the late 1940s at the Massachusetts Institute of Technology. Designed from the start for real-time control applications, Whirlwind evolved to be a key element in a proof of concept for national air defense. The paper starts with a review of Whirlwind and its evolving software environment, goes on to describe the current effort to decode some of the Whirlwind software artifacts remaining in museum archives, and then describes some of the material the work has made accessible, allowing us to further study some of the software developed during that formative decade.

1. Introduction

The late 1940's were seminal years in the development of electronic digital computing machines. ENIAC was showing what could be done with vacuum tubes, the famous *First Draft of a Report on the EDVAC* was published, both EDSAC and EDVAC were being built to embody the stored-program paradigm. At the same time, World War Two and the subsequent Cold War demonstrated the critical value of faster methods of doing massive computations.

Design of the vacuum-tube Whirlwind computer was also undertaken in the late 1940's, at the Massachusetts Institute of Technology. Unlike other machines of the day being designed to compute mathematical functions such as ballistic firing tables, the Whirlwind program had its start at the Servo Mechanisms Laboratory, with funding from the US Office of Naval Research, as a means of solving mathematical equations in real time for man-in-the-loop aircraft simulation. When the program was launched in 1944, it was assumed that an electronic analog computer would be the appropriate choice for such a simulator, but after a visit in 1945 to the Moore School at University of Pennsylvania, project lead Jay W. Forrester became convinced that digital techniques were better suited to the program's ambitious goals. This application of nascent electronic digital techniques required careful attention to computation delays, to provide a realistic experience for a pilot testing the design of an as-of-yet unbuilt airframe, forming a project-long focus on real-time applications.

After some four years of investigation and development, the Whirlwind computer first came into operation at MIT in late 1949. Although the goal of real-time aircraft simulation had been shelved by the time Whirlwind first came into operation, this one-of-a-kind machine was used for many research projects, large and small, including a significant change in focus, funded by the US Air Force, to air defense, serving as the prototype for the considerably more complex SAGE air defense computer. MIT was also

given liberal use of the machine for application to all manner of specialized research projects in civil or mechanical engineering, mathematics, statistics, and many other fields, plus the development of programming tools and environments

The machine remained in service at MIT until it was decommissioned in May of 1959, but the numerous projects completed using this unique resource left behind considerable material, both as memoranda, research papers, and as software on various media.

Writing programs during the time of Whirlwind was not a simple, user-friendly task. Programmers worked with pencil and paper, and only after careful review had their programs punched into paper tape to be read into the machine and executed. At first, programmers would have been responsible for writing out the binary codes of their programs for punching and double-checking, although project leaders rapidly foresaw and acted upon the ability of general purpose computers like Whirlwind to be programmed to simplify the task of programming by developing tools to do much of the laborious code translation with automation. The Whirlwind team's contribution to what at the time was Automatic Programming was the Comprehensive System, described below in Section 2.3.

Whirlwind left behind two major collections of software, both held in the archives at the Computer History Museum (CHM), one of punched paper tape, and another of magnetic tapes used as mass storage in Whirlwind. Using this material, in a joint project between the MIT Museum and the Computer History Museum, Len Shustek, Al Kossow and I launched a project to recover some of that software from the Whirlwind project, to make it accessible for further research on the history of computing during a formative time in the development of electronic computing.

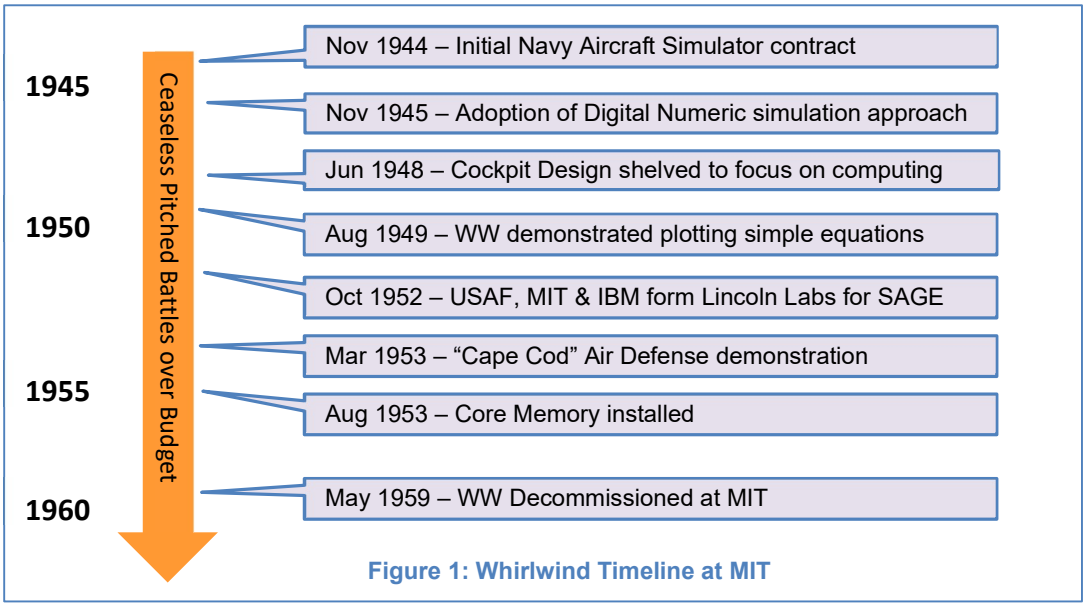


Figure 1: Whirlwind Timeline at MIT

In the remainder of this report, Section 2 gives some background on the architecture and structure of the machine to set the context. Sections 3 and 4 go on to describe current recovery work and system simulation.

2. Whirlwind Background

Whirlwind was one of the several machines designed and built in the immediate aftermath of John von Neumann's *First Draft of a Report on the EDVAC*,[1] and closely followed his suggestion of making a general-purpose machine using a single memory "organ" to store both program instructions and data.

Because the original goal was man-in-the-loop airframe simulation, Whirlwind designers Jay Forrester and Robert Everett were acutely aware of the need to manage latency in calculations. The Whirlwind team had a reputation for ignoring the high cost of their plans, and while there was some justification for that reputation, the machine did have to be practical to build. As a result, the team settled on an architecture based on a sixteen-bit words, and a sixteen-bit parallel bus interconnecting all the components. While sixteen-bit calculations would not suffice for all the equations in simulating the physics of an airframe, it would for many, and slower double-precision arithmetic could be used where needed.

Instructions were all a single word in length, most coded as a five-bit opcode with an eleven-bit address, allowing a maximum of 2,048 words of address space. Like other 1950s computers, the machine did not implement interrupts, direct memory access[2], caches, or any form of memory protection.

2.1 Whirlwind Applications

While most computer developments of the era were focused on computing mathematical functions such as ballistics firing tables (e.g. ENIAC), the Whirlwind project was launched from the start with a focus on real-time control and interaction, initially in the form of a man-in-the-loop aircraft simulator. Aircraft simulators had been created prior to Whirlwind, but the assumption had always been that analog computers were the only practical way to solve the problem.

But the Office of Naval Research's challenge was for a new, more-flexible simulator that would allow pilots to try out airframes that had not yet been built, requiring the solution of much more complex equations. The Whirlwind team studied the mathematical requirements, and saw that the conventional analog approach was not going to work.[3] At the same time, Jay Forrester had seen ENIAC at the Moore School, and was impressed by the capabilities of vacuum tube machines for high speed calculation.

From there, the Whirlwind team shifted focus from analog airframe simulation to building a digital computer capable of solving mathematical problems quickly enough to make a simulator respond to the pilot's controls like a real airplane.

Although the airframe simulation was postponed and ultimately canceled as post-War budgets shifted, the team retained the focus on applications that interacted with the physical world quickly enough to be considered "instant". After many extended budget battles, the Air Force contracted with MIT to shift the Whirlwind focus to continental air defense, a problem that had gained new urgency as the Soviet Union exploded a nuclear weapon in 1949.

The air defense work, under C. Robert Wieser, came to be focused on development and extension of the Cape Cod System.[4] and became the early prototype of ideas used in what became a massive air defense network named Semi Automatic Ground Environment (SAGE).[5]. While project status was reported regularly in many biweekly reports, almost none of the detailed technical documents of this work on Whirlwind have resurfaced yet.

However, there were twenty-four hours in a day, and MIT had many researchers eager to use high-speed computation to solve all manner of mathematical problems. Whirlwind Summary Reports itemize hundreds of research problems run on the machine -- often using programs written by the researchers themselves, and run at the most inconvenient of times to take advantage of unused computer time. Examples of research programs include:

- Strength of buildings
- Blast resistance
- Design of lenses
- Computation of tool movements for fabricating complex 3D objects[6]
- Application of numerical techniques to mathematic problems

The Scientific and Engineering Computation Group (S&EC), under Charles W. Adams, provided assistance and consulting, but programming was to be done by users. Figure 2 shows a snapshot of problems active on Whirlwind, as of November 1954.[7]

Adams' group was responsible for assisting researchers, but also for a wide range of diagnostics, programming-language "converters" (i.e. assemblers) and utilities and subroutines for managing storage, printing, input, output, etc., ultimately resulting in the Comprehensive System of Programming (see Section 2.3.1).

Along the way, S&EC produced demonstration programs for the continuous stream of visiting VIPs. And a few computer-based interactive games were created, some officially sanctioned, some not. Section 4.1 itemizes a few of the programs that have been recovered.

- Comprehensive System of Service Routines
- MIT Seismic Project
- An Interpretive Program
- The Aerothermopressor
- Coulomb Wave Functions
- Earth Resistivity Interpretation
- Data Reduction
- Special Prob1ems (Staff training, etc.)
- S&EC Subroutine Study
- Synoptic Climatology
- Construction and Testing of a Delta-Wing Flutter Model
- Blast Response of Aircraft
- Three Address Computer
- Servo Response to a Cosine Pulse
- Dispersion Curves for Seismic Waves multilayered media
- Ultrasonic Delay Lines
- Variation-Perturbation of Atomic Wave Functions and Energies
- Transformation of Integrals for Diatomic Molecules
- Course 6.25, 1954
- Investigation of Turbulent Flow
- Neutron-Deuteron Scattering
- Determination of the Critical Buckling B^2

Figure 2: Listing of Whirlwind Non-Air-Defense Programming Projects as of Nov 1954

2.2 WW Block Diagram

Figure 3 shows a contemporary interpretation of the Whirlwind computer (see Report R-221[8], Figure 36 for the view provided by the designers).

The machine fits readily into what has become the conventional stored-program von-Neumann architecture, centered on a few key registers:[9]

- A Program Counter indicates the address of the next instruction to execute.

- Arithmetic operations act on the 16-bit Accumulator (AC) and associated B Register (BR). The B Register is not directly accessible to the programmer, but is used in conjunction with the Accumulator for multiply, divide and shift operations that extend beyond 16 bits.
- Special Add Memory (SAM) is a register which holds carry-out state for use in multiple-precision arithmetic.
- The A Register (AR) saves state automatically, depending on the instruction. The important use is as a link register for subroutine calls; the address of the instruction following the jump to subroutine is stored in AR, and can be saved by the called subroutine as a return address.
- The In-Out Register (IOR) is used to communicate with I/O devices.

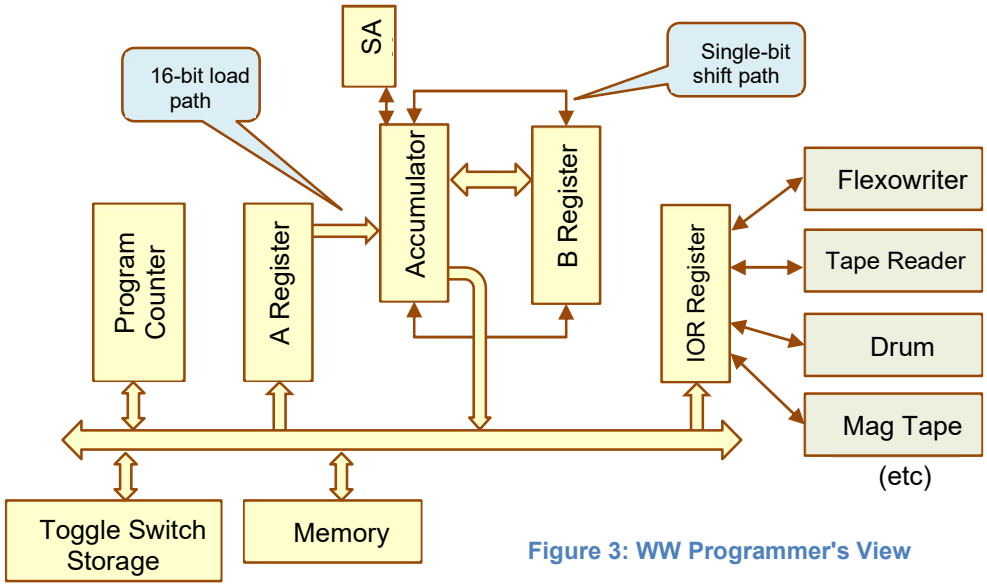


Figure 3: WW Programmer's View

Quantities stored in Whirlwind memory were normally assumed to be expressed as ones complement, sixteen-bit fixed-point binary values, in the range -1 to +1 (more precisely, -0.9999695 to +0.9999695). Scaling the problem to fit the dynamic range of the ALU was viewed as simply part of the programmer's job, aided by a selection of shifting and rounding instructions. Multiple precision arithmetic could be done by careful use of the SAM bits to convey carry and borrow bits from one word to the next.

With the exception of the instruction for multiple-precision arithmetic, any overflow or underflow would cause the machine to halt.

As a result of the choice of ones-complement, programmers also had the task of keeping track of separate values for +0 and -0, a significant difference when the only conditional branch instruction provided was conditional on the sign bit.

The most significant bit of each word (the sign bit) was labeled as Bit 0, while the least significant, of value +/- 2^{-15}, was labeled Bit 15. Because of the -1 to +1 interpretation of numbers, multiply and divide instructions justified results to the left end of the accumulator, not the right as is common in current machines.

Later in the program, Charles W. Adams' Scientific and Engineering Computation Group developed the Comprehensive System, offering programmers an interpreted instruction set that offered included full support for various floating-point number formats (see 2.3.1 below).

Among its contemporaries, Whirlwind was fast. A single-precision add would complete in 22 microseconds, hardware multiply in 34 to 41 usec, and a hardware divide in 71 usec, yielding a peak

instruction rate just short of 50,000 operations per second.[10] For comparison, ENIAC, brought on line only a few years earlier, required 2.8 msec to multiply two ten-digit decimal numbers, for about 350 multiplies per second.[11]

While this is painfully slow in modern terms, it's helpful to view Whirlwind in context. During the early 1940's, the largest "computer" in the US, and probably the world, was the US Government Mathematical Tables Project, which employed as many as 450 human computers in carefully-organized teams to compute tables such as logarithms, trigonometric functions and others. At a peak rate of about one add-per-second per person, that makes Whirlwind hundreds of times faster than the Math Tables Project.[12]

Whirlwind was never intended to be a commercial product, but the project still had a profound impact on the nascent computing industry.

A machine built in the style of the First Draft has a critical need for plentiful fast storage for both instructions and data. Although there were concerns about reliability, Forrester selected electrostatic storage, in which an array of binary digits could be stored on the faceplate of a cathode ray tube. In this configuration, a single storage tube could store as many 1,024 bits, yielding the storage needed in Whirlwind, with relatively fast random access to stored bits.

After several years of attempts to make storage tubes adequately reliable, Forrester and William N. Papian started in earnest on the development of a new technology, magnetic core memory. By summer of 1953, they had developed a core memory system that replaced the electrostatic storage tubes completely, providing the full complement of 2,048 words of core memory, with an access time of nine microseconds, yielding both a faster and much more reliable computer.

While replacing the Electrostatic Storage with core memory involved no initial architectural change to the way software worked, it did make the machine reliable enough to allow development of more complex programs. Within a few years, magnetic core memory enabled expansion of the memory system to 6K words from the original 2K words, ushering in an era of bank-switching where physical memory capacity exceeded the instruction set address limits.

A number of other organizations had been working on core memory technology at the same time (including IBM)[13], and patent rights over the invention were disputed in court for years[14]. But ultimately, the Whirlwind patent prevailed, and set the standard for computer memory systems throughout the computer business for decades to come.

2.2.1 Whirlwind I/O System

As a machine intended for real-time, person-in-the-loop applications, Whirlwind ended up with a large complement of I/O devices, each customized to fit Whirlwind's unique needs. However, like the rest of the project, the I/O system evolved considerably.

Throughout its life, the Whirlwind control room offered hundreds of buttons, switches, and indicator lights to give engineers a view of what was going on, and to influence the operation of programs. But I/O capabilities of the initial machine in 1950 confirmed the project’s interest in real-time computation.

- A "Flexowriter", an electrically-operated tape reader, printer and tape punch (similar to the Teletype machines used in newsrooms for some decades after Whirlwind) was the basic load device. Unlike modern computer interactions, the Flexowriter was not used for interactive communication with the machine until later in the program when the keyboard was connected.[15] Operators loaded tapes, pressed buttons, then retrieved printed output.
- Repurposed radar cathode ray tubes were used for displaying information in graphical format. Instructions were provided to position dots, and eventually line segments or simple characters, anywhere on the screen. The displays were used to display simulation results, plot mathematical functions, and to show current paths of tracked aircraft, derived from radar data.

When the machine came to life in 1949, there were specific operation codes for reading paper tape, printing to the Flexowriter, and displaying graphical figures on the cathode ray tube, but with only five

bits for opcodes and a growing list of new I/O device types, the I/O system was substantially redesigned in August, 1952[16] to replace the device-specific instructions with a more-generic I/O access method, using an independent address space to address devices.

Along with this change, “block transfer” instructions were added, allowing a single instruction to move entire blocks of storage to and from the drum or tape drives, paving the way for the beginnings of an operating system, where programs could be stored to or retrieved from drum with only the thirty two instructions available in the read-only Toggle Switch Memory. (See Comprehensive System Utility Program, Section 2.3.1).

Other notable I/O devices were added along the way:

- The Whirlwind team also developed the "Light Gun"[17] as a pointing device, ushering in a new kind of interactive computing. The Light Gun could be used to select among a number of dots drawn on the CRT display, and was put to use to select targets in air traffic control or air defense applications. The Light Gun also proved useful in unofficial computer games (see Section 4.2.1).
- A CRT equipped with a computer-controlled camera could be used to create permanent records of a display pattern.

Other devices connected to Whirlwind included:

- Five magnetic tape drives. In addition to the usual off-line backup, tape drives were used by the Comprehensive System Utility programs for ‘delayed printout’, where jobs would send output to the tape drives, to be transferred later by an operator to a special-purpose tape-to-Flexowriter machine.
- Two magnetic drums, each holding about 24K words. One drum was dedicated to keeping commonly-used software such as Comprehensive System components available for ready use; the second was specialized for buffering incoming radar data for the Cape Cod system.
- A high-speed Anelex printer was added late in the Whirlwind program.
- Specialized interfaces to connect with radar installations via telephone lines[18], plus experimental ground-to-air data links were added for Cape Cod development.
- Many specialized switches, buttons, and lights for user interaction, reflecting the fact that user interaction with computers using keyboard-driven command languages was still in the future.[19]

A complete list of I/O devices as of 1958 can be found in the *WHIRLWIND PROGRAMMING MANUAL*, document number 2M-0277[20]

2.3 Programming Whirlwind

The Whirlwind instruction set was lean, with 35 different op codes and only one addressing mode.

All instructions fit in a 16-bit word, and most instructions used one format, with a five-bit opcode and an eleven-bit address[21]. Most instructions would operate between the Accumulator and an operand designated by the address in the instruction. For example,

```
ad 100
```

would fetch the contents of location 100, add that to the contents of the accumulator, and store the result back into the accumulator.

The machine offered two branch instructions, one unconditional branch to a given location, the second a conditional branch dependent on the sign bit of the Accumulator, both branching to the address identified in the operand field of the instruction.

Table 1 gives the complete list of op codes available to the Whirlwind programmer (as of 1958). For a very carefully reviewed description of these instructions, refer to the *WHIRLWIND PROGRAMMING MANUAL*, (2M-0277).

Op	Instruction	Description
0	SI pqr	Select Input
0o01	-- unused --	
0o02	BI x	Block Transfer In
0o03	RD x	Read
0o04	BO x	Block Transfer Out
0o05	RC x	Record
0o06	SD x	Sum Digits (i.e., exclusive OR)
0o07	CF pqr	Change Fields (mem bank switch)
0o10	TS x	Transfer to Storage
0o11	TD x	Transfer Digits (store least significant 11 bits)
0o12	TA x	Transfer Address (store A Register)
0o13	CK x	Check (compare and halt if different)
0o14	AB x	Add B Reg
0o15	EX x	Exchange (swap AC and memory location x)
0o16	CP x	Conditional program (i.e. conditional branch)
0o17	SP x	Subprogram (i.e., unconditional branch)

Op	Instruction	Description
0o20	CA x	Clear and add (i.e., load memory into AC)
0o21	CS x	Clear and subtract
0o22	AD x	Add to AC
0o23	SU x	Subtract
0o24	CM x	Clear and add Magnitude
0o25	SA x	Special Add (used for multi-precision ops)
0o26	AO x	Add One
0o27	DM x	Difference of Magnitudes
0o30	MR x	Multiply & Round
0o31	MH x	Multiply & Hold
0o32	DV x	Divide
0o33	SL pn	Shift Left Hold (p=1) / Roundoff (p=0)
0o34	SR pn	Shift Right Hold (p=1) / Roundoff (p=0)
0o35	SF x	Scale Factor
0o36	CL pn	Cycle Left Hold (p=1) / Clear (p=0)
0o37	MD x	Multiply Digits no Roundoff (i.e. logical AND)

Table 1: Whirlwind Instruction Set

Notes:

- Five-bit opcodes go in the most-significant five bits; operand addresses in the lower eleven bits
- "x" represents an eleven-bit memory address for the operand.
- p, q, r, n are parameters for instructions that don't use memory addresses
- op codes are expressed in octal

It should be noted that a modern anathema, self-modifying code, was very much part of the Whirlwind plan. As with other machines of the era with no index registers, stack or explicit subroutine linkage, it was a common (and necessary) practice in Whirlwind code to modify instructions to either scan a list of operands, or to call a subroutine.

Here's a short sample Whirlwind program to add a list of numbers (using more modern assembler notation), updating the operand address at location 041_{22} in memory to add the next value in the list:

```
@0040:100106      loop: ca   sum       ; fetch current sum @@JumpedToBy a0045
@0041:110102       add: ad   list      ; add a number from the list @@ReadBy a0043
@0042:040106            ts   sum       ; store the sum so far
@0043:130041            ao   add       ; increment the instruction at location 41
@0044:130101            ao   count     ; increment the count
@0045:070040            cp   loop      ; branch back to repeat if negative…
@0046:000000            .word 0         ;   …otherwise, halt
                        .org 100       ;
@0100:000000      zero: .word 0         ; constant zero
@0101:177775     count: .word 177775    ; length of list to sum @@ReadBy a0044
@0102:000002      list: .word 2         ; first number to sum @@ReadBy add
@0103:000003            .word 3         ; second number to sum
@0104:000004            .word 4         ;
@0105:000005            .word 5         ;
@0106:000000       sum: .word 0         ; compute the sum and put it in this location
```

Figure 4: Sample Code - Add Up a List of Numbers

Subroutines also require self-modifying code. Figure 5 shows the more complex example of a subroutine in Whirlwind code[23], again, in updated format with symbolic addresses in place of numeric addresses.

In this subroutine, a single digit (i.e., a number between binary 0b0000 and 0b1001) is passed into the subroutine through the accumulator (AC). The address of the instruction following the subroutine call is passed in the A Register (AR) as a side-effect of the SP instruction used to call the subroutine.

The first instruction of the subroutine, at address @0177, is a TA, transfer address, instruction, which copies the least-significant eleven digits of AR into the least-significant eleven digits of the operand address, leaving the most-significant five digits unchanged. In this case, the result is to change the branch at the end of the routine (at location @0206), initially set by the programmer to branch to location zero (usually a Halt) to the current return address.

The subroutine uses a similar trick internally to overcome the lack of ALU registers or a stack. The AD (add to accumulator) instruction at @0200 adds the numeric value of the digit to be printed to the base address of a table of character values.[24] It then overwrites two EX, *exchange* instructions with this address. Executing the first *exchange* fetches the Flexowriter code from the table, and as a side effect, saves accumulator, overwriting the table entry. The RC, *record*, instruction does the actual print, then the following EX puts the table value back in its original place and restores the address to the AC. Finally, the routine returns to the caller by jumping to the address that was stored by the initial TA, at @0200.

```
; subroutine to convert an octal or decimal digit in AC into a flexowriter
;  character and print it.
;  Return the address of a word containing the character code in the AC

@0177:050206  print_digit: ta   w0206  ; overwrite the branch at the end with
                                       ; the return address in A Register
@0200:110455            ad   flexp     ; add a pointer to the base of the
                                       ;   flexo code table to the number in AC
@0201:044203            td   w0203     ; overwrite the address part of the
                                       ;   first EX instruction
@0202:044205            td   w0205     ; overwrite the second EX instruction
@0203:064000     w0203: ex   0000      ; fetch the flexo code from the table
                                       ;   and stash the table address in the
                                       ;   table itself(!) for later use
@0204:024000            rc   0         ; send AC to the printer
@0205:064000     w0205: ex   0000      ; restore the flexo code to its original
                                       ;   place in the table
@0206:074000     w0206: sp   0000      ; End of Subroutine; jump to return addr
```

Figure 5: Subroutine Code to Print a Single Digit

The Whirlwind instruction set did evolve during the first few years of the machine's operational life, although a growing body of programs kept hardware designers on a short leash.

The most significant change was introduction in 1952[25] of a uniform mechanism to access the growing body of I/O devices. Prior to this change, a handful of op-codes were dedicated to controlling the cathode ray tube display; subsequently, five op codes were defined to select, read and write all types of devices.

A special instruction was added for bank-switching after the additional 4K word core memory unit was added, overflowing the machine's 2K word architecture limit.

The 1958 Programmer's Manual, 2M-0277[26] describes the instruction set very carefully, and so far, all the recovered software appears to adhere to this generation of the instruction set.

2.3.1 Comprehensive System and the Development of Programming Methodology

During the decade of Whirlwind work, programming methodology evolved substantially, under the general heading of the MIT Comprehensive System of Programming:

- Early Whirlwind programs were coded by hand, and translated by hand into octal codes punched into paper tape to be loaded by a *read-in program* set in thirty-two sets of toggle switches. Figure 6 shows an example from Apr 1951.
- *Converter*, i.e., assembler, programs were developed to allow the programmer to write mnemonic opcodes and decimal or octal addresses, and have the machine do the conversion to binary codes. (See Figure 7, [27])
- Floating Addresses were developed to allow the programmer to use symbolic labels for memory locations, allowing the Converter to do the bookkeeping of assigning addresses for data elements in memory.
- S&EC also developed several "virtual machines", i.e., interpreters that gave the programmer the appearance of a more-capable underlying machine, with a commensurate reduction in execution speed. This work culminated in the 1956 version of the Comprehensive System[28], which offered the appearance of a machine like Whirlwind, but with multiple-precision or floating-point arithmetic, plus many subroutines for computing common mathematical functions.

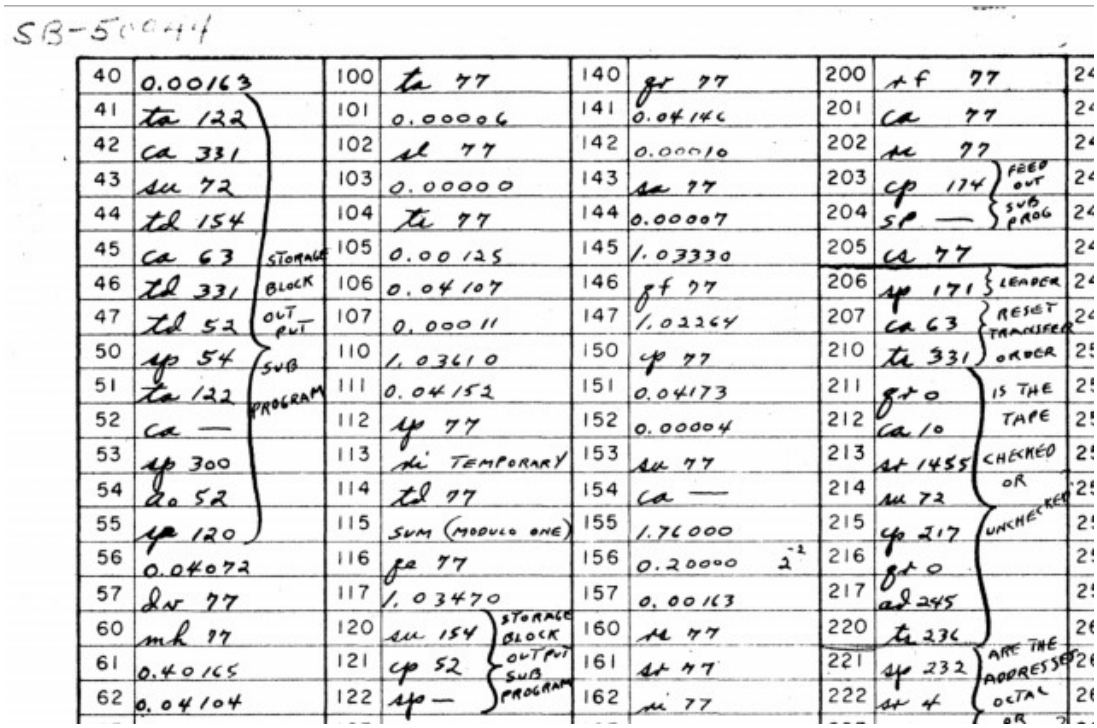


Figure 6: Whirlwind Coding Sheet

The Comprehensive System evolved in a number of steps under the direction of Charles Adams, all under the heading of Automatic Programming.

By September 1951, John Carr and J. T. Gilmore had issued an internal memo *METHOD OF PREPARING SUBROUTINES FOR THE SUBROUTINE LIBRARY*[29], a document that cites inspiration from EDSAC work at Cambridge University. While not an assembler manual, the document does describe features available at the time to programmers, using an assembler simply called the Basic Converter.

The Basic Converter of 1951 offered support for the following constructs (among others):

- Mnemonic operation codes
- Absolute and Relative numeric addresses
- Preset Parameters (“fixed once and for all at the start of the program”)
- Program Parameters (variable parameters passed into the subroutine)
- Allocation of Temporary Storage for subroutines

The intent of the work was to provide Whirlwind programmers with a set of pre-packaged subroutines, both Open and Closed,[30] that could be added to application programs. The 1952 library included numerous routines for formatting and printing numbers, and a few trigonometric functions. But the library also included an early implementation of emulated instruction sets, designed for writing programs that used extended-precision arithmetic. With these libraries, the programmer could write assembly-language programs, but have them appear to be executed by a computer with built-in support for a variety of floating-point formats, where each emulated instruction operated on two-word (i.e. 32-bit) operands. This idea was developed consistently as the Comprehensive System evolved.

A key requirement for assembly of routines from the subroutine library was the relative address, i.e., a mechanism for specifying addresses of branches and parameter references within the routine relative to the start of the routine, allowing a ‘subroutine’ to be spliced into an application program without modification.[31]

April 1952 saw the introduction of “floating addresses”, where programmers could use labels to identify addresses symbolically, delegating the task of assigning physical addresses to the Converter. Written by John W. Carr, the memo introducing the feature[32] also cites informal conversations[33] with Maurice Wilkes from Cambridge University.[34] Resolution of floating addresses required the addition of a second pass to the assembler, to correct references to the addresses resolved by the end of the first pass. This jump into automatic programming was not without controversy at the time, but it does illustrate Charles Adams’ focus on reducing the time needed to solve a problem, rather than strictly minimizing computer execution time. From the very first paragraph of Carr’s memo:

> *The philosophy of this memorandum is diametrically opposed to that of the programmer who uses octal notation, likes it, and is convinced that this is the most satisfactory way to write a program. Nevertheless, under certain conditions, the method proposed here may prove to be much more speedy, more easily corrected way to give complete machine instructions.*

By early 1954, the magnetic drum that had been attached the year before was available for routine operation[35], allowing the retirement of Converter programs that depended on magnetic tape for temporary storage in two-pass operation.

The last quarter of 1954 also saw the introduction of the Utility Control Program, marking the beginning of what might be considered an operating system, using the drum to store the subroutine library, and an executive program to load library routines from the drum as they were needed during the assembly of a program.

In December of 1955, one more group of improvements to the Comprehensive System was introduced, marking a level of maturity indicating that the system was in common use in the large community of Whirlwind programmers. The extensive revision of the programmer’s Comprehensive System Manual (M-2539-2; see Note 28) describes a number of new features, including:

- Tighter integration of Whirlwind native assembly coding and assembly-language coding in the emulated CS-II floating-point instruction set, allowing the programmer to switch between the two instruction sets with less overhead.
- The emulated instruction set grew to include support for indexing. While the underlying machine had no hardware facility for index registers, the utility had become clear, so indexing support was added to the interpreted instructions.
- A number of emulated instructions for cycle counting, plus flexible formatting of numbers for printing were added.
- The utility control program was extended to allow more jobs to be run under automatic control, without requiring an operator to load tapes for each one. [36]

The 1955 Comprehensive System continued the theme of using computer time to minimize time taken for program development. As noted in M-2539-2:

> *If it is desirable to get results quickly on a problem requiring only a small amount of computer time, the CS computer should be used as much as possible in order to minimize program preparation time. For problems requiring a large amount of computer time, careful consideration should be given to organization of the calculation and possible use of basic Whirlwind code. This can lengthen program preparation time by a considerable amount depending on the skill and experience of the programmer but may shorten computer time.[37]*

The Comprehensive System was not the only programming environment developed for Whirlwind. Although organizationally separate from Adams’ Scientific and Engineering Computing Group, by 1954, J. H. Laning and N. Zierler had published *A Program For Translation of Mathematical Equations for Whirlwind I*[38], a program that would accept mathematical formulae in algebraic notation, and translate them to executable code. Laning's work went on to play an important role in the development of navigation software for the Apollo program.[39]

3. Whirlwind Software Recovery

3.1 Sources

Recovery of Whirlwind software is only possible through the combination of project documents and software artifacts.

Whirlwind's status as an expensive and controversial[40] government research program drew plenty of scrutiny and required volumes of status reports, and the team's success in setting the direction for the much-larger SAGE program ensured that the work would not be lost when the machine was decommissioned.

In the year 2005 Bob Everett, by then having stepped down as President of MITRE Corp, arranged to have a large trove of Whirlwind documents declassified and scanned.[41] Those documents are now accessible online in a Whirlwind repository in MIT's Dome system (https://dome.mit.edu/handle/1721.3/37455). Other documents have been scanned at CHM and are available on bitsavers.org. Approximately 2,000 Whirlwind documents exist in electronic form, encompassing reports, memoranda, design studies, programming manuals and related paraphernalia.

There are also many papers not yet scanned, including over a hundred boxes of material in the MIT Archive,[42] and other documents at the Smithsonian.

Much of the Whirlwind work was classified at the time; there may remain more documents in classified Department of Defense archives, although those are hard to search.

There is also a collection of software artifacts. Along with the largest fraction of the original Whirlwind hardware, CHM acquired a collection of paper tapes and magnetic tapes containing Whirlwind software. While the Tape Librarian's index for the Whirlwind collection has not been found, there are clues:

- Some of the paper tapes have handwritten notes giving the name of the program.
- Some paper tapes and magnetic tapes are prefaced with metadata that give names, authors, and project numbers.
- Most have internal clues as to their purpose.

As noted, MIT, MITRE, the Smithsonian and CHM have all collected quite a few documents from the Whirlwind program, including technical reports, status and planning documents. While the documents describe many aspects of the machine and the project in great detail, it's rare to find a link between specific programs and documents describing the program. The Whirlwind project was carefully managed, but it was a one-of-a-kind research machine with a very specialized user community, so a coordinated set of user manuals was not a priority.

With that as background, we set about recovering data from the paper tapes and magnetic tapes.

3.2 Recovering the Tape Archives

The Whirlwind computer used a machine called a *Flexowriter*,[43] an electrically operated I/O device similar to a Teletype, with a keyboard and printer plus paper tape reader and punch, and was later augmented by a faster Ferranti Photo Electric Reader. The physical paper tape format was well-known and widely used at the time (Figure 7), and continued in use for years after as Teletype paper tape… holes representing 'one'-bits are punched in an inch-wide paper strip with sprocket holes down the middle, punched with up to seven bits of data for each sprocket hole.[44],[45]

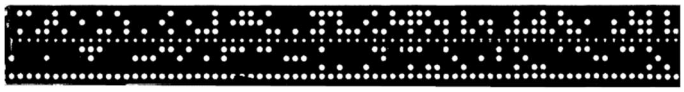


Figure 7: Flexowriter Paper Tape

The Flexowriter was the universal I/O device for Whirlwind, providing paper tape input and output, printed output and keyboard input (although keyboard input was a later addition), and as such, paper tape

was the medium used to load program code into the machine, regardless of how the program was initially produced.

Whirlwind had two magnetic drum devices for mass storage, with several uses:

- Commonly-used programs such as the Comprehensive System language converters were cached on the drum to allow fast access.
- The drum was also used in real-time computing to buffer radar data; radar signals were received asynchronously and stored on the drum, then read into Whirlwind memory when the programmer was ready.
- Intermediate data was stored on the drum as a computation progressed from one phase to another.

Magnetic tape was used as an archival medium. Programs or data that were not on the drum could be reloaded from tape when needed, and programs that produced a lot of output could send print data to magnetic tape for later off-line printing. By the end of the Whirlwind program, there were utilities to index and search tapes to look for specific programs by identification number, although formats for data tapes appear to have been completely arbitrary and up to each programmer. The physical device used by the Whirlwind team was a Raytheon telemetry recorder that could record six parallel tracks on half-inch wide magnetic tape.

Our current recovery of data from either kind of tape is essentially a two-step process:

- Read the bits from the physical media using the closest and safest available modern media device
- Interpret the formats to convert the bits back into something a Whirlwind programmer would understand.

Once recovered, data from the tapes has been used to drive analysis tools plus a simulation of Whirlwind (See Section 4).

The physical work of reading tapes has all been done at the Computer History Museum's archive site.

3.3 Paper Tapes at CHM

Reading the physical paper tape media requires care and patience, but it's not fundamentally hard. While a Flexowriter read paper tape by dragging the tape past mechanical pins, which could then fall through punched holes (or not), faster optical paper-tape readers, which required no mechanical contacts or wear on the tape, were already available (even on Whirlwind). For this project, we used a Remex Series 8000 optical eight-track reader, which would read the seven data tracks used by Whirlwind, and ignore the eighth track used by more-modern tapes.

During the year 2018, Al Kossow at CHM read each tape in the CHM archive using the Remex reader and wrote the contents out to files containing a bit-for-bit transcription of the tape.

While the physical tape material was mostly in good physical shape, a collection of 60-year-old paper objects that have not always been treated as rare artifacts certainly had some challenging spots. The original storage containers disappeared long ago. Some tapes have been maintained in climate-controlled facilities, but many were stored in a damp New England basement in the 1970's, and suffered some mold. A few had been spliced and repaired with long-perished transparent tape. But with some care, most could be read.

A total of 792 separate tapes were read, with various indications of what is on the tape:

- Many of the tapes have hand-written annotations on the leader, which have been transcribed as part of the modern file name.
- Some tapes are Comprehensive System source tapes. These are easy to identify, because they’re simply character strings in Flexowriter code (pre-ASCII days, but the same idea; letters, numbers, upper and lower case, plus a few symbols each have a six-bit code). Sadly, the source

code Comment had barely been invented, so the source tapes tend to be spare in clues as to what they do.

- Many binary tapes are in so-called "556" format[46], which is a loader format in which three successive six-bit characters from the tape are combined to form one 16-bit word. 556 format includes not just a stream of words, but also directives for where to load the words in memory, where the program should start, and even a "ditto" function to replicate the same word multiple times in memory.[47]
- There are also formats for post-mortem dumps and other support functions.

While the Whirlwind project was active there was a carefully documented procedure for having tapes numbered, punched, checked and filed in the Tape Room[48]. In practice, many of the tapes in our collection follow something close to the official numbering scheme, but with a lot of people coming and going at all hours of the day and night, and no automatic systems to check, it seems that a certain level of chaos was tolerated by the Tape Librarian. The original tape index logbook (i.e., the book into which numbers were entered) has not yet resurfaced.

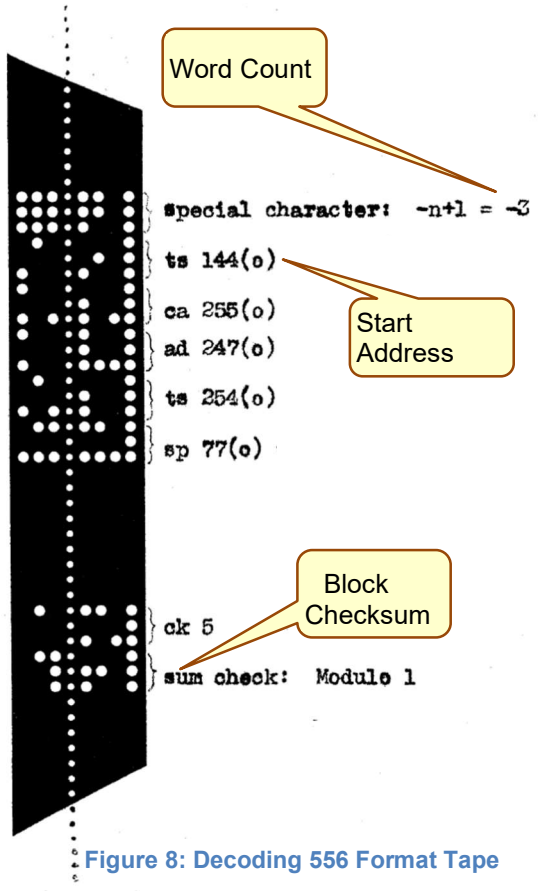


Figure 8: Decoding 556 Format Tape

556 Tapes (Figure 8) were designed to be read into Whirlwind memory by a loader, which in early days would have been entered in the 32 Toggle Switch Registers, i.e., the bank of 32 16-bit words, where each bit is represented by the on/off setting of a toggle switch. Later in the machine's life, the same 556 tapes were loaded by a drum-resident loader which implemented the same protocol (plus some additional commands). So far, an authoritative copy of the loader code has not surfaced, although the 556 command format is described in the Whirlwind document DCL-022.[49], [50]

This recovery project included creation of a Python program to read 556 tapes and convert them into a format representing the resulting Whirlwind core memory image just prior to starting the program, along with some metadata for any title information, start address or other clues that might be available. The programs contained in these core images can then be executed in a Whirlwind simulator (Section 4).

Indexing and cataloguing the paper tape collection is still to be done, but through a Python-based analyzer we have been able to discern some overall characteristics of the collection.

Tape Category	Number of Instances
Total tapes read	792
Binary executable tapes	312

Source files in Comprehensive System format	423
Text or unknown files	57
Tapes with credited author	470
Unique authors	53
Identified as diagnostics	65
Identified as utilities	82
Identified as radar test routines[51]	24

In addition to diagnostics and utilities (e.g., copying tape to drum, etc) there are a variety of demo programs, mathematical libraries, scientific problems, and thesis topics.

Beyond short programs for analyzing specific aspects of radar data, no tapes have been identified as specifically air defense.

Whirlwind tapes were relatively short. The average length is under 2800 characters, with the longest being about 17,000 characters.[52]

As noted, there are almost 800 paper tapes in the collection, but no index. An analysis tool was created to extract key characteristics from the collection in order to guide sorting and selection for further analysis.

The resulting chart, which can be seen at https://annals-extras.org/ww/, gives a number of fields:

- The Computer History Museum acquisition number and whatever note might have been hand-written on the tape leader.
- A tape number, title and author. There was a prescribed format for this information, often ignored.
- An ad-hoc guess as to whether it's a binary tape, Comprehensive System source, text, or none of the above.
- If the tape contains printable Flexowriter text, the text is reproduced in ASCII characters.
- The chart also contains fields for size and a modern MD5 fingerprint to help identify duplicates.

This chart forms the starting point for prioritizing tapes for further attention.

3.4 Magnetic Tapes at CHM

About 100 reels of Whirlwind magnetic tape are stored in the CHM archive, and in Nov 2019 we set about reading those tapes too.

The original Raytheon drives recorded six tracks on a half-inch tape, a format that has not survived the intervening decades. However, the nine-track data tape read heads that became the industry standard have improved substantially since 1955, so with a bit of creativity, the tapes have been read.

The Whirlwind designers knew that the magnetic media of the day was not of uniform quality, and suffered frequent short dropouts. To compensate, they recorded each bit on two parallel tracks, along with redundant clock tracks. Given the replication of both data and clock tracks, and the six-track tape, each column along the tape carried just two bits of information, requiring eight symbols to record one 16-bit word.

Figure 9: Whirlwind Magnetic Tape

In the case of magnetic tapes, we divided recovery of the bits on the tape into two steps:

- A digitized analog transcription of the six tracks on each tape was made and stored in (huge) files, as sets of analog samples from each track, all in CSV format.
- Len Shustek's analysis program[53] examined the digitized analog signals to find clock and data transitions, deskew data from heads that might not have been perfectly perpendicular, compare the redundant tracks, detect "tape marks", (i.e., out-of-band characters that indicate a logical dividing point between blocks of data) and reassemble groups of eight two-bit nibbles into 16-bit words. These words were then converted to the modern SIMH ".tap" format[54], more or less as they'd be read from a modern tape drive. (Note that there are optional check bits in "556" format blocks, but that's all by way of error detection.)

Figure 10 shows the alignment between the six-track tape and the 9-track heads on the modern, standard tape drive used to read it. The analog signals from the six read heads closest to the Whirlwind tape traces were digitized. Sample traces can be seen in the right side of the figure; the signals were strong and not cross-contaminated, so software could be used to detect the flux transitions on each track, decode the bits, and assemble the bits into Whirlwind 16-bit words. The software dynamically adjusted the bit timing, the gain control, and the track-to-track head skew as needed to minimize errors. Because all that was done in software, it could be repeated many times to find the best combination of parameters without physically rereading the tape – an important advantage, because the much of the magnetic oxide is often scraped off an old tape every time it is read.

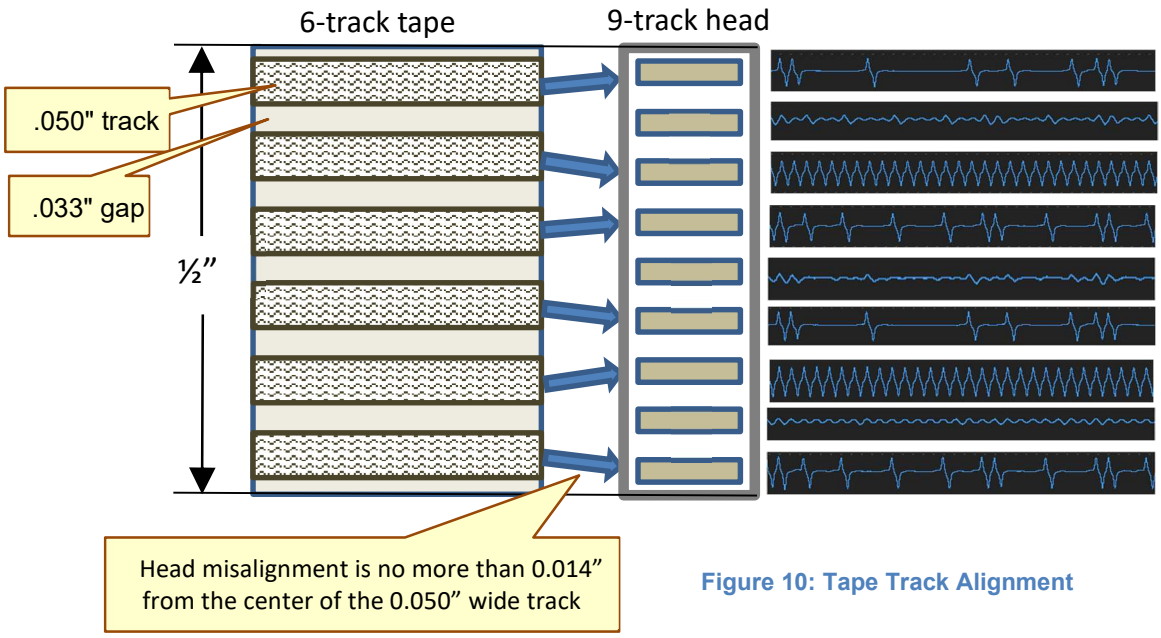


Figure 10: Tape Track Alignment

Magnetic tapes used with Whirlwind were organized into blocks of data, separated by tape marks, special out-of-band characters recorded on the tape in a way that they can't be confused with ordinary streams of words.

Several formats were used, one of which has been decoded to date.

Whirlwind did not have a file system, in the current sense of the word, to organize mass storage, but there was a mechanism to organize and find multiple separate objects on a tape, indexed by block number. Figure 11 from Whirlwind DCL-012[55] shows the indexing mechanism.

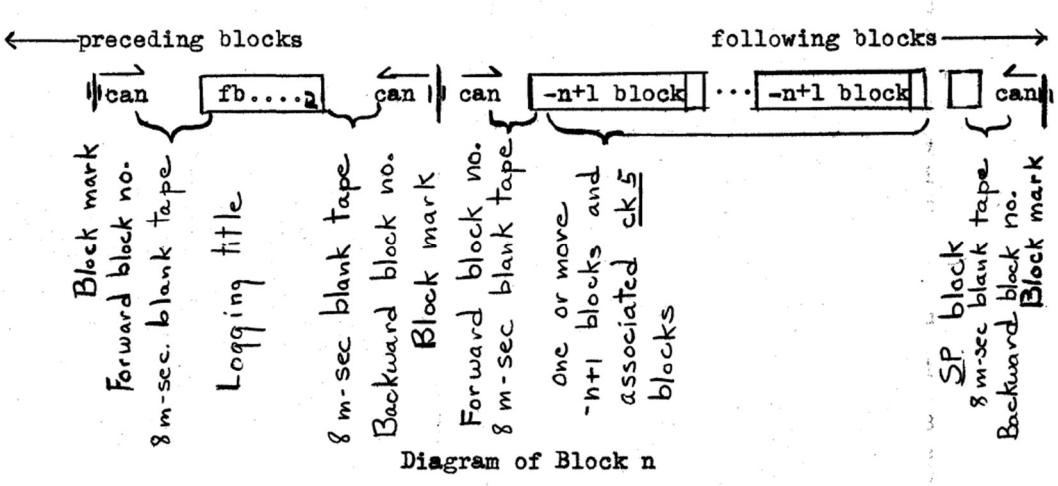


Figure 11: Whirlwind Magnetic Tape Block Format (From DCL-012)

In this format, each tape mark (aka *Block Mark*) is followed by a single-word record containing a Block Number. Subsequent tape records contain the content of the block in a format similar to the "556" loader format used for paper tape. The block ends with the same block number, but recorded in nibble-reversed

order, so that it can be read if the tape is being scanned backwards. Block numbers are required to be in sequential order.

Using this mechanism, the operator can locate a specific "file" by searching forwards or backwards for the desired block number.

Our Python-based magnetic tape decoder can find these blocks in the transcription of a magnetic tape, and extract each "file".

Analysis of the contents of the magnetic tapes has just started, and we do not have an index of what each tape contains yet. But there are certainly some easily identifiable diagnostic programs, and almost certainly some demo programs.

Tape Category	Number of Instances
Total Mag Tapes (so far)	58
"Files" recovered	11,014
(duplicate objects)	1,917
Executable 556 objects	1,029
Executables with project numbers	141
Text Files	Zero so far

Other tape formats which have not yet been decoded likely contain "drum dumps", which may include Comprehensive System programming languages and control programs.

So far, we have found no “source files” on magnetic tapes, i.e., files of printable characters to be input to an assembler, on magnetic tape. That probably reflects the fact that typists could only type to paper tape on Flexowriters, and once the source was compiled, there was no need to archive it in a separate format, as it would already have been cataloged and stored in the paper tape library.

4. Simulating Whirlwind

Given some recovered binary tapes, we went on to write an instruction level simulator of Whirlwind. The Whirlwind instruction set comprises only 35 separate instructions and one addressing mode. The machine provided no mechanism for interrupts or software traps, although arithmetic overflow would cause execution to halt. For a modern programmer, the use of 16-bit fixed-point ones-complement arithmetic is an unusual twist, but the bulk of the instruction simulator is implemented in about 1200 lines of Python code.

I/O devices are not so straightforward. The simulation environment currently includes the Flexowriter, paper tape readers, the real-time vector display, the "light gun" pointing device, and the magnetic drum. The machine was equipped with a large and ever-varying set of buttons and lights used to control its behavior, plus specialized interfaces for receiving real-time radar data; those are going to need more experience with actual code to determine how they were really used.

An instruction set simulator is an important part of the puzzle to recover software, but it's only one tool in the chain. Figure 12 outlines the tool chain that is used to decode, disassemble, and annotate the tapes, as part of an iterative process of coming to understand what's on them, So far, no examples of source

code and the exact corresponding binary have come to light, so tools to aid in reverse-engineering binary files from tapes are essential.

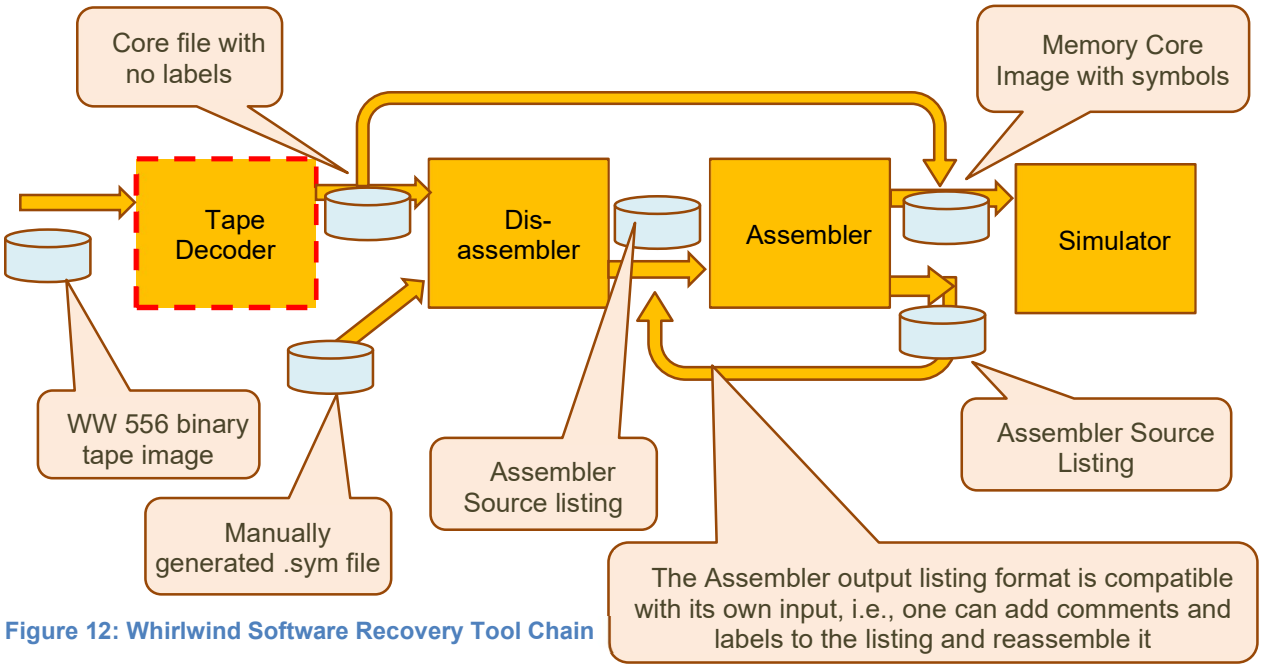


Figure 12: Whirlwind Software Recovery Tool Chain

The first tool in the chain, the Tape Decoder, has proven the weakest link. Unlike the instruction set, the binary load format, called "556" after the three 6-bit characters assembled to make 16-bit word, evolved substantially as operating-system-like features were added to the Comprehensive System, and use of the drum for non-volatile storage became routine. The initial version of the loader had to fit in the 32-word Toggle Switch Register; by the end of the Whirlwind program the same basic format had accumulated a variety of features for managing direct copy to the drum, patches, "ditto" (to load the same block repeatedly), and several other odds and ends. An authoritative document describing the loader format has not yet emerged, although Whirlwind document DCL-022[56] gives useful guidance.

Many paper tapes and magnetic tapes use variants of the 556 format, although there are quite a few magnetic tapes that use formats yet to be decoded.

Given the simple instruction set encoding, where every instruction is a single word, rough disassembly of a Whirlwind binary is accomplished with little more than a table lookup, although first-pass results can be of variable utility, given Whirlwind programmers' proclivity for writing self-modifying code, where the difference between a code segment and data segment can be fleeting at best. The disassembler also can infer subroutine calls, due to the distinctive sequence of instructions that does the call, then stores the return address in a branch statement at the end of the subroutine. The disassembler also proposes initial label names for instructions and data based on whether each word is read, written, or executed.

The output of the tape decoder can be executed by the simulator directly, but disassembly gives a starting point for understanding the contents of a tape, putting the code in human-readable form. From there the researcher comment the code, run it in simulation, watch how it executes, add more labels and comments to the recovered source, and repeat.

4.1 The Bouncing Ball

One of the first programs to run on Whirlwind was a demo to trace the path of a bouncing ball by solving the equations of motion in real time and displaying the result as a spot moving across the Cathode Ray

Tube screen. Versions of the bouncing ball demo became somewhat of a Whirlwind icon, having been captured in the video *Making Electrons Count*,[57] created to show off Whirlwind to the world sometime around 1953.

We have two versions of the Bouncing Ball program.

- An instructional example in a training manual
- A functional version recovered from paper tape.

4.1.1 The Instructional Example

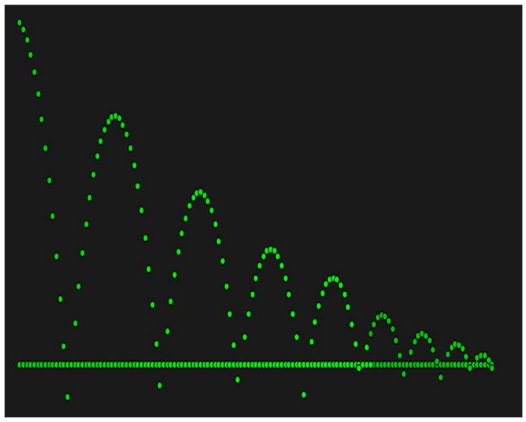


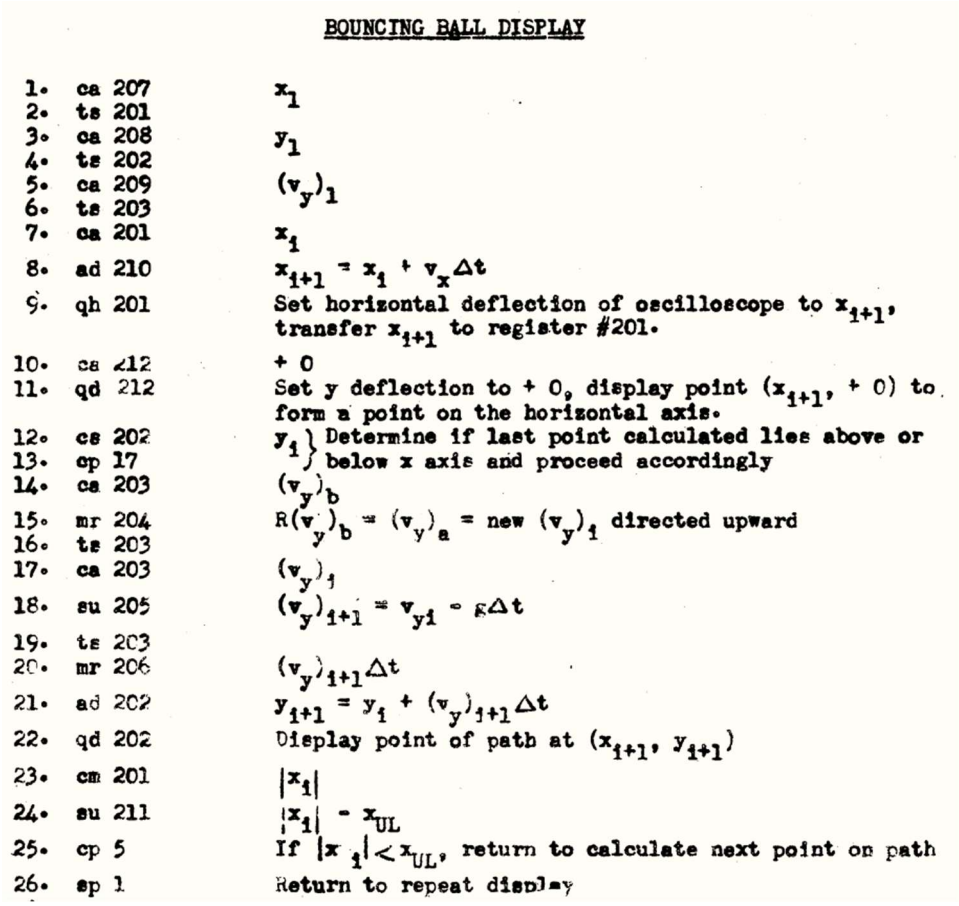
Figure 13: Bouncing Ball

The source for the instructional example is given in the Whirlwind training document R-196.[58] Given the efficient I/O, and hardware multiply instructions, the entire program fits in less than 32 words of storage -- important during initial Whirlwind debug, when toggle-switch and flip-flop storage was available, but electrostatic memory was still not reliable.

Restoration of the program illustrates the hazards of life in the era before on-line, cut-n-paste editing. The example in the training document includes several typos of the sort that people have a hard time seeing but computers reveal in a flash.[59]

Code in the example (see Figure 14) is written as a simple nested loop with the inner loop calculating and displaying each point of the curve traced by the ball, plus displaying the x-axis, while the outer loop repeats the process indefinitely to refresh the display.

The Bouncing Ball serves as a demonstration of Whirlwind's inherent real-time capability. The display controller allowed the programmer to place a dot or line on the display, but it didn't refresh the display; programmers could use a program-controlled camera to build up and capture complex images, but anything that was intended for direct viewing (such as the Bouncing Ball) it was the programmer's job to make sure calculations could be done quickly enough to re-draw the screen before the phosphor faded out.



BOUNCING BALL DISPLAY

1.	ca 207	x_1
2.	ts 201	
3.	ca 208	y_1
4.	ts 202	
5.	ca 209	$(v_y)_1$
6.	ts 203	
7.	ca 201	x_i
8.	ad 210	$x_{i+1} = x_i + v_x \Delta t$
9.	qh 201	Set horizontal deflection of oscilloscope to x_{i+1}, transfer x_{i+1} to register #201.
10.	ca 212	+ 0
11.	qd 212	Set y deflection to + 0, display point $(x_{i+1}, + 0)$ to form a point on the horizontal axis.
12.	cs 202	y_i } Determine if last point calculated lies above or
13.	cp 17	} below x axis and proceed accordingly
14.	ca 203	$(v_y)_b$
15.	mr 204	$R(v_y)_b = (v_y)_a$ = new $(v_y)_i$ directed upward
16.	ts 203	
17.	ca 203	$(v_y)_i$
18.	su 205	$(v_y)_{i+1} = v_{yi} - g\Delta t$
19.	ts 203	
20.	mr 206	$(v_y)_{i+1}\Delta t$
21.	ad 202	$y_{i+1} = y_i + (v_y)_{i+1}\Delta t$
22.	qd 202	Display point of path at (x_{i+1}, y_{i+1})
23.	cm 201	$\lvert x_i \rvert$
24.	su 211	$\lvert x_i \rvert - x_{UL}$
25.	cp 5	If $\lvert x_i \rvert < x_{UL}$, return to calculate next point on path
26.	sp 1	Return to repeat display

Figure 14: Bouncing Ball Instructional Example

4.1.2 The Bouncing Ball Tape

The Bouncing Ball evolved as Whirlwind became more stable and widely known. By the time *Making Electrons Count* was filmed around 1953, the program had grown from about 30 words to 300 words in length, with a key new feature, a hole in the "floor" that the ball would fall through if the position of a bounce coincided with the position of the hole.[60] With each iteration, the ball is launched with slightly less x-axis velocity, until ultimately it falls through the hole in the floor.

While more complex than the training example, this program is still written as a conventional procedural state machine, without using any of the advanced capabilities of the Comprehensive System for extended precision arithmetic. No subroutines are used.

The function of even a few hundred lines of code can be hard to visualize from nothing but the binary object, so an additional tool to extract flow graphs from simulation runs was added to the repertoire. The tool identifies contiguous blocks of code containing no branches, and then creates a graph showing how control is transferred among the blocks, counting the number of times each path is taken during a complete simulation run. As labels and comments are manually added to the machine code, the flow graph can be updated to identify the function of each block.

Figure 15 shows a small segment of the flow graph, drawing the x-axis, for the Bouncing Ball tape, covering three iterations of screen refresh.[61]

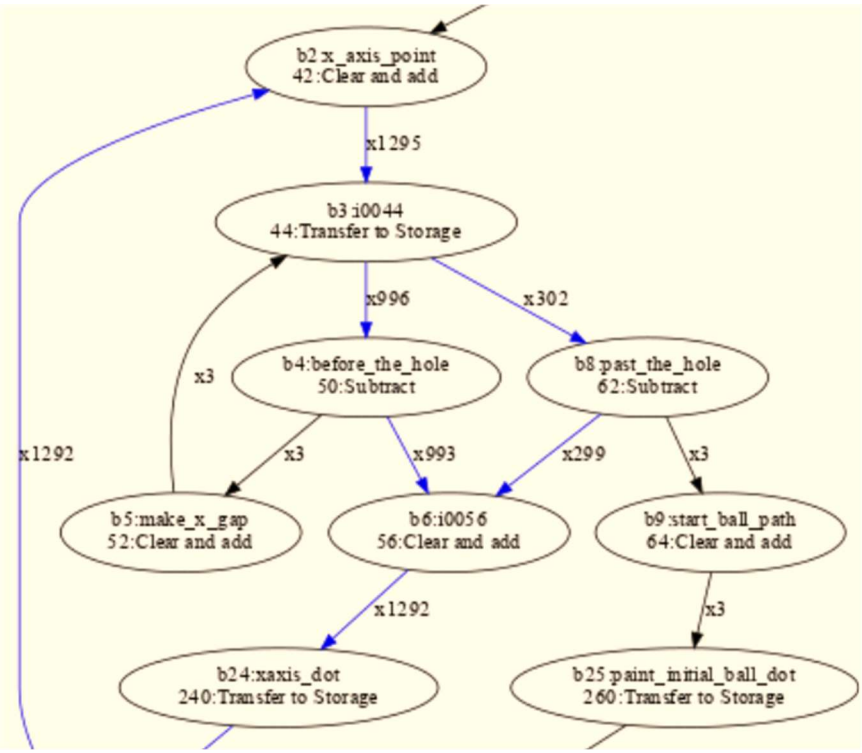


Figure 15: Segment of Bouncing Ball Flow Graph

Whirlwind code is not amenable to static analysis, given the common practice of modifying instructions during execution. While the Bounce tape is relatively restrained, the programmer did store at least one state variable in the program by changing the branch target of a key instruction.[62] We'll enhance the trace-driven flow-graph extractor with a pass of static analysis to identify possible execution paths for 'branches not taken' in the future.

4.2 Whirlwind Programs in Simulation

Of the binary tapes, a few programs from the collection function now in simulation. So far, every binary object examined appears to have been created with the Basic Converter, that is, not with the interpretive subroutines routines that evolved into the Comprehensive System.

- We have run diagnostics for CRT display and Toggle Switch Register. There are many more diagnostics, and as we learn more about the subtleties of the tape formats, they will become easier to run, and very useful in finding subtle misunderstandings in the simulator.
- A program named Quick Brown, and another called Mad Game illustrate the challenge when we're unable to match program tapes with documents, reports or memoranda. Both run without error, both have clear logical flows, but it's impossible to discern why, or when, they were written!
    - Quick Brown was likely a test program for communication with remote radar equipment during Cape Cod experiments; it prints the well-known Quick Brown Fox message in Teletype code (not Flexowriter code), and a set of incrementing codes, repeated 260 times.

  - Mad Game is more mysterious… it displays a fixed, stationary, triangular pattern of dots on the screen, and allows the user to turn them off one at a time, or in groups, with the Light Gun. No sign of competition, scoring, winning or losing.[63] The program is written in a very clear, repetitive style, with a short strip of code repeated for each dot on the display.

- A tape labelled "Jingle Bells" in fact does play the eponymous tune. An interview by Edward R Murrow with Jay Forrester on his "See It Now" television program, in December of 1951, ended with Dr Forrester wishing the audience holiday cheer by having Whirlwind play the tune. Whirlwind was hardly the first at this; Alan Turing and others had programmed computer-generated music as well by 1951.[64] As with other research machines, a loudspeaker had been attached to one bit of the Accumulator to give operators additional clues as to how the machine was behaving. The mechanism is a bit more complicated in simulation, where accumulator values from the simulation run must be extracted from the trace log and converted to MP3 to be heard.[65]
- “Calendar Demonstration Routine,” a program specifically designed for demonstration to visitors by computing the day of the week for arbitrary dates, has been partly recovered. This program is a rare instance where there is a memo that clearly corresponds to the tape[66], dated March 2, 1955. It also runs in a more complex execution environment than other programs to date, requiring drum storage as part of the load process.
- An implementation of the game of Blackjack deserves extended mention below.

It must be noted that while there are signs of a few programs in the collection that would have been used to test out radar algorithms, no air defense applications or demos appear to have made it into the collection.

4.2.1 Blackjack

While there is not a trace of supporting documentation, a Whirlwind program implementing an interactive game of the card game Blackjack (aka "21"), played with the Light Gun on the CRT display was found in the collection. Although there are few clues as to the date at which the program was first run, this is likely one of the first interactive video-based computer games.

In this game, the computer takes the role of dealer, and the player uses the Light Gun to indicate Hit or Stand, the two key actions in playing.

Figure 16 shows a screen shot of the simulator running Blackjack.[67]

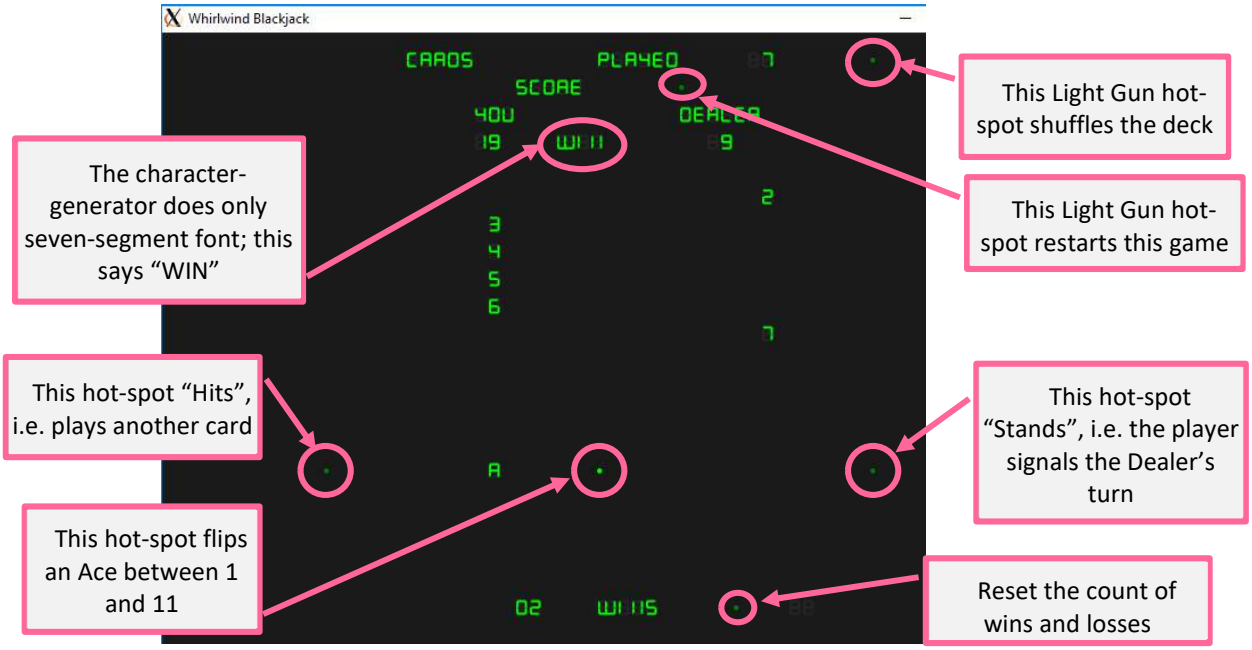


Figure 16: Whirlwind Blackjack Screen Shot

As with other recovered programs, this one appears to have been written in native assembler, although the structure is puzzling enough that it may have been cobbled together from components meant for other purposes. Whirlwind had no capability for generating random numbers, so the card deck is shuffled using an ad-hoc pseudo-random generator relying on repeated multiplication and shift operations.

5. Conclusions

Recovering software for a long-gone, one-of-a-kind research machine is no simple task. Most of the key participants are no longer with us to help clarify the details. But the Whirlwind project left a wide paper trail, so there is more than enough information to go on.

Although never produced commercially, Whirlwind left its mark on the industry:

- The reduction of magnetic core memory to practice set the direction of the entire industry for decades
- The Whirlwind Comprehensive System of Programming contributed the development of advanced assemblers that continued to be used alongside emerging high-level languages.
- Similarly, the Whirlwind team contributed to the development of what became the operating system, through the development of a batch processing system to manage the non-real-time workload of scientific calculation enabled at MIT by Whirlwind.
- Whirlwind was the key prototyping vehicle for the SAGE air defense system, which played a prominent role in the Cold War.
- Whirlwind showed that computers could do more than calculate mathematical tables, and that real-time, interactive computing was both possible, and useful.
- Experience gained from Whirlwind and the SAGE program gave IBM a boost in transitioning from a punched-card, unit-record company to a dominant player in the emerging field of computing.
- Analysis tools developed for this project will enable further investigation into the evolution of programming styles and practices.

This paper has only started to analyze the many remaining Whirlwind tapes, which will be explored in future work.[68]

Additional material, including a summary of the entire paper tape collection, as well as the binary transcriptions, can be found at https://annals-extras.org/ww/

6. Acknowledgements

The author would like to offer thanks to Dag Spicer, David Brock, and Len Shustek at Computer History Museum, and Deborah G. Douglas at the MIT Museum, for their generous advice and guidance.

Thanks to Al Kossow at CHM for safely reading and digitizing so many tapes, and Len Shustek for analysis of the digitized analog signals to extract the bits.

David Hemmendinger and Dave Walden have provided generous encouragement and assistance in the editorial process of preparing this paper.

I would like to thank the anonymous reviewers for offering valuable criticism and suggestions.

7. References

[1] For an edited version of von Neumann's seminal paper, see *IEEE Annals of the History of Computing*, Vol. 15, No. 4, 1993

[2] Except for the special case of the "buffer drum", where a data source could write one track while the computer could process data held in a previously written track.

[3] See K. Redmond and T. Smith *Project Whirlwind, The History of a Pioneer Computer*, Bedford, MA, Digital Press, 1980, pg 20. According to Redmond and Smith, Forrester came to see that the desired simulator would have to solve "at least 47 equations involving 53 variables with respect to time".

[4] See “The Cape Cod system,” C. Robert Wieser, *IEEE Annals of the History of Computing*, Volume 5, Number 4, October 1983. The program was named for the eponymous geographic cape in Massachusetts, and used radars located in Truro, MA.

[5] See K. Redmond and T. Smith, *From Whirlwind to MITRE*, Cambridge, MA, MIT Press, 2000 for an in-depth exploration of the transition to air defense work. The Air Force contracted both MIT and IBM to build SAGE in cooperation; MIT was to manage overall system design, IBM the production of the AN/FSQ-7 computer. IBM and MIT began formal cooperating on the design of the SAGE machine as early as September 1952, giving IBM substantial experience in real-time computing. For example, see A. Kromer, “M-1653 Engineering Relationships with IBM”, MIT, Cambridge MA. [Online]. Available: https://dome.mit.edu/handle/1721.3/39281

Note that most Whirlwind documents are identified as a member of a ‘series’, with a sequential number to identify the specific document (e.g., “M-1653”. Document series included Memoranda (M-series), Engineering Notes (E-series), Reports (R-series), Digital Computing Laboratory (DCL-series) or administrative documents (L-series)).

[6] See A. Siegel, “DCL-049 A Proposed Translation Program for the Numerically Controlled Milling Machine,” MIT, Cambridge MA, Jan 1955. [Online]. Available: http://bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-049_A_Proposed_Translation_Program_for_the_Numerically_Controlled_Milling_Machine_Jan55.pdf.

[7] See “DCL-15-3 “Scientific and Engineering Applications Problems,” MIT, Cambridge MA, Mar 1957. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-015-3_List_of_Programs_Mar57.pdf for a listing of all the non-air-defense MIT projects on Whirlwind from March, 1957, with identification of whether the work is for academic credit, or is sponsored.

[8] Margaret F. Mann, Robert R. Rathbone, John B. Bennett, "R-221 Whirlwind I Operation Logic," MIT, Cambridge MA, May 1954. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/R-series/R-221_Whirlwind_I_Operational_Logic_May54.pdf

[9] Current readers of Whirlwind Technical Reports should note that in the terms used in these reports, any kind of storage is referred to as a 'register', including memory locations. So the CPU had a handful of registers, but the core memory unit provided 2K (later 6K) more registers.

[10] See C. Muhle, "2M-0277 Whirlwind Programming Manual," MIT, Cambridge MA, Oct 1958. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/2M-0277_Whirlwind_Programming_Manual_Oct58.pdf. This document gives microsecond timing for each instruction.

[11] S. Lavington, *Moving Targets*, Springer, 2011, Appendix 10.

[12] The wartime Colossus code-breaking machine was fast too, but unsuited for mathematical calculation. See T. Haigh & M. Priestley, "Colossus and programmability," *IEEE Annals of the History of Computing*, Volume: 40, Issue 4, October 2018. [Online]. Available: https://doi.org/10.1109/MAHC.2018.2877912

[13] Emerson Pugh, *Memories that Shaped and Industry*, MIT Press, Cambridge, MA, 1984

[14] The MIT Archive finding aid for Magnetic Core Memory Records (AC-337) contains a summary of the patent dispute at https://archivesspace.mit.edu/repositories/2/resources/584.

[15] Report "6M-4361 Biweekly Report for Period Ending 1 June 1956", pg 16, MIT, Cambridge MA, Jun 1956. [Online]. Available: http://www.bitsavers.org/pdf/mit/lincolnLaboratory/MC665/6M-SERIES/6M-4361_BIWEEKLY_REPORT_19560601.pdf

[16] "M-1551 Initial Operation of the WWI Terminal Equipment with the New In-Out System", MIT, Cambridge MA, Jul 1952. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1451_Initial_Operation_of_WWI_Terminal_Equipment_with_the_new_In-Out_System_Jul52.pdf.

[17] The Light Gun evolved into the less-threatening and lighter-weight Light Pen on machines like the PDP-1 a decade later.

[18] Transmitting radar return signals over telephone lines was a research project in itself. Demonstrated by the Air Force Cambridge Research Center, Digital Radar Relay was the first approach, followed by Slowed Down Video (SDV) developed at Lincoln Labs. See Whirlwind report "L-86 Cape Cod System and Demonstration," MIT, Cambridge MA, Mar 1953. [Online]. Available: http://www.bitsavers.org/pdf/mit/lincolnLaboratory/MC665/L-SERIES/L-086_CAPE_COD_SYSTEM_AND_DEMONSTRATION_19530313.pdf. See also and Redmond and Smith, *From Whirlwind to MITRE*, pg. 256

[19] The Flexowriter keyboard was connected to Whirlwind in 1956, allowing Doug Ross's Manual Intervention (MIV) work. See D. Ross, "A Personal View of the Personal Work Station – Some Firsts from the Fifties", pg. 57, ACM Conference on the History of Personal Workstations, Nov 1985. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/history/Doug_Ross_from_History_of_Personal_Workstations_19860109.pdf,

[20] C. Muhle, "2M-0277 Whirlwind Programming Manual," MIT, Cambridge MA, Oct 1958. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/2M-0277_Whirlwind_Programming_Manual_Oct58.pdf.

[21] How do 35 op codes fit into five bits? Shift instructions operate on the Accumulator and B Register, don't require a memory address, and consequently had some spare bits left over to sub-decode the opcode field. Each of the three shift instructions had a sub-decode to influence the handling of the least-significant bits, and the SI I/O-Device-Select instruction did double-duty as the Halt instruction

[22] Whirlwind programmers seemed to switch back and forth between octal and decimal notations with abandon. We're sticking to octal here, unless otherwise noted.

[23] From Mag-Tape/Tape-68-Diags/68_003_fbl00-0-71. Comments and labels added by the author.

[24] Unlike ASCII codes, the ten decimal digits were not in sequential order in the Flexowriter code set, so a translator routine must do a lookup in a table relating the integer value to corresponding character code.

[25] "M-1551 Initial Operation of the WWI Terminal Equipment with the New In-Out System", MIT, Cambridge MA, Jul 1952. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1451_Initial_Operation_of_WWI_Terminal_Equipment_with_the_new_In-Out_System_Jul52.pdf.

[26] C. Muhle, "2M-0277 Whirlwind Programming Manual," MIT, Cambridge MA, Oct 1958. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/2M-0277_Whirlwind_Programming_Manual_Oct58.pdf.

[27] *Octal Program Form - Photoelectric Conversion* - Adams-Fox-Gilmore, Whirlwind drawing SB-50044, held by CHM Archive, from J. Gilmore, "M-1232 The Photoelectric Conversion Program," MIT, Cambridge MA, Jun 1951. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1232_The_Photoelectric_Conversion_Program_T-190-12_Jun51.pdf

[28] C. Adams et al. "M-2359-2 Comprehensive System Manual," MIT, Cambridge MA, Dec 1955. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-2539-2_Comprehensive_System_Manual_Dec55.pdf

[29] J. Carr, J. Gilmore, "M-1284 Method of Preparing Subroutines for the Subroutine Library," MIT, Cambridge MA, Sep 1951. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1284_Method_of_Preparing_Subroutines_for_the_Subroutine_Library_Sep51.pdf

[30] In line with Wilkes' notation, the manual distinguishes Closed subroutines, which return to the caller, and Open subroutines which are essentially inline routines that can be inserted at the desired spot in an application program, and don't depend on a call/return model.

[31] That is, the Subroutine Library was a cabinet full of paper tapes that the Flexowriter operator could duplicate into the application program during transcription of the programmer's source code to paper tape.

[32] J. Carr, "M-1445 Automatic Assembly of Programs," MIT, Cambridge MA, Apr 1952. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1445_Automatic_Assembly_of_Programs_Apr52.pdf

[33] ACM '52: Proceedings of the 1952 ACM national meeting (Pittsburgh) May 1952, https://dl.acm.org/doi/proceedings/10.1145/609784

[34] In fact, Wilkes credits Adams with a 'first' in comprehensive conversion routines in *The Preparation of Programs for an Electronic Digital Computer*, Second Edition, by M.V. Wilkes, D.J. Wheeler, and Stanley Gill, 1951, 1957, pg. 126. Terminology used by Adams' lines up with terminology in Wilkes' book, including Conversion vs Interpretation, Open and Closed Subroutines, and Floating Addresses.

[35] "M-2742 Biweekly Report, March 21, 1954," page 18, MIT, Cambridge MA, Mar 1954. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-2742_Biweekly_Report_March_21_1954.pdf

[36] See c. Adams, "A batch-processing operating system for the Whirlwind I computer," Proc. National Computer Conf., 1987, [Online]. Available:http://www.bitsavers.org/pdf/mit/whirlwind/history/Adams_A_Batch_Processing_System_for_the_Whirlwind_I_Computer_1987.pdf,

[37] C. Adams et al. "M-2359-2 Comprehensive System Manual," page Intro-5 MIT, Cambridge MA, Dec 1955. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-2539-2_Comprehensive_System_Manual_Dec55.pdf

[38] Laning, J.H. and N. Zierler. A Program For Translation of Mathematical Equations for Whirlwind I. Engineering Memorandum E-364, Instrumentation Laboratory, Massachusetts Institute of Technology; Jan 1954. [Online]. Available: https://archive.computerhistory.org/resources/text/Fortran/102653982.05.01.acc.pdf

[39] David Mindell, *Digital Apollo: human and machine in spaceflight,* Cambridge, MA, London, MIT Press, 2011.

[40] The Whirlwind program was controversial within the classified community that provided support and funding. Conversely, the public face of Whirlwind in popular media was quite positive; see Edward R. Murrow's interview of Jay Forrester on CBS's See It Now, December 1951 at https://infinitehistory.mit.edu/video/edward-murrows-see-it-now%E2%80%94jay-forrester-and-whirlwind-computer-1951

[41] Personal communication from Judith Clapp, an early Whirlwind programmer who organized the scanning effort

[42] See Department of Distinctive Collections, MIT Libraries, https://archivesspace.mit.edu/repositories/2/resources/1157, for a summary of the collection and finding aids.

[43] See https://en.wikipedia.org/wiki/Friden_Flexowriter
[44] Later paper tape systems went from seven to eight data bits per sprocket hole
[45] Whirlwind used one of the seven bits for a "valid" bit, so could encode a six bit character in each column.
[46] “556 Format” is so called because three characters on tape are combined to form one 16 bit word, five from the first, five from the second and six bits from the third.
[47] While the name "556" does specifically refer to the simple procedure of combining three six-bit characters into a 16-bit word, the term came to encompass the overall structure of the tape format, including what one might think of as 'loader' features. The term was carried over to magnetic tape records, even though there were no six-bit characters anywhere in sight on a Whirlwind magnetic tape.
[48] See M. Mckey “M-1770 Tape Preparation Requisitions,” MIT, Cambridge MA, Dec 1952. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/M-series/M-1770_Tape_Preparation_Requisitions_Dec1952.pdf for notes on the official procedures.
[49] F. Helwig, “DCL-022 Utility Control Program” MIT, Cambridge MA, Dec 1952. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-022_Utility_Control_Programs_Nov54.pdf
[50] The paper tape image in Figure 8 is from http://www.bitsavers.org/pdf/mit/whirlwind/Whirlwind_Paper_Tape_Format.pdf, an unnumbered Whirlwind document.
[51] Some of the radar data analysis tapes were likely authored by Will Crowther, later known for both the Adventure computer game and seminal work in developing the Arpanet. See https://en.wikipedia.org/wiki/William_Crowther_(programmer)
[52] It's radar analysis, "TAPE 1283 m*3 SITTLER AZ TRACK SIM"
[53] Len Shustek’s magnetic tape analysis software can be found at https://github.com/LenShustek/readtape
[54] Format is outlined in B. Supnik, “SIMH Magtape Representation and Handling” Aug 2006: [Online]. Available: http://simh.trailing-edge.com/docs/simh_magtape.pdf
[55] S. Best, “DCL-012 A Program for Transferring Binary Information Back and Forth Between Paper Tape and Magnetic Tape,” pg 6, MIT, Cambridge MA, Sep 1954. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-012_A_Program_for_Transferring_Binary_Information_Back_and_Forth_Between_Paper_Tape_and_Magnetic_Tape_Sep54.pdf
[56] F. Helwig “DCL-022 Utility Control Program” MIT, Cambridge MA, Nov 1954. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-022_Utility_Control_Programs_Nov54.pdf
[57] MIT Infinite History collection, https://infinitehistory.mit.edu/video/making-electrons-count-c-1950
[58] Hrand Saxenian, “R-196 Programming for Whirlwind I” MIT, Cambridge MA, Jun 1951. [Online]. Available: http://www.bitsavers.org/pdf/mit/whirlwind/R-series/R-196_Programming_for_Whirlwind_I_Jun51.pdf. See page. 53 for the program and a description of the parameters.
[59] For example, in the sample code, line 25 is the test at the end of the inner loop, which is shown branching back to line 5. However, Line 5 restores the y-velocity in the y direction to its initial value, undoing the new velocity calculation stored in line 19. The branch should return to line 7.
[60] The demo sequence can be seen at https://infinitehistory.mit.edu/video/making-electrons-count-c-1950, about 11 minutes from the start.
[61] The entire flow graph can be seen at https://annals-extras.org/ww/
[62] See address octal 047 in fb131-0-2690_bounce-annotated.ww at https://annals-extras.org/ww/
[63] We’re assuming the name means "mad" as in insane, not MAD as in Mutually Assured Destruction – There’s no implication of destruction and the cold-war term MAD is credited to the Kahn Institute in 1962, i.e., somewhat after the Whirlwind program at MIT was shut down.
[64] Edward R Murrow can be seen interviewing Jay Forrester in a recording of the show “See It Now”, Dec 1951. [Online]. Available: https://infinitehistory.mit.edu/video/edward-murrows-see-it-now%E2%80%94jay-forrester-and-whirlwind-computer-1951
[65] Blog article G. Fedorkow, D Brock, “Jingle Bits: Auditory Maintenance, Whirlwind Holiday Songs & the Dawn of Computer Music”, Dec 2018, [Online]. Available: https://computerhistory.org/blog/jingle-bits-auditory-maintenance-whirlwind-holiday-songs-the-dawn-of-computer-music/

[66] R. Hamlin, E. Raiffa, “DCL-57 Calendar Demonstration Routine”, MIT, Cambridge MA, Mar 1955. [Online]. http://www.bitsavers.org/pdf/mit/whirlwind/DCL-series/DCL-057_Calendar_Demonstration_Routine_Mar55.pdf

[67] The Whirlwind display controller offered a hardware seven-segment character generator, which of course cannot represent letters unambiguously. So "CAAO5 PLA4EO" actually says "CARDS PLAYED".

[68] Transcriptions of the entire tape corpus can be found at http://www.bitsavers.org/bits/MIT/whirlwind/