# Mechanical Turing Machine in Wood

# Explained

By

Richard J. Ridel

# Why did I build this?

By Richard J. Ridel

Well, I have always been interested in computers. I built my first one in 1976 using an Intel 8008 microprocessor. It worked. It was an 8 bit processor with a 1 meg clock and 4k memory. I didn't do much with it and subsequently realized it was more fun building it than using it. I still have the 4k memory board and the 8008 chip. The rest got dismantled and tossed over the years. I went to the movies to see "The Imitation Game." Loved the movie and looked up Allen Turing on the internet. I found his 1936 paper on computable numbers and got hooked. Another search on the internet yielded a small list of machines built by others. Several had electronic controllers controlling the mechanics. I can hear Mr. Turing say, "A computer running a computer. Whaaaat?" And a few came closer using only motors and relays and electrical widgets. Closer. But I was looking for something completely mechanical. Ergo, the challenge.

I'm going to make a huge assumption: If you are reading this then I'm sure you have read all about Alan Turing and his theoretical computing machines. Therefore, I do not need to go over all the specifications and requirements of the machine. I will, however, offer a few departures, though I am sure you can figure them out yourself. The data tape cannot be infinitely long. The character set cannot be infinitely large and the configuration table containing states cannot be infinite. Infinite is theoretical not practical.

The mechanical and size limitations began to dictate how many characters and states I would build into the machine. So I went on an internet search to see what others had determined. Why reinvent the wheel. I found the work of Steven Wolfram and his challenge. He offered $25,000 to anyone who could prove what the minimum data elements and states are to perform any calculation. In May 2007 Alex Smith, a 20 year old undergraduate in England, submitted a 40 page proof and won the prize. He proved that only 3 data elements and 2 states were needed. I have programming experience: Basic, C++, C, Assembly and Machine. So I downloaded a Turing Machine simulator to my iPad and programmed a few calculations. Then I restricted myself to the minimums (minima?) stated above and discovered I'm not that creative a programmer as I thought I was. But I found that 3 data elements and 3 states seemed to work and decided to implement those numbers.

I decided to make this out of wood, having woodworking skills and tools, with no electrical parts. I wanted an antiquish (not a real word) look with rounded corners and turnings and whatnots. I used pine, oak, maple and mahogany, with stain here and there. I

don't have mechanical training but I learn quickly, especially during trial and error work. I will talk about force and throw while explaining the various parts of the machine. Ah yes, trial and error. Do not think that I sat down, did some thinking and built this machine just like that. I have a box full of parts that did not work. Most of the machine was reengineered and rebuilt. The configuration table mechanism itself went through three iterations, one of them having over 100 parts. They now help populate the junk box.

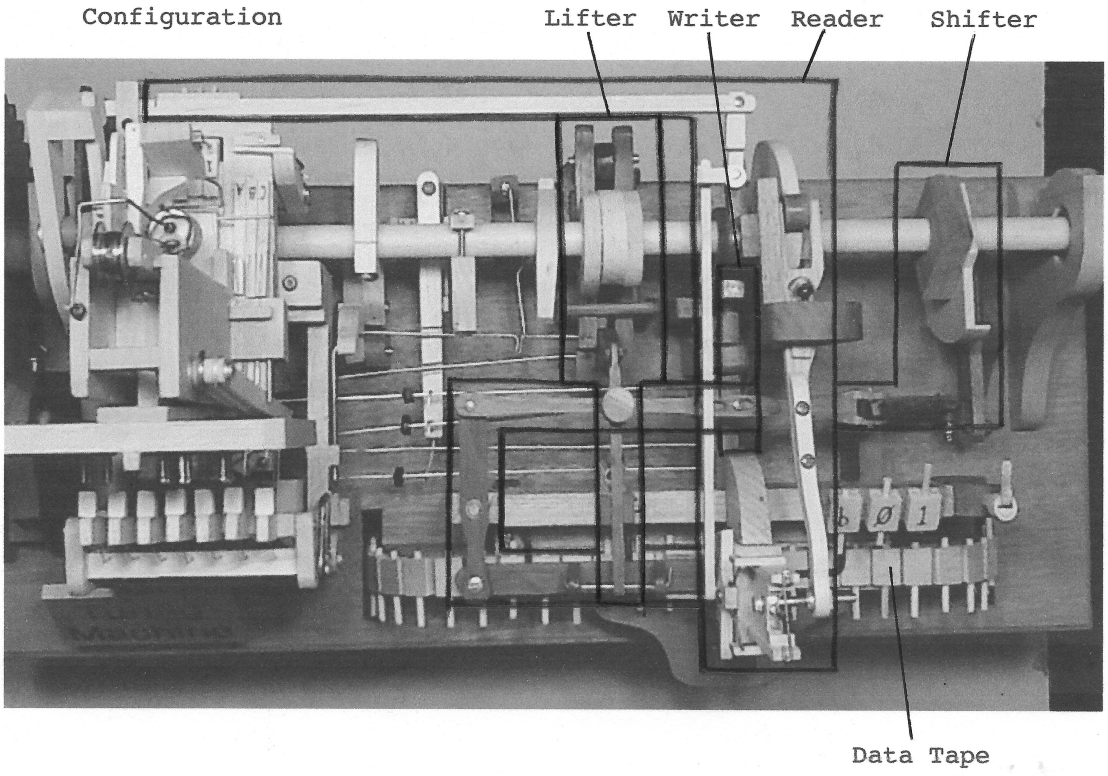
Okay, enough. Let's get to the good stuff.

There are four steps in one cycle of the machine. Later in this document I'll present a 360° chart showing all the action in one cycle.

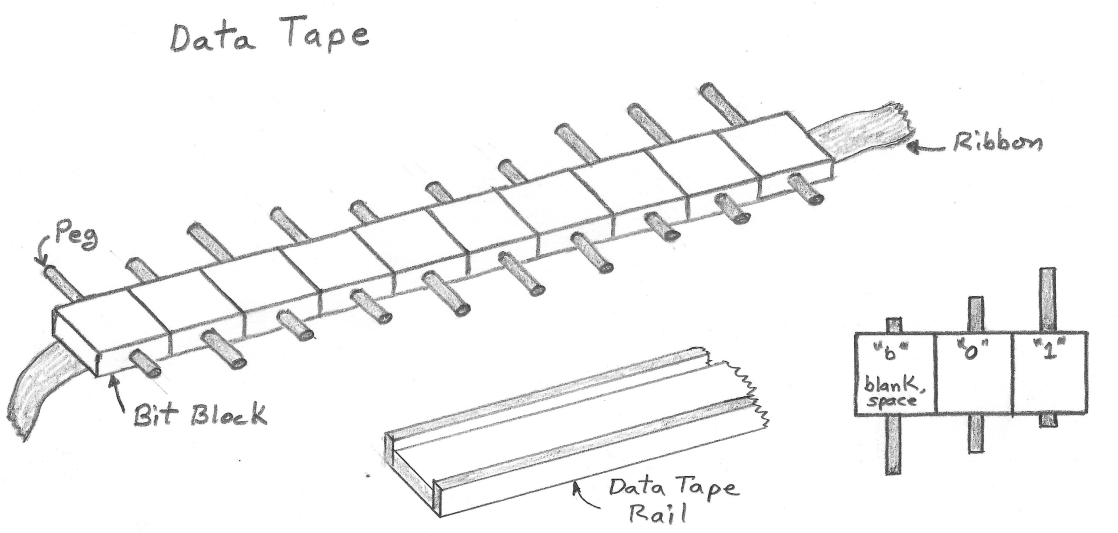
1. Read the Data Tape
2. Read the Configuration Table
3. Write to the Data Tape
4. Move the Data Tape

It is important to mention that an initial start condition must exist prior to executing the four steps above. I must mechanically set the Data Tape to a bit before the actual data and reset the configuration table to the start state, which is state 'A'.

Section Diagraam



The section diagram above shows the Mover as two parts, a Shifter and a Lifter. I will start with the Data Tape first, then the Mover, the Writer and the Reader. And finally the Configuration Mechanism.

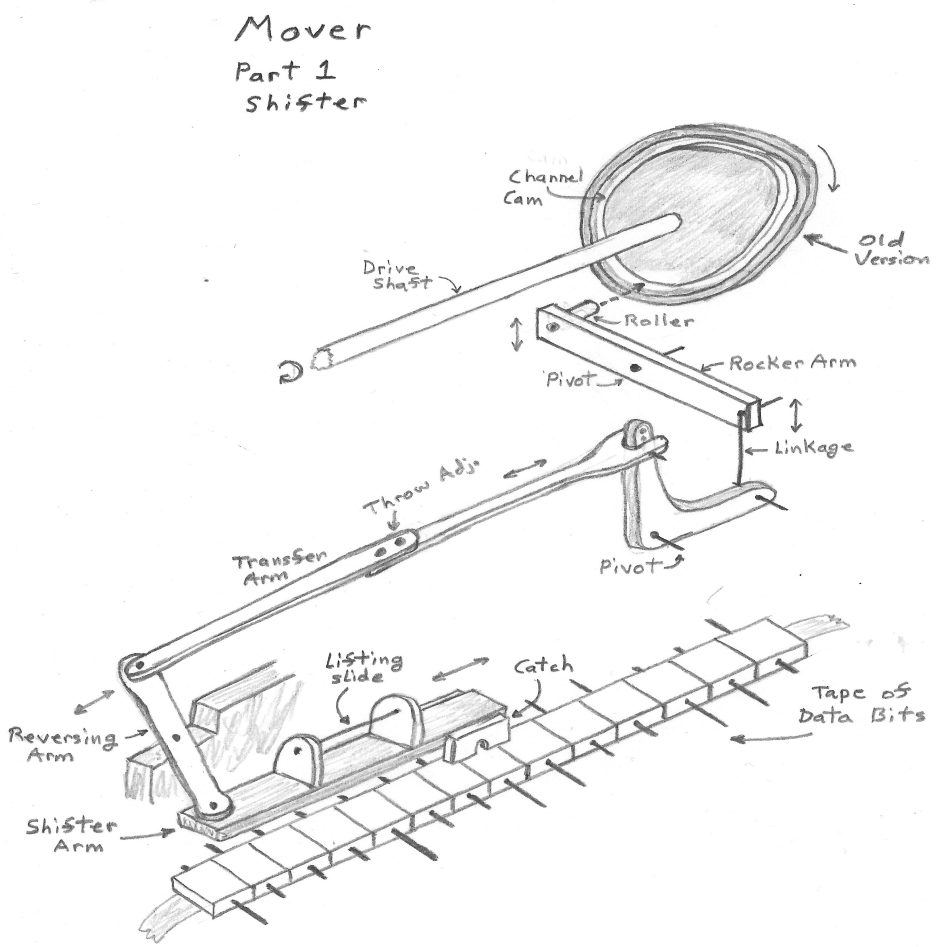



The Data Tape is built out of small blocks glued onto a synthetic ribbon. Pegs are inserted into the blocks. The pegs need to slide easily yet not fall out of the blocks. The forces needed to move the pegs will get multiplied as they are translated through the Writer mechanism. So the easier they slide the easier the remaining mechanism performs. The tape runs along a Rail. The sides of the Rail should not lift the Bit Blocks by the pegs. Precision is necessary when cutting the blocks and drilling the holes for the pegs. Consistent peg length is also necessary. Precision is also necessary when it come to the Catch on the Mover and the Sensors on the Reader.

At this time I would like to say a few words about measurements. I will not offer any here. To do so would demand a complete set of measurements for the whole machine. That I do not have. This machine was built from the ground up, from one part to the next. I cut Bit Blocks and pegs that I thought would work. If they didn't work I threw them away and started over. I built the Rail to fit the Data Tape, then the Catch to fit the pegs. And on, and on, and on.

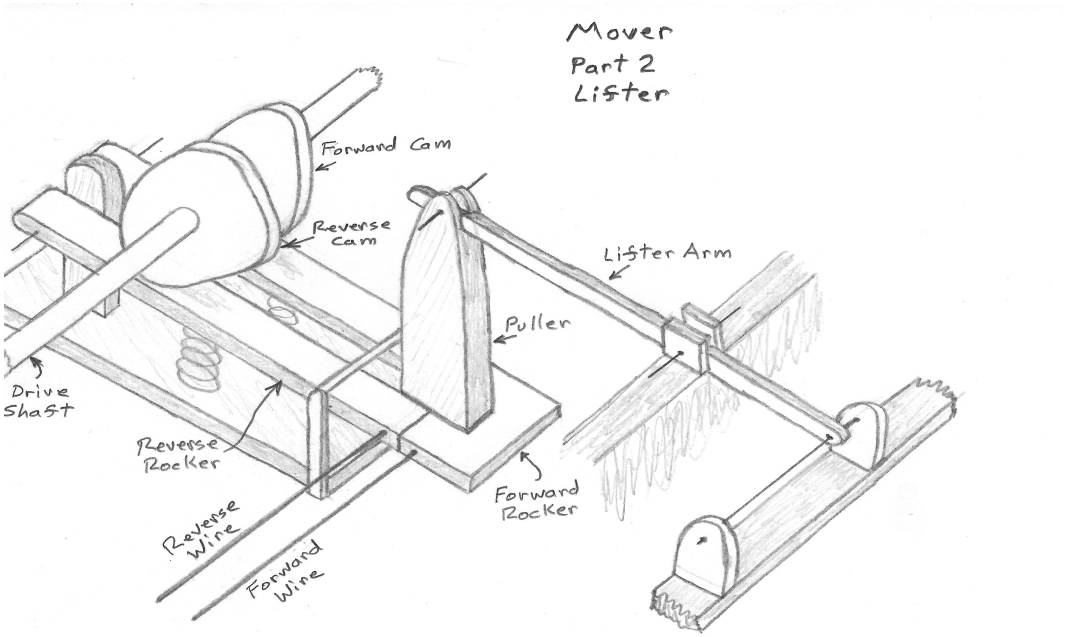
# The Mover

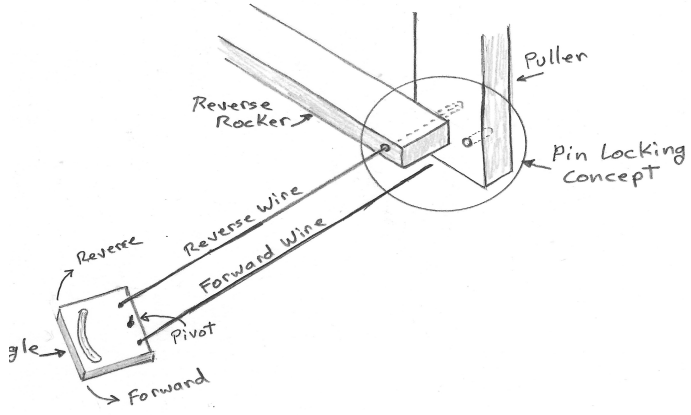
The Mover is comprised of two parts, a Shifter and a Lifter. A Catch on the Shifter Arm locks onto a Data Bit on the tape and shifts the entire tape either forward or reverse one bit. Without the action of the Lifter, however, the Shifter will move the tape forward then reverse leaving the tape in the original position. The Lifter will lift the Shifter Arm above the tape either during the forward or reverse motion of the arm thereby allowing the opposite action to take place. The Shifter mechanism by design reverses the tape first then forwards it. Lifting the Shifter Arm during the reverse action will not move the tape. Lowering the Arm to engaging the Catch at the end of the reverse action will shift the tape one bit forward during the forward action.




This diagram shows the motion of the Shifter Arm.

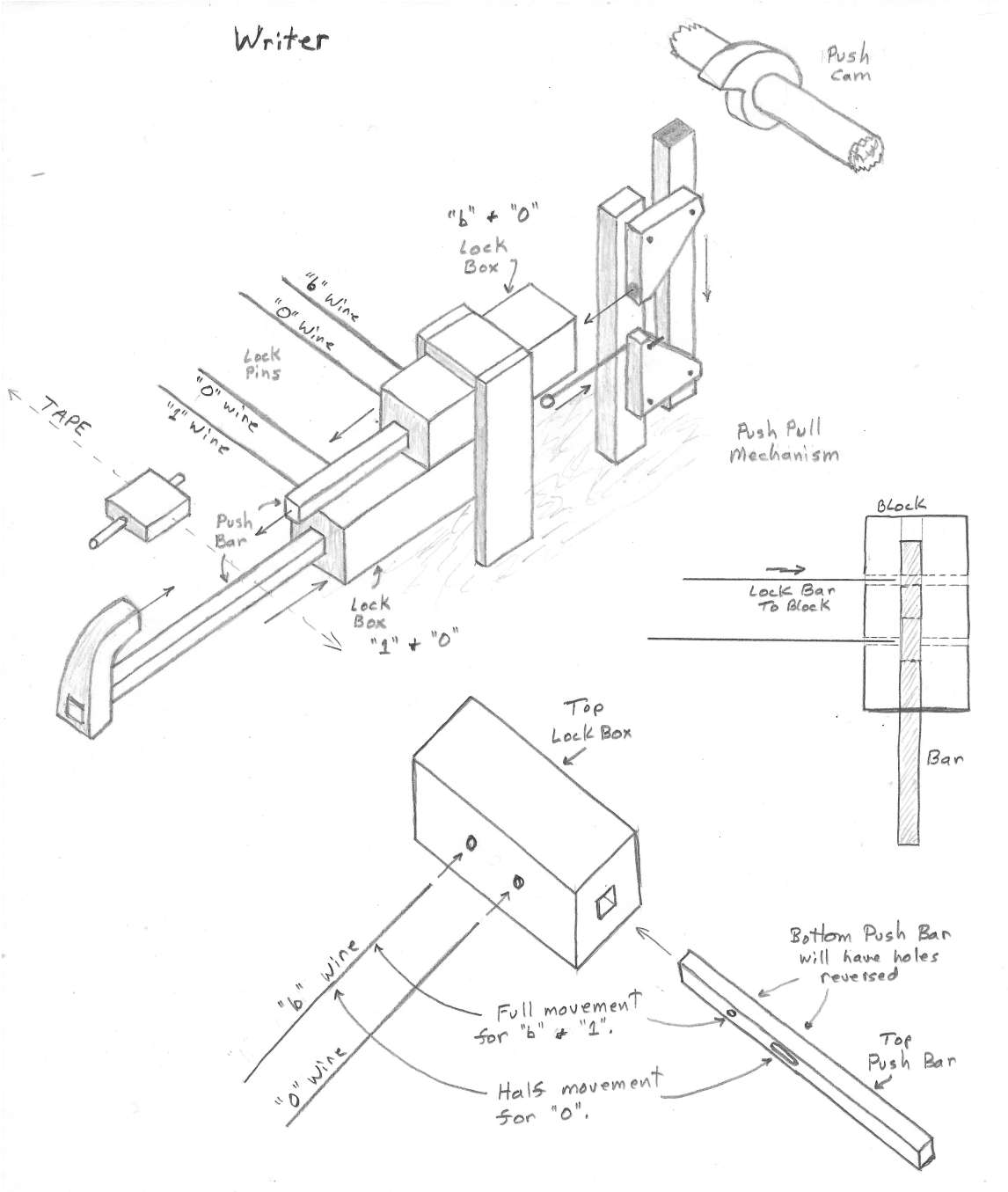
The Lifter has two cams that push down on Rocker Arms. The cams are positioned to lift the Shifter Arm at the precise moments in the movement cycle. Lift during reverse movement and forward motion of the tape results. Lift during forward movement and reverse motion of the tape results. The Forward and Reverse Wires control this action. Because the wires are moved by a Toggle, the Reversing Toggle, one wire will be moved into the Puller and the other wire will be moved out of the Puller. Both cannot, by design, be moved into the Puller at the same time. Both wires also rest in their corresponding Rocker Arms. When a wire is moved into the Puller any action on the Rocker Arm is transferred to the Puller. This process is called the 'Pin Locking Concept.' It essentially locks the Puller to the Rocker allowing the Shifter Arm to be lifted by one of the direction Cams. The following diagram shows the lifter action.



**The Writer**

The purpose of the Writer is to move the Pegs in the Bit Blocks. Either set them to ‘b’ or ‘0’ or ‘1’. If you look at the Data Tape diagram above you will see that the Writer will need to move the pegs all the way down or all the way up or position the peg exactly half way. Notice the small remainder of the peg when it is moved all the way. That small portion is needed for the Mover Catch to grab and allow the shifter to move the tape.



The Writer mechanism Lock Boxes push down and pull up all the way every cycle. The Push Cam moves the Push Pull Mechanism which moves the Lock Boxes. The Push Bars do the actual work of moving the Pegs. The Top Push Bar will push the peg down and the lower push bar will push the peg up through the End Finger. The movement will be one hundred percent. The Write mechanism will destroy itself if both Push Bars are allowed to push at the same time. Consequently, only one Push Bar should engage the peg for a full push at any one time. Only when a ‘0’ needs to be written will both Push Bars be locked into the Lock Boxes. These locking pins will only engage for half the movement for both Push Bars.

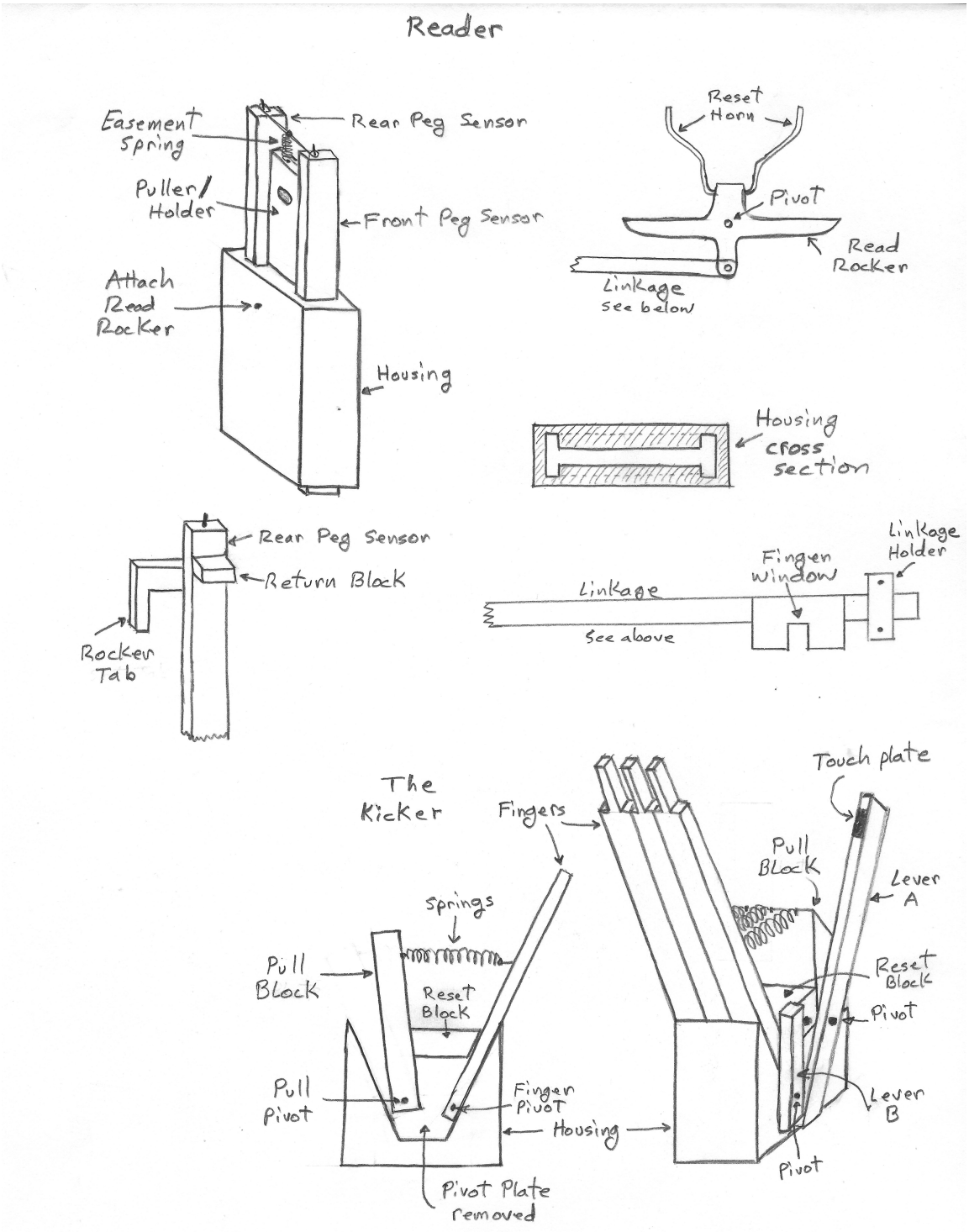
When the Control Wire in the Lock Box is moved through the Push Bar and into the other side of the Lock Box that Push Bar is locked in and will take the movement of the Lock Box. If the Control Wire is moved through the ‘0’ slot of the Push Bar then it will only move half of what the Lock Box moves. The length of the slot is equal to half of the full throw of the Lock Box.

Because the Push Bars will be floating most of the time there is a danger they could migrate out into the Data Tape area and may jam extended Pegs as they move across the writing area. Therefore, extremely light springs are needed to keep them from jamming the tape. The Lock Boxes also have light springs to keep them held in the rest position. Another purpose and for these springs is to keep the holes in the Lock Boxes and Push Bars aligned for a smooth movement of the Control Wires.

The Control Wires are connected to three Toggles: a ‘b’ Toggle, and ‘0’ Toggle and a ‘1’ Toggle. These Toggles are under the Configuration Mechanism and set by wires connected to the Table Hammers (as described later). The ‘0’ Toggle controls the ‘0’ wires for both Lock Boxes since both Push Bars need to move to set the peg exactly half way and since the initial state of the peg is not known by the machine (Not going to engineer that mechanism).

**The Reader**

I don't think I need to go over the Cam on the Power Shaft or the Actuator Arm that presses the reading Sensors down. That should be self-evident. All the other parts of the Reader are shown below.



The Reader detects the position of the Bit Pegs by the Peg Sensors. The Actuator Arm pushes the Puller/Holder down until it rests on the Bit Block. It does two things: hold the Bit Block from rocking in the Rail and pulling the Front and Rear Peg Sensors down onto the peg. All three elements are loose in the Reader Housing. When the Puller/Holder drops a spring on top pulls down on a wire that forces the Peg Sensors down. The spring is needed because the Peg Sensors could come down onto a Peg or down onto the machine frame. When reading a ‘b’, blank, the Front Sensor will rest on the Peg and the rear sensor will rest onto the frame. When reading a ‘0’ both Sensors will rest on the Peg. When reading a ‘1’ the Front Sensor will rest on the frame and the Rear Sensor will rest on the Peg. These differing positions of the Sensors will not allow a rigid connection to the Puller/Holder.

Each Peg Sensor has two more parts added: a Return Block and a Rocker Tab. The Return Block will be engaged by the Puller/Holder on its way back up. This will make sure Peg Sensors are up and out of the way of the Data Tape movement. The Rocker Tab will push the Read Rocker into a position that represents the position of the Peg in the Bit Block. The Read Rocker has a Reset Horn that returns the Read Rocker to its default position as the Rocker Tabs on the Peg Sensors return to the upper position. This default position is equal to the ‘0’ position of the Read Rocker.

The Read Rocker operates Linkage that translates its position to a Finger Window at the rear of the Configuration Mechanism. Three Fingers on the Kicker mechanism push Data Stops in the Configuration Mechanism but only one Finger will pass through the Window. That Finger will push its corresponding Data Stop as the data read on the tape.

The Kicker mechanism best displays the need to be aware of how far parts will be traveling and how much force will be needed to perform a necessary action. The consideration of travel and force was in the forefront of my thoughts when every mechanism was designed. The end of the Finger may need to travel an inch and push a Data Stop Block. And a cam on the Power Shaft would take too long. I needed a Wooden Collar with a rounded screw head, which is how the Finger Mechanism is moved, to quickly move the Fingers. A screw will not move a lever very far. So Lever A pushes Lever B, which has a pivot very close to the force. This amplifies the travel of the upper end of Lever B but also diminishes the force at that end. This then pushes the Pull Block which in turn pulls the Fingers forward using springs. But only one Finger passes through the Finger Window and pushes the Data Stop Block. The other two Fingers are held back but nothing jams because of the springs.

## Configuration Table Mechanism

This part of the machine is by far the most complex. In fact I believe I spent most of the time building three versions of this. The second version had over one hundred parts. But in the end I learned a lot and found out what will and what won't work. The final version is by far a work of, (dare I say it), nah I won't. This unit has four tasks to perform.

1. Receive input from the Tape Reader.
2. Line up the proper row on the program board.
3. Read the Program Board.
4. Set the output wires for the Mover and Writer.
5. Set the next cycle State.

You can see how it can be as complex as the remainder of the machine.

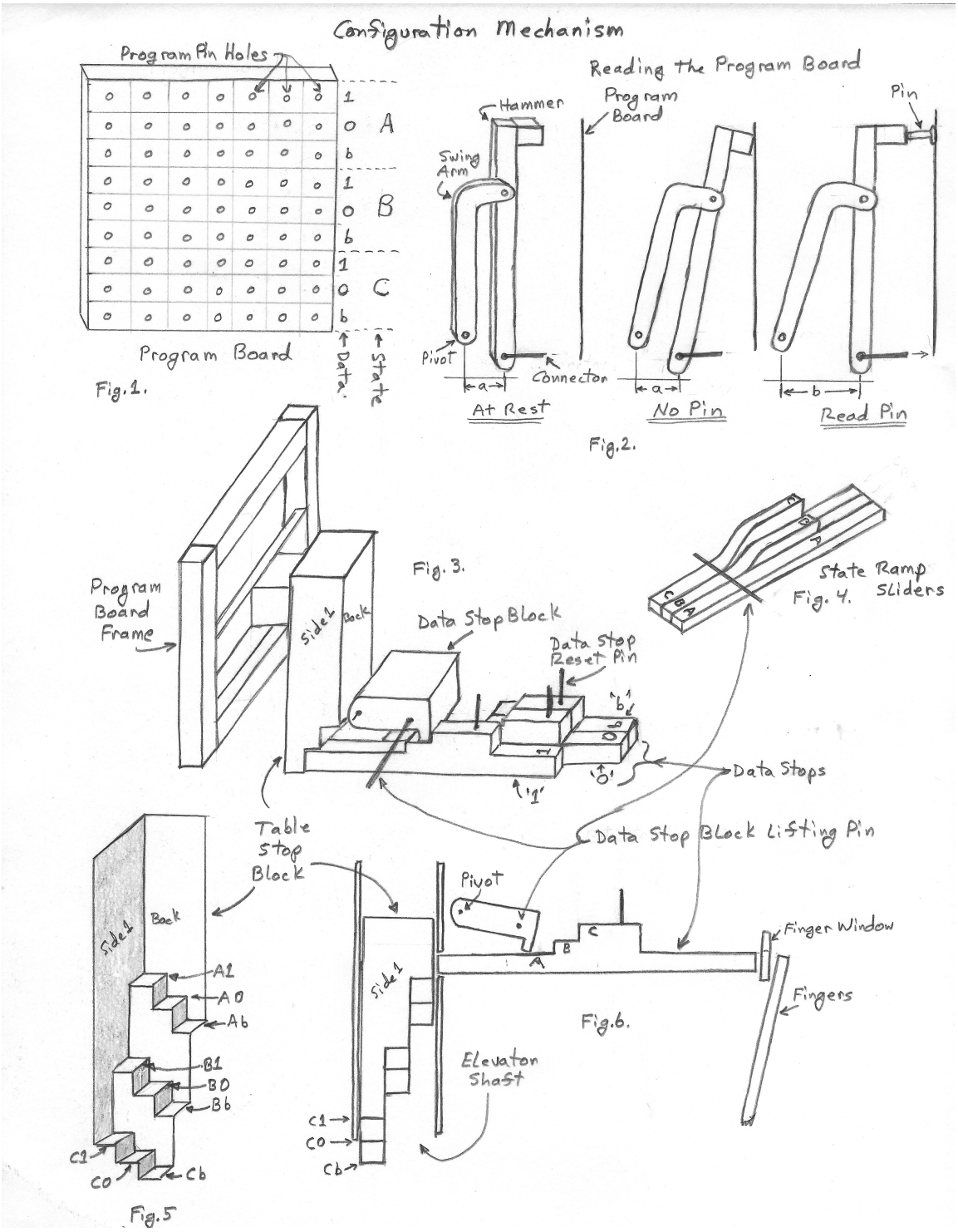
The initial state must be set before we start. The Data Stops need to be manually reset. The State Ramp Sliders must be manually reset which puts the machine in the 'A' state. The Mover and Writer wires need to be reset. And the Data Tape needs to be positioned. Now the machine is ready to begin computations.

The input is the work of the Fingers mentioned in the previous section. The Tape Reader sets the Finger Window and the Fingers, or one Finger, kicks the appropriate Data Stop forward.

With the State and Data Stops set the Table, which has previously been raised, can be lowered to the position matching the stops. With State 'A' and Data '0' the Table will lower to the point where row 'A0' is under the Hammer Heads. Figure 5 on the next page shows the Table Stop Block. It also happens to be one of those optical things. But I think I found a way to see it properly. Look at it as thought the steps are in a distant corner of a room. Now, look at it with the corner facing toward you, not away. You should be able to see the steps properly. You need to look at Figures 3, 5 and 6 to see how the Data Stop Block limits the travel on the Data Stops so only the 'A', in our case above, steps will effect the Table Stop Block. Since the Program Board is locked into the Program Board Frame which is connected to the Table Stop Block the proper row will appear under the Hammer Heads once that whole assembly is lowered and stops.

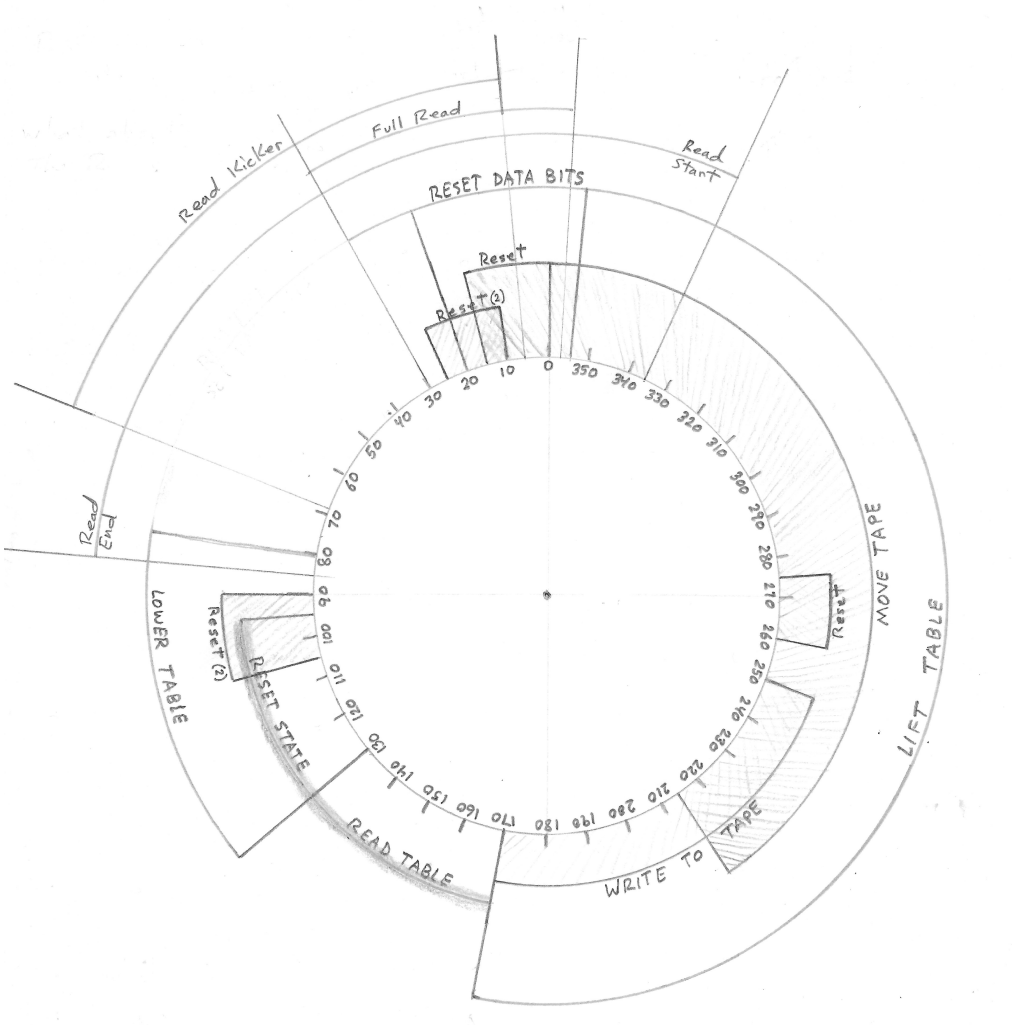
The Program Board is read by swinging the Swing Arm toward the Program Board. As you can see in Figure 2 when the Hammer hits the Program Board the lower end of the Hammer does not move. When it encounters a Pin the lower end of the Hammer also swings toward the Program Board. A Connector linkage transmits that movement to the Toggles under the Configuration Mechanism. These Toggles control the wires used in the Mover and Writer mechanisms. The three State

Hammers transmit the movement to the State Ramp Sliders as shown in Figure 4. This Slider remains set during most machine operations and is reset just before being set again. In effect this is the State Memory for the machine.



I mentioned the Toggles several times and one can be seen in the Mover Lifter diagram above. The ‘b’, ‘0’ and ‘1’ Toggles only have one wire attached at the upper right. The Toggle is turned by the linkage of the Hammer when a Pin is encountered. And the Hammer Linkage sits in a curved slot so it can return as the Hammer returns to its rest position. Therefore, the Toggle will not be pulled back to its rest position. Reset Arms will do that later. The turn or throw of the Toggle needs to be calculated so the wires engage properly.

I also mentioned a 360 degree diagram showing the activities of this machine. The best place to start and stop the machine is around the 300 degree mark. I made a wooden compass and fit it over the right end of the Power Shaft. Then I attached a wooden pointer. This enabled me to chart and see when things were happening. I could tighten up activities using overlap and adjust cams for an optimum operation.




This should help you understand the timing of the machine.

## Other Thoughts

There are a number of issues that need mentioning.

I did not build a safeguard into the Writing Wires. Things will break if the machine attempts to write a ‘0’ as well as a ‘1’. And wires can migrate and drift.

The Reader Rocker basically floats when reading a ‘0’. It could move and present the wrong data to the back of the Configuration Mechanism.

Build the machine so timing and throw are adjustable.

Build the machine so you can remove a part without having to completely dismantle the entire thing.

My part names are not the most logical nor clear.

My engineering technique is not the best, however, efficiency and simplicity can be ignored for artistic aesthetics.

In the end, I had fun building it. And I guess that’s all that matters.

## Conclusion

So there you have it. The machine is initialized, the Data Tape is Read and the data entered into the back of the Configuration Mechanism. The Configuration Table drops to the proper row and the Hammers read the Program Pins. They in turn set the wires that control the Writing of the Data Bits on the tape and then the movement of the Data Tape forward or reverse. They also set the State Slider for the State of the next cycle. Then various resets get the machine ready to do it all over again. That's it, QED.

I'm sure you have read about the "Halting Problem," Entscheidungsproblem. (Go ahead. Check the spelling on that.) Basically, Mr. Turing also theorized a Universal Machine that would take the data tape AND program pins as separate inputs to another machine. That machine would then be considered Universal as it would not need to be programmed. (Is that another challenge?) He did that in order to mathematically try to solve the Halting Problem. Which is; can a machine determine if a computation is complete or infinitely looping? And the result is a resounding, NO. It can't. Therefore, the problem.

My machine has a similar problem. According to the 1936 specifications of a Turing Machine the point where a computation is completed is not specified. It is mostly guesswork and not mechanical. In other words, you see that the machine processed the input and is done.

Or is it.