IBM 1401

A *Modern* Theory of Operation[1]

Gungsuh font

-- or --

What's Inside That 1401 Computer?

Version 1.0

Guy C. Fedorkow, with substantial content from Ken Shirriff

Version dates: Mar 2015, Apr 10, 2015, May 17, 2015, May 24, 2015, May 31, 2015

Merged Card Reader and Printer sections Jun 28, 2015

Second batch of 1402 Reader/Punch notes from Ken merged Jul 19, 2015

Many corrections merged, Aug 1, 2015

Merged notes on Qui-Binary, numerous corrections.

Merged notes on Core Memory, plus other corrections, Sep 7, 2015

Merged notes on Console operation, Sept 19, 2015

Corrections from Ron Williams merged, Oct 10, 2015

Re-organized Printer material, Apr 2016

Added Data Path and timing notes, May 2016

Added Tape Drive sections, Feb 2017

Caution! Construction Site!

If you see errors or can offer corrections or content, please contact guy dot fedorkow at gmail dot com.

1. INTRODUCTION 4

1.1 OBJECTIVE 4

1.2 1401 APPLICATION ENVIRONMENT 4

1.3 REFERENCE MATERIAL 5

1.4 TIME LINE 5

1.5 DESIGN TEAM 5

2. 1401 SYSTEM OVERVIEW 5

2.1 SPECIFICATIONS 6

2.1.1 Performance Metrics 6

[1] Actually, it's more a "Hypothesis of Operation" than a fully-qualified Theory…

2.1.2 Other Metrics 7
2.2 THE 1401 MACHINES AT CHM 7
2.3 THE MANY VARIATIONS 8
2.4 COMPARISON TO CURRENT COMPUTER ARCHITECTURES 8
3. INSTRUCTION SET 9
3.1.1 Data Representation 9
3.1.2 Character Set 10
3.1.3 Op Codes 12
3.1.4 Using the Word Mark 13
3.1.5 Addressing 14
3.1.6 Index Registers 14
3.1.7 Reserved Memory Locations 15
4. PRINCIPLES OF OPERATION 16
4.1 TOP-LEVEL BLOCK DIAGRAM 16
4.2 DATA PATH 16
4.2.1 Block Diagram Summary 18
4.3 DESIGN METHODOLOGY 20
4.3.1 Design Flow 20
4.4 CORE MEMORY REFRESHER 21
4.5 READING THE INSTRUCTIONAL LOGIC DIAGRAMS 21
4.6 CLOCKING AND CYCLE TIMING IN THE 1401 23
4.6.1 Clock Timing 24
4.6.2 Data Path Timing 25
4.7 INSTRUCTION CYCLE SEQUENCING 27
5. HARDWARE DESIGN 27
5.1 PHYSICAL LAYOUT AND PACKAGING 27
5.1.1 Locating Cards in a 1401 27
5.1.2 Placement of Logic Functions 29
5.2 READING THE ALD'S 30
5.2.1 Page Cross-References 32
5.3 CTDL LOGIC FAMILY 34
5.3.1 Gates 35
5.3.2 Latches 39
5.3.3 Triggers 40
5.3.4 Putting the Gates Together 42
5.4 MAJOR BLOCKS 44
5.4.1 Core Storage Memory in the IBM 1401 44
5.4.2 Expansion Memory Systems 61
5.4.3 Qui-Binary Arithmetic Unit 61
5.4.4 I/O Overlap 68

5.4.5 Multiplier 68
5.5 POWER SYSTEM 68
5.5.1 Marginal Power System 69
6. PERIPHERALS 69
6.1 1402 CARD READER/PUNCH 69
6.1.1 Reader 69
6.1.2 Logic Implementation 73
6.1.3 1402 Card Punch 76
6.1.4 The 1402 Card Reader/Punch Mechanism 76
6.1.5 Options and More Options 84
6.1.6 The Dynamic Timer and the Mysterious Vacuum Tube 85
6.2 1403 PRINTER 86
6.2.1 Printing Technology 86
6.2.2 Printer Numerology 88
6.2.3 Programming the Printer 89
6.2.4 Implementation 90
6.2.5 To Learn More 93
6.3 729 TAPE DRIVE 93
6.3.1 Tape Drive Basics 95
6.3.2 Programmer's View of the Tape System 103
6.3.3 Tape System Block Diagram 106
6.3.4 Timing and Clocking 111
6.3.5 Tape Data Path 116
6.3.6 Tape Transport Control Logic 120
6.3.7 Tape Option Logic Families 121
6.3.8 CHM 729 Tape Drive Emulator 122
6.3.9 Questions 123
6.3.10 Tape References 123
6.4 SERIAL I/O 123
7. SOFTWARE ENVIRONMENT 124
8. OPERATIONAL ENVIRONMENT 125
8.1 OPERATIONAL GUIDE 125
8.2 OPERATOR'S PANEL – THE 1401 CONSOLE 125
8.2.1 Default settings 125
8.2.2 Useful buttons 125
8.2.3 Console layout 125
8.2.4 Debugging with the console 127
8.3 INITIAL PROGRAM LOAD - BOOTSTRAPPING A PROGRAM 128
8.4 DEBUG TECHNIQUES 128
9. REFERENCE MATERIAL 129

9.1 REFERENCE DOCUMENTS 129
9.1.1 IBM Docs about the 1401 System 129
9.1.2 Programming the 1401 131
9.1.3 Peripherals 131
9.1.4 Docs about 1401 Technology 132
9.2 GLOSSARY 133
10. APPENDIX 133
10.1 KEN'S LIST OF 1401 OPTIONAL FEATURES 133
10.1.1 1401 Optional Features 133
10.1.2 Tape 138
10.1.3 1402 Card Reader / Punch 139
10.1.4 1403 Printer 139
10.1.5 1404 Printer 140
10.1.6 References 140
10.2 CARL'S SUMMARY OF LOGIC FAMILIES 140
10.3 REVISION HISTORY 142

1. Introduction

1.1 Objective

This document is intended to be used by individuals working on restoration of an IBM 1401 computer, or by anyone who's interested in how the machine works.

The doc outlines the machine's capabilities, but goes on to provide information on how the machine is organized, and how to read the primary source material.

There are many surviving documents on the IBM 1401, although none of the original engineering design specifications have been found yet. A lot has changed since 1959; this doc serves to provide a starting point to understand the basic concepts, and goes on to guide the reader in understanding how to interpret the surviving primary source material.

1.2 1401 Application Environment

The 1401 computer system was released in 1959, carefully aimed at displacing so-called Unit Record equipment (i.e. mechanical punch-card sorters, collators, tabulators, etc) for use in what was at the time a thriving market for punched-card machines in managing business operations.

[see guy's blog]

http://www.computerhistory.org/atchm/about-the-computer-history-museums-ibm-1401-machines/

1.3 Reference Material

There's a large collection of documents on the www.ibm-1401.info web page; See Section 9.1 of this document for an annotated bibliography of some of the docs that are critical to understanding this machine.

1.4 Time Line

Is there a simple picture that shows the evolution of electronic computing, e.g. von-Neumann-to-now, that would position the 1401? The famous paper *First Draft of a Report on the EDVAC* by von Neumann was distributed in June of 1945, less than 15 years prior to the release of the 1401.

In this document, "modern" and "current" refer to early 21st century, e.g. Year 2000-2015.

1.5 Design Team

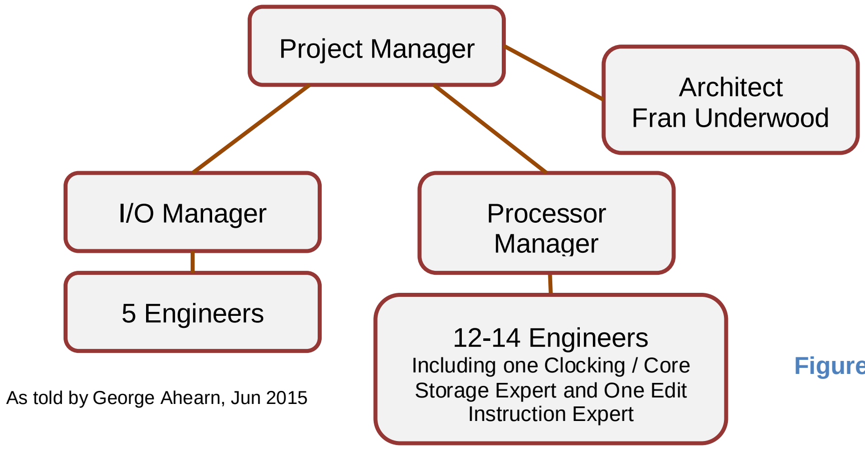

Project Manager
Architect
Fran Underwood
I/O Manager
Processor Manager
5 Engineers
12-14 Engineers
Including one Clocking / Core Storage Expert and One Edit Instruction Expert
As told by George Ahearn, Jun 2015

Figure 1: Design Team Org Chart

Names and biographies of some of the original 1401 design team members can be found at this link: http://ibm-1401.info/1950sTeamBios.html

2. 1401 System Overview

While the 1401 appeared early in the development of stored program computers, its overall architecture and structure would be very familiar to a modern computer architect.

- The machine features a single, unified memory system in which both instructions and data are stored. The memory system is addressed by character[2] , and based on magnetic core technology
- There are no plug-boards or mechanical configurations; a program is read in to core, then the machine operates on data until the job is completed.

[2] In the 1401, the unit of storage is a Character, corresponding to the symbols one could punch on a card. "Bytes" came later in the history of computing…

- The machine has a hard-wired instruction set, with instructions such as Add, Subtract, Move, Branch, etc.
- The machine features what has now become a conventional Instruction Fetch / Execute pipeline with an arithmetic unit in the data path, and additional logic for address calculation.

In addition to the computer itself, the initial 1401 system incorporated several tightly-integrated I/O devices:

- 1402 Card Reader/Punch
- 1403 Printer
- 729 Tape Drives
- RAMAC disk (added to the product later [date?])
- 1407 Typewriter Console (added to the product later [date?])

There's much more overview material at the http://ibm-1401.info/ home page.

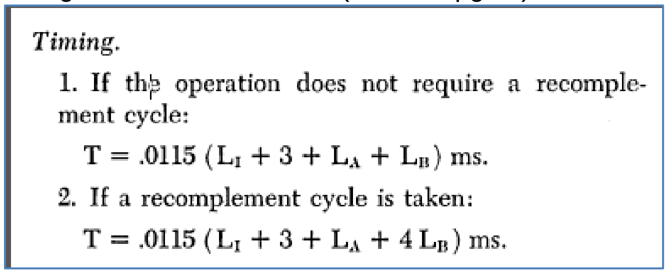
Also, see Ken Shiriff's article at
http://www.righto.com/2015/03/12-minute-mandelbrot-fractals-on-50.html

2.1 Specifications

Clock Frequency	87.5 kHz (11.50 usec cycle time) (11.5 usec is the time required to do one core memory operation)
Instruction Fetch Time	Up to eight cycles, or ~92 usec.
Memory Capacity	1400 characters minimum, 16,000 characters, maximum (using an IBM 1406 memory expansion chassis)
Nominal gate delay	~0.25 usec
Gate Count	Logic for a fully-configured 1401 system comprises about 3,000 plug-in circuit cards ("SMS" cards) Logic totals about 10,000 gates.

2.1.1 Performance Metrics

The 1403 Reference Manual [RefMan] gives detailed guidance on how to estimate instruction execution timing, and with no cache, prefetch, branch-prediction or other S/370-era performance-optimizers, timing and performance is quite predictable. Here's an example from the Reference Manual for how to estimate timing for an Add instruction (see IBM pg 29).



Timing.

1. If the operation does not require a recomplement cycle:

$$T = .0115\,(L_I + 3 + L_A + L_B)\ \text{ms.}$$

2. If a recomplement cycle is taken:

$$T = .0115\,(L_I + 3 + L_A + 4\,L_B)\ \text{ms.}$$

As an example, we can estimate the time to complete an Add instruction. Using a seven-character instruction (comprising the Add op-code and six characters of source & destination address), an operation to add two positive[3] six-digit numbers would take about 253 usec, resulting in about 4,000 adds per second.

Function	Cycles
Fetch op-code, A and B addresses	7
overhead	3
Fetch A operand	6
Fetch B operand and Store Result	6
Total cycles at 11.5 usec each -->	22
Total Microseconds -->	253

2.1.2 Other Metrics

1402 Card Reader	800 Cards per Minute
1402 Card Punch	250 Cards per Minute
1403 Printer	600 132-character lines per minute - normal alpha-numeric printing.

http://ibm-1401.info/Dhrystone.html (?? Did someone do this??)

2.2 The 1401 Machines at CHM

The Computer History Museum (CHM) has two operational 1401 systems, each with all the usual options:

- 16,000 characters of core memory
- Three or more 729 mag tape units each
- Optional index registers, branch conditions, etc
- Capable of running all IBM standard software not needing additional peripherals.
- Both 1403 controllers have buffers permitting concurrent computing and printing.[4]
- One of the machines has the “Overlap” option for concurrent computing and I/O, although this has been disabled to simplify maintenance.[5]

Both were built after the “Serial Number 25,000 Repackaging”

- 1st IBM 1401 from Hamm, Germany (built 1964)

German (DE) 1401:

Serial number 28,421

[3] Operations for mixed-sign numbers are more complicated and slower

[4] Yes, concurrent computing and I/O was a value-add option added later in the machine’s life cycle.

[5] Use of the Overlap feature on 1401’s in general was apparently so complex for programmers that it was rarely actually used.

Built: May, 1964

Elapse run time meter at arrival = 68,730.02 hours

Includes ‘overlap’ option

- 2nd IBM 1401 from Darien, Connecticut (built 1961)

Connecticut (CT) 1401:

Serial number = 25,478

Built: 1961

Run time meter at arrival = ? (The CT machine may not have a total elapsed time meter.)

This machine does not have the Overlap option.

For more on the story of how these machines came to CHM, see
http://ibm-1401.info/Connecticut1401ProposalV2-.pdf

2.3 The Many Variations

[Let’s add a few notes on how many Optional Features were available for the 1401; there are a lot!]

A list of optional features may be found in Section 10.1.

See also

https://archive.org/details/bitsavers_ibm140x225aturesCEApr61_10729447

2.4 Comparison to Current Computer Architectures

The 1401 was designed at a time when the basic architecture of a stored-program computer was beginning to stabilize. The machine has a uniform memory system used for instructions and data without restriction. There’s an instruction fetch pipeline that reads instructions comprising an op-code and some operands. There’s an arithmetic unit, and a handful of registers for storing operands and addresses.

But there are some unique features that are not common in current machines:

- BCD data representation: The 1401 does not “do” bytes; the fundamental unit of storage is an extended BCD character, encompassing numbers, the alphabet and some special characters
- Variable Precision Arithmetic: All operations in the 1401 are performed on variable-length strings of decimal digits or characters
- Word Mark: While the BCD character format stored in memory is closely aligned with punched-card character encoding, each character in memory has an additional ‘out-of-band’ bit called a Word Mark that’s used to mark the end of a variable-precision data object or instruction.
- No Stack: The hardware provided no call stack. Programmers could still store return address for subroutines, but there’s no help from the hardware
- Integrated I/O: The initial I/O devices were very tightly integrated with the CPU hardware; for example, punched cards always are read into a fixed memory address (locations 1 through 80) after the execution of a unique CPU instruction
- No Operating System: The 1401 does not have an OS. Application programs run directly on bare metal, and manage I/O devices directly. The first card of an application program deck must contain the loader that reads in and then executes the rest of the application.

- Optional Overlapped I/O: Overlapped I/O and processing was added a later option [when?]. Without the option, the programmer would execute an I/O instruction and stall until the card was read or line printed, continuing at the next instruction.[6]
- RAMAC Disk Drives and an IBM-1407 typewriter console were added later [date?]

3. Instruction Set

The 1401 offers about forty instructions, most of which directly reference operands in main memory. The machine has a handful of registers (See Data Path in Section 4.2), but each one has a specific purpose, and they are mostly referenced or updated as side effects of operations on memory.

3.1.1 Data Representation

The 1401 data path operates directly on BCD characters, rather than the (now more common) binary byte-and-word data objects found in most current architectures. The data path was designed to mesh (almost) seamlessly with Hollerith punched-cards, including the conventional punched-card character set.

A character stored in 1401 memory comprises 8 bits[7]:

Bit	Function
1	Binary values for BCD numbers, corresponding to rows 0-9 on a punched card[8]
2	
4	
8	
A	Zone A & B bits – see Character Set, Section 3.1.2
B	
CD	Check Digit – Odd Parity
WM	Word Mark; demarcates instructions and character strings. (See Section 3.1.4)

Objects are stored in memory as variable-length strings of characters or BCD numbers. The most-significant digit in a number is stored at the lowest address, and is signified by a Word Mark. Instruction addresses indicate the least-significant digit, at the highest address.[9]

- Most Significant Digit has the lowest address
- The Word Mark must be on the most-significant digit, at the lowest address of the object
- Note that a zero is represented as a BCD Zero character, (binary 0b1010) not zero, so the ALU needs translation before using the numbers.

[6] Apparently use of I/O Overlap was complicated and error-prone, so it didn't get used much

[7] But don't call it a Byte – that's sooo IBM-360…

[8] Almost… see next section for coding subtleties.

[9] In other words, arithmetic ops start with the least significant digit and auto-decrement addresses to get to the most-significant digit.

- A Negative number is indicated by setting the “B” Zone bit in the “units” (i.e., least significant) digit. See the Reference Manual [RefMan] pg XX for coding of negative numbers.

3.1.1.1 Signed Arithmetic[10]

Numbers in memory are Signed Magnitude ("true form"), although calculations are done in the ALU in Tens-Complement.

When adding two numbers with the same signs, no change in sign can take place, so the numbers are simply added. But when adding two numbers with different signs, the sign of the result is unpredictable. If the result has the opposite sign of the original destination operand, it must be complemented in memory to return it to “True form”, i.e. a sign and magnitude. This requires additional cycles, summarized below.[11]

When the signs of the two Add operands are different ("complement add"):

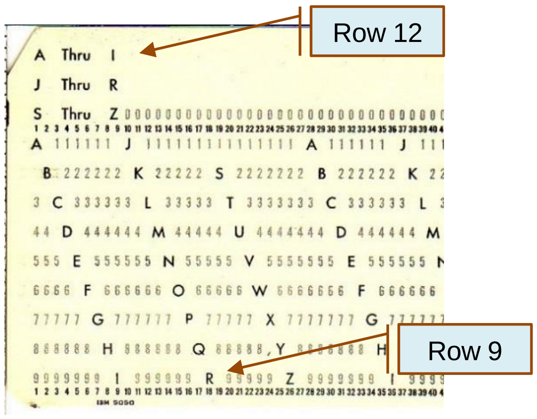
1. The ALU guesses that the A operand (of unknown length) is shorter/less than the B operand, and
2. The ALU Tens-complements the A operand while adding it to the B operand.
3. If there's a carry out of the high-order position, the guess was correct (and the resultant sign is B's sign) and the Add operation is done.
4. But if there's no a carry out of the high-order position, the guess was wrong (A is longer/greater than B), and the negative result is returned to its "true form" by serially tens-complementing it.

See the 1401 Reference Manual [RefMan] IBM pg 28 for the explanation.

See Section 5.4.3 for detail on the “qui-binary” arithmetic unit, used to improve the chances of detecting errors in arithmetic logic.

See https://en.wikipedia.org/wiki/Method_of_complements for explanation of complement arithmetic.

3.1.2 Character Set



From Ken Shirriff

Understanding the IBM punch card code

An IBM card can encode 80 characters in 80 columns. The card has 12 rows. The top row is the 12 row, and below that is the 11 row; these are both used for zone punches. Below that row are rows 0 through 9, for numeric data. The top of the card is known as the 12 edge, and the bottom of the card is the 9 edge. This is important to remember, since cards are fed into the card reader face down, 9 edge first.

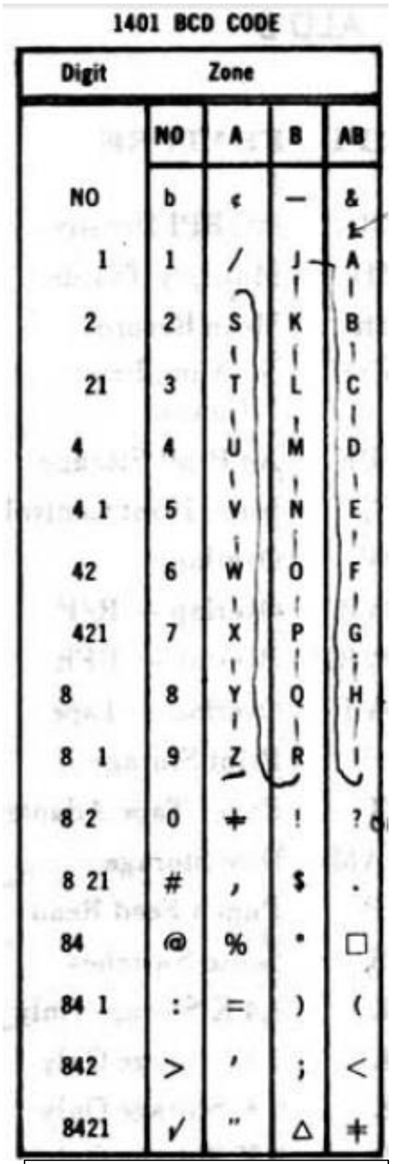
The punch card code is mostly straightforward, but has a few complications, closely tied to the 1401 BCD code.

The 1401 stores a character with a numeric part (bits 8421 storing a number from 0 through 15) and two zone bits (none, A, B, or AB). Note that the character '0' (zero) is stored internally as 10, not as 0 as you would expect.

[10] Summarized by R Garner

[11] Modern designers might ask why not stick with 9’s-Complement all the way through and dump the extra fixup cycles… We’d have to ask Fran, but this hybrid approach probably made it easier to print directly from memory, and to read a core dump without drowning in nines…

A character is punched into a card as a numeric (digit) part and a zone part. The following system is used:



1401 BCD CODE

Digit	Zone NO	Zone A	Zone B	Zone AB
NO	b	¢	–	&
1	1	/	J	A
2	2	S	K	B
21	3	T	L	C
4	4	U	M	D
4 1	5	V	N	E
42	6	W	O	F
421	7	X	P	G
8	8	Y	Q	H
8 1	9	Z	R	I
8 2	0	‡	!	?
8 21	#	,	$	.
84	@	%	•	□
84 1	:	=	)	(
842	>	'	;	<
8421	√	"	Δ	‡

from 1401-CE_Pocket-Ref-Man-56-389-.pdf

The numeric part (0-15) of the character is usually punched as follows:

- 0: no punch (example characters: blank or '&')
- 1-9: punched in that row (e.g. digit or letter)
- 10: punched as 8-2 or 0, depending on the circumstances. (e.g. character '0' or '!')
- 11-15: punched as 8-3 through 8-7. The second punch is the numeric value minus 8, so the two punches add up to the desired value. (e.g. '#' or '=')

The zone punches are usually punched as follows:

- No zone bits: no punch. (e.g. digit)
- Zone bits A: row 0 punched. (e.g. 'S' through 'Z')
- Zone bits B: row 11 punched. (e.g. 'J' through 'R')
- Zone bits AB: row 12 punched. (e.g. 'A' through 'I')

The main complication in the code is that the 0 row on a card can indicate a zone (A) or a numeric value (0). Also, the 1401 stores a blank character as numeric value 0, while a zero character is stored as numeric value 10. This results in special cases, which are handled as follows:

- BCD value 10 with no zone (digit '0') is punched as 0.
- BCD value 10 with zone A (record mark) is punched as 0-8-2 (since row 0 can't indicate a zone and a digit at the same time).
- BCD value 10 with zone B or zone AB ('!' or '?') are punched as 0-11 and 0-12 respectively. Interestingly, the 029 keypunch handles these differently, using 11-8-2 and 12-8-2.

- BCD value 0 with no zone (blank) has no punches.
- BCD value 0 with zone A (cent) cannot be read from a card by the 1401 (since a 0 punch is interpreted as zero). The 029 keypunch uses an 8-2 punch for this.
- BCD value 0 with zone B ('-') has an 11 punch.
- BCD value 0 with zone AB ('&') has a 12 punch.

See 1401 Reference Manual [RefMan] (page 170) for a summary of the IBM 1401's punch card code.

The pocket-guide http://ibm-1401.info/1401-CE_Pocket-Ref-Man-56-389-.pdf also gives a concise version of the character set definition. Note that the definition of some characters depends on which Printer Chain is installed.

[Note that not all characters appear on a keypunch keyboard; BCD in the 1401 era defines only 48 characters… it wasn't until later that EBCDIC defined a larger set]

3.1.3 Op Codes

Table 1 shows a summary of all the op-codes available to a 1401 programmer.

In each case, the op-code (underlined in this chart) is literally the character that would be punched on a card to trigger the function.

Most instructions are followed by the three-digit addresses of one or two operands. Addressing is a bit complicated (see Section 3.1.4) but for starters it's best to think of the address as a three-character decimal number.

So to move a number or string from location 100 to location 200, the instruction punched on the card would be

```
        M100200
```

Instruction Format	Instruction Function
`.,. III`	Halt, Halt and Branch
`B(I), B(I)d, B(I)(B)d`	Branch Unconditional, Branch Conditional, Branch Char. ?
`V III BBB d`	Branch on Word Mark or Zone
`D AAA BBB`	Move Digit
`Y AAA BBB`	Move Zone
`, AAA BBB`	Set Word Marks
`□ AAA BBB`	[Lozenge] Clear Word Marks
`/ (A), / (I)(B)`	Clear Storage, Clear Storage and Branch
`M AAA BBB`	Move
`L AAA BBB`	Load
`C AAA BBB`	Compare
`Z AAA BBB`	Move Zero Suppress
`P AAA BBB`	Move Record
`A AAA BBB`	Add
`S AAA BBB`	Subtract
`Q AAA`	Store A-STAR
`H AAA`	Store B-STAR
`? AAA BBB`	Zero and Add
`! AAA BBB`	Zero and Subtract
`E AAA BBB`	Edit

`1, 1 III`	Read Card, Read Card and Branch
`4, 4 III`	Punch Card, Punch Card and Branch
`2, 2 III`	Print, Print and Branch
`3, 5, 6, 7 III`	Combined Read / Punch / Print Ops (ops 1, 2, 4 can be combined into one instruction!)
`K d`	Stacker Select
`F d`	Forms Control
`# AAA BBB`	Modify Address (only in machines with a 1406 Memory Expansion)[12]
`@ AAA BBB`	Multiply
`% AAA BBB`	Divide
`U % U X d`	Unit Select (729 tape drive)
`M % U X BBB d`	Move (729 tape drive)
`L % U X BBB d`	Load (729 tape drive)
Prog.Load	Load programs from 1402 or 729 tape

Table 1: Op Code Summary[13]

http://www.textfiles.com/bitsavers/pdf/ibm/140x/1401ReferenceCard.pdf
http://ibm-1401.info/1401-CE_Pocket-Ref-Man-56-389-.pdf

3.1.4 Using the Word Mark

Each character in memory has a bit called a Word Mark, used to delimit numbers and strings, and also instructions.

For a number or character string, the Word Mark is *set* on the “leftmost” character of a field (lowest address, i.e., most-significant digit) and *cleared* on all other characters of the string.

For an instruction, the Word Mark is set on the first character of the instruction – the opcode. There must also be a Word Mark on the first character after the end of the instruction. Normally this will be the next instruction, but there must also be a Word Mark after the last instruction.[14]

Full rules for Word Marks are given on page 15 of the Reference Manual [RefMan].

Instructions that use A and B address fields can be *chained*. In this case, the A and B (or B) fields are omitted from the instruction by placing a Word Mark indicating the next instruction to truncate the full

[12] See http://www.ibm-1401.info/IBM1401-225-6541-0_1401_Optional_Features_CE_Apr61.pdf pg 8. The instruction logic is actually in the 1406 memory expansion chassis (!)

[13] Cribbed from Ron Williams

[14] This means that the execution unit fetches characters in an instruction until it hits a Word Mark, indicating the next instruction (with a couple of exceptions allowing the first instruction on a card to run without a word-mark; see IPL Section 8.3). This in turn means that the first character of each instruction is usually fetched *twice*, once to indicate the end of the current instruction, and again to start the next one. [find an xref to show this?]

instruction format. In that case, the contents of the A and B registers are unchanged from the previous instruction, allowing a sequence of instructions to use adjacent memory locations without having to reload A and B each time.

For examples of chaining, see the Reference Manual [RefMan], Section *Instruction Chaining*, ibm pg 20.

3.1.5 Addressing

Data objects (strings or numbers) are stored in memory as variable-length strings of characters or digits, most significant digit in low address, least significant at high address

➔ So multi-character instructions (like "Add") start executing at the high address and auto-decrement until a Word Mark signals the end

Addresses less than 1000 are indicated simply by three decimal digits in the address field of an instruction. For addresses in the range of 1,000-15,999, zone bits in the Units and Hundreds address digits are used to extend the range. Zone bits in the Tens digit are used to trigger address indexing (Section 3.1.6).

[the picture here is not yet approved]

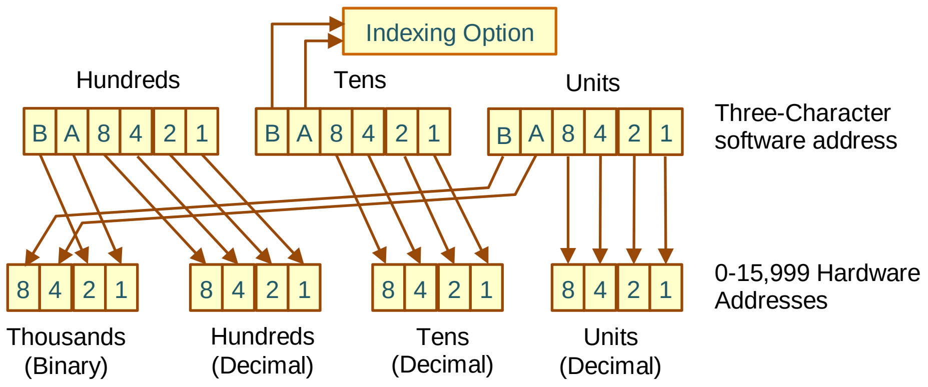


Figure 2: Extended Address Decoding

See Address generation

http://ibm-1401.info/1401AddressingStanP.html

3.1.6 Index Registers

The IBM 1401 could be equipped with optional Index Registers, a mechanism to allow a programmer to step through a loop while accessing a sequence of addresses, by adding the contents of an Index Register to the A and/or B operand address in a normal instruction.

For example, to add up the elements of an array, the programmer would initialize an Index Register to zero, then use an ordinary Add instruction with the source address indicating the first element of the array, but with Indexing enabled. After each access, another instruction would increment the Index register, so that next pass through the loop, the Add instruction fetches the next address in the array.

The 1401 provided three Index Registers, which are actually implemented as three sets of fixed locations in main storage. Which index register to use is indicated by the Zone bits in the Tens digit of the address in any instruction (See Figure 2 and Table 2).

Index Location	Tens Zone Bits	Core Memory Address
(no indexing)	No A Bit, No B Bit	n/a
1	A Bit, No B Bit	087-089
2	B Bit, No A Bit	092-094
3	A Bit and B Bit	097-099

Table 2: Index Register Addresses

IBM 1401 addresses are held in three digits, but because of the use of zone bits, they're not simply three digit numbers. When adding or subtracting addresses, as would be needed when incrementing the index register in a loop, there's a special instruction "Modify Address" (`# AAA BBB`) to add two numbers in address format. Index registers can be decremented by adding a large number to cause the field to wrap.

Programmers could still do indexing on machines without the Index Register option, by modifying the A or B instruction address of the code in memory.[15] Zone bits had to be calculated "manually" for addresses greater than 999.

Detail on Index Registers can be found in the Reference Manual [RefMan] on ibm pg 84. The same reference shows how addresses can be calculated, on ibm pg 106.

3.1.7 Reserved Memory Locations

There are a number of locations reserved for the card reader/punch, line printer, and other functions, as shown in Table 3.

Address	Function
0	Loop counter used to count down the 12 rows, used by Card Reader logic
1-80	Card Reader Data
087-089, 092-094, 097-099	Index Registers[16]
100	Loop counter used to count down the 12 rows, used by Card Punch logic
101-180	Card Punch Data
200	(unused by hardware)
201-332	Line Printer Output

Table 3: Reserved Memory Locations

All the gaps between reserved locations are free to be used. When the specific operations are not in progress, all locations can be used for other things.

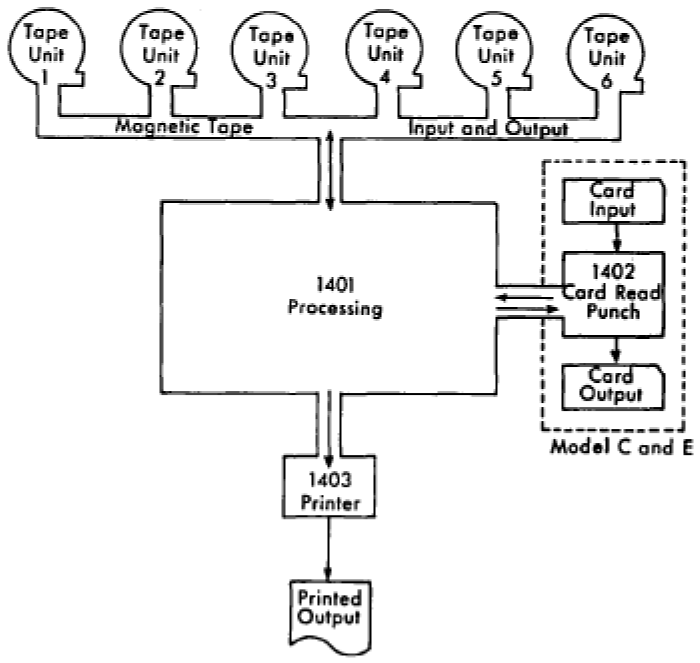
[15] OMG! Self-Modifying Code! It was actually a pretty good idea at the time…

[16] No, we don't know why there are unused locations between the reserved index registers. Probably someone saved some gates by decoding on a "fives" boundary.

See System Operations Reference Manual A-9. [http://bitsavers.trailing-edge.com/pdf/ibm/140x/A24-3067-2_1401_1460_System_Operation_Reference_Manual_Sep66.pdf ?]

4. Principles of Operation

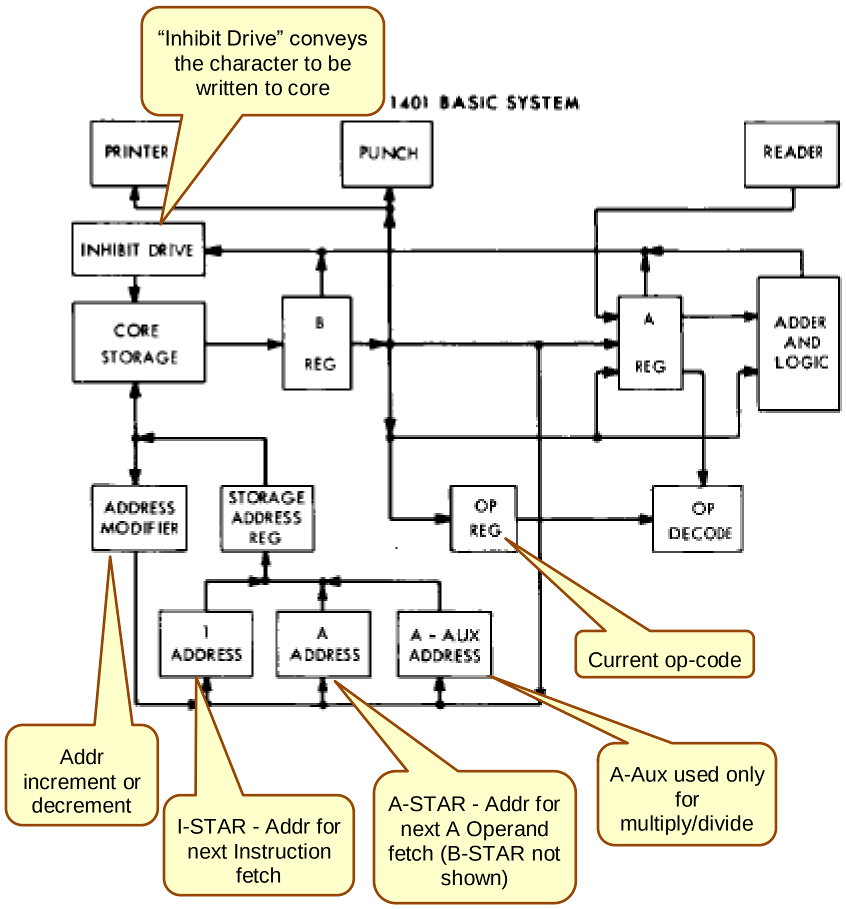
4.1 Top-Level Block Diagram



(Pg 53 in RefMan)

4.2 Data Path

There are a number of block diagrams of the processor's data path, containing registers and address calculation. An abstract diagram of the overall processor, from the Field Maintenance Manual ([FMM] pdf pg 8) is shown below.


"Inhibit Drive" conveys the character to be written to core
1401 BASIC SYSTEM
PRINTER
PUNCH
READER
INHIBIT DRIVE
CORE STORAGE
B REG
A REG
ADDER AND LOGIC
ADDRESS MODIFIER
STORAGE ADDRESS REG
OP REG
OP DECODE
I ADDRESS
A ADDRESS
A - AUX ADDRESS
Current op-code
Addr increment or decrement
I-STAR - Addr for next Instruction fetch
A-STAR - Addr for next A Operand fetch (B-STAR not shown)
A-Aux used only for multiply/divide

The FMM shows a few variants of the same diagram, including options for multiply/divide, tape drives and the Process Overlap feature (not present on the CHM machines).

There's a similar block diagram in G24-1477 pdf pg 6

The ILD set contains a more detailed data path block diagram which is critical to understanding many aspects of the machine. It's too big to fit in this doc, but the diagram looks like this:

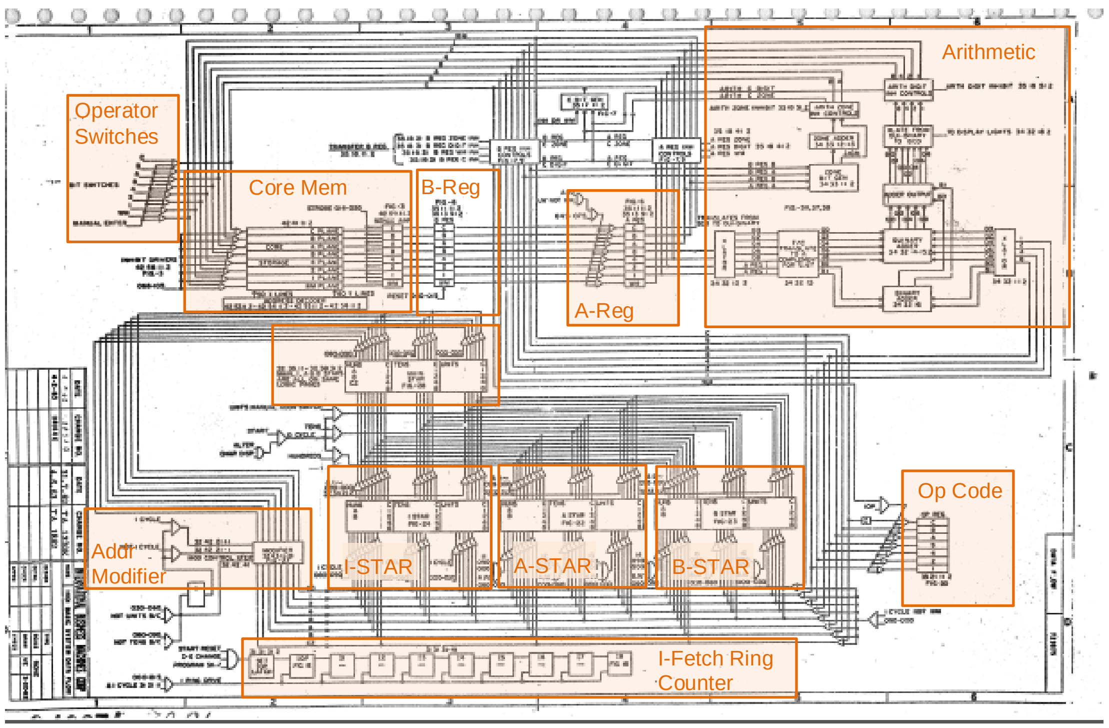


Figure 3: Data Path Block Diagram

The full-sized version can be seen at [ILD] pdf-pg 4, or [ILD2] pdf-pg 2.

4.2.1 Block Diagram Summary

The ILD block diagram in Figure 3 shows the essential elements of the computer's data and address calculation components, minus the I/O devices and a substantial amount of control logic.

Key elements in the block diagram include the core memory system, an arithmetic unit and a series of working registers. As noted in Section 3.1.3, the IBM 1401 instruction set focuses on memory-to-memory variable length operands, and as such, does not offer a general-purpose register set visible to the programmer. None the less, the instruction set does need working registers, and they are shown in the block diagram.

Important elements are as follows:

Core Memory – The entire memory system is represented by one block in Figure 3. The machine does a memory read and write every 11.5 usec clock cycle, with an address for each cycle comes from the Main STAR register. Part way through the cycle, a single character of Read Data from the selected cores is latched in the B Register for use in the computing part of the cycle. Towards the end of the cycle, a character must be written back to the same core location using the Inhibit Drive (aka write data) port into the memory system. (See Section 4.4 to learn what Inhibiting has to do with Writing. And see Section 4.6.2 for details on how the timing works.)

STAR – Storage Address Registers – There are a number of Storage Address Registers that hold working copies of addresses for various operations.

- **Main STAR** holds the address being accessed via the current machine cycle
- **I-STAR** contains the address of the next instruction to fetch

- **A-STAR**, **B-STAR** registers contain the next operand address for the A and B instruction operands.
- Some of the IBM 1401 optional features (e.g., instruction overlap and multiply/divide) use additional storage address registers.

Each STAR is three characters wide so that the full 16,000 characters of storage can be addressed. (See Section 3.1.5 for how memory addressing works)

As each cycle proceeds, the next address for the relevant STAR is computed by the block titled Modifier (Addr Modifier in Figure 3), which can add one, subtract one, or add three[17], depending on the cycle type.

Different versions of the IBM 1401 have additional STARs to accommodate instruction overlap or other features.

Detailed description of the STARs can be found in the IBM 1401 Data Flow doc [G24Flow] (IBM pg 3, pdf pg 7).

A & B Registers – Most instructions have A and B operands, but these refer to variable-length numbers stored at the given memory addresses in core memory. When an instruction is executed, operands are fetched from memory one character at a time so the operation can be carried out. These operands are stored in these single-character A and B working registers in the CPU.

These registers are implemented as simple latches, so there's a signal that clears the value in the latch from the previous cycle and enables the latch to hold new data when it becomes available.

In an unusual design style, it should be noted that the only way to get to the A register is through the B register… that is, the B Register *always* latches the last memory read, passing the character just read on to the A Register where it may be latched depending on the type of memory cycle. As a result, the A operand must be read first (at which A and B will both contain the A operand) followed by a read cycle that fetches a new operand into B but leaves A untouched. (See cycle sequencing in Section 4.7)

Arithmetic Unit – There is of course a block to do arithmetic, in this case, addition, subtraction and complement. Unlike most modern computers, the arithmetic unit does only arithmetic, not logical operations like bit-by-bit AND, OR, XOR operations. Arithmetic is done in serial on strings of BCD digits.

The Arithmetic unit utilizes a complex qui-binary adder that disassembles the 0-9 digit into a quinary part (0, 2, 4, 6, or 8) and an even-odd binary part, and adds them separately, as this allowed an error check to be implemented on each operation. See Section 5.4.3.

Op-Code – this single-character register holds the op-code of the current instruction. The output of this register goes to extensive instruction-decode logic.

I-Fetch Ring – A 'ring' in ILD-speak is a chain of Triggers (aka flip-flops, see Section 5.3.3) that count cycles. The I-Fetch Ring is used to track where the machine is in the process of fetching a multi-character instruction from memory. The Ring starts at the beginning with the op-code, and advances one bit down the chain until the last character of the instruction is fetched. Details of the ring can be found at [ILD] IBM-pg 14, pdf-pg 13.

[17] Yup, that's right, Add Three. See Section 6.2.2, Printer Numerology

4.3 Design Methodology

The 1401 is described by three independent sets of engineering drawings:

- Instructional Logic Diagrams (ILD's) are summary drawings, designed to show how various functional blocks in the machine work and how they're interconnected. ILD's have reference points to link to detailed schematics, but omit many details of the underlying implementation to allow more focus on overall function.

  ILD drawing format is relatively familiar as a logic diagram, although some of the symbols are different. Notes in the next section highlight a few aspects of the ILD format. The 1401 is described by about 90 pages of ILDs, formatted on 11x17" pages.

  Not all parts of the machine are described by ILD's; the focus is on instruction execution logic and data path, with not much coverage of I/O functions.

  ILDs were created after the completion of the design as teaching aids.
- Automated Logic Diagrams (ALD's) are the actual schematics that identify every gate, driver, pin, connector and any other components involved. ALD format is rather different from modern logic practice, so there's a section below on how to read an ALD.

  There are about 600 pages of ALDs for the full machine.

  Although they're a bit cumbersome, ALD's should be considered as the primary reference source for how the machine is wired.
- Standard Module System (SMS) plugin-card documents and schematics.

  There are over a hundred types of SMS cards, so understanding the details of a logic schematic often requires recourse to the transistor-level schematic of SMS modules.

ALD's break the design down to logic functions; gates, drivers, expanders, etc, where most functions are implemented in transistors on SMS cards. SMS documents give transistor-level schematics for each logic function on an SMS card.

In this document, there's a section on the ILD description of the machine, and then a subsequent section that outlines how some of the underlying plug-in cards work at a transistor level, and how to read the ALDs.[18]

4.3.1 Design Flow

See [SMS] pdf pg 12 for a block diagram of design flow.

Detailed designs for the machine would have been done with paper and pencil, and then coded on punched cards.

An automation tool would print the ALDs, and also generate a netlist for wiring the backplanes.

Backplanes were initially wired by hand, but later transitioned to Gardner-Denver automated wire-wrapping machines.

The design process did not involve any simulation or automated timing analysis.

There are IBM manuals describing aspects of the ALD preparation process at the following links:

[18] Modularity and abstraction are fine things, but at some point, it may be necessary to follow a block all the way from the 'abstract' ILD representation down to ALD to transistors to figure out what the heck it really does…

http://www.textfiles.com/bitsavers/pdf/ibm/logic/Understanding_Design_Automation_Mar62.pdf
http://www.textfiles.com/bitsavers/pdf/ibm/logic/Mechanization_Of_Engineering_Design_Data_Jan62.pdf
http://www.textfiles.com/bitsavers/pdf/ibm/logic/TR00.770_Automation_Of_Logic_Page_Printing_Jan61.pdf

4.4 Core Memory Refresher

[Add a beginner-intro to core memory]

- Magnetic Magic
- Fixed cycle time
- Destructive Read
- Read-modify write

4.5 Reading the Instructional Logic Diagrams

ILD logic pages would look familiar to current designers, except the symbols are different. Here's an example:

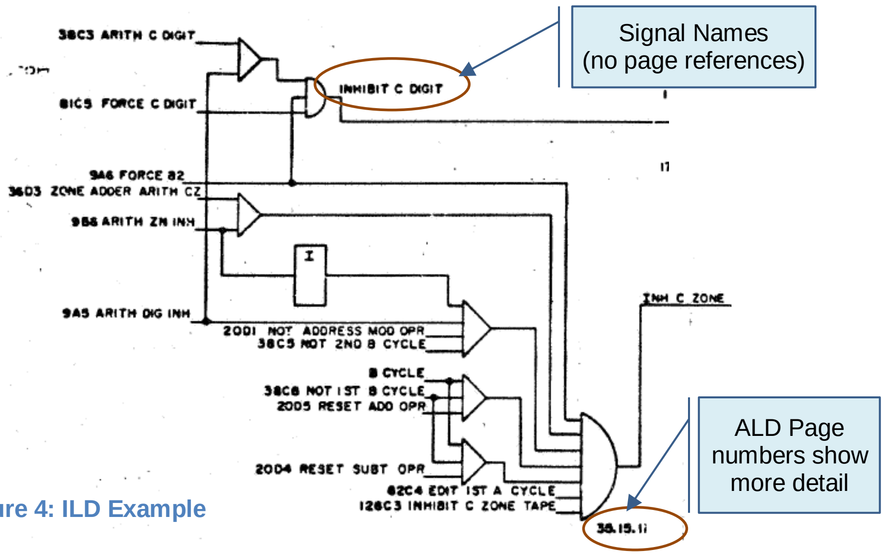


Figure 4: ILD Example

ILD's describe logical function, not wiring, so logic levels are generally assumed to be "positive true", loading and fanout aren't factored in, and exact devices aren't represented. That said, the ILDs often include page number references to the ALDs, where exact logic and implementation can be seen.

Some of the ILD symbols[19] are summarized below.

Gates

OR gates look like AND gates, while AND gates look like op-amps… Here's the decoder chart:

[19] It's been pointed out many times, but IBM was doing this stuff before there were commonly accepted definitions for logic symbols.

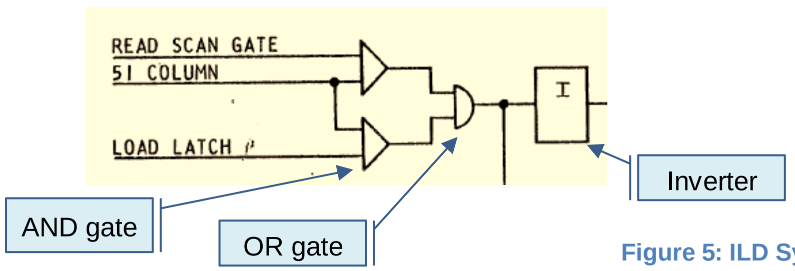


Figure 5: ILD Symbols

Latches

State such as machine registers is usually stored in level-sensitive latches. The ILD's are rather informal about exactly how each kind of latch works, but they're generally cross-coupled gates, either implemented with explicit gates in the ALD, or with special-purpose SMS latch cards.

But in either case, there will usually be Set and Reset inputs. In the data path, a latch will generally be cleared (reset to zero) at a time safely before it's about to be loaded. The data to be loaded into the latch will typically be gated with a clock phase that happens when that data is valid, at which time, it will be asserted onto the Set input of the latch.[20]

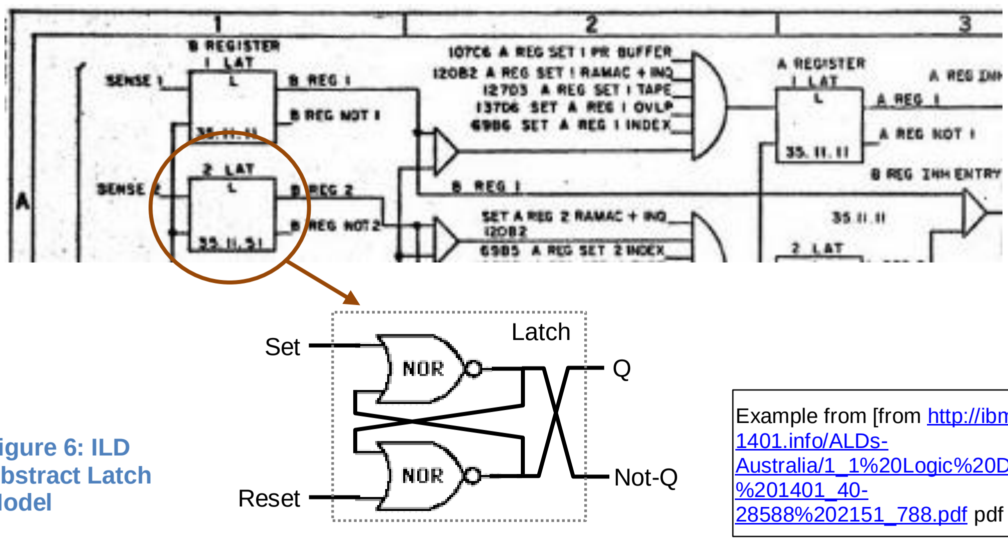


Figure 6: ILD Abstract Latch Model

Example from [from http://ibm-1401.info/ALDs-Australia/1_1%20Logic%20Diagrams%201401_40-28588%2151_788.pdf pdf pg 7]

Section 5.3.2 below gives more detail on the implementation of latches.

Triggers

The 1401 also uses edge-triggered flip-flops, often in timing generators.

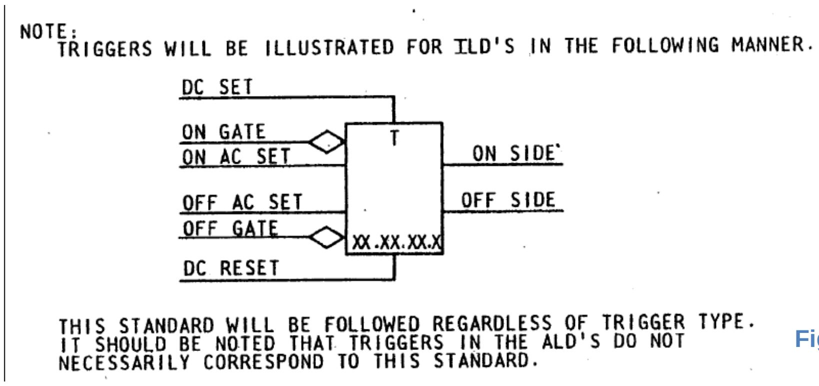
These devices generally don't have direct analogs in current logic design, but the versions found in the 1401 can be thought of as similar to a JK flip-flop, but with independent clock inputs for set and reset.

[20] That pretty much guarantees a glitch on output of the latch, even if its value will not be changed. The moral is: Don't pay attention to the output of a latch except during the times when it's expected to be stable!

Figure 7 shows the definition of a trigger in the ILDs.

- DC Set and Reset act as normal level-sensitive inputs to the flop
- Also, assuming setup and hold times are met, if “On Gate” is active and there’s a rising edge on “On AC Set”, the flop output will be ‘set’
- And if “Off Gate” is active and there’s a rising edge on “Off AC Set”, the flop output will be ‘cleared’

If AC Set and AC Reset are clocked at the same time, the trigger will act something like a JK flop, except that the state is not defined if On Gate and Off Gate are asserted simultaneously and the device is clocked.


NOTE:
TRIGGERS WILL BE ILLUSTRATED FOR ILD'S IN THE FOLLOWING MANNER.

THIS STANDARD WILL BE FOLLOWED REGARDLESS OF TRIGGER TYPE.
IT SHOULD BE NOTED THAT TRIGGERS IN THE ALD'S DO NOT
NECESSARILY CORRESPOND TO THIS STANDARD.

(Picture from [ILD] pdf pg 5, clock generation)

Figure 7: Trigger Definition

Delay Element

[Delay elements are used throughout the clock generation logic… add a note on what they do]

“Delay” box on ILD: Maybe http://files.righto.com/sms/AAF.html

4.6 Clocking and Cycle Timing in the 1401

The 1401 is a synchronous machine, clocked by a master clock source, but it doesn’t use the conventional synchronous logic design popular in current logic design. Instead, the master clock generates many clock phases during a single machine cycle; each phase or subphase is used to advance signals through a series of level-sensitive latches.[21]

There are two aspects to timing in the machine:

- Within an 11.5 usec machine cycle, there are fixed times at which various things happen, e.g, data read from Core, pass through the arithmetic unit, registers cleared or updated.
- Instructions execute through a series of machine cycles in which the instruction itself is fetched, then operands are fetched and stored.

[21] Like Mead and Conway “Two Phase Non Overlapping Clocking” in early Metal Oxide Semiconductor designs, only with a lot more phases!

4.6.1 Clock Timing

Each 11.5 usec machine cycle is divided into 120 units[22] by a master clock generator that creates a plethora of timed pulses representing different sub-phases of the cycle clock. Throughput the design, timing signals are represented by the time interval during which they are active; for example, the signal "Time 014-030" is goes active around 1.5 usec after the start of the cycle, and stays active until about 3.0 usec into the cycle (and happens to be used to latch data from Core into the B register).

Clock Logic is summarized on a page in the ILDs. See [ILD] pdf-pg 5.

Figure 8 shows the basic timing generation for the machine, although there are many other sub-phases generated for specific functions.

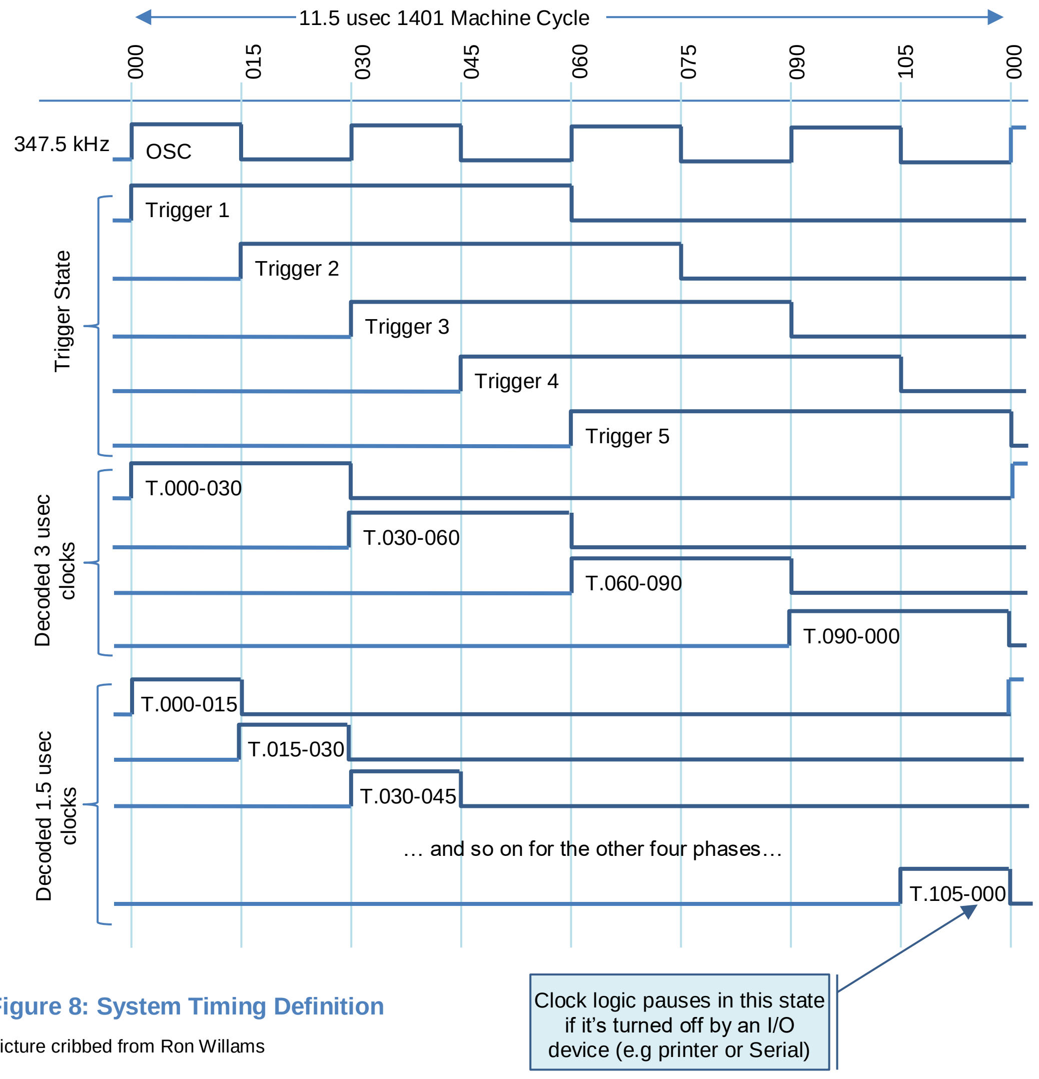


Figure 8: System Timing Definition

Picture cribbed from Ron Willams

[22] That means that a unit of time is approximately 0.1 usec

The time intervals on clocking signals are approximately in units of 0.1 usec (i.e., 120 units in 11.5 usec).

Core memory is destructive-read, so each 11.5 usec cycle will read a memory location, and either write the same data back, or write an updated operand back to the location.

During I/O ops, timing / clocking can be paused by the I/O device (printer and 'serial' for sure, not sure about the tape controller yet).

That means there's a "synchronizer" somewhere, not that it was recognized as such

Time T000 is presented via a connection point on the operator's panel to allow synchronization of an oscilloscope during debug. (See Section 8.2)

It's possible the actual clock source for the entire machine is on pdf pg 25, IBM's pg 31.10.11.2

http://ibm-1401.info/ALDs-Australia/2_7%201401%2040-28588%202151_788%2021-32.pdf

[can someone confirm?]

The master oscillator is an SMS card named "RK": Alloy-Oscillator 347.5KC Free Running (Crystal) (http://files.righto.com/sms/RK.html)

That makes the machine cycle 1 / (347.5 / 4) = 11.51 usec. The machine cycle rate is 86.8 kHz.[23]

4.6.2 Data Path Timing

During each 11.5 microsecond machine cycle, a number of functions may be completed simultaneously:

- A memory read and write to a single address, specified by the Main STAR (See Figure 3, Figure 9)
- An arithmetic operation
- Computation of a next address for either the program counter or an operand address register (e.g. A-STAR or B-STAR). Address registers are three characters long, so this calculation is done in three 3-usec steps, one character at a time.

Figure 9 below shows the flow of data through the pipeline; Figure 10 gives approximate timing.

[23] Some IBM marketing material seems to say that the master clock is 350 kHz, yielding a cycle rate of 87.5 kHz, or 11.5 usec. But engineering docs seem to all agree on 347.5 kHz as the master clock.

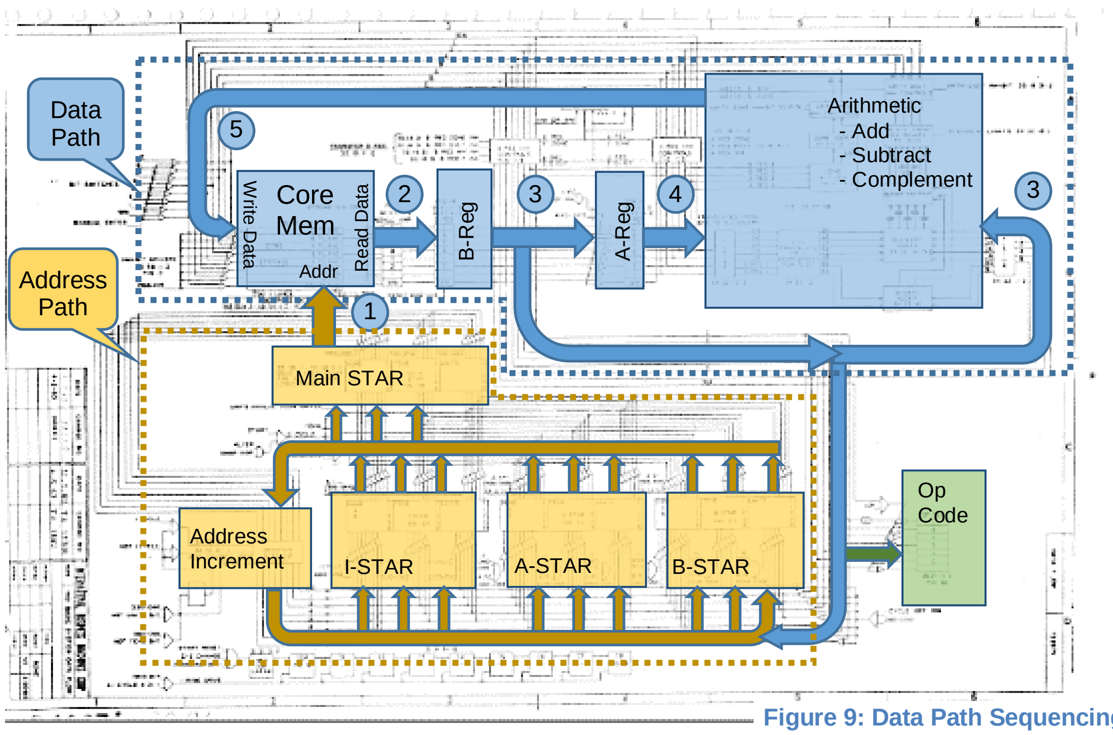


Figure 9: Data Path Sequencing

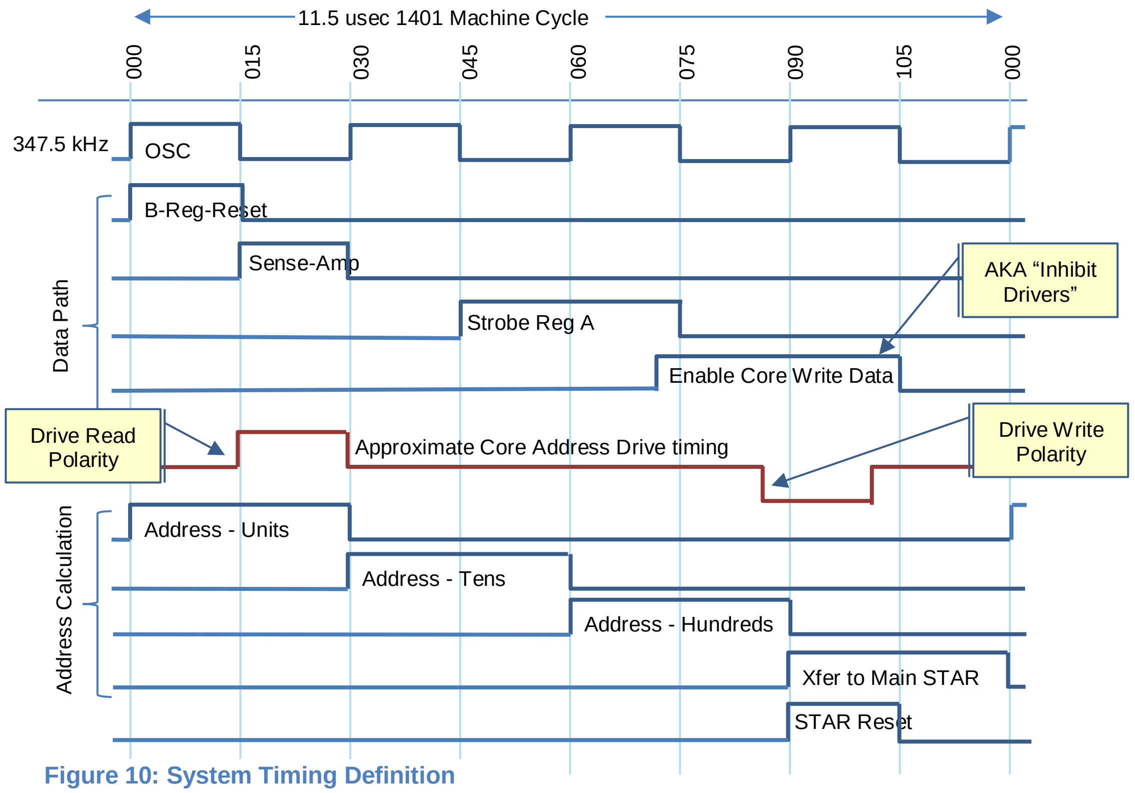


Figure 10: System Timing Definition

4.7 Instruction Cycle Sequencing

With no cache, no prefetch, no branch prediction or any of the other hardware features that make current machines so fast, the 1401 execution rate is very predictable.

Each instruction consumes a number of machine cycles to complete:

- Fetching the instruction itself may take up to eight cycles, one to fetch the op code, plus up to seven more for two operand addresses and an optional modifier
- Then execution of the instruction may take (possibly many) cycles, usually alternating between an A operand fetch and a B operand fetch and store.

See G24-1477 and R25-1496 for cycle-by-cycle timing for each instruction type

See [ILD2] for cycle-by-cycle timing diagrams for many instruction types

5. Hardware Design

So you're ***sure*** that your program is right and the hardware isn't executing it properly… now what?

- Find the function in question in the ILDs or manuals
- Find the bit in question on the ALDs (schematics). Section 5.2 shows how to read the diagrams.
- Follow the packaging breakdown in Section 5.1 to find the pin
- Snoop the bit with a scope probe
- Find the bug in your code.

5.1 Physical Layout and Packaging

See http://ibm-1401.info/1401CPUPhysLayout.html for annotated photos of the machine showing the mjor functions packaged in each card gate.

See http://ibm-1401.info/1401UnitPluggingChart.html for physical location and types of all SMS cards.

A low-end 1401 with few options could be packaged in a single frame (also called "two cubes").

5.1.1 Locating Cards in a 1401

Components of the 1401 are organized as Card Gates which hold SMS card. Up to eight Card Gates can fit together into a Module (aka 'cube'), with two modules per frame.

Figure 11 shows how the Card Gates are identified in a 1401. (See Figure 16 for a breakdown of the complete location identifier, including the Card Gate identification, as found on logic diagrams.)

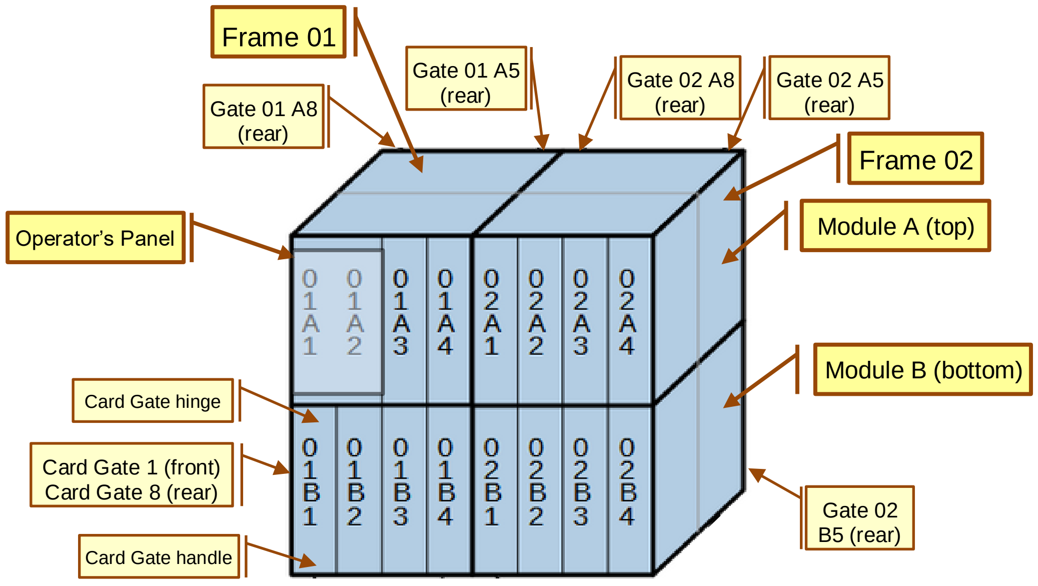


Figure 11: Location of Card Gates

Card gates are hinged on one edge, allowing them to swing open for maintenance. The gates in the upper module swing downwards, those in the lower module swing upwards. Within a Card Gate, there are up to 6*26=156 SMS card sockets, identified by row and column number.

Figure 12 shows a schematic of a 1401 frame with all the Card Gates open, giving column and row identifiers for front and back. This is an elevation view, looking at the left end of the machine (the end near the operator's console), with dotted lines showing clearance required for opening and closing card gates. Note that lower gates are hinged at the top, while upper gates are hinged at the bottom. Since cables exit the gates at the hinge point, this arrangement minimizes the length of wires through the chassis.

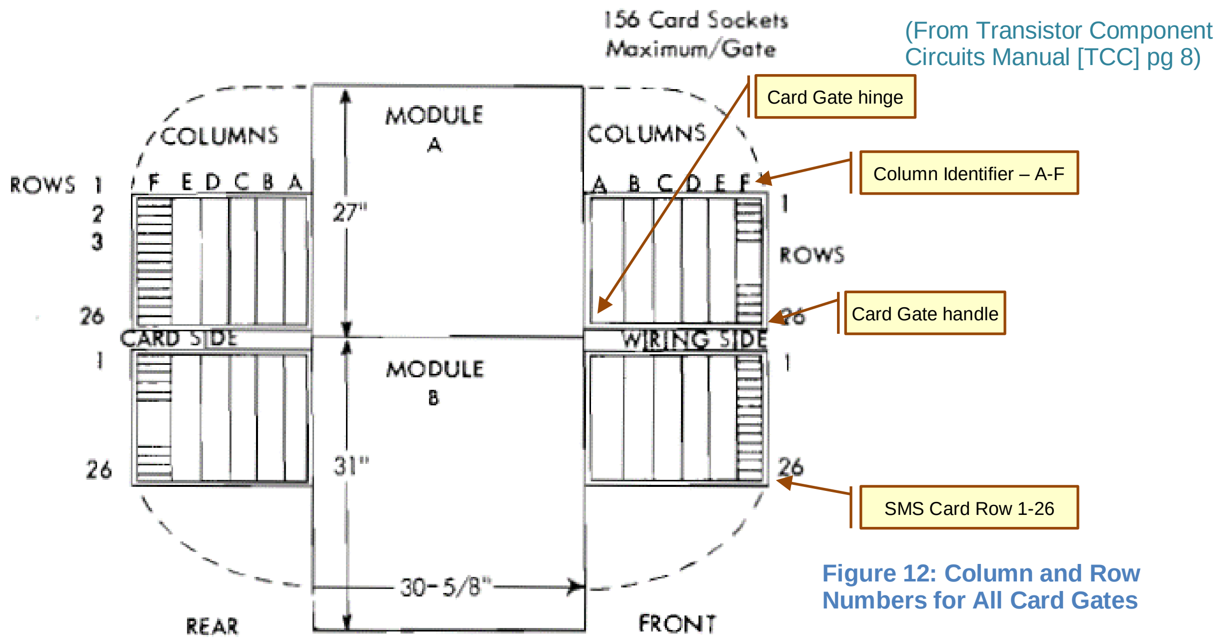


Figure 12: Column and Row Numbers for All Card Gates

Figure 13 shows a photo of the Real Thing…

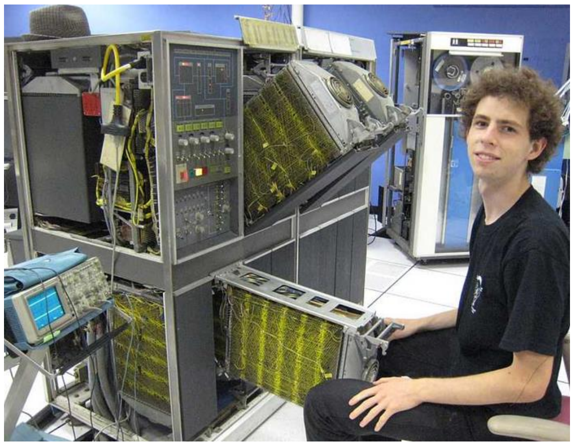


Figure 13: Upper and Lower Card Gates Opened for Service

5.1.2 Placement of Logic Functions

Figure 14 below, from the Field Maintenance Manual ([FMM] pdf pg 11), shows locations logic functions in card gates and the corresponding ALD pages.

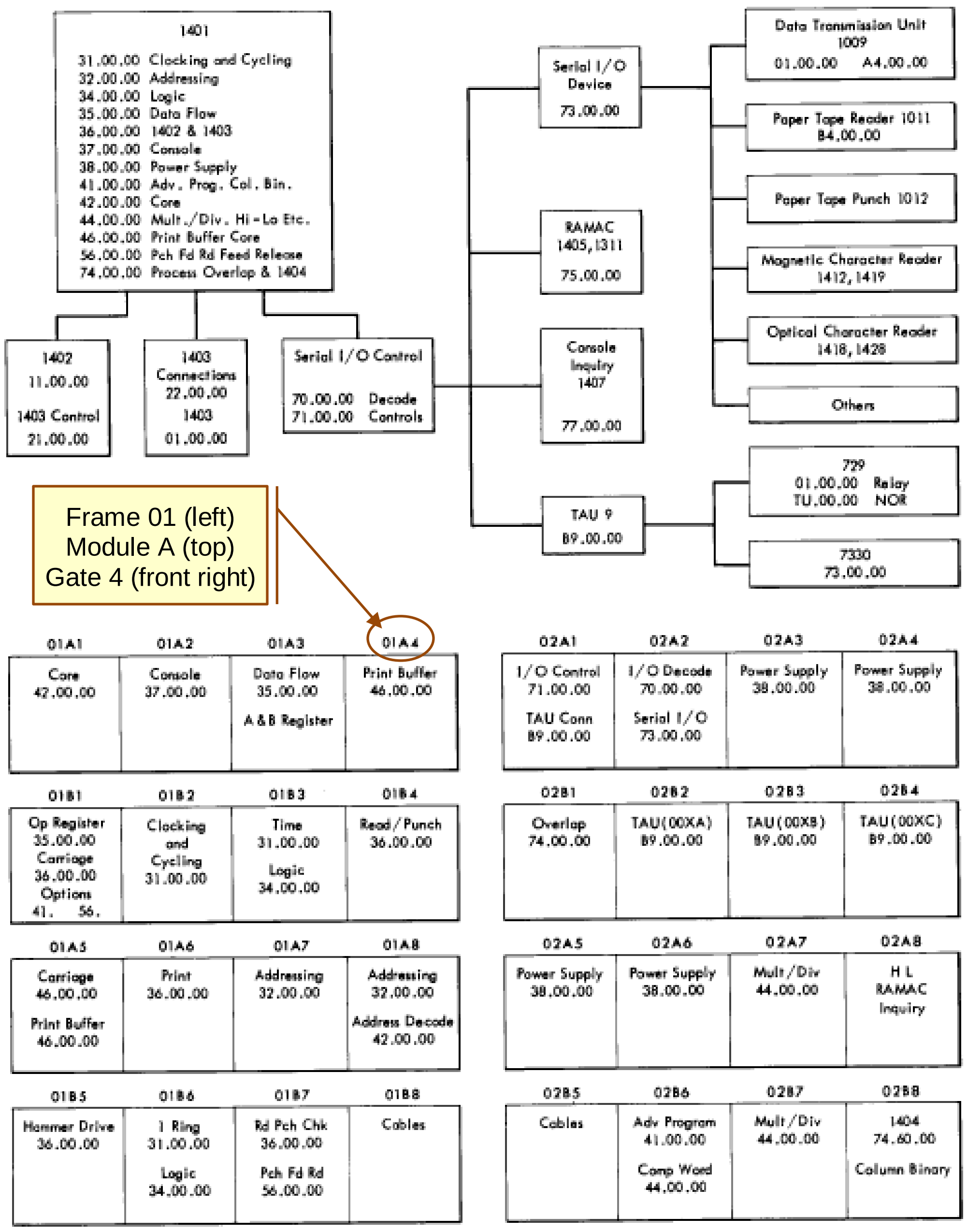

1401
31.00.00 Clocking and Cycling
32.00.00 Addressing
34.00.00 Logic
35.00.00 Data Flow
36.00.00 1402 & 1403
37.00.00 Console
38.00.00 Power Supply
41.00.00 Adv. Prog. Col. Bin.
42.00.00 Core
44.00.00 Mult./Div. Hi-Lo Etc.
46.00.00 Print Buffer Core
56.00.00 Pch Fd Rd Feed Release
74.00.00 Process Overlap & 1404
1402 11.00.00 1403 Control 21.00.00
1403 Connections 22.00.00 1403 01.00.00
Serial I/O Control 70.00.00 Decode 71.00.00 Controls
Serial I/O Device 73.00.00
RAMAC 1405, 1311 75.00.00
Console Inquiry 1407 77.00.00
TAU 9 B9.00.00
Data Transmission Unit 1009 01.00.00 A4.00.00
Paper Tape Reader 1011 B4.00.00
Paper Tape Punch 1012
Magnetic Character Reader 1412, 1419
Optical Character Reader 1418, 1428
Others
729 01.00.00 Relay TU.00.00 NOR
7330 73.00.00
Frame 01 (left) Module A (top) Gate 4 (front right)
01A1 Core 42.00.00
01A2 Console 37.00.00
01A3 Data Flow 35.00.00 A & B Register
01A4 Print Buffer 46.00.00
01B1 Op Register 35.00.00 Carriage 36.00.00 Options 41. 56.
01B2 Clocking and Cycling 31.00.00
01B3 Time 31.00.00 Logic 34.00.00
01B4 Read/Punch 36.00.00
01A5 Carriage 46.00.00 Print Buffer 46.00.00
01A6 Print 36.00.00
01A7 Addressing 32.00.00
01A8 Addressing 32.00.00 Address Decode 42.00.00
01B5 Hammer Drive 36.00.00
01B6 1 Ring 31.00.00 Logic 34.00.00
01B7 Rd Pch Chk 36.00.00 Pch Fd Rd 56.00.00
01B8 Cables
02A1 I/O Control 71.00.00 TAU Conn B9.00.00
02A2 I/O Decode 70.00.00 Serial I/O 73.00.00
02A3 Power Supply 38.00.00
02A4 Power Supply 38.00.00
02B1 Overlap 74.00.00
02B2 TAU(00XA) B9.00.00
02B3 TAU(00XB) B9.00.00
02B4 TAU(00XC) B9.00.00
02A5 Power Supply 38.00.00
02A6 Power Supply 38.00.00
02A7 Mult/Div 44.00.00
02A8 H L RAMAC Inquiry
02B5 Cables
02B6 Adv Program 41.00.00 Comp Word 44.00.00
02B7 Mult/Div 44.00.00
02B8 1404 74.60.00 Column Binary

Figure 14: Card Gate Assignments

There’s a diagram of cabling between units in the Field Maintenance Manual [FMM]http://ibm-1401.info/PPierce-ibm-225-6487-3.pdf pdf pg 19

5.2 Reading the ALD’s

Automated Logic Diagrams are a bit harder to read than the ILDs.

ALDs are printed on line printers, so everything is either a box or a line, i.e, all logic functions are represented as boxes with notations showing the function, and wires are shown as horizontal and vertical lines. Because ALDs were designed to be printed on line printers on 11x17 paper, the format is highly structured, with a five-by-seven grid of locations for logic blocks and predetermined channels for routing wiring. Figure 15 shows a sample from an ALD page.

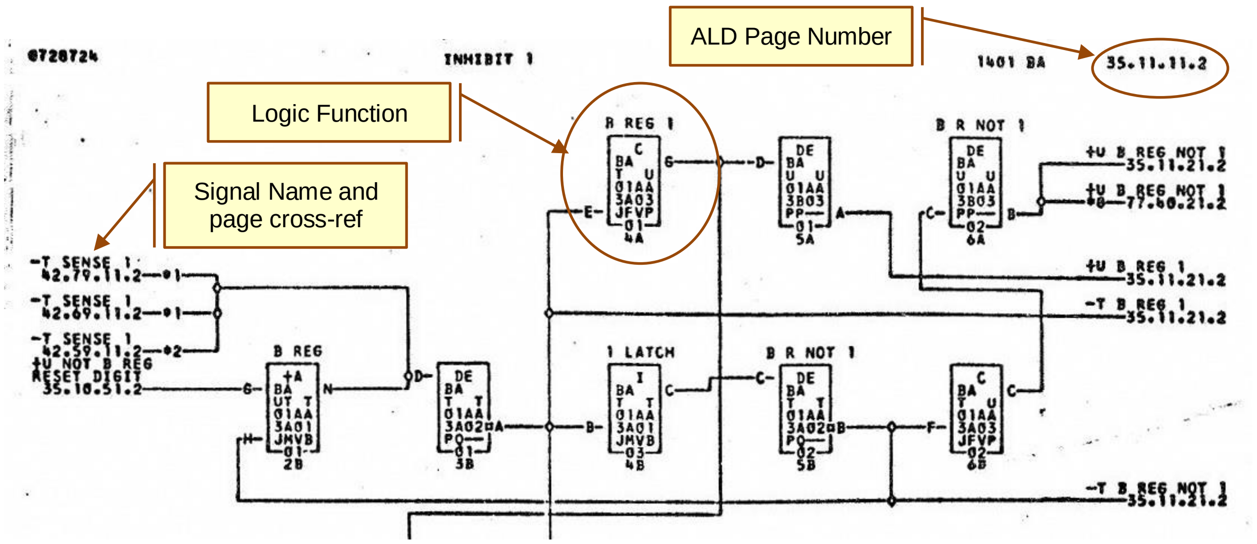


Figure 15: Sample ALD from Data Path

ALD page numbers are always four groups of digits separated by dots (e.g. 31.10.11.2), allowing grouping of functions and insertion of page numbers.[24]

Figure 16 gives the ALD Logic Block secret decoder ring. The standard notation in each block gives its function, pin numbers, logic levels, circuit type and physical location in the machine.

[24] “Like the Dewey Decimal System”, suggests Ignacio Menendez. See pg 13 of [SMS] for the official story on page numbers.

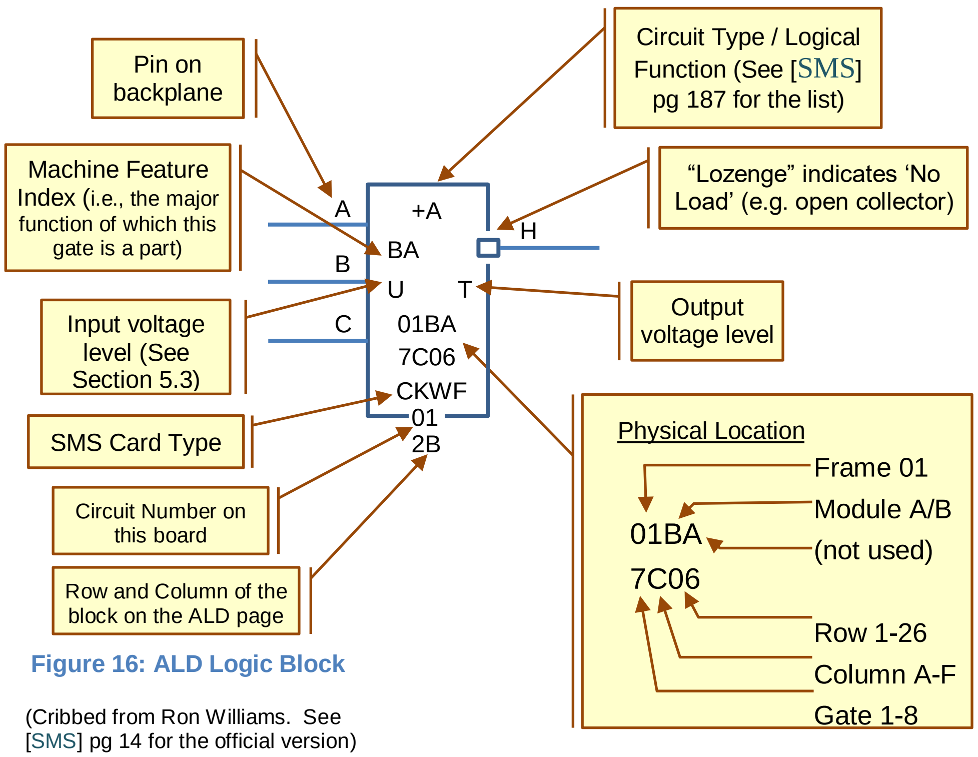

Pin on backplane
Machine Feature Index (i.e., the major function of which this gate is a part)
Input voltage level (See Section 5.3)
SMS Card Type
Circuit Number on this board
Row and Column of the block on the ALD page
Circuit Type / Logical Function (See [SMS] pg 187 for the list)
"Lozenge" indicates 'No Load' (e.g. open collector)
Output voltage level
A
+A
BA
B
U
T
H
C
01BA
7C06
CKWF
01
2B
Physical Location
Frame 01
Module A/B
01BA
(not used)
7C06
Row 1-26
Column A-F
Gate 1-8

Figure 16: ALD Logic Block

(Cribbed from Ron Williams. See [SMS] pg 14 for the official version)

For most gates, outputs on the lower half of the box are "In Phase" (or "In Ø"), while outputs at the upper half are Out of Phase (Out Ø). For example, for an AND gate, the in-phase output would provide AND, while the out-of-phase output would provide NAND.

A decoder chart for the "Machine Feature Index" field in an ALD gate can be found at http://ibm-1401.info/1401UnitPluggingChart.html. In this instance "BA" stands for "BAsic", i.e., non-optional data path.

There's a chart of all the Logical Function symbols (e.g. +A, DE, C, etc) on pg 187 Appendix D of [SMS]. In this case, "+A" means that the gate produces an output when both inputs are high.

5.2.1 Page Cross-References

The 1401 schematics are divided into hundreds of pages, so signals routinely cross from one page to another. On each schematic sheet, inputs to logic on the page are arranged along the left edge, while outputs are placed on the right edge of the sheet.

Each signal that crosses from one page to another is given a name, and a page number for where to find the matching signal.

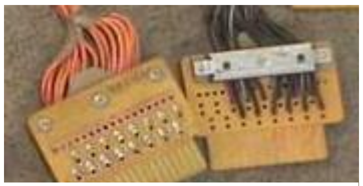


Figure 17: Edge Connectors (aka Paddle Cards)

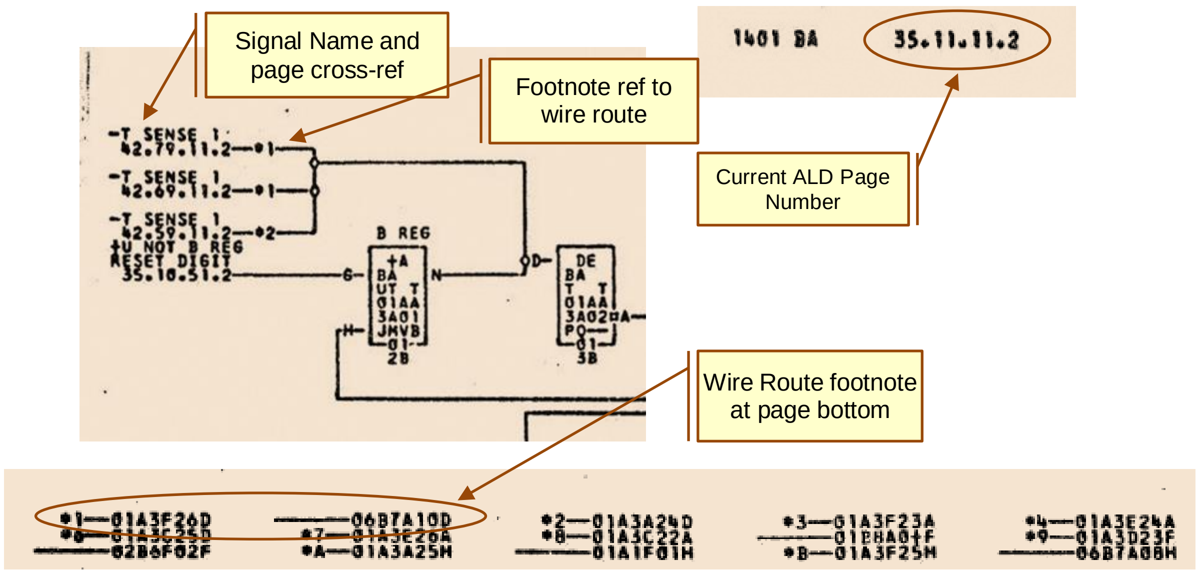
The signals along the left and right edges of the page show how to trace from one pin on a logic block to another pin on a different logic block on a different page. But that doesn't show how the signal actually gets there…

The 1401 logic is divided into dozens of Card Gates (e.g., see Section 5.1.1). Each card gate is an independent wire-wrapped backplane, which could be replaced. Connections arrive and depart the Card Gate via Edge Connectors,[25] i.e., reserved SMS card locations on the edge of

[25] Ok, so "most" signals arrive and depart via Edge Connectors. Engineering changes, field options and entropy in general result in all kinds of ad-hoc wiring in the actual machine…

the backplane that are populated by cable connectors instead of logic gates. The edge connectors are cabled together with a giant wiring harness to take signals from one logic gate, through an edge connector, to the wiring harness, to another edge connector, to the destination logic gate.

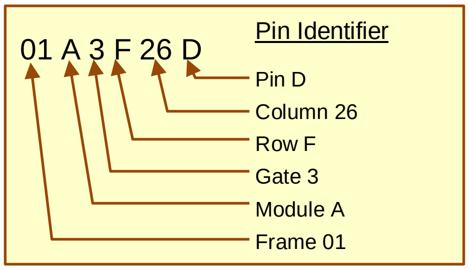
The edge connector pins used to transit signals from one card gate to another are identified on the ALDs in footnotes at the bottom of the page. Figure 18 shows an example.



(For the full page image, see http://ibm-1401.info/ALDs-VSnyder-Australia/3_7/728724.pdf)

Figure 18: ALD Wiring Path Footnote

Each entry in the footnote gives a pin identifier for the edge connector (or other types of connectors sometimes). In the 1401, the pin is identified by an eight-digit string[26] (almost) the same as the location identifier in the logic block shown in Figure 16.



In this example, the signal "-T SENSE 1" is heading from page 35.11.11.2 to page 42.79.11.2. But the page number is followed by " ***1**", indicating that it crosses from one card gate to another.

- Looking at the logic block, we see the signal terminates at a DE-type gate at location 01AA-3A02 (Frame 01, Module A, card gate 3, column A, row 02)
- At the bottom of the page, footnote *1 shows that the edge connector is in location 01A3 F26, and the signal passes though Pin D.

[26] Note that the string is numeric/alpha/numeric/alpha/numeric/alpha.

- A cable then takes the signal to 06B7 A10D. [Frame 06, you say? Yes, Frame 06, Module B is the 1406 memory expansion chassis(!)]

Connection points which are not part of conventional 1401 processor card gates are assigned locations that don't completely fit with the physical layout in Figure 11:

- Frame 06 is the 1406 expansion memory chassis
- Frame 02 "card gate" A0 is a connector panel used to route signals between frame 01 and 02, simplifying assembly of the two-cube processor

[And are there others?]

See pages 13 & 14 of Standard Modular System manual [SMS] for the details on signal cross-references.

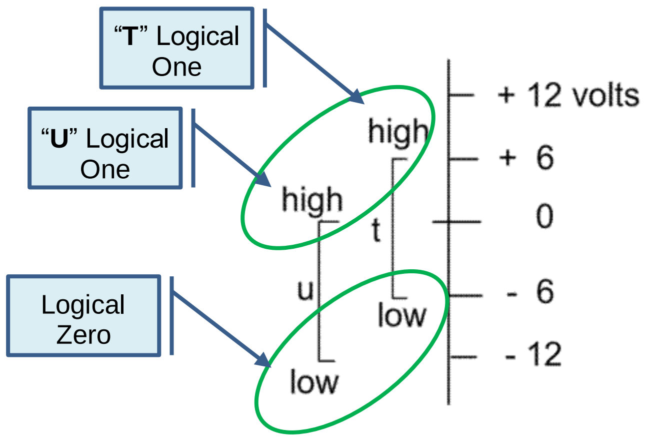
There's also an internal memo dated Oct 27, 1959 outlining guidelines for documenting "feed through" wire routing at http://ibm-1401.info/PokoskiJ-fdthru-Oct-59.pdf

5.3 CTDL Logic Family

While IBM developed a number of different transistorized logic families during the SMS (Standard Module System) era, most of the 1401 processor is implemented with one logic family, CTDL (Complementary Transistor Diode Logic),[27] using NPN and PNP Germanium Alloy junction transistors. About 3,000 SMS cards are used to implement an IBM 1401.

Considerable detail on CTDL can be found in two references; see [SMS] and [TCC] in the references.

The "complementary" part of CTDL refers to the practice of using two variants of logic gate, called P and N gates, using PNP and NPN transistors, with two different sets of voltage levels. "T" levels swing between +6v and -6v, "U" levels switch between 0.0v and -12v, Most gates have either T or U levels as inputs, and the opposite levels as outputs, so a typical circuit would have chains of logic of alternating P and N type gates.



Both U and T voltage levels feature a nominal 12v swing between zero and one, with a switching point approximately at the middle of their ranges, offering considerable noise margin

Figure 19: CTDL Voltage Levels

-------- Forwarded Message --------

Subject: Re: voltage levels

Date: Mon, 30 May 2016 13:51:11 -0700

[27] Except for the tape controller (TAU), which was re-used from a 709 computer design [right? Which one?]. See Section **Error! Reference source not found.**.

From: Carl Claunch <carlclaunch51@gmail.com>

To: Guy Fedorkow <guy.fedorkow@gmail.com>

>When I see a signal "-U TIME 000-075", what voltage is the "active" level?
>That is, I'd assume that during time 000-075, it would be "active", but the rest of the time, not.

-U is the same as NOT U

An inverted logic signal so it has the value 0 when in time 000 to 075 and binary value 1 when not in that range of times.

Thus, the signal -U TIME 000-075 is at -12V when the machine is in the early parts of the machine cycle and will be at 0V when in the later part of the cycle.

T is +6V at binary 1 and -6V at binary 0. That means a +T signal is +6V when on but a -T signal is -6V when on.

The key to keeping this straight is to mentally replace -U TIME 000-075 with:

(the same text but with a negation bar across the top replacing the minus sign)

Using the more modern representation, you know that this is inverted sense, with value 0 only when the signal is true.

Carl

The following section shows how CTDL gates are made.

5.3.1 Gates

There are two types of ordinary gates

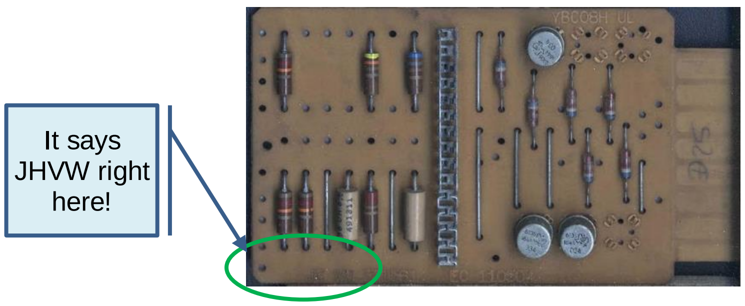
- An N-type[28] logic block takes T-level inputs, uses PNP transistors, and produces U-level outputs.
- A P-type logic block takes U-level inputs, uses NPN transistors, and produces T-level outputs.

5.3.1.1 N-Type Gates

Figure 20 shows a sample N-type 2-input AND gate (which IBM would call a "-A", or Minus AND, now known as a NOR gate, one that generates a logical zero output when either inputs are at logical one.) This gate accepts T levels on its inputs, and generates a U output level.

There are a zillion variations, but this particular gate type is identified as "JHVW"; this identifier is used on the logic diagrams (see Reading an ALD), and is embossed on the PCB.

The photo below shows the physical card, made of phenolic PCB material, with copper traces on one side and components on the other. Contacts to plug into the backplane are plated with gold to resist corrosion.



[28] The "gate type" corresponds to the doping of the base of the transistor; so an **N**-Type gate uses a P**N**P transistor.

The following image shows the IBM representation of the card:

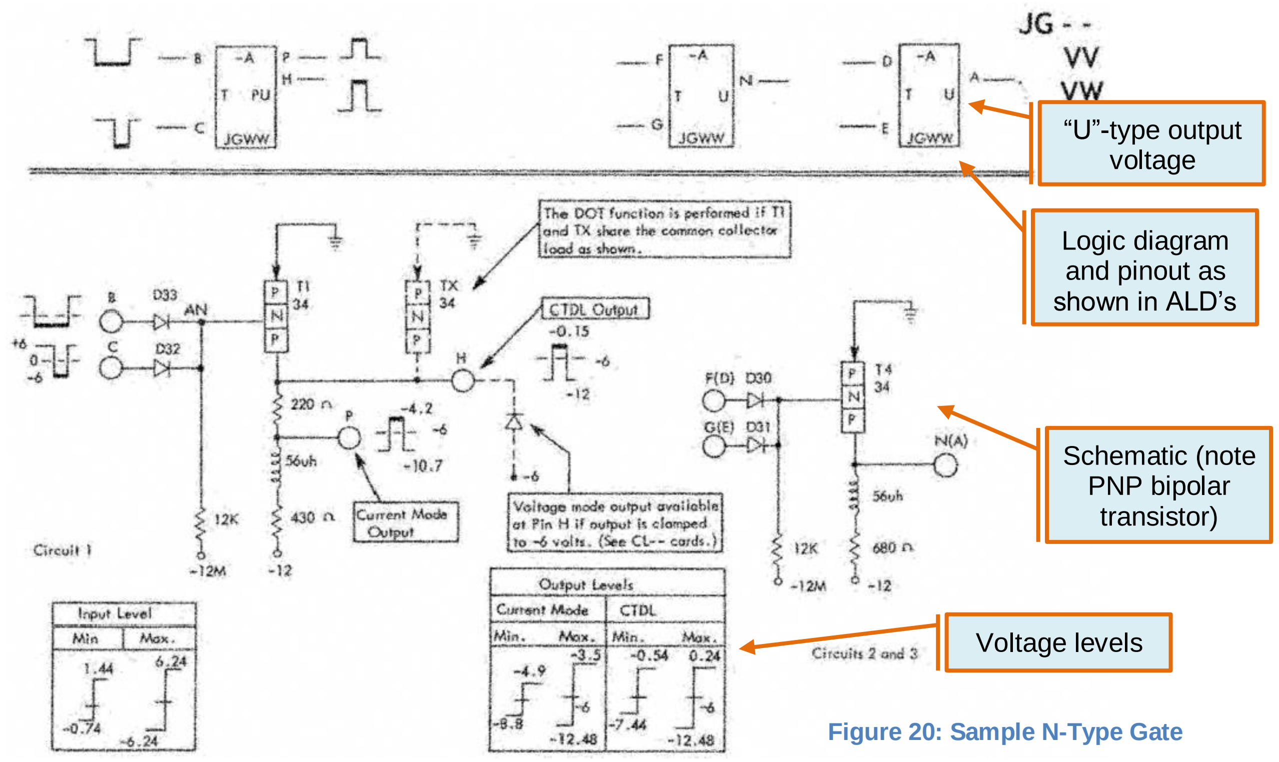
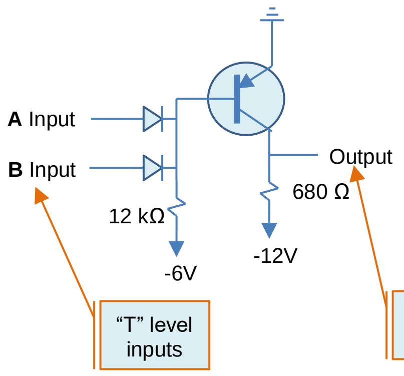

JG - -
VV
VW
"U"-type output voltage
Logic diagram and pinout as shown in ALD's
Schematic (note PNP bipolar transistor)
Voltage levels
JGWW
The DOT function is performed if T1 and TX share the common collector load as shown.
CTDL Output
Current Mode Output
Voltage mode output available at Pin H if output is clamped to -6 volts. (See CL-- cards.)
Circuit 1
Circuits 2 and 3
Input Level
Min
Max.
Output Levels
Current Mode
CTDL
T1 34
TX 34
T4 34
220 Ω
56uh
12K
430 Ω
680 Ω
-12M
-12

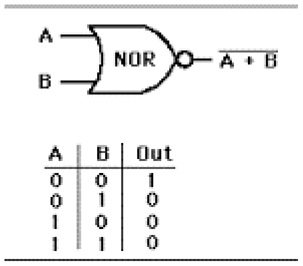
Figure 20: Sample N-Type Gate

From http://files.righto.com/sms/JGVW.html

Ignoring all the analog niceties, here's the simplified version of this gate. The output of the gate is 0V (aka +U) when input **A** and input **B** are both -6v [29] (aka -T)



A	B	Out
-6v	-6v	0v
-6v	+6v	-12v
+6v	-6v	-12v
+6v	+6v	-12v



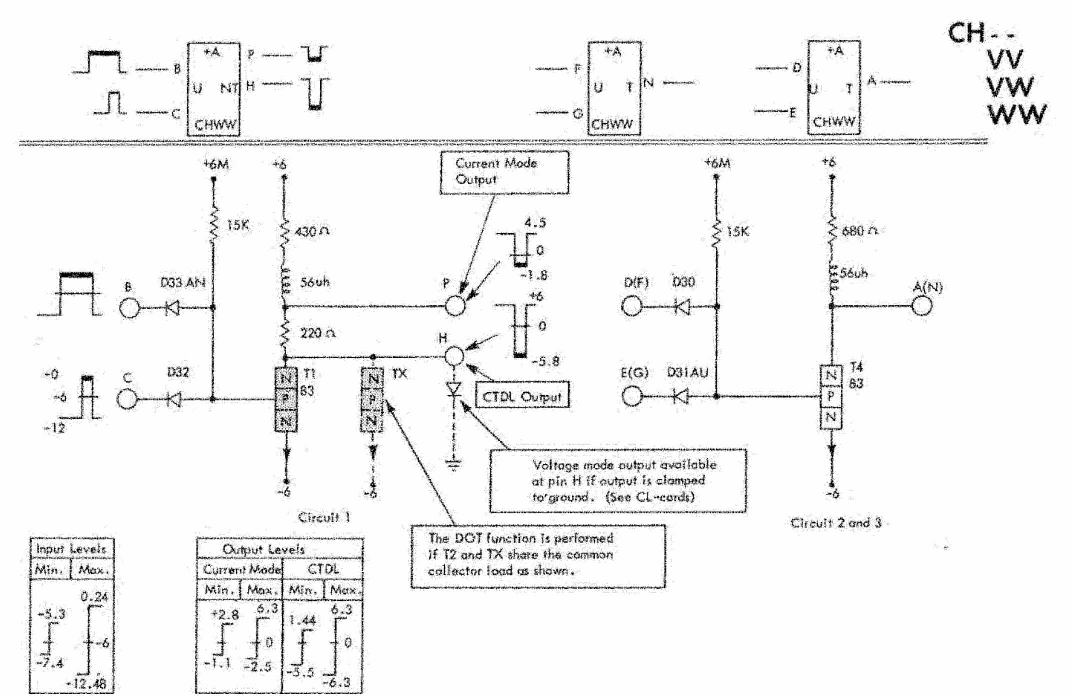
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

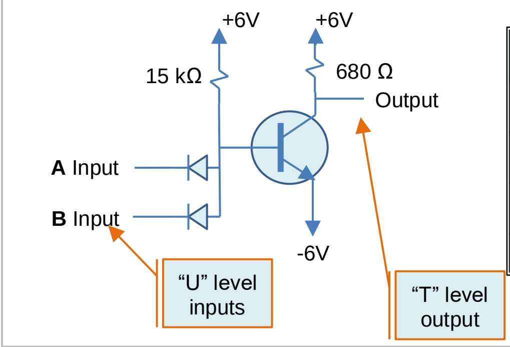
Figure 21: Simplified N-Type Schematic

5.3.1.2 P-Type Gates

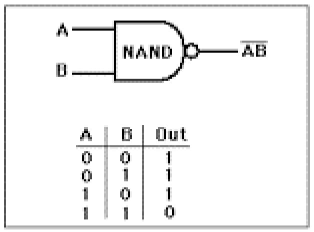
The N-Type gate takes T inputs and produces U outputs, and implements a NOR function. The P-Type gate does the reverse: U-Type inputs, T-Type outputs and NAND functionality.

[29] Put that 0v to +5v TTL stuff out of your mind; logic levels in the 1401 tend towards the negative voltages.





A	B	Out
-12v	-12v	+6v
-12v	+0v	+6v
+0v	-12v	+6v
+0v	+0v	-6v



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Figure 22: Simplified P-Type Schematic

5.3.1.3 "Dot" Gates

Transistors don't come cheap, so why not do logic with wire instead of transistors where possible…
The "dot" gate yields an implicit logic function by wiring output of compatible gates together.
The output of the gate below is 0V when inputs **A** and **B** are both -6v, or inputs **C** and **D** are both -6v.

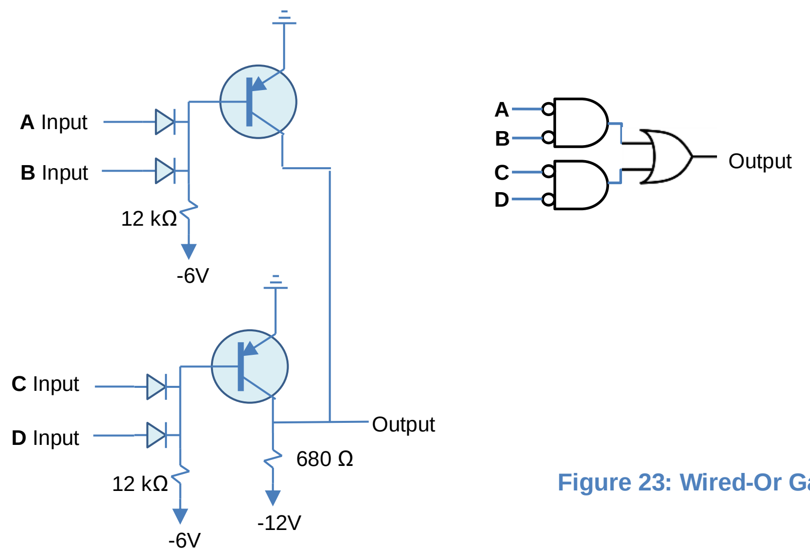
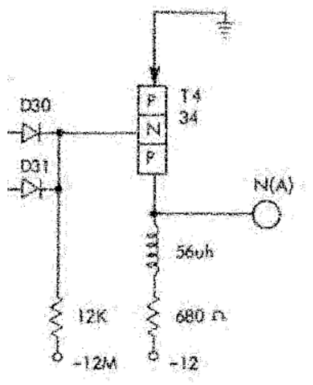


Figure 23: Wired-Or Gate

Gates which are suitable for wired-OR outputs have no internal load resistor, and may be indicated by a “Lozenge” symbol on the output pin on the ALD’s.

It’s important to remember that in the 1401 ALD’s, wired-OR functions may extend across many pages… For example, a bit in a data path may be wired as a ‘bus’, where any one of a number of sources can pull the signal high.

5.3.1.4 About That Inductor



Astute readers may note that many of the logic gates have small inductors in series with the load resistors that pull signals to their opposite state when the switching transistor is turned off (i.e, pulling outputs low in a N-Type gate). This inductor works to speed the transition from the On to Off state for the gate (i.e., the falling edge for an N-Type gate) by providing a constant current supply to charge any capacitance on the output of the gate. The IBM Transistor Component Circuits manual ([TCC] pg 45) refers to this inductor as a *peaking coil*.[30]

A simple Spice simulation shown in the next two figures illustrates the function of the load inductor; the green trace with the load coil transits the switching region more quickly than the blue trace, which relies on the resistor alone to pull the output low.

[30] As would any analog designer.

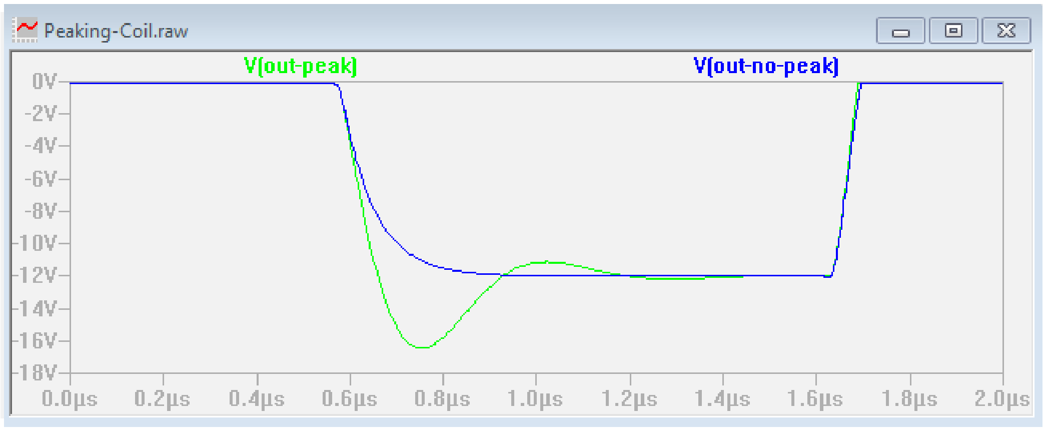


Figure 24: Simulation Output

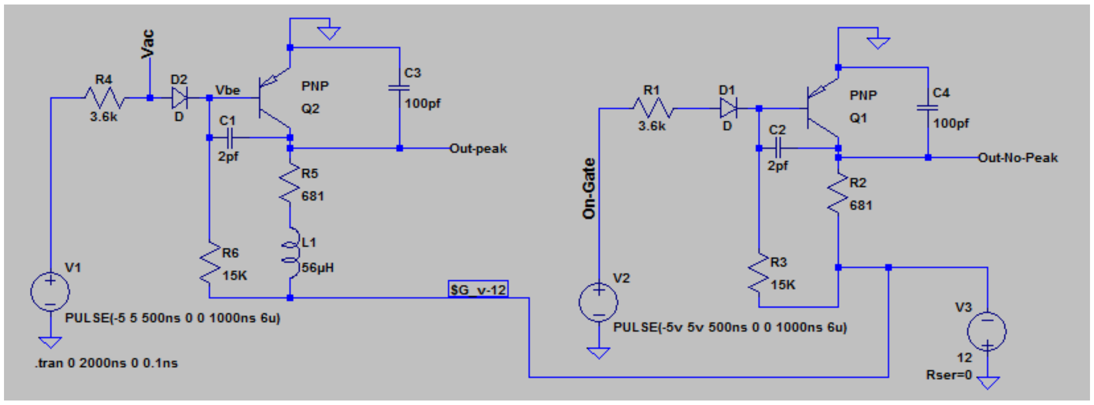


Figure 25: Simulation Schematic

5.3.2 Latches

Working state is stored in 1401 logic using Latches and Triggers, two variants of flip-flop circuits.[31]

Latches are level-sensitive, without a clock input, and may be implemented with individual gates wired together, or with single-purpose SMS cards that implement the latch function. In current terms, these would be seen as SR flip-flops.

Figure 26 shows the schematic of a dual latch SMS card used in the 1401 data path.

[31] In the 1401, Latches are level-sensitive and Triggers are edge-sensitive; this simple naming was not always true in later IBM machines, where the definitions started to blur. See Carl's note, Section 10.1.

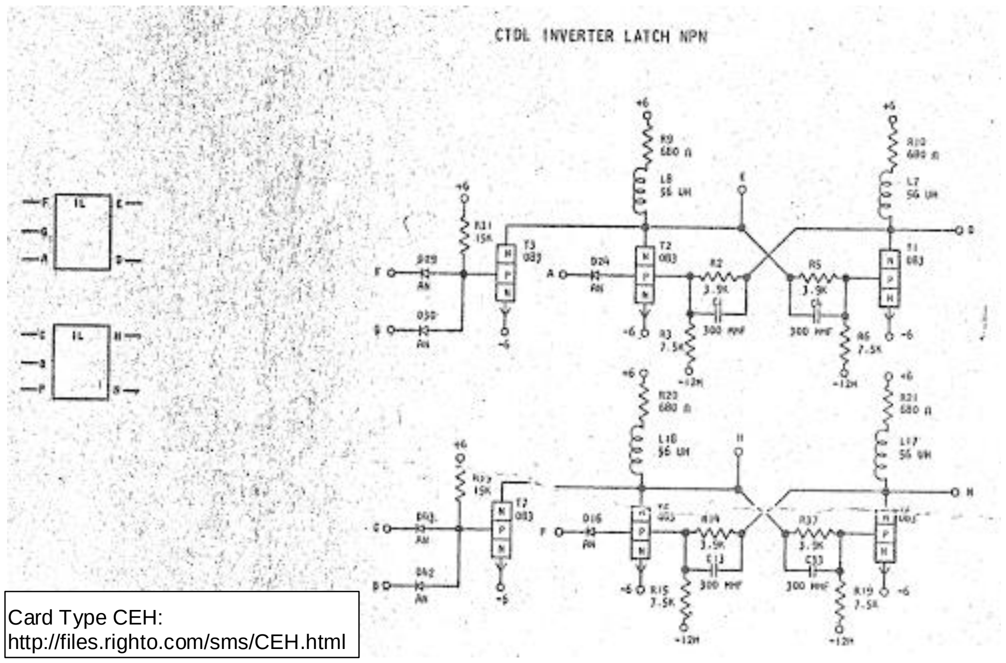


Figure 26: Level-Sensitive Latch

When looking at latches in the 1401 ALDs, remember that the outputs are not buffered, and a latch can be driven to a new stable state by forcing the output. Put another way, latches can be set and reset by their designated logic inputs, but also by (sometimes hard to see) wired-OR gates on their outputs.

As is pointed out in [FMM], debugging a latch that's triggered by a wire-OR yanking its output to a new state can be tough, as the new state of the latch back-drives the input that just set it to that state.

5.3.3 Triggers

Latches are level-sensitive and can be constructed with just cross-coupled latches. But edge-triggered flops need a bit more care.

Figure 27 shows an edge-sensitive trigger circuit.

The trigger has normal Level-sensitive Set and Reset inputs called DC Set and DC Reset

But it also has AC-Set/Reset and Gate inputs on each side.

- If the Gate input is 'high' (that would be a zero-volt "U" level) when the AC-Set/Reset input transitions from low-to-high, the flop will change state. [32]
- If the Gate input is 'low' (-12V), a rising edge on the AC-Set/Reset input has no effect.

The detailed waveforms for this magic trick are shown in Figure 28.

[32] Note the very long setup time – 3.75 usec according to the spec, though clock-to-q is under one usec.

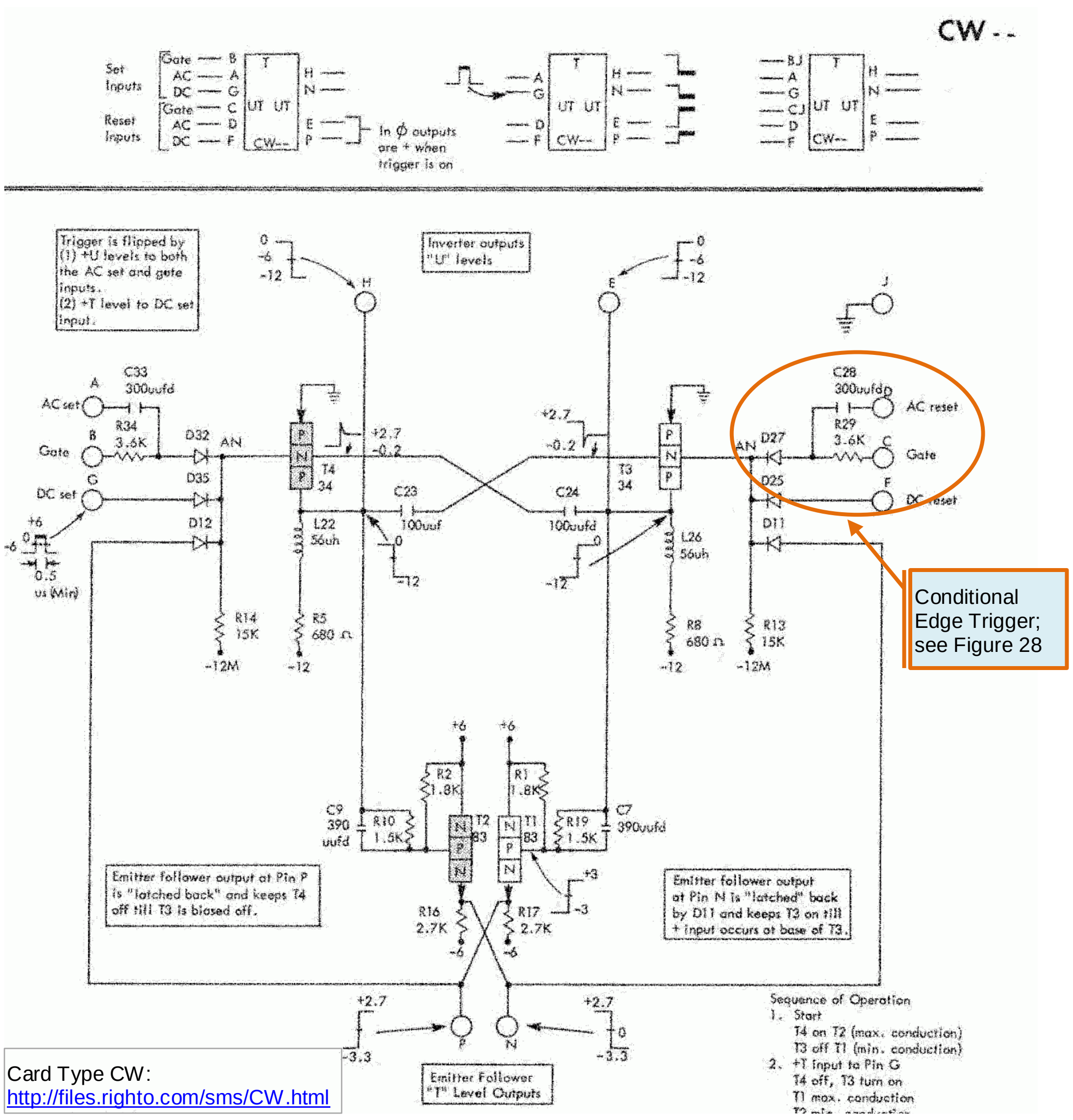

CW - -
Set Inputs
Reset Inputs
In ϕ outputs are + when trigger is on
Trigger is flipped by (1) +U levels to both the AC set and gate inputs. (2) +T level to DC set input.
Inverter outputs "U" levels
AC set
Gate
DC set
AC reset
DC reset
Conditional Edge Trigger; see Figure 28
Emitter follower output at Pin P is "latched back" and keeps T4 off till T3 is biased off.
Emitter follower output at Pin N is "latched back" by D11 and keeps T3 on till + input occurs at base of T3.
Emitter Follower "T" Level Outputs
Sequence of Operation
1. Start
T4 on T2 (max. conduction)
T3 off T1 (min. conduction)
2. +T input to Pin G
T4 off, T3 turn on
T1 max. conduction
Card Type CW:
http://files.righto.com/sms/CW.html

Figure 27: Edge-Sensitive Trigger

[The figure below needs some explanation for what it shows. And the legend for the blue trace can't be read on paper.]

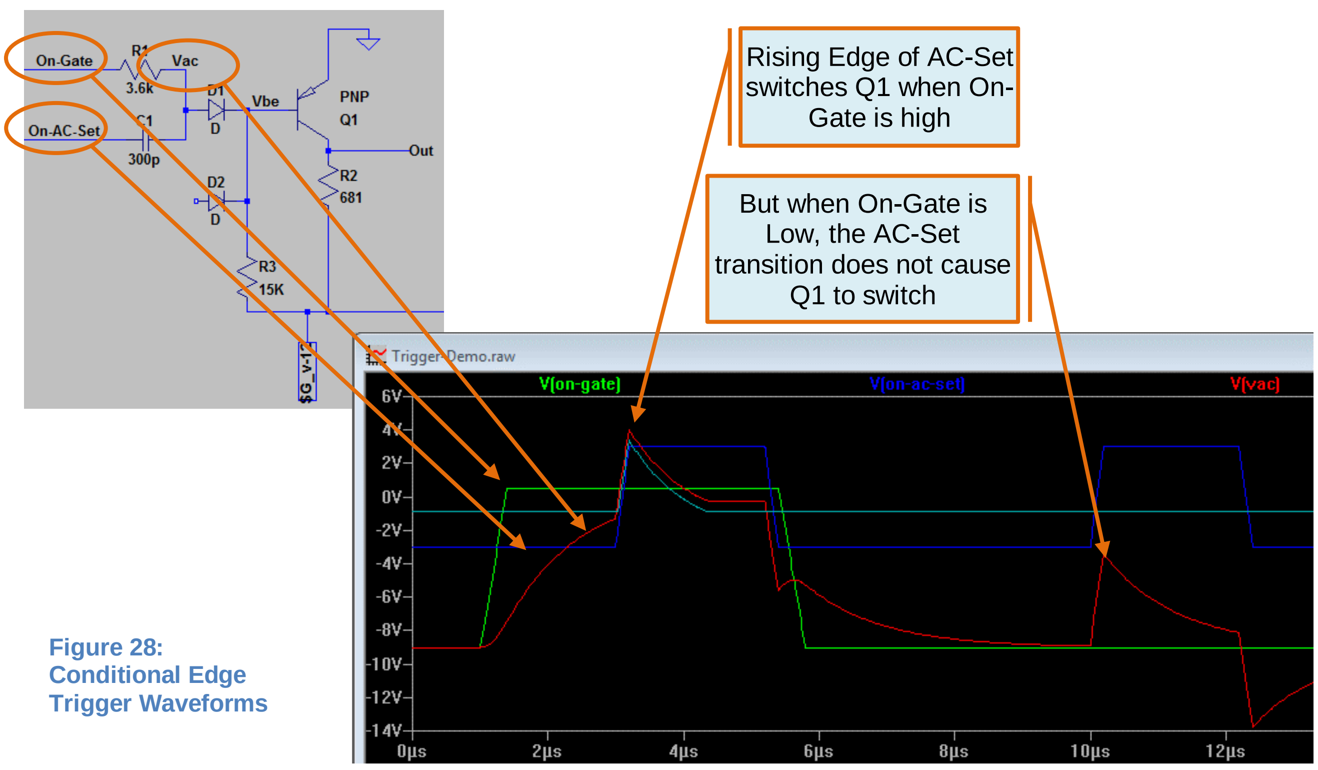


Figure 28: Conditional Edge Trigger Waveforms

And just to put this to use, Figure 29 shows the interconnect to make a Trigger into a Divide-by-Two (such building blocks would be used in the master clock generator).[33]

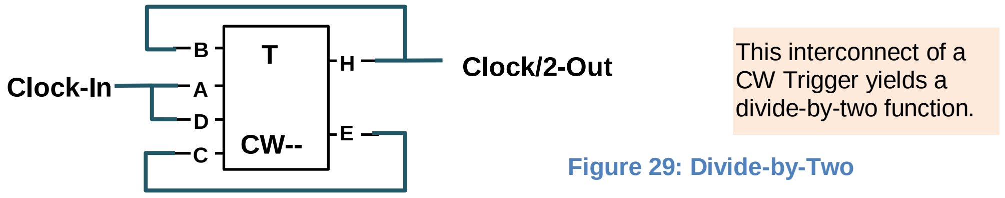

B
A
D
C
T
CW--
H
E
Clock-In
Clock/2-Out
This interconnect of a CW Trigger yields a divide-by-two function.

Figure 29: Divide-by-Two

5.3.4 Putting the Gates Together

Back uncounted pages ago, Figure 6 showed a tiny piece of the data path, encompassing one bit of the A and B registers. Here's the picture again, in Figure 30.

In the subsequent figure, we can see how that's implemented in actual gates, both in conventional logic symbols, and also as depicted in the ALD page (page 35.11.11.2 of [ALD]).

[33] May we suggest the use of one of the most excellent analog circuit simulators floating around to see how this actually works…

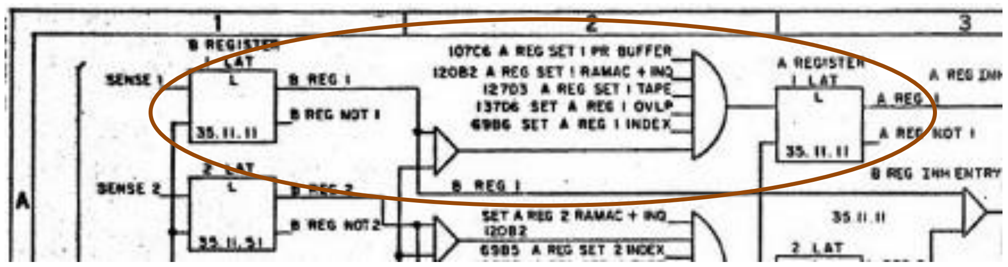
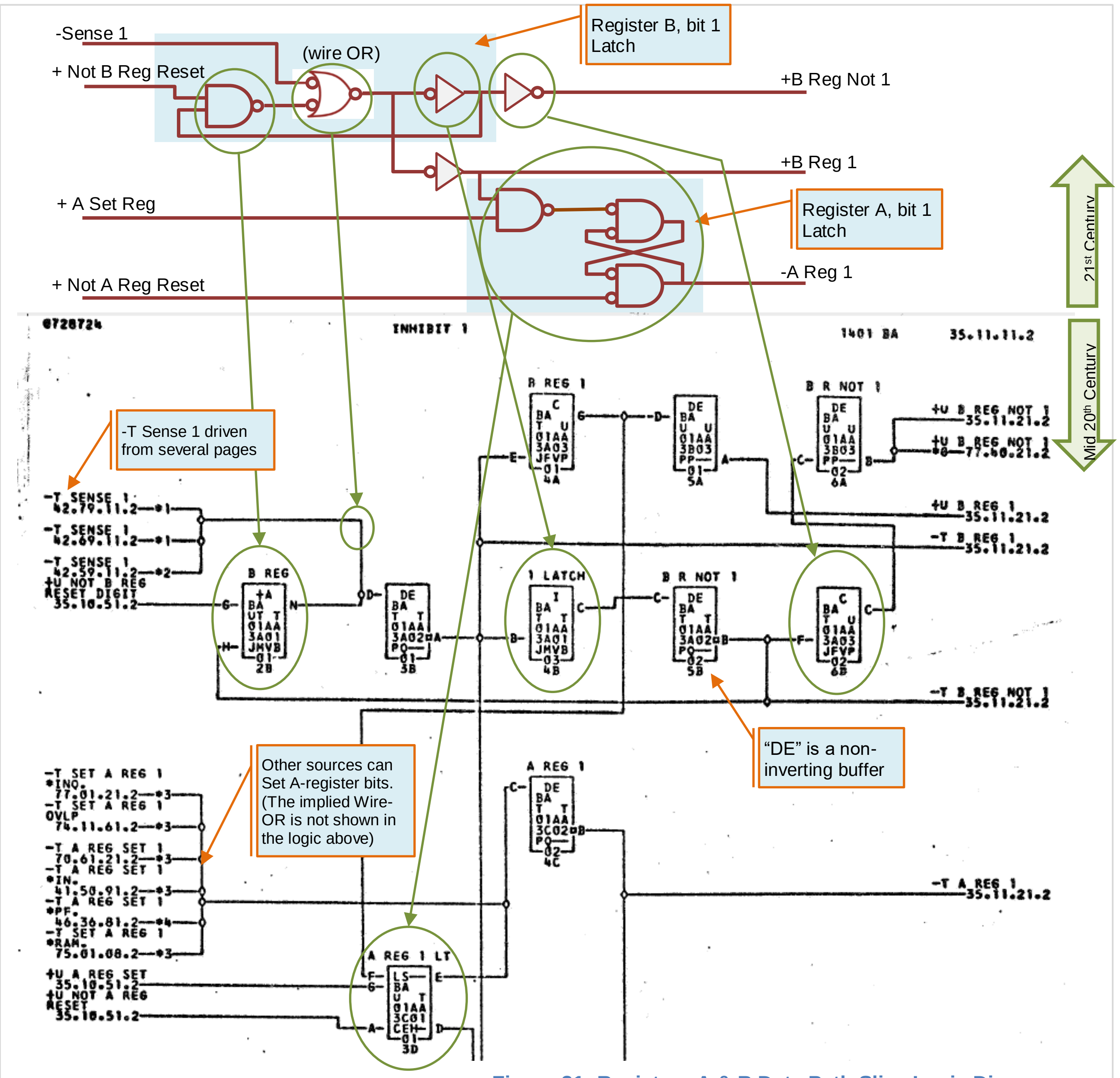


Figure 30: Registers A & B in ILD Format

Figure 31 shows the corresponding automated logic diagram. Inputs come in on the left of the page, outputs leave on the right side of the page.

It’s important to note that the registers are not clocked; each input is gated with a timing signal before reaching the schematic page, so there are few clocks on this page.


-Sense 1
+ Not B Reg Reset
(wire OR)
Register B, bit 1 Latch
+B Reg Not 1
+B Reg 1
+ A Set Reg
Register A, bit 1 Latch
+ Not A Reg Reset
-A Reg 1
21st Century
Mid 20th Century
INHIBIT 1
1401 BA
35.11.11.2
B REG 1
B R NOT 1
+U B REG NOT 1 35.11.21.2
+U B REG NOT 1 77.40.21.2
+U B REG 1 35.11.21.2
-T B REG 1 35.11.21.2
-T Sense 1 driven from several pages
-T SENSE 1 42.79.11.2
-T SENSE 1 42.69.11.2
-T SENSE 1 42.59.11.2
+U NOT B REG RESET DIGIT 35.10.51.2
B REG
1 LATCH
B R NOT 1
-T B REG NOT 1 35.11.21.2
"DE" is a non-inverting buffer
Other sources can Set A-register bits. (The implied Wire-OR is not shown in the logic above)
A REG 1
-T SET A REG 1 *INQ. 77.01.21.2
-T SET A REG 1 OVLP 74.11.61.2
-T A REG SET 1 70.61.21.2
-T A REG SET 1 *IN* 41.50.91.2
-T A REG SET 1 *PF. 46.36.81.2
-T SET A REG 1 *RAM* 75.01.08.2
+U A REG SET 35.10.51.2
+U NOT A REG RESET 35.10.51.2
-T A REG 1 35.11.21.2
A REG 1 LT
Figure 31: Registers A & B Data Path Slice Logic Diagram

5.4 Major Blocks

5.4.1 Core Storage Memory in the IBM 1401

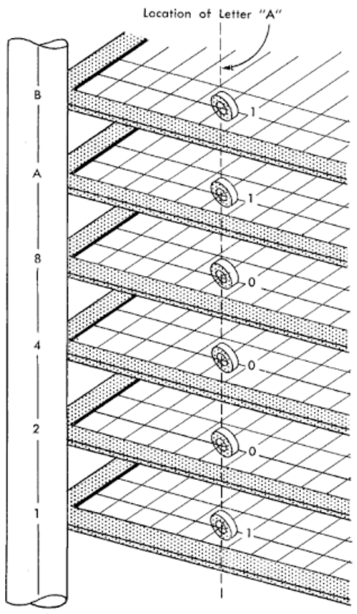
See generic overview of core-storage technology at http://ed-thelen.org/comp-hist/navy-core-memory-desc.html

Some theory-of-operation plus timing waveforms for 1401 core storage can be found in [FMM] pdf-pg 119

(Contributed by Ken Shiriff, August 2015; edited by Guy)

The IBM 1401, like most computers of its era, uses ferrite core memory, which was the leading memory technology from the mid-1950s until it was replaced by semiconductor memory in the early 1970s. For its time, core memory provided relatively dense, reliable, and inexpensive storage.

The 1401's main core memory provides 4,000 characters of storage, as well as special I/O storage. Each bit of data in memory is stored in a tiny ferrite ring or core. These cores can be magnetized in one of two directions, corresponding to a 0 or 1 bit. The cores are arranged into a grid of 4000 cores, called a plane. Each plane stores one bit of a character, so 8 planes are stacked up to store a character. (The additional I/O planes will be discussed later.)

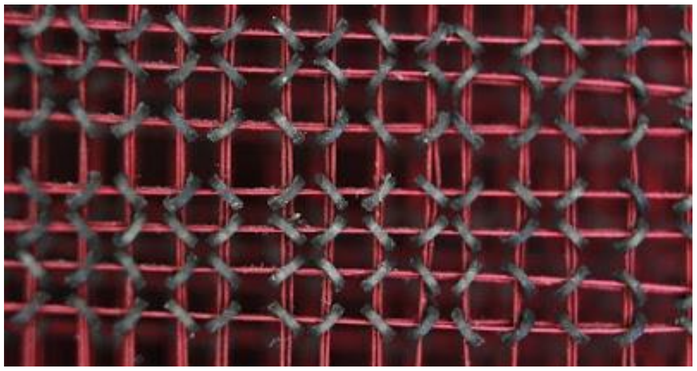


The diagram to the right, from the 1401 Reference Manual illustrates how the character 'A' is stored in core memory. To select an address, an X wire and a Y wire are activated, selecting the cores where those two wires cross. The bits from the selected location each plane make up the character. Since there are 4000 cores in each plane, the core module provides 4000 characters of storage.

5.4.1.1 Properties of Ferrite Cores

The physical properties of ferrite cores are critical to the operation of the core memory, so it is important to understand them. First, if a wire through a core carries a strong current, the core will be magnetized according to the direction of the current (following the right-hand rule). Current in one direction will write a 1 to the core, while the opposite current will cause the opposite magnetization and write a 0 to the core.

Hysteresis is a key property of the cores: current must exceed a threshold to affect a core's magnetization. A small current will have no effect on the core, but a current above a threshold will cause the core to "snap" into the magnetized state aligned with the current.

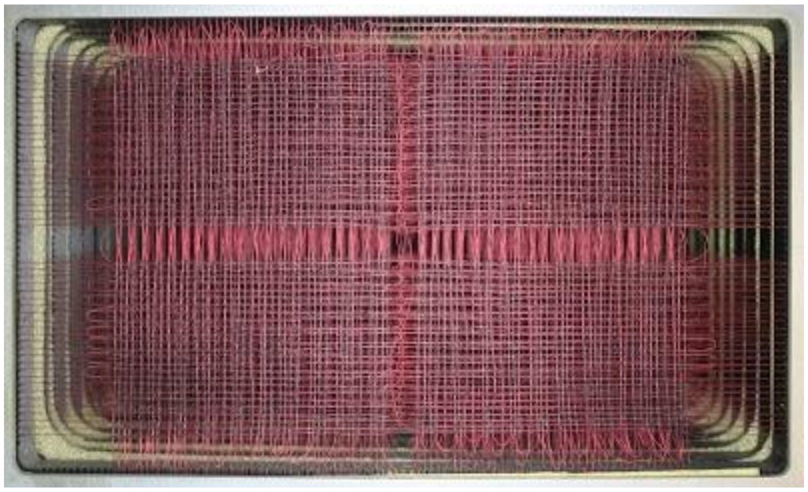


The hysteresis property makes it possible to select a particular core. A "half-write" current is sent through the appropriate X select wire and a "half-write" current through the Y select wire. The single core with the selected X and Y wires will have enough current to change state, but the other cores will not have enough current, and will remain unchanged.

The final important property is that when a core switches its direction of magnetization, it induces a current in a sense wire through the core (kind of like a transformer). If the core already has the target state and doesn't change magnetization, no current is induced. This induced current is used to read the state of a core. A consequence is that reading a core erases it, and the desired value must be written back to the core.

5.4.1.2 Structure of a Core Plane

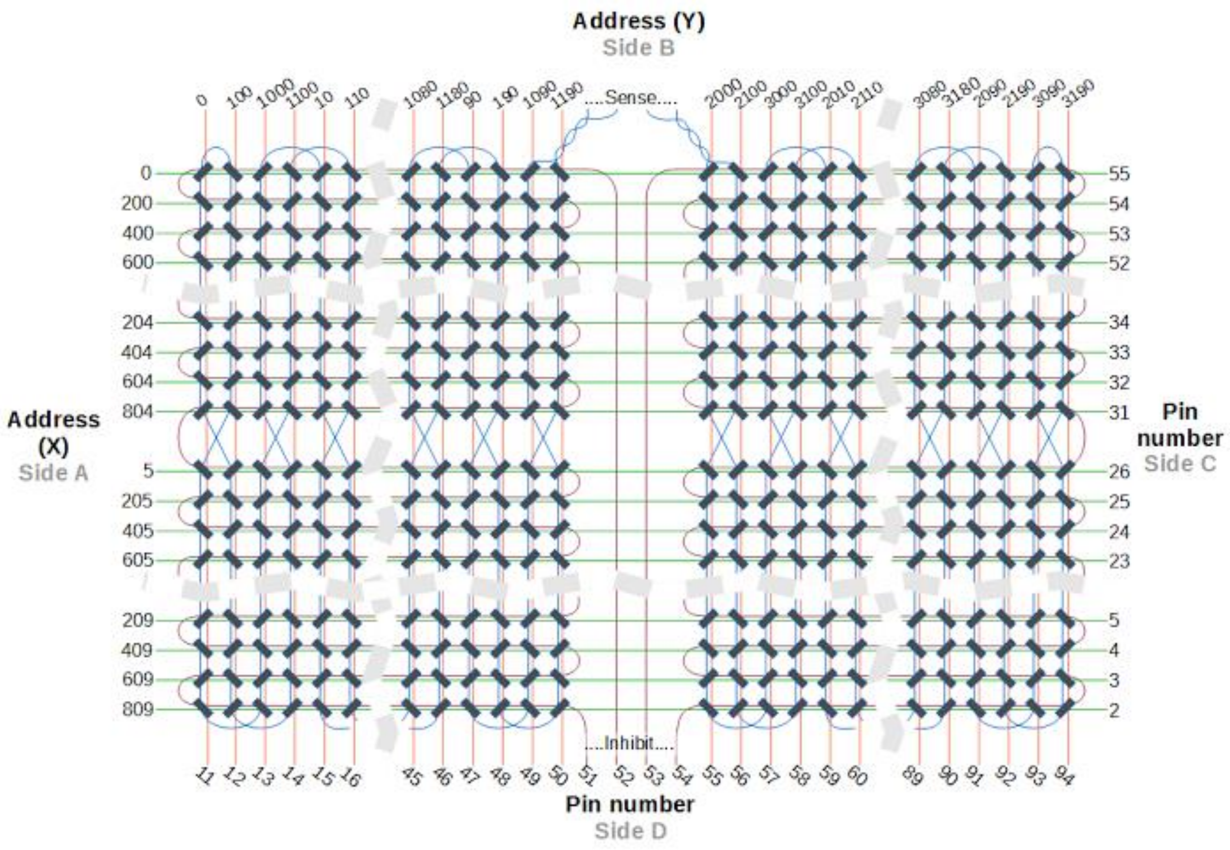
Each core plane has 4000 cores arranged as a 50x80 grid of cores. (The I/O planes are configured differently, and will be explained later.) To reduce interference, the ferrite cores are arranged in a "checkerboard" pattern with each core arranged diagonally in the opposite direction from its neighbors. Four wires pass through each core. The horizontal wires are the X select line and the inhibit line (used for writing). The vertical wires are the Y select line and the sense line (used for reading). The X and Y select lines go through all the planes, so all planes are accessed in parallel.



To read a core, the X and Y select lines magnetize the selected cores to the "0" direction. If the core was previously in the "1" state, the core's state change induces a current in the sense wire. If the core was already in the "0" state, no current is induced. Thus, the sense wire allows the bit stored in the core is determined. The read process destroys the previous value of the core, leaving it in the 0 state. Each plane has a sense wire threaded through all the cores in the plane.

To write a core, current of the opposite polarity is sent through the X and Y select lines to magnetize the core into the 1 state. To keep the core in the 0 state, a current is sent through the plane's inhibit line. The inhibit wire runs through all the cores in a plane parallel to the X select lines. By running the reverse current through the inhibit wire, the X line's current is canceled out, and the core remains unchanged. The inhibit current is too low to flip a core by itself, so other cores are not zeroed out. Each plane has an inhibit wire threaded through all the cores in the plane, so the appropriate value can be written to each plane.

The diagram below shows the reverse-engineered wiring topology of a IBM 1401 core memory plane. Most of the core has been cut out of the diagram, as indicated by the dotted gray lines. The sides of the plane are labeled A through D, matching the 1401 documentation. The A and C sides have 56 pins, while the B and D sides of the plane have 104 pins. Not all the pins are connected.



The X select lines are in green and the Y select lines are in red. The select lines are generated in a complex way (described below), so cores are not arranged sequentially. To find the address of a core on this diagram, add the labels on the X and Y select lines.

Each half of the core plane (0-1999 and 2000-3999) has a separate sense line loop, but they are usually wired together. The two sense lines are in blue and run in the Y direction. The sense lines are carefully arranged to avoid picking up interference. The lines cross over along the midpoint to cancel out noise from the Y select lines - the sense line runs in the opposite direction along half of each Y select line, so any induced signal will be canceled out. In addition, the sense lines are twisted as they exit the middle of the plane, to avoid picking up interference. (Many other core memory systems avoid interference by running the sense line diagonally, but the 1401 uses a rectangular layout.)

Each half of the plane has a separate inhibit line. The two inhibit lines are in brown and run next to the X select lines, which they inhibit. The two lines are normally driven separately to reduce noise, but have the same signal. Since the inhibit line switches direction each row, alternating X select lines are also driven in opposite directions.

5.4.1.3 Address Decoding

The IBM 1401 uses a three-character address to access 4000 locations (addresses above 4000 will be ignored for now.) The decimal parts of the three characters specify the address from 000 to 999. The two zone bits of the hundreds character provide the thousands digit 0 to 3.

Address decoding for the core module is fairly complex. The tens digit is decoded into ten separate input lines (0 through 9), as is the units digit. The hundreds digit is split into the three top bits (called "even hundreds") and the low-order bit (called "odd hundreds"). The value of the even hundreds generates five input lines: 0, 2, 4, 6, and 8. The odd hundreds generates two input lines: 0 or 1.

The X and Y select lines is output by *matrix switches*.[34] (See Section 5.4.1.10) Each matrix switch takes two sets of input lines and activates an output line based on the input values. The X matrix switch has the 5 even hundreds lines as one input and 10 units digit lines as the second input. There are 50 combinations of inputs, so the X matrix switch has 50 outputs, which are used as the X select lines. For the Y matrix switch, the thousands (4 values) and odd hundreds (2 values) are combined to yield 8 input lines. The second Y input is 10 tens digit lines. The output from the Y matrix switch activates one of the 80 Y select lines. This addressing mechanism may seem overly complicated, but it minimizes the hardware required for address decoding.

5.4.1.4 Physical layout of the core module

The core module consists of 16 frames - 14 core planes and two terminal planes. The first 8 frames hold the character data planes. The next 8 frames are used for I/O. The frames are stacked to form the core module. The following table shows the usage of each frame:

Frame	Core Plane Function
1:	Bit 8
2:	Bit 4
3:	Bit 2
4:	Bit 1
5:	Bit A
6:	Bit B
7:	Bit C (parity)
8:	Write Mark
9:	Terminals for frame 10
10:	Read station 2 brushes (RD2), punch brushes (PCH)
11:	Terminals for frame 12
12:	Read station 1 brushes (RD1), print hammers (PRT)
13:	XU 11
14:	YU 12
15:	XL 13
16:	YL 14

The picture below shows one side of the core module, side D, showing the large amount of wiring required. Frame 16 (YL) is at the left and frame 1 (bit 8) is at the right. The two matrix switches are on the front of the module: the 8x10 switch for the Y select lines is at top, and the 5x10 switch for the X select lines is at the bottom.

[34] It would be good to reorganize the material on Matrix Switches; maybe it's just me, but I think the operation of the matrix switch transformers is not at all intuitive…

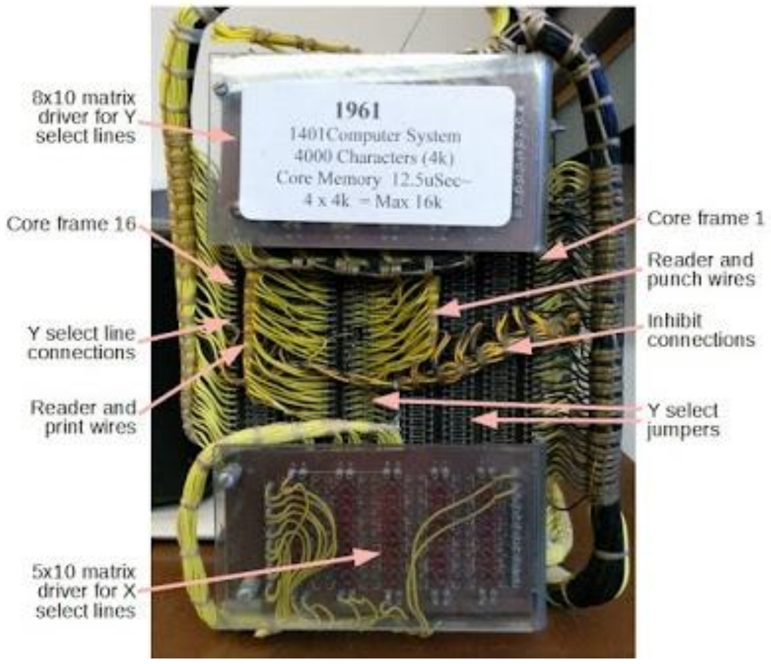


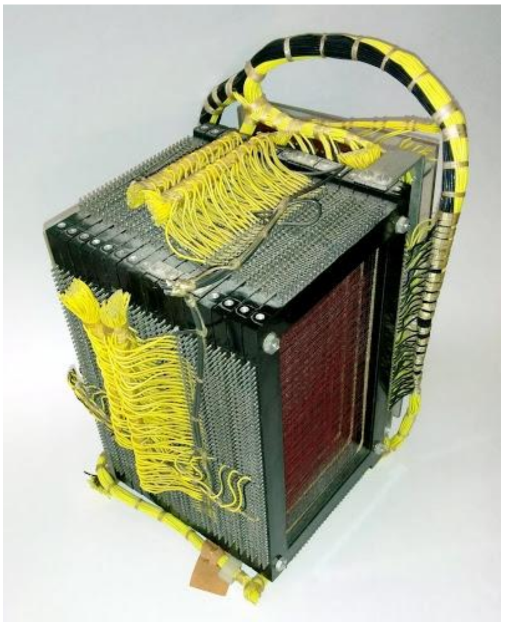
Figure 32: Core Memory Switching Matrix and Core Planes

Much of the wiring in this picture is for the Y select lines, which flow through all the planes. The yellow wires at the left connect the Y select lines on frame 16 to the 8x10 matrix switch or termination resistors. The horizontal Y select jumpers are barely visible; they connect pairs of planes, allowing the Y select lines to flow through the whole module. At the far right, the Y select lines of frame 1 are connected to the 8x10 matrix switch and termination resistors.

Two bundles of wires connect to I/O planes near the middle of the module. One connects the brushes in the card reader and the printer hammer drivers to terminals on frame 11, which are connected to the cores in frame 12. The other bundle connects read brushes and punch check brushes to terminals on frame 9, connected to the cores in frame 10. In between, short horizontal yellow wires in the middle jumper the Y select lines across frame 11. The horizontal wire bundle across the middle of the planes connects to the inhibit lines of each plane.

The photo below provides another view, focusing on the data plane wiring. At front is frame 1, the core plane for data bit 8. The gray cores are visible on red wires. The other 15 frames are layered behind frame 1. At top is side D of the module, with the two matrix switches on top. The 8x10 matrix switch is connected to the Y select lines at the front and back of side D. These wires alternate with black wires that connect the select lines to termination resistors. The 5x10 matrix switch is connected to the X select lines at the front and back of side C, on the left. This side also has jumpers between planes to connect the X select lines.

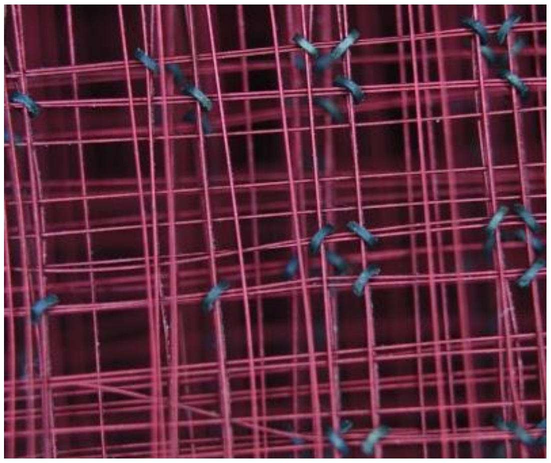
The photo below shows the other sides of the core module, focusing on the I/O wiring. On the right is frame 16, one of the hole count frames. This frame only has 297 cores. At the top is side A, with wires from the read brushes, punch check brushes, and print hammers wired directly to cores in frames 10 and 12. At the left is side B, which also has wires from the read brushes, punch check brushes, and print hammers. The two wires connected to the middle of side B of each frame are the sense wire connections.



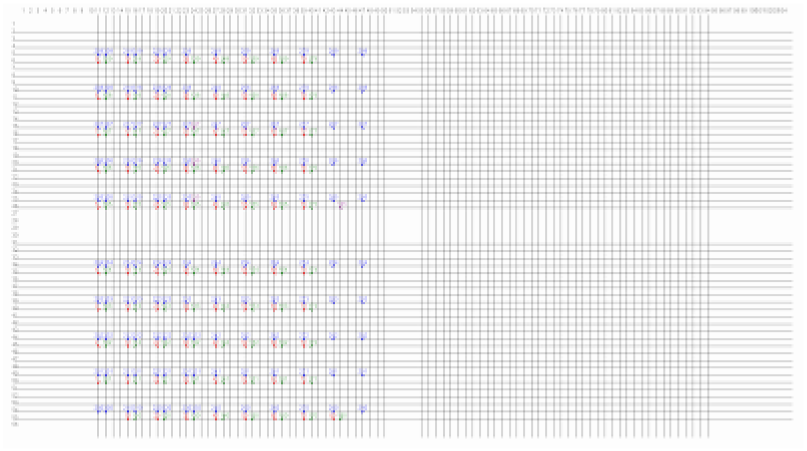
5.4.1.5 The I/O frames

One unusual feature of the core module is the eight special-purpose I/O frames: six core planes and two terminal frames. These planes have multiple uses. They hold 80 positions of data when a row is read during card reading or punch checking. They are used for validity checking during card reading, card punching, and printing.

The I/O planes are addressed exactly the same as the data planes, and share the X/Y select wires. However, the I/O planes are very sparse, with only 297 cores rather than 4000 cores, so most locations have no storage, as can be seen in the photo below.



The planes have cores at addresses 1 to 80 for reading, 101 to 180 for punching, and 200 to 332 for printing. Inexplicably, there are also cores at addresses 185, 335, 336, and 337. The diagram below shows the layout of I/O cores. Reader cores are red, punch cores are green, print cores are blue, and other cores are purple.



The I/O planes are not read and written like data planes. While they are addressed in parallel with the data planes, data read from these planes goes to the appropriate I/O circuitry, rather than the regular data path. Four of the planes have values written by the I/O circuitry. Two planes have individual cores wired to read brushes[35] or print hammers, so the cores are modified directly; they are not written using the normal core addressing / inhibit circuitry. The other four planes are written by I/O circuitry in the 1401.

The first direct-wired plane is frame 10, which holds data from read station 2 brushes (RD2) and the punch check brushes (PCH). The second direct-wired plane is frame 12, which holds data from read station 1 brushes (RD1) and the print hammers (PRT). As a card passes through the card reader, 80 brushes check each position in a row. If there is a hole, the brush makes connection passing a current through a wire connected directly from the card reader to the core module. Each wire from a brush is wrapped around the corresponding core, so the core is set to 1 if a hole is present. Since there are three read stations with 80 brushes each, there are 240 wires between the reader/punch and the core module. Each of the 240 brushes is wired directly to a separate core. There is no buffering or driver between the

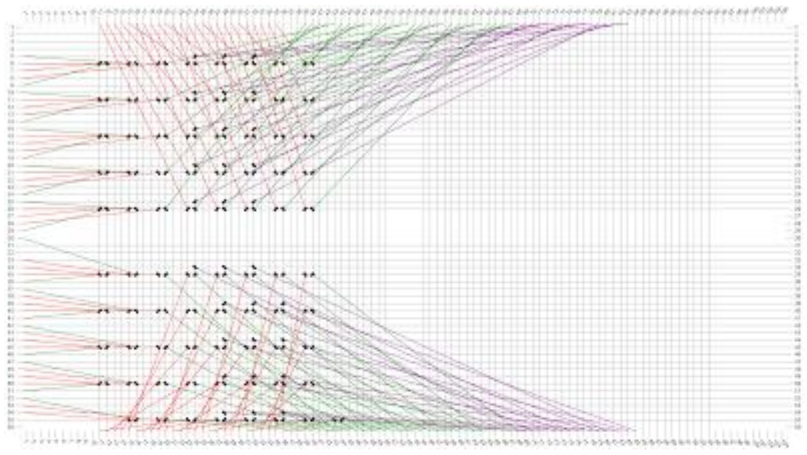
[35] See Section 6.1.2.4 to learn about what drives these signals

brush and the core, just a direct wire that is wrapped five times around the core and connected to a -20V common wire. (Field Engineering page 95 and (ALD 42.40.51.1.)

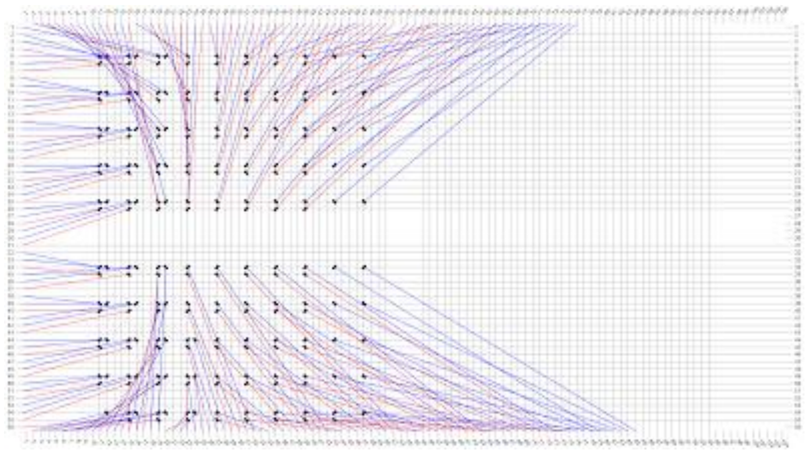
The print hammer cores are also wired individually. Each of the 132 hammer driver is connected directly to a core to record the operation of the hammer. After passing through a core, each of these 132 wires is connected to a -60V common wire. (Field engineering manual, page 107,ALD 42.40.52.1 and 36.38.61.2-36.39.91.2.)

Because of the large number of I/O wires that must be connected and routed, separate terminal frames are used above the core frames. Frame 9 provides the terminals for the cores in frame 10, and frame 11 provides the terminals in frame 12. That is, frames 10 and 12 have the cores along with select, sense, and inhibit lines, while the brush and hammer wires are connected to frames 9 and 11, which then have the wires wrapped around cores in frames 10 and 12. The -20V and -60V common wires are internal to the frames.

The following diagram shows how the terminals in frame 9 are wired to cores in frame 10. Read indicates read 2 brushes, green indicates punch check brushes, and purple indicates 1404 read brushes (unused).



The following diagram shows the wiring between frame 11 and the cores in frame 12. Red indicates read 1 brushes and blue indicates hammers.



The four remaining I/O planes are XU 11, YU 12, XL 13, and YL 14 (frames 13 to 16). These planes are used for read/punch hole counting and for print validity checks.

During the card read process, a pair of planes (XU/XL) or (YU/YL) counts the holes in each column at the first read station. This count is verified at the second read station; if the counts don't match, the read fails. For punches, the counts are verified against the expected number of holes. (It's not exactly a count; XU or YU tracks the presence of any hole, and XL or YL maintains the parity of the count.) Because one card is being read/punched while a second is being verified, two counts must be maintained. Alternate cards switch between XU/XL and YU/YL. (ILD page 62).

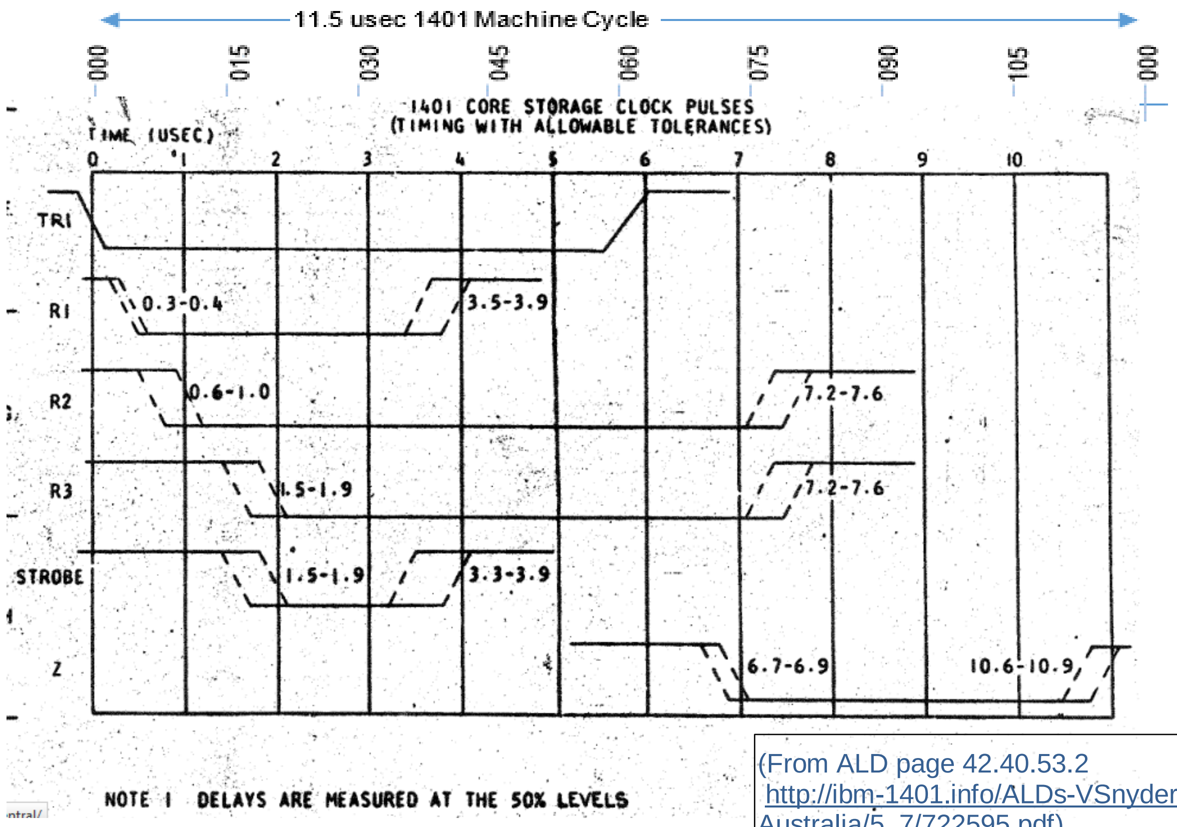
Printing uses print-compare cores in the XU plane. These are set if a hammer should fire for the character position. These are compared with the hammer-fire cores (set by the hammer drivers) to verify that the correct hammers fired. Plane YL holds print-line complete cores that verify that every character position either printed a character or holds a non-printable character. If a core is not set, something went wrong with the printing logic. Finally, area YU has print-error storage cores. If a print error is detected, the corresponding core is set, allowing the location of the fault to be dumped out. (ILD page 72 and Data Flow pages 62-63.) When the optional Print Storage feature is installed, the Print Storage core module is used instead of these planes.

5.4.1.6 Core Memory Timing

The IBM 1401 has an 11.5 microsecond machine cycle, which corresponds to one core read/write cycle. When describing the circuitry, however, the machine cycle is rounded to 12 microseconds for convenience. Timing is described in units of 0.1 microseconds, so each cycle has a time value from 000 to 120. Each cycle is broken into eight clock phases of 015 units (1.5 microseconds). For example, a time interval can be denoted as 045-060. Note that these times are about 4% off from the real time values. Some timing signals, including core timing controls, are not aligned on intervals of 015; they are generated from a CEA delay card. (ILD page 1.) [merge and xref clock section]

The read operation starts with the input pulses to the matrix switches. The X matrix switch inputs are gated by timing signals R1 (000-036) and R2 (008-075), and the matrix generates a read pulse for 007-035. The Y matrix switch inputs are controlled by timing signals R1 (000-036) and R3 (015-075). The Y matrix generates a read pulse for time 015-035. These read pulses select the desired cores in storage, possibly triggering signals on the sense lines. The sense amplifiers are gated by Strobe A 014-030 to read the value from the cores. The value is stored in the B register, which was cleared from 000-015.

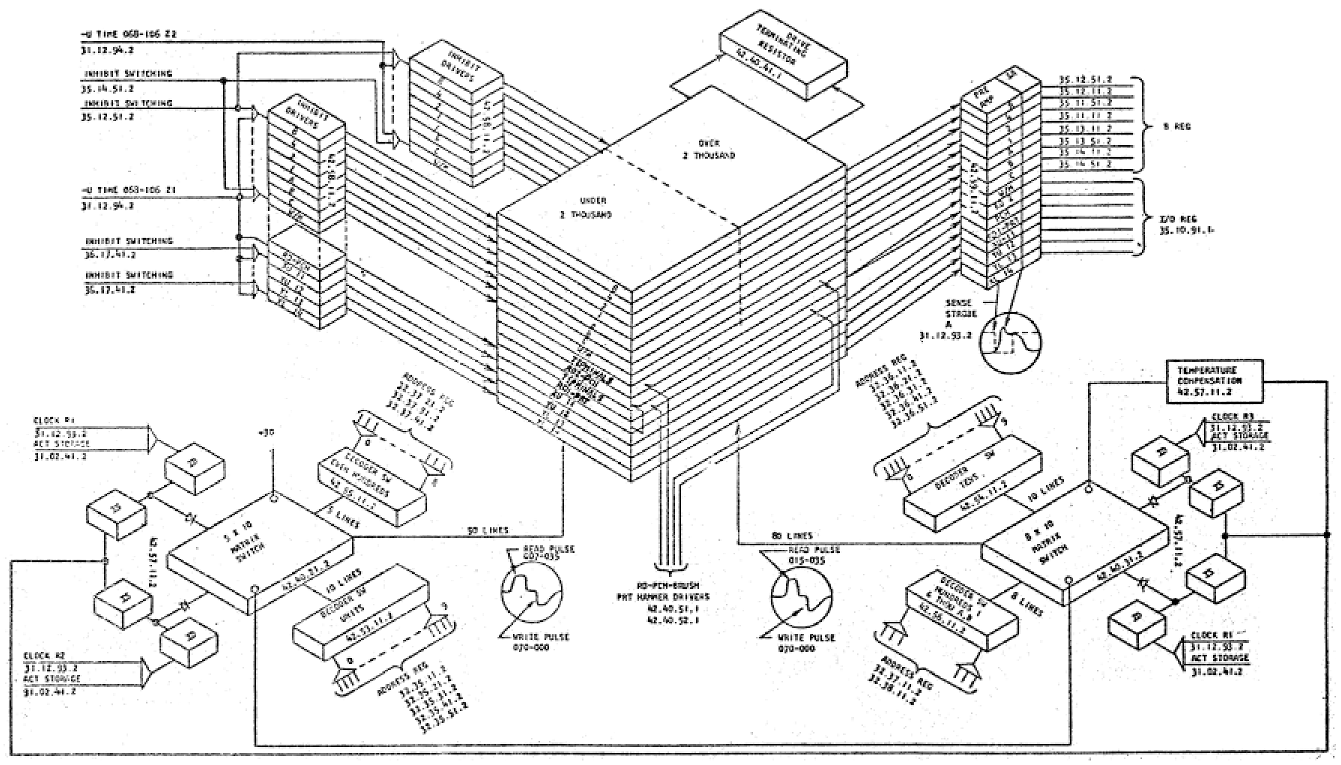
The read operation is followed by the write operation. The write timing signals Z1 and Z2 (068-105) activate the inhibit drivers. The write is triggered by the negative-phase select pulses from the matrix switches, generated when the switch core resets from 070-000. Thus, the write finishes just at the end of the machine cycle, in time for the next cycle.



(From ALD page 42.40.53.2 http://ibm-1401.info/ALDs-VSnyder-Australia/5_7/722595.pdf)

5.4.1.7 The Core Module Circuitry

The detailed diagram below is the key to understanding the 1401's core memory system and is worth close study. This diagram (ALD 42.41.11.2) shows the physical arrangement of the 16 frames in the core memory module, along with the driver circuitry. The inhibit drivers are at the upper left, feeding each core plane. The sense amplifiers are at the upper right. The 5x10 X matrix switch is in the lower left, and the 8x10 Y matrix switch is in the lower right. Note the read brushes, punch brushes, and print hammer drivers are wired directly into the core module through the terminal frames. The "IS" and "ID" boxes feeding the matrix switch are the current source (buffer cards) and current drain (switch cards) driving the matrix switch, respectively. The diagram also shows the timing of the read and write pulses, and how they have opposite polarity, writing 0 and 1 respectively.



The diagram below annotates the Switch Core Circuitry diagram. It shows in detail the circuitry used to drive one of the 4000 cores in a plane. The notation in this diagram may be confusing. The rectangles indicate SMS cards. The letter inside the rectangle indicates the function: A is an AND gate, DP is a current driver, an D is a switched driver. The text above an SMS rectangle ("F 14") is the position of the card in gate 01A1. The letters outside the box ("A", "B", "C") are pins on the card. The numbers ("42.57.11") indicate the relevant ALD. R1 and R2 are read time signals. Numbers ("014-030") indicate a timing signal.

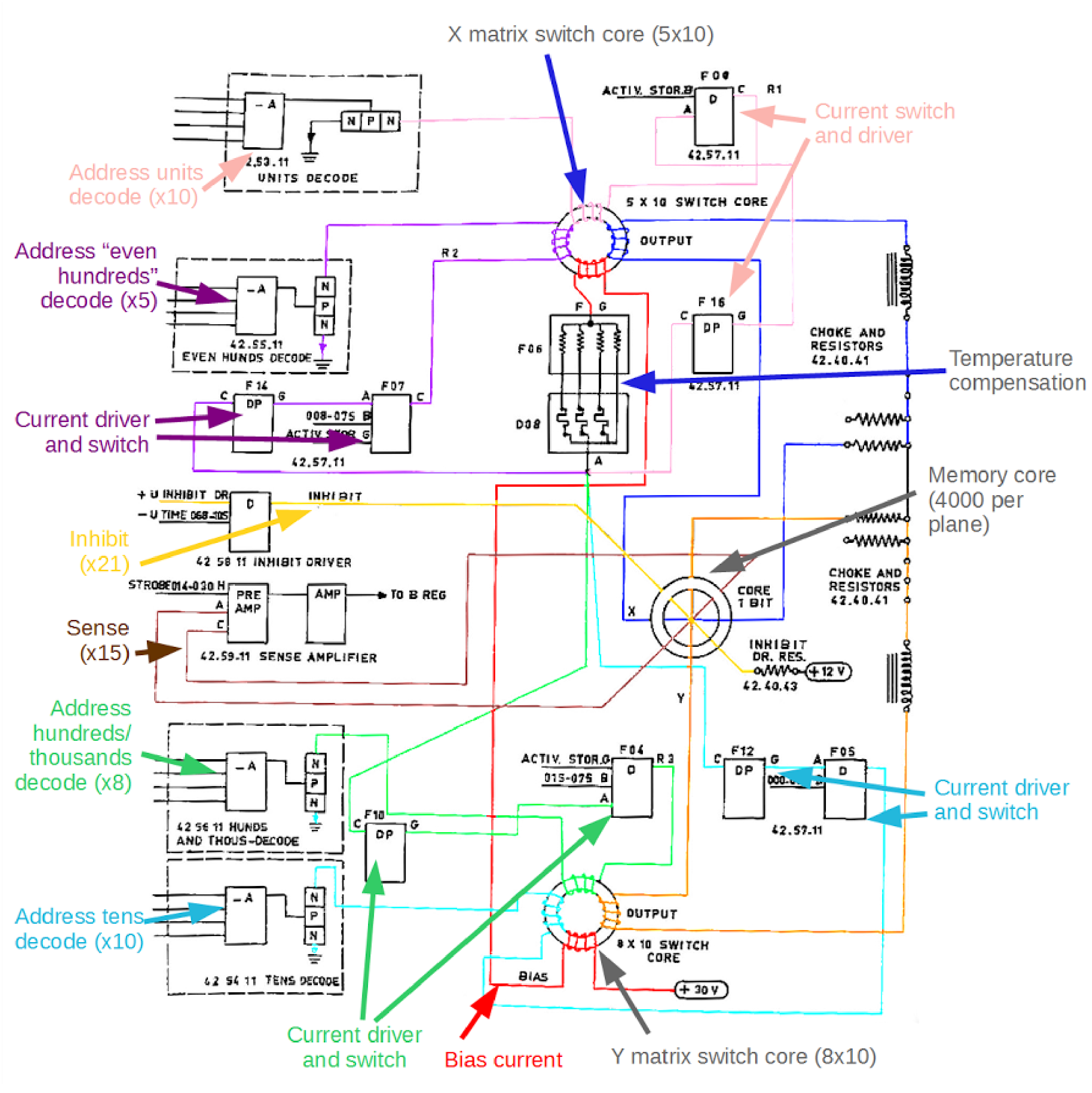


Figure 33: Matrix Switch Cores[36]

The yellow line shows one of the inhibit drive circuits. The inhibit drive signals come from the processor and are gated for the appropriate timing.

The brown line shows one of the sense line circuits, driving a sense pre-amp and amplifier, and then feeding the B register.

[36] We don't know the source of this diagram; if anyone can identify the IBM doc from which the original black-and-white copy without annotation would have come, please speak up. See http://ibm-1401.info/1401MemoryOverview-b-.jpg for the original page.

The red line shows the bias current which passes through all the matrix switch cores. The properties of ferrite cores change with temperature, so the current is adjusted by a temperature compensation circuit (an AKB thermal switch card and an AKC memory emitter resistor card). This current then splits four ways, going to the X matrix decode drive lines (pink and purple) and the Y matrix decode drive lines (green and cyan). Note that only one switch core from each switch matrix is shown.

The outputs from the matrix switches (blue and orange) are the select lines that go through the cores and are terminated by chokes (inductors) and resistors. Only one of the 50 X select lines and one of the 80 Y select lines are shown.

5.4.1.8 The Inhibit Drivers

As explained earlier, during the write phase, a 1 will be written to the selected core unless the inhibit line is active. In that case, the inhibit line counteracts the X line; without enough current to flip the core, the core remains at 0. However, the implementation in the 1401 is more complicated than this and there are 21 inhibit drivers in total (16 for data planes, 4 for hole check planes, and 1 for the row bit core planes).

To write a character back to storage, the 1401 generates 8 inhibit drive lines: bits 1, 2, 4, 8, A, B, check, and write mark. These lines are generated by selecting from the A register, B register, manual console entry, arithmetic value, or expanded edit (ILD page 10). However, these inhibit lines do not go directly to the core planes. Instead the data planes are split in half and each half has separate inhibit drivers and inhibit lines. The planes are split into addresses 0-1999 and 2000-3999, but both halves receive the same inhibit value. The motivation of the split is to reduce sense noise. (See Optional Feature pages 6 and 7.)

The I/O planes have their own inhibit drives, and each plane has a single inhibit line. The XU, YU, XL, and YL check planes have inhibit drives that are generated by the hole counting circuitry in the reader / punch circuit (ILD page 62).

The RD1-PRT plane and RD2-PCH planes are not written under computer control, but by the individual wires from the brushes or hammers to the cores. Thus, the inhibit signal for these planes is always activated for writes, and a read/write cycle clears the cores to 0. A single inhibit driver provides the inhibit signal for both planes; the inhibit lines for the planes are connected by a jumper between frame 10 and frame 12 (ALD 42.40.43.2). The PRT cores do not have any inhibit, so they are set to 1 at the end of a read/write cycle. When a hammer fires, the associated core is set to 0 by the direct wire.

The inhibit lines are driven by AQV driver cards, which provide the necessary current (ALD 42.50.11.2).

There are 13 termination resistors for the inhibit lines on a termination board. The split inhibit lines for the data lines share termination resistors even though they have separate drivers (ALD 42.40.43.2).

5.4.1.9 The Core Sense Circuit

When a core switches state during a read, the sense circuit must pick up this signal and output the bit value. A key challenge for the sense circuit is detecting this relatively small signal in the presence of large switching signals. The sense line switches direction at the midpoint of each column to avoid picking up interference. A consequence of this is the input signal may be a pulse of either polarity.

The sense line forms a loop through the cores in a plane, with both ends of the loop terminating at WX differential pre-amplifier card. The pre-amplifier is somewhat different from a modern differential amplifier in that it is symmetrical and gives a positive output for handles a pulse of either polarity. The pre-amp is strobed at time 014-030, to read the sense line during the read pulse. The output of the pre-amplifier goes to the AQX final amplifier and then to the logic circuit that uses this bit.

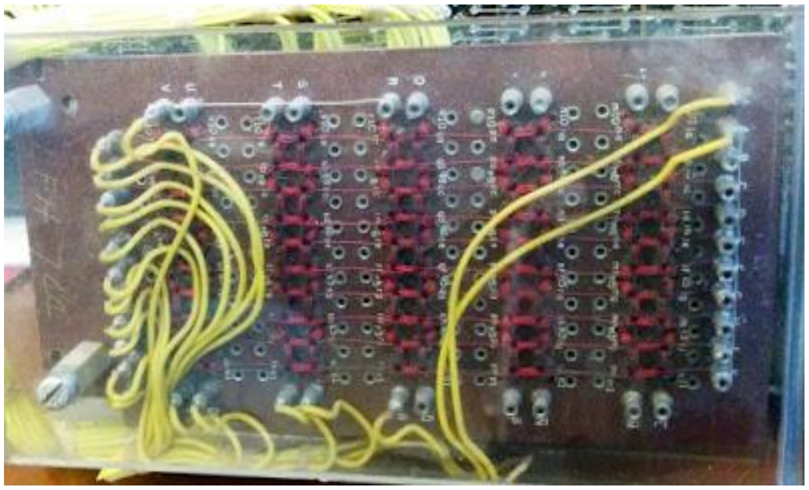
For the data planes, the sense lines in the two half-planes are wired together to make 4K planes. Each I/O plane has a single sense line through the populated cores. An exception is frame 10, which has separate sense lines for the RD2 cores and the PCH cores.

The sense circuitry is described in ALD 42.59.11.2 and ILD page 4.

5.4.1.10 The Matrix Switches

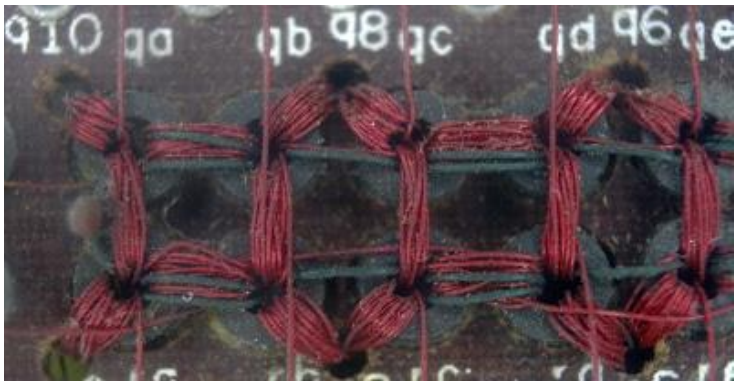
Generating the X and Y select signals is a tricky problem. The drive signals must have a positive pulse of the right current and duration for reading, followed by a negative pulse for writing. In addition, the number of select lines is large (50 X and 80 Y), so hardware costs would be excessive if each line had its own driver circuitry.

The 1401 uses an interesting solution to drive the select signals. Matrix switches generate the select signals by using a set of ferrite cores. But instead of storage, these cores are used for their switching properties. As with storage, the matrix switch depends on the "coincident current" property, where two signals of sufficient current will cause a core to snap to the opposite magnetization. But instead of being used for storage, the cores in the matrix switch generate a drive signal.



The photo above shows the X matrix switch, with 5 row inputs, 8 column inputs, and 40 outputs (connected on the back). The switch consists of 50 cores in a 5 by 10 grid, with 5 lines driving the rows and 10 lines driving the columns. Each core also has an output winding and a bias winding. When two input lines are triggered, the corresponding core flips state, generating a pulse on the output winding. When the input lines are released, the bias winding flips the core back to its original state, generating a negative pulse on the output winding. Thus, the desired one of the 50 outputs has a positive pulse followed by a negative pulse, which is just what the core module requires for read followed by write. (ILD pages 2 and 3.)

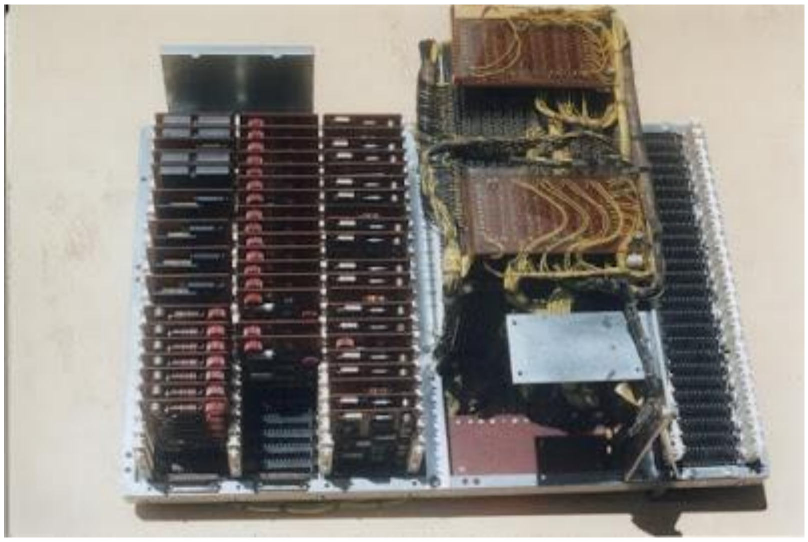
The photo below shows the wiring of the matrix switch cores. The bias wire (black) is wound through pairs of cores three times. Each horizontal input wire (red) is wound through pairs of cores about twelve times, as are the vertical input wires. Since the bias current is four times the input current, the winding numbers ensure that the bias cancels one input: two inputs will set the core, one input will have no effect, and no inputs will reset the set core (due to the bias current). Each core has an output wire wound diagonally about twelve times.



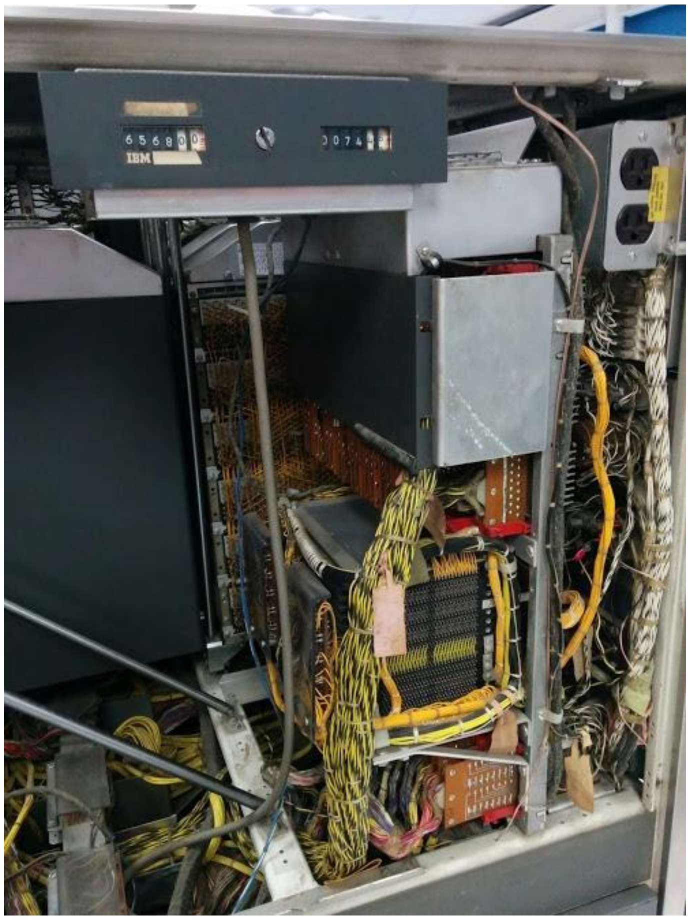
The address decoding circuitry that drives the matrix switch uses an interesting shortcut. You'd expect that the circuit decoding a digit value would use all 4 bits of the digit, but the circuit manages to use a 3-input AND gate for each value. For instance, 2 is detected as not 8 AND not 4 and not 1, ignoring the value of the 2 bit entirely. The trick is that if 8, 4, and 1 bits are all clear, the value must be 2 (keeping in mind that 0 is represented as 10). As another example, 3 is detected as 2 AND 1 AND parity, which is only matched by the digit 3 (ILD page 3).

5.4.1.11 Physical Layout of Core Memory in the 1401

The following picture (courtesy of intaretro) shows the core memory module mounted in its gate 01A1, along with the many SMS cards required for drivers and amplifiers. The gate has Column F at the left, holding the DKA driver cards and AQW current source cards that drive the matrix switch boards, and the AQV inhibit driver cards. Column E is next, holding the AQU address decode cards. The address lines plug into the empty sockets at the bottom. Column D holds WX and AQX sense amplifiers, and a couple other cards. The core module is mounted with the matrix switch cards on top. At the far right, the empty Column G has connectors for the paddles with the hundreds of wires from the brushes and print hammers.

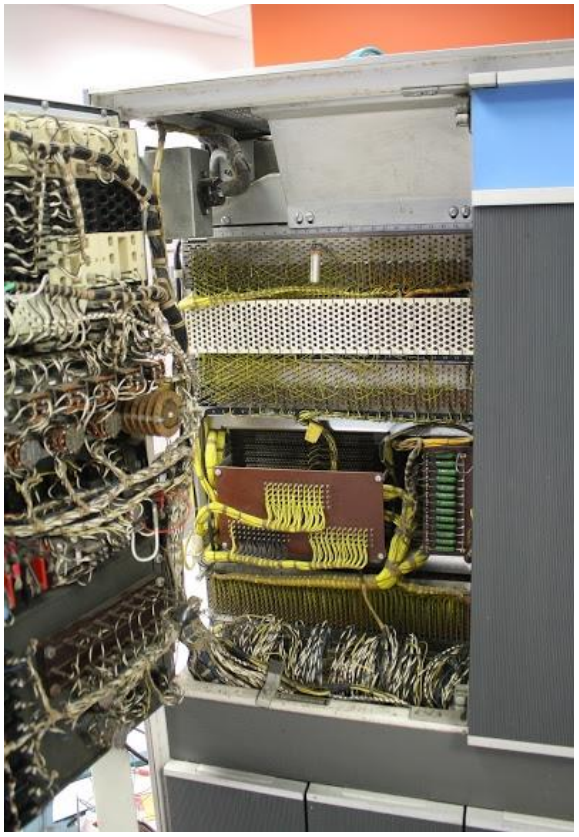


The photo below shows the core module mounted inside the IBM 1401 mainframe, looking into the left end of the computer. The core module is behind the bundle of black and yellow wires, mostly address lines. The matrix switches are on the left. The colorful brush and hammer wires are connected via paddles underneath the core module. The SMS driver cards are above the core module, mostly behind a metal cover for airflow.



The photo shows some other interesting features of the 1401. At the top of the computer is the time meter used for billing, measuring the time the computer has been running; customer time is on the left and unbilled maintenance time is on the right. In the upper right is the "convenience" outlet located inside the computer. At the far right is the wiring on the back of the front console. Unlike most of the gates in the IBM 1401, the core gate does not swing out, but is screwed into place. Other gates are visible behind the core module, and to the left.

The back of the core module can be accessed by opening the computer's front panel, as seen below. The console switches, lights, and wiring are on the left. The core module itself is in the center, mostly hidden behind the brown termination resistor circuit board.



5.4.1.12 Core System References

The IBM 1401 core module is documented in detail in ALD 42. For more information on core memory, see Coincident Current Ferrite Core Memories and Magnetic Core Memory Systems.

5.4.2 Expansion Memory Systems

Expansion Memory

5.4.3 Qui-Binary Arithmetic Unit

(Contributed by Ken Shirriff, Aug, 2015)

The IBM 1401 stores digits in binary-coded decimal (BCD), but for arithmetic it uses a different, unusual representation internally: qui-binary code. By using qui-binary code, the 1401 can perform arithmetic quickly while ensuring the calculations are correct.

In qui-binary code, each decimal number is split into a qui part (0, 2, 4, 6, or 8) and a binary part (0 or 1). For example, 3 is split into 2+1, and 8 is split into 8+0. Each digit has a unique qui-binary representation.

In the 1401, the qui part is labeled Q0, Q2, Q4, Q6, Q8 and the binary (bi) part is B0 or B1. Thus, a digit can be represented internally with 7 signals: Q0, Q2, Q4, Q6, Q8, B0, and B1, where exactly one qui signal and one binary signal are active.

The following table summarizes the qui-binary representation.

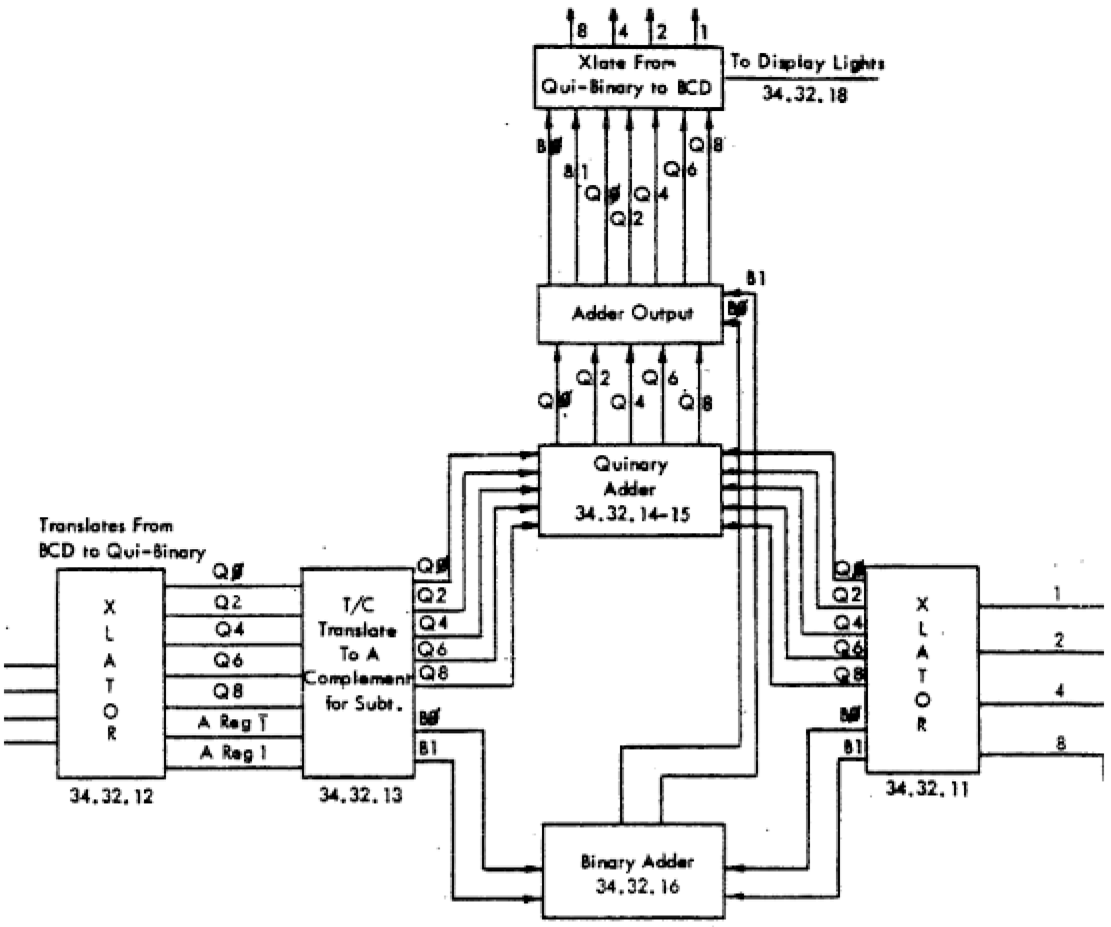
Digit	Qui	Bi
0	Q0	B0
1	Q0	B1
2	Q2	B0
3	Q2	B1
4	Q4	B0
5	Q4	B1
6	Q6	B0
7	Q6	B1
8	Q8	B0
9	Q8	B1

The main advantage of qui-binary is error checking. A qui-binary number that has 0 or 2 qui signals, or 0 or 2 binary signals indicates an error. Checking for arithmetic errors is an important feature since the IBM 1401 is a business computer. Messing up a day's payroll checks would be catastrophic, so it is important that errors are detected immediately and stop processing. Qui-binary arithmetic may also be faster than binary arithmetic, since it avoids delays due to the carry propagating through four bits.

To add two digits, the 1401 first translates the digits to qui-binary. A special qui-binary adder adds these two qui-binary digits (along with any carry). Finally, the result is translated back to BCD. While this seems like a roundabout process, it provides error checking that would be hard if the digits were added directly. While the 1401 has a parity bit for each character, it's hard to check parity while adding BCD numbers directly.

The circuitry used for addition/subtraction uses many SMS cards, and consumes most of the logic in card gate 01B3 (See Section 5.1).

The following block diagram from the 1401 ILD shows the data flow through the adder. The digit from the A register enters on the left, and is translated to qui-binary by the translation circuit (labeled XLATOR). This qui-binary value goes through a translate/complement circuit, which generates the 9's complement value if needed for subtraction. The value in the B register enters on the right and is converted to qui-binary, but without optional complementation. (The IBM 1401 performs operations on memory locations and the A and B registers provide temporary storage for a character read from core. They are not general-purpose registers as in most microprocessors.)



8
4
2
1
Xlate From Qui-Binary to BCD
To Display Lights 34.32.18
B0
B1
Q0
Q2
Q4
Q6
Q8
Adder Output
Quinary Adder 34.32.14-15
Translates From BCD to Qui-Binary
XLATOR
34.32.12
A Reg T
A Reg 1
T/C Translate To A Complement for Subt.
34.32.13
34.32.11
Binary Adder 34.32.16

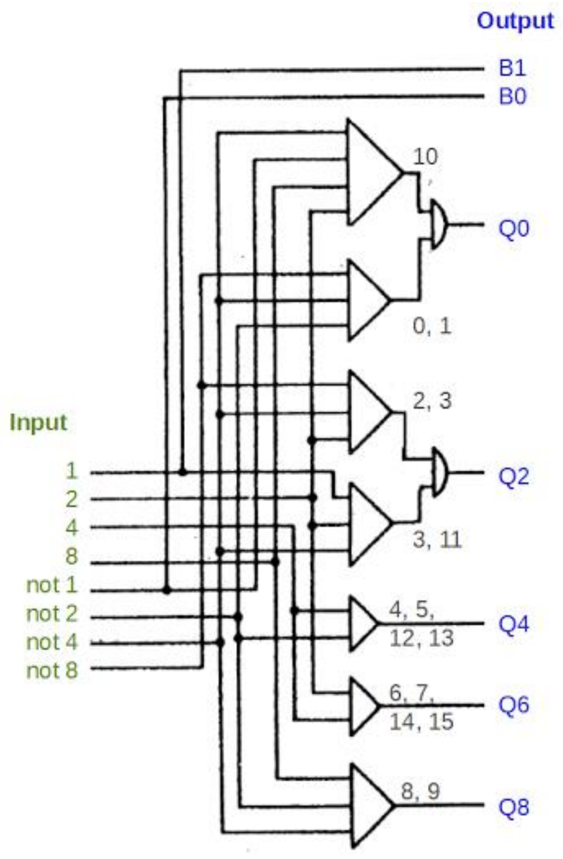
Figure 34: Overview of the arithmetic unit in the IBM 1401 mainframe.

The B0/B1 binary bits and an input carry are added by the binary adder at the bottom. The quinary bits are added with a special quinary adder. The adder output circuit applies any carry from the binary add to the quinary bits, generating the qui-binary result. Finally, a translation circuit converts the qui-binary result to BCD, sending the BCD value to the front panel display and memory. The error checking circuit verifies that the qui-binary has exactly one qui and one binary bit set.

BCD to qui-binary translation

To examine the addition/subtraction circuitry in more detail, we'll start with the logic to convert a BCD digit to qui-binary, as seen below. This diagram is based on IBM page 25 of the [ILD] (pdf pg 21).

One surprising feature of the translator is that it accepts binary inputs from 0 to 15, not just "valid" inputs 0 to 9. Input 10 is treated as 0, since the 1401 stores the digit 0 as decimal 10 in core. Values 11 through 15 are treated as 3 through 7. Thus, every binary input results in a valid (but perhaps unexpected) qui-binary value. As a result, the 1401 will perform addition on non-decimal characters.



The circuit in an IBM 1401 mainframe to translate a BCD digit into qui-binary code.

The logic is implemented by an AND-OR logic structure[37] that is common in the 1401. Each bit of the BCD digit, as well as each bit's complement, is provided as input. Each AND gate matches a specific bit pattern, and then the results are combined with an OR gate to generate an output.

In qui-binary, the B0 and B1 bits are trivially generated from the low-order binary bit. If the input is even, B0 is set, and if the input is odd, B1 is set.

The first AND gate above matches binary 1010 (decimal 10), and the second AND gate matches binary 000x (decimal 0 or 1). Thus, Q0 will be set for inputs 0, 1, or 10. Likewise, Q2 is set for inputs 2, 3, or 11. The other Q outputs are simpler, computed with a single AND gate.

The translation circuits for the A and B registers are similar, although the A register path also has the complement circuit, which will be described later.

9's complement circuit

The complement circuit is used for subtraction or negative numbers. The circuit generates the 9's complement of a digit, i.e. 9 minus the digit. For example, the 9's complement of 3 is 6, and the complement of 5 is 4. To subtract a number, the 9's complement of each digit is added (along with a carry). For example, consider 432 - 145. The 9's complement of 145 is 854. 432 + 854 + 1 = 1287. Discarding the top digit yields the desired result 287.

[37] Remember that ILD logic symbols are different from modern ANSI symbols; See Figure 5.

To see how complementation works in qui-binary, consider 3 (Q2 B1). Its complement is 6 (Q6 B0). The general pattern for complementation is B0 and B1 get swapped. Q0 and Q8 are swapped, and Q2 and Q6 are swapped. Q4 is unchanged; for example, 4 (Q4 B0) is complemented to 5 (Q4 B1).

The 'complement' circuit below uses AND-OR logic configured as a multiplexer. For each output, one of two inputs is selected, based on the complement input. (The box labeled I is an inverter.) If complementation is not selected, each input passes through to the output unchanged. This circuit is from page 25 of the ILD. If complementation is selected, the lines are swapped as described earlier. Output B1 comes from input B0, while output B0 comes from input B1. Q0 and Q8 are swapped, and so on.

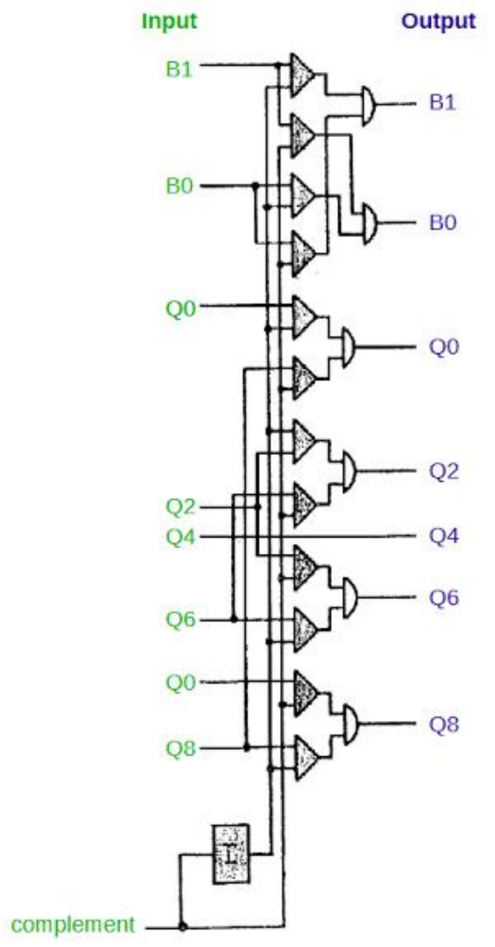


Figure 35: IBM 1401 Complement Circuit

Quinary adder

The circuit below adds the quinary parts of the two numbers. The qui part of the A register is on the left, the qui part of the B register is on the top, and the qui output is on the right. Each qui result has separate outputs depending if there is a carry or not. An example is Q2 + Q4, shown in red. These two inputs trigger the Q6 output. This circuit is from IBM page 25 of the ILD.

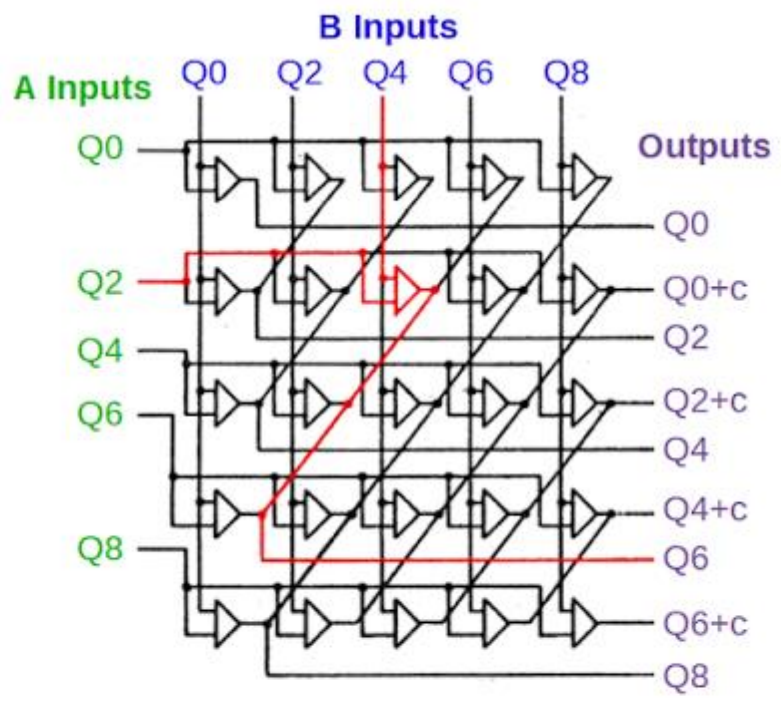


Figure 36: Quinary Addition Circuit, adding quinary parts of two qui-binary digits

The quinary adder is implemented with AND-OR logic, but it uses wired-OR logic. Instead of an explicit gate, the AND outputs are wired together to produce the OR output. While the quinary adder looks symmetrical and regular in the schematic, its implementation uses three different SMS cards: 3JMX and 4JMX AND/OR gates, and JGVW AND gates, depending on the number of AND gates feeding the output.

Qui-binary to BCD output

The diagram below shows the remainder of the qui-binary adder, based on page 26 of the ILD. The qui-binary carry circuit, in the blue box, processes the carry signals from the adder circuit. Using simple OR gates, it generates the qui signals and the qui carry signal. The next circuit, in the green box, applies any carry from the B bits, incrementing the qui component if necessary. For instance, adding 3 + 5 is Q2 B1 + Q4 B1. This generates Q6 + B0 + B carry. The B carry increments the qui component to Q8, yielding the result Q8 B0 (i.e. 8).

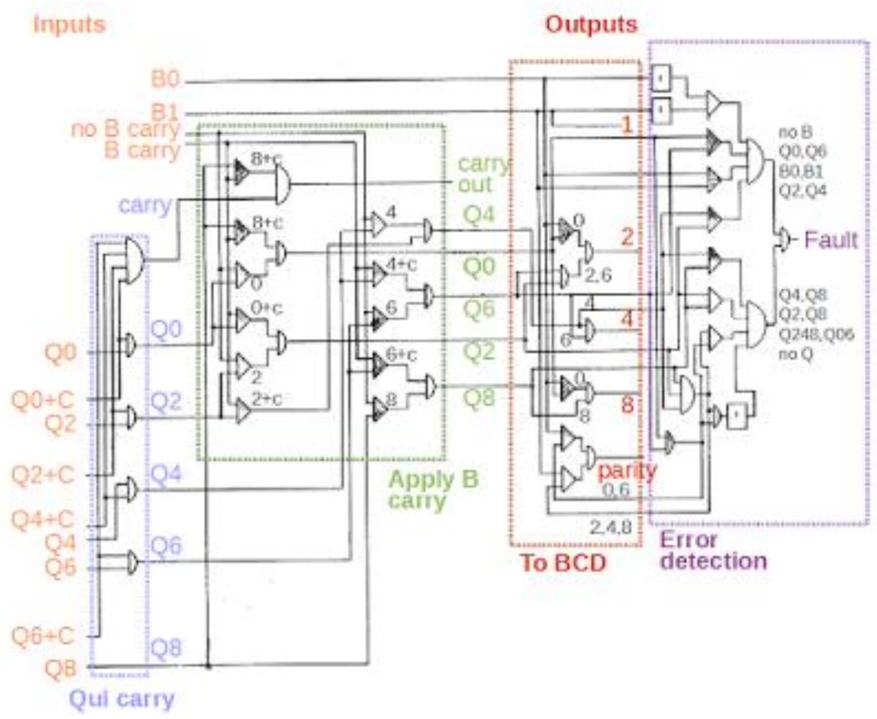


Figure 37: Conversion a Qui-Binary Sum to a BCD Result

The translation circuit, in red, converts the qui-binary result to BCD, using straightforward AND-OR logic. Note that 0 is represented in the 1401 as binary 1010, so the 8 and 2 bits are set for Q0 B0.

The parity output is generate by combining the binary parity (even for B0; odd for B1) with the qui parity value. The qui even parity signal is set for Q0 or Q6, while the qui odd parity signal is set for Q2, Q4, Q8. Note that representing 0 as binary 1010 instead of 0000 doesn't affect the parity.

The final circuit, in purple, is the error detection circuit which verifies the qui-binary result is valid. The AND-OR circuit detects a fault if no B bits are set or both B bits are set. Instead of testing every qui bit combination, it implements a short cut from the qui parity circuit. If the even qui parity signal and the odd qui parity signal are both set, this indicates multiple qui lines are set, triggering a fault. If neither qui parity signal is set, then no qui lines are set, also triggering a fault. The parity check misses a few qui combinations (such as Q0 and Q6 set), so these are tested separately. The result is that any invalid qui-binary result triggers a fault.

Conclusion

Addition/subtraction on the 1401 is considerably more complex than the single-digit addition discussed here. First, the signs must be checked to determine if the operation is a true-add or complement-add. Multiple digits are processed until a word mark is encountered. Overflows are counted using the zone adder. A negative result must be recomplemented. Arithmetic on the 1401 and the qui-binary adder are discussed in detail in 1401 Instruction Logic [R25], pages 49-67

Qui-binary should be distinguished from the more common bi-quinary encoding, where the bi part is 0 or 5, and the quinary part is 0, 1, 2, 3, or 4. Bi-quinary is used in abacuses as well as the IBM 650 and various Univac models. An article by Carl Claunch describes the history leading up to qui-binary arithmetic in detail. A later IBM patent (#3308284) describes qui-binary arithmetic using tunnel diodes.

5.4.4 I/O Overlap

5.4.5 Multiplier

5.5 Power system

Power-up and Power-Down sequencing?

[Contributed by Ken Shirriff, edited by Guy, July 2015]

One unexpected feature of the 1401 system is that the 1402 Reader/Punch provides the power to the 1401 and the rest of the system. The external AC line power is not connected to the 1401, but only to the 1402, where the 1402 provides power to the 1401, which then powers all the other peripherals (printer, tape drives, external memory).

The 1402 receives three-phase 208 (or 230) volt AC line power, which goes into circuit breakers (i.e. overload-protection breakers), contactors (relays to switch power on and off) and then into power supplies inside the 1401 and 1402. The Power On, Power Off, and Emergency Off buttons on the 1401 are actually wired to the 1402, since that's where the power is connected. The 1402 feeds power to the 1401 through two thick power cables, providing -20V DC, -60V DC, +/- 3V DC marginal check, 115V AC, 130V AC, and three-phase 208V AC to the 1401.

The 115V AC supply is simply used to power the convenience outlet and blower fans in the 1401 chassis.

The 130V AC delivered to the 1401 chassis is conditioned by a ferro-resonant transformer,[38] a large, passive device[39] that reduces voltage variations presented to the various AC-DC bulk power supplies in the system.

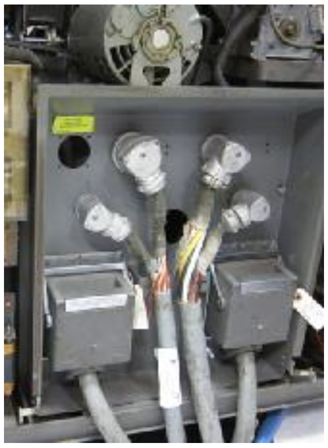


Figure 38: Power and Signal Cables in the 1402

The 1401 powers its own logic circuitry, and also feeds -12V DC, +30V DC, and +6V DC back to the 1402. Each power cable splits in two at the 1402, so there are 4 power connectors on the 1402: PWA, PWB, PWC, and PWD. The 1401, of course, has additional power supplies.

The 1401 Model D system is an exception to all this. Since it doesn't include a card reader, the 1401 receives AC line power directly, driving power supplies in the 1401 chassis. Since several additional gates (01B4, 02A7, 02A8, 02B7, 02B8) are used for power supplies, the Model D does not support multiplication/division.

For details on power wiring, see Installation Manual - Physical Planning. IBM 1401 Data Processing System, page 24.

[38] Ferro-resonant transformers provide some regulation by arranging that a portion of the iron core goes further into 'saturation' as the input voltage increases. See http://www.electroncoil.com/ferroresonant_transformers.php. See also http://ibm-1401.info/60cycleSMSPowerSupply.pdf for more on ferro-resonant transformers and IBM SMS power supply design in general. And see http://ibm-1401.info/TricksFor50-60cycles.html#ferroresonant

[39] Robert Garner says: "Btw, this is where the term "big iron" came from (the large iron-core transformers)" Got a citation?

5.5.1 Marginal Power System

[The power supply manual http://ibm-1401.info/60cycleSMSPowerSupply.pdf implies that the marginal voltage supply is actually wired temporarily in series with a main supply voltage to do marginal testing… is that true??]

6. Peripherals

[make a block diagram;]

The key point to introduce this section is that so much of the I/O logic is in the 1401 itself.

- The 1402 reader/punch has relays for motion control, but no logic for reading or decoding cards; signals from read brushes are wired one-by-one through giant cables to logic in the 1401 chassis.
- Similarly, magnet drivers for the card punches are located in the 1401 and wired to the 80 individual punch actuators.
- The 1401 chassis also contains 132 magnet drivers to activate print-hammer solenoids in the 1403 printer.

6.1 1402 Card Reader/Punch

“About the Oily, Geary Things”

http://ibm-1401.info/IBM-229-4016-1-IBM1402FE-PP-150.pdf

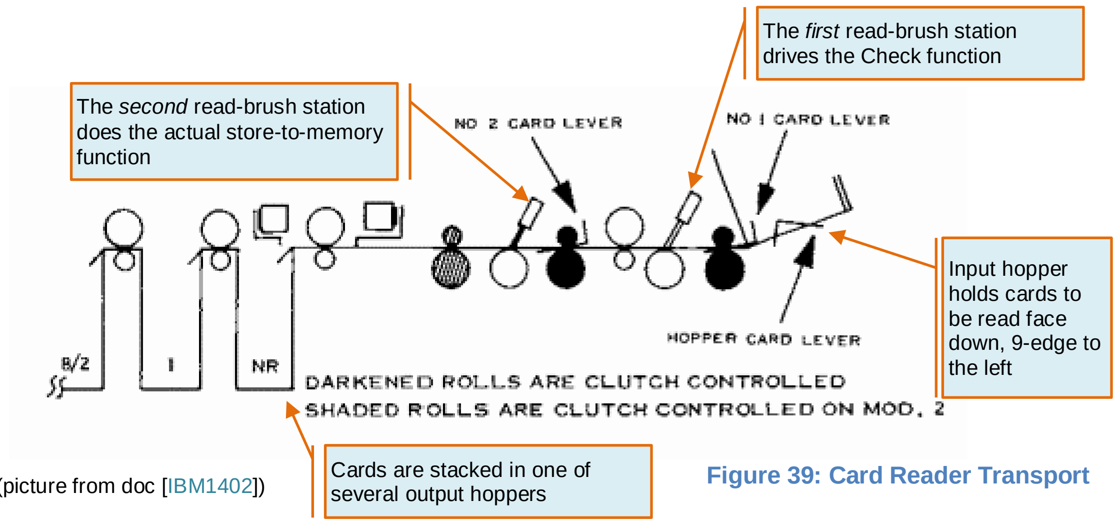
Add a picture from the 1402 doc, pdf pg 56

6.1.1 Reader

The 1402 card reader design is said to have been be leveraged from 088 (or 188) Collator machine. In this design, cards are carried past the read stations “9 Edge First” (See Section 3.1.2 for a review of punched card coding) by a mechanical transport that picks one card at a time from the input hopper, then pushes each one past two sets of electrical ‘read brushes’ -- sets of eighty tiny wire brushes, each arranged to contact a ground point when there’s a hole in the card at its particular column, but not conduct current when there’s no hole. The mechanical transport carries each card past the read stations and then deposits it in one of three stacker pockets.

The transport carries each card past the Check station first, then the Read station.[40] It’s important to note that the transport has several cards in flight at once, so while the Read station is reading one card, the Check station is reading the following card.

[40] Aaargh!!! How can they Check it before they’ve Read it?? Push onwards, dear reader. The “Check” function actually accumulates the parity of the number of holes that were found in each column. It does that first, then again at the read station. At the end of the process, if the parity hasn’t returned to zero, then the Check and Read stations must have seen different numbers of holes in the column, indicating an error of some sort.


The *first* read-brush station drives the Check function
The *second* read-brush station does the actual store-to-memory function
NO 2 CARD LEVER
NO 1 CARD LEVER
Input hopper holds cards to be read face down, 9-edge to the left
HOPPER CARD LEVER
R/2
1
NR
DARKENED ROLLS ARE CLUTCH CONTROLLED
SHADED ROLLS ARE CLUTCH CONTROLLED ON MOD. 2
Cards are stacked in one of several output hoppers

(picture from doc [IBM1402])

Figure 39: Card Reader Transport

There's a second transport in the 1402 for the card-punch function; in that one, cards start from the opposite end of the machine and flow left-to-right (instead of right-to-left as above). The punch system goes "12 edge first" so that cards from the read path and the punch path land in the output hoppers the same way around. More on the punch system in Section 6.1.3.

For the card reader, a mechanical transport that reads eighty columns at a time had two advantages:

- It's what Collators and Sorters did, so the mechanical design could be easily leveraged
- It's faster; for a given rate of card movement (around 6.5 feet per second, no?) it's possible to process more cards if they are read 80 columns at a time rather than 12 rows at a time.

The card reader reads each card twice to ensure that it's error-free:

- The first set of brushes is the "Check" stage, where the "count"[41] of the number of holes in each column is accumulated in 80 sets of check-bits stored in core
- The second set of brushes also accumulates a count of the number of holes, but then causes the read data to be written to Core.

The machine halts if the hole-count check doesn't get the same answer at the Check and Read stations, and the Reader Check Light is illuminated on the 1402.

The row-at-a-time reader mechanism did leave the 1401 designers a complex problem of storing 80 partial results in core memory as they arrived row-by-row. The actual mechanism to get card data transferred to memory is rather complex, but is described in detail in the Instruction Logic manual [R25], starting on Page 76.

In understanding this material, it's helpful to remember a couple more points:

- The 1402 itself contains no SMS logic; as a result, 80 read-check wires, 80 read wires, 80 punch wires and 80 punch-check wires are cabled from the 1402 to the 1401 in two thick 200-pin cables.
- The 1402 motion control is all done by relay logic and cam-operated electrical switches called Circuit Breakers in the 1402 chassis, although there's an optical sensor called the Solar Cell CB[42]

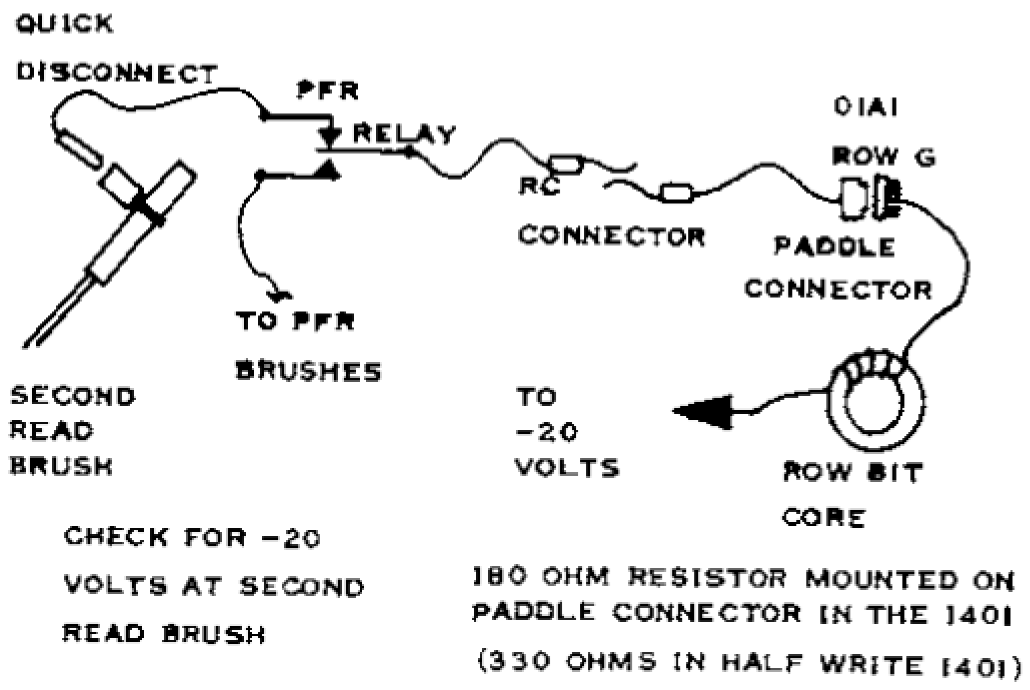
[41] The "count" is compressed to two bits per column; one bit to say there was at least one hole, another to accumulate the parity of the count.

[42] The optical sensor is called a Solar Cell Circuit Breaker in the IBM docs. In this context, a Circuit Breaker is any kind of switch activated by a cam or mechanical actuator.

in the mechanical path that generates a 600 usec pulse each time the card is positioned under the brushes ready to read a row.

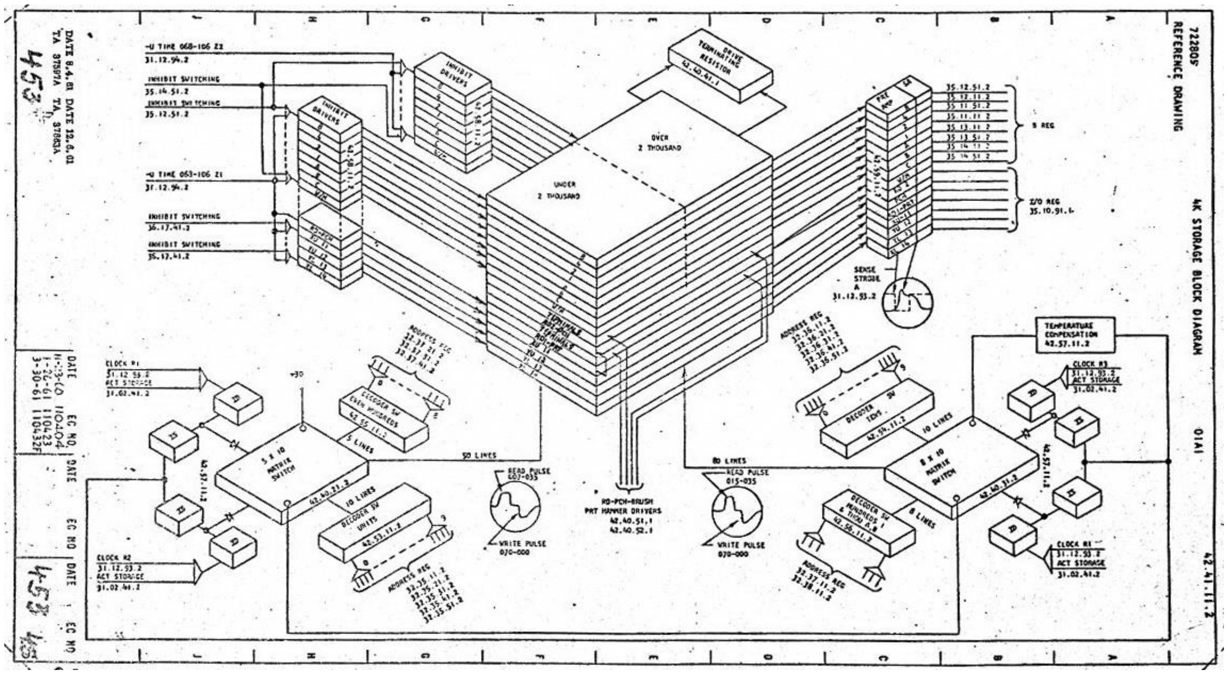
All of these partial results and hole-counts are stored in a special reserved segment of core memory, shown in the logic diagrams at [ALD], page 42.41.11.2. This additional memory is addressed using the same STAR mechanism as regular memory, but the read-write data is passed through a separate data plane specialized for I/O operations (card read and punch, [and printer functions?]).

IBM-229-4016-1-IBM1402FE-PP-150-reader-punch-ce-manual.pdf



(From IBM-229-4016-1-IBM1402FE-PP-150 Reader-punch-ce-manual.pdf, doc pg 2-7)

While these additional planes can be accessed using normal 11.5 usec read-clear-write cycles, there is also special wiring for reading cards. Each read brush (all 160 in the reader side of the 1402, plus 80 more to check the punch results) is connected directly to a wire that passes through exactly one core in this Row-Bit Plane. Synchronized by the pulse from the Solar Cell CB, that allows all 80 bits from a particular row to be stored into 80 cores simultaneously, called Row-Bit Cores, without the use of multiplexers or sequencers.



(full sized image of ALD page 42.41.11.2 at http://ibm-1401.info/ALDs-VSnyder-Australia/5_7/722805.pdf)

The rest of the operations to the I/O core planes are conventional read/write cycles, where cores can be read-and-cleared or written one core at a time.

Once the Row Bits have been written, a scan of the 80 locations (addressed as 1 to 80) is started to do checks and move results to the main core memory.

The scan uses the B-STAR register to generate core plane addresses, which address both main memory, the row-bit and check planes.

For the Check operation, there are two bits per column, stored in the Check Plane, which are used as follows:

- One bit is set to indicate that at least one hole was found in the column
- A second bit is flipped each time a hole is found in the column, effectively keeping a running parity of the number of holes.

When the card passes through the Read station, the contents are written to Row-Bit cores as above, and the hole count is accumulated as above. But the scan also copies contents of the row into main memory:

- At the start of the scan, the A Register is loaded with the BCD value corresponding to the row just read (e.g., on row 9, the A Register would contain 8 and 1 bits.)
- On Row 9 (the first row), the A register is written to the core memory location corresponding to the column if a hole is present (i.e, column 1 goes to core location 1[43], column 10 goes to core location 10, etc).
- For subsequent rows, for each column with a hole, the contents of the A Register are ‘combined’ [mostly “OR’d”, no?] with whatever was in the core memory location. This way, the numeric part is accumulated and then the zone bits are added when the card gets to rows 0, 11 and 12. (See Section 3.1.2 again to remember how Hollerith card encoding works)

[43] Why location 1, not 0? Because “accounting machine people would never understand zero-base counting”. Don’t worry, location zero is not wasted, it’s used for a row counter.

As always, there are special cases for Zero (stored as 0b1010).

- At the end of the processing each row, core memory location zero is updated, acting as a row counter

At the end of the 12-row (the last one), the Check planes can be compared to make sure Read and Check stations saw the same number of holes in each column.

To read the many more details, see the following docs:

- Instruction Logic manual [R25] pdf pg 76; and
- IBM1401 Data Flow (Form G24-1477-0) [G24Flow] pg 49 (pdf pg 53)
- IBM1402 Reader Punch Customer Engineering Service Index [IBM1402] (Form 229-4016-1)

Notes

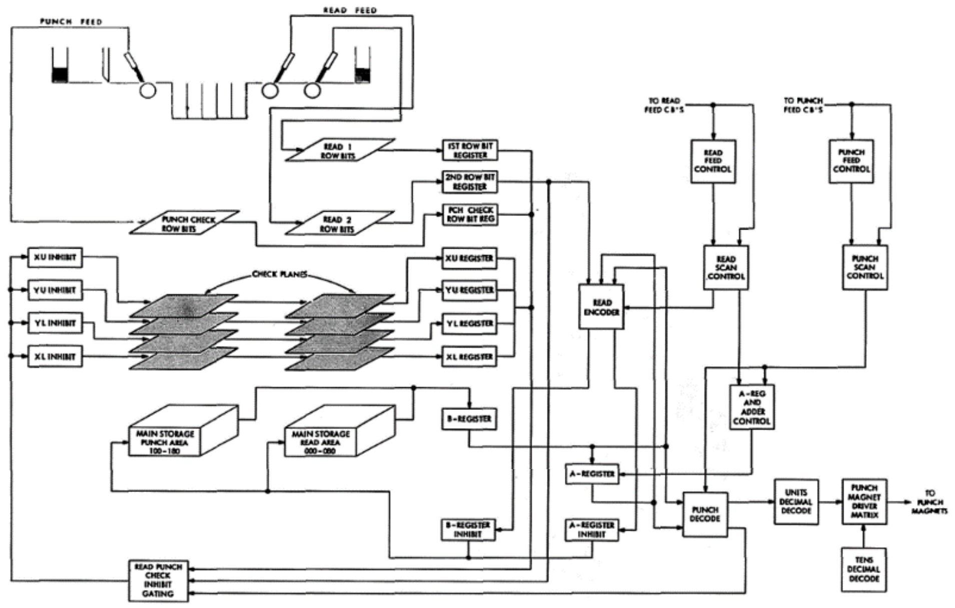
- Just in case this is sounding too straightforward, remember that different cards are present at the Check and Read stations. So there are actually two sets of extra core planes for Row Bits and Check bits, and they're used alternately. Thus, the planes used for hole-counting are XU, XL, YU, and YL, each consisting of 80 bits.
- There were two vacuum tubes hidden in the 1402 chassis, which are part of the assembly that is used to set timing of the mechanical feed path. (See Section 6.1.5)
- Mechanical timing in the 1402 must be carefully maintained; the Service Manual [IBM1402] describes how to adjust timing for the machine.[44]

6.1.2 Logic Implementation

[The following notes are from Ken Shirriff]

The diagram below shows data flow through the 1402 and 1401. The 1402 circuitry is in the upper-left and all the rest is the 1401. This figure is from the 1402 Customer Engineering manual page 5-2.

[44] The Expert Advice for the timing adjustment procedure is to start on the first page and progress to the last; trying to tweak starting in the middle of the process is the path to grief.



The Reading Algorithm

The 1401 uses a surprisingly complex process to read a card into memory. You might expect that the card reader reads each character of the card and sends the character to the 1401 as 6 bits. Or maybe the card reader sends each character as 12 undecoded positions. But the real mechanism is totally different: the card is read sideways, and all 80 positions in a row are sent to the 1401 in parallel. The card reader is entirely electro-mechanical, and all the processing happens in the 1401.

The following is a simplified description of the card reading algorithm. The algorithm is described in detail in 1401 Data Flow, pages 49-54.

The rows of the card, from 9 through 0, then 11 and 12, are processed one at a time.
After each row is read at read station 2, it is processed as follows:
First, the A register is set to the value of the row: 9 through 1, then zone A (for row 0), zone B (for row 11) and zone AB (for row 12).
Next, the B address register steps through the columns 001 to 080. For each column, the current memory value is read into B. If the associated row-bit core is set, the A register value is combined with the B register value. The result is written back to memory to update the character according to the hole's value.
At the end of this process, locations 001 through 080 hold the characters from the card.

Thus, each position of the card is processed individually, from the lower left to the upper right.

6.1.2.1 The Read Circuitry

This section discusses some of the card reader circuitry, focusing on the punch card code processing and how exceptions in the punch code are handled. The circuitry is housed in gate 01B4. The Intermediate Level Diagrams (ILD) show the gate-level circuits for the reader/punch in diagrams 60 through 68.

The value of the A register corresponds to the current row (9 through 1, then zones A, B, and AB). The B register holds the value currently stored in memory for that character. The signal RD2 at 60B1 (i.e. ILD #60 section B1) indicates a hole at the current row/column.

Reading a zero needs to be handled specially: it is punched in the zero row (i.e. zone value A), but is stored in BCD as 10. This is handled by the gate at ILD 60B1, which causes BCD 10 (8+2) to be written to memory.

The IBM 1401 handles MLP (multiple-line printing) punches, a feature of the IBM 403 tabulating machine (p113). A MLP card was indicated by a 9, 8, and another punch all in a single column. The 1401 ignores the 8-9 punches on an MLP card, reading the remaining value (1401 Reference manual, p170). The gate to handle a 9-8 MLP punch is at 60C1. It causes the read value to be cleared when a 9-8 punch is detected, so the remaining punch will be the value read.

The gates at 60C1 detect the first row (9 normally, but for column binary, row 3 restarts the process.)

At 60D3, two digit punches for the same character (excluding 8) triggers a validity check. The gate below triggers a validity check for two zone punches. Note that 11-0 and 12-0 are okay because the 0 will be converted to BCD 10 before the zone row is read. The gate below that triggers a validity check for 8-1.

The gate at 60B3 triggers a validity check for 8-2, unless it is 0-8-2.

At 60B4, the gate with puzzling inputs "A reg not 4, A reg 1, A reg not C" turns out to match the row 1. It is used to switch processing from numeric rows to zone rows.

The data flow to the A and B registers is on ILD pages 10 and 11. The gates at 60D2 control the inhibit lines for A and B, controlling updates to these registers.

6.1.2.2 Hole Counting for Validity Checks

When the card passes through the first read station, the holes in each column are counted. At the second read station, the hole count is verified to check the validity of the read.

It turns out, though, that it's not exactly a count but a simpler two-bit value used in this process. The two bits start off cleared. The first bit (denoted U) is set when a hole is encountered. The second bit (denoted L) is toggled on each hole. Thus, three states are distinguished: no holes, odd number of holes, and even number of holes.

For the second read, the transitions are reversed. The first bit is cleared when a hole is encountered, and the second bit is toggled. If all goes well, both bits will be clear after the second read. Otherwise, there is an error.

This will catch missing a single hole, or missing all the holes. There are some count errors it will miss, such as reading 1 hole versus 3, since it doesn't store exact counts.

Hole counting requires two separate sets of storage, since while the first card is being validated, the second card is being read. For each card, the 1401 toggles between two sets of row bit storage planes, labeled X and Y. Thus, the planes used for hole counting are XU, XL, YU, and YL, each consisting of 80 bits.

6.1.2.3 The Hole Counting Circuitry

At ILD 62C1, a trigger switches between row bits X and row bits Y on each card. For state X GATE, row bit X indicates a hole at read station 1 (RD 1) and row bit Y indicates a hole at RD 2. For Y GATE, the bits are swapped. This selection is done by the gates at 62C3. (During punching, the bits come from either Punch Check Decode or B Reg Punch.)

Latches at 62C3 hold the current state of the bits as read from the cores.

At 62C4, gates generate the new values for the hole counts. XU (or YU) is set when a hole is detected during the count phase and is cleared when a hole is detected during the verify phase. This will trigger an error if a hole was detected at RD 1 but not RD 2. It ignores the case where a hole was detected at RD 2 but not RD 1. XL (or YL) is generated from the XOR of the old value and the current row bit, so the value is toggled on a hole.

The gates at 62B6 trigger an error if either hole count bit is set at the end of the process.

6.1.2.4 The Card Reader and Core Memory

Card reading uses special core planes separate from the regular memory planes. Six additional planes are used: two for the row-bit cores, and four for the hole counting. These planes are not fully populated since there are only 80 columns to store. These planes have separate outputs from the regular memory planes, but use the same addresses (001 through 080).

Read station 1 is connected to cores in plane RD1 (frame 12) and read station 2 is connected to cores in plane RD2 (frame 10). Surprisingly, each brush is wired directly to a particular core with a separate wire, rather than using the inhibit line like the regular cores. Thus, the row can be written to the row-bit cores in parallel, without addressing. Two separate frames (9 and 11) consisting just of terminals and wiring sit above RD1 and RD2 to handle the connections to individual cores. Four planes are also used to count the holes: XU 11, YU 12, XL 13, YL 14 (frames 13-16). Cores in these planes are written in the standard way, using inhibit lines.

The sense lines for each plane are shown in ILD 4 and the inhibit lines are in ILD 3. The details of the core memory are given in ALD 42 (page 13).

6.1.2.5 Conclusion

Card reading on the IBM 1401 is surprisingly complex. Massive cables connecting the brushes in the card reader directly to individual cores in memory. The logic for this operation is all in the 1401 itself, with the computer processing each of the 80 by 12 card positions individually to generate the card characters in storage. Multiple validity checks ensure the accuracy of this process

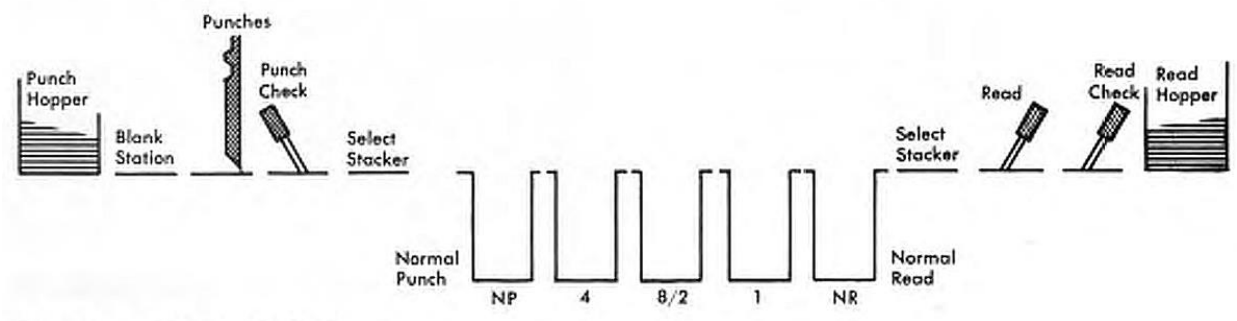
6.1.3 1402 Card Punch

6.1.4 The 1402 Card Reader/Punch Mechanism

[contributed by Ken Shirriff, July 2015, edited by guy]

The IBM 1402 high-speed card reader/punch can read 800 cards a minute or punch 250 cards per minute. Given that most 1401 installations (except the Model D) included a 1402 card reader/punch, it was a key component of the 1401 system. The 1402 supported high-volume card processing, with a file feed that could be loaded with 3000 cards at a time. A careful operator could add cards to the feeder as the machine ran, allowing almost continuous operation. The card reader and punch provided high accuracy by reading each card an additional time to verify the hole counts. It also provided misfeed, jam, and invalid punch detection.

The following diagram shows the layout of the 1402. The punch takes cards from the left, they pass through the punches and then the punch check brushes, and go into one of the stacker bins. The reader takes cards from the right, they pass through the two sets of brushes, and are stacked.



Card feed schematic from Reference Manual. IBM 1401 Data Processing System, page 10.

If the PFR (Punch Feed Read) feature is installed, a second set of brushes is installed in the "blank station" on the left. This allows a card to be read and then punched in a single pass. See Section 6.1.5.

6.1.4.1 Principles

This section outlines some basic principles of the 1402.

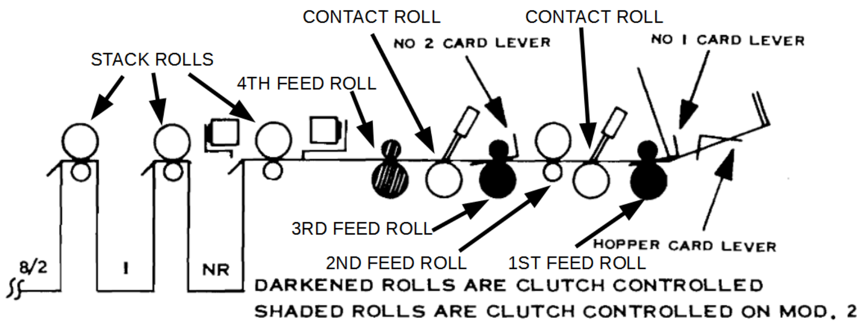
- Cards are read and punched row-by-row, not column-by-column, operating on a row of 80 positions all at once.
- The read and punch sections are opposite. The reader moves cards right to left, 9 edge first. The punch moves cards left to right, 12 edge first, ensuring that reader-unit and punch-unit cards would end up stacked the same way in output hoppers. With a bit of care, skillful programmers would be able to merge read and newly-punched cards in the same output hopper.
- The read and punch units are mostly independent: separate motors, separate clutches, separate feed paths, separate brushes, and separate cables to the 1401.
- [guy] Ken, 'Mostly independent' begs the question of what wasn't independent... I assume power and chassis were shared, but do you think there are other shared elements? Maybe there was some kind of shared control over the stacker bins?

  The reader runs at 800 cards per minute and the punch runs at 250 cards per minute. These are the key timing numbers that everything depends on.
- The reader has an 800 RPM drive shaft. Everything is timed off the rotation angle of this drive shaft: 0 to 360 degrees, and one rotation corresponds to one card. The mechanical components are synchronized to this shaft via gears and belts. The electrical timing is synchronized to this shaft via cams that open and close circuit breakers (switches). Likewise, the punch timing is synchronized to the 250 RPM drive shaft.
- The reader feed mechanism has some parts that run whenever the reader motor runs, and some parts that run only when a card is being fed. A clutch separates them, and the clutch is triggered for each card. Likewise, the punch has a clutch separating the clutched and unclutched mechanisms, and the clutch is triggered for each card to punch.
- The card reader is electro-mechanical, using relays to provide simple logic control. (Exceptions: a few transistors to amplify the solar cell signal. Transistors on SMS cards for power supply regulation. Two vacuum tubes (!) power the diagnostic dynamic timer.)
- The 1401 triggers the reading or punching of each card. Once triggered, the complex timing of the read/punch is under control of the 1402.
- The card processing logic is all in the 1401. The 1401 converts hole patterns to characters and performs error checking, as described in Section 6.1.2. The 1402 has brushes to read holes and punch magnets to punch holes, but does not interpret the cards at all. The 1401 has the punch drivers to drive the punch magnets in the 1402.
- The thick reader cable between the 1402 and the 1401 includes 160 wires to connect each brush at the two read stations to the 1401. The thick punch cable includes 80 wires for the punch check brushes and 80 wires to drive the punch magnets (see Figure 38). Each brush is connected directly to a core in a special plane in the 1401's core memory, where current through each brush directly writes the corresponding core, in parallel, without any addressing.

6.1.4.2 The Card Feed Mechanisms

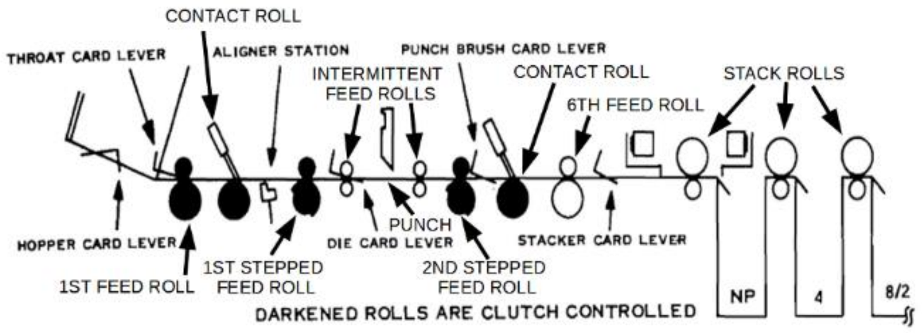
The diagram below shows the reader feed mechanism. Cards pass right-to-left through the mechanism. The feed rolls move the cards, while the contact rolls are used along with the brushes to read the holes. The stack rolls feed cards into the appropriate stacker. The mechanical drives fall into two sections: unclutched, which run whenever the motor is running; and clutched, which run only when the clutch is engaged, as indicated in some of the figures below.

Card levers detect the passage of cards through the unit. The levers indicate when a card is in position to be read. Levers are also used to detect jams.



Cards are read in three machine cycles. In the first cycle, the card moves from the hopper to the read check station. (At this point, the card is still half-visible in the hopper.) In the second cycle, the holes are counted at the read check station and the card proceeds to the read station. In the third cycle, the card is read, the count is checked, the data is sent to the 1401, and the card proceeds into the hopper.

The diagram below shows the punch feed mechanism. Cards pass left-to-right through the mechanism. The stepped feed rolls are eccentric, so they grip the card only during part of the revolution. The intermittent feed rolls are driven by a geneva drive - the card is moved to each of the 12 punch rows and held stationary while being punched.



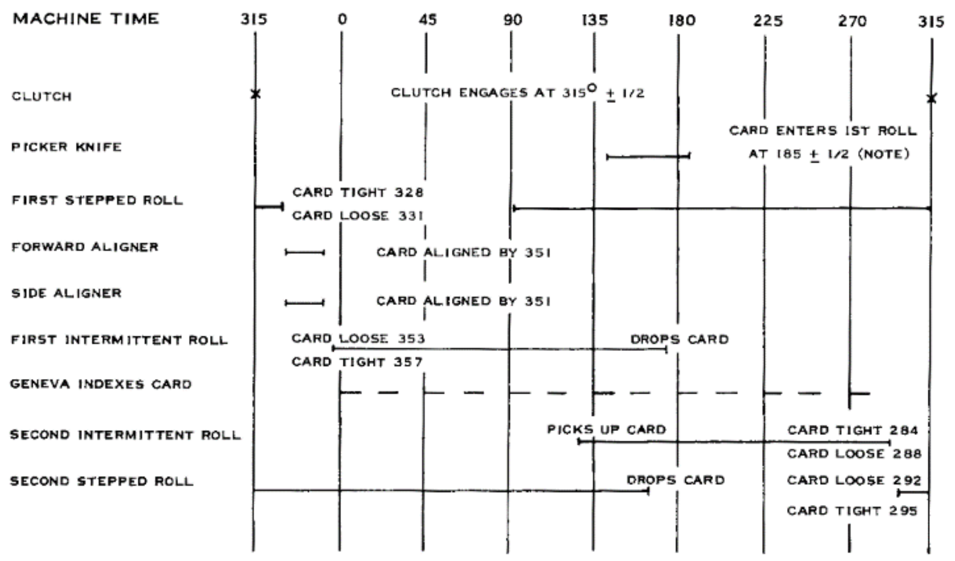
A punch operation takes four machine cycles. In the first cycle, the card moves from the hopper to the blank station. In the second cycle, the card is moved to the punch station. In the third cycle, the card is punched and moves to the punch check station. In the fourth cycle, the hole count is verified at the check station and the card continues into the stacker.

The punch timings are shown in the diagram below. In the first cycle, the card is selected by the picker knife and picked up by the first feed roll. In the second cycle, it is fed by the stepped roll. In the third cycle it is advanced through the punch station by the intermittent feed rolls. The intermittent rolls are driven by the Geneva[45] drive, which holds the card steady at the 12 punch positions. In the fourth cycle, the second stepped roll feeds the card through the punch check station to the feed roll and stacker rolls. The diagram is indexed by "machine time" - the angle from 0 to 360 degrees, with each cycle starting at 315 degrees when the clutch engages. This diagram is from the 1402 Customer Engineering manual, page 4-5.

[45] A Geneva Gear is a mechanism by which continuous rotary motion of a drive shaft is converted to a start-stop motion, which happens to be perfect for punching cards… The Geneva drive automatically advances the card by one row, then holds it steady while the punch mechanism is activated, driving punches through the card and removing them in time for the move to the next row. There's an animation of a Geneva Drive at http://ibm-1401.info/GenivaMechanism.gif. See also https://en.wikipedia.org/wiki/Geneva_drive.

For each machine cycle, a new card enters the reader (or punch) as the previous cards advance to the next station. There are two sets of check planes since the card in the check station is different from the card in the read (or punch) station.

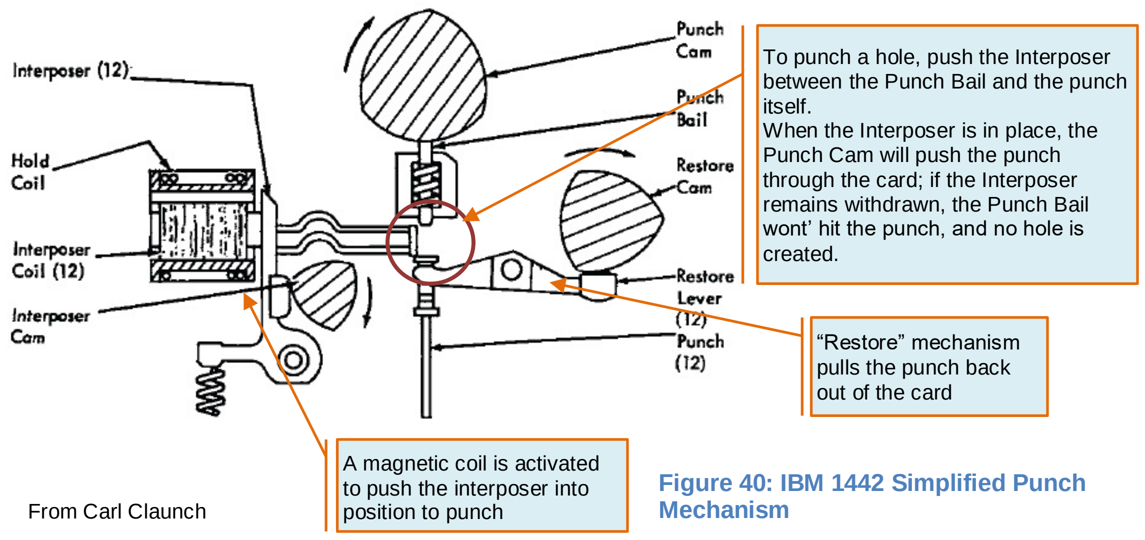
The 1402 automatically ‘pre-fetches’ the next card to keep the pipeline full. When a job stops, it may be necessary to use the Non-Process Run-Out key to move remaining cards through the mechanism and into the stacker, clearing the way for the next job.



6.1.4.3 The Punch Mechanism

You might expect that holes are punched in the card by the 80 punch magnets, but the mechanism is a bit more complex. It takes quite a bit of force to push a punch through card stock, so a mechanism is used to control the punch indirectly.

A simplified version of the 1402 mechanism is shown in Figure 40, where the key idea is that a piece called an Interposer is inserted at just the right moment between the head of the punch and a “Punch Bail,” driven by a cam turned by a powerful motor. If the Interposer is in place, it transmits force from the cam through the bail to the punch; if not, the cam spins without acting on the punch.


Interposer (12)
Hold Coil
Interposer Coil (12)
Interposer Cam
Punch Cam
Punch Bail
Restore Cam
Restore Lever (12)
Punch (12)
To punch a hole, push the Interposer between the Punch Bail and the punch itself.
When the Interposer is in place, the Punch Cam will push the punch through the card; if the Interposer remains withdrawn, the Punch Bail wont' hit the punch, and no hole is created.
"Restore" mechanism pulls the punch back out of the card
A magnetic coil is activated to push the interposer into position to punch

From Carl Claunch

Figure 40: IBM 1442 Simplified Punch Mechanism

The 1402 uses a slightly more complex mechanism shown in Figure 41 to do the same thing… again, an interposer pivots so it can be between the Punch Drive Bail and the punch, or out of the way. If the punch magnet is energized, the interposer will pivot between the Punch Drive Bail and the punch. Otherwise, the interposer is not in the path of the punch drive bail. The cam shaft moves the punch bail down, and if the interposer is in the way, the punch bail pushes the interposer into the punch, punching a hole in the card with very little force directly from the magnetic coil. If the interposer is out of the way, no hole is punched.

The punch mechanism is described in detail on page 4-14 (pdf pg 53) of the 1402 Customer Engineering manual. [IBM1402]

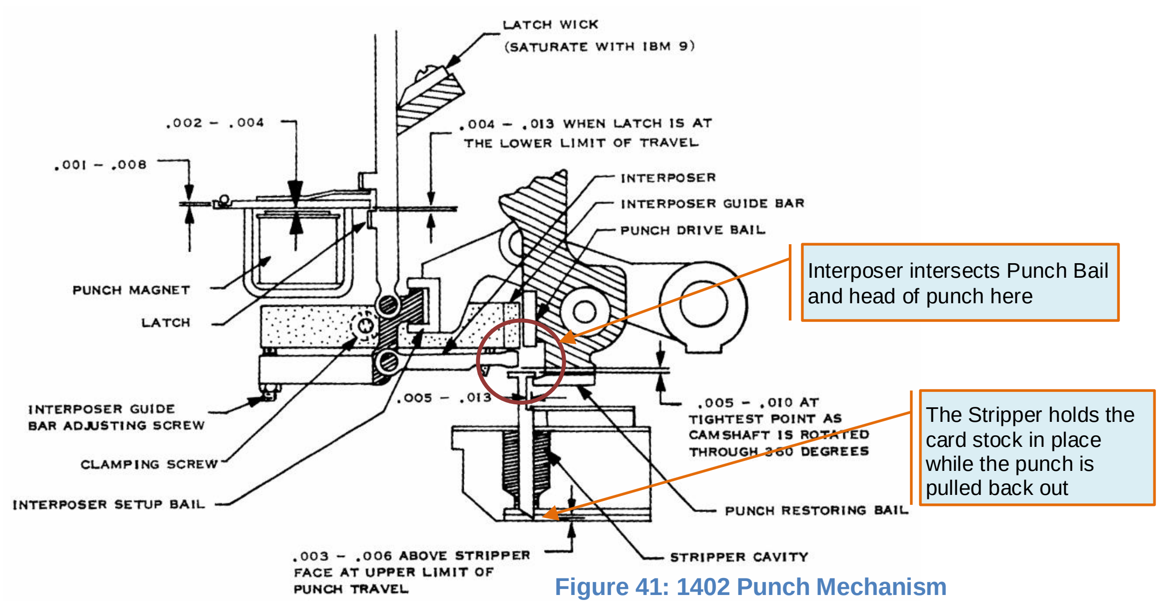

LATCH WICK (SATURATE WITH IBM 9)
.002 – .004
.001 – .008
.004 – .013 WHEN LATCH IS AT THE LOWER LIMIT OF TRAVEL
INTERPOSER
INTERPOSER GUIDE BAR
PUNCH DRIVE BAIL
PUNCH MAGNET
LATCH
Interposer intersects Punch Bail and head of punch here
INTERPOSER GUIDE BAR ADJUSTING SCREW
.005 – .013
.005 – .010 AT TIGHTEST POINT AS CAMSHAFT IS ROTATED THROUGH 360 DEGREES
CLAMPING SCREW
INTERPOSER SETUP BAIL
The Stripper holds the card stock in place while the punch is pulled back out
PUNCH RESTORING BAIL
.003 – .006 ABOVE STRIPPER FACE AT UPPER LIMIT OF PUNCH TRAVEL
STRIPPER CAVITY

Figure 41: 1402 Punch Mechanism

6.1.4.4 Drive Shafts and Timing Angles

The card reader is driven at 800 rpm, corresponding to 800 cards read per minute or 75 milliseconds per cycle. The card reader clutch has six teeth, so it will take a maximum of 25 milliseconds to engage,[46] providing the Early Read feature, which avoids waiting for a full 75 millisecond cycle if the 1401 spends too much time processing the last card. (See System Operation Reference Manual. IBM 1401. IBM 1460, page E-7 for detailed processor timings.)

The card punch operates at 250 cards per minute, so its main shaft is driven at 250 rpm, or 240 milliseconds / cycle. The card punch clutch has 4 teeth, so it will take a maximum of 60 milliseconds to engage.

These two drive shafts are the key to the timing of the card reader and punch, taking the role that a clock fills in an electronic circuit. One rotation of the shaft corresponds to one mechanical cycle - one card read or punched. Everything is timed relative to the rotation angle of the shaft, from 0 to 360 degrees. The mechanical components are synchronized to the driveshaft since they are connected via gears and toothed belts. The electrical components are synchronized via cams that are driven from these shafts.

Remember that the card reader and the punch are independent mechanically: there are two separate motors, two separate clutches, two separate timing angles, and so forth.

A visual representation of critical timing angles is provided for maintenance by the Dynamic Timers (See Section 6.1.6).

6.1.4.5 Cams

The electrical timing signals in the 1402 come from circuit breakers (CB) that are driven by cams. The shape of the cam controls the angle at which the circuit breaker turns on and off. (Note that a "circuit breaker" is essentially a switch in this context, unrelated to a protective circuit breaker that trips under current overload.)

The 1402 has several sets of cams and circuit breakers:

- RCCB: read continuous running circuit breakers
- RLCB: read clutched circuit breakers
- PLCB: punch clutched circuit breakers
- PCCB: punch continuous running circuit breakers
- PACB: punch continuous running circuit breakers (high speed)
- SCCB: solar cell circuit breaker (optical, not a cam)

The read cams are driven by the 800 RPM read shaft, so they trigger at the appropriate angle in the read cycle. Each cam is shaped to close and open the circuit breaker at the right time.

The RCCB cams are three-lobed to allow the clutch to engage at three different points in the cycle. (Section 6.1.4.4) (The “Early Read” option?)

The PCCB punch cams are driven by the clutched 250 RPM write shaft and close and open circuit breakers at the right angle in the punch cycle, defining the timing of the punch cycle.

The timing of the PLCB punch clutched cams is more complicated. Because the punch clutch has 4 teeth, it can engage at 0 degrees, 90 degrees, 180 degrees, or 270 degrees. Thus, the unclutched punch angles can differ from clutched punch angles by a multiple of 90 degrees. The unclutched cams are generally 4-lobed, so the signals switch on and off four times in a 360 degree revolution and it doesn't matter how the unclutched and clutched angles line up. In other words, the clutched cams have the true angle of the cycle, while the unclutched cams are modulo 90 degrees. (The motivation behind this is to

[46] Because the clutch turns 180 degrees during a read cycle, the 6 teeth correspond to three engagement points per cycle

reduce the latency of a punch operation that would be caused by the slower punch shaft. You only need to wait a maximum of 60 ms for the clutch to engage, rather than 240 ms for a full revolution.)

The high speed PACB circuit breakers produce a signal for each punch row through slightly complicated timing. They are driven by a shaft rotating at 1333 1/3 RPM, compared to 250 RPM for the regular punch drive, synchronized to the slower punch drive through gears. Punching each row takes 22.5 degrees (of the 250 RPM punch shaft), or 1/16 of 360 degrees. That gives 12 time intervals to punch and 4 for card movement. During the time the 250 RPM shaft spins 22.5 degrees, the high-speed 1333 1/3 RPM shaft will spin 120 degrees. Thus, 120 degrees of the high speed shaft corresponds to one punch row. The high-speed cams have three lobes separated by 120 degrees so they turn on and off three times in a revolution, once for each row of the card.

The solar cell circuit breaker (SCCB) generates a signal as each row of a card passes under the read brushes. It consists of a light-sensitive semiconductor (the solar cell), a light, and a rotating slotted disk, and is triggered when light shines through a slot. On readers that predate the solar cell, six cams were used in combination to generate the read signals.

6.1.4.6 Relays

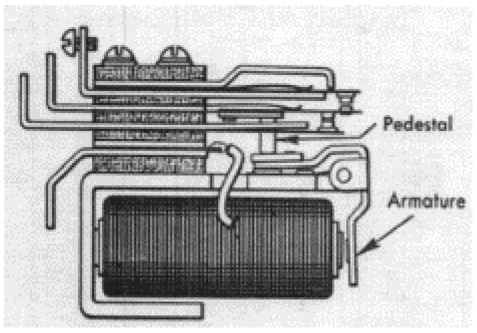


Figure 42: Duo Relay

The control logic inside the 1402 is implemented with relays, rather than transistors.

The relays are described on pdf-page 33 (IBM 35-36) and pdf-pg 35 (IBM 37-38) of the 1402 wiring diagram [Wiring1402]. Heavy duty relays switch power on and off for circuits in the 1042 and (strangely) the 1403 printer. Low-power, high-speed “permissive make” (PM) wire-spring relays are used for logic. These relays often have two winding: the "pick" (P) winding activates the relay, and the "hold" (H) winding will hold the relay in the activated state. Each relay has multiple sets of normally-open and normally-closed contacts. Relays are used for all kinds of logic functions, for example, to detect error conditions from the “card lever” (CL) switches at various points in the feed path. Other relays manage the run/stop state for the reader and punch and control the motors.

See pages 2-3 and 4-1 in the Customer Engineering Service Index [IBM1402] for logic diagrams. Note that although gate symbols are used, the logic is implemented with relays and diodes. The logic implemented by the relays is shown on page 5-3.

See Commutation and control for information on the types of relays used by IBM.

The 1402 is controlled by about 30 relays, plus 14 more for machines with the PFR feature (which does not include either CHM system) Relays are shown in the Wiring Diagram gate (pdf page 33]

6.1.4.7 Connection to the 1401

Besides the power cables (See Section 5.5), there are two thick signal cables between the 1401 and the 1402: one for the reader and one for the punch. The reader cable has 80 wires for the read brushes from the first station and 80 wires for the read brushes from the second station, all of which are wired directly to cores in the 1401 (Section 6.1.1). There are also about two dozen control lines. The punch cable has 80 wires for the punch magnets and 80 wires for the punch brushes. There are also about 18 control lines. (See wiring diagram page 29 (31/32)).

For a diagram of the row bit core wiring, see page 95 of Field Engineering Maintenance Manual. 1401 Data Processing System.

6.1.4.8 Reader-Punch Glossary

Chute blade:	Lever to direct cards into the appropriate stacker.
Early read:	An optional feature allowing the read clutch to engage at three points. This reduces the maximum latency to start a read from 75 ms to 25 ms. The readers in the CHM have this feature.
Intermittent feed:	Card feed roller driven by Geneva gear so the card stops at each punch row position.
Joggler:	Mechanical shaker to jog cards into position at input feed or in the stacker.
PA emitter:	Magnetic or mechanical signal triggered at each punch position.
Picker knives:	Blades to push a card off the stack and feed it.
Solar cell:	A semiconductor light sensor to detect each row position.
Stepped feed:	Eccentric card feed roller.

6.1.4.9 Understanding the wiring diagram

The wiring diagram is somewhat difficult to understand. This section lists some abbreviations and symbols used. Unless you're studying the wiring diagram, you may want to skip this section.

6.1.4.9.1 Relays

Relays are indicated by HD*n* (heavy duty), Du*n* (duo), or just *n* (permissive make). DU contacts are labeled AU, BU, AL, and BL. Under a relay coil symbol is a P (pick) or H (hold). The relay contact symbol is marked with the relay number; the contact symbols may be on different pages than the coils. See wiring diagram page 35 (37-38) for a list of relays and the page sections for each relay.

6.1.4.9.2 Cams

PA, PC, PL, RC, RL: cams (described above). Cams are often marked with the make (M) and break (B) angles. Cams are described on page 22 (23-24) onwards.

6.1.4.9.3 Connectors

PWA, PWB, PWC, PWD: the power connections. RC: the 200-pin read connector for the cable to the 1401. (Also reader cam.) PC: the 200-pin punch connector for the cable to the 1401.

6.1.4.9.4 Switches

Switches and lights are listed on page 19 (19-20). SW: switch. BS: brush selection switch. CS: CB selection switch. CT: control transfer switch. DS: display switch.

6.1.4.9.5 Terminals

Terminals are listed on page 20 (21-22): CST: console terminals. DCT: DC terminals. GT: gate terminals. PKT: pocket terminals. PMT: punch magnet terminals. RMT: read clutch magnet terminals

6.1.4.9.6 Other

CL: card lever - a switch triggered by the presence of a card. (Also sometimes clutch.)

RD: resistor/diode, listed on wiring diagram page 31 (33-34).

Thick lines on the schematic indicate wires modified as part of an Engineering Change. Sections of the schematic are indicated as "Section 2A"; these correspond to the numbers at the top of each page and the letters along the left, dividing each page into quadrants.

6.1.4.10 References

- Customer Engineering Service Index. IBM 1402 Reader Punch (Field Engineering manual).
- Parts Catalog. 1402 Card Read Punch.
- Customer Engineering Reference Manual. 1402 Card Read-Punch.
- Wiring Diagram. 1402 Card Read Punch.

6.1.5 Options and More Options

6.1.5.1 Punch Feed Read

With the PFR feature, there is a second set of brushes in the punch unit, but not an additional set of 80 wires to the 1401, so you might wonder how this works. The #2 read and the punch read share the same set of wires: a set of fourteen 6-position PFR relays switch the 80 wires between the two sets of brushes as needed.

[Note that there are two interwoven features here, PFR and SRF:

- SRF lets instruction execution take place after a read or punch is started.
- PFR lets a card be read before being punched.

Needs a bit more disentangling…

-------- Forwarded Message --------

Subject: Re: 1402 description

Date: Wed, 15 Jul 2015 00:25:05 -0700

From: Ken Shirriff <ken.shirriff@gmail.com>

To: Ed Thelen <ed@ed-thelen.org>

I found some information on this in the 1402 manual: http://bitsavers.trailing-edge.com/pdf/ibm/140x/A24-3072-2_1402_rdrPunch.pdf

Read Punch Release

The Read Punch Release special feature is available for those IBM data processing systems using the 1402 Models 1, 3,4, 5, and 6. With this feature, card movement in the read feed can be initiated by either the normal read instruction or the start read feed (SRF) instruction. When the SRF instruction is used, 21 milliseconds of read start time is made available to the processing unit.

Card movement in the punch feed can be initiated by either the normal punch instruction or the SRF instruction. Use of the SRF instruction makes 37 milliseconds of punch start time available to the processing unit. The programmer should note that failure to give the read instruction within 21 milliseconds after an SRF command will result in a reader check. Similarly, failure to issue the punch instruction within 37 milliseconds after an SRF command will cause a punch check.

From the instruction timing document: https://ia601608.us.archive.org/20/items/bitsavers_ibm140xA24ctionandTimingSummary_941849/A24-6447-0_1401_1460_Instruction_and_Timing_Summary.pdf

SRF (start read feed) is opcode 8 and SPF (start punch feed) is opcode 9.

Ken

We note that PFR complicates the read-check after punch. The assumption is that columns to receive punches have not already been punched, i.e, the programmer would read a part number in one field of the card and punch a quantity (for example) in another field of the same card.

That reminds me that I never asked what it is that finishes checking that the brushes after the punch station compare correctly with what was supposed to be punched...

6.1.5.2 Read Punch Release

[not present in CHM machines]

6.1.6 The Dynamic Timer and the Mysterious Vacuum Tube

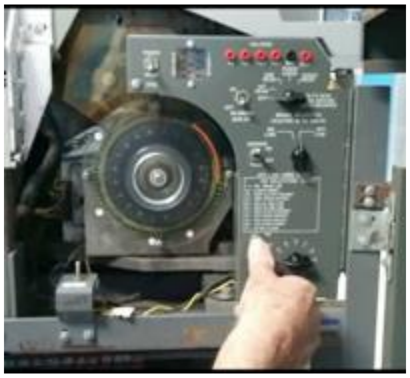


Figure 43: Dynamic Timer in Operation

T. J. Watson is said to have proclaimed "Solid State in '58"[47], but that didn't percolate *all* the way down through the organization…

Cards go whizzing through the 1402 much faster than can be seen and debugged by the unaided eye, so the 1402 Reader and Punch both include a special diagnostic mechanism called a Dynamic Timer to offer a visual representation of critical timing angles, allowing customer engineers to adjust the many cams and card levers properly.

Each dynamic timer consists of a rotating disk synchronized to the machine cycle, so the disk spins 360 degrees for each machine cycle. (i.e. the read timer spins at 800 RPM and the punch timer spins at 250 RPM.) On the disk are two neon lights that spin with the disk and are triggered by a selected circuit. The lights will illuminate an arc of the circle corresponding to the time its input is active. This visually indicates the angles during which the signal is active.

The dynamic timers are controlled by the CE Service Panel, which allows selection of the desired cam, magnet, or brush to be displayed. Use of this panel is described in Section 4 "Service Aids" of the 1402 Customer Engineering Reference Manual [CE-Ref1402] [It is?? I'm not seeing it yet. Wrong xref?]

The dynamic timer circuit uses two vacuum tubes, one to develop a high voltage power supply, the second to switch the neon bulbs on and off at the right instants. (See the 1402 Wiring Diagram [Wiring1402] pdf page 16 for the circuit diagram.)

The dynamic timer was carried forward from an IBM 088 (and 188?) Collator. [Got an xref?]

```
Subject:Re: 1402 alignment/test circuit
   Date:Thu, 2 Jul 2015 13:04:29 -0700
   From:William Flora <billflora@gmail.com>
     To:Guy Fedorkow <guy.fedorkow@gmail.com>
     CC:Ed Thelen <ed@ed-thelen.org>, Ken Shirriff <ken.shirriff@gmail.com>, Robert
        Garner <robgarn@mac.com>, Claunch, Carl <Carl.Claunch@gartner.com>,
```

[47] See http://ibm-1401.info/1401Origins.html#in58. It was Wallace McDowell who issued the policy in 1957, but the memo didn't contain the catchy phrase.

```
        stpaddock@sbcglobal.net <stpaddock@sbcglobal.net>, BillFlora@comcast.net
        <BillFlora@comcast.net>
There are 2 vacuum tubes in the 1402 for driving the dynamic timers.  Both
reader and punch use the same 2 tubes. You will find these tubes in all 1402's.
The same type of dynamic timer exists in the 088 machines. I will be happy to
show you where these tubes are, but you can't see them very well without
removing stuff that is in the way.

Bill Flora
```

6.2 1403 Printer[48]

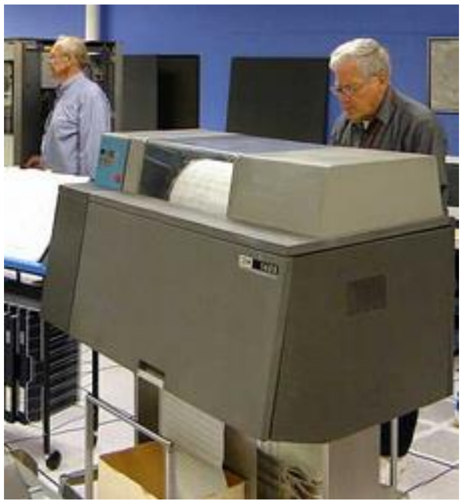


Figure 44: IBM 1403 printer at CHM

No computing job is complete until you can print the results, and for that, the IBM1401 shipped with its own high-speed printer, the 1403. Although launched as part of the original IBM 1401 family in October 1959, the 1403 went on to enjoy a long life of its own, setting the standard for print quality until development of the laser printer in the 1970's[footnote?]. The 1403 prospered well into the IBM S/360 era with development of a special purpose control unit (the IBM 2821) to connect the 1403 to the more-modern I/O infrastructure of the S/360 machines.

Prior to the 1401/1403 combination, the printing function was integrated with punch-card-controlled accounting machines, where there was little value in printing faster than cards could be read. But with the advent of electronic computing, driven by tape drives and ultimately disks, much faster printing became a valuable feature.[49]

To meet that need, the 1403 could print over 600 lines (i.e., 10 pages) per minute, with faster rates for specialized character sets and print jobs.

6.2.1 Printing Technology

The 1403 is a *Chain Printer*[50], which is to say that there are a number of mechanical elements that must operate in unison to form printed images of characters on paper:

- The critical element carrying images of characters is a *print chain*, a sequence of type slugs, each of which carries the image of a character, connected together like links in a bicycle chain. The print chain whizzes horizontally past paper that is oriented for vertical movement.
- Ink is applied to paper with a very large ribbon that's the width of the paper (14"), and 100 feet long, stored on a roll.
- Behind the paper, there is a hammer for each of the 132 print columns provided by most 1403 models.[51]

[48] This section originally contributed by Ken Shiriff, edited by guy

[49] For more on printing technology leading up to the 1403, see https://en.wikipedia.org/wiki/IBM_1403.

[50] The 1403 models 3 and N1 used what IBM called a print *train*, rather than a print *chain*… almost the same, but type slugs fit into a track as part of the removable print assembly. [I assume the CHM's machines use print Chains, not Trains… anyone know why two models in the middle of the sequence went with the Train mechanism?]

[51] Not every model printed 132 columns per line of type, but that number became a de-facto standard. [xref?]

To print a character, a print hammer in the desired column is fired at just the moment that that the desired character on the chain is aligned with the column, pressing the paper against the ribbon, with the image of the character behind it on the chain.

Figure 45 shows the 1401 with its cover open, and the print-chain and ribbon mechanism swung aside, as would be done to feed new paper.

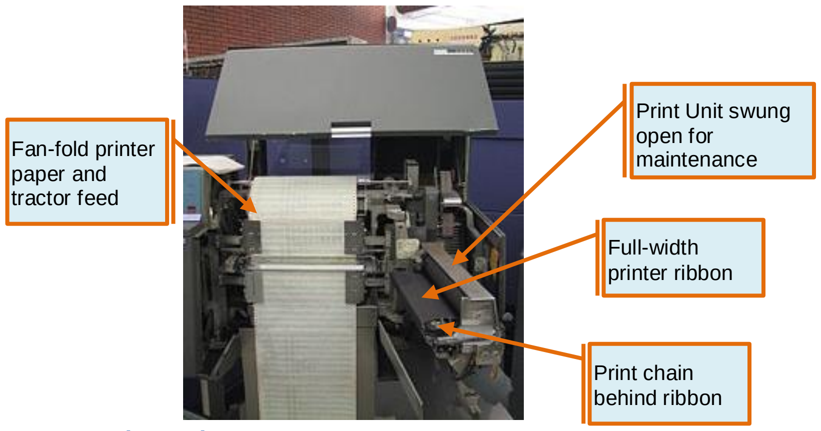

Fan-fold printer paper and tractor feed
Print Unit swung open for maintenance
Full-width printer ribbon
Print chain behind ribbon

Figure 45: Printer Elements

Figure 46 shows a schematic view of the key elements of a chain printer. In the 1403, the 132 print hammers are located behind the paper, one for each column of type. With the print unit closed and ready for operation, the ribbon is in front of the paper, and then the chain containing type slugs is in front of the ribbon. A print hammer activated by a magnetic coil presses the paper and ribbon against a type slug on the chain to make an impression.

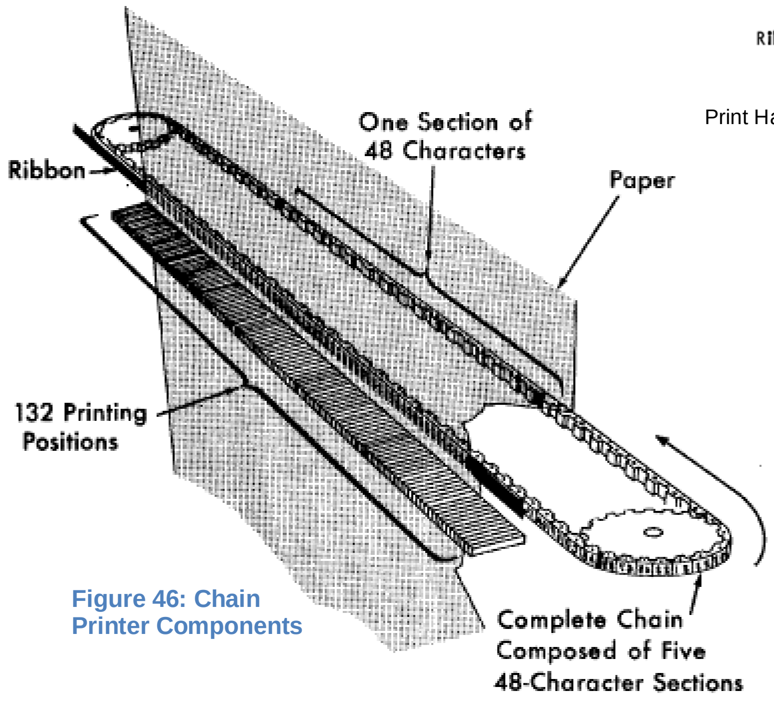

One Section of 48 Characters
Ribbon
Paper
132 Printing Positions
Complete Chain Composed of Five 48-Character Sections

Figure 46: Chain Printer Components

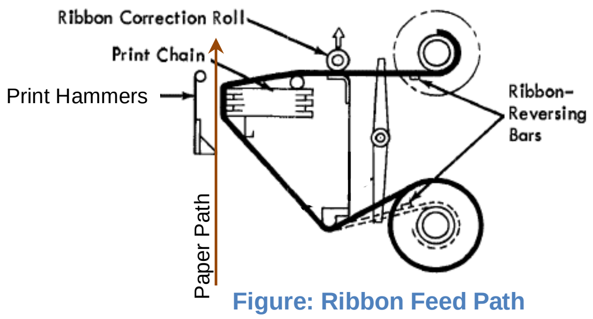


Figure: Ribbon Feed Path

All of these print hammers going off can get rather noisy… with the cover closed, at about 2 feet in front is about 85 dbA. With cover open, noise levels rise to about 94 dbA. (for details).

6.2.1.1 Printer Paper

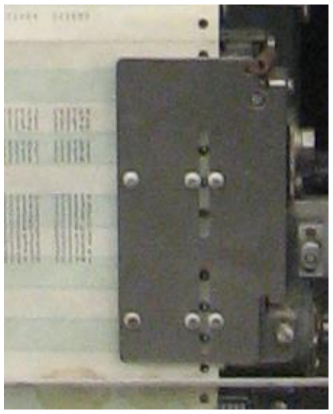


Figure 47: 1403 Tractor Feed

The 1403 printer prints on *fan-fold* continuous forms paper, which is to say that an entire box of paper is a single continuous sheet, perforated to allow separation of individual sheets. Conventional fan-fold forms include a row of holes along the right and left edges to engage pins in the *tractor feed* mechanism, which keeps the paper aligned.

Many kinds of specialized forms could be used in the 1403, ranging from multi-part carbon-paper forms to make duplicate copies in one pass through the printer, to specialized forms for printing checks or envelopes, all designed to fit the continuous-feed mechanism.

The 1403 printer includes a carriage control tape reader, for reading a replaceable paper tape that can be punched to indicate the top-of-form or specific areas of the form to skip over, allowing a wide range of sizes and shapes of forms to be printed.

6.2.2 Printer Numerology

The secret to the printer's performance is a chain with character that spins quickly enough to complete a full rotation of the chain about three times per second. The printer has 132 hammers, one for each print position. When a character on the chain passes a spot where it should be printed, the hammer fires, printing the character. This process is more complex than it sounds, and is described in detail below.

Here are the key numbers for understanding the translation from a line of print stored in the 1401 memory to a sequence of print hammer firings.

- There are 132 print columns
- There are 48 printable characters in a character set (See Section 6.2.3.1)
- Print hammers are spaced exactly 0.1" apart (i.e., the printable part of the page is 13.2" wide)
- Type slugs are spaced at .1505" apart, yielding approximately two type slugs for every three print columns
- There's an opportunity to fire a hammer ever 11.1 microseconds

Of these, that tiny bit of extra width between type slugs (0.1505" vs an exact 3:2 ratio at 0.15" spacing) is critical… The print chain spacing is very slightly wider than the hammer spacing, so as the print chain moves a very small amount, successive hammers and characters are aligned. (This is similar to the principle of a vernier scale or moiré pattern.)

Given the approximate 3:2 ratio, every second type slug almost lines up with every third hammer, with a discrepancy of 0.001". But the chain moves left 0.001" every 11.1 microseconds, with the result that every 11.1 microsecond, another type slug and print hammer will be perfectly aligned, offering the possibility of printing a character. After 132 * 11.1 = 1,465.2 usec, an interval called a *scan*, there will have been a chance to print one character in every column.

So we know that during a scan, a type slug will have aligned perfectly with each one of the 132 print hammers, but we still don't know when to actually fire the hammer. For that, the designers use a lot of read cycles from memory.

Every 11.1 usec, when a type slug and a print column are perfectly aligned, the machine reads the core memory location corresponding to that column to find out what should be printed, and compares that to the character that's on the type slug that's aligned with the column. If they match, the hammer is fired, and the character printed. If not, try again next time, when a different type slug will be in place. There are only 48 possible printable characters, so after 48 scans, every column will have had a chance to print every character, and the line will be complete.

There's special logic to compute both the next column and the next character.

Because of the almost-3:2 ratio, a type slug will align next with every *third* print column. The string of characters to be printed is stored in memory starting at location 201, so the print operation starts reading location 201 and then continues reading every third location in the range 201 to 332, until all 132 locations have been read. There's a special "increment the memory address by three" operation for this purpose in the memory controller. [xref ILD ibm pg 19, pdf pg 18]

There's a separate hardware counter (ILD ibm pg 70, pdf pg 61) to keep track of which character is aligned at a given moment. The counter is reset by a sync pulse from the printer when the chain passes its home position (got a name for that signal? What kind of sensor is it?), and then counts every 11.1 usec as the chain moves (what's the count sequence?)

The remaining logic is simpler in concept; the character that should be printed is read from the memory location corresponding to the print column and stored in the B Register. From there, it's compared to the counter that indicates which type slug is positioned in front of the column. If they match, the hammer is fired. (See Section 6.2.4.4 for details on this mechanism.)

6.2.3 Programming the Printer

6.2.3.1 Character Set

Prior to ASCII, laser printers and Unicode, multiple character sets were accommodated by changing keycaps on keypunch machines and by installing different printer chains.

Many different IBM 1403 chains are available; the standard ones are the 'A' alphameric chain and the 'H' Hollerith chain. The CHM printer has the A chain character set (but with + in place of &):

```
1234567890#@/STUVWXYZ‡,%JKLMNOPQR-$*ABCDEFGHI&.
1234567890#@/STUVWXYZ‡,%JKLMNOPQR-$*ABCDEFGHI+.
```

The H chain has a few different special symbols:

```
1234567890='/STUVWXYZ‡,(JKLMNOPQR-$*ABCDEFGHI+.)
```

For more information on different chains, see 1403 Fonts and Chains.

6.2.3.2 Carriage Control

The printer also provides low-speed and high speed carriage control (i.e. paper feeding). While an ordinary print operation advances the paper by one line at a time, there's a secondary mechanism, controlled by a carriage tape in the printer, that advances the paper through the printer much more quickly to skip over areas that will remain blank.

The carriage tape is constructed by the programmer to match the size of the expected form, with punches in the tape to indicate specific points.

To cause the printer to skip to a specific place on the form (e.g. skip to the top of the next page, or skip to the place on the form where an address should be printed), the programmer would execute a *Control Carriage* instruction, specifying which channel on the carriage control tape should contain a punch to indicate the location. (RefMan ibm pg 52)

Channel One on the tape would normally have a single punch to indicate Top of Form.

Models of the 1403 with dual-speed carriage control can skip at 33 or 75 inches-per-second (i.e., about 7 pages per second), so disaster can ensue if there's no Top of Form punch in the carriage control tape…

Is there anything that should be said about the mechanical aspect of the paper feed path?

6.2.3.3 1401 Instructions for Printing

Operation of the printer is controlled directly by the programmer. A line to be printed is assembled in memory locations 201-332, and when it's ready, a *Write a Line* (ibm pg 47) instruction can be executed. That starts the print operation, prints all the characters, and advances to the next line. Unless there's a Print Storage option installed (see section xx), instruction execution stalls while the machine prints.

Program execution resumes after the line has been printed, approximately 84 msec later for the standard print chain. To achieve 600 lines per minute, or one line per msec, the programmer must assemble the next print line in the remaining 16 msec, while carriage control completes moving the paper into position for the next line (ibm pg 46)

There are a number of instructions to combine printing with reading or punching cards, offering some small efficiency gain in partially overlapping operations.

There are also instructions to test status, as well as carriage control instructions. For details, see the 1401 Reference Manual (ibm pg 47 and on).

- W Write (also write word marks and branch)
- WP Write and punch
- WR Write and read
- WRF Write and read punch feed
- WRP Write read and punch
- BC9 Branch on carriage channel 9
- BCV Branch on carriage overflow
- BPCB Branch printer carriage busy (with print storage option)
- BPB Branch printer busy (with print storage option)
- BIN Branch indicator (including printer error)
- CC Carriage control
- CCB Carriage control and branch

With the Print Storage option, most instructions for operating the printer remain the same, but program execution is only stalled for 2 msec instead of 84 msec when a *Write a Line* instruction is issued. See Section 6.2.4.3 for more on this feature.

6.2.4 Implementation

This section contains further notes on the implementation of the 1403.

6.2.4.1 Logic

One interesting thing about the 1403 printer is that all the logic is in the 1401 computer. The printer is electro-mechanical (with a hydraulic system for paper feed). The 1403 printer is connected to the 1401 by two 160-pin signal cables and a 13-pin power cable. The 132 hammers in the printer are connected by 132 wires to the 1401, which contains the printer drivers. Another 66 wires provide -60V return for each pair of hammers.

6.2.4.2 Error Checking

As with the 1402 card reader, the printer logic includes extensive error checking, using additional core planes: hammer fire area, print compare area, print-line complete, and print-error storage.

One validity check is "hammer fire - print compare". A hammer fire circuit tests for a hammer that fails to fire when it is energized or a hammer that fires without being energized. Each hammer driver is wired directly to a hammer-fire core, which records when a hammer fires. For each print position, the print-compare core is when a hammer is supposed to fire, that is when the character in memory matches the character on the chain. As each storage location is accessed in a print scan, the hammer-fire core and print-compare core from the previous print scan are checked for a mismatch (character matched but hammer didn't fire or character didn't match but hammer fired).

Another check is "print-line complete", which checks that all printable characters are printed. These cores are cleared at the beginning of a print operation. A core is set if the corresponding storage does not have a printable character, or if there is an opportunity to print the character. If the core is not set at the end of the print operation, there was a printable character but no opportunity to print it. An error is also triggered if the core is already set when a print-compare match is found, indicating an attempt to print a character twice.

If there is an error, the corresponding core in the print-error storage plane is set to record which position had the error. The storage scan feature will scan storage for errors. The scan will stop on the location of any print errors, indicating the character position of the fault. (The mode switch is set to STORAGE SCAN for this feature.) See 1401 Reference Manual for details.

Error checking uses positions 201 through 332 in four core planes. The hammer drivers are connected to plane 11 (terminals) for the cores in plane 12 (RD1-PRT); each hammer driver is connected directly to a specific core (without addressing). The print compare cores are in plane 13 (XU11). The print-error storage cores are in plane 14 (YU12). The print line complete cores are in plane 15 (YL13).

Other error checking includes parity checks and synchronization checks between the chain and the 1401.

6.2.4.3 Print Storage Buffer

On the standard 1401, processing is blocked while the printer is in operation, as almost every available memory cycle is used during the 86 msec needed to scan a complete line. The optional print storage feature provides much more processing time by quickly copying the print line from main core to a special purpose small core memory used as a print buffer, allowing the 1401 to resume processing while the printer reads the characters from the print buffer.

Printing a line takes 100 milliseconds. In a standard 1401, the processor is blocked for 84 milliseconds of this time, leaving just 16 milliseconds of processing. With print storage, the processor is blocked for just two milliseconds to copy the characters to the buffer, leaving 98 milliseconds of processing time. (1401 Reference Manual, page 82.)

Print storage uses its own core module, which is smaller than the main core module. It consists of 12 planes in a 10x14 grid. In addition to 8 data planes, the module has a hammer fire plane, a print line compare plane, an equal check plane, and a print error check plane. The hammer fire plane is written directly by the hammer drivers; each hammer response line is wired individually to five turns around the associated core.

The print storage feature filled gate 01A4 (which held the core), along with part of 01A5. The print storage feature cost an extra $375 a month, more than multiply-divide (which was $325 a month).

The print storage feature includes two new branch codes: branch on printer busy and branch on carriage busy, which can be used to avoid accessing the printer while it is in operation (which would block processing). A print storage scan feature is also provided to scan the print storage for errors. This is controlled by setting the auxiliary mode switch to "Print Storage Scan" setting on the auxiliary console and then performing a storage scan.

See [1401 Data Processing System Optional Feature]() (page 117) for details on how print storage operates.

6.2.4.4 More Gruesome Details: How the Hammers Interact with the Print Chain

The key to understanding the printer is understanding the relationship between the moving print chain and the fixed hammers. If the print chain needed to move a full character position to line up with the next hammer, printing would be slow.

As noted in Section XX, the print chain and print hammers are carefully spaced so that one hammer aligns with a type slug at a time, offering a sequence of opportunities to print. This section goes through the sequencing of hammers in a bit more detail.

As the print chain moves, the alignment shifts from one to another hammer and a different chain element. One *print scan* is completed after all 132 hammers have been aligned with the chain and had an option to print a character. Each of the 132 hammers must have the option to print each of the 48 possible characters, requiring 48 print scans for each line.

The chain to hammer spacing is just slightly larger than a 3:2 ratio, so the alignment pattern is a bit tricky. The alignment pattern starts with hammer 1 with chain element 1, then hammer 4 with chain element 3, then hammer 7 with chain element 5, hammer 7 with chain element 10, and so forth. Finally, hammer 130 is aligned with chain element 87. At the end of this process, every third hammer has aligned with every second chain element. 44 different chain elements have lined up with 44 different hammers, giving 44 opportunities to print a character. This is called one *print subscan*.

The second subscan starts right after the first, when hammer 2 aligns with chain element 2. This is followed by the alignment of hammer 5 with chain element 4, then hammer 8 with chain element 6, and so forth. At the end of this subscan, 44 more hammers have had an option to print.

The third subscan starts right after the second, when hammer 3 aligns with chain element 3. The subscan continues with hammer 6 aligning with chain element 5, and so forth, until 44 hammers have had an option to print.

After the third subscan, all 132 hammers have had an option to print a character, completing a print scan. As explained above, 48 print scans are required for a line, so each hammer has an option to print each of the 48 different characters on the chain.

To see exactly how this alignment process works requires looking at the dimensions of the hammers and chain elements. The hammers have a 0.100 inch spacing (all measurements in inches), while the chain elements have a .1505 spacing. Starting from 0, hammers will be at positions 0, .1, .2, .3, .4, .5, .6, etc. The chain elements will be at positions 0, .1505, .301, .4515, .602, etc. Note that hammer 1 is aligned with element 1. If the chain shifts to the left 0.001, hammer 4 (.3) will be aligned with type 3. After another shift of .001, hammer 7 (.6) will be aligned with element 5. As the chain moves, every third hammer will be aligned with every second chain element until finally hammer 130 is aligned with chain element 87.

At the completion of the subscan, the chain has shifted by 0.043, which is less than half of a character width. At this point, chain element 2 is almost lined up with hammer 2. After an additional shift of .0075, element 2 is synchronized with hammer 2. As the chain moves, a second subscan takes place with element 2 aligned with hammer 2, element 4 with hammer 5, element 6 with hammer 8, and so forth until hammer 131 is aligned with chain element 88. This completes the second subscan. As before there were 44 opportunities to print a character.

After the second subscan, hammer 3 is almost aligned with chain element 3. A small additional shift of 0.0075 aligns them, and the third subscan takes place. Hammer 3 aligns with chain element 3, then hammer 6 with chain element 5, hammer 9 with chain element 7, and so forth until hammer 132 is aligned with chain element 89.

There’s an animation of print chain alignment at:

http://ibm-1401.info/KenShirriff-1403Animation.html

6.2.4.5 More Gruesome Details: Print Timing

This section summarizes the timing of the printer. The chain moves at 90.3 inches per second, so each movement of .001 takes 11.1 microseconds. Thus, a hammer is aligned every 11.1 microseconds. For the 44 hammers in one subscan to align takes takes 484 microseconds. It takes an additional 47 microseconds to align the chain for the next subscan, yielding a total time of 555 microseconds per subscan and 1665 microseconds per scan. Multiplying by 48 characters yields about 80 milliseconds per line. Adding about 20 milliseconds to advance the paper, the printer can print 600 lines per minute. (These numbers are from the IBM 1403 Service Manual, which explains the printing process in detail. The numbers don't exactly add up due to rounding. Unfortunately, the Service Manual is not available online.)

The printer can also be configured with a numerical chain with 16 characters. Only 16 scans are required per line, rather than 48, resulting in 26.7 microseconds print time per line, which yields 1285 lines per minutes (after adding 20 milliseconds to advance the paper). The "preferred character set" feature arranges characters on the chain so the most common characters are presented more often, and can get up to 1400 lines per minute. (See IBM 1403 Printer Component Description.)

(We note that with the Print Storage option, the special core memory used as the print buffer has exactly 11.1 usec cycle time (how do we know?). But without the Print Storage option, memory cycles are executed by the 1401's main memory system with an 11.5 usec cycle time. The timing discrepancy must cause a slight horizontal shift in character spacing)

6.2.5 To Learn More

For more information on the printer, see IBM 1403 Printer Component Description.

The print instruction, print scan process, timing and circuitry are described in detail in IBM 1401 Instruction Logic (pages 88-94). The print instruction is described step by step in 1401 Data Flow (pages 59-63).

6.3 729 Tape Drive[52]

The IBM 727 Tape Drive was first introduced in (yearXX?) for use with the IBM 709 vacuum-tube computer. This drive was replaced by the IBM 729 Model I, also implemented in vacuum-tube technology.

The IBM 729 Model II and beyond switched to transistor logic, and were used with all models of the IBM 7000 series transistorized computers.

The IBM 1401 with the 729 tape drive option was introduced on October 5, 1959 (footnote?)

From Iggy: Here's part of the Official 1401 Announcement:

> *The all-transistorized IBM 1401 Data Processing System places the features found in electronic data processing systems at the disposal of smaller businesses, previously limited to the use of conventional punched card equipment. These features include: high speed card punching and reading, magnetic tape input and output, high speed printing, stored program, and arithmetic and logical ability*

https://en.wikipedia.org/wiki/IBM_727

[52] With considerable help from Ignacio Menendez, Carl Claunch and Bob Feretich, Dec 2016

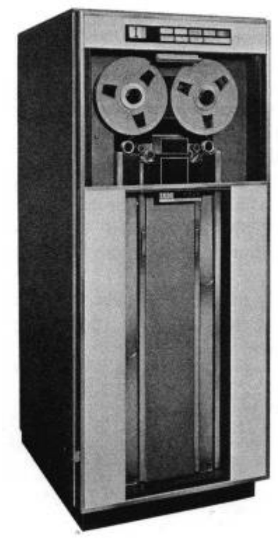
https://en.wikipedia.org/wiki/IBM_729

http://ibm-1401.info/729-Info.html
http://bitsavers.trailing-edge.com/pdf/ibm/magtape/729/223-6845_729_CEman_1959.pdf
http://bitsavers.trailing-edge.com/pdf/ibm/magtape/729/223-6988_729_CE_Mar62.pdf
http://ibm-1401.info/729-Info.html (Ed Thelen's *An IBM 729 Tape Unit Refresher*)

When IBM introduced the tape drive on September 25, 1953, as part of the IBM 700 series (right? footnote?) it gave its customers the first high-density storage mechanism. By the time of the IBM1401, a full-sized 2400-foot reel of 556 bits-per-inch tape would hold between 2 and 12 million characters, equivalent to 30K to 150K punched cards[53], depending on the "blocking factor", offering about 400 times more data capacity per unit weight than conventional punched cards.[54] Tape drives quickly became the de-facto standard for mass storage and archival purposes, allowing large amounts of data to be moved between machines, or to be kept in storage for later retrieval.

Tape drives also provided much faster I/O than cards. At 600 cards per minute, a 1402 could transfer data into a 1401 at 800 characters per second[55]. An IBM 729 Model 2 drive at 556 bits per inch density could achieve 41,700 characters per second.[56]

This high-speed I/O opened a new market for the 1401 as an I/O manager for the 7000 series machines. For 1401 users, high-speed tape also offered substantial acceleration for algorithms like sorting, where all the data typically could not fit into core memory.[57]



The IBM 729 tape drive had a long life, and survived through quite a few revisions and enhancements. Table 4 shows three of the common[?] models used with the IBM 1401, offering increasing speed and density as the technology evolved.

[53] Depending on the "blocking factor"; See section 6.3.1

[54] A single two-pound magnetic tape could hold 12 million characters of data, while the same information on punched cards might take up 160,000 cards, requiring 80 boxes and weighing in at 800 pounds.

[55] 600 cards per minute _> 10 cards per second. 80 characters per card -> 800 char per sec.

[56] 556 char per inch, 75 ips -> 41,700 char per sec.

[57] Remember, no disks until later. Sorting tasks using only the card reader and punch on the 1401 was so slow that IBM 086 electro-mechanical sorters continued to be used for some time.

	729-II	729-IV	729-V	729-VI
Tape Speed (inches per sec)	75	112.5	75	112.5
Record Density (bits per inch per track)	200 – Low 556 – High	200 – Low 556 – High	200, 556, or 800	
usec per character	67 – Low 24 – High	44 – Low 16 – High		
Character Rate (char per sec	15,000 – Low 41,667 – High	22,500 – Low 62,500 – High		
Tape Capacity @ 1600 byte block size				

Table 4: IBM 729 Tape Drive Models

- The 1401 could also be used with lower-performance IBM 7330 tape drives (although CHM doesn't have any)
- The CHM drives are Mod II, Mod IV and V.
- [Mod I was tube-based and only worked with the 709 (we think)]

The IBM 1401 can be equipped with an optional tape controller, contained in the 1401 frame (called a Tape Adapter Unit, or TAU), connecting to as many as six IBM 729 Model II, IV, V or VI, or 7330 tape drives. The TAU design added to the 1401 relatively late in the design process, and was leveraged from the IBM 7000 series computers, but re-used in the 1401 as-is [got a footnote for this assertion? Can we add a note to the main doc about the relationship between 7000 series and 1401? Is there a simple story for how the two were used together? Is it simply "write your print job to tape on the 7000 machine and print it with the cheaper 1401 machine?]

6.3.1 Tape Drive Basics

Magnetic Tape was the first form of mass storage introduced to hold data sets that were too large to conveniently handle with punched cards. Each 2400-foot tape could hold the data from 30,000 cards, to more than 150,000 cards depending on conditions.

Like conventional audio and video tape, information on a 729 mag tape is recorded as magnetic flux transitions impressed onto an iron-oxide material bound to a plastic film. Unlike audio or video tape, information is not recorded continuously, but rather is broken into blocks of data, a few hundred to a few thousand characters in length, each of which must be read or written block at a time without interruption. Blocks of data on tape are separated by Inter Record Gaps, also known as inter-block gaps in some documentation.

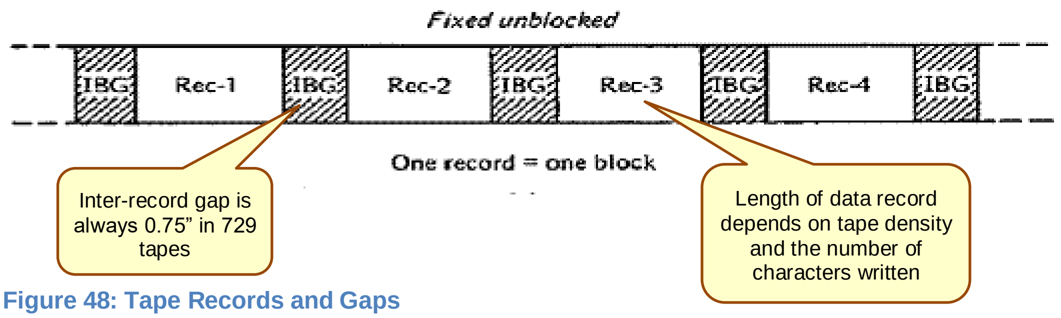

Fixed unblocked
IBG
Rec-1
IBG
Rec-2
IBG
Rec-3
IBG
Rec-4
IBG
One record = one block
Inter-record gap is always 0.75" in 729 tapes
Length of data record depends on tape density and the number of characters written

Figure 48: Tape Records and Gaps

Once started, a tape record must be read from beginning to end at a constant data rate. At the end of the record, the tape motion is stopped while the processor prepares for the next record, at which point the tape started up again to move the next block past the read/write heads, all within the space allocated to the IRG (0.75" for IBM 729 drives)[58]. This requirement for lightning start-stop of tape movement results in considerable complexity in the mechanical tape transport system, and is a principal factor in the design of the drive. (Section6.3.1.1.1).

Each record comprises a sequence of equally-spaced characters written to the tape with a seven-track *read/write head*. The record is followed by a Longitudinal Redundancy Check character at the end of the block. For 729 drives, each character is seven bits wide, as in the 1401, with a six-bit character plus a check bit (called a Vertical Redundancy Check (VRC) bit). Longitudinal parity is accumulated as characters as the record is processed, so it can be checked at the end of a read-record operation.

In the 1401, tapes can be written with either even or odd parity (depending on restrictions below) as determined by control bits in the 1401 read and write instructions.

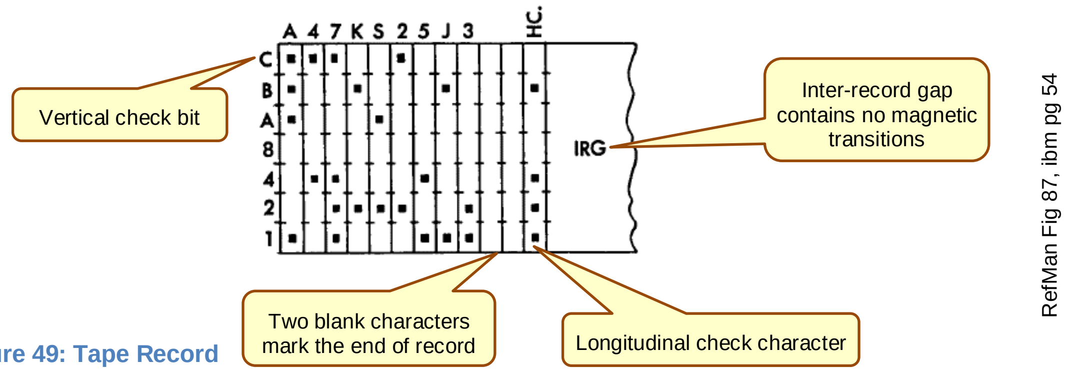

A 4 7 K S 2 5 J 3
HC.
C
B
A
8
4
2
1
IRG
Vertical check bit
Inter-record gap contains no magnetic transitions
Two blank characters mark the end of record
Longitudinal check character
RefMan Fig 87, ibm pg 54

Figure 49: Tape Record

The end of each tape record is indicated by a gap of two character spaces with no transitions, followed by the Longitudinal Check character. Every character within the record must have at least one "one" (see notes on character set below Section 6.3.1.2)

Tape Capacity

A full-sized 2400-foot reel of 556 bits-per-inch tape will hold between 2 and 12 million characters, equivalent to 30K to 150K punched cards, depending on the "blocking factor", i.e., the number of 80-character cards combined into each tape block.

Small tape blocks are inefficient, due to the fixed 0.75" inter-record gap. Blocks that are too large are hard to use because of the amount of core memory consumed during reads and writes, as well as the risk of an error rendering the whole block unusable.

A block of 1600 characters, corresponding to 20 card images, yields about 80% efficiency on a 556 BPI tape.

Detailed calculation can be seen at http://ibm-1401.info/TapeBlocking.html

6.3.1.1 Mechanical

Tape drives include considerable mechanical complexity, most of which is there to manage the simple problem of quick start and stop times, which minimize the amount of tape wasted on inter-record gaps.

[58] Manual control from the TAU can move tape continuously, but it is not possible to write a program that doesn't stop between records

To achieve the quick response times, the 729 drive decouples motion of the tape itself across read-write heads from motion of the reels, hubs and the bulk of the 2400-foot tape. This is accomplished by keeping a loop of tape in each of the two Vacuum Columns, as shown in Figure 3.
The vacuum column is 42" long, 2.5" wide and 0.51" deep, just deep enough to accommodate the width of a standard tape without binding. Air is evacuated from the bottom of the columns by a blower, pulling the tape into the column and forming a loop. When the capstan is engaged to start tape motion, tape is pulled out of one column and played into the other, yielding very low inertia, for fast start and stop.
A vacuum-driven feedback loop then activates the heavy reel hubs to fill or empty the columns as needed to keep the loop within limits.

From 223-6988_729_CE_Mar62.pdf ibm pg 36

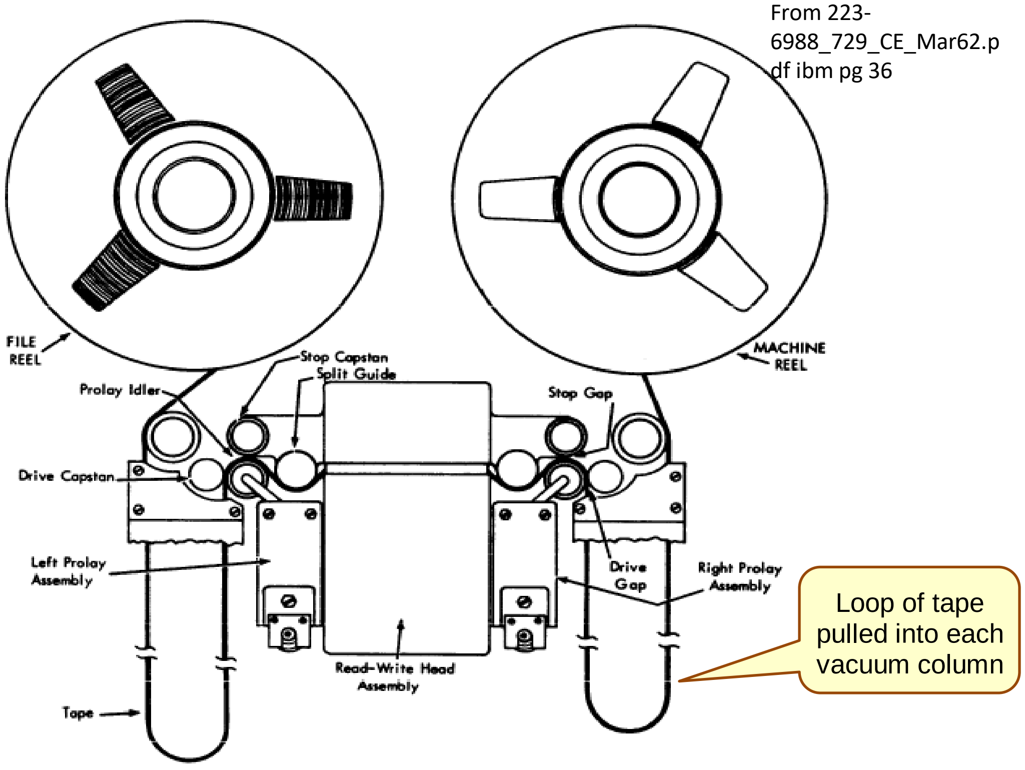


Figure 35. Path of Tape through Magnetic Tape Unit

Figure 50: Tape Path in IBM 729 Drives

6.3.1.1.1 Tape Movement

Tape is moved across the read-write heads through the use of rotating and stationary *capstans*, plus a mechanism called a *prolay*. While the 1401 cannot read and write in the reverse direction, it is important to be able to backspace over tape records, so the mechanism is arranged to allow forward and backwards operations. During a backspace operation, data can't be read to storage, but it the TAU does monitor read signals to detect the next gap between blocks.

Once the tape is loaded, the forward and reverse capstans turn continuously, so key to all of this is the magnetically-controlled prolay, a mechanism that can move an idler wheel from a neutral position to cause it to either squeeze the tape against a rotating capstan to make the tape move, or a stationary capstan to make it stop. There are capstan and prolay assemblies on each side of the head assembly, allowing movement in both directions.

Tape is always 'pulled', not pushed, so when moving left to right, the right capstan is engaged for motion, and the left capstan is engaged to stop.

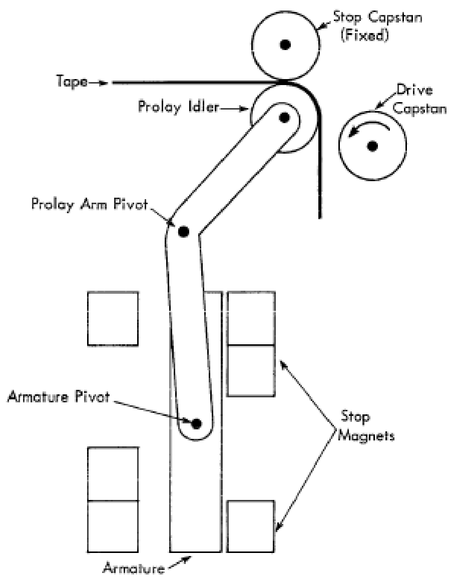


FIGURE 4.2-1. RIGHT PROLAY WITH STOP MAGNETS ENERGIZED

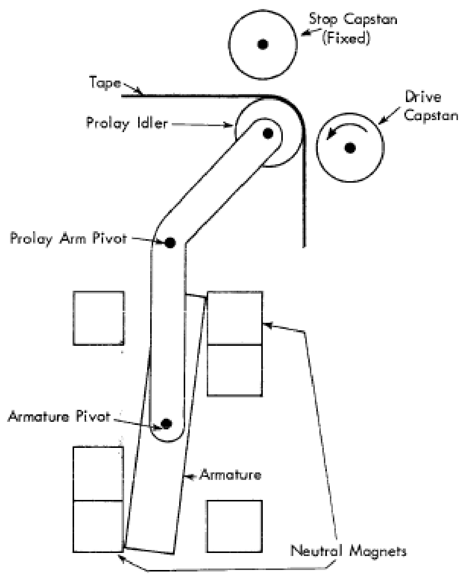


FIGURE 4.2-2. RIGHT PROLAY WITH NEUTRAL MAGNETS ENERGIZED

6.3.1.1.2 Take Up and Supply Reels

The capstans and prolays control the movement of the few feet of tape in the columns and crossing the read-write heads. But the remainder of the 2400 feet of tape are managed by a separate mechanism that operates the supply and take-up reels (called File and Machine reels in IBM parlance). In between the two is a buffer of tape held in the Vacuum Columns, by a bit of negative air pressure.

Vacuum columns are shown in Figure 4. Air is exhausted from the bottom of each column[59], pulling the tape tight. There are vacuum actuated switches at approximately 1/3 and 2/3 the length of the columns. If the tape loop is above the top vacuum switch, the appropriate reel motor is activated to put more tape into the column; if it's below the lower vacuum switch, the motor is activated to remove tape from the column.

When the motors are not engaged to put tape into or out of the columns, a brake is applied to the reel hubs.

Operation of the reel hubs and vacuum columns can be seen in this video: https://www.youtube.com/watch?v=7Lh4CMz_Z6M

[59] Think "vacuum cleaner motor", not "laboratory vacuum pump". It's said that the initial tape drives did indeed use vacuum cleaner motors, although later versions used more sophisticated vacuum pumps.

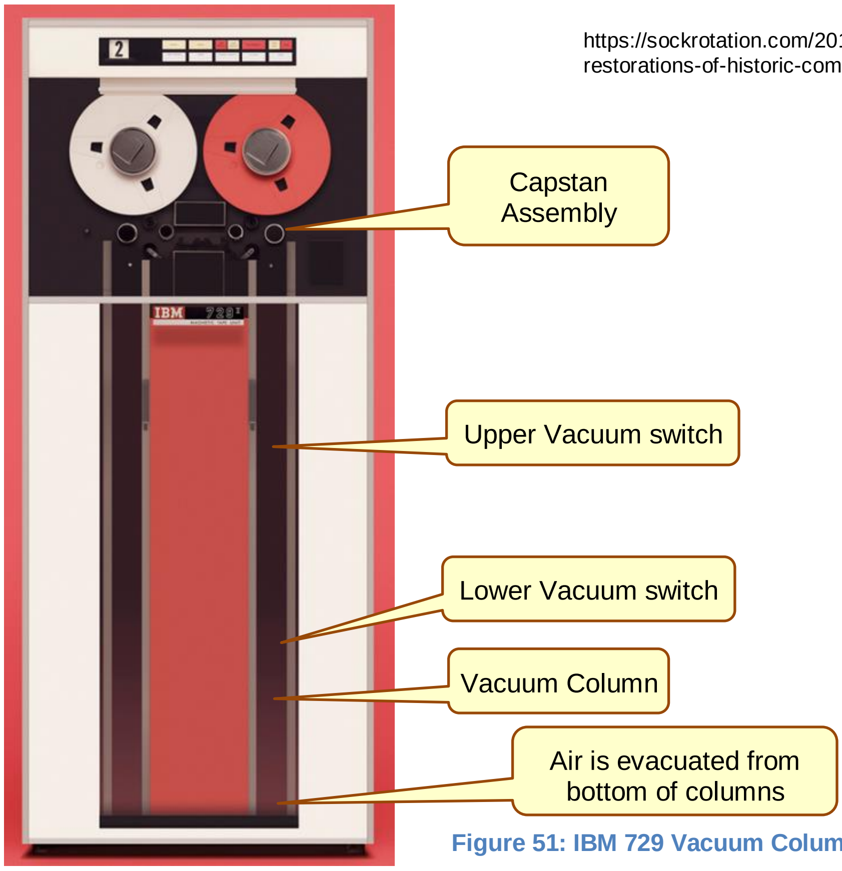

https://sockrotation.com/2016/05/13/colourful-digital-restorations-of-historic-computers/ibm_729/
Capstan Assembly
Upper Vacuum switch
Lower Vacuum switch
Vacuum Column
Air is evacuated from bottom of columns

Figure 51: IBM 729 Vacuum Columns

In fact, the motors that drive the reel hubs turn continuous, but their motion is transferred to reel hubs by two clutches, one for the forward and one for the reverse directions.

Figure 5 shows the mechanism behind the reel hubs… the tape drive chassis contains two AC motors that turn all the time, in opposite directions. These motors use rubber belts to drive magnetically-controller clutches for each direction on the reel hubs. Depending on which clutch is activated, each reel can either spill tape into its column, or wind it up, pulling tape out of the column.

Each hub has a third clutch which serves primarily as a brake, but with a secondary function in loading the tape (described in the next section).

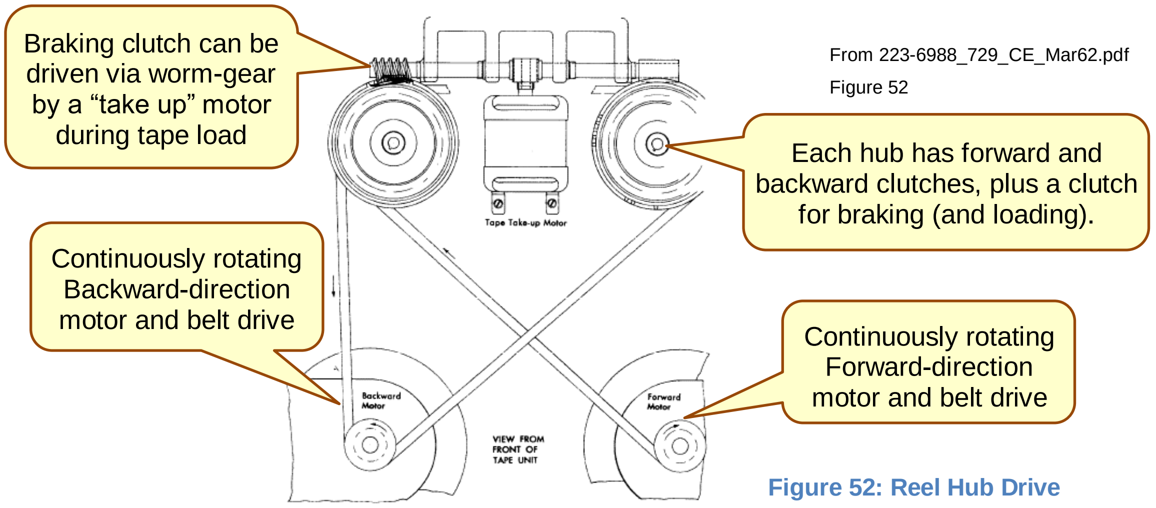

Braking clutch can be driven via worm-gear by a "take up" motor during tape load
Continuously rotating Backward-direction motor and belt drive
Tape Take-up Motor
Backward Motor
VIEW FROM FRONT OF TAPE UNIT
Forward Motor
From 223-6988_729_CE_Mar62.pdf
Figure 52
Each hub has forward and backward clutches, plus a clutch for braking (and loading).
Continuously rotating Forward-direction motor and belt drive

Figure 52: Reel Hub Drive

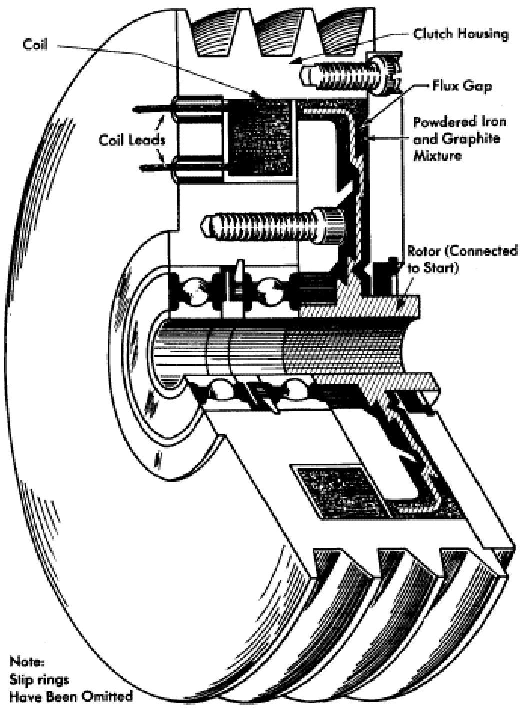

Coil
Clutch Housing
Flux Gap
Coil Leads
Powdered Iron and Graphite Mixture
Rotor (Connected to Start)
Note: Slip rings Have Been Omitted

Figure 53. Magnetic Clutch

Figure 53: Reel Hub Magnetic Clutch

The clutches use a magnetic powder material; when deactivated, the two halves of the clutch are disengaged, but as electromagnets in the clutch are activated, the powder begins to solidify, transferring motion from motor or brakes to the hubs. The clutches are not just 'on' or 'off' – modulating the current level can provide partial engagement. This capability is critical when terminating a high-speed-rewind cycle; the braking clutch is gently engaged to slow the reel hubs with snapping the tape.

Each clutch has an electromagnet that must rotate freely; power is transferred to the electromagnet via carbon brushes.[60]

[60] These brushes are subject to wear, and were all replaced in the CHM tape drives in year 2016. The CE manuals give numerous other maintenance tips.

6.3.1.1.3 Load and Unload

There are a number of mechanisms that are coordinated to simplify the task of mounting a tape and preparing it for read and write operation.
With the IBM 729, loading a tape was a semi-automated procedure.[61]

- The operator 'mounts' a reel of tape on the "File" hub (the one on the left)
- The operator manually threads tape through the open head assembly, and winds it on to the Machine Reel, the empty reel that normally stays put on the right hub.
- The operator manually winds the tape forward until is passes the "load point", a patch of reflective metallic tape sensed with a photocell, marking the official start of the useable part of the magnetic tape, usually about ten feet in.
- The operator then commands the IBM 729 to "load" the tape, and the rest happens under automatic control:
    - The vacuum pump is energized
    - The Stop Capstans are engaged to keep the tape from moving across the heads
    - The head assembly is closed by a motor drive
    - Each reel is slowly turned to release tape, which is pulled into the vacuum columns by its low air pressure.[62],[63]
    - Capstan drive motors are activated and start to spin the two capstans. The capstans are actually retracted when the motors are deactivated to get them out of the way of the load path, but starting the motors pushes the capstans forward into place.
    - Once tape is in the columns, it's moved in the reverse direction to back up to the Load Point, i.e., the patch of silver foil on the tape marking its beginning point.
- At that point, the tape is ready for use.

6.3.1.1.4 Rewind

Once a processing operation is complete, tape must be wound back from the take up reel to the supply reel. This could be done simply by backspacing to the beginning, but that's too slow for Type-A Data Center Managers.
The 729 offers a multi-stage rewind mechanism.

- To start, tape is unloaded from vacuum columns
- A special high-speed rewind motor that directly drives the supply hub's shaft is activated.
- There's an optical sensor that determines how much tape is left on the takeup hub, yet to be rewound
- Once the layers of tape on the takeup reel are less than about half an inch thick, the high speed rewind motor is stopped, tape is reloaded into the columns, and the backspace capstan is engaged to take the tape back to its load point, where it stops

- To remove the tape entirely, tape may be manually unloaded again
- The operator then manually winds the last few feet of tape onto the supply hub and removes the finished tape for storage.

Figure 7 shows the location of the rewind motor on the left side of the IBM 729.

[61] Future tape drives in the IBM 360 series used mechanisms with sophisticated control over air pressure to load a tape without requiring any help from the operator beyond mounting the tape on the supply hub.

[62] often making an amusing honking sound as the tape is pulled into the columns!

[63] This proves once again that Nature may abhor a vacuum, but IBM CE's bet their jobs on one.

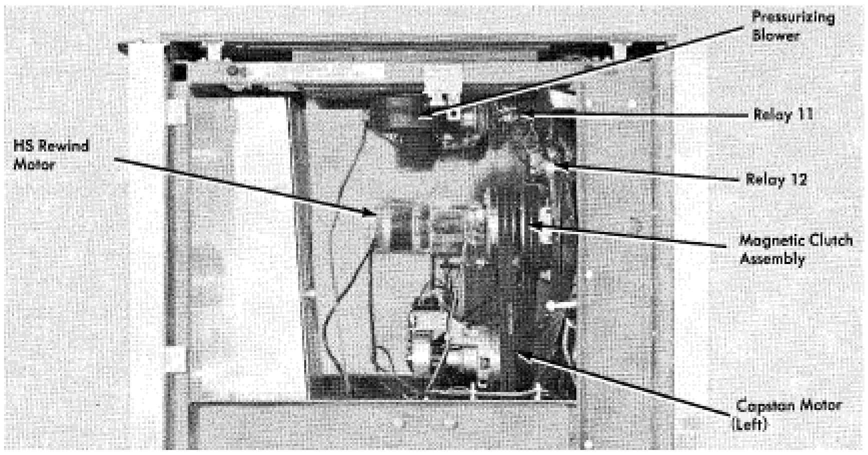
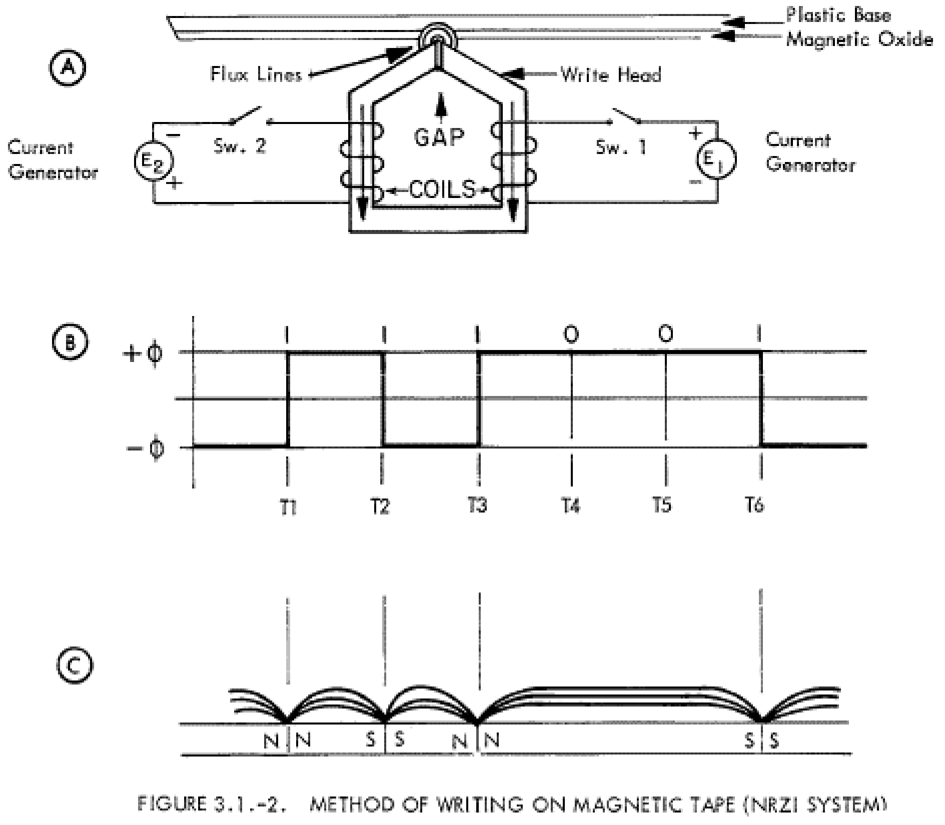


Figure 54: High Speed Rewind Motor

6.3.1.2 Encoding

Magnetic tape is coated with an iron-oxide material that exhibits magnetic hysteresis, that is, the ability to be magnetized in one direction or the other, and to retain that magnetization. Data is encoded on tape by changes in the direction of magnetic flux, in a coding scheme called NRZI – Non-Return-to-Zero-IBM.[64]

[64] Yup, that's what it says. 223-6988_729_CE_Mar62.pdf, Pg 7. Carl Claunch adds: *IBM referred to NRZI as Non-Return-to-Zero-Inverted in the 360 era and beyond, documenting 2400 and 3400 tape drive series that succeeded the 729*.



223-6845_729_CEman_1959.pdf, ibm Pg C9
223-6988_729_CE_Mar62.pdf, ibm pg 7

Figure 55: NRZI Tape Encoding

It should be noted that data on IBM 729 tape is "self-clocked", i.e. there's no separate clock track to indicate where characters are located. Ones are coded as a flux transition, while zeroes are simply the absence of a flux transition. As a result, every character must have at least one "one", so that read logic can tell the difference between "zero" and "nothing"

This has two implications for the tape system:

- There are some restrictions on data that can written to tape (e.g., an all-zero character with even parity cannot be stored on tape; see Section 6.3.4.3)
- Skew between the seven tracks that form a character must be carefully controlled, so that when the first bit of a character appears, the rest of the bits will assuredly turn up within a narrow window (2.3 usec for Model II, 1.5 usec for Model VI) See Section 6.3.5.4 on skew compensation.
- The current record's LRC character is maintained in the "Write Head Trigger". Resetting the "Write Head Trigger" to the NRZI initial state results in writing the LRC character to tape.

6.3.2 Programmer's View of the Tape System

While the tape drive feature is optional for the IBM 1401, support for the feature is wired deeply into the machine.

Tape operations are controlled by several special instructions and branch conditions

- M % U *x bbb d*[65] Move

  This class of instructions causes a tape block to be read from tape into core memory, or written from core to tape. The "B" address specifies the core memory address of the buffer, while the "A" field is used as a modifier to specify which tape unit to use, plus the parity sense to be used during the operation.
- U % U *x d* Unit Control

  This instruction triggers a variety of tape movement operations, such as load/unload, or backspace operations.
- L % U *x bbb d* Load

  This instruction also reads and writes tape, but does so in a way that transfers Word Marks between tape and core storage. This does not make it possible write and read any binary character from tape, but it does make it possible to read executable code from tape without replacing word-marks.[66]

There is also a selection of branch conditions to allow the programmer to sense tape error conditions or the End-of-File condition.

None of the tape instructions explicitly specify the length of the operation… as usual for the 1401, data length is controlled by the use of Word Marks.

- On write-to-tape, the M instruction gives the address of the first character (the 'most significant', or lowest address character) of the record to be written to tape. The position past the last character (at the highest address) to be written is indicated with a Group Mark with Word Mark in memory. Each write operation causes one Record to be written to tape, which the controller terminates with a checksum character and an inter-record gap.
- Read-from-tape is a bit more complicated, because the programmer can't tell in advance how long the tape record will be. As usual, the tape read instruction contains the address of the start of a buffer into which characters will be read, with a Group Mark with Word Mark at the end of the allocated space. If the read operation hits the word mark, the operation is stopped, and the rest of the block discarded. In the normal case that the record fits in the buffer, the end of the record is indicated in memory by a Group Mark character (CBA 8421 = 011 1111).[67]

Tape instructions are described in detail in [RefMan] ibm pg 57 and [Optional Features] ibm pg 75

[Bob Feretich says:

Regarding data transparency

I haven't tried this, but if Write/Read Tape is used (without tape marks) any character can be written or read from tape with the exception of the "Special Blank" (CBA8421=0010000) which upon reading is always converted to Blank (CBA8421=1000000) in memory. Word marks are lost, but then they are not part of the BCD Character Set.

I also think that the Write/Read Tape with Word Marks will pass everything, except special blanks and word separator characters. Most of these special characters are just "use conventions". There is no special reason that that the "Tape Mark" means end-of-file. (Although the EOF flag will be turned on if the first character of a record is a "Tape Mark".) I have written programs to write lots of Tape Marks to tape to create tapes consisting of multiple tape volumes. Scanning for special characters is expensive in terms of hardware, I don't believe that the 1401 designers would choose to add extra hardware to check for these special characters.

[65] Notation: [check this again!] This is an "M" with the word-mark, followed by a "%" sign and the letter "U". "*x*" is replaced by a numeral (1 to 6) for the unit number, "*bbb*" with the memory address, and "*d*" is replaced by a character which specs read or write.

[66] Keep in mind that 1401 machine instructions are not binary-coded… Op-codes, addresses, and all the rest are printable characters (except for Word Marks).

[67] The idea of Data Transparency does not figure in the 1401 – there are a number of special characters that a programmer must be careful to not write on a tape, e.g., Even-Parity Space, Group Mark, etc.

That said… Choosing even parity was a mistake. It must have been some part of a Data Interchange Standard before NRZI became commonly used.]

6.3.2.1 Character Set on Tape

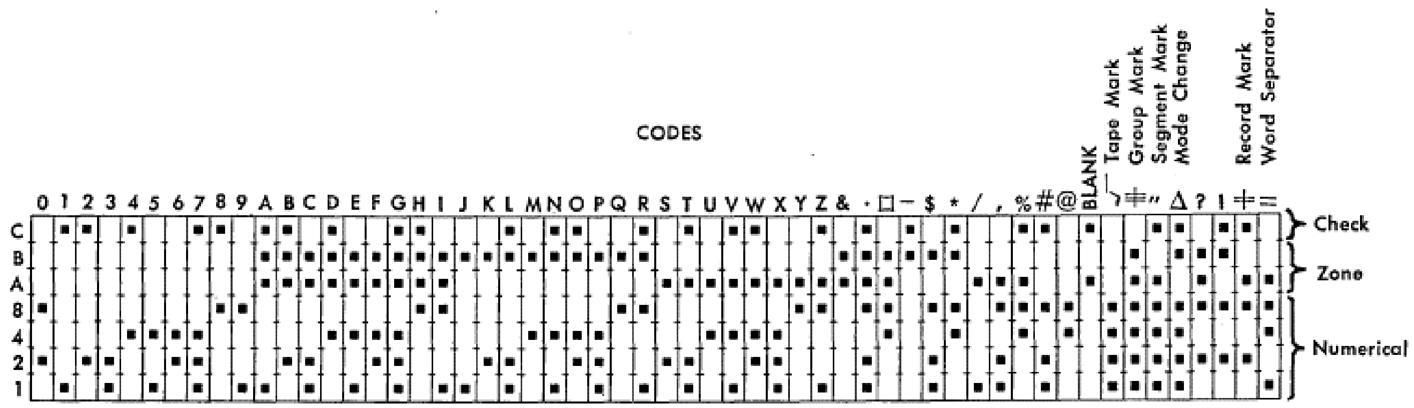
While most character codes written to tape are the same as the corresponding codes in the rest of the 1401 system, the tape system is not "transparent", and imposes some limits on what can be written on tape.[68]

Figure 9 shows the customary character set for IBM 1401 tapes. There are a couple of restrictions:

- The default setting for IBM 1401 tapes is "even" parity[69], i.e., an even number of ones in each character, including the Check bit. In the card reader, Blank characters are written to memory as a zero character, i.e. 00 0000.

  But tape encoding depends on at least one bit on in each character (See Section XX), so an all-zero code can't be written. Instead, the hardware turns on the Zone A bit when writing a blank to tape, converting it to a ¢ (Cent), and turns the Zone bit back off again when the tape is read.[70]

  That in turn means that a Blank character cannot be written to tape with even parity, and that if a programmer uses a ¢ character (01 0000), it will be converted to a space once it's read from the tape (right?).
- A number of characters are also reserved for use as marks and separators, shown in Figure 9.



RefMan Fig 86, ibm pg 53

Figure 56: Character Coding on IBM 729 Tape

Is there a reference doc that says what the hapless programmer is to do?

Bob Feretich says

> It's worse than that. In the days of the 1401 there was no such thing as standardized tape labels (a.k.a. file directories). There was no standard way to name the files on tape. The lack of that drove the development of IOCS (C24-1462), which eventually became part of the operating system. That is once application programmers decided that having each application perform its own I/O error retry and handling was not productive.

[68] This is all hard for your humble editor to believe… there must be a compatibility constraint with the IBM 7000 series from which this tape system was derived. Who would purposely design a machine that couldn't handle a Blank character without special-case processing??

[69] Apparently it's not just the default… the Programmer's Manual doesn't say how to write odd parity at all; only the Field Engineering manuals reveal the secret.

[70] Can someone find a page reference saying this?

6.3.2.2 A Note on Word Marks[71]

The 1401 makes multiple uses of the extra bit in each character in memory called a “word mark” (See Section XX), although that bit is not handled well on I/O operations. Punched card devices can’t set the Word Mark bit, so program object decks read via the card reader must contain extra 1401 instructions to explicitly set Word Mark bits. The magnetic tape subsystem provides the programmer a choice. The Read and Write Tape instructions that use the ‘M’ opcode ignore the Word Mark bit, just like punched cards. The Read and Write Tape instructions (“with Word Marks”) will copy Word Marks from/to magnetic tape.

The 7-track nature of the tape character does not accommodate this occasional extra bit, so the TAU performs a transparent character stuffing operation on the magnetic tape. Upon writing to tape, when a character with a set Word Mark bit transferred, two characters are written to tape. The first character is the “Word Separator” character (0x1D) followed by a second character containing the BA8421 bits of the character in memory, complete with the proper parity.

Upon reading from tape (using a “with word marks” instruction), if the “Word Separator” is encountered on the tape, the TAU suppresses this character, but turns on the Word Mark bit on the next character received from the tape. This feature makes magnetic tapes much friendlier to program object files than punched cards.

This feature ensures that Word Marks can be saved and restored, but does not solve data transparency issues for blanks and separator characters.

6.3.3 Tape System Block Diagram

Figure 10 shows how the tape system is integrated with the 1401 data plane.

Much of the tape control and data path is supplied by subsystem called a Tape Adapter Unit (TAU), originally designed as part of the IBM 7000 computer (footnote?). The TAU design was picked up and moved to the 1401 essentially as-is, with additional logic added to the 1401 to interface the TAU with the rest of the 1401 computer.

[71] Contributed by Bob Feretich, edited by guy

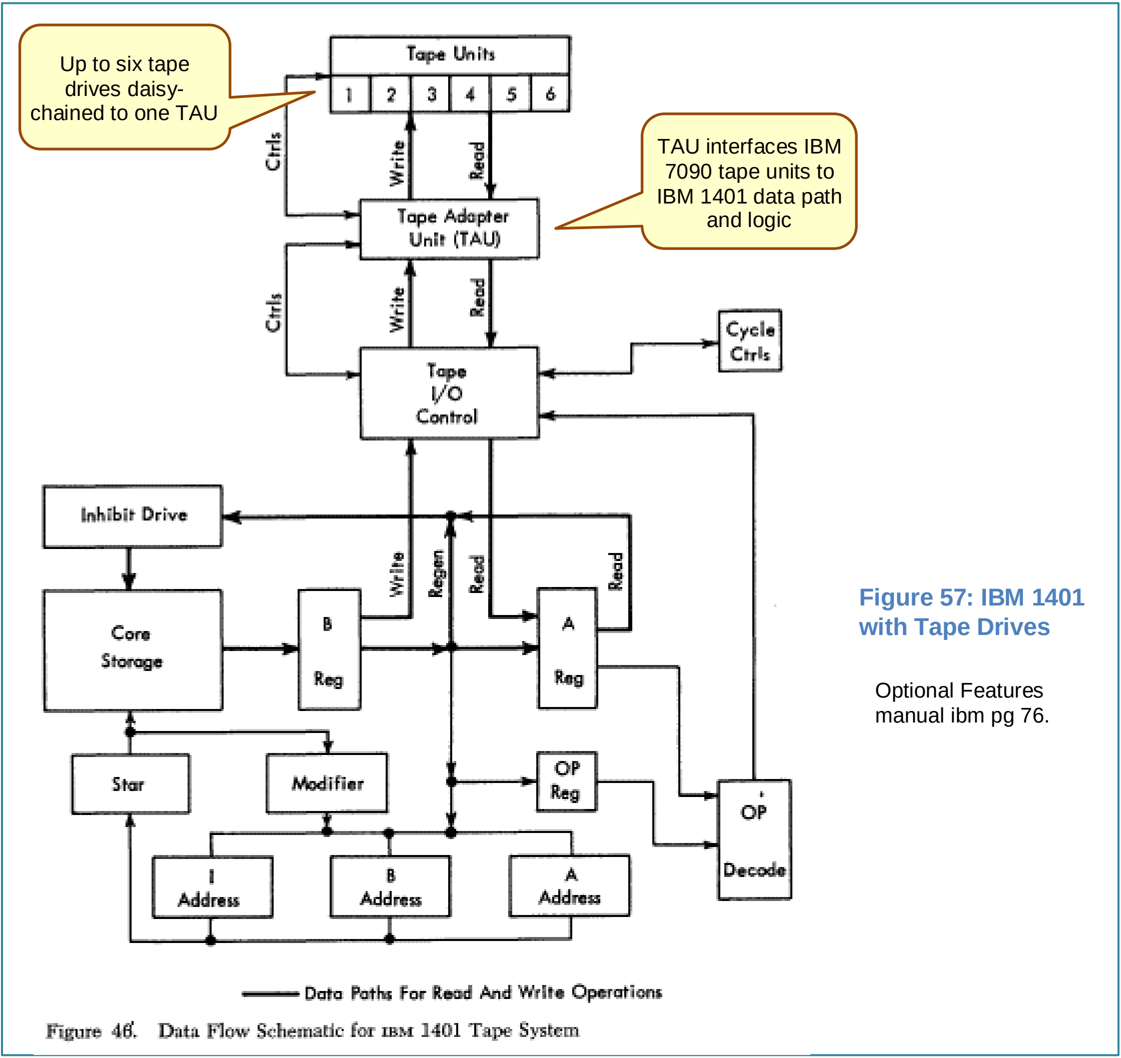

Up to six tape drives daisy-chained to one TAU
TAU interfaces IBM 7090 tape units to IBM 1401 data path and logic
Tape Units
1 2 3 4 5 6
Ctrls
Write
Read
Tape Adapter Unit (TAU)
Tape I/O Control
Cycle Ctrls
Inhibit Drive
Core Storage
B Reg
Regen
A Reg
Star
Modifier
OP Reg
OP Decode
I Address
B Address
A Address
Data Paths For Read And Write Operations
Figure 46. Data Flow Schematic for IBM 1401 Tape System

Figure 57: IBM 1401 with Tape Drives

Optional Features manual ibm pg 76.

Data path interconnect shown in **Error! Reference source not found.** is relatively simple – the TAU is c onnected into the 1401 data path so that it has access to the ordinary read and write paths into core memory, used by every multi-character instruction.

6.3.3.1 Partitioning

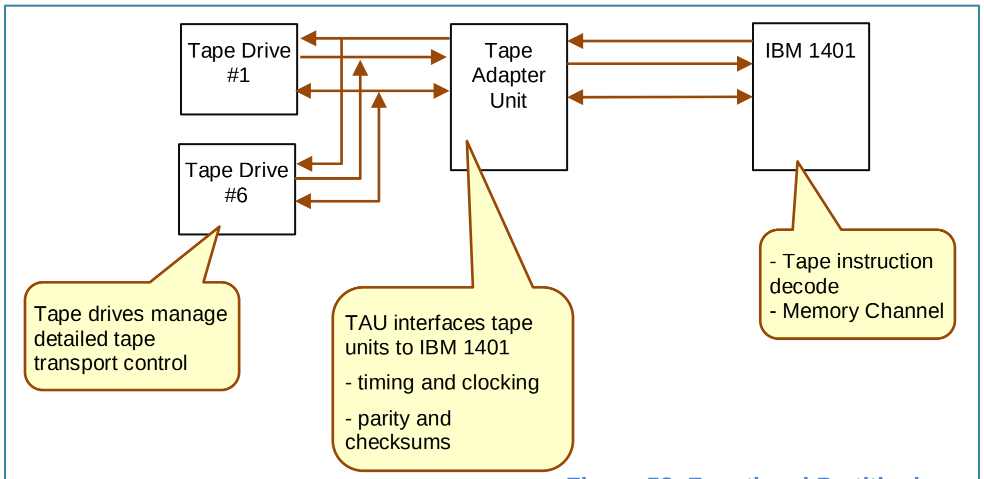

Tape Drive #1
Tape Drive #6
Tape Adapter Unit
IBM 1401
Tape drives manage detailed tape transport control
TAU interfaces tape units to IBM 1401
- timing and clocking
- parity and checksums
- Tape instruction decode
- Memory Channel

Figure 58: Functional Partitioning

Tape Drive

The tape drives themselves contain substantial logic for managing low-level aspects of tape transport such as vacuum columns, reel hubs, load and unload operations, in addition to the analog components for interfacing to read-write tape heads. Various models of the 729 were implemented in relay and electronic logic, using a low-speed CTRL logic family (See Section 6.3.7.1).

Functions in the tape drive itself include

- Read preamps
- Write drivers
- Transport control (e.g. load, forward, backward, rewind, unload)
- Drivers for motors and brakes

TAU

The Tape Adapter Unit (TAU) is a set of three card gates re-used from the IBM 7000 series computers. Unlike the rest of the IBM1401, the TAU uses an SMS logic family called Saturated Drift Transistor Diode Logic (SDTDL)[72] (See Section 6.3.7.2)

TAU functions include:

- Read-character and Write-character state machines
- Read Data analog-digital conversion & clock recovery
- Data transfer operation sequencing. Four operational phases for a transfer ensure the integrity of inter-record gaps (See Section 6.3.4.2)
- Calculation and verification of Vertical (VRC) and Longitudinal (LRC) parity bits
- Write Echo Error and other error detection

The TAU also contains a control panel to allow the CE to operate the drives without need for software intervention. (Section 6.3.3.3)

[72] We think it's faster than the 1401 logic family CDTL, because the 7000 series was designed for performance. But I can't find a doc reference comparing the speed. Let me know if you can find an xref!

1401 Tape Option

The 1401 itself contains specialized logic to control the tape drives, including instructions decoded to trigger tape operations (<u>L</u>oad, <u>M</u>ove, <u>U</u>nit Select), plus a number of branch conditions used to sense the status of the tape drives.

- Tape operation instruction decode
- Tape branch conditions
- Timing interlocks and level conversion between CDTL and SDTDL logic levels

6.3.3.2 Signal Interfaces

Table 5 below shows the signals that form the interface between the TAU and the Tape Drive, and the TAU and the rest of the 1401 system, derived from ILD page 723750.

INPUTS FROM 729 TAPE DRIVE to TAU	**OUTPUTS TO 729 TAPE DRIVE from TAU**		**INPUTS TO TAU from 1401**	**OUTPUTS FROM TAU to 1401**
8VPP READ BUS 1 BIT 729	WRITE BUS 1 BIT 729		1 BIT WRITE DATA LINE	R-W REG 1 BIT
8VPP READ BUS 2 BIT 729	WRITE BUS 2 BIT 729		2 BIT WRITE DATA LINE	R-W REG 2 BIT
8VPP READ BUS 4 BIT 729	WRITE BUS 4 BIT 729		4 BIT WRITE DATA LINE	R-W REG 4 BIT
8VPP READ BUS 8 BIT 729	WRITE BUS 8 BIT 729		8 BIT WRITE DATA LINE	R-W PEG 8 BIT
8VPP READ BUS A BIT 729	WRITE BUS A BIT 729		A BIT WRITE DATA LINE	R-W REG A BIT
8VPP READ BUS B BIT 729	WRITE BUS B BIT 729		B BIT WRITE DATA LINE	R-W REG B BIT
8VPP READ BUS C BIT 729	WRITE BUS C BIT 729		C BIT WRITE DATA LINE	R-W REG C BIT
SEL + RDY WR 729	WRITE PULSE 729		WR CALL	END READ DELAY
SEL + RDY RD 729	WR CK CH 729		WR TM CALL	WRITE COND
SEL + REWIND 729	SET WR STATUS 729		DISC CALL	BUSY
SEL + T1 On 729	SET READ STATUS 729		READ CALL	LOAD POINT
HIGH DENSITY 729	BACKWARD 729		REWIND CALL	CHECK CHAR
WRITE ECHO 729	GO 729		REWIND + UNLOAD CALL	SEL + T1 ON
SEL + LP 729	REWIND 729		BACKSPACE CALL	ERROR
SEL + NOT LP 729	REWIND UNLOAD 729		BACKSPACE CALL	FWD STOP DELAY
SEL + TI OFF 729	TURN OFF T1 729		TAU RESET	WC 2
SEL AND READY MOD 2 729	TURN ON T1 729		ODD REDUNDANCY CALL	WC 8 TR
SEL AND READY MOD 4 729			TURN OFF T1	WR RC-5 OR READ RC-7
MOD 5 OR MOD 6			MAN WR DISC	WD 52
			MANUAL STOP ON FRROR	WD 72
			MANUAL STOP ON ERROR	WD 320 OR WD 1088
			EARLY SAMPLE	WD 49
			EARLY SAMPLE	SEL + READY MOD 4
			AMPLIFIER BIAS	SEL + RDY HI 7330
			COMPARE CHECK CE	729 HIGH DENSITY 800 OR 556 BPI
			REG A ONLY	RC 4 TR
			REG B ONLY	WR TM DRIVE
			TURN ON GO	TU READY
			NOT MANUAL OP	
			SET TAU ERROR LATCH	
			ERASE CALL	
			DENSITY SW 200 + 556	
			DENSITY SW 200 + 556	
			DENSITY SW 200 + 800	
			DENSITY SW 556 + 800	
			DENSITY SW 556 + 800	

Table 5: TAU Interface Signal Chart

See [TAU-ILD], pdf page 2 for the full chart, including signal cross-references to locate pin numbers and page references.

6.3.3.3 TAU CE Panel

The TAU includes a special control panel for use by CE’s when servicing drives, located in the 1401’s Gate 02A1.

The panel allows manually-activated reads and writes of arbitrary data patterns.

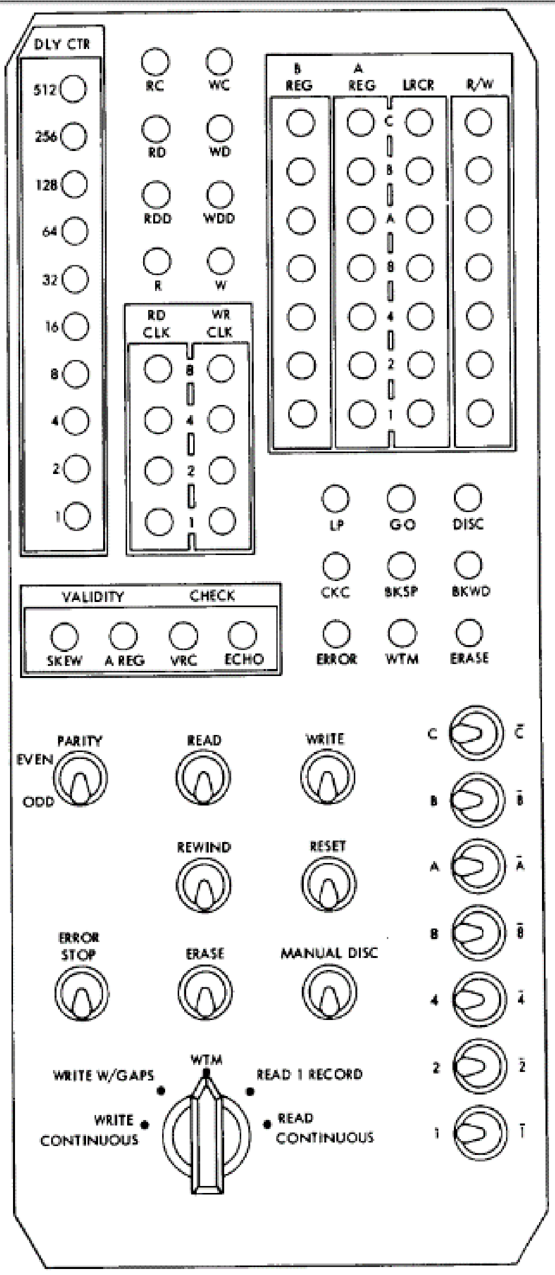


Figure 116. TAU 2 Tape CE Panel

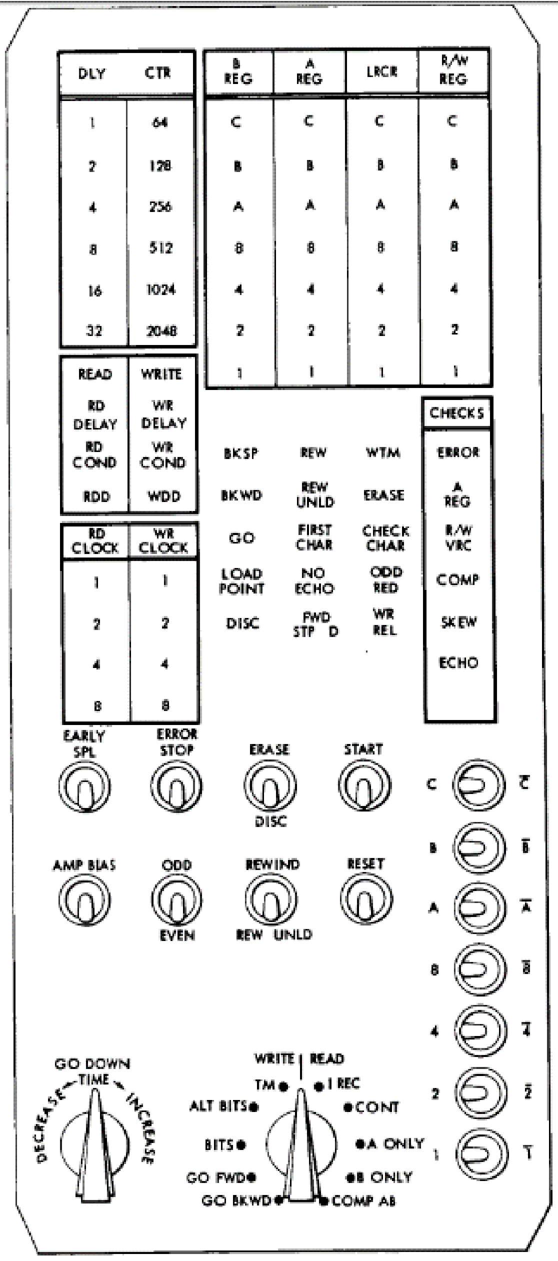


Figure 117. TAU 9 Tape CE Panel

From IBM Field Maintenance Manual 225-6487-3, IBM page 153; both CHM machines have TAU 9 controllers

6.3.4 Timing and Clocking

"*It's a forest of oscillators in there*…" – Iggy

While the bulk of the 1401 is synchronous, driven from a single 11.5 usec master clock that defines a core memory cycle, it can be seen that tape drives, driven by motors, belts and brakes, didn't get the memo on synchrony, and must be timed independently. The TAU is also largely synchronous, but there are two clock domains, “microsecond domain” and “millisecond domain”; and the specific clock source selected is dependent on the tape drive model and tape density of the tape drive with which the TAU is in session.

The difference between the memory cycle clocking domain and the tape drive’s data path timing domain is managed by the TAU on a cycle-by-cycle basis, with the IBM1401 clock state machine held in a wait-state until the TAU is ready for the next cycle. The tape transport has many operations that must be managed by fixed time delays (such as the length of time it takes to start and stop during an IRG); these time-based operations are also managed by the TAU, and depend on the specific model of 729 or 7330 drive.

6.3.4.1 Tape Transport Sequencing

The TAU also has a single 12-bit counter used to time a variety of mechanical functions such as allowance for acceleration and deceleration when starting or stopping movement of tape. This timer also is driven by one of a selection of oscillators ranging from 6.7 kHz to 1 MHz, depending on the specific operation, type of tape drive and density. While some documents refer to this timer as switching from Microsecond to Millisecond mode depending on the operation and delay required, it’s actually more granular than that, with specific oscillators for each condition.

The 1401 CPU initiates a Read or Write Tape operation by making a “Read Call” or “Write Call” to the TAU. The glue logic between the TAU and 1401 (this logic is not part of the TAU gates or schematics) stops the CPU’s clock state machine and holds it stopped until the TAU is ready to transfer data.[73]

The Read Call or Write Call each begin a four phase operation:

	Read Operation		Write Operation	
	Phase Name	TAU-9 Indicator	Phase Name	TAU-9 Indicator
First Phase	Selection Phase	READ	Selection Phase	WRITE
Second Phase	Read Delay Phase	READ+ RD-DELAY	Write Delay Phase	WRITE + WR-DELAY
Third Phase	Read Phase	READ+ RD-COND	Write Phase	WRITE + WR-COND
Fourth Phase	Read Disconnect Delay Phase	READ+ RDD	Write Disconnect Delay Phase	WRITE + WDD

The first phase is the Selection Phase. In this phase, the TAU opens a session with a specific tape drive. The TAU indicates that it is in this phase by illuminating the “Read” or “Write” indicator on the TAU-9 control panel. (The lights indicating transfer phase below the Read or Write indicators are all off to start.) The TAU asserts the “Select” signal to the tape drive to which it is to use for the read or write data transfer. The selected tape drive responds by applying its characteristics and state to the I/O interface. The drive will inform the TAU of its type and model number, its mode (read, write, or rewinding), its error condition (tape indicator), its density setting (high or low), whether it is positioned at the “load point” (detecting the reflective load point strip), and its state of readiness. The drive will also open an analog channel from the read head preamplifiers to the TAU. Since these signal lines are common to all tape drives, only the selected drive is permitted to assert to these signals.

[73] Note that some 1401 have the “Overlap Option” installed. This option permits the CPU to overlap some of the tape I/O operation. The effect of this option is not considered here.

The IBM 729 identifies itself as a 112.5 IPS drive (Model V or VI) by asserting the MOD_5_OR_MOD_6 signal. Otherwise, it is identified as a 75 IPS drive (Model II or IV).

- If the drive is a 729-II or 729-V and is ready to perform an operation, it asserts SEL_AND_READY_MOD_2_729.
- If the drive is a 729-IV or 729-VI and is ready to perform an operation, it asserts SEL_AND_READY_MOD_4_729.

By interrogating the MOD_5_OR_MOD_6, SEL_AND_READY_MOD_2_729, SEL_AND_READY_MOD_4_729, and HIGH_DENSITY_729 signals from the selected tape drive, and the three DENSITY_SW signals from the 1401, the TAU can deduce the drive type with which it is in session, and the tape density with which it is to operate. If no SEL_AND_READY_xxx signal is active, then the TAU will hang waiting for the selected drive to become ready. (Usually the operator hasn't finished loading the tape on the drive yet.)

Once it has received the ready assertion and determined the drive type, model, and density, the TAU selects the "Microsecond" and "Millisecond" oscillators to be used for this session.

- For a write operation, the TAU then asserts the SET_WR_STATUS_729 signal and waits for the selected drive to respond with SEL+RDY_WR_729. If the tape is write protected, then the TAU will hang waiting for the tape's "write ring" to be inserted; otherwise the drive will energize its "Write Head". Movement of the tape across the energized head will erase the tape.
- For a read operation, the TAU then asserts the SET_RD_STATUS_729 signal and waits for the selected drive to respond with SEL+RDY_RD_729. If the drive *was* in write mode, this will de-energize its "Write Head".

All of the above is performed without clocking via combinational logic. The sequencing is performed via the signal/response handshakes between the TAU and tape drive. Once SEL+RDY_WR_729 or SEL+RDY_RD_729 is received, then the TAU proceeds to the next phase.

The second phase is slightly different for read and write. In either case, the second phase starts the motion of the tape and provides time for the tape to obtain the desired stable speed (75 or 112.5 IPS)

- The Write Delay Phase also generates the second half[74] of an IRG. The TAU indicates that it is in this phase by illuminating the "WR DELAY" indicator on the TAU-9 control panel. (The WRITE indicator remains on.)
- The Read Delay Phase also starts the motion of the tape and provides time for the tape to obtain the desired stable speed, but the elapsed time allowed is shorter than the corresponding time of the Write operation. The TAU indicates that it is in this phase by illuminating the "RD DELAY" indicator on the TAU-9 control panel. (The READ indicator remains on.)

To initiate motion, the "GO_729" signal is activated by the TAU, commanding the drive to engage the prolay, causing the tape drive to accelerate the tape as the Delay Counter starts counting using the "Millisecond oscillator". During a write operation, the "Write Head Trigger" is in its reset state, so magnetic tape is being erased. The length of delay for this phase depends on Read or Write, and also whether the tape started at the Load Point (a longer gap is used for the first block).

	Read Operation	Write Operation
Tape at Load Point	Delay Counter = 160 (24ms@75IPS)	Delay Counter =320 (48ms@75IPS)
Tape Past Load Point	Delay Counter = 44 (6.6ms@75IPS)	Delay Counter =50 (7.5ms@75IPS)

[74] The first half of the IRG was written at the end of the previous block…

Table 6: Delay Counter Settings for IBM 729 Drives

Once the TAU delay counter reaches the specified count, the TAU will enter the next phase, which transfers data to or from the magnetic tape (see Section 6.3.5.2 for Write and 6.3.4.3 for Read). The TAU indicates that it is in this phase by illuminating the "RD COND" or "WR COND" indicator in addition to the READ or WRITE indicator on the TAU-9 control panel.

Following completion of data transfer, the fourth phase is the Read or Write Disconnect Delay Phase. The TAU indicates that it is in this phase by illuminating the "RDD" or "WDD" indicator in addition to the READ or WRITE indicator on the TAU-9 control panel.

The first part of this final phase manages reading or writing the LRC character, while the second part decelerates the tape (see respective sections below).

6.3.4.2 Write Data Transfer Timing

As a recording medium, magnetic tape is "self-clocking", that is, the magnetic medium itself has no mechanism to delineate bit boundaries. As such, it's up to the TAU to ensure that the bit density written onto a tape comes out at approximately the right density. Too many bits-per-inch, and reliability will be compromised, too few and valuable space is wasted. And too much difference from the standard densities will cause interoperability problems when tapes are written on one drive and read on another.

Write timing is determined by a "Microsecond" crystal oscillator in the TAU, selected for this session, based on drive type and density. Frequencies range from 240 KHz to 1 MHz for 729 drives. Each type of tape drive has its own tape movement speed, determined by the rotational speed of the capstan, and its own settings for high and low density (see Table 4); these parameters determine which oscillator should be selected to time the tape write operations. Density is configured manually by the operator; there's a rotary switch on the operator's panel to pick combinations, and then the drives themselves have a "High/Low Density" switch on the front panel.

As noted in Section 6.3.4.1, when a Write operation is started, the TAU issues a command to engage the prolay and start the tape moving. It times out a period long enough to allow the tape to get up to speed, and then starts writing characters to the tape at a rate timed to produce the right number of bits per inch, without stopping or pausing until the end of the record.

During the write operation, the appropriate TAU "Microsecond" oscillator steps a write state machine called the Write Clock through sixteen states. The TAU holds the 1401's 11.5 usec core-memory clock stalled at state T105-T120 by preventing the 1401's state machine from advancing (See ILD pdf pg. xx). When the TAU's state machine logic needs the next character from memory to write to tape, it issues [name the signal] to the 1401 to allow its memory clock to complete one 11.5 usec cycle and fetch the next character from memory.

So while the TAU and 1401 microsecond clocks are asynchronous, during a write operation, the memory cycle rate is controlled by the TAU write state machine. (And Yes, Virginia, that means there's a synchronizer in there somewhere; maybe we can identify the flop(s)[75])

At the end of the write operation (indicated by a Group Mark with Word Mark in the 1401 memory), the TAU pauses about two character times, writing a small gap, then writes the longitudinal check character to signal the end of the record, accomplished by entering the Write Disconnect Delay Phase (the fourth phase).

[75] Bob Feretich says: The synchronizer is not in the TAU logic set. It must be in the "1401 glue logic somewhere. Robert Garner and I had some discussions about metastability and the 1401, especially since metastability was not well understood until much later. The pluses for the 1401 was that its logic was noisy and each stage had a very high (relative to ECL, where metastability was a nightmare) gain per stage. Both of these factors worked in favor of a low incidence of problems. However, we were both too lazy to search for the synchronizer.

In this phase, TAU starts the Delay Counter (using the Microsecond oscillator), which counts to 60 (about equivalent to 2.25 character times), and asserts WR_CH_CH_729. This causes the 729 Write Head Trigger to be reset, which results in the writing of the LRC to the tape. Although the TAU never sent the LRC to the 729, the recorded LRC passes over the Read Head and is sent to the TAU. The received LRC is compared to the TAU-calculated LRC to verify correctness of the record.

The Delay Counter is then restarted in Millisecond mode and when the count reaches 20 (3 ms @ 75IPS) The "GO_729" signal is deasserted causing the 729 to switch the prolay to "stop" and decelerate tape.

The write portion of the operation is complete, but until the LRC character is received from the 729, the Read state machine will remain active. The Delay Counter continues to count in Millisecond mode and by count 36 the LRC is expected to be received. If it has not, an error is signaled. At count 128, the "RD COND" is turned off and the "RDD" indicator is turned on. At count 144, the "RDD" and "WRITE" indicators are turned off and the Write Operation is complete. This phase has resulted in the completion of read-back error checking and creation of the first part of an IRG.

Note that the entire time this phase has been in progress, the recently-written data has passed from the 729 Write Head to the Read Head (actually these are portions of a single combined Write/Read Head; See Figure 15). The read-back data is checked for VRC validity using the "high-clip" detectors and aggressive skew constraints. Detected faults are reported as an error. While this is occurring the Read indicators will be outputting the state of the TAU Read state machine which is operating several characters behind the Write state machine. Also the 729 wraps each WRITE_PULSE_729 data strobe back to the TAU via the WRITE_ECHO signal. The TAU also checks that no strobes are missing.

6.3.4.3 Read Data Transfer Timing

On a read operation, timing is driven from the tape movement itself, to accommodate slight variations in speed due to any number of mechanical parameters.

Each character read from tape is timed independently.

Once the tape movement is up to speed, the Read Phase of the operation is started and the read timing logic waits for a First Bit, (essentially a logical-OR of the seven parallel read amplifiers) to identify the First Character of the record.

Unlike the TAU crystal-controlled "Microsecond" oscillators used by the Write operation the TAU starts an oscillator controlled by an inductor-capacitor tank circuit running at the appropriate speed for the tape drive model and density. This oscillator runs the Read Clock (RC) state machine through up to twelve states to do the following:

- Wait for the worst-case skew between the seven lanes of bits,
- Check the parity of the previous character in RW-Reg at RC=3, and the current character in TAU A-Reg at RC=7, and then
- Trigger a 1401 core memory cycle to write the character to memory.

Again, although the 1401's main oscillator continues to run, its 11.5 usec memory clock is held at T105-120 while waiting for a character from the TAU.

Once the read state machine has completed its cycle, it stops the selected read oscillator from oscillating,[76] and commences waiting again for an asynchronous arrival of the First Bit of the next character.

[76] For a modern guy, this is a mighty unusual practice, to halt the oscillator and expect it to start at exactly the right frequency on the first cycle.

After the First Character of a record is received, the Delay Counter is started in “Microsecond” mode. Since the Delay Counter is not running when looking for the First Character of a record, the TAU will wait indefinitely for the First Character, however for subsequent characters, if the Delay Counter reaches 36 (37 us @ 75 IPS @ 800 CPI) the TAU recognizes the end of the data record and now expects the LRC character. Unlike other characters the LRC character may be all zeros (no flux changes, which is really the absence of a character). If the Delay Counter reaches 128 the TAU gives up and assumes the LRC is all zeros. When the LRC character is received or when the TAU gives up on waiting for the LRC character, the fourth phase is entered.

The forth phase is the Read Disconnect Delay phase. The TAU indicates that it is in this phase by illuminating the “RDD” indicator on the TAU control panel. (The READ indicator remains on. The “RD COND” indicator will be off.)

The TAU Delay Counter continues to count in “Microsecond” mode. When the Delay counter reaches 144 (150us @ 75 IPS @ 800CPI), The “READ” and “RDD” indicators are turned off and the operation is complete.

6.3.5 Tape Data Path

6.3.5.1 Tape Read Data Path

During tape read operations, 15-30 millivolt signals are generated by the tape read head windings[77], and passed directly to preamplifiers built into the tape drive itself, one for each of the seven channels. The selected unit then drives the read signals in analog form, with no further processing, onto the daisy chain linking tape drives to the TAU in the IBM 1401. Nominal voltage at this point is about 8 volts peak-to-peak for a read-after-write signal.[78]

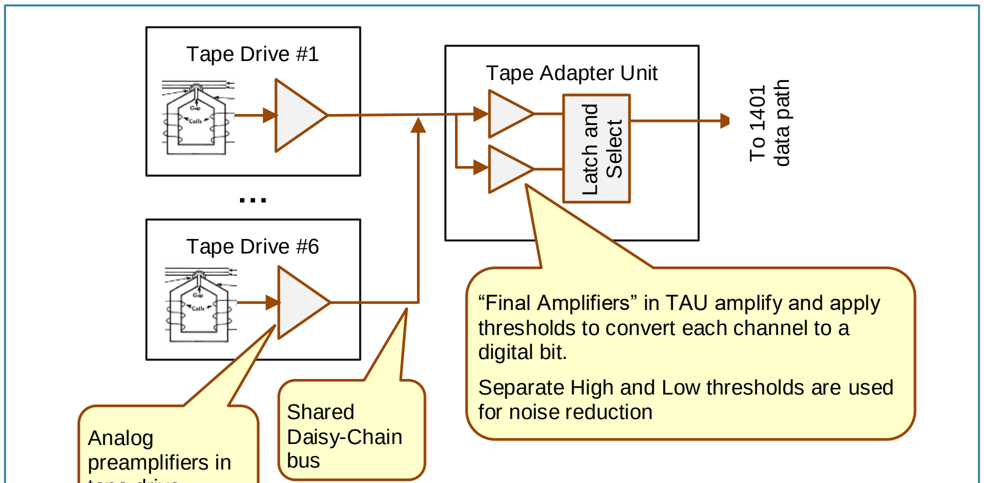

Tape Drive #1
...
Tape Drive #6
Tape Adapter Unit
Latch and Select
To 1401 data path
“Final Amplifiers” in TAU amplify and apply thresholds to convert each channel to a digital bit. Separate High and Low thresholds are used for noise reduction
Shared Daisy-Chain bus
Analog preamplifiers in tape drive

Figure 59: IBM 729 and TAU Data Path

Within the TAU, signals are passed through *Final Amplifiers* that increase the amplitude and rectify the positive and negative pulses, and then pass them through threshold detectors to discriminate *ones* from *zeroes*.

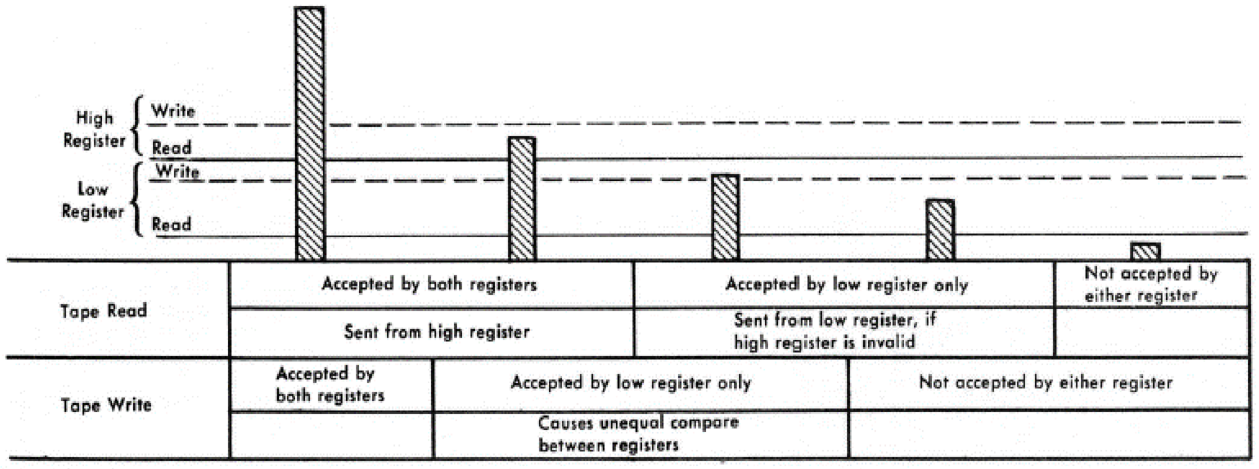
[77] Details on tape head parameters can be found in 223-6988_729_CE_Mar62.pdf ibm pg 25

[78] See 223-6988_729_CE_Mar62.pdf ibm pg 34 for waveforms.

There's a bit of subtlety about the thresholding mechanism. Each read channel is applied to two Final Amplifiers, one set with a high threshold, the other with a lower threshold, used to detect marginal signals from the tape, as shown in Figure 13.

- On data read back during a write, signals must exceed the high threshold. If any just-written bits only exceed the low threshold, an error is declared.
- During ordinary read operations, both thresholds are reduced.   If the high-threshold register collects a character with good parity, that's used. If the high-threshold character has a parity error, indicating a missing bit, the low-threshold character will be accepted.

Logic to carry out these functions can be found in [TAU-ILD] Figure 190, pdf pg 12. More detail on the use of thresholds for error control is found in Section 6.3.5.3.



Tape Read	Accepted by both registers	Accepted by low register only	Not accepted by either register
	Sent from high register	Sent from low register, if high register is invalid	

Tape Write	Accepted by both registers	Accepted by low register only	Not accepted by either register
		Causes unequal compare between registers	

Figure 93. Relative Sensitivity Levels
(from RefMan ibm pg 57)

Figure 60: TAU Read Thresholds

6.3.5.2 Tape Write Data Path

When writing tape, the TAU places each character on the Write Bus, so it can be received by the selected drive. Each "one" written to the tape results in a reversal of the flux in the write head, as shown in Figure 14.

The IBM729 has a separate Erase head which is activated during a write operation to ensure that the entire width of the tape is uniformly erased.

The Write Data Path can only be activated if there's a plastic write-enable ring present in the reel mounted in the supply hub.

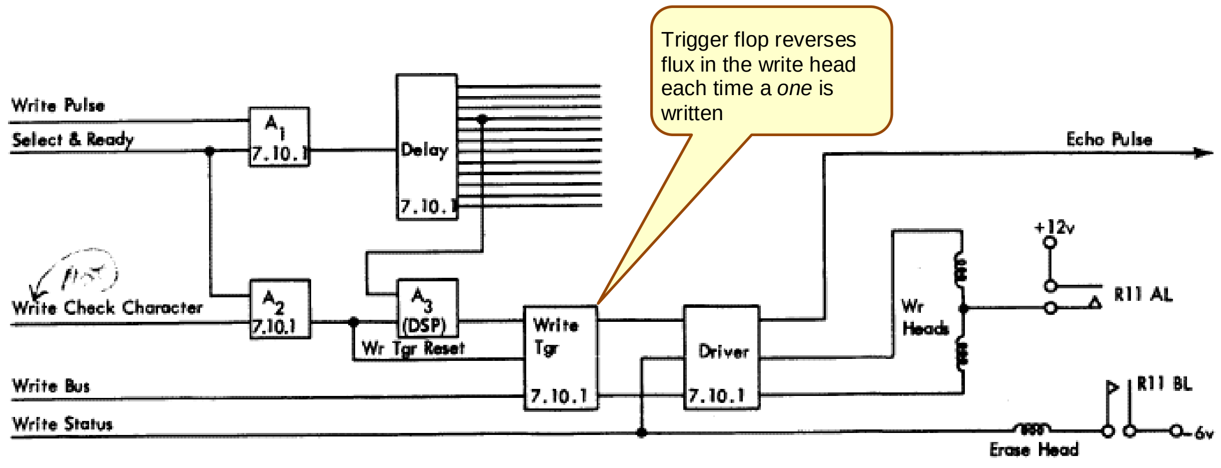


Figure 71. Write Circuit
(from 223-6988_729_CE_Mar62.pdf ibm pg 65)

Figure 61: Write Data Path in IBM 729

6.3.5.3 Error Checking

Each block read back from tape is protected by two parity checks

- A Vertical Redundancy Check is the parity check bit written in the seventh track of the tape. VRC can be even or odd parity (See Section 6.3.1)
- The Longitudinal Redundancy Check is parity across each track in a tape block, and is checked when the end of a tape block is detected.

The IBM 729 also does two checks on write data as it's written:

- The tape write head circuitry develops a Write Echo signal that checks that the write heads were activated. This signal is returned to the TAU, and while it doesn't check that the correct data was written, it does confirm that the heads were activated.
- Tape is written as it moves past the write gap in the read-write. The tape then moves past the read head gap, where it's read back from the tape. Read and write head gaps are about 0.3" apart, so at 200 bits per inch, the recently-written character is read back about 60 character times after the write was done, by which time the actual character values written are long gone. But the read-after-write can check for bad parity or weak signals, and indicate that the tape block didn't record properly. There's a mechanism to erase a gap of about three inches and try again, in the likely circumstance that the error was caused by a flaw in the tape itself.

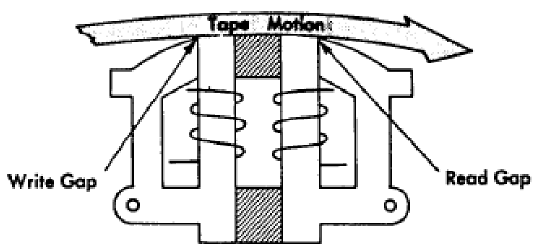


Figure 62: Dual-Gap Write/Read Head

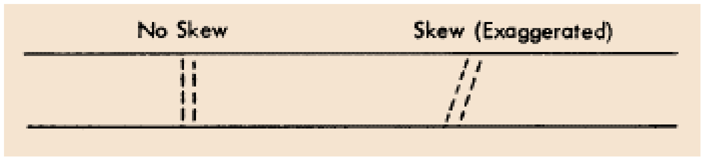
[RefMan] IBM pg 56

Bob Feretich outlines the dual-threshold read-error check mechanism:

> Every character that is read from the tape is transferred to the TAU's A-Reg (the high-clip result) and the B-Reg (the low-clip result). A stronger flux change needs to be detected to set a bit to '1' in the A-Reg.
>
> The A-Reg is parity checked. After every character is received, the A-Reg and B-Reg are compared.
>
> When tape Write operations are performed, data is written to the tape as the tape passes over the write head. The read head is ¼ inch from the write head. The written data is read back into the TAU as the tape passes over the read head. Although it is not possible to compare the read-back data to the original data (many characters would have been written during the time the tape moved that ¼ inch), the received character is parity checked. The read-back data must be received properly in the A-Reg under tighter skew conditions than used for a Read Tape operation. Otherwise, the Write Tape operation will signal an error. The most common reason for this error is the oxide being worn out on the tape, causing weak flux transitions.) If the error occurs, the application program is expected to backspace and rewrite the record. If rewriting fails, then the application program is expected to backspace and "Skip and Blank Tape" to skip over the defective tape segment, then rewrite the record.
>
> This mechanism results in a newly written tape having strong flux transitions and tight skew throughout. When the Read tape operations are performed, bad A-Reg parity or an A/B miscompare results in the contents of the B-Reg being stored into memory. The error condition is raised and the application program should reread the record, but if the reread failed, the correct data may still have been stored into memory because a weak flux transition may still have been captured by the "low-clip" B-Reg. It provides a possibility of data recovery from old tapes.

6.3.5.4 Skew Adjustment

Data recovery from IBM 729 tape depends on the assumption that all the bits of a character in the seven parallel tracks arrive at the same time. As density went up, this became more of a challenge as small variations between heads and circuitry would cause each track to have slightly different delay, called *skew*. In the IBM 729, there was circuitry specifically to compensate skew,



For many applications, interoperability between tape drives was critical, so that a tape written on one drive could be stored and read much later on a second drive. To achieve reliable interoperability, it's necessary to independently compensate skew on write and read data paths.

Skew is manually adjusted by selecting taps in an inductor/capacitor passive delay line. Figure 16 shows the outline of the circuit.

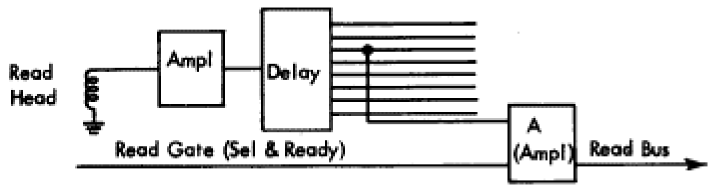


Figure 70. Read Circuit
(from 223-6988_729_CE_Mar62.pdf ibm pg 65)

Figure 63: Read Delay Compensation

To do the complete job, the CE would periodically carry out these steps[79],[80]

- Adjust mechanical skew
  - Mount a factory-calibrated *master skew tape* which is written with all-ones, guaranteed to be aligned properly
  - Read the tape using the CE panel controls (right?)
  - Observe the signals from the tracks corresponding to the “1” bit and “C” bit (the two edges of the tape) at the read-preamp output on the read-bus, and adjust mechanical angle of the head assembly until these two align.
- Adjust electronic Read Skew
  - Mount the same Skew Tape
  - For each channel, observe the signal at the read-preamp output on the read-bus, and adjust the taps on the delay lines until all seven align
- Adjust Write Skew next
  - Mount a scratch tape, and write all-ones using the CE Panel
  - For each channel, observe the signal at the read-preamp output on the read-bus, and adjust the taps on the delay lines until all seven align

By the end of the process, worst-case skew must be less than 2.3 usec for Model II, or 1.5 usec for Model IV.[81]

6.3.6 Tape Transport Control Logic

Bob Feretich says:

> You should probably write a little about the "Rewind", "Rewind & Unload", and "Tape Indicator" logic. All of these are implemented in the TAU/729 by combinational (unclocked) logic. The failure of one of the handshake signals results in everything being hung. There is no indication if it was the CPU, TAU, or 729 that failed. The Emulator manual (Section 6.3.8) has a pretty good description of each of the 729/TAU interface signals as well as a discussion of the "Rewind" and "Rewind & Unload" operations.
>
> Tape Indicator seems to be a giant set/reset flip-flop that has its feedback paths span the TAU and the 729; and has some subtle side effects on TAU operation. I troubleshot a problem in this logic before. The failure of one of the signals in the feedback loop caused the TAU/drive to hang and requires the troubleshooter

[79] Described in 223-6988_729_CE_Mar62.pdf pg 28

[80] CHM typically doesn’t do the whole procedure, given that master skew tapes are hard to find, and interoperability between drives isn’t a primary requirement for museum activities.

[81] See 223-6988_729_CE_Mar62.pdf, ibm page 28.

(me) to dig through pages and pages of TAU and 729 ALDS to identify the entire feedback circuit. Unfortunately, I don't recall the implementation. This is an area where I am not sure that I got the Emulator implementation correct.

6.3.7 Tape Option Logic Families

The IBM 1401 tape option involves two additional logic families beyond the CDTL logic used in most of the 1401.

- The TAU uses SDTDL (Saturated Drift Transistor Diode Logic), as was used in the IBM 7000 series.
- Motion Control logic in the IBM 729 is implemented in CTRL (Complementary Transistor Resistor Logic), a lower-speed resistor-transistor logic family.

In addition, there are special logic-converter cards in the 1401 to convert logic levels in the generic TAU module into the CDTL used in the rest of the 1401.[82]

See Section XX for a summary of all the logic levels used in SMS cards.

6.3.7.1 IBM 729 Logic Family

The 729 tape transport contains quite a bit of logic to manage motors, solenoids, indicators, error checks, etc, and it's all implemented in Complementary Transistor Resistor Logic.

[We thought this is true, but I can't prove it at the moment!]

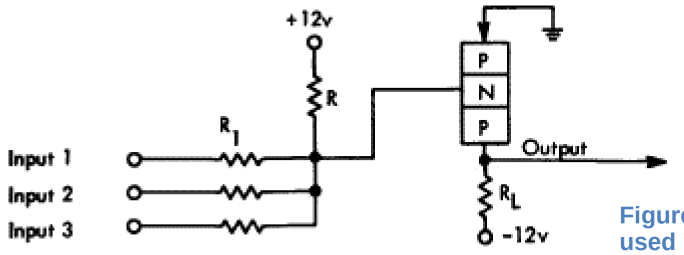


Figure 64: Typical CTRL Logic used in IBM 729

Logic family CTRL

+S = 0.0v, -S = -12v Defined in [SMS Card Manual] ibm pg 16

+R = +12v -R = 0v

6.3.7.2 TAU Logic Family

The Tape Adapter Unit uses Saturated Drift Transistor Diode Logic (SDTDL), a form of Diode-transistor logic using drift-field transistors. Faster. A two-input SDTDL gate can have a turn-on time of 75 to 100 nsec (faster than the xx nsec CTDL logic used in the 1401)

[82] e.g., card type XX on ALD page XX

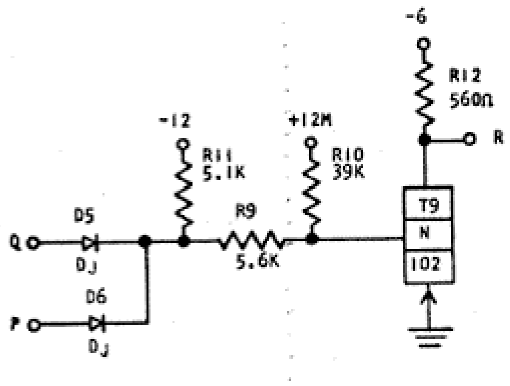


Figure 65: Typical SDTDL Logic used in TAU

http://files.righto.com/sms/DGT.html

Example Cards DGT, DFR, DHF, DHB, DGY

6.3.7.3 TAU-to-1401 Level Converters

Card EY, see ALD 71.11.11.2 [?? It's a card full of resistors!]

6.3.8 CHM 729 Tape Drive Emulator

Early in the process of restoring the 1401 machines at CHM, restoration team members Bob Feretich and Grant Saviers designed and built the CHM 729 Tape Drive Emulator (aka Tape Channel Analyzer), a purpose-built device to emulate a string of up to six 729 tape drives. The Emulator is plugged in to the TAU in place of physical tape drives, and can be used to exercise the TAU and 1401 software that manages the tape drives, both by emulating normal operation, and also simulating some error conditions that aren't easy to produce with the real machine.

While the Emulator plugs in to the 1401 using the same kind of electrical interface and cable connector as a "real" 729, the rest of the analyzer is more modern technology, with an embedded PIC microcontroller and a USB interface to a PC web-server that runs the software component of the Emulator. The web-server permits the 1401 to read/write tape files via WiFi from/to a user's notebook computer. The web-server also maintains a library of demonstration and diagnostic object code tapes from which the 1401 can be booted.

The Emulator specification also includes detailed descriptions of some of the sequences of interaction between the TAU and its tape drives; many of these form extended, distributed asynchronous state machines and can be hard to debug.

The Emulator is described in a specification at http://ibm-1401.info/TapeXnalAnalyzerSpec_1_3.pdf. Additional engineering documentation can be found at http://ibm-1401.info/FeretichTapeDriveEmulator.html

6.3.9 Questions

What's "Compressed Tape" mode do?

- It can be used to suppress leading zeros on numeric fields. See page ibm 99 in the 1401 optional features manual

Are there scanned ALDs for the 729?

6.3.10 Tape References

http://ibm-1401.info/0KenShirriff/IBM_Customer_Engineering_manual_of_instruction_Tape_Adapter_Unit_223_6847_2.pdf

Optional Features Manual pg xx covers many aspects of the tape system, ibm pg 75

http://ibm-1401.info/MagneticTapeVisualization.html

IBM Field Engineering Maintenance Manual 1401 System, 225-6487-3
The "Tape Adapter Unit (TAU-9)" section (starting page 138) acts as a Theory of Operation for the IBM 1401 TAU.

http://ibm-1401.info/729-Info.html

http://bitsavers.trailing-edge.com/pdf/ibm/magtape/729/223-6845_729_CEman_1959.pdf

http://bitsavers.trailing-edge.com/pdf/ibm/magtape/729/223-6988_729_CE_Mar62.pdf

6.4 Serial I/O[83]

There's a general-purpose I/O channel for "serial" devices that transfers a character at a time through a character-at-a-time interface.

The Serial I/O port is used for a variety of interface types, for example, the IBM 1407 Console typewriter, or the special-purpose reader for magnetic lettering on bank checks.

While the Serial I/O feature is independent of the tape controller, it does use the same instruction types.

[It sounds like it does share some logic, but probably not any in the TAU; check the ILDs]

At CHM there actually is a custom-built adapter to attach the IBM Serial port to a USB adapter, allowing connection to a PC]

See Optional Features ibm pg 111 for details.

[83] Not exactly a four-pin Universal Serial Bus (USB) – *This* serial bus connects with a 200-pin connector… The 1401 Serial Bus is more accurately a bit-parallel, character-serial interface.

7. Software Environment

The ibm-1401.org web site has numerous resources for programming the machine, including links to a simulator

- 1401 Software Development
    - ROPE (Ron's Own Programming Environment)
    - Punching object decks from ROPE to a keypunch
- A Guide to IBM 1401 Programming by Daniel D. McCracken, 1962
- Programming the IBM 1401 computer by Emanuel Melichar
- *IBM 1401: A Self-Instructional Programmed Manual* by Saxon and Plette http://ed-thelen.org/comp-hist/ProgrammingTheIBM1401.pdf

For a list of IBM software, see IBM Program Catalogues:

- *Catalog of Programs for IBM 705 – 1410 -7010 – 7070 – 7072 – 7080 – 7740 - 7750 Data Processing Systems* C20-1602-8 – Jun 1968 - http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/1410/C20-1602-8_1410_pgmCatJun68.pdf
- *Catalog of Programs for IBM 1240-1401-1420-1440-1450 and 1460 Data Processing Systems* GC20-1601-10 Jan 1971 https://ia801608.us.archive.org/5/items/bitsavers_ibm140xGC2gramsJan71_8095197/GC20-1601-10_Catalog_Of_Programs_Jan71.pdf

From Bill Worthington, Oct 2016

As a "refresher" on the program Type designation:

Type I programs were written and funded by IBM. They were usually system related programs -- sort, compilers, disk utilities, etc. The Catalog of Programs calls them "IBM Programs."
Type II programs were also written and funded by IBM, Here you'd find what I'll call industry-supported applications -- Shared Hospital Accounting System was one that I was familiar with after joining IBM's Providence, RI.. The Catalog of Programs calls these "IBM Programs" too.
Type III programs were programs written by IBMers and put into the public domain. I wrote a 1440 core dump program which didn't destroy as much content as the IBM provided memory dump. I had another that was submitted as the program library was being shut down. They were "contributed programs." (It's in the Museum's archives.)
Type IV programs were submitted by customers into the public domain too. It was a time of sharing.

Type I and II programs were developed by IBM and support was provided.
Type III and IV were unsupported meaning that if something broke, the author might try to find time to develop a fix. ("Call your mother because she might care.") The author's priority was on other things.

Can we describe software workflow?

See an example of code at Ken Shiriff's blog:
http://www.righto.com/2015/05/bitcoin-mining-on-55-year-old-ibm-1401.html

Do we have Python yet?[84]

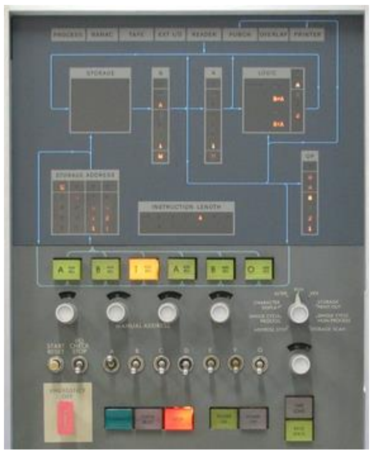
[84] Actually, no. It would be possible to write a C-like cross-compiler that would generate a binary to run on a 1401 [someone's doing it, no?] but Python relies on a complex run-time p-code interpreter that has to be resident on the target machine. Not likely to ever fit in the small memory available in a 1401.

8. Operational Environment

8.1 Operational Guide

Are there instructions for turning the machine on and starting it up?

8.2 Operator's Panel – The 1401 Console



Modern programmers are accustomed to a wide array of software-based debugging tools to see what's going on inside their program. In the case of the 1401, that's done with the operator's panel, which controls single-step, shows register contents, can examine and change memory locations, plus other functions required to start and stop programs or control power.

[notes from Ken Shiriff, Sep 2015]

The IBM 1401 doesn't require much console interaction in normal use, but there are some useful things to know about it. The console also provides debugging features.

8.2.1 Default settings

Several console switches must be in the correct positions or else the 1401 will not operate normally. The I/O CHECK STOP toggle and A toggle should be up. The mode switch should be in the RUN position. The TAPE SELECT dial should be in the N (normal) position. On the auxiliary console, the CHECK STOP toggle should be up. The auxiliary mode switch should be set to OFF.

While these settings are useful defaults, the full description of all the console switches and lights can be found in the Reference Manual [RefMan] at ibm-pg 109 (pdf-pg 121).

8.2.2 Useful buttons

The POWER ON and POWER OFF buttons turn the 1401 system on and off. The CHECK RESET button lights if there is a fault, and is cleared by pushing the button. The START RESET button resets the system (except for address registers and core).

8.2.3 Console layout

The upper part of the console gives a block diagram of the 1401's logical elements. The B, A, LOGIC, STORAGE ADDRESS and OP blocks are illuminated to show the character value in that component. The LOGIC block also shows comparison status. The INSTRUCTION LENGTH block shows the length of the current instruction. The various labels light up in red if there is a fault in that component.

Each STAR register has an illuminated button. One of the buttons lights up to show which register is being displayed in STORAGE ADDRESS. Below these buttons are MANUAL ADDRESS dials, which can be used to enter an address.

To the right of the address dials is the Mode switch, which is very important since it controls the operating mode of the 1401 (e.g. Run, Single Cycle, Stop-on-Address and other debug modes).

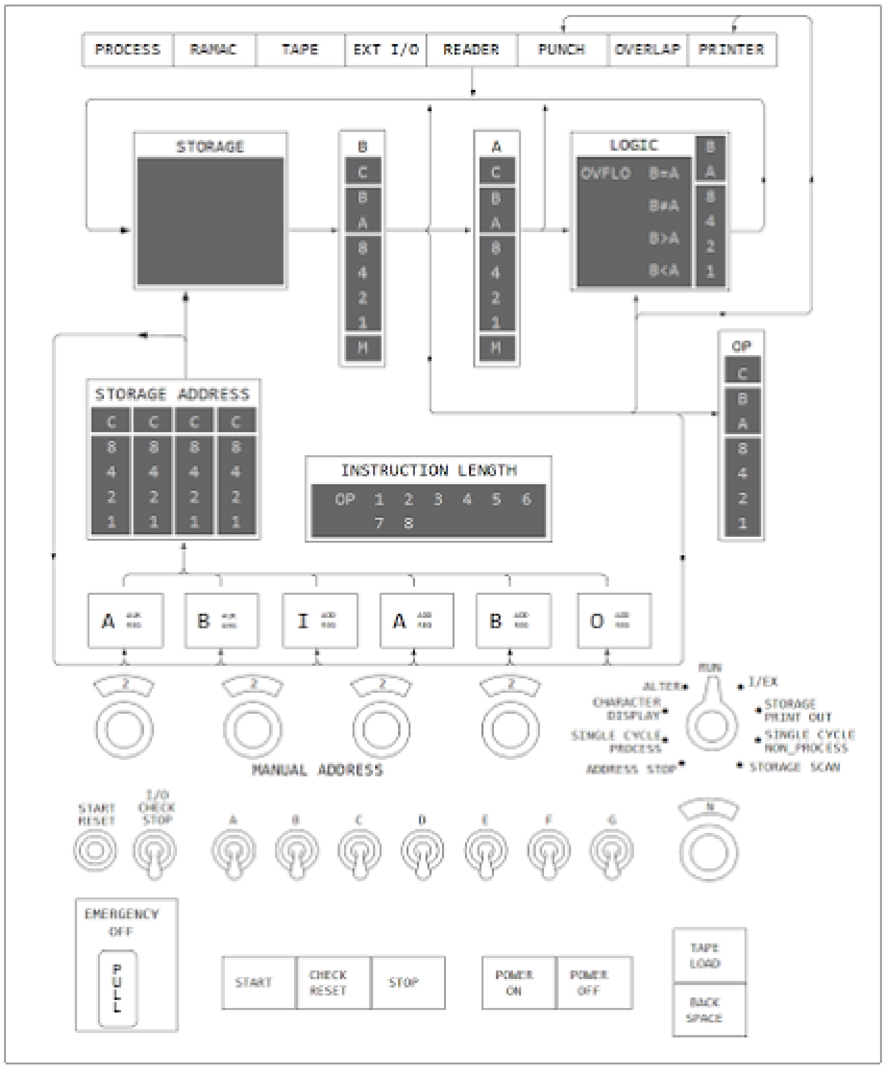


Figure 66: Diagram of the IBM 1401 Console.

The next line of switches consists of the START RESET button, which resets the system (except for address registers and core); the I/O CHECK STOP toggle, which enables I/O validity checks; the sense switches (switch A enables last card detection); and the tape mode dial.

At the bottom of the main console are the EMERGENCY OFF handle; the START button, which restarts execution; the CHECK RESET button, which lights if there is a fault, and is pressed to reset the fault; the STOP button, which stops execution; the POWER ON and POWER OFF buttons, which turn the entire system on and off; the TAPE LOAD button, which loads a tape; and the BACKSPACE button, which rewinds the tape one record.

Underneath the main console is the auxiliary console, which provides additional controls that are mostly not used in normal operation.

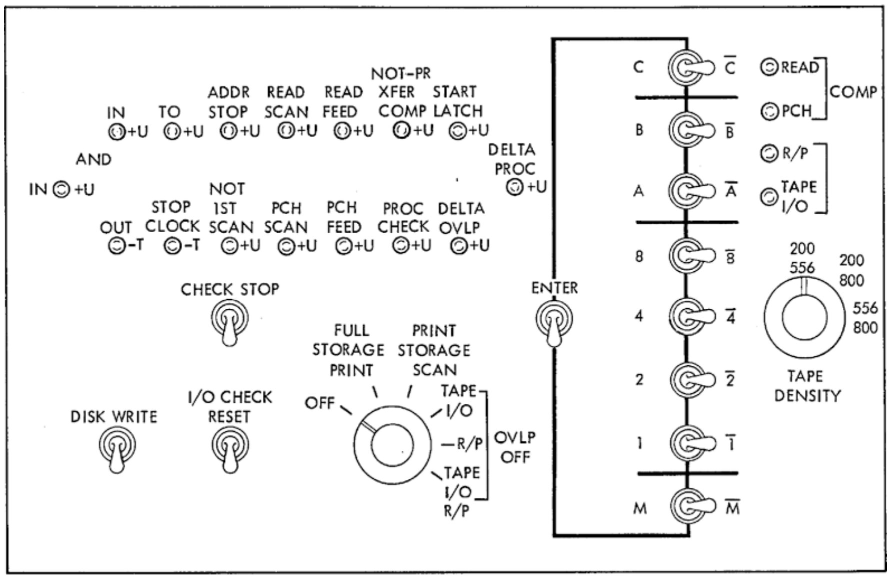


Figure 67: Diagram of the IBM 1401 Auxiliary Console[85]

The auxiliary console has multiple sync points for connection to test equipment (i.e., an oscilloscope)

- The CHECK STOP switch enables stopping on error conditions.
- The DISK WRITE switch enables and disables disk writes for testing.
- The I/O CHECK switch is for CE use and resets an I/O fault.
- The auxiliary mode switch selects print mode and controls overlap functions.
- The bit switches and ENTER switch are used to enter a character into memory. (Note that 0 is stored as 8 2 C.)
- The STERLING dial selects the desired shillings/pence encoding. (This optional dial is not shown but is above the auxiliary mode switch.)
- The TAPE DENSITY switch is used for tape I/O.

8.2.4 Debugging with the console

The console provides several operations that are useful for debugging. Registers and memory can be viewed and modified. Memory can be dumped to the printer. The most common operations are described below; the Reference Manual (pages 109-118) describes the console thoroughly.

8.2.4.1 Viewing Registers or Memory

To view a STAR register, press the register's button and its contents will be displayed in STORAGE ADDRESS.

[85] From the 1401 Reference Manual [RefMan]

To view a storage location, set the mode switch to CHARACTER DISPLAY. Set the desired address on the dials. Press START. The character will be displayed in the B register.

8.2.4.2 Modifying Registers or Memory

To modify an address register, set the mode switch to ALTER. Set the address on the dials. Press the desired register button. Press START.

To modify a memory location, set the mode switch to ALTER. Set the address on the dials. Set the character using the bit switches on the auxiliary console. Toggle ENTER on the auxiliary console. Note: make sure the parity is correct (odd), or the 1401 will check-stop when it uses the value.

To fill all memory with a character, set the mode switch to STORAGE SCAN. Enter the bit pattern on the auxiliary console switches. While holding ENTER up, press and hold START. Then release ENTER. Memory will be written until START is released. (This doesn't seem to be documented.)

8.2.4.3 Single-stepping

To step through instructions, set the mode switch to I/EX. Press START to read one instruction from storage, and press START again to execute the instruction.

To step through instructions one memory cycle at a time, set the mode switch to SINGLE CYCLE PROCESS and press START to complete one cycle.

8.2.4.4 Printing storage

To dump out 100 characters of storage to the printer, set the mode switch to STORAGE PRINT OUT. Enter the desired (thousands and hundreds) address on the dials. Press START.

To dump out all of storage to the printer, set the mode switch to STORAGE PRINT OUT. Set the auxiliary mode switch to FULL STORAGE PRINT. Press START.

8.3 Initial Program Load - Bootstrapping a Program

http://ibm-1401.info/InitProgLoad.html

http://ibm-1401.info/VansOverview.html#loading

[there's also a button on the console to load a program from tape, instead of punch cards]

8.4 Debug Techniques

Single-step
Stop-on-address, using the front-panel address switches
Trap-on-something by patching the binary
There are diags in the file drawer in the back room; what do they do?

9. Reference Material

9.1 Reference Documents

Many docs, both primary and secondary, have been collected at the CHM 1401 Restoration Team's web page, http://ibm-1401.info. Other docs have been scanned and collected at http://bitsavers.trailing-edge.com/pdf/ibm/140x.
This section identifies some of the key documents required to understand the 1401 system.

9.1.1 IBM Docs about the 1401 System

[RefMan] IBM Reference Manual / IBM 1401 Data Processing System (Form A24-1403-5, aka The Brown Book)

http://bitsavers.trailing-edge.com/pdf/ibm/140x/A24-1403-5_1401_Reference_Apr62.pdf
191 pages

Doc pg 164 shows how a program loader works

Doc pg 170 gives the punch-card code chart

[R25] IBM 1401 Data Processing System / Instruction Logic (Form R25-1496)
http://ibm-1401.info/1401InstructionLogic-R25-1496.pdf
106 pages
This doc gives detailed cycle-by-cycle descriptions of each of the instructions in the instruction set, indicating Instruction vs Execution phases, how address registers are used, when Word Marks will terminate an instruction, and what logical operations are performed.
The doc also contains detailed descriptions of the I/O instructions for the Card Reader/Punch and Printer.

http://ibm-1401.info/1401InstructionLogic1960.pdf gives what appears to be an earlier draft of the same material, without an IBM doc number. Written in 1960 by T. Mierswa, 104 pages.

[G24Flow] IBM 1401 Data Processing System / IBM1401 Data Flow (Form G24-1477-0) Second Revision
http://www.textfiles.com/bitsavers/pdf/ibm/140x/G24-1477-0_1401_dataFlow.pdf
68 pages, Second Edition, published May 1967
This doc describes data flow through the 1401, with a simplified block diagram of the data path as a starting point. It describes how instruction fetch cycles work, and cycle-by-cycle behavior for the various instruction types. [How is this one different from [R25], aka R25-1496?]
[There's an almost-identical copy at http://ibm-1401.info/G24-1477-0_1401_dataFlow-3.pdf; This looks like the first edition of the same doc, without a formal cover page.]

IBM 1401 Data Processing System / IBM 1401 DATA PROCESSING SYSTEM BULLETIN (Form G24-1477-**0**) (First Edition? Undated…)
There are Version 1 and Version 2 copies of this doc floating around

IBM General Information Manual; 1401 Data Processing System (Form D24-1401-1, February 1960)
http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/1401/D24-1401-1_General_Information_Manual_IBM_1401_Feb60.pdf
This doc is an introductory overview to the 1401. It's not a primary reference, but it does contain a useful summary of the instruction set, timing calculation, operator's consoles, tapes, etc.

[ILD] IBM 1401 Data Processing System / Instructional Logic Diagrams (ILD) Volume 4 (Form 56-398)
http://ibm-1401.info/ILDs_Aug62-Enhanced-TOC.pdf

84 pages
These pages have many hand-written notes in German (presumably from the German machine)

> [ILD2] An alternate copy of the ILD's can be found at:
> http://ibm-1401.info/ALDs-Australia/1_1%20Logic%20Diagrams%201401_40-28588%202151_788.pdf
> 156 pages
> This set contains numerous instruction-cycle timing diagrams, starting at PDF pg 93, and fewer scribbles and annotations.

[ALD logic diagrams (i.e. schematics for the 1401) are at this web site, but need more categorization:
http://ibm-1401.info/ALDs-VSnyder-Australia/ALDs-fromAustralia.html]

[ALD] This page gives an index of each page of the ALDs, with a link to individual PDF files.
http://ibm-1401.info/ALDs-VSnyder-Australia/List_1_7-7_7-logic.html

IBM Form F20-0208 General Information Manual, IBM 1401 Data Processing System, From Control Panel[86] to Stored Program
http://bitsavers.informatik.uni-stuttgart.de/1401/docs/F20-0208.pdf
42 pages
This doc is an introductory manual for students new to store-program computing; it's intended to introduce the concepts in the 1401 in preparation for a training course, and to help students understand how problems that had previously been solved on plug-board-controlled unit-record machines could be implemented in the stored-program world.

IBM Form A24-1420-0 Reference Manual 1401 Data Processing System, Original Equipment Manufacturers' Information
1401OEM-Info.pdf
Description of cabling, installation, and operator's panels with definition for all the buttons and lights.

[FMM] IBM Field Maintenance Manual (Doc number 225-6487-3)
http://ibm-1401.info/PPierce-ibm-225-6487-3.pdf
See page 21 (pdf pg 23) for info on margining power supplies and clocks for reliability checks.
See pdf pg 19 for cabling
This doc contains many timing waveforms, guidance for repair, instructions for Customer Engineer panels and test points, as well as advice on how to keep your oscilloscope in good operating condition.
The doc contains detailed flow diagrams for various machine cycles.

Single-page map of logical function to physical card gates and ALD pages
http://ibm-1401.info/1401-Loc-ALD-.jpg
This single page shows the assignment of logical functions to the chassis locations that hold card gates or functional units such as power supplies in a 1401 processor
[try to identify the source doc…]

> Date: Thu, 04 Jun 2015 15:34:02 -0700
> From: ed@ed-thelen.org

[86] The "Control Panel" in the title refers to unit-record plug-boards, not the knobs and buttons on the front of a 1401!

To: Guy Fedorkow <guy.fedorkow@gmail.com>, Robert Garner robgarn@mac.com

I have added a document called

"Instruction Logic"

104 pages, written in 1960 by T. Mierswa

which may be useful.

http://ibm-1401.info/index.html#1401-Processor

http://ibm-1401.info/1401InstructionLogic1960.pdf

9.1.2 Programming the 1401

http://bitsavers.trailing-edge.com/pdf/ibm/140x/A24-3067-2_1401_1460_System_Operation_Reference_Manual_Sep66.pdf

http://www.cap-lore.com/Hardware/1401Manual.html (a 1401 simulator? Where did this come from?)

There's a short list of programming reference guides at the end.

9.1.3 Peripherals

Add the TAU document

Tape Controller / Tape Adapter Unit (TAU)

Form 223-6847 (December 1960)

[IBM1402] IBM Customer Service Index; IBM 1402 Reader Punch (Form 229-4016-1; aka Field Engineering Manual)

http://ibm-1401.info/IBM-229-4016-1-IBM1402FE-PP-150.pdf [this copy is cleaner]

84 pages

http://ibm-1401.info/1402ReaderPunchServiceIndex-229-4016-1.pdf [this appears to be a duplicate]

82 pages

Copyright 1963, 1965

This doc contains everything about the Reader/Punch, including adjustments for timing, cabling, and many mechanical details. The doc is the "ILD" of the 1402 – explanations of how the drive train, power supplies, and control logic works, plus trouble-shooting guides and adjustments for various components.

Some of the relay logic is described in terms of And-Or logic diagrams in this doc.

[CE-Ref1402] Customer Engineering Instruction-Reference; 1402 Card Read-Punch

https://archive.org/details/bitsavers_ibm140xce2PunchCEManual1962_12788025

70 pages

Copyright 1960, 1961, 1962

This doc has been OCR'd, so the text is searchable

[Wiring1402] http://ibm-1401.info/1402CardReadPunchWiringDiagramSerial-20611.pdf

58 pages

Wiring diagrams and schematics

9.1.4 Docs about 1401 Technology

[SMS] IBM Customer Engineering Manual of Instruction: Standard Modular System Component Circuits (Form 223-6875-1)
http://ibm-1401.info/IBM-StandardModularSystem-Neff7.pdf
236 pages.
All about SMS cards: This doc describes the logic family used in the 1401, plus:

- Packaging conventions (e.g., types of card gates)
- Design Flow [pdf pg 12]
- How to Read an ALD
- Detailed descriptions of loading rules, "dot" gates (i.e. wire-and, wire-or)
- Detailed descriptions of many card types, arranged alphabetically by card type (e.g. "AR" or "CW")
- Index of card types and functions (in the middle of the doc; pdf pg 182)
- Instructions on repair of cards (hint: don't heat the PCB too much with your soldering iron, lest traces separate from the pcb material!)
- Power supply conventions
- Transistor types
- Glossary of terms

The doc is scanned, but not OCR'd, so it's not searchable.

[TCC] IBM Customer Engineering Manual of Instruction: Transistor Component Circuits

Form 223-6889

http://ibm-1401.info/**Form223-6889**-TransistorComponentCircuits.pdf

151 pages

Document provided by Harlan C. Snyder

This doc covers theory of operation for many kinds of SMS circuits. Read here if you want to learn how an edge-triggered flip-flop works…

PDF Pages 105 to 124 describe Complementary Diode Transistor Logic, the family used in the 1401.

The doc is scanned but not OCR'd

Ken Shirriff has compiled a database of many of the SMS cards used in the 1401 and other technology of the era:
http://www.righto.com/2015/03/a-database-of-sms-cards-technology.html
http://files.righto.com/sms/

Single page on how to read gate symbols on an ALD
http://ibm-1401.info/JVG-ALDs-001.pdf
See also http://ibm-1401.info/RWilliams-Quiz-ALD-Hints-.jpg from Ron Williams

Page pointing to diagnostic manuals (some links don't work)
http://www.piercefuller.com/oldibm-shadow/diagman.html

BitSavers Sites
This doc lists all the IBM docs that are available on BitSavers (as of xx?) : http://ibm-1401.info/docss.html
Paul Pierce's collection: http://www.textfiles.com/bitsavers/1401/1401-docs.html
Al Kossow's collection: http://www.textfiles.com/bitsavers/pdf/ibm/140x/

9.2 Glossary

ILD
ALD

10. Appendix

10.1 Ken's List of 1401 Optional Features

[Compiled Fri, 11 Sep 2015, revised June 2016]

"Installation" in the list below indicates whether the option is installed on either or both of the CHM machines.

- DE: from Hamm, Germany (built 1964)
- CT: from Darien, Connecticut (built 1961)

10.1.1 1401 Optional Features

10.1.1.1 Expanded storage

The 1406 Storage Unit adds up to an additional 12K in 4K increments, providing a total of 16K of System Storage.
Installation: DE: Y, CT: Y
Instructions: Modify Address
Machine Feature Index code (MFI): EX/1K/2K/4K/8K/12K/1M/2M/4M/8M/12M/16M

10.1.1.2 Advanced programming (#1060)

Adds indexing and new instructions.
Installation: DE: Y, CT: Y
Instructions: Store A-Address Register (SAR) , Store B-Address Register (SBR), Move Record (MCM). Indexed addresses.
Price: $108 monthly, $4040 purchase.
Machine Feature Index code (MFI): APF/IN/MR
Cards: 105 in 02b6

10.1.1.3 Expanded print edit (#3835)

Adds asterisk protection, floating dollar, decimal control, and sign control left to editing.
Installation: Standard on Model C
Instructions: Additional characters in format string for Move Characters and Edit (MCE)
Price: $20 monthly, $795 purchase.

Machine Feature Index code (MFI): EE
Cards: 13 in 01b6

10.1.1.4 High-low-equal (#4575)

Test for high/low/equal after compare.
Installation: DE: Y, CT: Y
Instructions: Branch if High, Low, or Equal Compare (BH, BL, BE). Branch if Unequal (BU) is standard.
Identification: Comparison lights on console
Price: $76 monthly, $2865 purchase.
Machine Feature Index code (MFI): HL
Cards: 37 in 02a8

10.1.1.5 Multiply-divide (#5275)

Provides multiplication and division.
Installation: DE: Y, CT: Y
Instructions: Multiply (M), Divide (D)
Identification: Aux Star buttons on console
Price: $333 monthly, $11950 purchase.
Machine Feature Index code (MFI): MD
Cards: 100 in 02b7, 83 in 02a7

10.1.1.6 Serial I/O (#7080)

Allows a serial I/O peripheral to be attached.
Installation: DE: Y, CT: Y
Instructions: I/O ops
Price: $101 monthly, $3845 purchase.
Machine Feature Index code (MFI): IO/IOA/IOC
Cards: 76 in 02a2

10.1.1.7 Print storage (#5585)

Adds a buffer to hold a print line, increasing available process time.
Installation: DE: Y, CT: Y
Instructions: Branch on Carry Busy, Branch on Printer Busy
Identification: Core module in gate 01a4
Price: $386 monthly, $12890 purchase.
Machine Feature Index code (MFI): PF
Cards: 119 in 01a6, 85 in 01a5, 59 in 01a4

10.1.1.8 Column Binary (#1990)

Processes column-binary-encoded cards and tapes.
Installation: DE: N, CT: Y
Instructions: Read Column Binary (RCB), Punch Column Binary (PCB), Move and Binary Decode (MBD), Move and Binary Code (MBC), Write Binary Tape (WTB), Read Binary Tape (RTB)
Price: $101 monthly, $3685 purchase.
Machine Feature Index code (MFI): CH
Cards: 46 in 02b8

(See http://ed-thelen.org/comp-hist/JVG-ChineseBinary.doc for anecdotal history by John Van Gardner, and http://ibm-1401.info/PokoskiJ-mods-May-59.pdf for a note from Fran Underwood)

10.1.1.9 Bit Test (#1470)

Test any bit in a character. Requires Column Binary.
Installation: DE: N, CT: Y
Instructions: Branch if Bit Equal (BBE). d modifier specifies bits
Price: $20 monthly, $848 purchase.
Machine Feature Index code (MFI): BT
Cards: 2 in 01b6, 1 in 01b1

10.1.1.10 PFR control (#5895)

Required for Punch Fead Read on some 1402 models.
Installation: DE: N, CT: N
Price: $56 monthly, $2095 purchase.
Machine Feature Index code (MFI): RP
Cards: 10 in 01b4

10.1.1.11 Processing overlap (#5730)

Overlap computation and processing.
Installation: DE: Y, CT: N
Instructions: Overlap On/Off/Reset (SS), Branch if Indicator On: Read Busy, Punch Busy, Tape or I/O Busy (BIN)
Identification: Overlap light on console
Price: $255 monthly, $15380 purchase.
Machine Feature Index code (MFI): OV/OVR/OVRP/OVT
Cards: 136 in 02b1, 13 in 02b2

10.1.1.12 Sterling currency

Supports pound/shilling/pence arithmetic.
Installation: DE: Y (disabled), CT: N
Instructions: Probably d-character.
Identification: Mode switch on aux console
Machine Feature Index code (MFI): SBA/SEC/SEE/SMD
Cards: 138 in 02b7, 134 in 02b8, 128 in 01b3, 108 in 02a7

10.1.1.13 Sense switches (#7600)

Adds six sense switches (B-G) to the console. Switch A is standard.
Installation: Standard on Model C
Instructions: Branch if Indicator On (BIN)
Identification: Switches on console
Price: $15 monthly, $582 purchase.
Machine Feature Index code (MFI): SS
Cards: 4 in 01b6

10.1.1.14 Read punch release (#6040)

Allows card read/punch to start while processing continues.
Installation: Standard on Model C
Instructions: Start Read Feed (SRF), Start Punch Feed (SPF)
Price: $25 monthly, $1005 purchase.
Machine Feature Index code (MFI): FP
Cards: 8 in 01b4

10.1.1.15 Print control (#5539)

Adds 32 print positions to the basic 100, for the 1404 printer.
Installation: Standard on Model C
Price: $61 monthly, $2510 purchase.
Machine Feature Index code (MFI): 132
Cards: 15 in 01b5

10.1.1.16 Print control, add'l (#5540)

Adds 32 print positions to the basic 100, for the 1403 printer.
Installation: Standard on Model C
Price: $61 monthly, $2510 purchase.
Machine Feature Index code (MFI): 132

10.1.1.17 Printer (1404) adapter (#5563)

To attach a 1404 printer.
Installation: DE: ?, CT: ?
Price: $25 monthly, $1410 purchase.

10.1.1.18 Space suppression (#7246)

Prevents the printer from advancing to the next line after printing.
Installation: DE: Y, CT: Y
Instructions: d-character S with Write Line. Added by Ron Williams.
Price: $78SUC monthly, $63 purchase.
Machine Feature Index code (MFI): XSS
Cards: 2 in 01a6

10.1.1.19 Compressed tape (#2010)

Read tape records written with zero elimination by a 7070/7074.
Installation: DE: N, CT: N
Instructions: Read Compressed Tape (MU), Move and Insert Zeros (MIZ)
Price: $35 monthly, $1330 purchase.
Machine Feature Index code (MFI): CW
Cards: 21 in 02b6

10.1.1.20 Disk storage drive adapter (#3339)

Allows a 1311 Model 001 Disk Storage Drive to be attached.
Installation: DE: N, CT: N
Price: $101 monthly, $4925 purchase.
Machine Feature Index code (MFI): RAM/1311
Cards: 60 in 02a8

10.1.1.21 Direct data channel (#3271)

Permits two 1401s to transfer data between systems.
Installation: DE: N?, CT: N?
Instructions: Signal Control Instructions (SS), Branch Instructions (BIN), Move Instructions (MU, LU)
Price: no charge monthly, no charge purchase.

10.1.1.22 Numerical print control (#5380)

Required for Numerical Print on some 1403 models.
Installation: DE: Y, CT: N
Price: $50 monthly, $2045 purchase.

Machine Feature Index code (MFI): NU
Cards: 51 in 01b5, 10 in 01a6

10.1.1.23 Read-compare adapter (#5991)

Required for Read Compare with a 1404.
Installation: DE: N, CT: N
Price: $76 monthly, $4365 purchase.
Machine Feature Index code (MFI): CFC
Cards: 4 in 01b4, 4 in 01a7. 1 in 01b1, 1 in 01a5

10.1.1.24 Card feed (1404)

Installation: DE: Y, CT: N
Machine Feature Index code (MFI): CF
Cards: 1 in 01a5, 1 in 01b1

10.1.1.25 Selective tape listing control (#6412)

Required for Selective Tape Listing on some 1403 models.
Installation: DE: Y, CT: N
Price: $86 monthly, $4415 purchase.
Machine Feature Index code (MFI): STL
Cards: 2 in 01b1

10.1.1.26 Dual speed carriage

Printer skips lines faster after first eight lines.
Installation: DE: Y, CT: Y
Machine Feature Index code (MFI): DS
Cards: 25 in 01a5

10.1.1.27 Adapter, 51-column feed (#1013)

Installation: DE: N, CT: N
Machine Feature Index code (MFI): FCF
Cards: 7 in 01a8

10.1.1.28 Console auxiliary adapter (#2263)

For connection of 1409 console.
Installation: DE: N, CT: N

10.1.1.29 Console inquiry station adapter (#2272)

For connection of 1407 console.
Installation: DE: N, CT: N
Machine Feature Index code (MFI): INQ
Cards: 20 in 02a8

10.1.1.30 Attachment circuitry

Circuitry for I/O operations
Installation: DE: Y, CT: Y
Instructions: Read (R), Write (W, M), Branch on EOF (d=K), Branch on tape error (d=L), unit select (%xx)
Machine Feature Index code (MFI): CM
Cards: 53 in 02a2

10.1.1.31 Group mark 12-5-8 12-7-8 (M94618)

Allows different card encoding for group mark read and punch, selected by switches in the card reader. See 90.10.12.2.
Installation: DE: N, CT: Y
Identification: Switches in 1402.
Cards: gateÂ 01B7: A14-B19

10.1.1.32 Reader/punch code 8-2 and A-bit compatibility (898148)

Provides A-bit compatibility between 1401 and 1410 systems. It reads and punches an A-bit as 8-2 instead of 10. (36.23.41.2)
Installation: DE: N, CT: Y
Machine Feature Index code (MFI): ABIT
Cards: 1 in 01b7

10.1.2 Tape

Supports 729 Models II, VI, V, VI and 7330 Magnetic Tape Units.
Installation: DE: Y, CT: Y
Instructions: Special tape operations, usually with %Ux as the A address.
Machine Feature Index code (MFI): MT
Cards: 74 in 02b2, 127 in 02b3, 91 in 02b4, 65 in 02a1

10.1.2.1 Any tape drive

Machine Feature Index code (MFI): M0
Installation: DE: Y, CT: Y

10.1.2.2 High-speed tape drive?

Machine Feature Index code (MFI): M1
Installation: DE: Y, CT: N

10.1.2.3 729 II or V (low speed)

Machine Feature Index code (MFI): M2
Installation: DE: Y, CT: Y

10.1.2.4 7330

Machine Feature Index code (MFI): M3
Installation: DE: N, CT: Y

10.1.2.5 729 IV or VI (high speed)

Machine Feature Index code (MFI): M4
Installation: DE: N, CT: N

10.1.2.6 High speed + 556 density?

Machine Feature Index code (MFI): M5
Installation: DE: Y, CT: N

10.1.2.7 Any 729?

Machine Feature Index code (MFI): M6,729
Installation: DE: Y, CT: Y

10.1.2.8 800 CPI

Machine Feature Index code (MFI): M8
Installation: DE: Y, CT: N

10.1.2.9 Tape Intermix

Machine Feature Index code (MFI):
Installation: DE: Y?, CT: ?

10.1.3 1402 Card Reader / Punch

10.1.3.1 Early card read (#3550)

Allows card reading mechanism to engage sooner. The read clutch has three engagement points instead of one.
Installation: DE: Y, CT: Y
Identification: Read Sync button in 1402
Price: $10 monthly, $241 purchase.

10.1.3.2 Interchangeable 51-column read feed (#4150)

Allows the 1402 to read 51 column cards
Installation: DE: ?, CT: ?

10.1.3.3 Punch Feed Read (PFR) (#5890)

Allows a card to be read and then punched.
Installation: DE: N, CT: N
Instructions: Read-Punch Feed (RF), Read-Punch Feed and Write (WRF)
Identification: Additional read station and relays in 1402
Price: $28 monthly, $1045 purchase.

10.1.4 1403 Printer

10.1.4.1 Auxiliary ribbon feeding (#1376)

Supports polyester film ribbon on the printer.
Installation: DE: ?, CT: ?
Price: $88 monthly, $2000 purchase.

10.1.4.2 Interchangeable chain cartridge adapter (#4740)

Allows 1416 printer chain cartridges to be changed.
Installation: DE: Y, CT: Y
Price: $88 monthly, $2030 purchase.

10.1.4.3 Selective Tape Listing (#6411)

Allows printer to print on multiple adding-machine tapes.
Installation: DE: N, CT: N
Price: $225 monthly, $5260 purchase.

10.1.4.4 Numerical print control (#5381)

Faster printing with a numeric chain.
Installation: DE: ?, CT: ?
Price: $437 monthly, $30520 purchase.

10.1.5 1404 Printer

10.1.5.1 Read compare (#5990)

Read data from punched cards in 1404 printer feed.
Installation: DE: n/a, CT: n/a
Price: $190 monthly, $10620 purchase.

10.1.5.2 Interchangeable chain cartridge adapter (#4740)

Allows 1416 printer chain cartridges to be changed.
Installation: DE: n/a, CT: n/a
Price: $88 monthly, $2030 purchase.

10.1.6 References

Special Feature Instructions: Describes instructions associated with special features.
IBM Sales Manual (1979): Extensive list of machines, options, and prices.
Optional Features : Describes optional feature implementation for CEs.
1401 Model Info from Service Consultant Guides (1976): Options and prices for 1401.
General Information Manual: Describes some optional features.
Custom Features: Lists many custom features (Request for Price Quotation, RPQ). These features are not listed above and it is unknown if any are installed (apart from the Group Mark feature).
Technical Handbook of System Engineering: Lists 1401 options and peripherals.

10.2 Carl’s summary of Logic Families

Describe basic logic gates / SMS technology
See "Standard modular system" (ref #xx)

From Carl Claunch’s logic family decoder doc: http://ibm-1401.info/Terminology.html

IBM'S OFTEN NONSTANDARD TERMINOLOGY AND PRACTISES FOR FLIPFLOP AND LATCH CIRCUITs

by Carl Claunch < Carl . Claunch @ gartner . com >

This section will serve as a 'Rosetta Stone' for readers familiar with modern terminology and digital design practices, helping them interpret the names and practices used by IBM, which can be confusing and thus tough to follow when reading documentation. Many nearly universal design principles that are used today are missing in the 1401 and other mainframes of the 1950s, 1960s and even into the 1970s.

One has to remember that the circuits, logic components and best practices for the entire industry were being pioneered back when these machines were designed. Many times IBM employees were the inventors of a principle but even in those cases when others first came up with an approach, industry terminology hadn't settled to a commonly accepted choice; the same idea would be described different ways depending on the preference of each company or academic researcher. Today, however, many of these terms are universally agreed, well defined concepts, albeit usually different from the original choices made by the inventors and by IBM engineers.

As an example, an IBM engineer invented the logic circuit family now universally named ECL (Emitter Coupled Logic), but the inventor called it current mode or current steering and that name continued to be used by IBM mainframe design engineers. The components used widely as flipflops in the 1401 and throughout the later 360 and 370 systems, were mostly called a trigger by

IBM. The IBM trigger circuit does not correspond directly to any of the standard flipflop types that are covered in today's digital logic books and courses. The trigger circuit is flexible, able to be used in more than one way, each usage has similarity to common flipflop and latch types.

A flipflop or a latch is a memory device, used to hold some state or condition in a digital system. People often differentiate these by whether they are edge or level sensitive, allowing input signal changes to pass though and change the output state either just at the edge or during the entire time an enabling signal is active. A latch is level sensitive, in that modern usage, while a flip flop is edge sensitive. IBM, on the other hand, uses the term latch to refer to a combination of AND and OR gate components that act as a modern day latch, other times building a latch (as we would recognize one) using their trigger component.

Circuits that depend on a memory are generally called sequential logic, often implementing a 'state machine' which moves between defined states based on the memory of its current state plus some input conditions that determine the next sequential state. Modern designers consider it a best practice to control the change of state in a flip flop or other memory in sequential circuits using the rising or falling edge of a special signal, a clock. This global clock is distributed to many gates simultaneously such that they all change their state together, in synchronization, at the clock edge. Flipflop components you would find today have a clock input that accepts this signal. The clocked flip flop holds its output state steady until the next clock edge, when it switches to a new state based on the then current input values.

The input conditions that exist at the time of the clock edge determine which state the output will take on. Proper operation often requires minimum periods of time that inputs must be stable at their intended values (setup time) prior to the clock edge and as well minimum times may exist that the input values must remain steady (hold time) after the edge has occured. The mandated duration to keep inputs steady is very short compared to the duration of a clock cycle, but if these minimum times are not met, the logic gate can fail to operate as intended or enter pathological states (e.g. metastable states).

Today many would say that a latch is a level sensitive device rather than edge sensitive one. That is, changes in the input can change the latch output state at any time the enabling signals attain an activating level. If a clock signal is hooked to a latch, it allows the latch to transparently pass input signal changes to the output for one of the two clock levels. During half the clock cycle, while the clock signal was in one of the two binary states, the latch would be changing its output continuously based on the other input signals, while in the other half of a clock cycle the latch freezes the output state that existed just as the clock level was reached. It acts as a memory for half a cycle and a bit of combinatorial logic changing in real time during the other half of a cycle. Contrast this to a clocked flipflop whose output state is frozen at all times except at the instant of the chosen clock edge (either when the clock is raising from 0 to 1 or the falling edge when it is dropping from 1 to 0).

If changes in input conditions can alter output states transparently, it has the same timing (race) hazards of combinatorial logic. If a designer intended that both 'set' and 'reset' signals were to be inputs causing an appropriate latch to toggle to its opposite state, but due to slight timing variations, one of the signals arrived first, the result could be the opposite state of what was intended since the level sensitive gate would first act on the single input and then change again when the tardy signal arrived. A change in the output state might make its way through other gates that affect input conditions, which could lead to unstable, glitchy or unintended operation. The practice of changing sequential states at a clock edge goes a long way to eliminate these risks. This practice of using a global clock to synchronously change states in sequential logic is essentially missing entirely in the IBM mainframes. In the era in which these systems were designed, the practice was not well established as it is today. Further, the cost of individual gates were very high, and the additional transistors and other parts needed to add clock synchronization would have driven up the size, cost and power demands of the system.

Latch and flipflops are now referred to with names like D, SR and JK, plus variants and extensions of these. Each type can be a level sensitive latch or, with additional components inside, operate as a clocked, edge sensitive flipflop. D (Data) type is the simplest, it sets its output condition to the state of its single input signal. T type (Toggle) devices will alternate output states when the T input is active. SR types (Set-Reset) (SR) have a set and a reset input, which operate as you would expect from the names of the inputs, but the behavior may not be defined if both S and R inputs are simultaneously active. J-K types are like SR, the J corresponding to a set and the K corresponding to reset but if both J and K are active at the same time, it toggles just like a T type. During the vacuum tube era, flip flops and latches were called triggers, but that is relatively obsolete terminology today. IBM engineers retained the name trigger for the circuit type even as it evolved to transistor and then integrated circuit designs. The trigger is IBMs building block for flipflops and latches. Their trigger circuit can be used as a level-sensitive or an edge sensitive device, and it is most analogous to a JK type.

IBM refers to the edge sensitive operation of the trigger as "AC" mode and used a capacitor signal on the input line to indicate this on SLT era logic diagrams but not on the 1401 generation diagrams. The letter assigned to the input on the component on 1401 diagrams indicates its role - the G input is the DC mode that is level sensitive to set or reset the trigger. The A and C inputs are the edge trigger and gate for AC mode operation. However, this AC mode use is not a clocked JK, because the J and the K sides of the trigger are individually edge sensitive. In a clocked JK flipflop of today, one clock is used for the edge that activates the function requested by the J and K inputs. If J is on, K is off, then the flipflop is set on the clock edge. If the J and K are both on, the flipflop toggles. With the IBM trigger circuit, an edge on one line drives the output change based on the gating input signal, but as there are two different edge inputs, there are essentially two independent sides for J and K.

The reset (K) gating input may be on but it would not reset the output of an AC mode trigger if the reset triggering input did not present a rising edge. At that same instant, if the set (J) gating input is on and its set trigger input rising edge has arrived, that would be setting the output state of the flipflop. With a single clock on a JK flipflop, having both J and K inputs on would cause a toggle when the clock arrived. On the IBM AC mode trigger, if both sides are not triggered simultaneously, only the triggered side would affect the gate, either setting it or resetting it, ignoring the untriggered input. This behavior is not one that would be familiar to a modern digital designer and could lead you to misunderstand the behavior of a 1401 or 360 design.

When IBM was putting a latch into a circuit, the engineer could use either a pair of AND/OR gates or the trigger circuit in DC mode. For the 1401, the engineers preferred to use the pair of combinatorial gates whereas in the 1130, the preference was more to the use of the trigger component, and in 360 it varied by model and by the portion of the machine. Thus, these seemed to be stylistic predilections of individual designers. There are very valid reasons to make the choice on a case by case basis, especially with this higher circuit density of SLT where one may have spare AND/OR gates on a card already being used, while use of a trigger could mean an incremental card is needed; obviously, the opposite situation could equally pertain. With SMS, this is less likely to occur because the density is much closer to a one to one ratio of logic function and card than with SLT.

The trigger circuits were not exactly the same across the various logic families in SMS nor between SMS and SLT, adding another small complication when reading logic diagrams of those eras. The current mode families of SMS are more generally operating as edge sensitive devices, than do the CTDL or SLT trigger components.

The term binary trigger is used with current mode trigger circuits for essentially an edge sensitive T latch. DC mode trigger was used to mean a gate that sets if both inputs are on, resets if both inputs are off, but does not change state if the inputs are mixed. This is not analogous to any standard flipflop type today, but would provide some protection from race hazards where one input lagged the other; during the short interval where the inputs were mixed, the binary trigger would do nothing, but when the pair of inputs were both present it would set or reset as selected. DC mode Slit P is a term used with current mode triggers for an SR latch with a reset line added. Bipolar trigger is the term IBM used for a D latch, the where an edge on one input would cause the latch to take on the state present on the other input at that instant. It does not transparently pass the input to the output at any time, thus is not exactly like a D latch or flipflop.

CTDL family triggers are gates with individually 'clocked' J and K inputs, each side with an independent data input and a gating (edge sensitive or AC) input. The CTDL trigger also has asynchronous reset and set lines (DC set and reset). When the IBM SLT trigger is used with both its set and reset inputs tied together, it is acting like a T flipflop and would be referred to by IBM as a binary trigger. If the DC mode set and reset lines are used, they operate like an SR flipflop and are level sensitive. They can also be thought of as the asynchronous set and asynchronous reset that can be added to the basic flipflop/latch types. More often than not, the trigger circuit is used as a level sensitive SR latch, using just the DC model (level sensitive) inputs. As you can see, IBM tended to use DC and AC rather than level sensitive and edge sensitive terminology.

10.3 Revision History

Date	Who	Notes
Mar 2015	Guy	Outline reviewed with Carl Claunch
May 27, 2015		First Draft posted to 1401 web site
May 31, 2015	Guy	Corrections to First Draft

Jun 3, 2015	Guy	Meeting at CHM with Carl Claunch, George Ahearn and other members of the restoration team.