GUNFIGHT

midway

COMPUTER SERVICE MANUAL

video game data library   vol. IV

Gun Fight Computer Service Manual

For The Midway 8080 Microprocessor Game Series

A comprehensive analysis of the Gun Fight game computer.

Technical Production by Laurel Publications

Table of Contents

SECTION I GENERAL INFORMATION

1.1 Introduction

1.1.1 Why Use A Microprocessor? The integrated microprocessor (often abbreviated simply as μP) has caused a revolution in the electronics industry for the contemporary μP now makes possible real computers for only a few hundred dollars. Ever since the advent of video games, designers have often wanted to use an actual, general-purpose type computer to operate their games and, in fact, a few custom games have been constructed using fairly powerful minicomputers. But it has never been possible to use such a computer for production video games simply because the cheapest available minicomputers cost thousands, rather than hundreds, of dollars. But today, with microprocessor chips available for $20 and under, systems comparable in power to commercial minis can be built for maybe a tenth of the cost. Obviously the μP is a reality today and is rapidly changing the face of the electronics industry. But what of the reasons designers are so interested in applying them to video games? After all, video games have been successfully manufactured for a number of years using the random-logic type of architecture, so why change now?

Actually, there are quite a few reasons, but most of them boil down to two considerations: economics and design flexibility. But before getting into all of this, let's take a couple of steps backwards and talk about earlier techniques for implementing video game designs.

1.1.2 Random Logic Vs. Processor Control. Before μP systems burst upon the scene, video games were invariably designed using a dedicated "computer" custom built from random-logic elements which are assembled in such a way as to generate both the video images displayed on the screen and the various rules by which these images are controlled. The typical random-logic video game computer consists of a large number of SSI and MSI devices wired together to produce the desired effects. Since only small and medium scale devices are used, this type of computer tends to require a large amount of circuitry and PCB real estate to generate even the simplest displays and algorithms.

While the random-logic approach clearly works, it has several serious drawbacks which become particularly evident when this technique is compared with operating a game via processor control. Probably the most significant disadvantage is the tremendous amount of development time required to generate each new game since an entirely new computer is required for every distinctly different game theme. Furthermore, bugs which show up in the field at a later date can become a serious problem and one which may be extremely difficult to correct since all the boards must be recalled from all over the country (or even all over the world) and the custom printed circuit board modified by cutting traces and adding jumpers to implement the engineering change.

Now, compare this with the way a microprocessor game is developed.

First, the central processing unit (CPU) is designed. Although the CPU may not contain a very large chip count as compared with a random-logic design, be aware that many of the chips found on a CPU board are LSI circuits which contain many more individual components than the simpler MSI varieties. So, although the physical size of the CPU circuitry may be smaller, the overall complexity of the system is far greater.

Microprocessor manufacturers not only build the μP itself, but they also generate associated componentry necessary to the operation of the microprocessor chip. A good example is the 8216 Bidirectional I/O Drivers used to control the 8080 system data bus. These chips are absolutely necessary to the operation of the 8080 and the manufacturer of the devices also supplies the schematic for their connection. Furthermore, almost all μP manufacturers also supply a developmental system which the engineer can often use to "breadboard" his design. In fact, sometimes all that the engineer really needs to do is to write his own custom program and eliminate any unnecessary circuitry from the developmental system. Actually, the system is already half designed before the video game designer sits down and starts work and this can obviously result in a tremendous savings of engineering time.

So, after playing with the developmental system for a while, the designer creates the actual CPU design for the video game while taking into consideration the types of functions the game system will need to perform. One of the major considerations is the fact that the CPU must work for a large number of different game themes. If the system is in fact designed according to this consideration, it should be useful for an extended period of time and this is the main reason the μP architecture is so attractive, for once the company has a viable design, a number of advantages are quickly realized. Obviously, the most important is the fact that the development costs required to get the initial system up can be spread out over the entire run of machines which might last for several years. Furthermore, only a relatively small number of parts need be stocked at a considerable savings to the manufacturer. And, since the manufacturer knows a great many CPU boards will be sold during the production run, a large number of them can be produced at once which greatly reduces the costs involved in making high quality printed circuit boards. For example, one of the larger costs associated with manufacturing PC boards is setting up for drilling. Obviously, the more boards that are drilled, the cheaper the set-up cost for each board will be.

But there are many more advantages with the μP architecture other than savings realized in engineering and the actual construction of the PCBs. Prior to shipping, all boards must be thoroughly tested and since a single CPU system is used for a great many different games themes, there is no need for test technicians to become familiar with a new random-logic design every month or so.

1.1.3 Advantages of Software Control. The foregoing examples have all been hardware-oriented advantages, but what we are really dealing with here is a software-oriented system wherein the software itself offers a great many benefits which may or may not be immediately obvious. For example, once a reliable CPU has been created, implementing another game theme mainly becomes a matter of writing new software and this can often be accomplished in a tenth of the time that would ordinarily be required to assemble a new hard-wired, random-logic design. Also, if bugs show up in the new game, these can be worked out on a

software level in such a way that the program is corrected and shipped out at only a fraction of what it would take to recall a large number of random-logic PCBs and modify them to correct the problem.

Software systems are quite flexible and this aspect offers a number of advantages not possible with random-logic designs. For instance, several objects can be moved on the CRT simultaneously using a μP without increasing system complexity. However, were a designer to try to do this with a random-logic game, he would quickly discover that he needs a separate motion circuit for each different image to be moved. Were he to want very much action on the CRT, he could easily end up with a game not feasible to produce with random-logic elements.

Software controlled video games are more easily modified in the field and this could conceivably be of great significance depending on the actual circumstances. Sometimes a game looks very good in a test situation, but receives little play in the field. Were the manufacturer not able to correct this situation, that company could encounter trouble. However, if the game happened to be a software oriented system, the manufacturer could still rectify the situation even after the games had been shipped from the factory for all the company needs to do to correct the matter is to write a more acceptable game program, burn it into a number of PROMs and send them out free of charge to the disappointed operators.

And lastly, once people become used to a particular μP system, technicians everywhere will be able to repair games using that system. Also, since most μPs function similarly, there is a great deal of positive transference from one system to another meaning that technicians will already be acquainted with the general architecture of the system before they sit down and become familiar with the specifics of that system.

Now that we have discussed some of the reasons that microprocessors are so quickly gaining in popularity for video games (and a tremendous number of other systems as well), it is time to turn our attention to the specifics of the Gunfight game system. In the following pages, we will discuss the general architecture of the Midway 8080 CPU mother board and the Gunfight game board as well as the specific operation of all phases and components of the system itself. In conclusion, we will present a number of troubleshooting approaches for dealing with specific problems and problematic areas as well as providing a generous amount of actual troubleshooting data so that video game technicians with ordinary troubleshooting equipment will be able to fully explore the circuitry found in this game.

1.2 A Typical Game Sequence

1.2.1 The Attract Mode. The condition of the game prior to the start of the game is known as the attract mode and this mode is characterized by the display in Figure 1.2.1-1. During the attract mode, the left cowboy simply walks onto the screen, fires a shot and retreats again only to repeat the same process a few seconds later. The name of the game (Gun Fight), the score remaining from the last game (0 0) and the announcement that the game has ended (Game Over) are displayed on the CRT as well. And, since the function of the attract mode is to entice prospective

GUNFIGHT

players into depositing their coins, the words "Insert Coin" are placed on the screen.
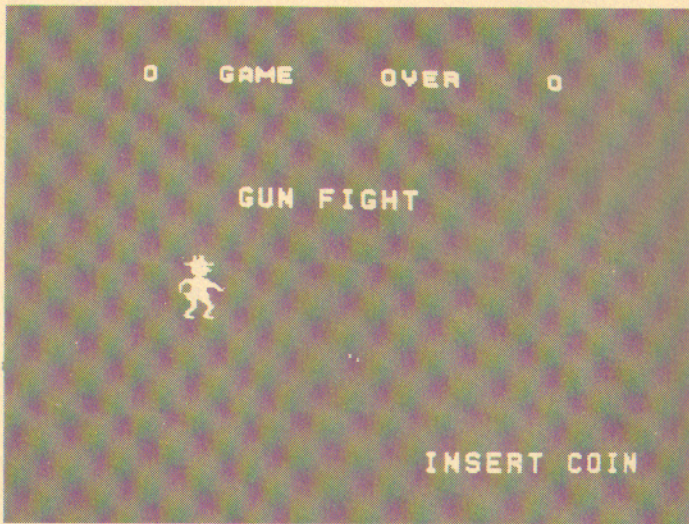

Figure 1.2.1-1 Attract Mode Display

1.2.2 The Credit Mode. After the players have deposited a coin, the CRT announces, "Hey pardner! You have credit. Push (start) button." The game will not commence until the start button is pressed.
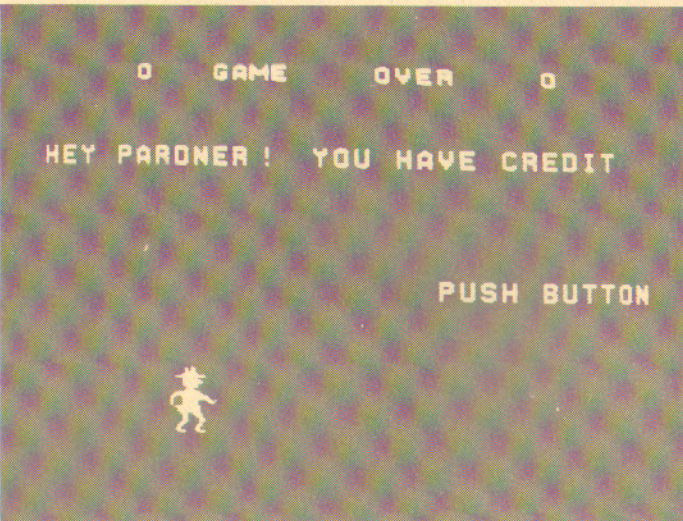

Figure 1.2.2-1 Credit Mode Display

1.2.3 The Play Mode. After pressing the start button, the game begins. At this point, the players' controls become "live" and each player discovers that he can move his cowboy in any direction within his playing area, that he can control the position of his cowboy's gun arm and that he can fire a shot from his gun.

In addition to the two cowboys, a number of other images are displayed on the CRT. Under each cowboy is a series of symbolized shells or bullets and, as with a real "six-shooter," each cowboy is allowed only six

shots per shoot-out. As each shot is fired, one of the bullet images is removed from the bottom of the screen. Also displayed at this time are the game timer (center numerals) and each player's individual score (left and right numerals).
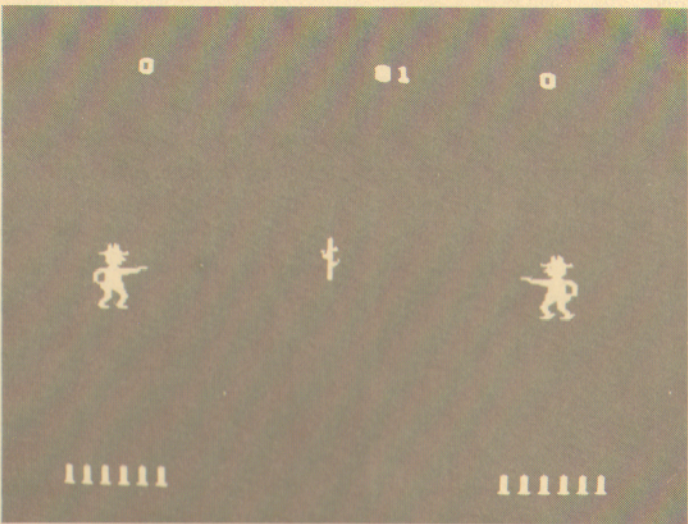
Figure 1.2.3-1 Initial Play Mode Display

When one cowboy shoots the other, the cowboy who is hit falls down into a prone position, the words "Got me" are displayed above the body and the cowboy's hat is placed over his face (Figure 1.2.3-2). Simultaneously, the successful player is awarded a point for scoring the hit and this is indicated by incrementing his set of score numerals.

If one cowboy uses up all his ammunition without hitting the opposing cowboy, the shoot-out timer begins counting down. The shoot-out timer is located midway between the left player's score and the game timer. When this timer reaches zero, a new shoot-out is automatically started by the computer.
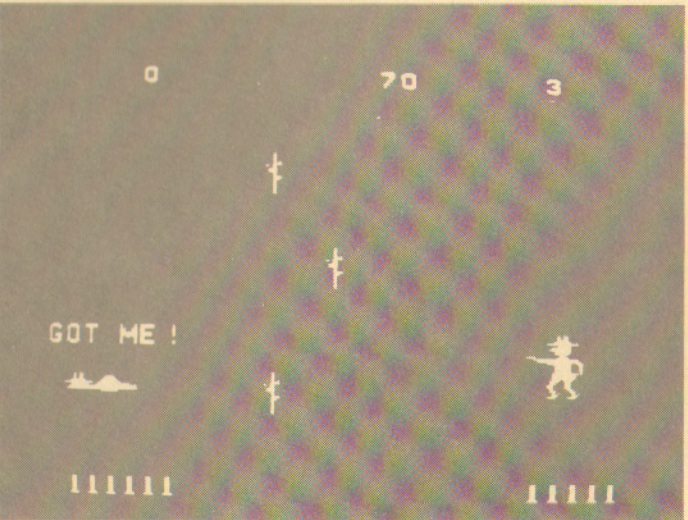
Figure 1.2.3-2 "Got Me" Display

The game continues in this fashion until the game timer is decremented down to zero, however, after each shoot-out an additional object is placed on the screen to increase the difficulty of the game. The first object is a single cactus, to which a second cactus is added at the next shoot-out, and so on until several cacti, pine trees and a moving stagecoach are all displayed on the CRT at the same time.

It is interesting to note that each player is given his own set of objects which he can hide behind safely without fear of being shot. For example, the left player can hide behind the left pine tree and also shoot through this pine tree. But if the right player tries to shoot the left player through the left tree, his shot will only blow off a small piece of the tree and not hit the left player. The stagecoach is an object which is held in common for both players in that neither player is able to shoot through the stagecoach.

1.3 Electrical Adjustments

1.3.1 Introduction. In this system, there are four main categories of electrical adjustments and these are: (1) TV monitor adjustments, (2) power supply adjustments, (3) operator adjustments and (4) sound system adjustments.

1.3.2 TV Monitor Adjustments. These adjustments are located on the TV monitor itself where the four most commonly used ones (A-D) are found on a panel accessible from the rear of the monitor. Vertical liniarity and height are adjusted via pots located on a printed circuit board in the monitor and the yoke is adjusted at the rear of the picture tube. More detailed information concerning the locations of these adjustments is found in the Motorola monitor manual.

A. HORIZONTAL HOLD: Adjustment is indicated only if the picture is off center horizontally, if the images appear warped or if the entire picture is broken up into a series of diagonal lines (Figure 1.3.2-1). Adjust for a stable, horizontally centered picture.
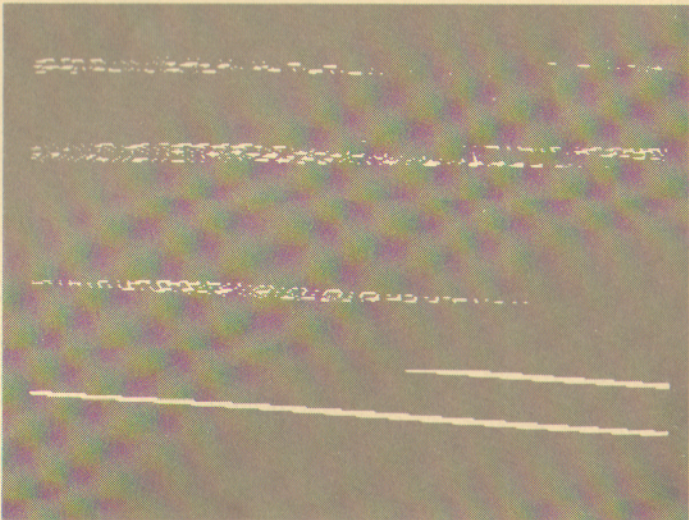
Figure 1.3.2-1 Horizontal Hold Adjustment Will Generally Rectify This Problem

GUNFIGHT

B. VERTICAL HOLD: Adjustment is indicated if the picture appears to be rolling up or down the CRT or if the picture is stable, but not centered vertically. Adjust for a stable, vertically centered picture.

C. BRIGHTNESS: Brightness is adjusted before contrast. Adjust for a dark background.

D. CONTRAST: Adjust so the white images are as bright as possible against the dark background without being blurred.

E. VERTICAL LINIARITY: Adjust only if the images at the top of the CRT appear to be compressed. If adjustment is indicated, adjust the V LIN pot so that the stagecoach image does not become compressed as it travels from the bottom to the top of the CRT.

F. VERTICAL HEIGHT: Adjust only if the entire picture appears compressed vertically. Be aware that there is some interdependence between the liniarity and height adjustments.

G. YOKE: Yoke adjustment is indicated only if the entire picture is off center and the H HOLD adjustment has insufficient effect. Adjust both yoke rings simultaneously for optimum centering of the image area within the borders of the CRT.

1.3.3 Power Supply Adjustments. The following power supply adjustments are performed at the power supply PCB (P.C. 80-901) and the specific adjustment locations are indicated in Figure 1.3.3-1. Arrows indicate direction of increase.

*This adjustment performed only if I.C. 3235 is used in mother PCB location C5.

Figure 1.3.3-1 Power Supply Adjustments

A. +5 VOLT SUPPLY: Connect a VOM or DVM to power PCB edge connector pin 11 (or 12) and ground and adjust pot A for a reading of exactly +5 volts.

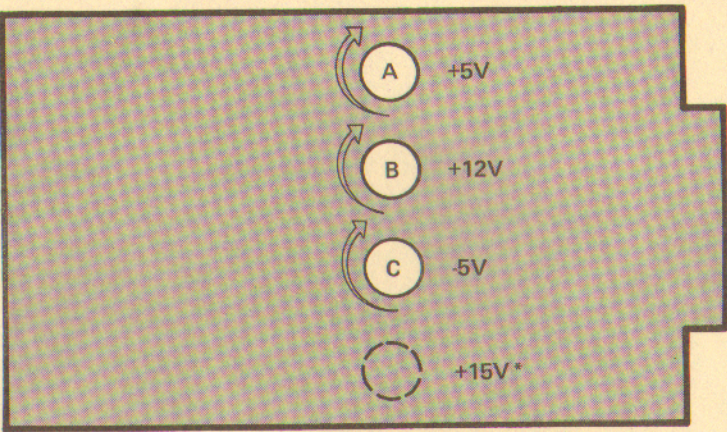B. +12 VOLT SUPPLY: Connect test instrument to power PCB edge connector pin 1 (or 2) and ground and adjust pot B for a reading of exactly +12 volts.

C. -5 VOLT SUPPLY: Connect test instrument to power PCB edge connector pin 5 and ground and adjust pot C for a reading of exactly -5 volts.

D. +15 VOLT SUPPLY: Perform this adjustment only if 3235 clock driver is found at mother PCB location C5. Connect test instrument between pin 8 of the power PCB edge connector and ground and adjust pot D for a reading of exactly +15 volts.

1.3.4 Operator Adjustments. These adjustments are performed by setting individual on/off switches found in a DIP package located at game PCB position C1 (see Figure 1.3.5-1). The following adjustments affect game length, plays per game, credit and coins required for game credit. Please be aware that the adjustment procedure varies according to game PCB revision number. If the game PCB carries the revision number 597-907B, use the information in the center columns. If the PCB carries revision number 597-907C, D or E, use the information under the right columns.

A. GAME TIME: To adjust the total number of time units per game, set the switches according to the table below. For example, if you have a B revision game PCB and you desire a game length of 90 time units, set switches 2 and 3 to the OFF position.

TIME	PCB 597-907B SW#2	SW#3	PCB 597-907C,D&E SW#5	SW#6
90 UNITS	OFF	OFF	OFF	OFF
80 UNITS	OFF	ON	ON	OFF
70 UNITS	ON	OFF	OFF	ON
60 UNITS	ON	ON	ON	ON

B. COINS PER GAME CREDIT: This procedure adjusts the number of coins required to produce game credit. Please be aware that game credit may allow the player 1 to 4 games depending on how the next adjustment procedure is performed. Adjust according to the table below. For example, if 1 coin per credit is desired for a B revision game PCB, set switches 4 and 6 to the ON position.

NUMBER OF COINS	PCB 597-907B SW#4	SW#6	PCB 597-907C, D&E SW#1	SW#2
4 COINS	OFF	OFF	OFF	OFF
3 COINS	ON	OFF	ON	OFF
2 COINS	OFF	ON	OFF	ON
1 COIN	ON	ON	ON	ON

C. PLAYS PER GAME CREDIT: This procedure adjusts the number of games awarded the player once game credit has been achieved. Adjust according to the table below. For example, if 1 play per game credit is

desired adjust switches 5 and 7 to the ON position if you have a B revision game PCB.

NUMBER OF PLAYS	597-907B SW#5	SW#7	597-907C, D&E SW#3	SW#4
4 PLAYS/CREDIT	OFF	OFF	OFF	OFF
3 PLAYS/CREDIT	ON	OFF	ON	OFF
2 PLAYS/CREDIT	OFF	ON	OFF	ON
1 PLAYS/CREDIT	ON	ON	ON	ON

1.3.5 Sound System Adjustments. Volume adjustments are provided for each individual sound as well as master volume for each player. This has been done so that all the sounds may be adjusted to the same general level. Begin by adjusting the left player's individual sounds together, then repeat this procedure for the right player. Finally, balance the left player's total volume to the same level as the right player. The locations of these adjustments are indicated in Figure 1.3.5-1 and the pots are adjusted according to the table below. The arrows indicate the direction of volume increase.

A. Left player hit volume.
B. Right player hit volume.
C. Left player shot volume.
D. Right player shot volume.
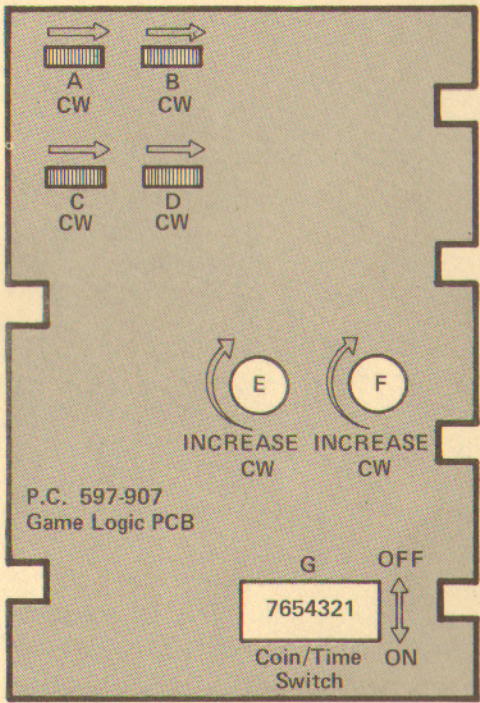E. Left player master volume.
F. Right player master volume.

Figure 1.3.5-1 Sound And Coins/Plays/Time Adjustments

GUNFIGHT

1.4 System Architecture

1.4.1 System Subassemblies. Figure 1.4.1-1 shows the basic electrical subassemblies required for this system. The system has a separate power supply PCB which provides nearly all the regulating and other functions required to power the integrated circuits and other components. The power supply is connected by the wiring harness to the CPU mother board (also known as the mother PCB), the game board and the player's controls. The players' controls are located on an external control panel mounted to the front of the cabinet and this subassembly consists mainly of a joystick and a gun control for each player. The players' controls are connected only to the game PCB and are the main inputs to that part of the system. The game PCB is considered to be a peripheral device which does some processing of the input signal flow in terms of "smoothing" motion generated by the mother PCB. The game PCB "talks" to the mother PCB over the multiplexed data bus, however the CPU writes to the game PCB over the output data bus. Both busses are unidirectional (at least between the boards) meaning that information can pass in one direction only.

The mother PCB contains the CPU, the memory and a number of other important devices. The CPU is controlled by program information stored within the ROMs (Read Only Memories) and it uses the inputs from the players' controls entering the CPU over the multiplex data bus to change playfield images and actions according to a set of rules stored in the program. All the images including the numbers, letters, cowboys, trees, etc. are stored in the ROMs as well.

After the CPU has performed the required operation to the data, the information is stored in the RAM memory located on the mother PCB. Although some of this RAM is used as a "scratch pad" memory, most of it is used to store the next frame to be displayed on the TV screen. In other words, the RAM memory contains a pattern of bits each of which corresponds to one of the displayable points on the CRT.

The game PCB is essentially a peripheral device and as such, it gathers information (i.e. the players' control settings) to be processed by the CPU. However, this part of the system also contains the shifter, a sub-circuit used to "smooth" out the jerky motion generated by the mother PCB. The shifter allows single-bit increments of motion to be generated from the larger 8-bit shifts produced by the CPU. The game PCB also produces the various sounds required by the system and controls them so they are heard only at the correct moments in the game.

1.4.2 Signal Flow. There are a number of busses and other significant signal lines which connect the mother and game PCBs to each other and enable data and control signals to be passed from one board to the other. These busses and lines are the multiplexed data bus, the output data bus, the address bus, the sample line and the reset line.

First of all, a definition of the word "bus" is in order. By bus, we mean a port of eight parallel data lines where each line represents one bit of information. The total of eight bits across the bus is known as one byte.

1.4.3 MULTIPLEX DATA BUS: The multiplex data bus is unidirectional and it carries information only from the game PCB to the mother PCB. Four types of information are found on this bus at different times: left player controls, right player controls, operator instructions and recirculating shifted data. When the CPU reaches a place in its program where information about the settings of the left player's controls is needed, it sends out an instruction over the address bus which is decoded by the game PCB and used to select the binary information representing the settings of the left controls.

1.4.4 OUTPUT DATA BUS: The operation of this bus is slightly more complicated than that of the multiplex bus because the output data bus is bidirectional on the mother PCB, between the 8080 CPU and the bus drivers, but it is unidirectional between the mother PCB and the game PCB. On the mother PCB, this bus is used both to enter data from other circuits (i.e. the RAM memory) into the CPU so the CPU can digest the new data and also to output the digested date from the CPU back into memory or to some other device. But between the mother PCB and the game PCB, the data on this bus can flow only from the mother to the game PCB. In this case, the CPU — after "crunching" on the data — uses the output bus to write digested data into the shifter circuit of the game PCB. After being shifted, this data returns to the mother PCB via the multiplex data bus.

1.4.5 THE ADDRESS BUS: Part of the address bus (bits A^8, A^9 and A^{10}) is peeled off the 16-bit wide mother PCB address bus and sent to the game PCB and be aware that this is a unidirectional bus on both boards. On the mother PCB, the full 16-bit wide address bus is used to address memory (both RAM and ROM), however the part of the address bus taken to the game PCB is used to address that peripheral for the type of data (player control settings, operator instructions, etc.) called for by the CPU program.

1.4.6 THE SAMPLE LINE: The CPU uses the sample line to notify the game PCB that the CPU is writing information to that peripheral. This is necessary because the game PCB must know that the data being placed out on the output data bus is for the game PCB. The sample line is a single control line and it can only contain one bit of information.

1.4.7 THE RESET LINE: The reset line originates from the power PCB which has a power-on-clear circuit that outputs a signal when the game is powered up. This signal is used to clear all the latches on the game PCB so the game begins with all invalid information cleared from the latches.
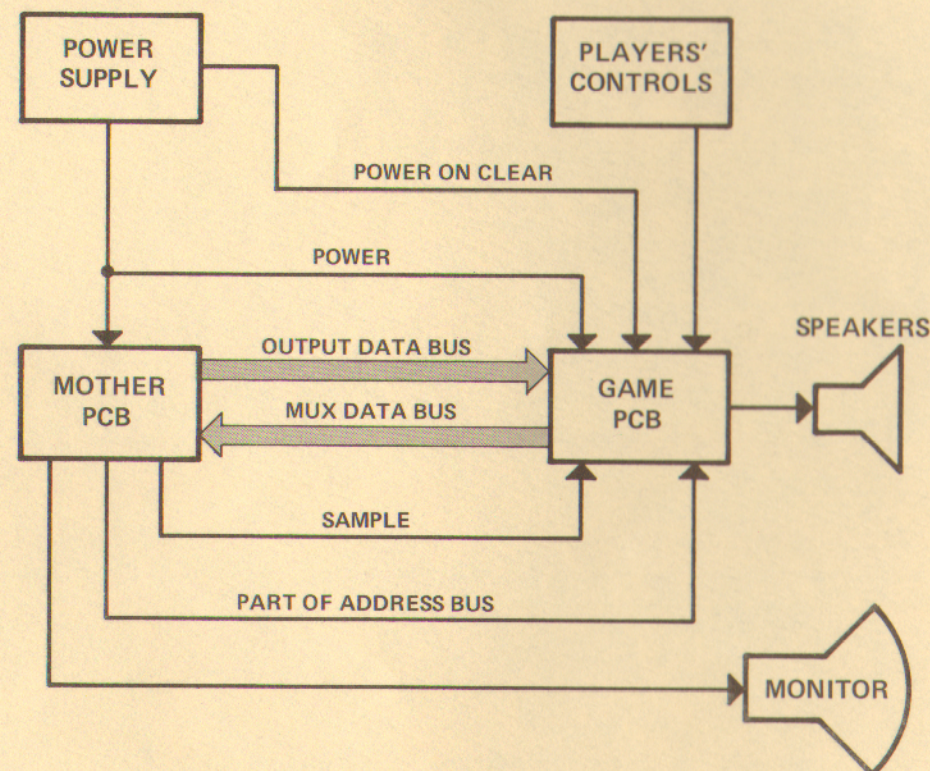
GUNFIGHT SYSTEM BLOCK DIAGRAM

Figure 1.4.1-1 System Block Diagram

1.5 Mother PCB Architecture

1.5.1 Introduction. This discussion of mother PCB and 8080 architecture is intended for orientation purposes only so these areas are covered on a very superficial basis as the main thrust of this manual is toward understanding how the game PCB works in this system. Obviously, the game PCB will make little sense if it is removed from the context of the system as a whole, so some information must be provided about the operation of both the mother PCB and the 8080 itself. However, the following information is not intended to convey complete understanding of the operations of these parts of the system. If you require more detailed information, please be aware that an entire manual has been devoted to this subject.

The mother PCB block diagram (Figure 1.5.1-1) illustrates the major subcircuits found on the mother board, the busses that interconnect these circuits and the direction of signal flow for each type of information. The major sections in this area are sync and timing, RAM memory, RAM data latches, video output, CPU input selector, I/0 drivers, status latch, interrupt flip-flop, the CPU, the ROM selector and the ROM memory itself. The major busses are the address bus, the instruction bus, the bidirectional data bus, the output data bus, the RAM data bus and the multiplex data bus.

1.5.2 Sync and Timing. This section generates the master timing signal CLOCK which is processed in several ways to generate a two-phase clock ($\phi 1$ and $\phi 2$) for the entire system which is available in both normal TTL voltage levels for the majority of the circuitry and also in a "high voltage" (12 volt) level for the 8080 CPU. The signal CLK is also divided down by a divided chain to provide clock submultiples used both in the generation of the sync signals and to read out the contents of the RAM memory in such a way that the information is displayed coherently on the CRT.

1.5.3 The RAM Memory. All the RAMs in this section are 4K by 1-bit dynamic devices. Since they are 1-bit devices, it takes eight of them to construct a byte. And, since they are dynamic devices, they must constantly be refreshed, otherwise their contents will be lost. However, this type of RAM is constructed in such a way that reading the information out of memory is sufficient to refresh the memory. A total of 8K bytes of RAM are required by this system and the RAM is addressed both by clock submultiples as well as by the address bus. The address bus is used to address parallel data in and out of RAM for use by the CPU, however the clock submultiples read out the contents of the RAM a byte at a time into the shift register in the video output section and thence onto the CRT. In passing, we would like to mention that the stack (and a few other types of data as well) are also stored in the RAM memory on a temporary basis. These areas of the memory are reserved for use by the CPU and are not read out onto the CRT.

Note that only chip H8 is drawn as it is actually connected. All the other RAM chips are connected in the same way, but they are drawn in a "condensed" manner to make the drawing easier to read.

1.5.4 RAM Data Latches. These latches simply take RAM data and hold it stable until the CPU is ready to accept the information.

1.5.5 Video Output. This section is composed only of a single shift register which accepts a byte of parallel data from RAM and clocks it out in a serial format so each bit may be displayed sequentially as the electron beam scans the CRT.

1.5.6 CPU Input Selector. This section selects between four separate types of signals: RAM data, multiplex data bus information, instruction bus information and restart instructions where the selection process is controlled by the output of the status latch. In other words, when the CPU reaches a place in the program where control setting input data from the game PCB is required, the status latch puts out a signal which selects the multiplex data bus information in the input selector section.

1.5.7 I/0 Drivers. These devices control the direction of data flow between the CPU and the rest of the mother board circuitry under command from the CPU itself. In other words, when the CPU is writing data to a peripheral it sends out a signal to the drivers which causes them to shut off the input lines and enable only the output lines from the CPU. However, when the CPU needs to read data, it sends out a "read" instruction to the drivers which causes them to turn the output lines off and allow the inputs through.
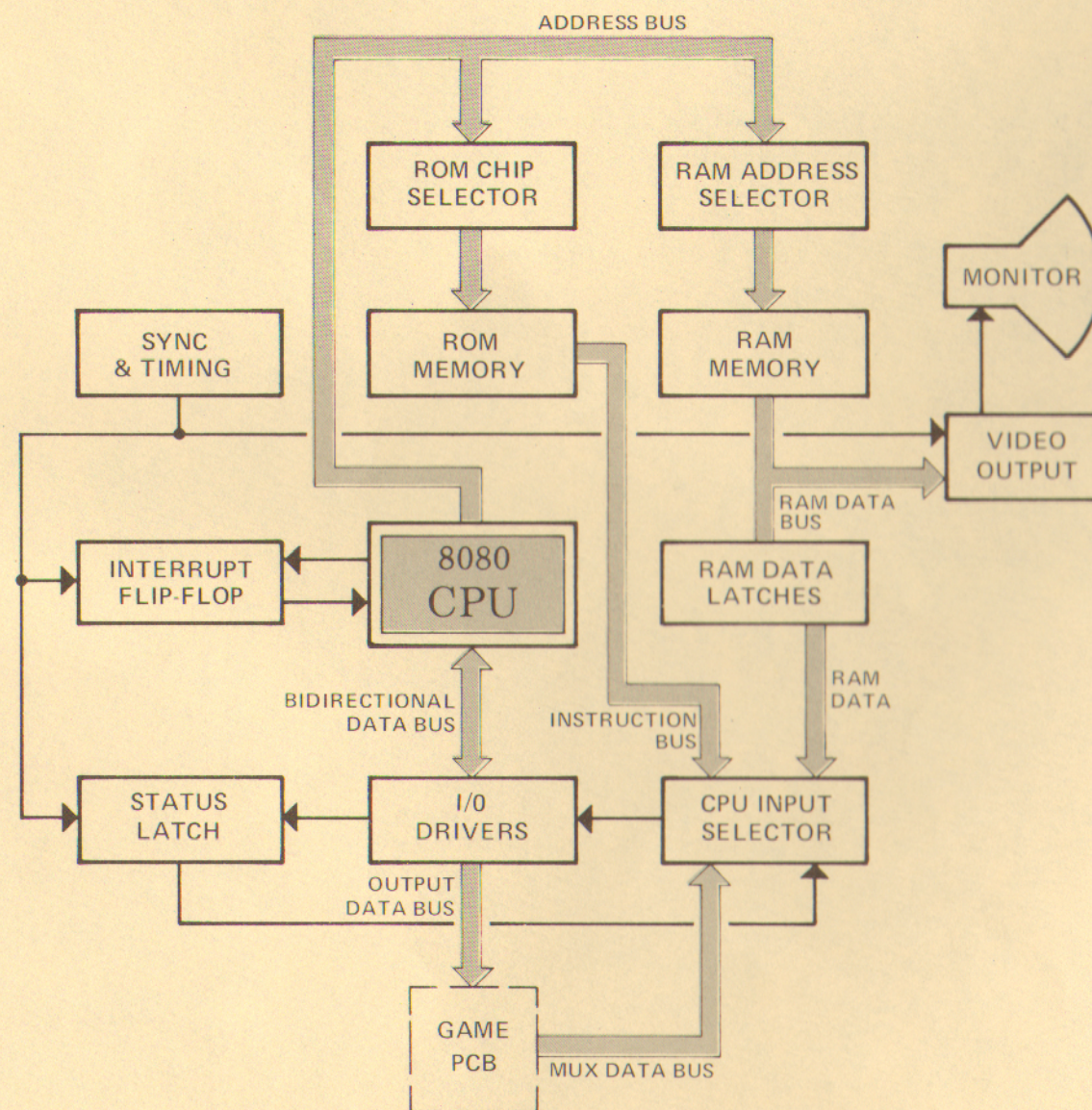

Figure 1.5.1-1 Mother PCB Block Diagram

GUNFIGHT

1.5.8 Status Latch. This is just a regular hex latch used to indicate to the rest of the system the type of operation the CPU will be performing during the present machine cycle. During the very first part of the machine cycle, the status is latched and held for the entire cycle.

1.5.9 Interrupt Flip-flop. This small section provides a "real-time" (i.e. a non-software dependant) clock for functions such as the game timer.

1.5.10 The ROM Selector. This section decodes part of the address bus and uses the resulting information to turn on the proper ROM chip when the information in that chip is needed by the CPU.

1.5.11 The ROM Memory. Whereas information can be written into and read out of the RAM memory, data can only be read from the ROMs. ROMs are useful in this application because their contents are non-volatile. Were the game program (and image information) held within RAM and the power accidentally turned off, the entire program would have to be reloaded. The ROM memory stores both the game program as well as all the images used.

1.5.12 The 8080 CPU. The 8080 is the "brain" of the system for it provides all the control logic required to operate the game. Under program control, the CPU processes data which is provided both by memory and by the game PCB. The program is stored in the ROM memory and it directs the CPU by giving it an instruction directing the CPU to perform the required process to the data. Each instruction usually carries with it the address of the memory location where the data is to be found. The sequence of program steps is controlled by the program counter which is incremented at each instruction and simply steps through the program in a sequential order.

The illustration above is a more-or-less simplified block diagram of the significant internal sections and the required external devices (denoted by dotted lines). One of the most important items to keep in mind is that the data bus ($D_0 - D_7$) is bidirectional both in the 8080 itself and between the 8080 and the bus drivers. In other words, data can flow from memory into any one of several areas (i.e. the instruction decoder or the general purpose registers) and it can also flow in the opposite direction (i.e. from the accumulator to memory). The address bus, however, is not bidirectional. The data on these lines can only go from the program counter or stack pointer to memory.

A good way to become familiar with the way a CPU chips works is to follow it as it goes through a complete cycle. For example, we want to perform the function LDA which means load accumulator. It takes one byte to define this instruction and two additional bytes to define the location in memory of the data to be loaded since the address bus is a 16-bit wide ($A_0 - A_{15}$) bus.

The most convenient point at which to begin it after the CPU has been reset. At this point, the program counter is reset to zero and it places the next name of the first program step (0000) to be executed out to memory over the address bus and fetches the instruction held at this location. The instruction returns to the CPU via the data bus and enters the instruction decoder. The decoder instructs the other parts of the 8080 what to do and when to do it by enabling and disabling various chunks of circuitry according to the decoded instruction and clock signals $\phi 1$ and $\phi 2$. At this point, the control circuitry takes over and the CPU does the function as instructed which might consist of fetching data from memory by the program counter to that memory location. Once it has acquired all the data it needs, it performs the indicated function which — in this case — consists of loading the data into accumulator A. However, it can also perform one of many other functions such as swapping internal registers or adding two sets of data from memory together. After the specified operation has been completed, the result is placed back out onto the data bus and a control signal is generated which informs the proper external device (either memory or a peripheral) that the data is meant for it. The control signal is used to dump the data into memory or into the interface unit of a peripheral device. In the case of Gun Fight, this means entering the information into the latches of the shifter section of that peripheral (the game PCB).

So much for a typical instruction cycle. Before moving on, a few other areas of the 8080 deserve brief attention and these are the registers, the stack pointer, the program counter, the memory, the peripherals, the accumulator and the ALU. Since we have already discussed the instruction decoder and the control circuitry, these areas require no further attention.

The general purpose registers are used as a "scratch pad" memory for the CPU. These registers provide a convenient place for storing intermediate results as the CPU is solving a problem. There are six general purpose registers available to the user and two temporary registers accessible only to the machine itself.

$D_0 \cdot D_7$

BIDIRECTIONAL DATA BUS

ACCUMULATOR
TEMP. REGISTER
ALU
INSTRUCTION DECODER
REG. SELECT
GEN. PURPOSE REGISTERS
STACK POINTER
PROGRAM COUNTER

CONTROL CIRCUITRY

$A_0 \cdot A_{15}$

ADDRESS BUS

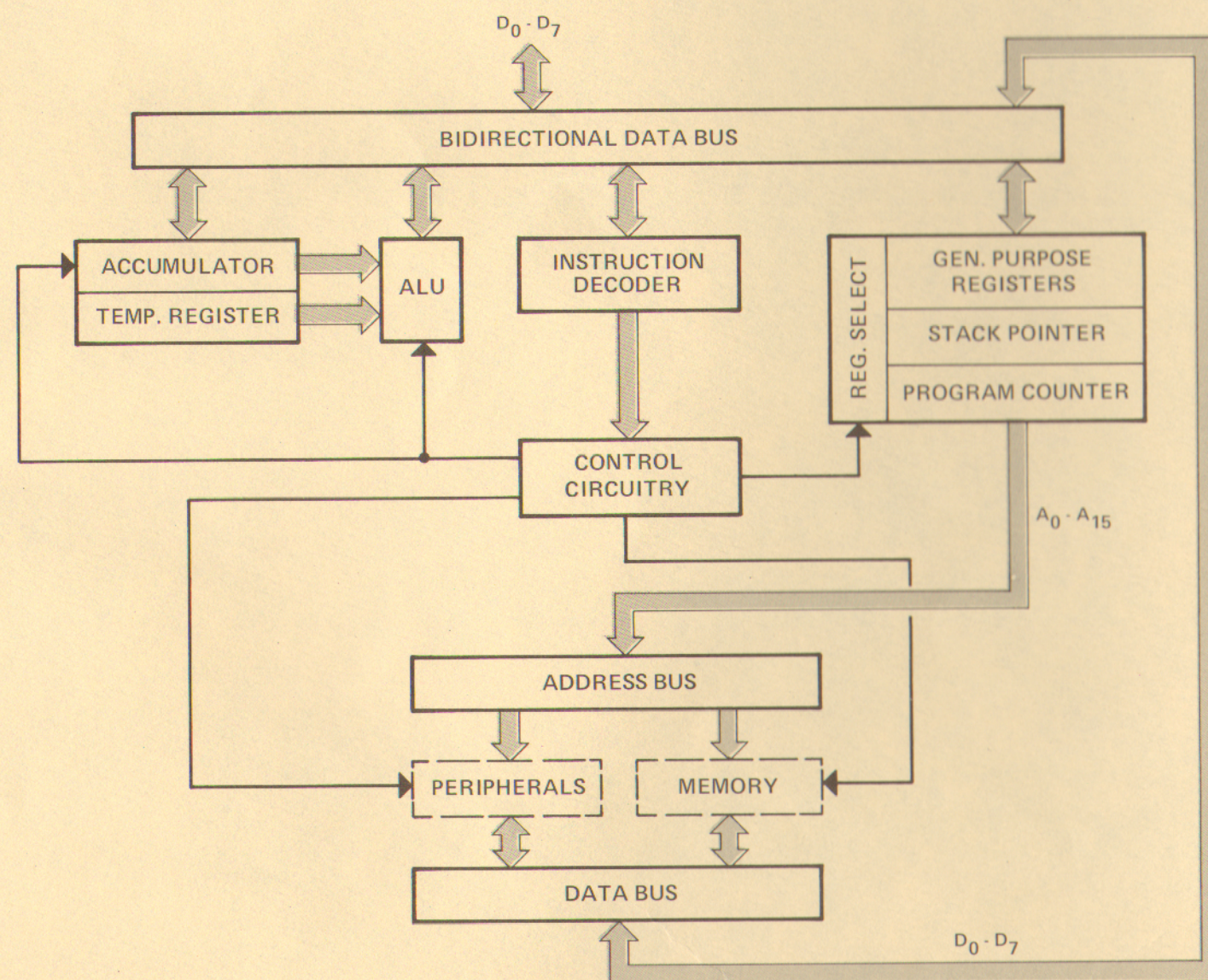PERIPHERALS
MEMORY

DATA BUS

$D_0 \cdot D_7$

Figure 1.5.12-1 8080 CPU Block Diagram

GUNFIGHT

The stack pointer is an area of memory in which the CPU may store information or the program counter so the processor knows where to return after performing a subroutine call. Since the stack is held in the RAM memory, it is essentially unlimited which means almost any number of subroutines may be nested within each other. The stack operates in a first-in, last-out manner which means that the first data entered is placed at the very bottom of the stack.

The program counter is just a regular 16-bit binary counter which outputs memory addresses to find instructions, data or whatever else is stored at the particular location in memory. The program counter is also used to address data from the interface registers of peripheral devices.

The memory in the case of Gun Fight consists of the 8K RAM memory and 4K of ROM all located on the mother PCB. All memory is addressed the same regardless of what kind of memory chip is used. As we have already mentioned, the ROM memory holds the program and the images while the RAM memory is used to hold the next data to be displayed on the CRT as well as providing a place for the CPU to store information on a temporary basis.

The peripherals in a general purpose computing system may include many types of devices used to collect and enter or to output and display data. Some of these include printers and TV terminals which are used to output and display data or keyboards and custom electronic subsystems which are used to input information. However, in the case of Gun Fight, the only peripheral is the game PCB itself. In reality, there are two output peripherals found on the game PCB and one input peripheral. The CPU writes data into both the digital video shifter which controls the images and into the analog sound section which generates and controls the sound. The CPU receives data from the players' controls via the game PCB.

The accumulator and temporary registers are a set of ordinary registers except that they are expressly dedicated to the ALU. Since the ALU must be capable of performing an operation of two arguments (i.e. adding two numbers together), it must have two different registers.

The ALU or arithmetic logic unit is the circuit used to perform the actual arithmetic operation on the number (or numbers). The ALU can also be used to provide the equivalent of logic functions such as ORing two numbers together.

1.6 Game PCB Architecture

1.6.1 General Functions. The game PCB contains a number of input and output peripherals which collect or output data. In fact, there are three output peripherals and one input peripheral where the functions of these subsections are (1) to collect data from the players' controls and operator adjustments, (2) to provide smooth image motion, (3) to generate and control the various sounds and (4) to trip the coin counter. The general relationships of these circuits and the directions of data flow are indicated in the following block diagram (Figure 1.6.1-1). In this part of the system, all data enters via the output data bus ($D_0 - D_7$) from the

mother PCB and leaves the game PCB to the mother PCB via the multiplex data bus ($MX_0 - MX_1$).

1.6.2 Players' Controls. The players' controls are the main input to the entire system. These controls consist of a gun control with trigger and a separate joystick for each player. The joystick controls contain four switches each where each switch represents a single direction for one of the two players and these switches are connected to pins 5, 6, 9, 10, 13, 14, 17 and 18 of the game PCB edge connector. When the left player wants his cowboy to move upward, he presses his joystick forward which closes the line to pin 14 (LMUP). Similarly, if he wants his cowboy to move up and to the left (or at a 45° angle in other words), the lines to both pins 14 and 10 must be closed simultaneously.

The gun arm control is quite a bit different than this simple joystick arrangement. Underneath this control there are three signal wipers (and one ground wiper) which run over a set of pads on a printed circuit card mounted perpendicularly to the control panel. As the wipers make and break contact with these pc pads, a binary code representing the desired position of the gun arm is generated according to the table to the right.

		SIGNAL LINE		
		LG1 or RG1	LG2 or RG 2	LG3 or RG 3
GUN ARM ANGLE	UP 45°	1	1	0
	UP 30°	1	0	0
	UP 15°	1	0	1
	CNT 0°	0	0	1
	DN 15°	0	0	0
	DN 30°	0	1	0
	DN 45°	0	1	1

0 = GND
1 = VLED
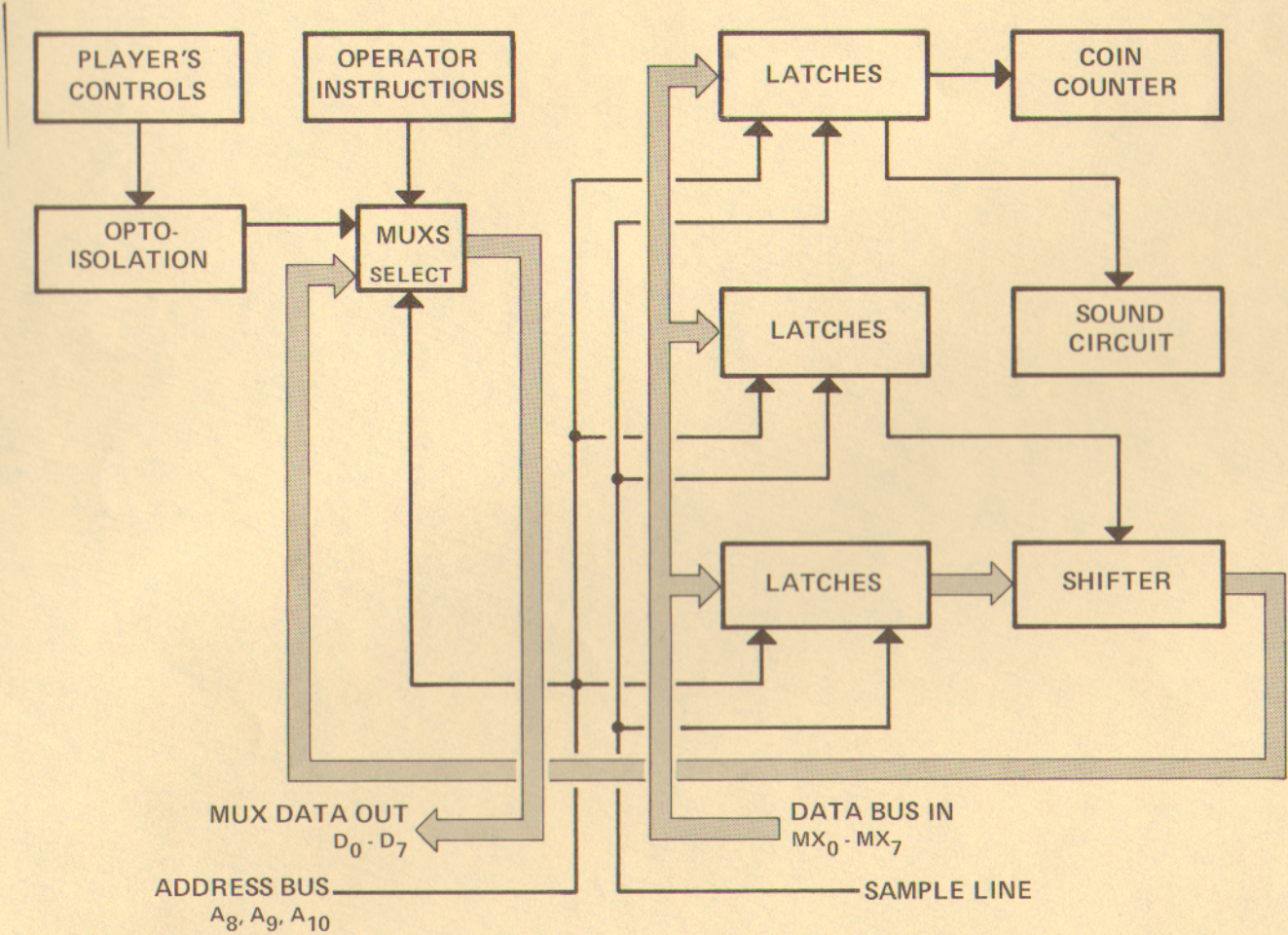
Figure 1.6.2-1 Gun Arm Angle Code

Figure 1.6.1-1 Game PCB Block Diagram

GUNFIGHT

For example, if the player wishes to raise his cowboy's gun arm up to a 45° angle, the code 1 1 0 is generated when the player moves the gun control to the correct position.

1.6.3 The Opto-isolators. These devices are used only to isolate the players' controls from the rest of the circuitry for protection reasons so that no possibly destructive electrical potential can occur. The opto-isolators do not affect the content of the signal line in any way. See Section 1.7.2 for more details concerning the operation of the opto-isolators.

1.6.4 Operator Instructions. These instructions are entered by the operator at the DIP switches located at game PCB position C1 and are used to define the length of the game, the number of plays per game credit and the number of coins per game credit.

1.6.5 The Multiplexers. The multiplexers are included so that the CPU on the mother PCB can request the type of information it needs when it needs it.

There are four different types of information entered into the multiplexers and these are: (1) right player control settings, (2) left player control settings, (3) operator instructions and (4) shifted data.

For example, when the CPU reaches a place in its program where it needs to know where the left player wants to position his cowboy, it sends out an instruction over the address bus which is connected to the select lines of the multiplexers. The code on the address bus at this time selects the left player's control setting inputs and these signals then appear at the mux outputs. The binary code representing the control setting is then passed to the CPU over the multiplex data bus for use by the CPU. By sending out slightly different codes over the address bus, the CPU can request information on the right player's controls or the settings of the operator adjustments.

1.6.6 The Latches. The main function of all these latches is to hold the information coming in on the data bus stable until it can be used by one of the output devices (the shifter, coin counter or sound generator). The data is latched in two passes. During the first pass, the lowest 7 bits are latched and the next eight bits are latched in the second pass. The D_0 bit (the LSB) is dropped in the process due to the construction of the circuit and because it is not needed. The latches are clocked by a special signal which results from NANDing the SAMPLE line with address bit A^{10}. The combination of these two signals allows the CPU to define the output port (the latches in this case) to which it is writing.

1.6.7 The Shifter. The shifter is controlled by one of the latches (E6 to be specific) which simply latches up the three LSBs (D_0, D_1 and D_2) of the data word coming in over the output data bus and uses them to control the shifting process. The correct word coming in on the bus is selected by the combination of the SAMPLE line and address bit A^9 which determines the output device the CPU wants to write into.

Motion in this system is not achieved by the familiar "slipping counter" process normally associated with random-logic video games, rather it is generated by moving data words around in memory. As we have mentioned before, the RAM memory contains a reflection of the CRT in that what is in the RAM at one instant will appear in the next frame of the monitor scan.
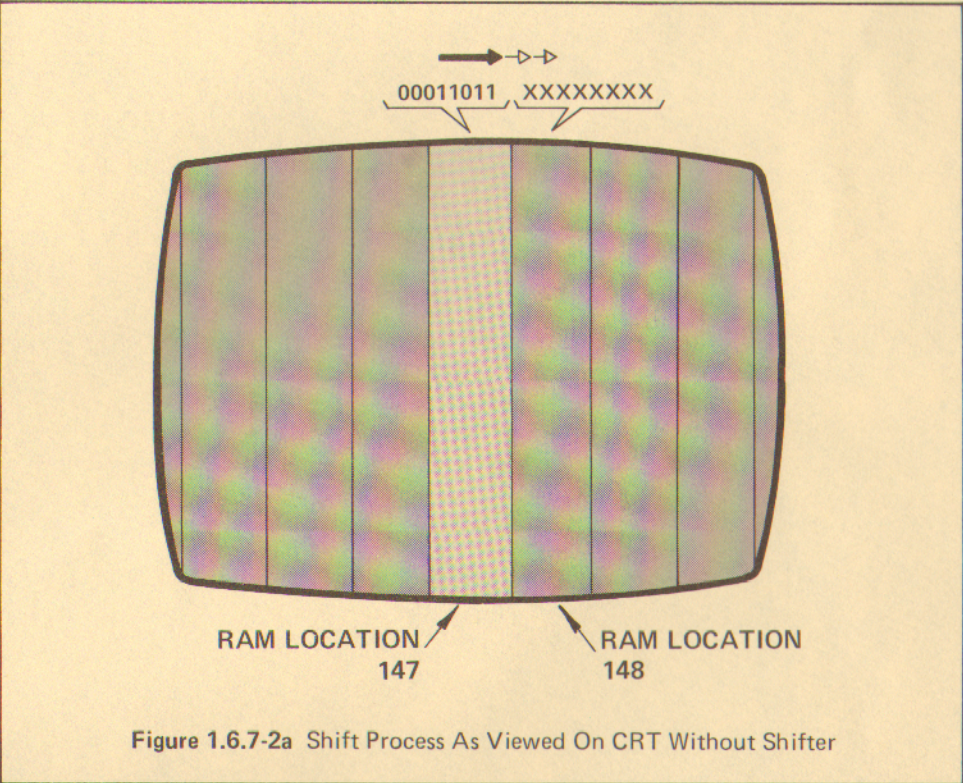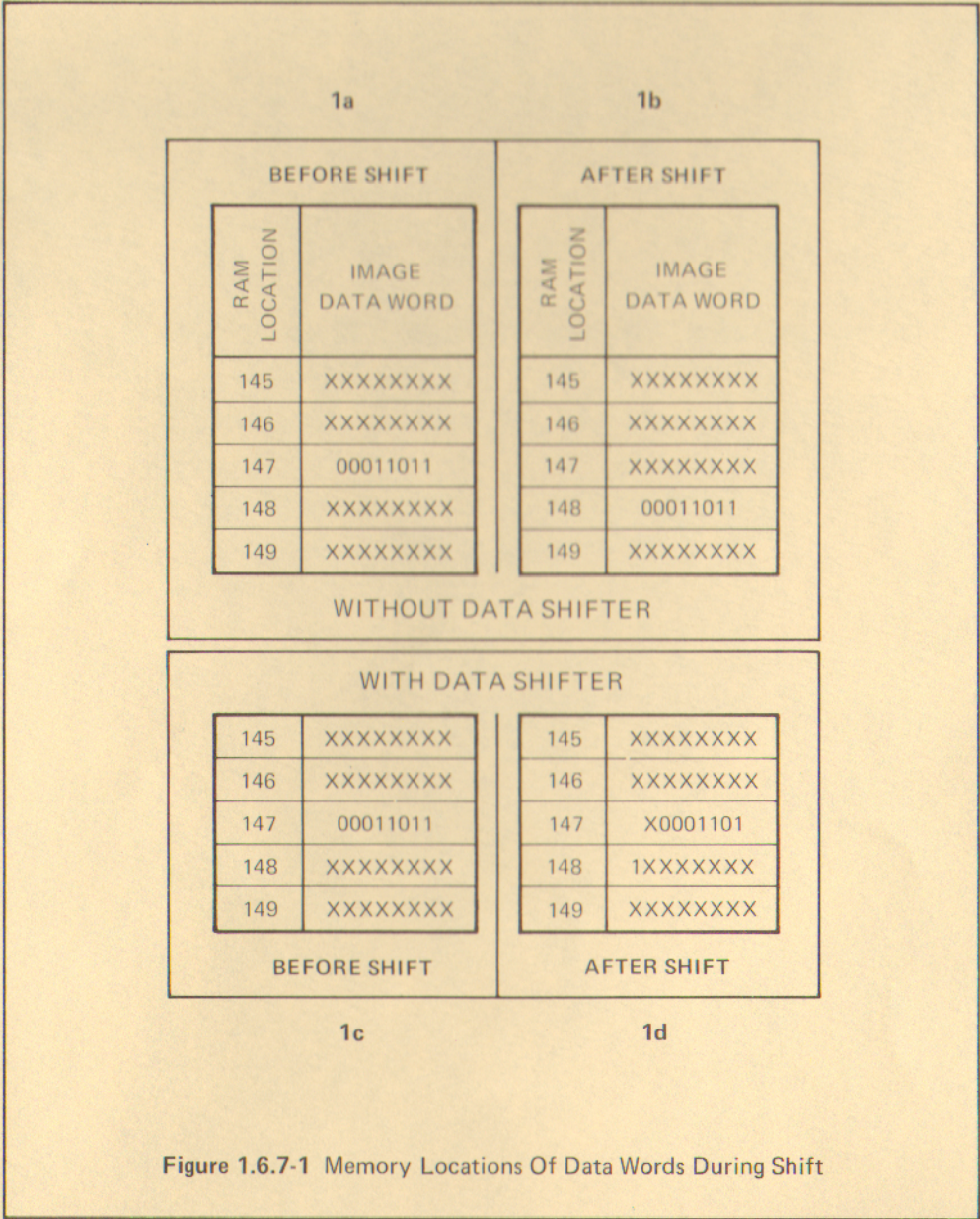
The problem is that the CPU can only move data in memory in jumps of eight-bit blocks due to the construction of the memory system. So a circuit known as the shifter is employed to convert this single eight-bit jump into a series of eight single-bit "hops." For example, let's say we have the data word 00011011 which represents part of an image stored in ROM and we want to move the image to the right on the CRT. Assume also that this word is presently stored in memory location 147 (see Figure 1.6.7-1a). Were a shifter not employed in the game PCB, the CPU would take the word 00011011 and simply dump it into the next memory location, which is 148 (Figure 1.6.7-1b).

Now, when the shifter is employed to move the image data word, an entirely different process results which is illustrated by Figures 1.6.7-1c and d. In this case, the data word is first taken from the RAM memory location 147 and entered into the shifter circuit in the game PCB which shifts it up one bit only and the shifted word is then rewritten back into RAM memory so that the last bit of the word that was in location 147 is now the first bit in location 148. Now that you understand what this process looks like in memory, we shall illustrate the same process using representations of the TV monitor screen.

Each location in memory is represented by a vertical column on the CRT. Although we have illustrated only seven columns, be aware that there are actually sixteen columns extending from one side of the CRT to the other. Notice also that we have labelled two of the columns 147 and 148 in each figure and these numbers represent those areas of memory.

To illustrate what would happen without the shifter circuit, we have included Figures 1.6.7-2a and 2b. Since the CPU can shift data in memory only in 8-bit "leaps", these two figures show that the part of the image (and hence the entire image) will jump over to the right by one whole column each time the CPU tries to move the image.

However, with the shifter circuit the process is actually very smooth and this method is illustrated in Figures 1.6.7-2c and 2d. Whereas the entire word was shifted a whole column in Figures 2a and 2b, the word in Figures c and d is shifted to the right only one bit. You can easily see that the last bit of the word of column 147 in Figure 1.6.7-2c is now in the first bit position of column 148 in Figure 1.6.7-2d. The result is smooth, realistic motion.

1a			1b	
BEFORE SHIFT			**AFTER SHIFT**	
RAM LOCATION	IMAGE DATA WORD		RAM LOCATION	IMAGE DATA WORD
145	XXXXXXXX		145	XXXXXXXX
146	XXXXXXXX		146	XXXXXXXX
147	00011011		147	XXXXXXXX
148	XXXXXXXX		148	00011011
149	XXXXXXXX		149	XXXXXXXX

WITHOUT DATA SHIFTER

WITH DATA SHIFTER

145	XXXXXXXX		145	XXXXXXXX
146	XXXXXXXX		146	XXXXXXXX
147	00011011		147	X0001101
148	XXXXXXXX		148	1XXXXXXX
149	XXXXXXXX		149	XXXXXXXX
BEFORE SHIFT			**AFTER SHIFT**	
1c			1d	

Figure 1.6.7-1 Memory Locations Of Data Words During Shift

00011011   XXXXXXXX

RAM LOCATION 147   **RAM LOCATION 148**

Figure 1.6.7-2a Shift Process As Viewed On CRT Without Shifter

GUNFIGHT

$\overbrace{XXXXXXXX}$ $\overbrace{00011011}$

147 148

Figure 1.6.7-2b Shift Process As Viewed On CRT Without Shifter

X0001101 1XXXXXXX

147 148
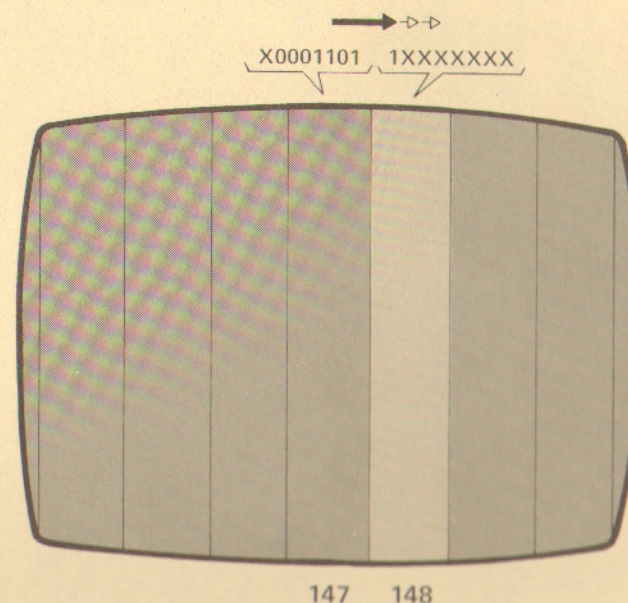
Figure 1.6.7-2d Shift Process As Viewed On CRT With Shifter

00011011 XXXXXXXX

147 148
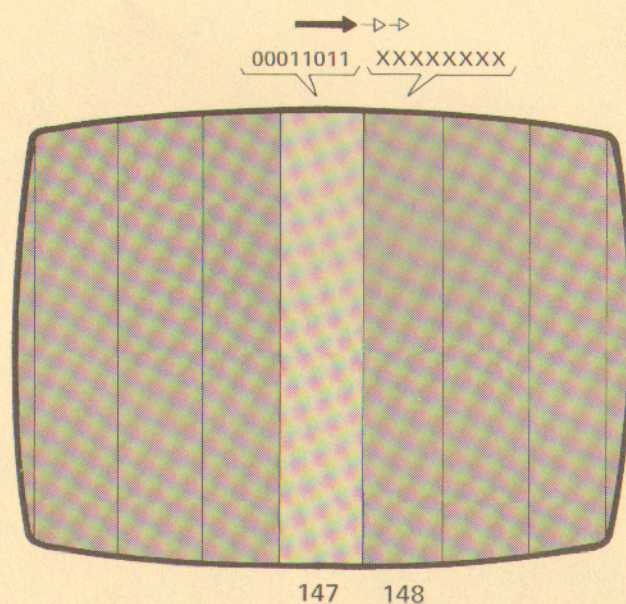
Figure 1.6.7-2c Shift Process As Viewed On CRT With Shifter

(TOP VIEW)

ISO-LIT D74

PIN NO.	FUNCTION
1	ANODE
2	CATHODE
3	CATHODE
4	ANODE
5	EMITTER
6	COLLECTOR
7	COLLECTOR
8	EMITTER

LED CHIPS ON PINS 2 AND 3
PT CHIPS ON PINS 6 AND 7

Figure 1.7.2-1 Basic Multiplexer Logic Diagram

1.7 Operation of Specific Game PCB Components

1.7.1 Introduction. Before delving into the actual theory of operation of the Gun Fight game PCB, a few relevant comments need to be made about the function and operation of the various components from which the game PCB is constructed. Hopefully, the following information will facilitate both the comprehension and troubleshooting of the game PCB circuitry.

Fortunately, the componentry used in the game PCB is ordinary stuff, especially in comparison to the devices located on the mother PCB. Probably, you are already familiar with some of the game PCB devices (the multiplexers, latches and decoders, however a few others (the opto-isolators, Schmitt triggers and data shifters) may not be quite as familiar so we will stress these last items to a greater extent.

1.7.2 The Opto-isolators. Opto-isolators do just what their name implies — they electrically isolate one component from another using the optical characteristics of a light emitting diode (LED). The reason for using these devices is to eliminate any electrical potential which might occur between the control panel and the game computer. An undesirable potential of this nature can arise from a number of causes. A well-known
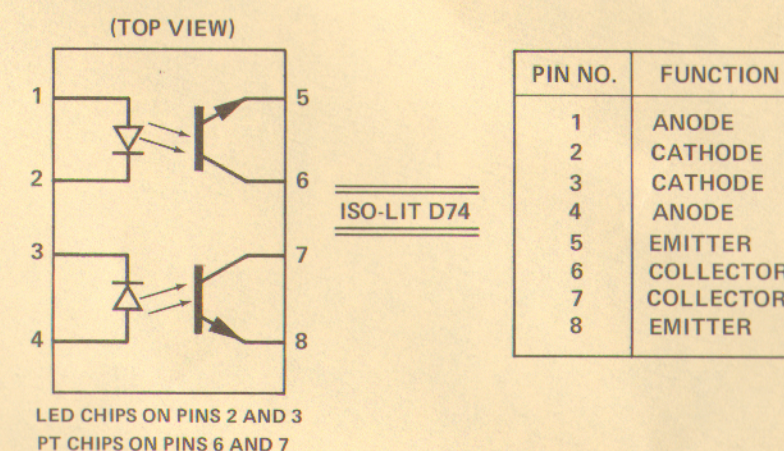
one stems from player accidentally or intentionally releasing large static discharges to the machine which can play havoc with the digital circuitry. Other causes include poor ground connections and operators inadvertently shorting out the wiring while repairing or collecting the machine. The opto-isolator transmits only the data from the control panel, not the actual electrical signal.

Actually, opto-isolators have been around for quite some time, even before the advent of today's remarkable semi-conductor techniques. The earliest types used either incandescent or neon lamps as light sources coupled with cadmium or lead photo-resistors as sensors. Contemporary opto-isolators generally use a gallium arsenide infrared emitting LED as the light source and a silicon photo-transistor as the sensor. The source and sensor are totally enclosed in the device package so the light emitted is not visible and the source cannot be triggered by ambient light.

The type of opto-isolator used in this application is either an ISO-LIT D74 or its equivalent, the MCT-6. This type of device contains two opto-isolators in a single package used to isolate separate control inputs. The control inputs enter the device on pins 2 & 3 or 3 & 4 where they go directly to the LED chip. The outputs come from pins 5 & 6 or 6 & 7 from the phototransistor chips. It is interesting to note that this device is not monolithic; the LED and phototransistor are mounted on different chips.

The control inputs are normally HI and when the control associated with that input is operated, the line drops to ground and illuminates the LED which emits infrared light. This light is sensed by the phototransistor, which then outputs a pulse of constant potential. A separate supply voltage (V_{LED}) is used to operate the LED and notice that this voltage is connected to the side of the LED opposite from the control input. So, when the control input drops to GND, this voltage is allowed through

GUNFIGHT

the LED and turns it on.

1.7.3 The Schmitt Triggers.
These devices operate essentially like 7404 inverters in that the input waveform is inverted, however Schmitt triggers also are used to square up the waveform so it becomes more acceptable to the digital circuitry. Although it is a good idea to use Schmitt triggers in this sort of application, their use is not absolutely necessary and a 7404 may be used temporarily if the 7414 Schmitt trigger is not readily available.

Figure 1.7.3-1 shows how the Schmitt Trigger cleans up a messy input waveform into a nicely squared up signal.
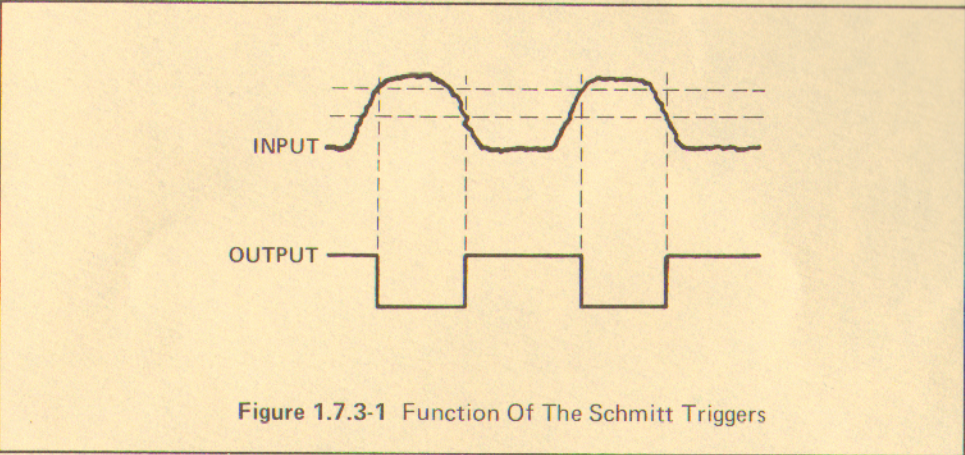

Figure 1.7.3-1 Function Of The Schmitt Triggers

1.7.4 The Multiplexers.
All the multiplexers found on the game PCB are 74153 dual four-to-one devices which means there are two of the same four-input multiplexers in the same chip. In general, multiplexers are used when it is necessary to select one data line from a group of lines so the desired data can be sent to other devices for further processing.

Figure 1.7.4-1 shows the simplest multiplexer, the single two-input. In this case, input data lines A and B are selected by the state of select signal S. When S is HI, the top AND gate is enabled and it passes whatever signal is present at the selected data input. Simultaneously, the HI at S is inverted LO to disable the lower gate and lock out the other data line. But if S drops LO, the top gate is disabled and the second data line is enabled through to the output.

A single select line can only be in one of two states: HI or LO, hence it can select only one of two signals. But if we have four signals from which we must choose one, a second select line is required. Two selects then have the capability of four states: 0 0, 0 1, 1 0 and 1 1 where each different code may be used to select one of four data lines. Figure 1.7.4-2 shows how the simplest four-input multiplexer is constructed and notice that there are two selects (S_0 and S_1) which controls the four input lines (I_0 through I_3).

If both S_0 and S_1 are LO, gates 2, 3 and 4 are disabled since a LO S_0 dis-

---

ables Gates 2 and 4 and a LO S_1 disables Gates 3 and 4. Gate 1 is receiving two HIs because the LOs at S_0 and S_1 are inverted first before being connected to it.
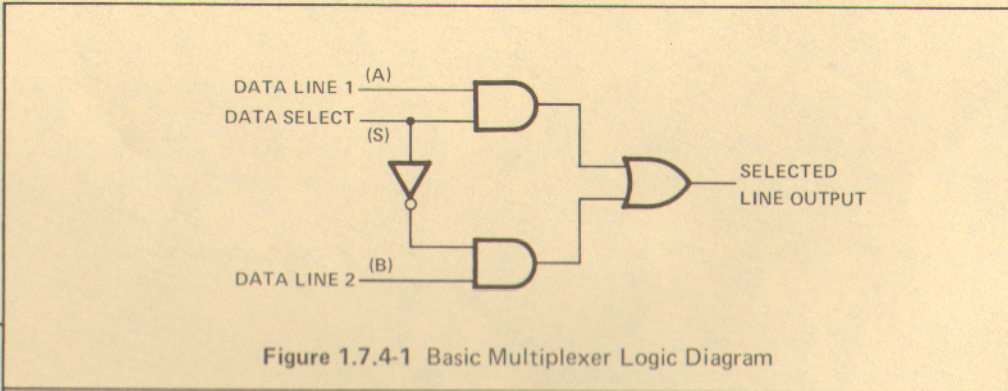

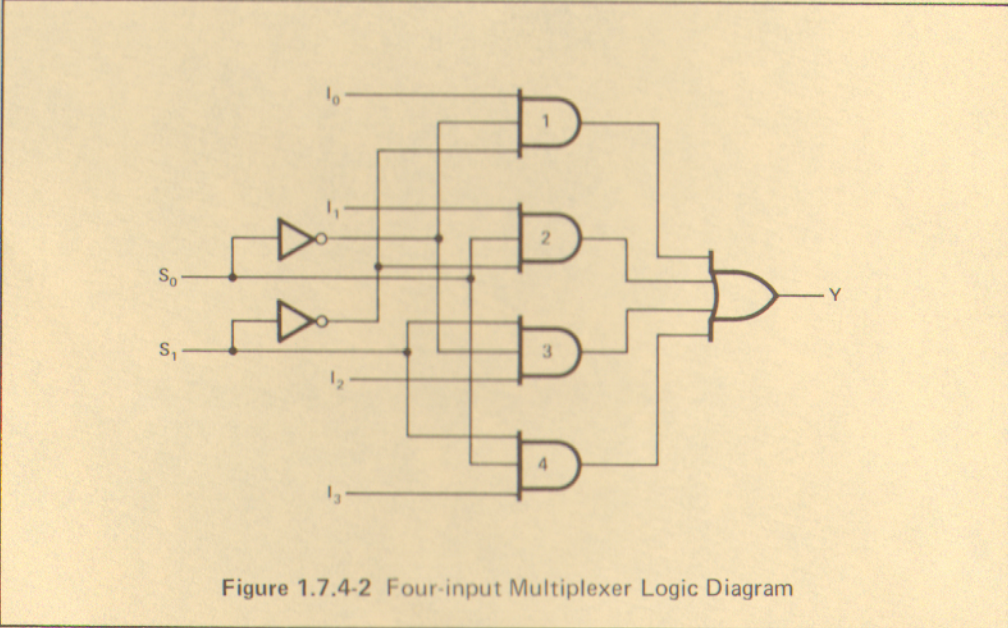**Figure 1.7.4-1** Basic Multiplexer Logic Diagram

Figure 1.7.4-2 Four-input Multiplexer Logic Diagram

Consequently, Gate 1 is enabled and it passes whatever signal is at I_0. The second count occurs when S_0 and S_1 are HI and LO respectively which enables Gate 2 to pass input I_1.

Actual integrated four-input 74153 multiplexers are a bit different. First of all, there are two complete multiplexers per 16-pin DIP. Also, an additional enable input is provided which is connected to all the gates so the entire chip may be turned off. Generally, this enable input is known as a strobe and it may be used to turn off some of the multiplexers in a circuit while others are allowed to continue to operate.

1.7.5 The Latches.
The latches found in this game are very simple devices which consist mainly of four flip-flops. The latches are used in this application to hold data in a stable configuration until it is needed by other devices. When the data has been used and is no longer needed, the latch is cleared. This quad latch can store up to four bits of data at a time, so it can retain a number as large as 1 1 1 1. In reality, the quad latch consists of four single-bit latches connected by common clock and

---

clear lines. Any number of latches may be ganged together to handle larger numbers.
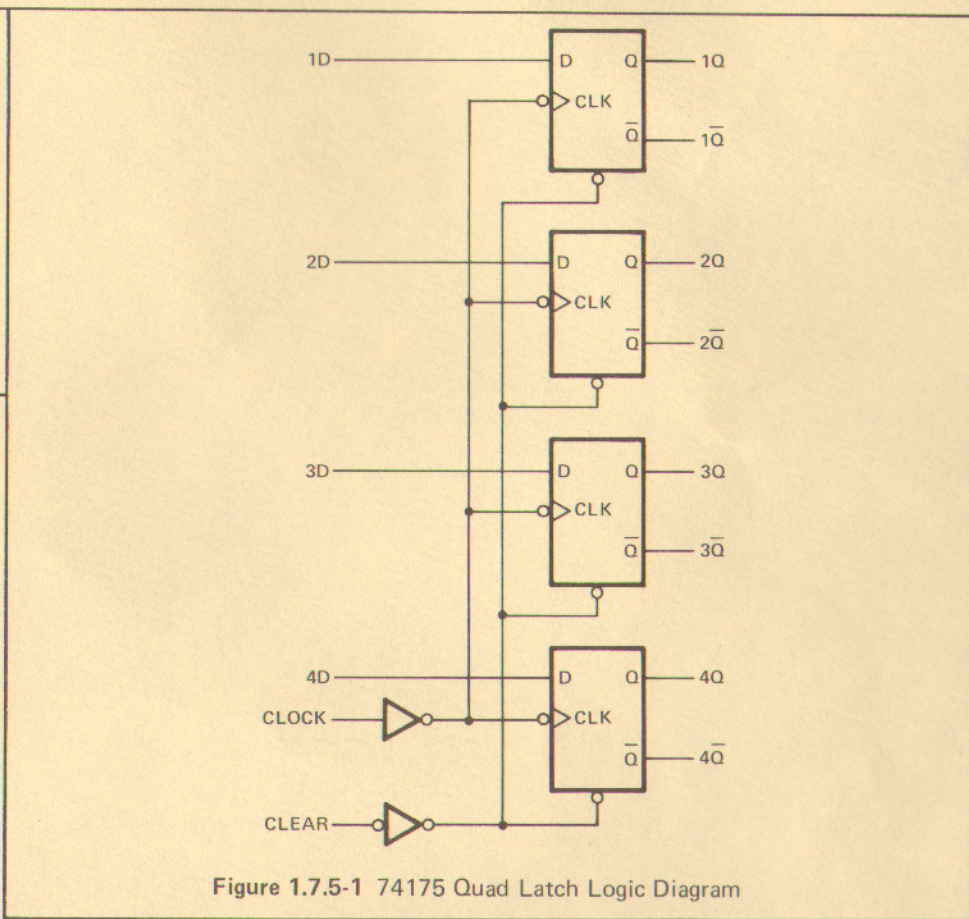
Figure 1.7.5-1 74175 Quad Latch Logic Diagram

The data to be latched is entered at the D inputs to the flip-flops and clocked out when needed by the signal present at the CLK input. Since both the Q and $\overline{Q}$ outputs of the flip-flops are brought out to chip pins, both the true data and its inverse are available. A pulse on the clear line simply resets the Q outputs of all the flip-flops to zero. Since D-type edge-triggered flip-flops are used for the 74175, the data is latched into the device and appears at the devices outputs simultaneously when the flip-flop sees the rising edge of the clock pulse.

1.7.6 The Data Shifters.
The data shifter is used so a four-bit section of a seven-bit word can be selected according to the states of the select lines. In the game PCB, this feature is used to enable smooth motion. By shifting the word a bit at a time, motion in increments of one clock pulse each can be achieved.

A truth table can be used to explain the operation of the device more easily than words. In the table on the next page, you can see that if the select code (at inputs S_0 and S_1) is 00, inputs I_6, I_5, I_4 and I_3 appear at outputs Y_0, Y_1, Y_2 and Y_3. However, if the select code is incremented to the next count (10), inputs I_5, I_4, I_3 and I_2 appear at the device outputs.
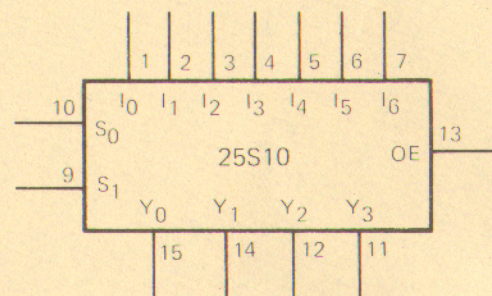
Figure 1.7.6-1 25S10 Data Shifter Pin Configuration

S_0	S_1	Y_0	Y_1	Y_2	Y_3
0	0	I_6	I_5	I_4	I_3
1	0	I_5	I_4	I_3	I_2
0	1	I_4	I_3	I_2	I_1
1	1	I_3	I_2	I_1	I_0

Figure 1.7.6-2 25S10 Truth Table

Figure 1.7.7-1 7442 Decoder Logic Diagram

Since this device has tri-state outputs to enable OR-tying of the outputs of several shifters, an input is needed so that the outputs of one device may be turned off while the outputs of the other shifter(s) are enabled to be the valid ones. Otherw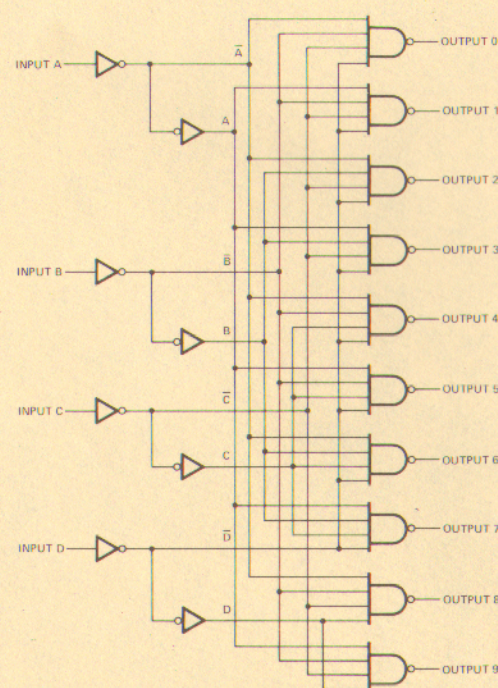ise, the outputs of the devices would be trying to pull each other. So, an output enable is used for this function. When this input is at the LO logic level, the device outputs are valid. But when OE rises HI, the outputs are turned off.

1.7.7 The Decoder. The only digital integrated circuit not yet discussed is the 7442 one-of-ten decoder found at the very right edge of the game PCB schematic. This type of decoder simply converts a BCD (Binary-Coded-Decimal) number to a purely decimal equivalent. By binary-coded decimal, we mean a binary number which cannot be greater than decimal nine or 1 0 0 1. If the binary number increments past 1 0 0 1, it becomes 0 0 0 0 again.

For example, if the BCD number at the device inputs is 0 0 0 0 (decimal 0), output 0 is selected and drops LO. When the BCD input is incremented to 0 0 0 1 (decimal 1), output 1 is selected and it then drops LO.

SECTION II THEORY OF OPERATION

2.1 The Power Supplies

2.1.1 Introduction. This power supply has been designed in such a way that it can be used for all the games in the series for which the CPU mother PCB was designed. Although this game does not draw the full amount of power these supplies are capable of generating, other games in the series do so the supply must be designed to operate the biggest power hog. The power PCB generates quite a few different voltages, both regulated and unregulated, used to power a number of different types of componentry. The +15V, +12V and -5V supplies utilize integrated voltage regulators in a configuration common to video games to generate closely regulated voltages used to power the integrated circuits and other components as well. The +5V supply, however, is a relatively high current source carefully regulated by a standard fold-back, current-limited circuit design which compensates both for line voltage and load fluctuations. In addition to all these regulated supplies, two unregulated supplies are also created to power incandescent lamps (not used in this game) and the LEDs of the opto-isolators.

2.1.2 SENSE and COM Lines. One interesting feature of the power PCB which is not found in other games is the use of the SENSE and COM lines to detect any IR drop which might occur in the ground line between the power PCB and the mother PCB. IR drops across a ground line can cause several problems, including the annoying hum bar rolling up the TV screen which is so infuriating to operators and players. Essentially, both the SENSE and COM lines are ground lines. Notice in the wiring diagram that the SENSE and COM lines are connected together just before they enter Pins 20-21 of the mother PCB. Since the SENSE line is connected to the COM line at this point, it can be used to sense any IR drop which might occur between Pin 14 of the power PCB and Pin 15 of the mother PCB. Since the SENSE line is used as the GND reference for the entire power PCB, any IR drop which occurs across the COM line simply offsets the entire power supply system by that amount and thereby eliminates any problematic conditions which might otherwise occur.

2.1.3 The Transformer. The transformer reduces the 117VAC line voltage down to two center tap voltages: 9V and 16.5V. The 19.5V winding is a "high tap" used only if local power is consistently too low to operate the system. In this case, the wires from the 16.5V winding are unsoldered and placed on the 19.5V winding. The transformer also has a 14V secondary winding used directly +0 light a number of display lamps as well as being further processed for V_{IND} and V_{LED}.

2.1.4 The +12 Source. The 16.5VAC waveform from the transformer secondary is full-wave rectified by the two 1N4004 diodes and filtered by the $6000\mu F$ capacitor before the waveform is placed at the input of the LM340 T-12 integrated voltage regulator. The sense line is used as the GND reference for the regulator to compensate for any IR drop in the ground line as discussed previously. Adjustment provisions have been made by incorporating a voltage divider network composed of the 2700Ω resistor and the 100Ω trimpot. By adjusting the trimpot, the regulator can be further offset from SENSE GND to compensate for any minor deviation. The resulting voltage is further filtered by the $0.1\mu F$ capacitor to eliminate fast transients. The resulting closely regulated +12V source is made available to the rest of the machine through Pins 1 and 2 of the power PCB edge connector.

2.1.5 The +5V Source. The 9V center-tap winding of the transformer is first full-wave rectified by the two 1N5624 diodes and then filtered by the $20,000\mu F$ capacitor. This waveform is placed at the collector of the 2N3055 pass transistor and more or less of this voltage is allowed through depending on load and other factors. The actual regulator in this circuit is the LM305 which operates a 2N2905 amplifier transistor. The LM305 senses the output voltage across the 270Ω resistor and uses this waveform to control the base of the amplifier transistor which is necessary in this circuit to provide sufficient current to operate the pass transistor. If the LM305 senses a drop in voltage, it turns on the amplifier transistor which in turn activates the pass transistor. When the pass transistor is activated, it allows more voltage through to the output to compensate for the drop in voltage. The circuit also senses the amount of current across the .18Ω resistor and, if the current exceeds the safe limit determined by the value of this resistor, the amplifier transistor is shut off which turns the pass transistor off, thereby limiting current to a safe level.

2.1.6 The -5V Supply. The 9V winding is again full-wave rectified, but since the cathodes of the two 1N4001 diodes are wired to the transformer secondary this waveform is negative with respect to GND. This neg-
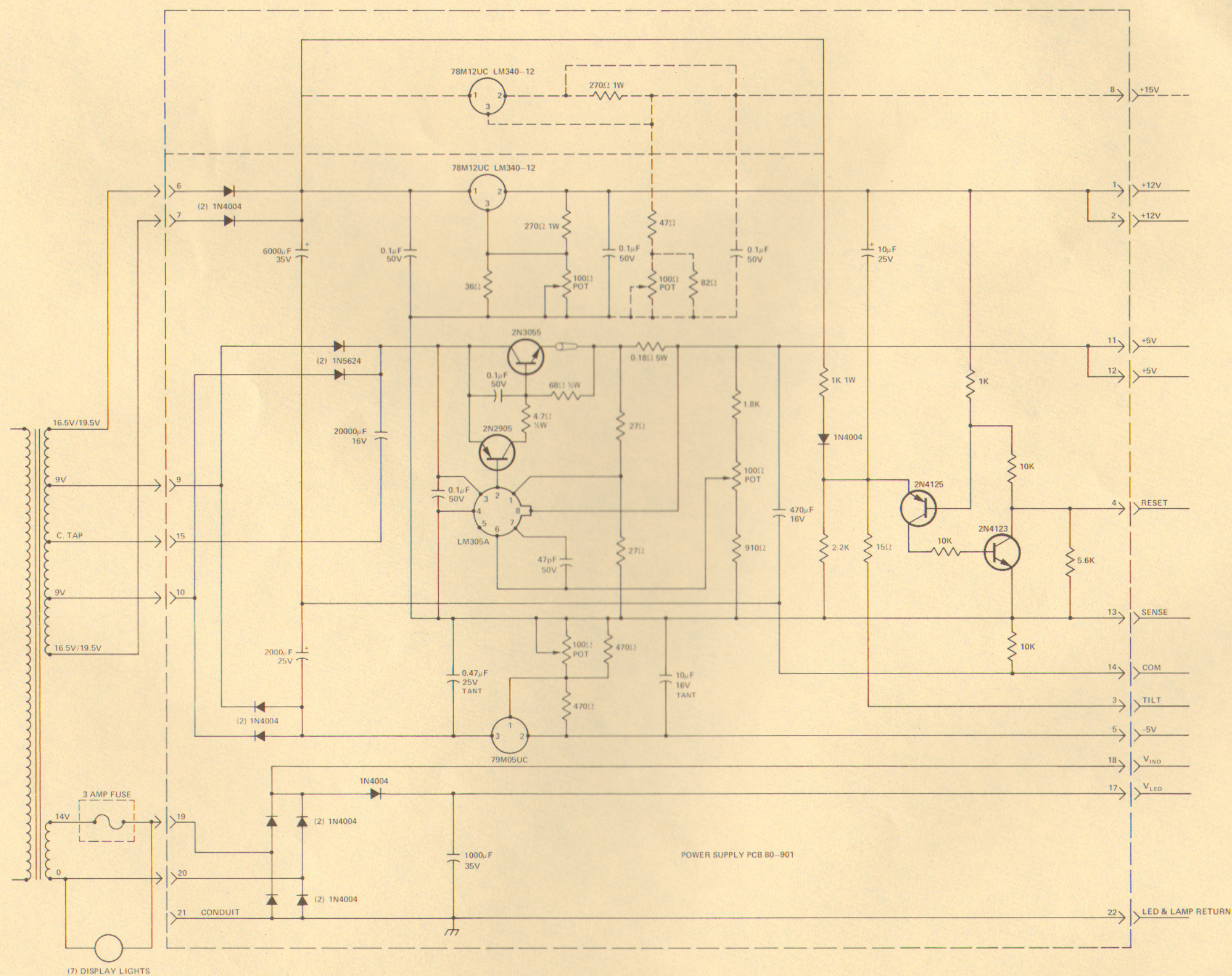
GUNFIGHT

Figure 2.1.1-1 Power Supply Schematic

ative waveform is filtered by the 2000µF capacitor and notice that the other side of this and all the other filter caps are tied to COM ground. This waveform is then placed at the input pin of the LM320 T-5 integrated voltage regulator which can be further offset from GND by adjusting the 100Ω trimpot to compensate for any minor deviation from the specified -5V level. The resulting fully regulated voltage is further filtered by a 10ΩF capacitor to prevent load fluctuations from disturbing the operation of the regulator.

2.1.7 The RESET Line. This part of the power PCB provides a power-on-clear signal when the game is first turned on. Since random information loaded into many parts of the computer when power is first applied, this signal is necessary to clear this meaningless data away so the computer can start operating with a "clean slate."

The operation of this part of the circuit is really quite simple. When the game is first turned on and all the supplies come up, the 10µF capacitor connected to the emitter of the 2N4125 starts charging through the 1K resistor. Until the cap is fully charged (which takes about 1ms), there is a higher voltage at the base of this transistor than at the emitter so the transistor remains shut off. Since this transistor is shut off, the 2N4123 is also turned off and the reset line is pulled up to the HI logic level which resets the 8080 and clears the latch array on the Game PCB. But, when the cap has been charged to such an extent that the emitter is more positive than the base, the transistor is turned on which activates the 2N4123 so the reset line drops LO and allows the system to begin operating.

2.1.8 The Lamp Supplies. This area of the power PCB generates V_{IND} which may be used to power certain lamps and also V_{LED} which is the voltage necessary to power the infrared LEDs in the opto-isolators found in the game PCB. The V_{IND} voltage is developed by full-wave bridge-rectification of the 14V secondary winding and this pulsating waveform is taken directly to the lamps when used. Although this source is not used in Gunfight, it is used in other games of this series where certain lamps need to be controlled by the logic. Since a pure AC voltage is not efficiently switched by a transistor, it must first be rectified. The V_{LED} voltage is developed by the same initial process used to create V_{IND} however it is further filtered by the 1000µF capacitor before it is sent to the opto-isolators.

2.2 Operation of the Game PCB Digital Circuitry

2.2.1 Introduction. In this section, the game PCB is discussed as a functional part of the processing system as a whole. Discussed are four main areas of the Gun Fight game PCB as they relate to the handling of CPU data and these areas are:

- Output Data: This area consists of the latches and associated circuitry for data to be output from the system to the sound generator and the coin counter.

- Video Data: The shifter circuit is a system for smoothing horizontal motion.
- Input Data: This area controls which input information is passed to the CPU for further processing.
- Audio Data: The sound system is an output peripheral which controls and generates all game sounds.

This section will treat these areas in such a way as to lend understanding both to the mechanics of the actual data flow as well as the controls implemented by the processor to execute that flow. Hopefully, the reader will gain an insight to the roles that the four board functions play in the processing system as a whole.

2.2.2 The Game PCB as an Output Peripheral. There are two main devices in the game PCB which may be considered as external to the processing system: (1) the coin counter and (2) the sound generator.
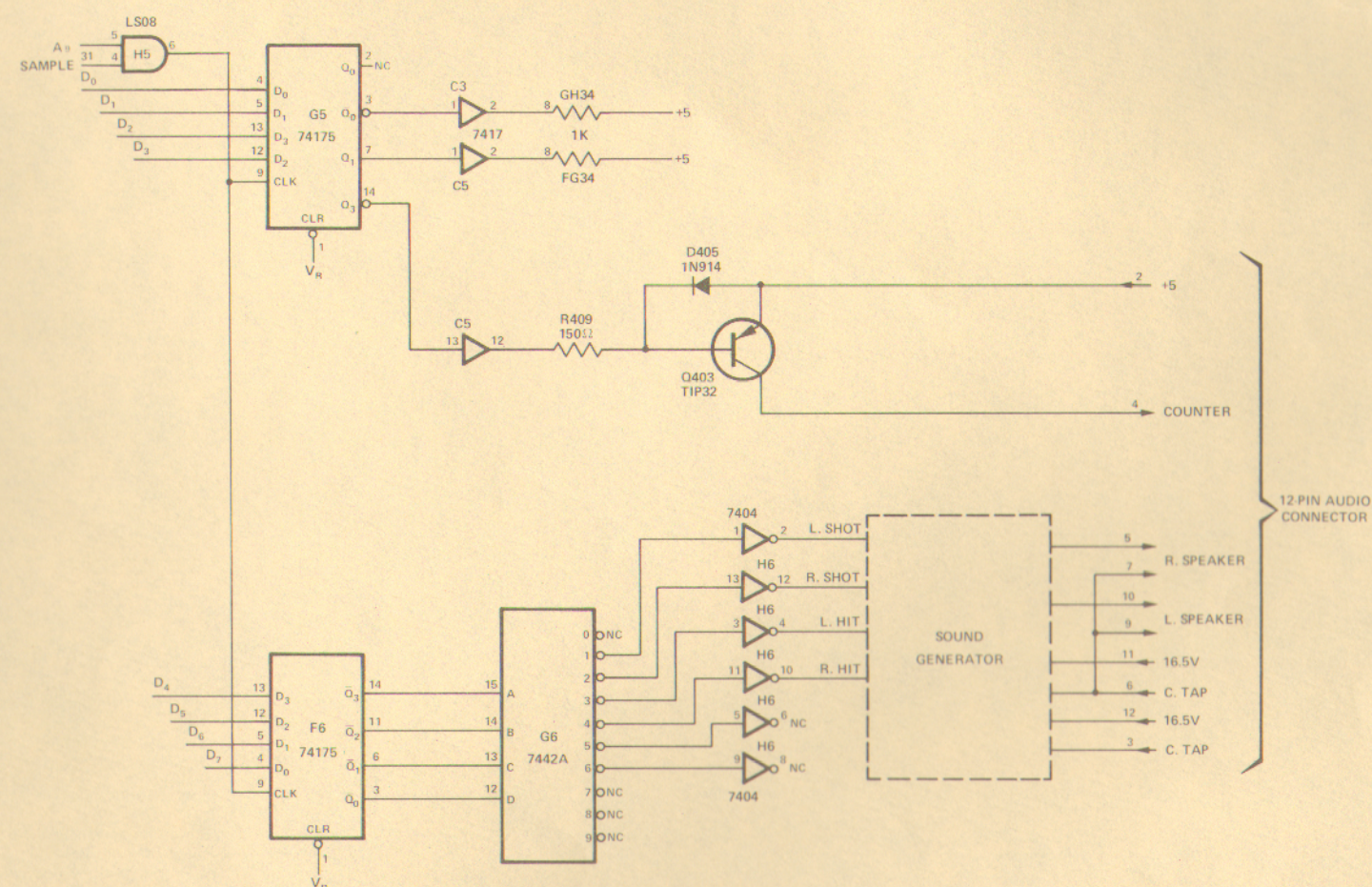

Figure 2.2.1-1 Game PCB Output Devices

Both of these peripherals must be kept under rigid processor control to insure that they will perform their functions at the proper time and — in some cases — for a required length of time. This necessitates two blocks of circuitry and these are indicated by the grey areas in Figure 2.2.2-1. First, a circuit is needed to inform the peripheral which of its functions it is to be performing at any given time and second, another circuit is required which is capable of transforming the signals used by the CPU into a format usable by the peripheral.

2.2.3 The Output Data Latches.

Data intended for the peripheral devices — like all data in the system — is placed on the output data bus and arrives at the game PCB as data bits D_0 - D_7. Since these lines are used for a great many functions, it is impractical to simply place the output data out on the bus and leave it there until the desired peripheral function has been completed for this would consume far too much time. Therefore, it is necessary to get the data to the device for which it is intended and free the data bus as quickly as possible so it can continue to perform other processor duties and this is the function of data latches F6 and G6. It is the responsibility of these two chips to sample the data bus lines when output data is present and hold it unchanged until instructed to sample again by the processor. This way, data can be placed at the inputs of the peripheral devices for as long as is required while only tying up the data bus for a short period of time.

The control signal which informs the latches that output data is on the bus and should be sampled is generated by pin 6 of NAND gate H5 and is placed at pin 9 of the latches. The inputs to this NAND are address bit A_8 and the SAMPLE signal from the mother PCB. SAMPLE, the second input to the control NAND, is gated signal derived by the mother PCB circuitry expressly to indicate to the game PCB circuitry that data on the bus is destined for the game PCB and should be captured by one of several banks of latches and F6 and G5 form one of these banks.

The output of NAND gate H5 is normally HI, but is driven LO when both A_8 and the SAMPLE signal are HI and, at this time, the control signal to the latches may be thought of as "cocked." Since these latches are rising-edge triggered, data cannot be entered until the control signal at pins 9 again rises to a HI logic level and, while this signal is still LO, the data bus must be prepared by removing the data from memory and placing it on the bus so it is present at the latch inputs. When these preparations have been completed, SAMPLE drops LO again, the output of H5-6 rises HI and data on the bus is latched up. This data is then held in an unchanged state at the latch outputs until the CPU again goes into a sector of the program in which A_8 is HI and again tells the game PCB latches to sample.

2.2.4 Decode and Interface.

Once the 8-bit data word is latched into F6 and G5, the information contained in the word must be made both intelligible to the peripheral and electrically compatible with it. We will discuss these aspects separately for the two peripheral devices.

Since the coin counter is a simple on/off device, a single bit at a HI or LO level is sufficient to control its function. A HI bit latched into G5-12 (data bit D_2) will increment the counter once by energizing its coil, however a LO bit must then be clocked into the latch on the next output data latching cycle to de-energize the coil and permit the counter to be incremented again when the next coin is deposited. Since the coil of the counter requires much more current than is available from the latch output, amplification and level conversion is required and note that two levels of current gain are provided.

The HI latched bit is inverted as it leaves the latch output G5-14 and enters the buffer at C5-13, which can provide more current than the latch, but still not enough to power the coil. The second stage of gain is provided by driver transistor Q403 where a LO on the base of this device will turn it to the ON state and provide sufficient current to the coil. Conversely, if a LO level is latched into G5-13, it will be inverted HI as it exits the latch and amplified by the buffer. This amplified HI turns the transistor off and, deprived of its current source, the counter is de-energized and is ready for the next coin to be deposited.

A slightly more complicated situation exists in the case of the sound system. Originally, provisions were made for five separate sounds, however only four are used in the actual system and only four of the data bus lines (D_4 - D_7) are available for the selection of the particular sound to be heard (data bits D_0 - D_2 are used elsewhere on the game PCB and are not utilized as output data although they are latched at G5). At any rate, this requires that information arriving at the latch in the form of a four-bit half-word be decoded into five separate bits and this function is performed by one-of-ten decoder G6.

The four data bits, instead of directly turning on the particular sound, are used as a binary address to turn on one of the ten decoder outputs, four of which are used to control sounds. This addressing procedure for the sound decode is illustrated by Figure 2.2.4-1. A LO output from G6 (only one output can be LO at any given time) is inverted by one of the H6 inverters and simply turns on the specified sound. The sound will continue indefinitely unless a new sound is addressed or until all zeroes are latched into the address decoder.

SOUND	D_4	D_5	D_6	D_7
LEFT SHOT	0	1	1	1
RIGHT SHOT	1	0	1	1
LEFT HIT	0	0	1	1
RIGHT HIT	1	1	0	1
NONE	ALL OTHER CODES			

Figure 2.2.4-1 Sound Decode

2.2.5 Video Data and CRT Motion.

Before delving into a detailed discussion of the shifter circuit, it would be well worth the time to briefly review some of the fundamentals of the processor's handling of video data and, specifically, its ability to simulate motion on the TV screen.

Video patterns, such as the cowboy, cactus and pine tree, are permanently stored in the ROMs in the form of 8-bit words. A block of 8-bit words, with the proper bits programmed HI or LO, may be made to generate a pattern resembling a real-world object when transferred to the monitor CRT in an orderly, sequential way and Figure 2.2.5-1 illustrates how this is done.

At the initialization of the program, these blocks of data words are transferred from the ROM memory and loaded into the RAM memory where most are stored throughout the game. The initial address in RAM, hence the initial position in which the image will appear, is fixed by the program. Consequently, the images will always appear in the same locations each time the game is started or powered up. Some images (i.e. the pine trees and cacti) retain their initial address location in RAM for as long as they appear on the screen and are simply read out each frame at the same location. Others (i.e. the cowboys and the stagecoach) are movable by changing the location of the video pattern in RAM so that their positions on the CRT changes with successive frames.

Vertical motion is a relatively simple process, at least when compared to horizontal movement of the images. When reading from or writing to RAM, processor addresses A^5 through A^{12} may be thought of as a vertical address for lines on the TV screen, where address 00000000 is the extreme top edge of the CRT. If the processor takes a word from a given location in RAM and rewrites it back into RAM at a different address location, the video pattern corresponding to that word will appear to have shifted up or down on the CRT when read out by the electron beam. The direction of the shift is a function of whether the new address location in memory is greater or smaller than the old one. Since the line numbers of the raster scan increase toward the bottom of the screen, greater memory addresses produce a downward shift. The amount of position shift is one raster line for each unit change in the vertical address.

While the actual mechanics of moving a complete video image up or down the CRT in a coutinuous and orderly way is a bit more complex than the simplified description above and requires a great deal of processor attention as well as program instructions, the principle involved should be fairly clear. A video image is nothing more than a coherent block of words. In order to move the image as a block, each individual word must be read out of RAM and written back into RAM in a new location. As long as each word in the block is moved the same number of address units, the image will remain coherent and, as long as the amount each image is moved is not excessive or unsteady, the image will appear to move in a smooth manner.

Vertical motion is conveniently produced by the CPU itself and requires no other external circuitry. Horizontal motion, on the other hand, is
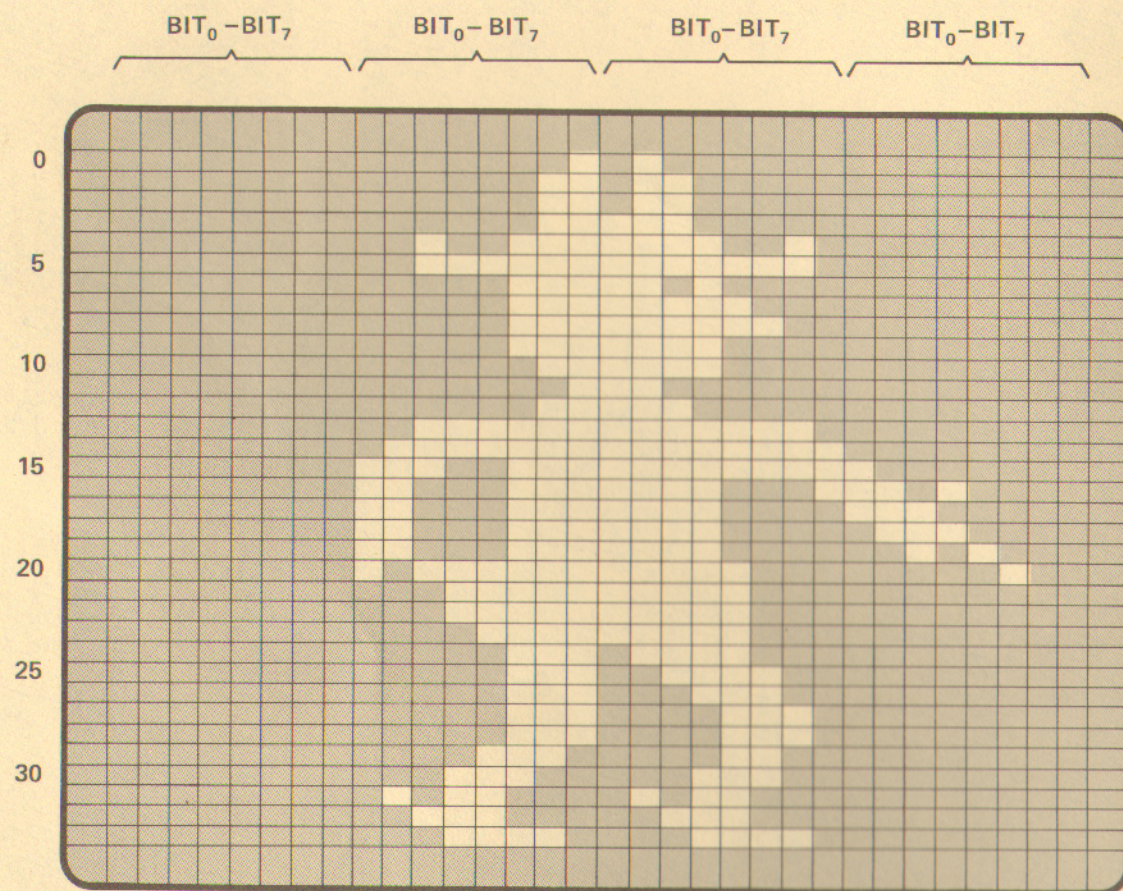
GUNFIGHT

BIT$_0$–BIT$_7$ BIT$_0$–BIT$_7$ BIT$_0$–BIT$_7$ BIT$_0$–BIT$_7$

Figure 2.2.5-1 Video Image Generation

a more complex process party because additional circuitry is required to effect smooth, realistic motion. However, the basic principle of horizontal motion remains the same. Processor address bits A^0 through A^4 may be thought of as horizontal screen addresses where each new address number represents one of sixteen word locations across the scan line. However, each word location identifies eight bits of data in RAM and hence eight "spots" on the screen stretching across the scan line for about ½ inch. If the processor were to take a word out of RAM and rewrite it back in at a new address, the word will not appear to have moved one bit location across the screen but rather eight bit locations or almost ½ inch. Again, the direction of the shift is dependent on whether the location of the new address in memory is greater or less than the old address location. In this case, greater address locations cause shifting to the right because the electron beam scans the raster starting from the left and proceeding to the right.

In principle, there is nothing wrong with this method of data transfer and in fact this is what the processor does do to move the image. However, if the image actually moved in eight-position jumps, the effect would be jerky, uneven motion. So, a block of circuitry has been designed to take the processor's eight-position location jump and generate eight in-between shifts of one position each and thereby smooth out the jerky, discontinuous motion produced by the processor.

2.2.6 Shifter Circuit Logic. The shifter circuitry is indicated by the grey areas in Figure 2.2.6-1 and it consists of three subsections which are: (1) a 15-bit latch composed of A6, B6, C6 and D6, (2) an addressable shifter at A3, B3, A5 and B5 and (3) an address latch at E6.

The 15-bit latch operates in very much the same way as the output data latch described in Section 2.2.3. The control signal is generated at H5-11 by the NANDing processor address bit A^{12} and the SAMPLE line. When both these signals are HI, data destined for the first eight bits of the 15-bit latch are placed on the data bus lines. The subsequent fall of SAMPLE drives the NAND output HI and the next rising edge admits the 8-bit word on the bus into the first eight locations of the latch. The second set of latches consist of seven bits and do not connect to the data bus. Rather, the data to be latched into these positions is the word writ-

ten into the first eight bits of the latch on the previous latching cycle. During the first latching cycle, an 8-bit word — call it "word 0" — is latched into the first eight bits of the latch. During the next latching cycle, a new word — call it "word 1" — will be admitted into the first eight positions of the latch while seven bits of word 0 will be transferred to the other seven locations in the latch. Note that the least significant bit — bit 0 — of word 0 is not transferred to the second part of the latch. It is simply lost. Similarly, on the next latching cycle word 2 will be latched into the first word position of the latch while word 1 is transferred into the second set of seven latches. Bit 0 of word 1 is also dropped.

The outputs of this 15-bit latch are then placed at the inputs of the 25S10 shifter chips (see Section 1.7.6). These devices function in much the same way as a multiplexer with one significant difference. Instead of the address and enable lines determining which input is reflected by the output, the address lines and enable of the shifter determine which four of the seven inputs are output from the four shifter output lines. Any combination of four adjacent inputs may be selected by one of the four possible states of S_0 and S_1. Since two of the shifters are enabled by S2 and the other two shifters by S2, two of the four shifters will always be enabled at any given time so that an 8-bit word will be generated from eight of the fifteen bits held in the latch. The latch outputs are wired to the shifters in such a way as to allow any eight adjacent bits to be output as the shifted word, depending on the particular address at S_0 and S_1. Figure 2.2.6-3 shows which eight bits of the latch are output for any particular address and enable combination.

The address latch E6 captures three bits off the data bus (D_0, D_1 and D_2) when its clock input (pin 9) rises HI. These three bits are under program control and determine the address code to be presented to the shifters. The clock for the device is output from H5-8 as the NANDed signal of processor address bit A^9 and SAMPLE. As before, data intended for use as shifter address code will be placed on the first three bits of the data bus during the time A^9 and SAMPLE are HI (H5-8 is LO). When either of these two signals returns LO, the signal at pin 9 of the latch will rise HI and the data will be latched into E6. The data thus latched will select a new address code for the shifters.

2.2.7 A Typical Shifter Sequence. The foregoing describes the operation of the individual shifter circuit sections but it does not relate the shifting process to the CPU or the other processor circuitry. In order to tie these areas together, we will analyze the following hypothetical example and take a step by step walk through a typical shifting process.

Suppose we have an 8-bit word stored in a certain RAM location, say horizontal address 3. This 8-bit word is displayed on the CRT by being read out of RAM once each frame. Since it is in RAM location 3 it will appear at the far left of the screen where bit 0 is the LSB and occupies the leftmost position in the word. We will also assume that we need to transfer this data word one position to the right, or to horizontal address 4. We could do this simply by reading the word out of RAM location 3 and writing it back in at location 4, but this would appear as a discon-

GUNFIGHT
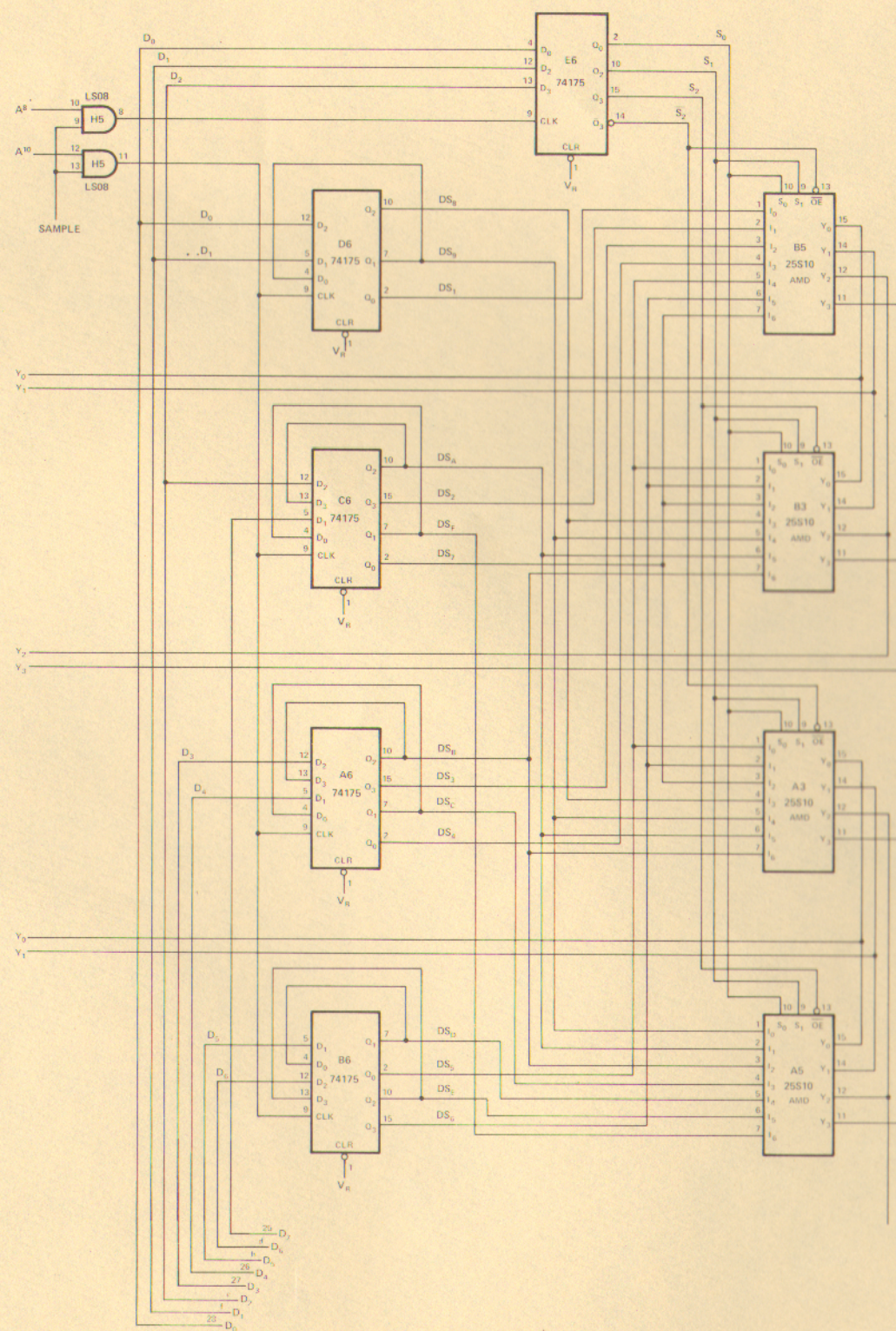
Figure 2.2.6-1 Game PCB Shifter Section
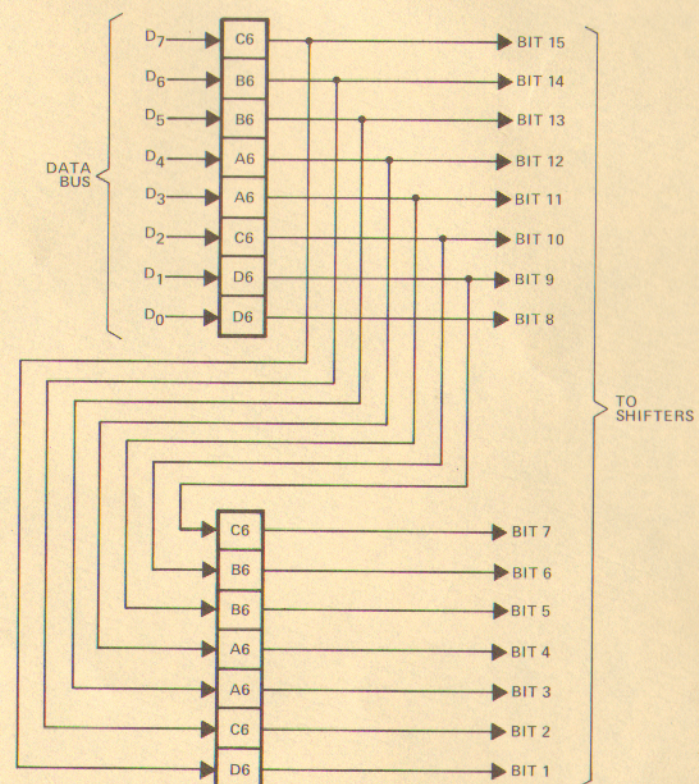
Figure 2.2.6-2 15-Bit Latch Operation

Figure 2.2.6-3 Data Shifter Truth Table

SHIFTER ADDRESS			WORD OUTPUT							
S_0	S_1	S_2								
0	0	0	8	9	19	11	12	13	14	15
1	0	0	7	8	9	10	11	12	14	14
0	1	0	6	7	8	9	10	11	12	13
1	1	0	5	6	7	8	9	10	11	12
0	0	1	4	5	6	7	8	9	10	11
1	0	1	3	4	5	6	7	8	9	10
0	1	1	2	3	4	5	6	7	8	9
1	1	1	1	2	3	4	5	6	7	8

GUNFIGHT

16

tinuous eight bit jump when viewed on the screen. Instead, we will use the shifter circuit to move the data word a bit at a time to the right during each frame and accomplish the entire transfer in eight frames of a single position shift each.

STEP 1. Read word 4, the location we wish to transfer into, out of RAM and into a temporary holding register. This register may be internal to the CPU or some unused location in RAM.

At this point we have the two words involved in the transfer operation being held where they may be read at any time but where they will not be altered by subsequent operations. Word 3, the word we wish to transfer, is held in ROM while word 4, the word occupying the place word 3 is expected to move into, is safely held in the temporary register. This is absolutely necessary since the CPU must keep track of where in the RAM memory the video image is held. When the processor attempts to move that image, it must first be able to locate its image data in RAM and thence onto the screen. However, the processor is capable only of locating data in 8-bit words, so video image data must always be transferred from one whole word location to another. Partial word transfers, such as three bits to the right, would place the image data in a location overlapping two words and the processor would not be able to address the beginning or end of the image. So what must be done is to hold the image data and word 4 in an unchanged state until we have shifted the image through the eight intermediate single position shifts. Once this has been done, the word will be written into the new location and the processor informed that the data pattern can now be found at the new location.

STEP 2. Read word 2 (the word immediately to the left of the word to be transferred) out of RAM and latch it into the first eight bits of the 15-bit latch.

STEP 3. Read the data word out of ROM and latch it into the first eight bits of the 15-bit latch. Note that at the same time this occurs the data already in the first eight bits of the latch (word 2) will be transferred back into the second seven bits of the latch.

STEP 4. Set the data lines to the following values:

$$D_0 = 1$$
$$D_1 = 0$$
$$D_2 = 0$$

and latch this 3-bit word into the shifter selector latch E6. This will set the shifters to output a new word where the LSB is bit 7 of the 15-bit latch and the MSB is bit 14 of the same latch. This new word will essentially be the data word shifted to the right one bit, its MSB being shifted out of the word and its LSB being moved up from word 2.

STEP 5. Take the output word from the shifter and place it in RAM for TV display at location 3.

STEP 6. Read word 4 out of its temporary register and latch it into the first eight bits of the 15-bit latch. Note that this will also take the data already in the first eight bits (the data word) and transfer them to the second set of seven bits in the latch.

STEP 7. Without changing the shifter select controls, again read the output word from the shifter and place it in RAM location 4 for TV display. The new word thus generated will also be built in such a way as to place the MSB in the least significant position of word 4. In effect we have moved word 4 one bit to the right, losing the MSB and taking one bit from the data word as its LSB.

This completes one pass in the transfer operation. At this point, we have the original data word still held in ROM, the original word 4 in a temporary register, while words 3 and 4 in RAM are the data word shifted one bit to the right. We may keep this arrangement for as long as we wish, depending on how we want the image to move. If we desire slow motion, we may keep everything the same for 8, 16 or any other number of TV frames. On the other hand, we can just as easily produce rapidly-appearing motion merely by shortening the period of time the data is kept unchanged. However, at some time we will have to continue the transfer operation by making another pass with the shifter (steps 8 to 19).

STEP 8. Read word 2 (the word immediately to the left of the original data word location) out of RAM and latch it into the first eight bits of the 15-bit latch.

STEP 9. Read the data word to be moved out of ROM and latch it into the first eight bits of the 15-bit latch. Note that this simultaneously transfers word 2 into the second seven bits of the latch.

STEP 10. Set the shifter address select lines to the following values by latching the appropriate 3-bit word into E6:

$$S_0 = 0$$
$$S_1 = 1$$
$$S_2 = 0$$

STEP 11. Write the word output from the shifters into RAM location 3 for video display. This new word must be constructed in the following manner. The LSB of the word will be the second MSB of word 2, while the MSB of the new word will be the second most significant bit of the data word. In effect the new word is the data word shifted two positions to the right, the two most significant bits being shifted out of the word altogether while the two least significant bits are moved up from word 2.

STEP 12. Read word 4, the new location for the shifted data, out of its temporary holding register and write it into the first eight bits of the 15-bit latch. This will transfer the data word to the second set of latch bits.

STEP 13. Without changing the shifter select control lines, read the

output word from the shifters into RAM location 4. Again, the new word will be the original word 4 shifted two bits to the right, the rightmost two bits being shifted out of the word and the leftmost two bits being moved up from the data word.

At this point, we still have the original data word stored in ROM while the original word 4 is similarly still being held in the temporary register. The data in RAM that is being displayed on the screen at word locations 3 and 4 is the original data word shifted two bits into word 4. Again, we may keep these RAM words unchanged for as long as desired, but in practice they are held in the 2-bit shift configuration for as long as they were held in the 1-bit shift positions. As long as data spends equal amounts of time in each of the eight shift configurations, the motion across the CRT will be smooth and continuous.

It is not necessary to repeat the entire latching and shifting procedure again to explain how the rest of the transfer is accomplished since each pass of the shifter will be the same as those described previously except that a new shifter address code will be latched to select right shifts of 3, 4, 5, 6 or 7 bits. After the 7-bit shift has been selected by an address code of 1 1 1, the program changes to the following operation.

STEP 14. Read word 2 out of RAM and latch it into the first eight bits of the 15-bit latch.

STEP 15. Set the latch E6 to select a shifter address of 0 0 0.

STEP 16. Read the word output from the shifters (the unaltered word 2) and write in into RAM location 3.

STEP 17. Read the data word we are moving out of RAM and latch it into the first eight bits of the 15-bit latch.

STEP 18. Word 4 in its original form may now be cleared out of its temporary register. We must now increment the horizontal accounting scheme in the CPU to indicate that the data pattern now resides in location 4 in RAM. The processor must log all such transfers so that it can find the data in RAM should it become desirable to effect another transfer.

The above is a simplified description of what the shifter and processor must do to move a single data word one location to the right. By expanding on the principles outlined above, more complex motion problems may be solved. Consider, for example, the case where the video image we wish to move one word location to the right is not a single data word, but a block of data words four words wide by sixteen lines tall. To move such a block merely requires repetition of the above steps for the required number of word locations and the required number of lines. In the example above, only three words were actually sent to the shifters to be involved in the transfer: (1) the word immediately to the left of the data word, (2) the data word itself and (3) the new location word. To move a longer block of words, on each pass we would send to the shifter the word immediately to the left of the first data word, the first data word,

GUNFIGHT

the second, third, fourth data words and finally, the new location of the rightmost data word. For each pass we need only set the shifter addresses once for a one, two, etc. bit shift to the right and we need store only the last word of the string in the holding register. If we performed such a pass across five or six word locations on one raster line, the other lines of the image could similarly be shifted by simply conducting an identical shifter pass on each line in the image. In this manner a block of video data of any size can be moved one (or more) word locations to the right in a series of eight passes over the block.

Leftward motion is accomplished similarly by eight consecutive passes over the data words. Suppose that our example above had been to move the single data word one location to the right or into word 2. On our first pass, we need to transfer the LSB of the data word into the most significant position of word 2. As can be seen from Figure 2.2.6-3, this condition will be satisfied by selecting a shifter address code of 1 1 1 for the first pass and essentially generating all the shift code in the opposite sequence as in the previous example. By running through the shifter codes in reverse, we can generate left motion in an almost identical manner to right motion.

2.2.8 Multiplexed Game Data. Although the program of the CPU is basically fixed and predetermined where the same processes are repeated over and over, there are a few areas which may be altered and motion is a good example of this. Although the actual transfer format of a data word to a new location is a fixed procedure, the actual data transferred and the shifter select codes used depend on a number of program variables. These variables include which image is to be moved, how long the image is to be moved and whether the image is to be moved up, down, left or right (or even an additive combination). These parameters are controlled not by the program, but by the player and are presented to the CPU in the form of an 8-bit word that the CPU can use to fill in the "blanks" in its program. The conversion from the players' control signals to an 8-bit word is the function of the opto-isolators and data multiplexers indicated in Figure 2.2.8-1. These multiplexers perform an additional function as well, namely to take the shifted data words and return them to the CPU at times specified in the program.

2.2.9 The Players' Control Inputs. Since we have already discussed the function and construction of the opto-isolators used in this game in Section 1.7.2, there is no need to go into great detail here other than to discuss the role of these devices in the system as a whole. Also, since all the inputs work in the same way, we need discuss only one — the coin switch input.

E1-2, the coin switch input, is normally at a 12V level but when the switch is closed to GND, the current flow causes the LED inside the chip to illuminate and the infrared light thus emitted turns on the output phototransistor whose collector is available at pin 7 of the chip. The output logic state of the isolator is therefore the same as the input, since a turned-on transistor essentially shorts the collector to ground. If the coin switch were suddenly closed, the output transistor in the chip would turn to the OFF state and the only voltage appearing at the output pin
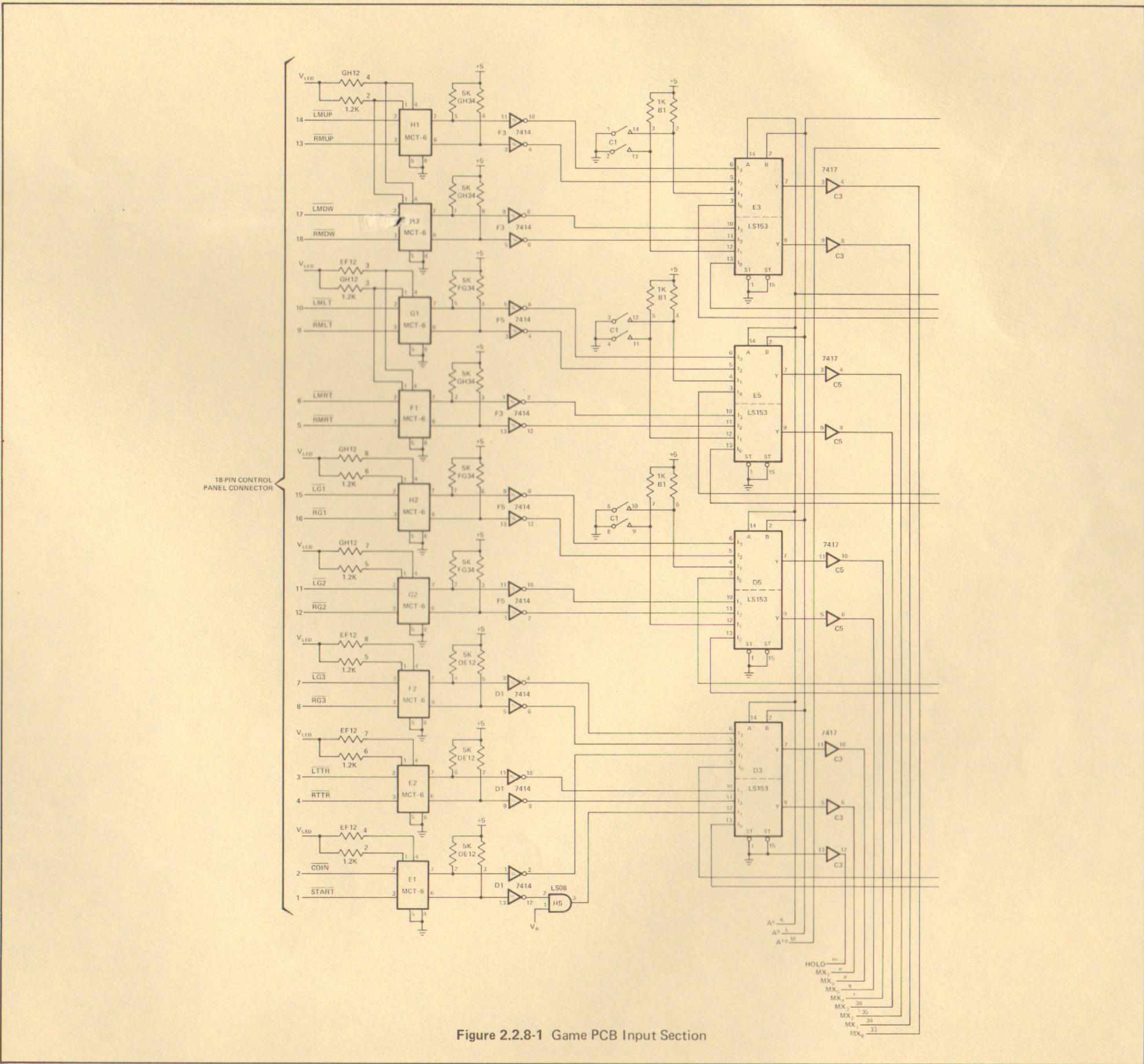

Figure 2.2.8-1 Game PCB Input Section

GUNFIGHT

would be the +5V supplied by the 3.9K pull-up resistor. All the opto-isolators and their inputs operate identically.

However, the opto-isolators are linear devices. They do not have sharp threshold voltages and may assume a voltage between 0 and 5 volts, depending on the input level. The output signal may rise or fall at a relatively slow rate and may even waver in the transistion. This is not critical or problematic when dealing with imprecise devices like the joysticks, but it does become highly undesirable when such signals are meant to become digital data input to a microprocessor. The CPU only has a few microseconds to look at the players' controls, so the voltage levels it sees during those short times must be clearly defined logic HIs or LOs. The purpose of the Schmitt-triggered inverters after each opto-isolator is to transform the slow linear transition of the waveform into fast-transition digital data. The Schmitt triggers have a very narrow threshold voltage range within which they distinguish between a HI and a LO. Consequently, a slow or unsteady rising signal at the inverter input will be output as a purely digital signal with sharp transitions occuring between stable TTL voltage levels. These buffered outputs are then sent to the data multiplexers prior to selection by the CPU.

2.2.10 Game Data Multiplexers. The game data multiplexers E3, D3, E5 and D5 select one of four inputs to be output as a single data bit. The four sets of inputs selected are determined by the address controls on pins 14 and 2 of the chips, which correspond to processor address lines A^8 and A^9 respectively. Since these addresses are under program control, the 8-bit data word dumped on the bus which eventually returns as input data to the CPU is selected by programming the four possible states of A^8 and A^9. Figure 2.2.10-1 shows the four possible combinations of A^8 and A^9 and the data bits placed on the bus for each combination.

The four words that may be selected are as follows:
1. Word 0 consists of eight bits of left cowboy data. This data includes the positions of the four motion switches, the trigger switch and the three bits representing the gun arm position. The player gun arm control moves a wiper across the three rows of pc traces and generates eight different binary codes depending on the setting of the control (see Figure 2.2.10-2).
2. Word 1 includes all the above information, except for the right cowboy.
3. Word 2 is the operator-definable word which consists of information about the setting of the coin switch, the start switch and the DIP switches which select the number of plays per credit, the number of coins per credit and the length of the game.
4. Word 3 is the video data word which is selected at the multiplexers for return to the CPU following a shift operation.

Each of these four 8-bit words are selected at the appropriate times to supply the CPU with the information it needs to complete its program. Since the bits of the game data word must run off the game PCB to the mother PCB, each bit is buffered as it exits the multiplexer output. Buffering keeps down noise on long traces and in edge connectors as well as supplying a larger than normal amount of current for driving the inputs they eventually feed on the mother PCB.

2.3 The Sound Circuitry

2.3.1 Sound Generation. Figure 2.3.1-1 contains all the circuitry required to generate the game's repertoire of four sounds which are the left and right shot and the left and right hit. Each separate sound requires its own amplifier and filter section and these are indicated by the grey areas
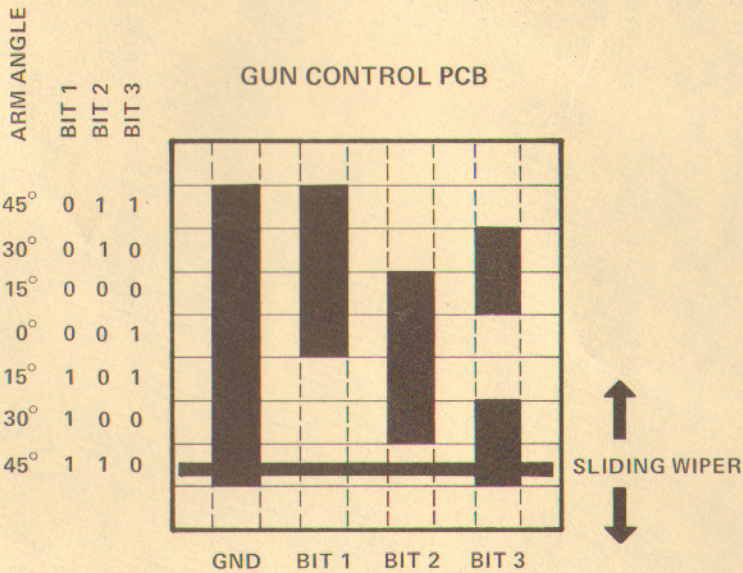

Figure 2.2.10-2 Gun Control Printed Circuit Board

ADDRESS SELECTS		MULTIPLEXED GAME DATA WORD							
A^8 A^9	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0 0	LEFT COWBOY UP SWITCH	LEFT COWBOY DOWN SWITCH	LEFT COWBOY LEFT SWITCH	LEFT COWBOY RIGHT SWITCH	LEFT COWBOY ARM WIPER, BIT 1	LEFT COWBOY ARM WIPER, BIT 2	LEFT COWBOY ARM WIPER, BIT 3	LEFT TRIGGER SWITCH
1 0	RIGHT COWBOY UP SWITCH	RIGHT COWBOY DOWN SWITCH	RIGHT COWBOY LEFT SWITCH	RIGHT COWBOY RIGHT SWITCH	RIGHT COWBOY ARM WIPER, BIT 1	RIGHT COWBOY ARM WIPER, BIT 2	RIGHT COWBOY ARM WIPER, BIT 3	RIGHT TRIGGER SWITCH
0 1	DIP SWITCH NUMBER 1	DIP SWITCH NUMBER 2	DIP SWITCH NUMBER 3	DIP SWITCH NUMBER 4	DIP SWITCH NUMBER 5	DIP SWITCH NUMBER 6	DIP SWITCH NUMBER 7	START SWITCH
1 1	SHIFTER DATA WORD, BIT 0	SHIFTED DATA WORD, BIT 1	SHIFTED DATA WORD, BIT 2	SHIFTED DATA WORD, BIT 3	SHIFTED DATA WORD, BIT 4	SHIFTED DATA WORD, BIT 5	SHIFTED DATA WORD, BIT 6	SHIFTED DATA WORD, BIT 7

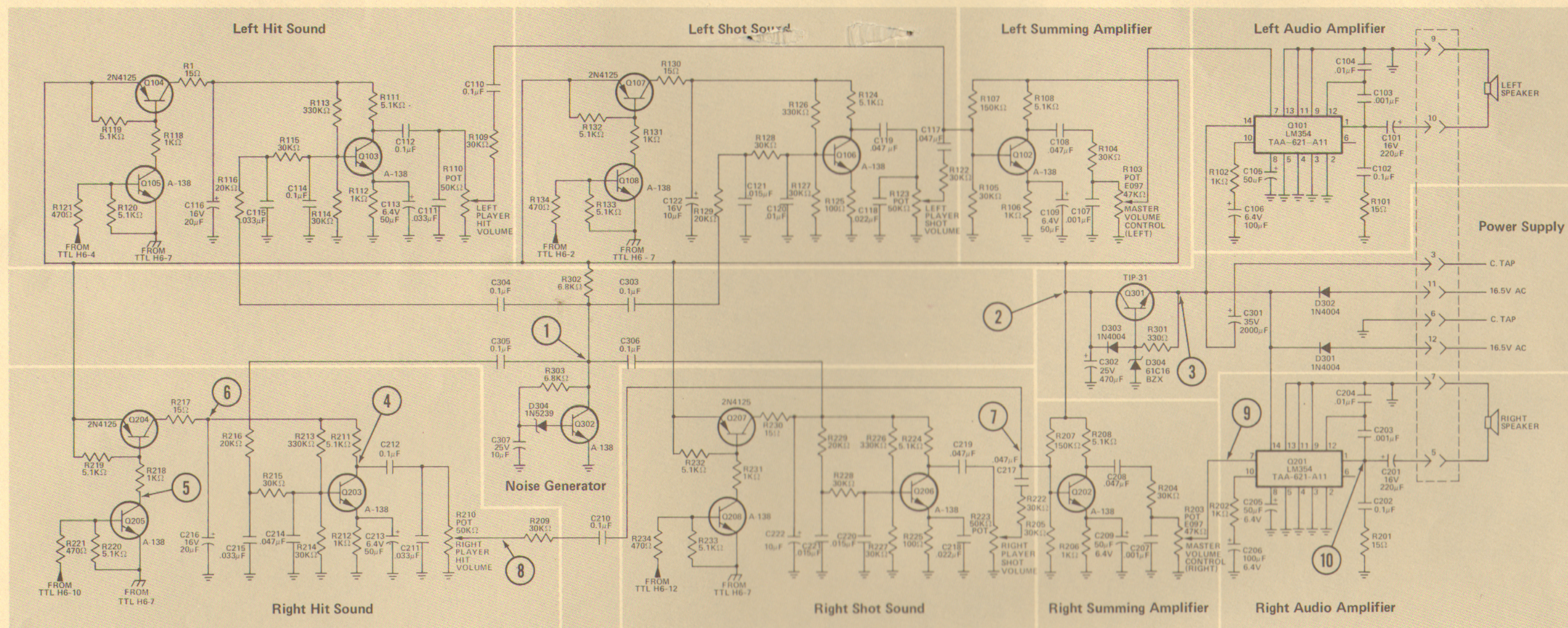Figure 2.2.10-1 Multiplexed Game Data

Figure 2.3.1-1 Sound Circuit Schematic

of the schematic. Each of these sections gates, amplifies and filters a shared raw noise waveform in a characteristic way that produces a sound identifiable with each of the players. Since all four sections are more or less the same, we need discuss only one (RIGHT HIT) and note any areas which differ.

The sound circuit analysis is divided into two general sections. First, the power control circuit and white noise generator are discussed together for these two subsections are used in common for the generation of all four sounds. The second part of the sound circuit analysis deals with the generation of the single example sound, RIGHT HIT.

2.3.2 Power Control and Raw Noise Generation. Unprocessed AC power arrives at the sound circuit from the transformer in the form of a center tap line and two AC lines which are 180° out of phase with respect to each other. The center tap is at a zero volt potential with respect to the AC lines and it is used as circuit ground for the entire sound system. It should be noted that the sound circuit analog ground is completely separate from the digital ground used for the rest of the circuitry.

The raw AC waveform is full-wave rectified by two power diodes D301 and D302 and the resulting pulsating DC waveform is filtered by the large 2,000μF capacitor C301. This unregulated pulsating DC waveform is allowed to trickle through zener diode D304 which places a steady reference voltage of 17V at the base of Q301 due to its reverse bias characteristics. The voltage applied to the base of this transistor appears at the emitter with a drop of .6V in potential so a well regulated DC level of approximately 16V is maintained on the V$_{ss}$ bus at all times. The current required to maintain this level is drawn from the 28V unregulated DC line. The regulated 16V DC line is the main power source for all the audio amplifier stages.

Q302 is the white noise generator for the entire system. Zener diode D304 is biased in such a way that it operates in the "breakdown" region
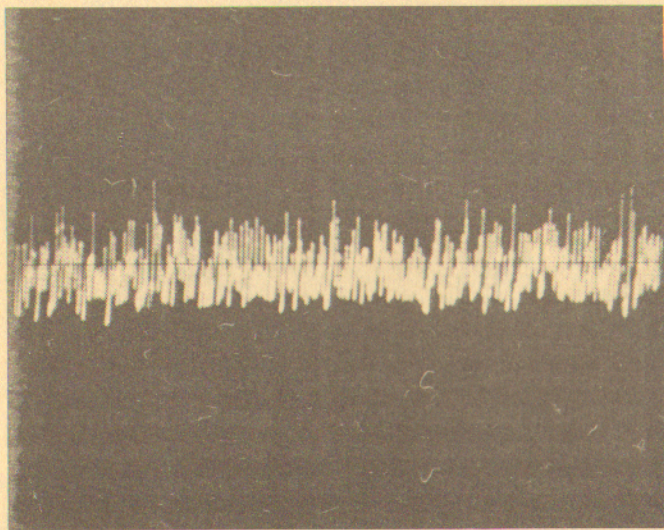

Figure 2.3.2-1 Raw Noise Waveform (TP-4)

near the "knee" — a region inherently noisy in zeners. The random noise produced by the current through the diode is applied to the base of transistor Q302 and amplified in a common emitter arrangement. This amplified noise appears as an output signal on the collector of the transistor where it is capacitively coupled into the four audio circuits through the four large coupling caps (.1μF).

2.3.3 Generation of the RIGHT HIT Sound. Q204 and Q205 act as a two-stage digital switch to provide high current capability for quickly charging capacitor C216. The emitter of Q205 is wired directly to digital GND at H6 on the game PCB while the base of this transistor is resistively coupled to the output of the driving inverter at H6-10. When the CPU activates the RIGHT HIT sound, it does so by driving H6-10 LO which completes a conduction path through the base-emitter junction of Q205, thereby turning that transistor to the ON state.

The collector current for this transistor is derived from the base of Q204 which is also switched ON by the conductive state of Q205. Q204 is capable of high current gain and is used to dump a great deal of current relatively quickly into capacitor C216. The HI signal has completed its job of charging the cap and is no longer needed so a LO bit is latched into the output data latch on the next latching cycle turning Q204 and Q205 to the OFF states. The supply voltage cap C216 will now be allowed to discharge through the only path open to it — the collector current flow through Q203.
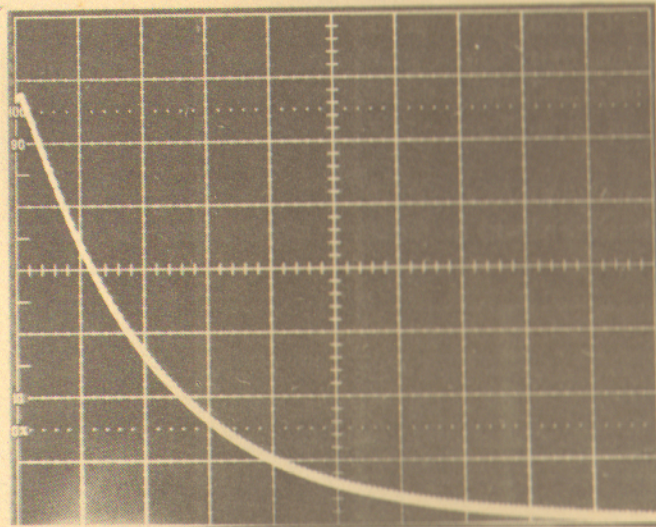

Figure 2.3.2-2 Charging Of Shaping Capacitor (TP-6)

C216 acts as the main power source for the common emitter amplifier stage built around Q203 where R212 and C213 are the gain and amplifier elements. The gain of this stage is proportional to the power supplied to it from C216, which charges at a high rate but discharges much more slowly through the collector resistor R211 and the stage transistor itself. As a result, stage gain at Q203 rises very sharply causing a sharp attack in the sound waveform when it is switched on, however it falls off slowly causing a slow decay of the waveform and, hence, the actual

sound itself. The pattern is characteristic of explosion sounds which rise very sharply, then die away in a slow rumble.

The actual signal amplified by this variable gain stage is the random noise from the noise generator circuit discussed previously. This waveform is coupled to the base of Q203 through the two-stage passive, low pass filter constructed from R216, C215, R215 and C214. These filter elements screen out some of the higher frequency "pops" and "crackles" but pass the lower frequency "rumbles" and "thumps." The filtering components have been chosen to shape a characteristic sound in each of the four sound circuits, hence the actual values of these components differ from one circuit to the next giving rise to four distinct and identifiably individual sounds.
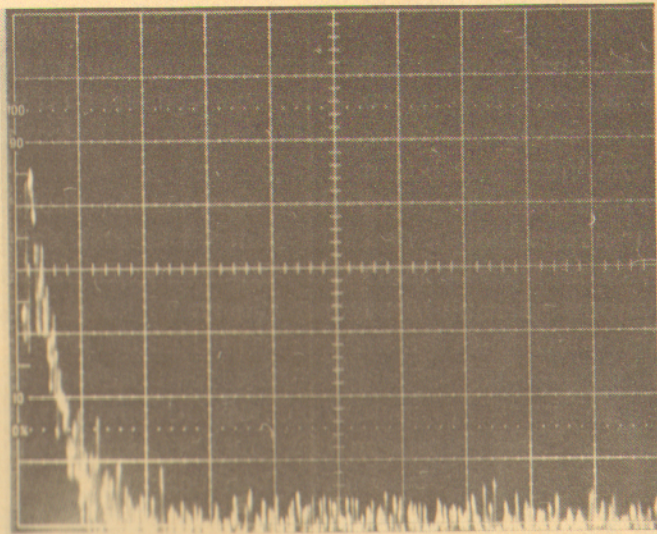

Figure 2.3.2-3 Resulting Noise Waveform (TP-7)

After it is filtered for tone and amplified for attack and decay characteristics, the audio waveform is capacitively coupled out of the stage through C212 and placed at the 50K trimpot R210 which controls the volume of the sound and performs a bit of final filtering. It is then capacitively coupled by C210 into the base of Q202 where it is added to the shot sound circuit output and again amplified. Note that the valves of the filtering and coupling elements in the output circuit of Q203 are different from the corresponding elements in the other sound blocks. Again, these differences are due to the desired tonal characteristics needed to output four distinctly individual sounds, hence the component values vary for each different sound. The principle involved and the use of these components, however, remains the same in each sound block.

Q202 is a summing stage for the two right sounds (HIT and SHOT) and also provides another level of amplification to bring the signal up to a level high enough for input to the integrated power amplifier. The signal is placed at the base of the Q202 stage and capacitively coupled out of the collector through C208. R203 and C207 provide the master volume control for the right player as well as acting as a filter stage. At this point, the sound is shaped and amplified enough to require only a power gain

GUNFIGHT

of sufficient magnitude to drive an 8Ω load (the speaker) and this final bit of amplification is the only function of integrated amplifier Q201.

Q201 is an LM354 integrated amplifier capable of about 5W. It accepts an input audio signal at pin 7 and outputs it from pin 1. Pins 10 and 12 are frequency compensation pins and the passive filter devices wired to these pins roll off for a flat response. Pin 8 decouples the supply ripple. The high power output is capacitively coupled to the speaker through C201 and this passive filtering on the output pin provides a slight feedback and acts as a very low frequency filter as well as eliminating any 60 cycle hum from the LM354's unregulated power supply.

Although only this one section of audio circuitry has been analyzed, be aware that the other three work in an identical fashion except for the values of the passive filtering components which are purposely different to produce distinctly individual tones. The entire circuit is designed as a stereo system where the right player's sounds occur in right channel and the left player's sounds occur in the left channel. Total sound output is about 2W, RMS.

SECTION III TROUBLESHOOTING PROCEDURES

3.1 Introduction. In this section each of the four circuit blocks found on the game PCB will be discussed in terms of actual signal descriptions and timing. Wherever possible, the exact values of the timing configurations will be given, however — due to the nature of the circuitry — this will not always be possible. The processor address lines and both sets of data busses are under program control and — unless one has an actual program to read and follow — the signals will appear more or less random. If possible, the activity appearing at a particular test point will be described in terms of a known game condition (i.e. the credit mode) compared with the associated monitor display. In some cases, patterns of signals that would ordinarily appear random may be made to take on a predictable structure once the number of program parameters and interrupts have been reduced to a minium. In other cases, blocks of circuitry may be tested by forcing inputs to known states and watching for predictable results in another part of the system. Wherever such techniques are usable, they will be given.

3.2 Control Signal Procedures. Each of the following sections describes a particular troubleshooting procedure for a specific test point. The locations of these test points are indicated on the overall game PCB schematic found on page 26. While many of these areas may be tested with a logic probe, please be aware that comprehensive testing of the game will require the use of an oscilloscope.

3.2.1 Test Point 1. TP 1 is the control signal responsible for latching output data into latches G5 and F6. This signal is normally a steady HI which only pulses LO when output data is expected to be on the data bus. This condition will occur whenever the trigger switch is closed on either the left or right gun and whenever the coin switch is operated. A logic probe at G5-9 will show a steady HI level changing to a rapid pulse indication whenever any of the above three switches are closed. The pulsing indication will last only for a very brief period of time and will probably be indistinguishable from a single pulse indication.

On the scope, the steady HI level will change to one of two states depending on whether the trigger switch or the coin switch initiated the activity. If the trigger switch activated the load signal, four LO level pulses will be visible. Each of these pulses will be about 500ns wide and the four pulses will occur in two pairs where 60ms exists between members of a pair and 215ms between the pairs themselves. The first pair of pulses sets a shot sound into the latch, leaves it there for about 60ms so the timing capacitor in the sound circuit may charge, then removes it by latching a zero into the latch with the second pulse of the pair. The second pair of pulses (occurring 215ms after the completion of the first pair) is a scan for the hit sound. If a hit has been scored on the playfield, the data line corresponding to the correct hit sound will be HI and the first pulse of the second pair will set the hit sound latch. As before, the second pulse in the pair will clear the latch after about 60ms charging time on the timing cap and this activity is illustrated in Figure 3.2.1-1.
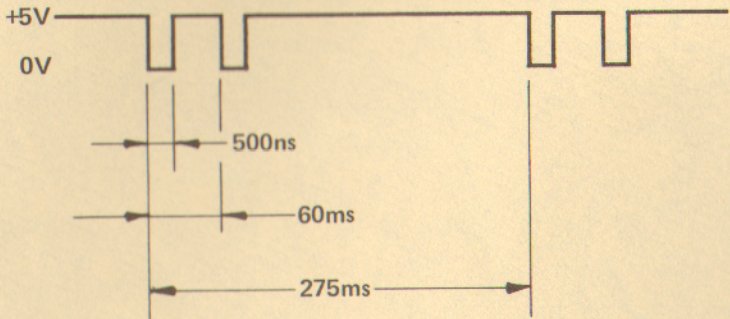

Figure 3.2.1-1 Test Point 1

If the output latch is being clocked by the activation of the coin counter, the process will be somewhat different. There will be only one pair of LO pulses where each pulse has a duration of about 500ns and where the two pulses occur about 120ms apart. The initial pulse sets the coin counter latch and activates the coil of the counter. After about 120ms, the second pulse will latch a zero into the counter latch to deactivate the coil.

3.2.2 Test Point 2. This test point describes the operation of the shifter latch clock and note that this signal is motion dependent. When there is no motion occurring on the CRT, the line will remain at a steady HI logic level. A logic probe connected to H5-11 will reveal a HI with a steady pulse indication whenever any motion occurs on the screen. On the scope, the line becomes active as per Figure 3.2.2-1. LO pulses of about 500ns duration occur in groups of four. These quad groups occur in 4.5ms bursts and the bursts have a frequency of about 34ms.
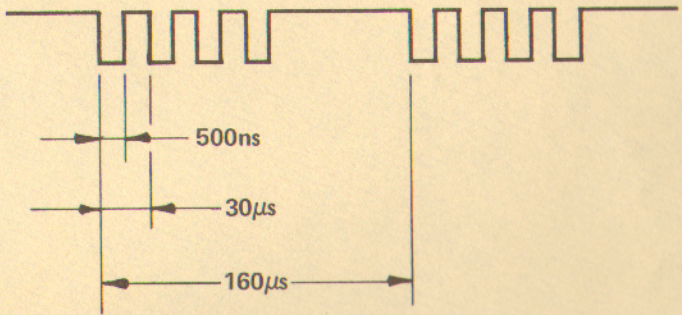

Figure 3.2.2-1 Test Point 2

If this signal is not functioning properly, there will be distinct problems with the way the images move on the CRT. Fixed or motionless images will be unaffected, however. With no clock signal, data words comprising the moving images will not be latched into the 15-bit latch and consequently cannot appear on the screen. The images will either be completely absent or a few words of the image will be reproduced where the entire image should be, giving a chaotic or broken-up effect to the object being displayed.

3.2.3 Test Point 3. This test point describes the operation of the shifter address load which clocks data off the bus for decoding into the eight shifter codes. A logic probe connected to H5-8 will display a steady HI

GUNFIGHT

whenever there is no motion occuring on the screen. But, when motion of any sort is occuring, the pulsing indicator in the probe should illuminate. Oscilloscope analysis reveals that the activity of the load line during screen motion is a set of LO pulses of about 100ns duration and with a frequency of about 30ms.

With the load signal disabled, new shifter codes cannot be latched into the latch at E6 and smooth horizontal motion of the images cannot occur. Images will be forced to move in 8-bit jumps which is evidenced by very jerky CRT motion. Motion may also occur in the reverse where leftward motion being all that can occur or it is even conceivable that any sort of motion could be prevented. The images themselves will be present and complete, however. Failure to change the shifter codes should not cause distorted or missing images, just distorted or missing motion.

3.2.4 Test Points 4, 5 & 6. Test points 4, 5 and 6 are found at D3-14, D3-2 and H5-12 respectively. These three signals are the three processor address lines which are used extensively on the game PCB. During normal game operation, the information found in these three lines appears almost random since the precise activity at any given time is dependent on the actual program steps being run. However, if the game is placed in the credit mode, a basic pattern can be seen and checked. The credit mode is probably the most basic of all the game's modes because there is absolutely no activity on the CRT. All motion ceases, there are no flashing words and the timer does not count. The processor may be thought of as "idling" in this mode. Figures 3.2.4-1, 2 & 3 illustrate address bits A^8, A^9 and A^{10} respectively when the game is in the credit mode. As can be seen from these photographs, there is a distinct pattern to each signal. Duration and frequency of activity cycles are basic to each address bit. These patterns are fundamental to the processing activity and — even though the game mode changes or is made more complex by motion — these same basic patterns will still exist on the signal line.
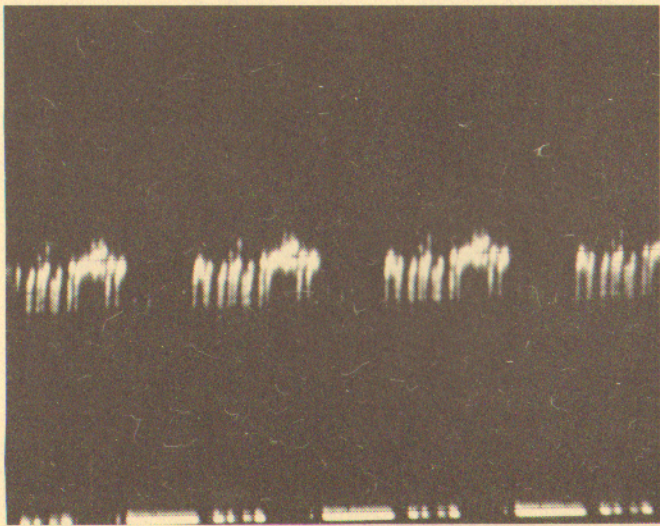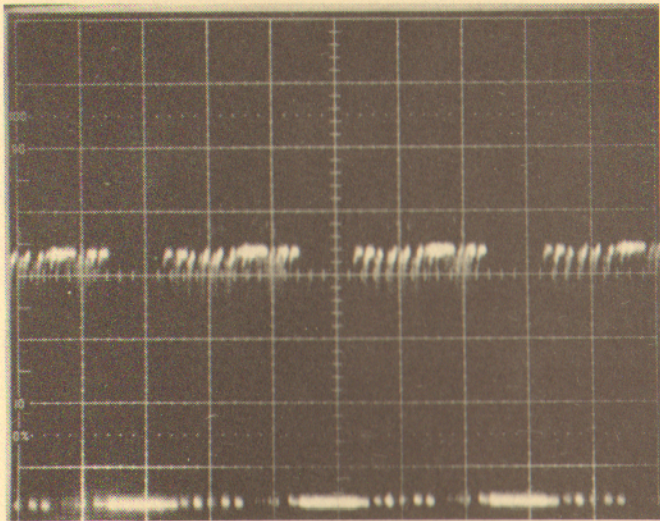

Figure 3.2.4-1 Address Line A^8 (TP-4)

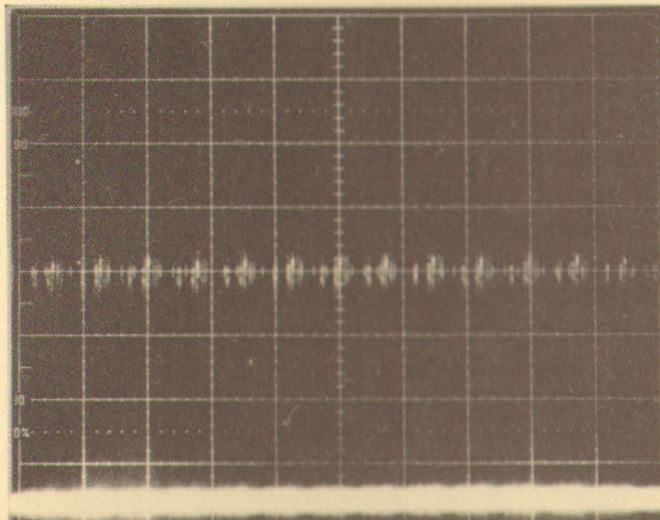Figure 3.2.4-2 Address Line A^9 (TP-5)

Figure 3.2.4-3 Address Line A^{10} (TP-6)

Whether they may be isolated and viewed with the same clarity of the three pictures presented here will depend on the degree to which processing activity deviates from the credit mode. Background pulses due to motion and cyclic changes from the timer will complicate the scope display until eventually it is unrecognizable and contains what appears to be random information.

A logic probe connected to any of these three test points will display HI, LO and PULSING indications at all times as long as the game is powered. No other distinguishable logic probe activity can be firmly stated.

3.2.5 Test Point 7. This signal (SAMPLE) is also a motion related signal and will remain at a steady LO level as long as no motion occurs on the screen. Motion of any sort will cause a logic probe connected to H5-13 to indicate a pulsing signal for as long as the motion continues.

These LO pulses on the SAMPLE line have a duration of 500ns and are grouped in bursts of 4 or 5. Spacing between the members of a group is about $30\mu s$, although an entire group of pulses usually lasts about $150\mu s$. Pulse bursts occur with a frequency of about 30ms for as long as motion is present on the screen and Figure 3.2.5-1 illustrates these timing relationships.
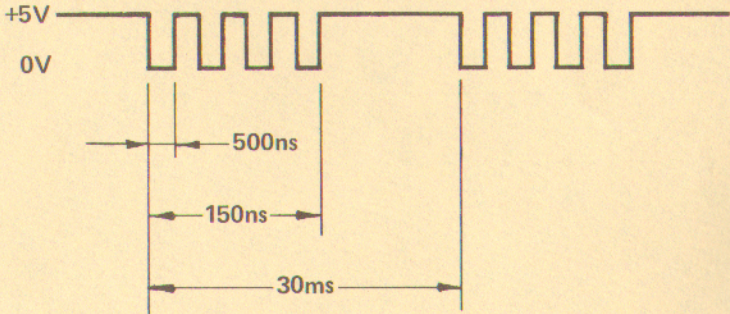

Figure 3.2.5-1 Test Point 7

3.2.6 Test Points 8, 9 and 10. These test points correspond to shifter selects S_0, S_1 and S_2 and are found at locations E6-2, E6-10 and E6-15 respectively. The activity of these signals is rather complex and depends on a number of parameters, including the game mode. These are basically motion oriented signals and are essentially square waves. We will discuss the activity of these lines for each possible game configuration.

During the attract mode, there are three sets of different activity patterns: one for no motion on the screen, one for cowboy motion and finally one for the motion of the bullet. During the period of no motion, all three lines will contain steady logic levels. Whether these levels are HI or LO is indeterminable since they are basically left in the state they were in when motion last occured. As the cowboy begins to move onto the screen, all three lines become active producing an indication of HI, LO and PULSING in logic probe. As mentioned before, these signals are all square waves where S_0 has a frequency of 70ms, S_1 = 140ms and S_2 = 280ms. This type of activity will continue until the bullet is fired. Bullet motion during the attract mode is under shifter control and so the shifter selects will become active at this time. The frequencies for bullet motion during attract are: S_0 = 8ms, S_1 = 16ms and S_2 = 32ms. The faster shifter address changes result in the bullet motion being faster

than the cowboy motion. Also, note that the bullet in the attract mode moves much more slowly than the bullet fired in the game mode. This results because during the attract mode — the bullet is moved through the shifter and undergoes address relocation of one bit at a time whereas in the game mode the shifter is not employed for the bullet motion and it moves in 8-bit jumps.
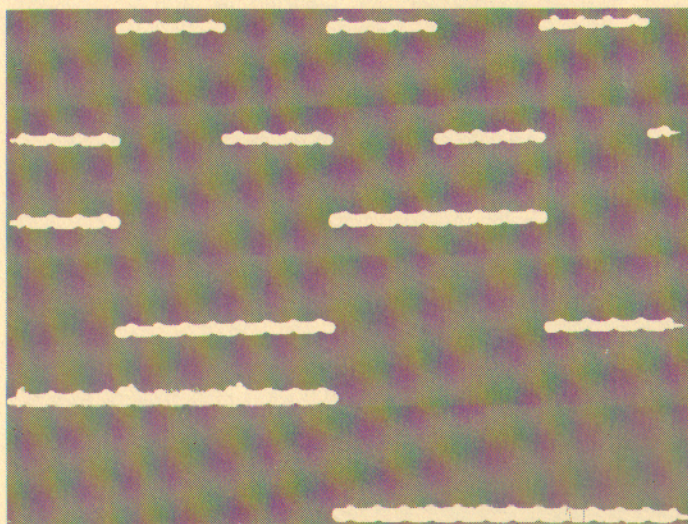

Figure 3.2.6-1
Comparison Of S_0 (Top), S_1 (Center) & S_2 (Bottom) With Left Motion
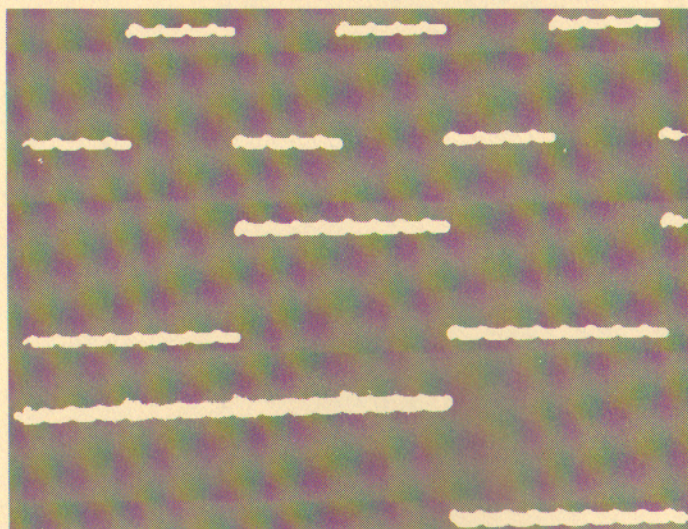

Figure 3.2.6-2
Comparison Of S_0 (Top), S_1 (Center) & S_2 (Bottom) With Right Motion

In the game mode, the three address selects are the same for no screen motion and horizontal cowboy motion. Indications and measurements are the same as in the attract mode. The shifter, it must be remembered, is a horizontal motion device and does not take part in vertical motion. This can be noticed in that the wagon motion and vertical cowboy motion will not cause the shifter address lines to become active. In the game mode, the bullet will not cause address activity either, with a slight exception which we will discuss in a moment. The higher speed of the

bullet in the game mode is the result of the image being jumped across the screen in 8-bit jumps that do not require shifter involvement. Cowboy arm motion also does not employ the shifter and will not cause these test points to become active, except as noted below. In general, the shifter address lines will remain in the state they were set to last before new horizontal cowboy motion occurs, at which point all three lines will become active in a square wave format as described for attract mode cowboy motion.

The exceptions noted above are as follows. Each of the three lines will be set unconditionally HI whenever the left arm position is changed and will be set unconditionally LO whenever the right arm position is changed. These changes will occur only once — if at all — as required to put the signal lines in the proper unconditional states. Similarly, all these test points will change as needed to become unconditionally LO whenever the left trigger switch is closed and will be set unconditionally HI whenever the right trigger is operated. These exceptions obviously only occur in the game mode. Simultaneous motion that would otherwise set the bits to opposite (conflicting) states is handled in a multiplexed manner, the signal lines being left in the state determined by the last arm or trigger switch to be operated.

3.3   Data Bus Procedures. As we have mentioned several times before, there are two sets of data busses used to carry information between the mother and game PCBs. The multiplex data bus carries information from the game PCB to the mother PCB while the output data bus carries information in the reverse direction. Both data busses are tri-state meaning that their signal levels may be HI, LO or "floating." The activity of both sets of busses is highly complex at all times during the course of the game. An oscilloscope applied to any data line will show only general chaos and a logic probe will merely reveal HI, LO and PULSING for all game modes and states. While this may be helpful information in a few cases (i.e. if a line stays LO constantly), the logic probe will, in general, not yield conclusive information. Therefore, we recommend the following procedures in attempting to troubleshoot the game PCB data busses. The following two sections discuss the procedures for troubleshooting both busses separately.

3.3.1   Output Data. If the problem occuring in the system is (1) a constant sound, (2) a constant lack of sound, (3) a constant energization of the coin counter coil or (4) lack of coil energization, it would be advisable to thoroughly investigate the output data latches, inverters and drivers. If the game is functional except for a single data signal (eigher a sound or the coin counter), the following procedure is recommended.

We will use the RIGHT SHOT sound as an example since all the sound functions operate identically. Test Point 11 (H6-13) is normally LO, however it will pulse HI for 60ms whenever a valid shot is fired by the right player's control. The shot is considered valid when the game is in the play mode and when less than six shots have been fired during the current shoot-out. If the signal at TP 11 conforms to the above description, the problem is in the sound section of the game PCB and not in the output data bus.

If the signal at TP 11 does not correspond to the above description, a thorough analysis of the output latching circuit should be undertaken in the following manner. By tying C3-12 to V_{CC}, the processor may be placed in the hold state so that all the data lines are forced to the floating state where they are essentially disconnected. While the data lines are in this state, F6-9 may be pulsed with a logic pulser to simulate a load pulse to the output data latches. With the latch inputs floating, the latch output pins will all be set LO following the load pulse. With a clip lead, each of the latch inputs at F6 may be individually grounded to a logic GND point and a new load pulse injected. Each time a new input line is grounded, during a load pulse only the corresponding latch output should rise to a HI level and all the others should remain LO. If this procedure is completed without finding any error, the decoder may be tested in the following manner.

By latching all floating inputs into the data latch, all zeroes will appear at the latch outputs and decoder pin 1 must be the only decoder output pin to drop to a LO logic level. Similarly, if a binary 1 is forced on the latch outputs by grounding F6-13 and loading the latch, pin 2 of the decoder must be the only output pin to drop LO while all the others remain HI. Each binary number between 0 and 9 may be forced at the latch outputs by grounding the proper input pin during a latch load and, for each binary combination, one and only one decoder output pin should fall to the LO level (for binary numbers greater than 9, all decoder outputs will remain HI). If the decoder passes the test, the inverters should be tested by monitoring their outputs while forcing their corresponding inputs first to a LO level and then to a HI level. The resulting state of the output should be at the opposite level as the input. This last test will complete the test procedure for the output data circuitry and, if the game still does not function properly, the malfunction must be elsewhere, either in the player's controls, the game data flow or in the processing of this information by the CPU.

The coin counter may be tested in a similar manner. Test Point 12 is normally HI and pulses LO for 120ms each time the coin switch is operated in any game mode except for the hold state. If this does not occur, test the latch at G5 in the same manner as outlined above. A working latch indicates that the problematic area lies outside the output data circuitry and in the game data flow, the coin switch wiring or the processing of the information. If this signal does occur as described yet the counter will still not operate, test the driver at C5 in the same manner as outlined above for testing the inverters. Note, however, that C5 does not invert the signal and that the output of the device simply reflects the input. If the driver does in fact work properly, use the oscilloscope to monitor the collector of Q403 while the input of C5 (pin 13) is forced to both the HI and LO states. While the input is held LO, the collector of Q403 must show about +5 volts and while forcing C5-13 HI, the transistor of the collector must fall to ground. If all of the above behavior is observed yet the counter will still not function, either the counter wiring or the counter mechanism itself is malfunctioning.

3.3.2   Game Data. Troubleshooting the flow of game data over the multiplex data bus involves testing the opto-isolators, the Schmitt triggers and

24

the multiplexers. In the case of the opto-isolators and Schmitts, the game may be in any mode or configuration. Each input pin to the isolators must show about 12 volts when the corresponding player control switch is not being activated. Operating a switch (the right trigger switch for example) will cause the corresponding isolator input to drop to ground (E3-2 in this case). The isolator output pin will respond by outputting the same level as applied to the input, but at TTL logic levels. An open switch will force a TTL HI from the output while a closed switch will force the output LO. Opposite states must be detected at the outputs of the appropriate Schmitt triggers, as these devices also invert the signal level.

Troubleshooting the multiplexers is a more difficult task and it may be less time consuming in some cases to simply replace a suspected device. However, these devices may be tested in the following manner if desired. Processor bits A^8 and A^9 may be isolated from the CPU and placed in a floating condition by removing the game PCB from the mother PCB edge connector and taping edge connector pins K and L. Then re-insert the game PCB. A multiplexer address of 00 can be selected by grounding both select lines simultaneously. The signal states at output pins 7 and 9 must match the input signal levels at pins 6 and 10 respectively. For example, the input pins may be forced HI while the output pins are monitored to see that they also output HI logic levels. After all the multiplexers have been tested with an address select of 00, the grounding clip lead on pin 14 of the chip may be removed. The input signal will be regarded as a HI and the multiplexer address will increment to 01. Again, output pins 7 and 9 may be observed for a matching state with respect to input pins 5 and 11 respectively. In a similar manner, all four binary address select combinations may be constructed at the multiplexer select inputs and the outputs checked to see that they are being properly selected.

3.4   Shifter Procedures. In general, game PCB problems are those associated with sound, coin counting and the player's controls. This last category includes proper game timing, credit score value, cowboy arm position, triggering of the bullet and cowboy motion. Problems which involve objects which are supposed to move but do not may very likely be centered in the shifter circuitry.

An extremely detailed discussion of how to test the shifter circuit components will not be presented here since most of this information has already been discussed previously. The procedure for testing latches has been thoroughly discussed and the shifters themselves are tested in very much the same way as the multiplexers except that the data is shifted instead of merely being reflected. However, there is one area which does merit an in-depth discussion and this is deciding whether it is the shifter or the CPU circuitry which is the problematic area.

The shifter is active only for images which undergo horizontal motion. This includes the cowboys and the bullet during the attract mode. All other images do not go through shifter data modification so problems with motionless or vertically moving images do not involve this area of circuitry. In addition, the game board generally will not cause gross distortions in the images. If the entire picture rolls (either vertically or horizontally) or is broken up diagonally, the TV monitor may need adjustment or sync from the CPU may be malfunctioning.

Problems that arise in the shifter circuitry generally fall into two categories: data and control. Data bits to the shifter make up the actual image on the screen and if any of them are missing or incorrect the effect will be very obvious. Cowboy images with black or bright white vertical stripes, broken up arrays of dots where a coherent image should be or even solid blocks of white (or black) may very well turn out to be shifter problems involving the flow of shifted data. These malfunctions will appear with the above symptoms while everything else on the screen appears perfectly normal and, in many cases, even the motion of the distorted image will be normal.

Control problems in the shifter will most likely result in jerky or incorrect motion of an otherwise correct and coherent image. Shifter codes which do not change will result in motion of eight bit jumps which will be very noticeable on the CRT. Single held bits in the shifter codes will cause slightly jerky motion or even a wavering of the image and this effect may be evidenced by a "three steps forward, two steps back" sort of motion. In some cases, motion may be completely absent and we have even seen instances where the image left replicas of itself side by side in its wake as it moved across the screen. These replicas may appear as whole images, parts of the image or even as a solid white bar across the CRT.

3.5   Sound System Procedures. Although an analysis of the sound board has already been presented in a fairly complete manner, we would like to take a minute to outline some general troubleshooting techniques for the audio section of the game PCB.

Whenever the problem has been narrowed down to the sound section, the first place to check is the collector of the digital switch (i.e. Q205 for the right hit sound section). Collector voltage must be about 16 volts during normal operation but should drop very near to ground whenever the right cowboy is hit by a bullet. Keep in mind that the collector will drop low only for a very short period of time (about 60ms) before it returns to the 16 volt level. If this does not occur, do not go any further in the audio section until the digital game output data circuitry is known to be functioning properly and the sound section power supply levels have been thoroughly checked as well. In this case, the collector of the power supply transistor Q301 must appear as in Figure 3.5-1, while the emitter of the regulator should appear as a flat 16 volts. If the power supply voltages are as they should be and the digital section of the game PCB is also functioning properly yet the proper behavior is still not seen at the collector of Q205 (TP5), replace the faulty components in the Q205 circuit. If TP5 checks out O.K., continue as follows.

First check TP6 for proper charging and discharging of capacitor C216. The proper charging pattern is illustrated in Figure 2.3.2-2. This exponen-

tial fall-off will occur immediately following a digital activation. The falling voltage is used to shape a pulse of noise generated at TP4. The collector of Q302 must show a great deal of random white noise (Figure 2.3.2-1). If white noise is present and pulse shaping correct, then TP7 is the next place to look to insure the summing of these two signals has been made. Figure 2.3.2-3 illustrates the waveform which should appear at this test point. This is the shaped noise pulse and the characteristics of this waveform should not change from this point on in the audio circuit. In other words, the same waveform should also be found at test points 8 and 9 except at a higher level of amplification.

At test points 8 and 9 there are two additional points to note and these are the volume controls. Changing the volume control setting for the right hit sound will cause the signal at TP8 to be boosted or attenuated and it will appear larger or smaller with respect to the center setting of the trimpot. This is also true of TP9, where changing the right side master volume control will cause the noise pulse to be modified in its overall gain value and hence its peak voltage will also change. If the signal at TP9 appears as in Figure 2.3.2-3 and both volume controls (hit sound and master volume) appear to be functioning, then the problem is most likely in the integrated audio amplifier or the speaker wiring.

The best way to troubleshoot the LM354 audio amplifier is simply to replace it. There is no easy way to actually test the circuit although the capacitors used with it may be checked with an ohmmeter for shorts. The speaker itself may also be tested with an ohmmeter to insure that its windings are not open. If the caps and speaker are not faulty and the input signal at pin 7 of the LM354 is valid, the chip must be replaced.

GUNFIGHT

Figure 1.1.1-1 Gun Fight Digital Schematic