Chapter 7
A Collection of War Games

It would be possible to devote at least several more chapters to all sorts of special game controls, first presenting some basic control circuit and then demonstrating a specific application. But this is a good time to change the form of presentation, describing a game first, and then showing some of the special controls it uses.

Although you can certainly build any of the games in this chapter without referring to anything else in this book but Chapter 2, it would be wise to complete your study of the entire book first.

For instance, you will notice that the figures used in the war games described here are more rectangles, and not interesting, complex figures. You have the option of using the material in Chapter 4 to generate such figures from the simpler ones used here.

For the sake of overall simplicity, the games in this chapter do not include any audio and scoring circuits. These, too, can be added with great effect later on, but of course, you must complete a study of the material relating to audio effects and scoring first.

And finally, you ought to be aware of figure-rotation effects. None of the figures in this chapter can be rotated on the screen; so if you want to add the rotation features, you must study the more advanced chapter dealing with that particular subject.

Don't be mislead into thinking the games in this chapter are overly elementary, however. While these games are, indeed limited with respect to the finer niceties of video-game technology, their control schemes are as complex and meaningful as any you will find anywhere.

This chapter strikes at the very heart of TV games, demonstrating the essential control features without confusing the issue with a lot of added features (intended by others to make you buy the product or to keep on shoving quarters in the slot). You can add the razzle-dazzle later on it you want.

MISSILE ATTACK II

Here is an extended version of the Missile Attack game already described in Chapter 6. This version requires two players, one for controlling the attack missile and another for controlling the antiballistic missile. Player B's task is to program the path of the attack missile, then launch it toward a target. The attack missile is always launched from the left-hand side of the screen and at an initial altitude programmed by Player B. The target is located at the lower right-hand corner of the screen, and Player B can strike that target using a wide variety of programmable velocities, initial altitudes, and rates of descent.

Player A's task is to protect the target by launching antiballistic missiles. The antiballistic missile can be launched only from the bottom of the screen, but Player A can program the horizontal position of his missile just prior to launching it.

Neither player has any control over their respective missiles once they are launched.

This game features a pair of program panels that add special interest to the whole affair. The players get the impression they are working with a computerized system in a fashion that is not found on any commercial TV games. This is not an easy game to play. It is a challenging game bound to spark the interest of players who like to think clearly, act quickly, push buttons, and throw switches.

The Basic Game Plan

The flow chart in Fig. 7-1 outlines the automatic control scheme for Missile Attack II. Under normal conditions, Player B launches the attack missile first, presumably setting it on its path toward the target. Player A then responds by launching the antiballistic missile.

Assuming both missiles are launched and set, the system enters a 4-point-decision-making mode. Suppose the antiballistic missile strikes the attack missile before anything else happens. (This would be a score for Player A.) Whenever A = B, the image of the attack missile flashes on the screen for about one-half second before it is reset to its initial position again. The antiballistic missile is immediately reset the moment A = B.

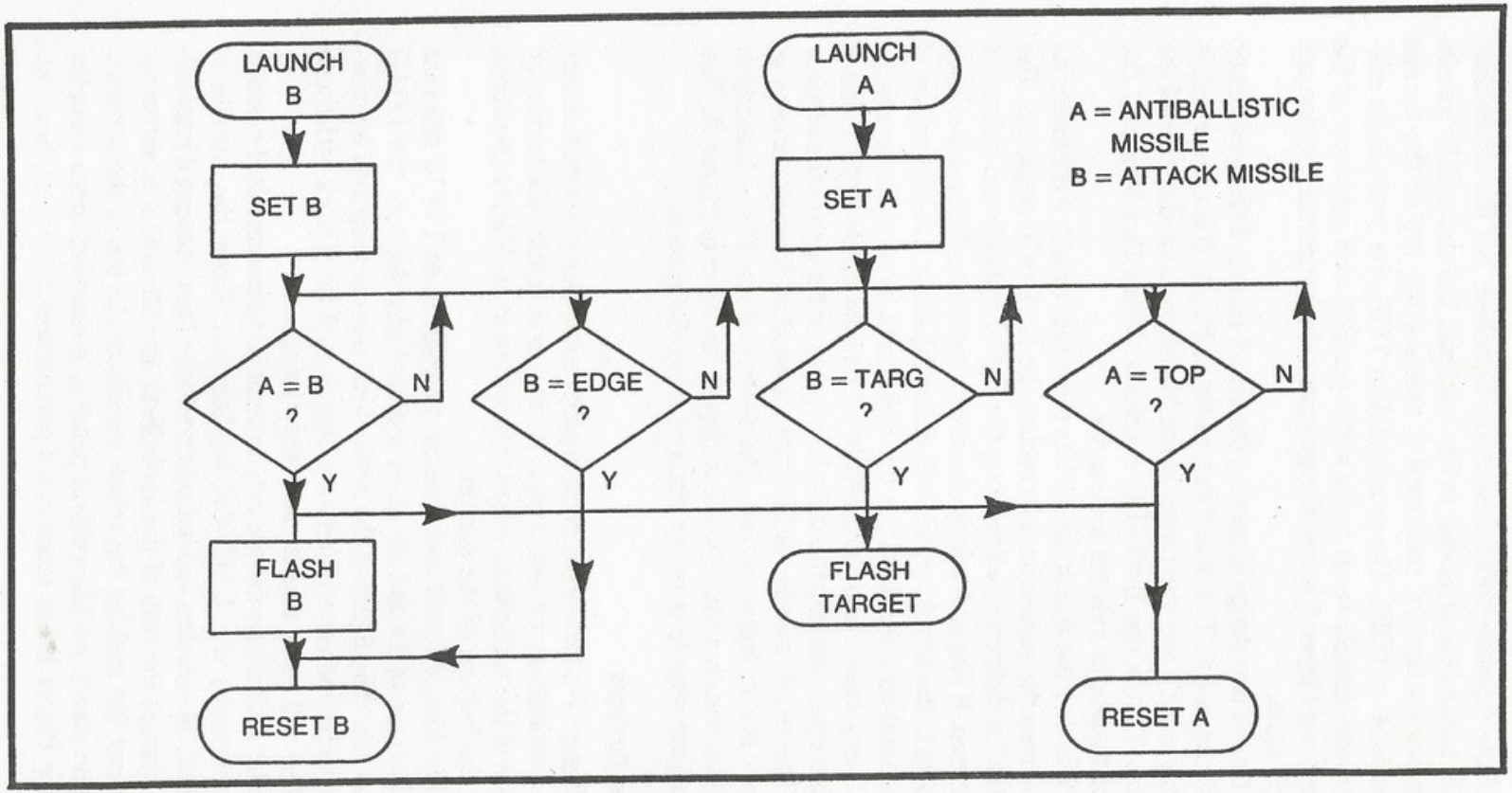


Fig. 7-1. Flowchart for Missile Attack II.

Now suppose both missiles are launched, but the antiballistic missile misses the attacker. In this instance, the antiballistic missile continues its upward path until it reaches the top of the screen (decision A = TOP). The moment A = TOP, the antiballistic missile is immediately reset to its initial position at the bottom of the screen, and Player A is free to reprogram the horizontal position and launch again.

One of two things happen to the attack missile if the antiballistic missile misses it. The best thing, as far as Player B is concerned, is that his missile hits the target in the lower right-hand corner of the screen. This is the B=TARG condition. When that happens, the target flashes to indicate a clean hit.

Whether the attack missile hits the target or not, it continues its path across the screen until it reaches the B=EDGE condition. The "EDGE" is defined as a line along the right-hand side and bottom of the screen. If Player A misses the attack missile and Player B misses the target, the attack missile will either fall short (hit the bottom of the screen) or overshoot (hit the right-hand side of the screen.)

In any case, the attack missile is immediately reset to its initial position upon reaching the right-hand edge or bottom of the screen.

The attack missile, incidentally, is blanked from the screen as long as it is in its initial position. This feature allows Player B to adjust the initial launch altitude without tipping his hand to Player A. The antiballistic missile is never blanked from the screen.

Player Controls

Figure 7-2 shows the control panel and other relevant features for the antiballistic missile, Player A. Player A adjusts the horizontal position of the antiballistic missile anywhere in 16 different positions along the bottom of the screen.

The idea is to set switches SF through SI to 1 of 16 different combinations of 1s and 0s, then momentarily depress the LOAD pushbutton. The figure of the antiballistic missile responds by moving to the position specified by the switches. A bit of work with these switches will show exactly how they work.

The antiballistic missile can be launched from its initial position by depressing the LAUNCH pushbutton. Once the missile is launched, all switches and buttons are locked out, making it impossible to change the path of the antiballistic missile until it is automatically reset (by striking the attack missile or the top of the screen).

The attack missile control panel is somewhat more complex because Player B has more initial parameters to control. See Fig. 7-3.

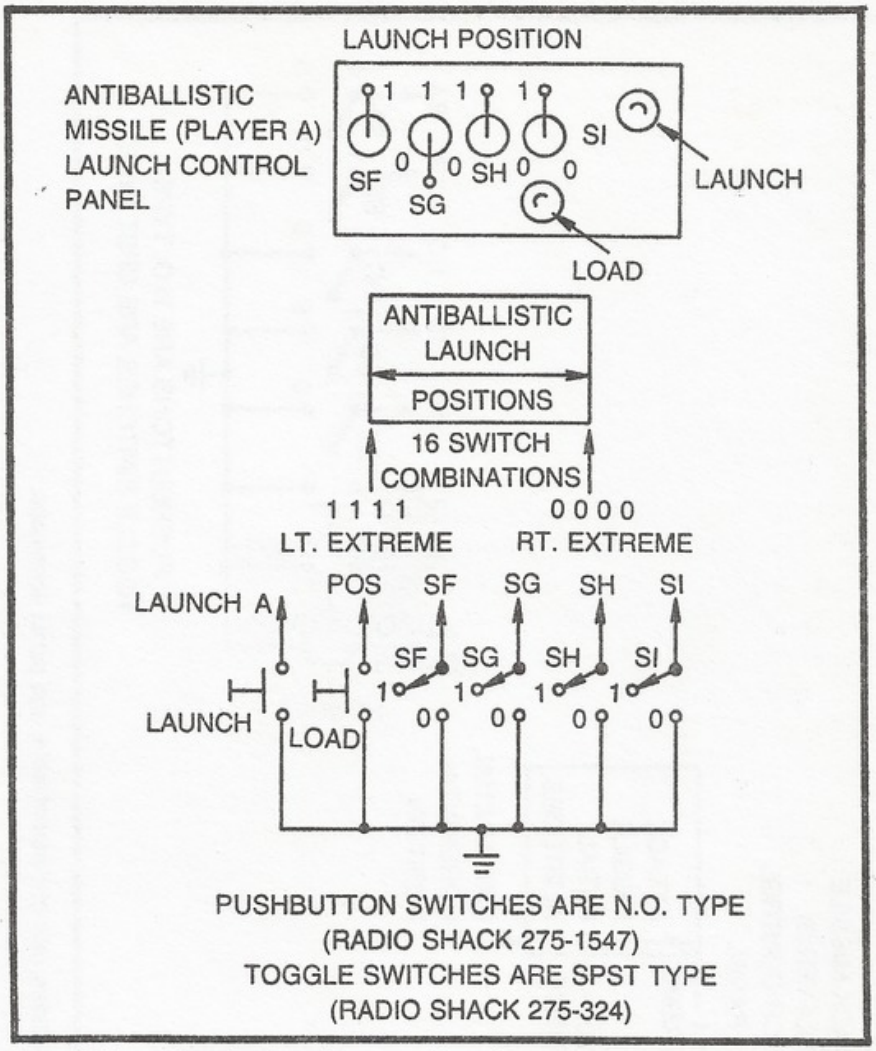


Fig. 7-2. Player A control panel, range of launch positions, and panel schematic.

Player Bs controls are enabled only while the attack missile is in its blanked, initial position. Player B can set the initial attack altitude to any one of eight positions in the upper half of the screen by setting the positions of switches SC, SD, and SE. Setting all three INITIAL ALTITUDE switches to their 1 positions fixes the maximum altitude, while setting them all to their 0 positions fixes the minimum altitude. There is no need to depress any sort of loading button in this particular case.

Player B also sets the attack velocity and rate of descent. There are four possible velocities and rates of descent set by switches SA and SB. See the chart in Fig. 7-3. Setting the attack velocity is a matter of first setting SA and SB to the desired positions, then depressing the VELOCITY pushbutton. The same two switches,

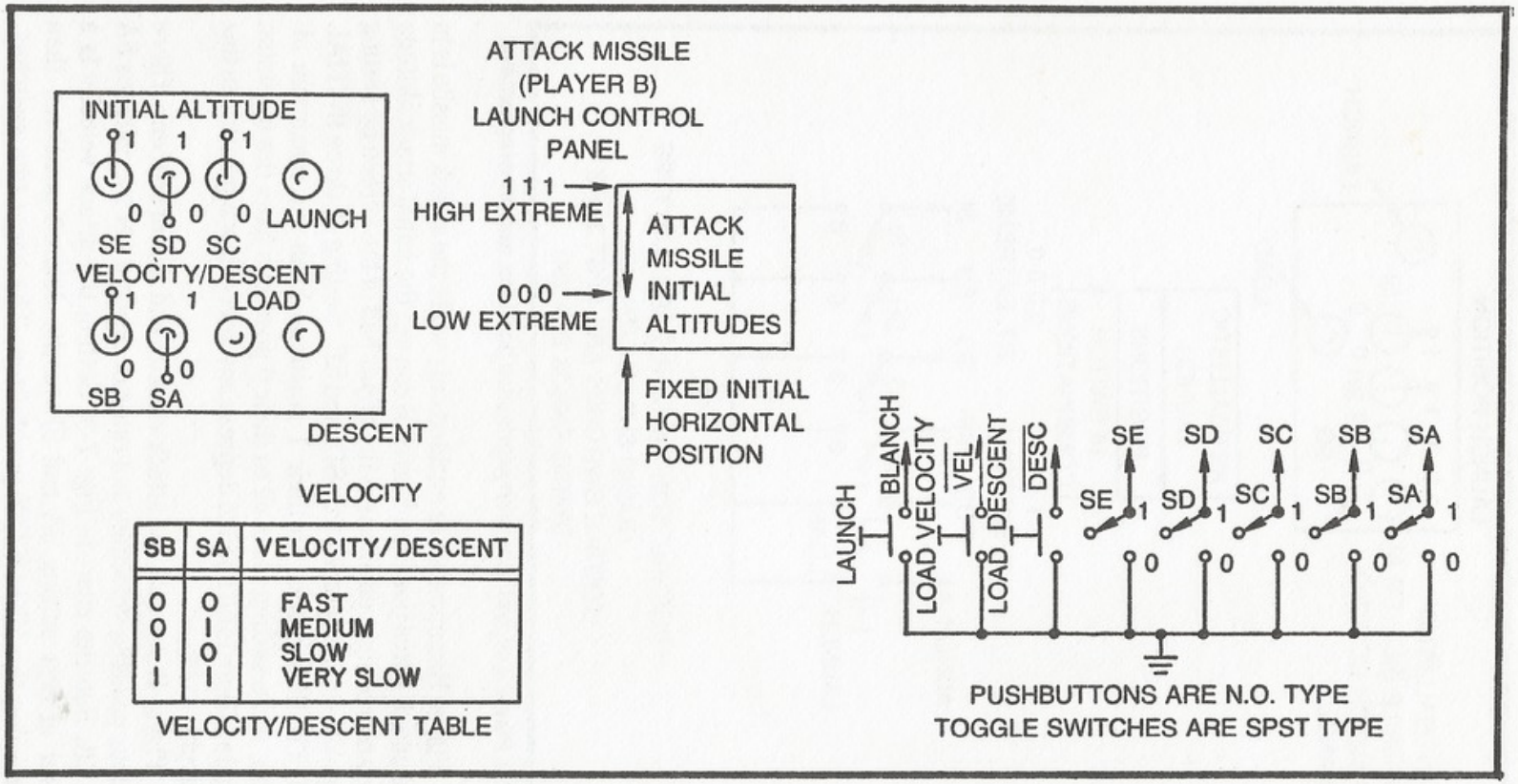


Fig. 7-3. Player B control panel, range of launch altitudes, rate-of-descent table, and panel schematic.

SA and SB, are used for setting the rate of descent, but that parameter is loaded into the system by depressing the DESCENT pushbutton.

The attack missile is then launched by depressing the LAUNCH pushbutton. Once the LAUNCH button is depressed, Player B is committed to the path he programmed just prior to the launching operation. All controls are locked out until the attack missile is blanked and reset to its initial position.

There are a number of different attack velocities, rates of descent and initial altitudes that direct the attack missile to the target. Player B's task is to come up with one of these proper combinations that avoids the antiballistic missile. Part of Player B's strategy, however, might be to throw Player A out of position with a false run, then make a deadly strike the next time around.

The Flashing Image Circuit

The only portion of this game that has not beem considered in some detail in earlier chapters is the one that causes the images to flash when they are hit. The attack missile flashes whenever the antiballistic missile hits it, and the target flashes whenever the attack missile hits it. In either case, the flashing effect serves as positive confirmation of a score.

Figure 7-4 illustrates the basic image-flashing circuit that will be used in a number of different games throughout this book. The IC in this case is a 556 dual timer. Section IC1-A is connected as a monostable multivibrator that is triggered into action by a brief negative-going signal, FLASH START.

The output of IC1-A can be used for resetting or timing other game operations, but more importantly, this timing allows a free-running multivibrator to generate a pulsing output signal. Whenever this oscillator, IC1-B, is gated off, it delivers a logic-0 level to one input of IC2. In the case of a NOR gate, this means the gate is open, allowing an inverted version of an IMAGE signal to pass through uninterrupted.

The moment the oscillator is gated on, however, it generates sequences of 1 and 0 outputs, thereby interrupting the IMAGE signal at the same rate. The overall effect is that the IMAGE is flashed on and off at a rate determined by the values of R2, R3, and C2, and for a period of time determined by the values of R1 and C1.

Game Block Diagram

Figure 7-5 shows the three major portions of the Missile Attack II game. The first diagram, Fig. 7-5a, represents the main control

portion for the antiballistic missile, Player A. Player B's control circuitry is blocked out in Fig. 7-5b. The overall game control scheme is shown in Fig. 7-5c.

Referring to the block diagram in Fig. 7-5a, assume the antiballistic missile is in its reset condition, resting at the bottom of the screen. The ALAUNCH signal is in a logic-1 condition at this time and the R-S flip-flop is allowing two things to happen. First, it is enabling the horizontal position logic circuit so that Player A can manually adjust the horizontal position of his missile via the switches on his control panel. Second, the reset flip-flop is in a condition where the initialization-control circuit is feeding VRST pulses to the vertical-slipping counter. The significance of this latter condition is that the slipping counter is synchronized to the vertical-counting system in the Sourcebox. There is no vertical motion of the antiballistic missile.

The figure logic block is responsible for combining the antiballistic missile's horizontal data (from the horizontal-position logic block) and its vertical data (from the VM outputs of the slipping counter) to create the image on the screen. The antiballistic missile's figure signal is designated AFIG in Fig. 7-5a.

As long as the antiballistic missile is in its reset condition, then, it cannot move in a vertical direction because its vertical slipping counter is synchronized to the 60-Hz VRST signal from the Sourcebox, but it can be manually moved in the horizontal direction by the control panel and horizontal-position logic circuit.

Player A launches the antiballistic missile by depressing his LAUNCH pushbutton. This action generates a brief negative-going pulse at ALAUNCH, setting the R-S flip-flop to its launch mode. Two things happen then: The horizontal-position logic is locked out so that the player no longer has control over the missile's horizontal position, and the initialization-control circuit is set to allow the slipping-counter action to take place. The antiballistic missile thus rises from the bottom of the screen at a rate determined by the VC programming of the vertical-slipping counter. (The programming for this counter is not shown in Fig. 7-5a because it is internally fixed at a relatively fast rate).

All of the circuits blocked out in Fig. 7-5a can be found in earlier chapters of this book. Look for the R-S flip-flop, horizontal-position logic, and initialization-control circuits in Chapter 6. The vertical-slipping counter is found in Chapter 5, and the figure logic is a simple variation of the basic figure-generating circuits in Chapter 3.

Figure 7-5b outlines the main control circuits for the attack missile. This system includes elements that are nearly identical to

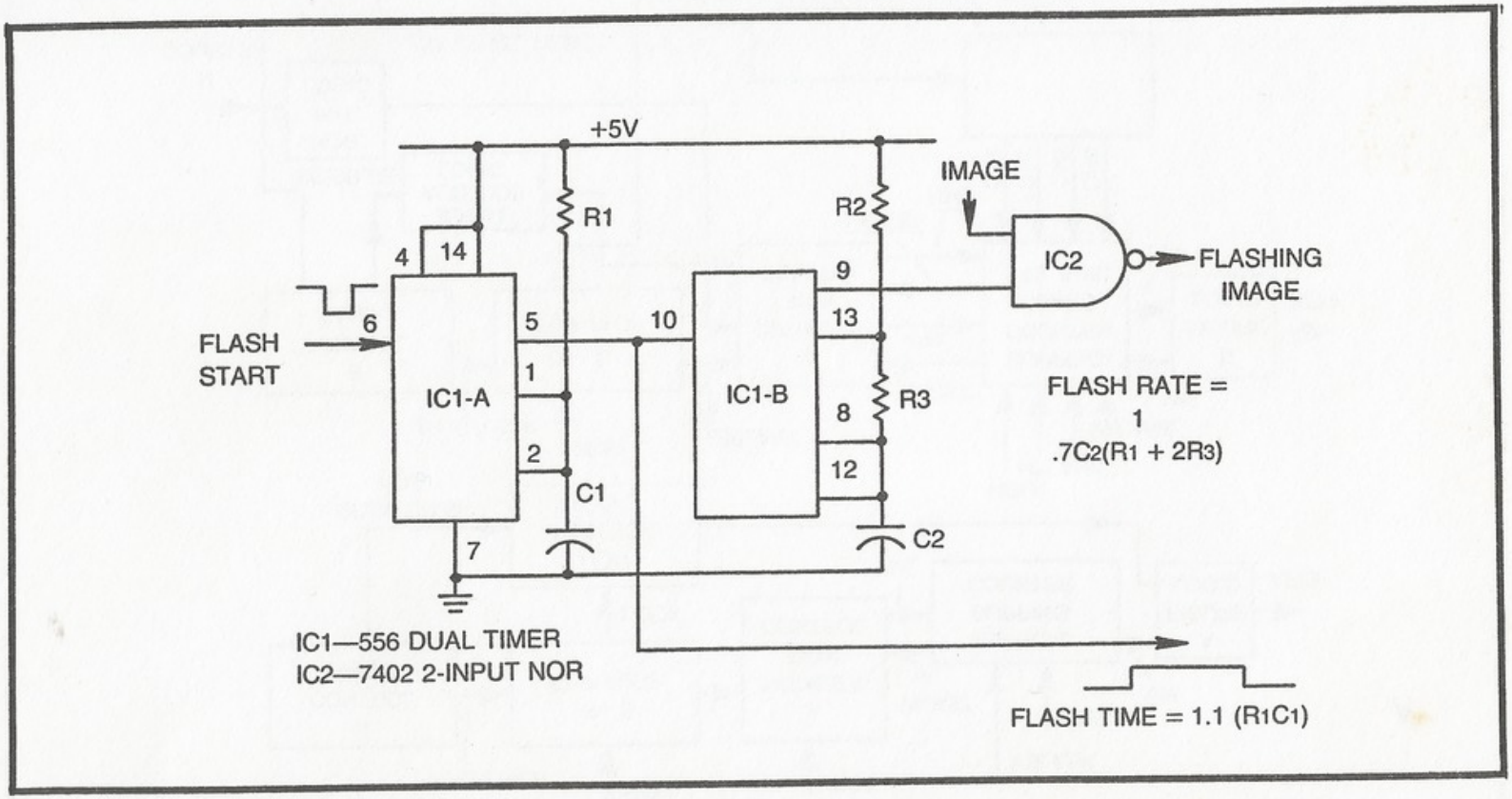
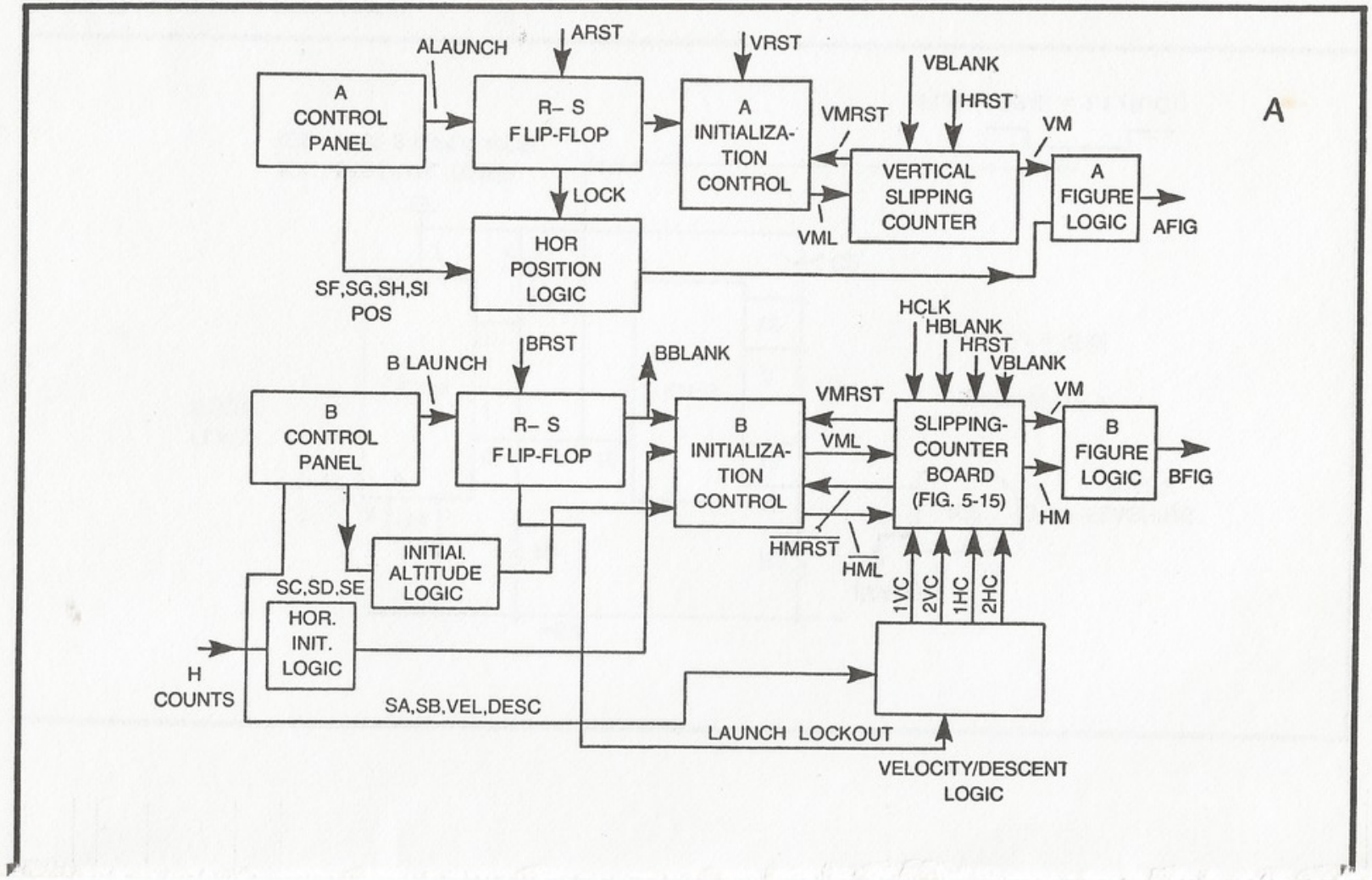


Fig. 7-4. The basic image flashing circuit.


A
ALAUNCH
ARST
VRST
VBLANK
HRST
A CONTROL PANEL
R– S FLIP-FLOP
A INITIALIZA-TION CONTROL
VMRST
VM
VERTICAL SLIPPING COUNTER
A FIGURE LOGIC
AFIG
LOCK
VML
HOR POSITION LOGIC
SF,SG,SH,SI POS
HCLK
HBLANK
HRST
VBLANK
B LAUNCH
BRST
BBLANK
VMRST
VM
B CONTROL PANEL
R– S FLIP-FLOP
B INITIALIZA-TION CONTROL
VML
SLIPPING-COUNTER BOARD (FIG. 5-15)
B FIGURE LOGIC
BFIG
HM
HMRST
HML
INITIAL ALTITUDE LOGIC
SC,SD,SE
1VC
2VC
1HC
2HC
HOR. INIT. LOGIC
H COUNTS
SA,SB,VEL,DESC
LAUNCH LOCKOUT
VELOCITY/DESCENT LOGIC

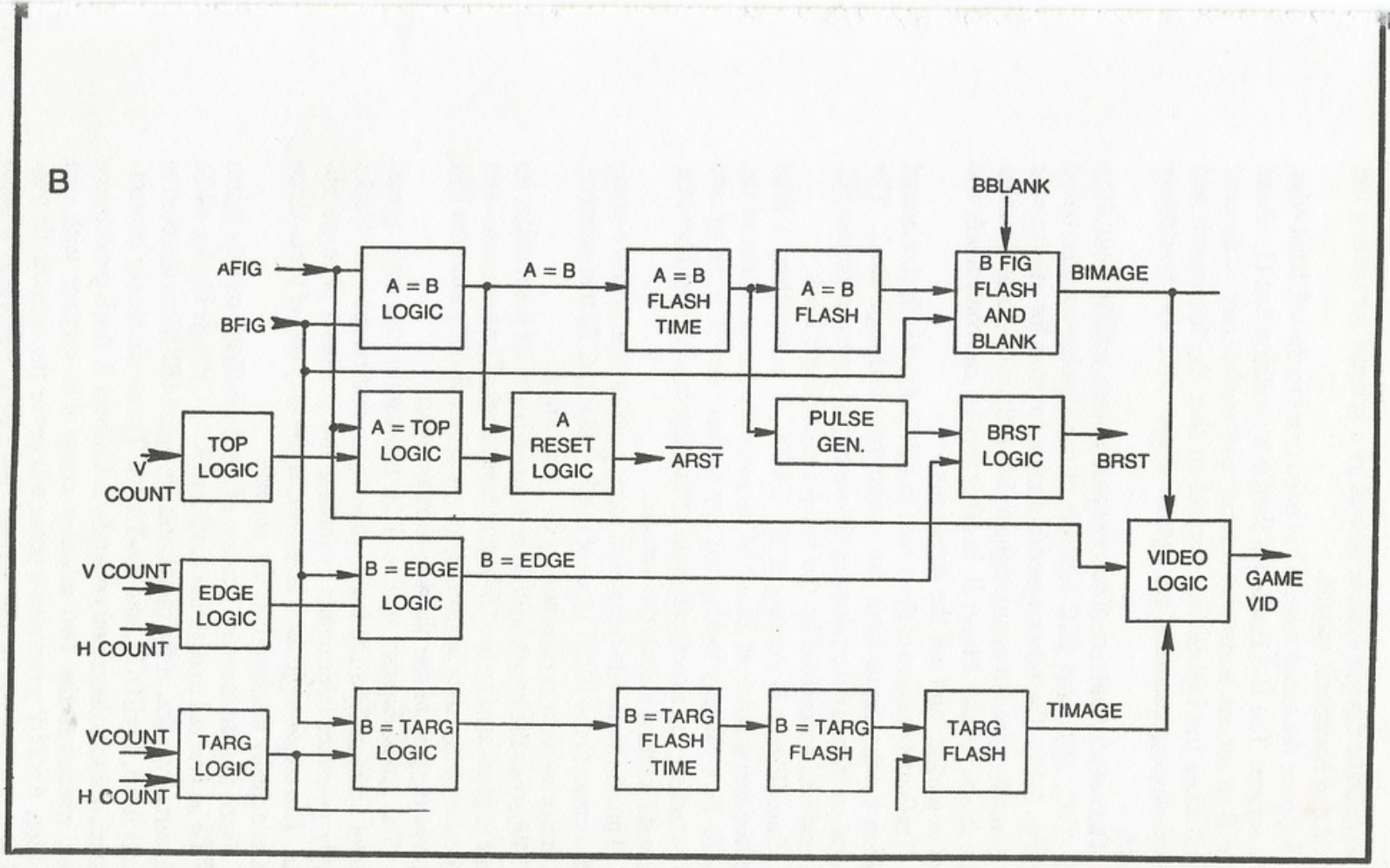


Fig. 7-5. Block diagrams for Missile Attack II. (a) Antiballistic missile section. (b) Attack missile section. (c) Game logic.

the antiballistic missile system as well as a scheme for setting the vertical and horizontal speeds.

Suppose the attack missile has been reset to the left-hand edge of the screen. The $\overline{R}$-$\overline{S}$ flip-flop is then in a condition that (1) allows Player B to set the horizontal velocity and vertical rate of descent and (2) fixes the initialization control so that the horizontal- and vertical-slipping counters are synchronized with the Sourcebox counts.

The velocity and rate of descent parameters actually control the 1VC, 2VC, 1HC, and 2HC controls of the slipping-counter board (see Fig. 5-15). The other speed/direction controls for this counter are internally wired to assure right-to-left and top-to-bottom motion of the attack missile. Player B, in other words, can control only the rates of motion, and not the directions.

The initial position of the attack missile is fixed in the horizontal position by the signal from the horizontal-initial-logic block. The vertical position is determined by the setting of control switches SC, SD, and SE, assembled by the initial-altitude-logic block.

Once Player B depresses his LAUNCH pushbutton, a brief negative-going pulse at $\overline{\text{BLAUNCH}}$ switches the condition of the $\overline{R}$-$\overline{S}$ flip-flop, thereby locking out any further control of the missile and switching the synchronization of the slipping-counter board for automatic motion across the screen.

The attack missile remains in motion until a brief positive-going pulse occurs at the BRST input of the $\overline{R}$-$\overline{S}$ flip-flop. At that moment, the entire system is returned to its reset state.

Whether the attack missile is launched or in its reset state, its image is generated by the "B" figure logic block. The horizontal- and vertical-counting data for this figure are taken directly from the HM and VM outputs of the slipping-counter board.

The block diagram in Fig. 7-5c represents the main game-control system. The top-logic, edge-logic, and target-logic blocks simply generate information regarding the positions of those objects. The target-logic data, however, is the only one of these three that is actually displayed on the screen.

Note that an inverted version of the antiballistic missile figure ($\overline{\text{AFIG}}$) is directed to both the A=B and A=TOP logic blocks, while an inverted version of the attack-missile figure (BFIG) is directed to blocks A=B, B=EDGE, and B=TARG. These are simply contact-detection blocks described generally in Chapter 6. A=B generates a pulse whenever the two missiles come into contact with one another, A=TOP generates a pulse whenever the antiballistic missile reaches the top of the screen, B=EDGE generates a pulse

whenever the attack missile contacts either the bottom or right-hand edge of the screen, and B=TARG generates a pulse whenever the attack missile hits the target.

Much of what remains to be explained in Fig. 7-5c can be determined from the flow chart in Fig. 7-1. The A-reset logic block, for instance, is responsible for resetting the position of the antiballistic missile whenever A=B or A=TOP. This reset pulse is designated $\overline{\text{ARST}}$.

Whenever A=B occurs, the pulse from that block also initiates a flash-time timer in the A=B flash-time block. This block, in turn, generates a logic-1 level that entables the A=B flasher. And when the timing is over, the pulse generator produces a brief pulse that ultimately resets the position of the attack missile.

The attack missile is also reset the instant it reaches the bottom or right-hand edge of the screen. The BRST logic block thus generates a negative-going attack missile reset pulse ($\overline{\text{BRST}}$) whenever the A=B flash timing is over or the missile contacts the edge of the screen.

Whenever the B=TARG block senses a contact between the attack missile and the target, it also initiates a flash timer. This timer enables the B=TARG flashing circuit which, in turn, switches the target-image data off and on in the target-flash block.

Returning to the attack-missile figure for a moment, note the B-figure flash and blank block has two separate control inputs. The image data for the attack missile (BFIG) can be completely blanked off the screen while it is in its reset position. The $\overline{\text{BBLANK}}$ signal from the attack-missile control system is responsible for this blanking effect.

The same attack missile image, however, is also blanked on and off by the A=B flash circuit. In either case, the unblanked, blanked, or flashing image data emerges as the BIMAGE signal.

The BIMAGE, $\overline{\text{AFIG}}$, and TIMAGE signals are all combined into the game's composite figure video at the video-logic block.

The flashing circuits in Fig. 7-5c are all described in connection with the circuit in Fig. 7-4; the top-, edge-, and target-logic blocks are variations of the figure generators in Chapter 2. The contact-sensing circuits are generally described in Chapter 6.

If you have been studying this book diligently, you will find nothing new here at all.

Circuit Diagrams

Figures 7-6, 7-7, and 7-8 show the circuit diagrams for the Missile Attack II game. The game calls for four circuit boards, the

three just mentioned and a slipping-counter board from Fig. 5-15. Of course there are two separate control panels that have already been described in Figs. 7-2 and 7-3.

After studying the game's flow chart and block diagrams in some detail, there is little need for a highly detailed description of the circuits themselves. The overall wiring block diagram in Fig. 7-9 will prove invaluable when analyzing the operation of the system at the circuit-board level.

Figure 7-6 shows a board containing most of the control elements for the antiballistic missile system. ICs 1-A and 6-F, along with C1 and R7, merely transform the negative-going launch signal into a brief, negative-going pulse. IC2-A and IC2-C make up the $\overline{R}$-$\overline{S}$ flip-flop that is set so that pin 3 is at logic 1 and pin 8 is at logic 0 whenever that pulse occurs.

ICs 2-B, 2-D, and 1-B make up the initialization control circuit. Whenever the antiballistic missile is in its reset position, this circuit delivers an inverted VRST pulse to the loading input of the vertical-slipping counter, IC7, and IC8. Whenever the missile is launched, however, the initialization-control circuit delivers the output of IC1-D back to the load input of the slipping counter.

The four sections of IC3 determine the pattern of inverted and noninverted H-count signals for determining the initial horizontal position of the antiballistic missile. See details in connection with the circuit in Fig. 6-17. These position-determining signals pass directly through IC9, a quad D latch, as long as the system is in its reset condition and the LOAD button is depressed. IC4-A is responsible for ANDing the launch status information from the $\overline{R}$-$\overline{S}$ flip-flop and the $\overline{POS}$ logic level from the player's LOAD button.

Whenever the LOAD button is released or the missile is launched, pins 4 and 13 of IC9 go to logic 0, thereby placing the latches into their memory modes. The horizontal-position data that existed the moment the circuit is latched then remains fixed at the inputs of IC5-A.

The position of the antiballistic missile in the horizontal plane is actually defined by IC5-A. Since this pluse is too wide for generating the horizontal width of the antiballistic missile, it is shorted in the circuit made up of IC1-C, IC6-C, R6, and C2. The value of C2 can be changed to suit the designer's own impression of how wise the antiballistic missile figure ought to be.

The height and vertical position of the antiballistic missile are both determined by the pin-12 output of the higher-order vertical-slipping counter, IC8. This output is inverted by IC6-D and effec-

tively ANDed with the horizontal portion of the image at IC4-B. The noninverted version of the missile figure is then inverted by IC6-E before applying it to other portions of the system.

The circuit in Fig. 7-7 deals mainly with the control aspects of the attack missile. The launching pulse is formed by IC1-A and IC4-A in a fashion identical to the launching circuit for the antiballistic missile. The $\overline{R}$-$\overline{S}$ flip-flop, composed of IC1-B and IC1-C, is set to its launch condition by the negative-going launch pulse from IC4-A, and it is reset by a negative-going version of the BRST pulse from the output of IC4-B.

The initialization-control portion of the attack missile circuit is embodied in IC8, actually a quad 2:1 multiplexer. Only two of the four sections are used here, but that is adequate for shifting the operation of the slipping-counter board (Fig. 5-15) between the initializing pulses and motion-generating pulses.

The attack-missile system is initialized as long as the output from IC1-B is at logic 0, thereby directing vertical-synchronizing pulses to the $\overline{VMRST}$ and horizontal-synchronizing pulses to $\overline{HMRST}$ inputs of the slipping counter.

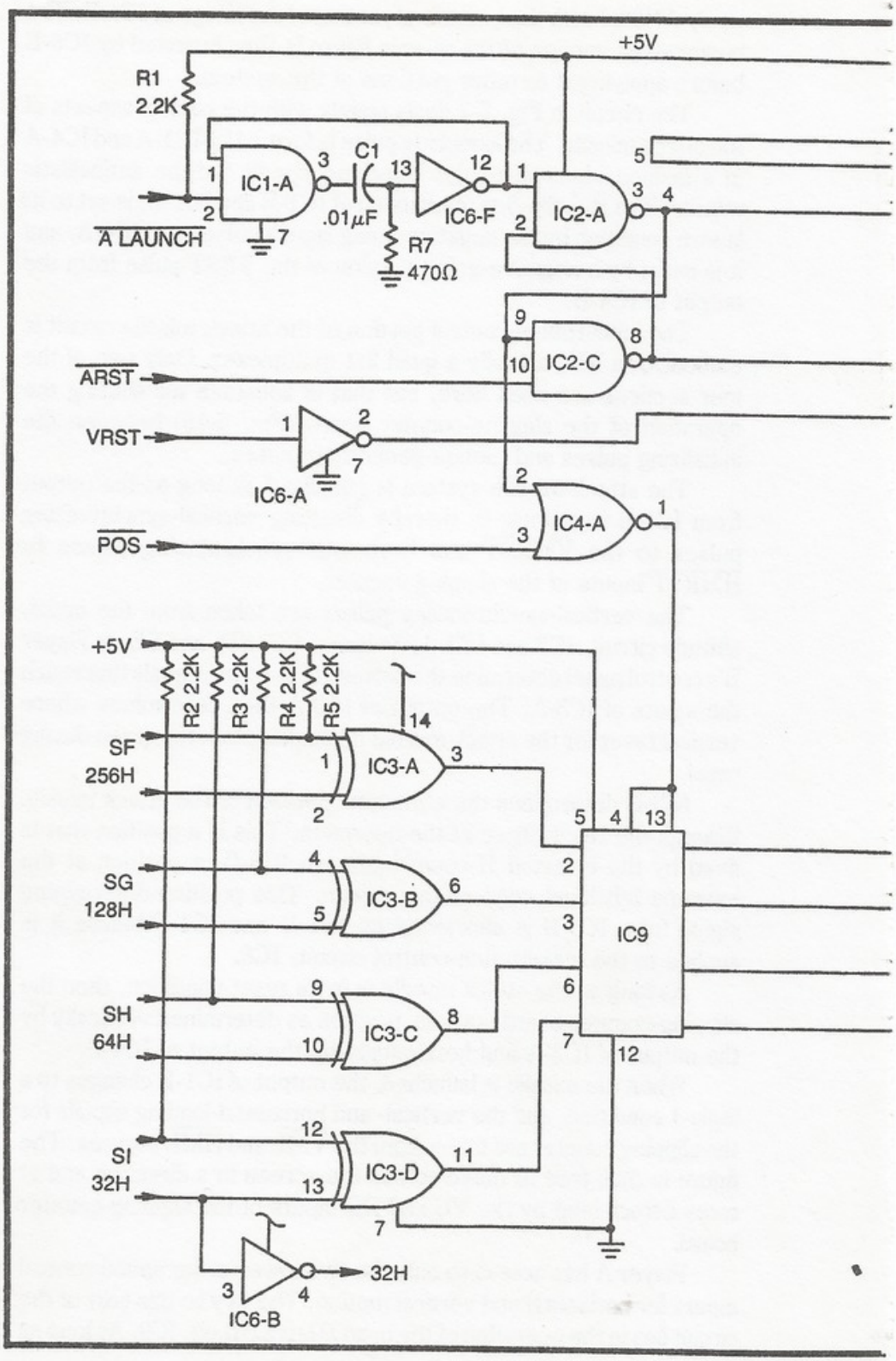
The vertical-synchronizing pulses are taken from the initial-altitude circuit, IC7 and IC3-A. Switches SC, SD, and SE on Player B's control panel determine the pattern of V-count signals that reach the inputs of IC3-A. The output of IC3-A then determines where vertical reset for the attack missile takes place on the screen during reset.

IC3-B determines the horizontal position of the attack missile through the reset phase of the operation. This is a position that is fixed by the inverted H-count inputs to IC3-B, a position at the extreme left-hand edge of the screen. This position-determining signal from IC3-B is shortened by IC1-D and IC4-F before it is applied to the initialization-control circuit, IC8.

As long as the attack missile is in its reset condition, then the slipping-counter board holds its position as determined vertically by the output of IC3-A and horizontally by the output of IC4-F.

When the missile is launched, the output of IC1-B changes to a logic-1 condition, and the vertical- and horizontal-loading signals for the slipping counter are taken from the $\overline{VML}$ and $\overline{HML}$ sources. The figure is thus free to move across the screen in a direction and at rates determined by the VC and HC inputs of the slipping-counter board.

Player A has access to only the two lower-order speed control inputs for horizontal and vertical motion. The key to this part of the circuit lies in the operation of the quad D latch circuit, IC9. As long as


+5V
R1
2.2K
C1
.01μF
R7
470Ω
IC1-A
IC6-F
IC2-A
IC2-C
IC6-A
IC4-A
IC3-A
IC3-B
IC3-C
IC3-D
IC9
IC6-B
A LAUNCH
ARST
VRST
POS
R2 2.2K
R3 2.2K
R4 2.2K
R5 2.2K
SF
256H
SG
128H
SH
64H
SI
32H

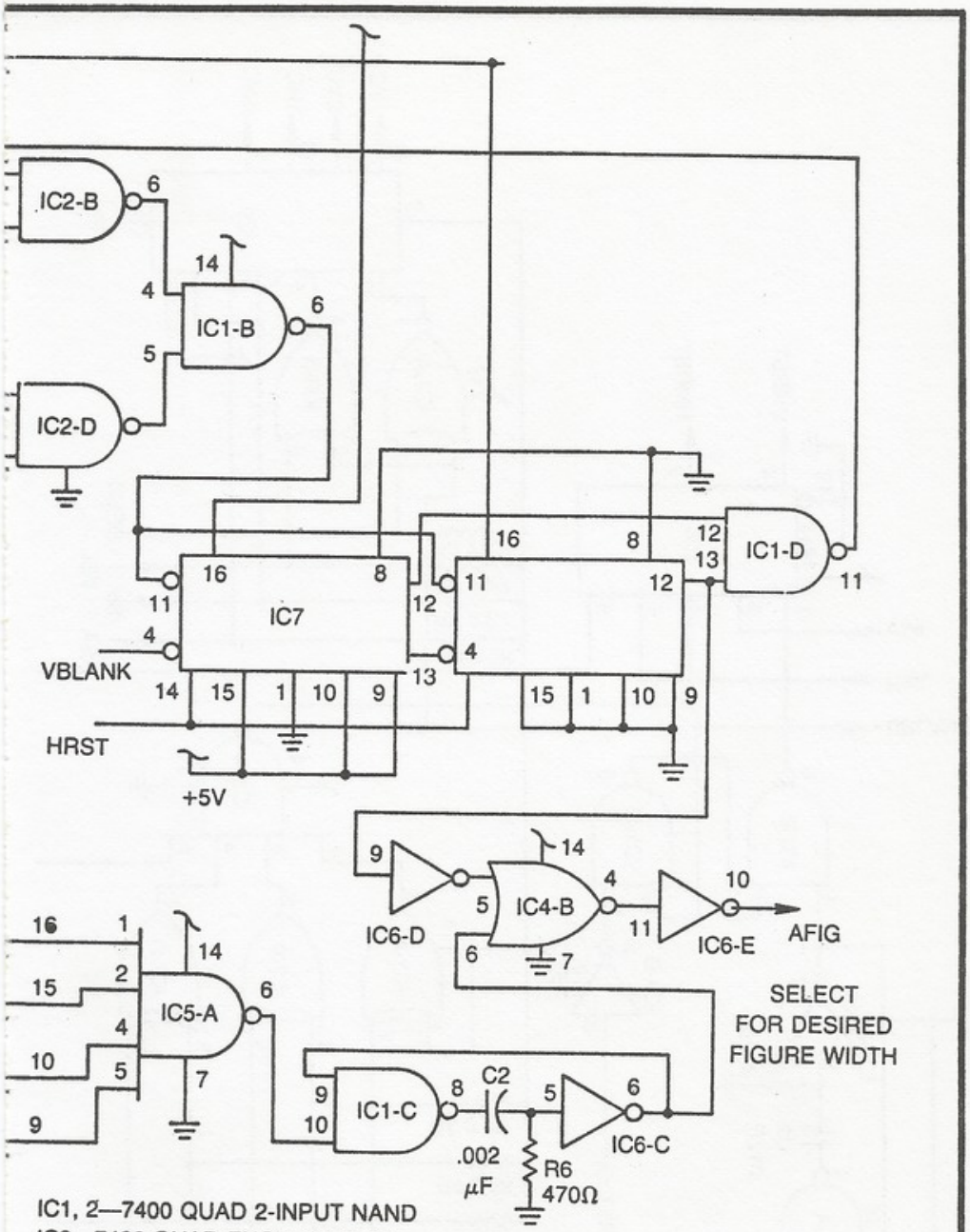
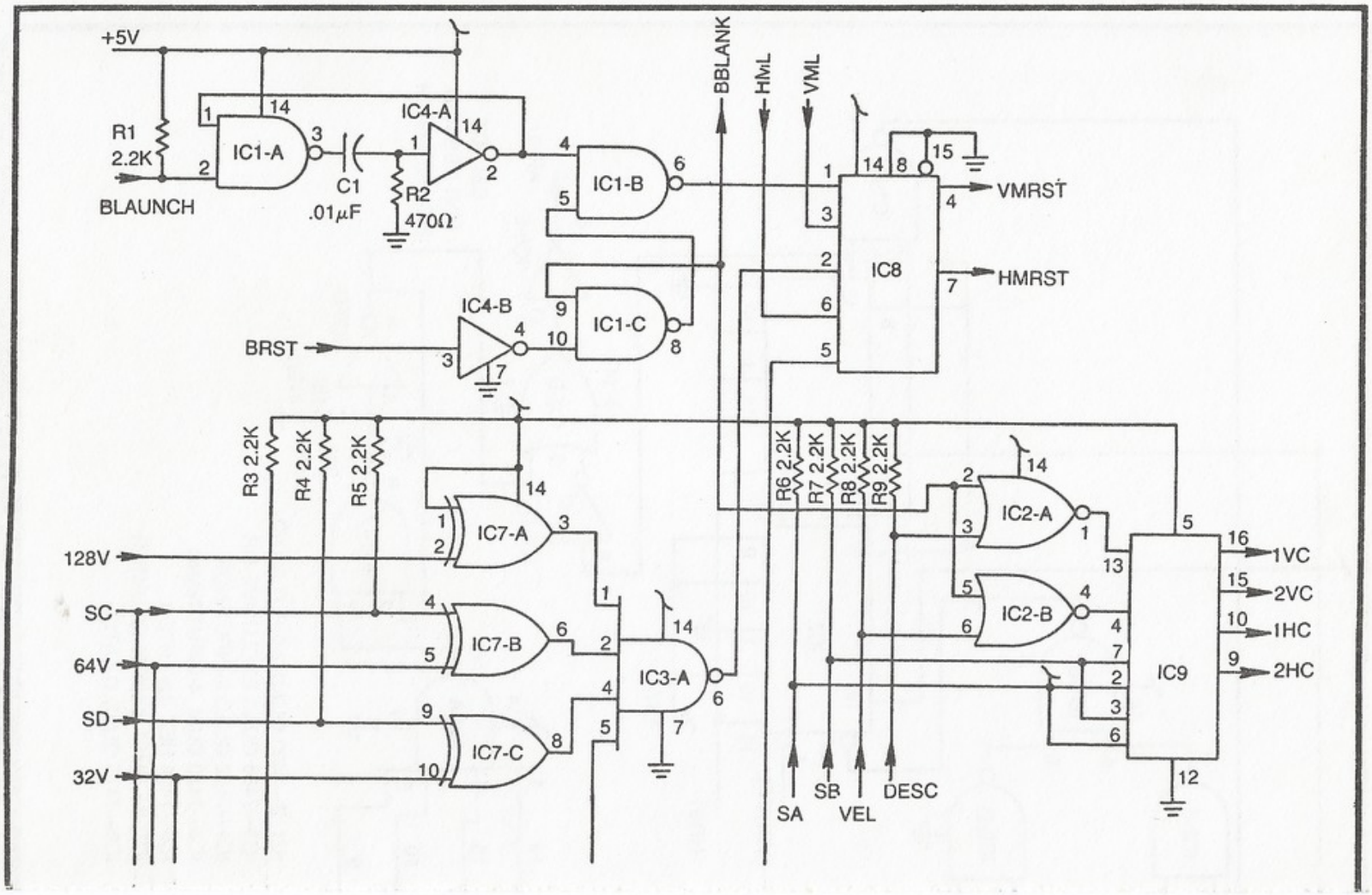


Fig. 7-6. Schematic diagram for the antiballistic missile control circuit board.


+5V
R1
2.2K
BLAUNCH
IC1-A
C1
.01μF
IC4-A
R2
470Ω
IC1-B
BBLANK
HML
VML
IC8
VMRST
HMRST
IC4-B
BRST
IC1-C
R3 2.2K
R4 2.2K
R5 2.2K
R6 2.2K
R7 2.2K
R8 2.2K
R9 2.2K
IC7-A
IC7-B
IC7-C
128V
SC
64V
SD
32V
IC3-A
IC2-A
IC2-B
IC9
1VC
2VC
1HC
2HC
SA
SB
VEL
DESC

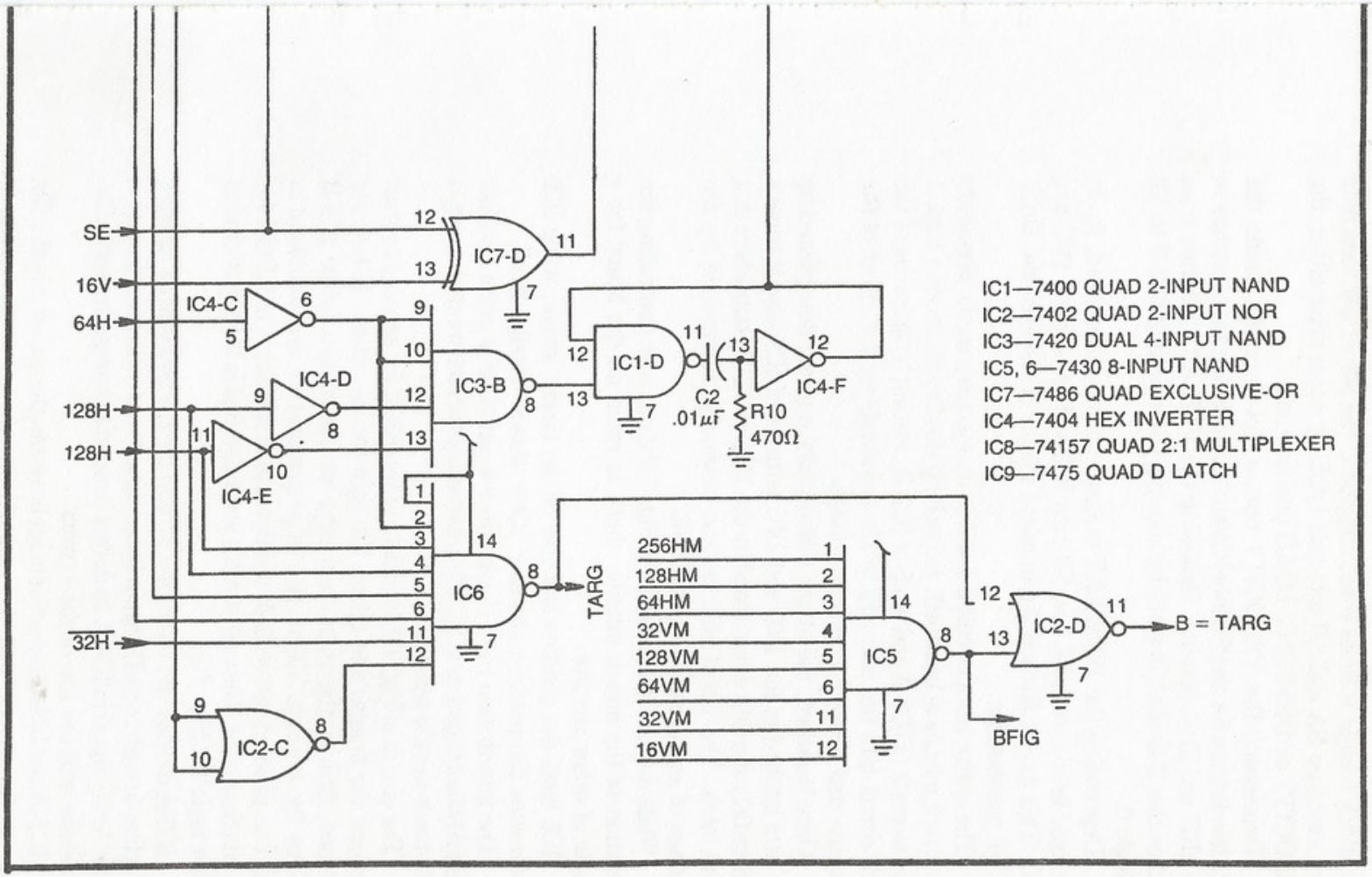


Fig. 7-7. Schematic diagram for the attack missile control circuit board.

the attack missile is in its reset condition, the player can load data from switches SA and SB into this latch by depressing either the VELOCITY or DESCENT LOAD pushbuttons.

Depressing the VELOCITY button, for example, feeds the logic levels from SA and SB into IC9 and through that set of latches to the 1HC and 2HC outputs. Releasing the VELOCITY button then latches that 2-bit data in place by allowing the output of IC2-B to fall to logic 0.

Depressing the DESCENT button, on the other hand, feeds the logic levels from SA and SB into IC9 and to outputs 1VC and 2VC. This data is latched the moment Player B releases the DES CENT pushbutton.

The entire latch circuit is fixed in its memory mode whenever the attack missile is launched. Launching the missile feeds a logic-1 level to pin 2 of IC2-A and pin 5 of IC2-B, placing both halves of the latch circuit into the memory mode and disabling the effect of the velocity- and descent-control switches.

Once launched, the attack missile falls at a rate determined by the data latched at the 1VC and 2VC outputs of IC9, and it travels horizontally at a rate determined by the 1HC and 2HC outputs of that same latch. The initial altitude was originally determined by the settings of switches SC through SE.

While most of the circuitry in Fig. 7-7 deals with controlling the operation of the attack missile, there is room on the board for a couple of other circuits.

IC6 fixes the position and size of the target image, while IC5 determines the position and size of the attack missile, itself.

The information from these two 8-input NAND gates is effectively ANDed together in IC2-D to sense contact between the target and attack-missile images.

The circuit in Fig. 7-8 is mainly responsible for sensing contact between the images and taking the appropriate action. IC1-A, for instance, defines the right-hand edge of the screen, while IC1-B defines the bottom. These two edge-parameters are combined in IC3-A to yield an active-high (noninverted) definition of EDGE for the attack missile. These three ICs are embodied in the block labeled "edge logic" in Fig. 7-5c.

IC8-A defines the top of the screen for the antiballistic missile. Its active-low (inverted) output is combined with an inverted version of the AFIG signal in IC4-A to detect contact between the antiballistic missile and the top of the screen.

IC5-A and IC5-C detect contact between the attack missile, the edge of the screen, and the antiballistic missile, respectively. If

there is contact between the attack missile and the edge of the screen, the output of IC5-A goes to logic 1 and ultimately through IC3-B as a pulse for resetting the position of the attack missile.

If, on the other hand, there is contact between the two missiles, the output of IC5-C goes to logic 1. This signal is inverted by IC5-D and applied to the trigger input of flashing-timer IC7-A. The output of that timer enables the missile-flashing IC, IC7-B. This rectangular waveform is inverted by IC2-E and applied to the attack-missile-logic circuit, IC1-C. When the flashing interval is over, the negative-going edge of the timing pulse at pin 5 of IC7-A is shaped and applied as one of two attack-missile-reset pulses to IC3-B.

Contact between the attack missile and the target is sensed by IC2-D in Fig. 7-7. This pulse is inverted by IC4-C in Fig. 7-8 and used for enabling the target-flashing timer, IC6-A. The flashing output from IC6-B modulates the target image from $\overline{\text{TARG}}$ at IC4-D, where it is then inverted and ORed with the image information for the two missiles in IC8-B.

A complete slipping-counter board is an integral part of the system, as shown in the wiring block diagram in Fig. 7-9. This board is already described in connection with the circuit diagram in Fig. 5-15. Figure 7-9 merely shows how it is interfaced with the three game boards just described.

Note carefully the programming of the 4HC, 8HC, 4VC, and 8VC connections to the slipping-counter board in Fig. 7-9. These connections must be made in order to make the attack missile move in the proper directions at all times.

Construction and Assembly Hints

Each of the boards described in this section can be built on a 4- by 4-inch plug-in boards (Radio Shack 276-153) and interconnected by means of edge-card connectors (Radio Shack 276-1551).

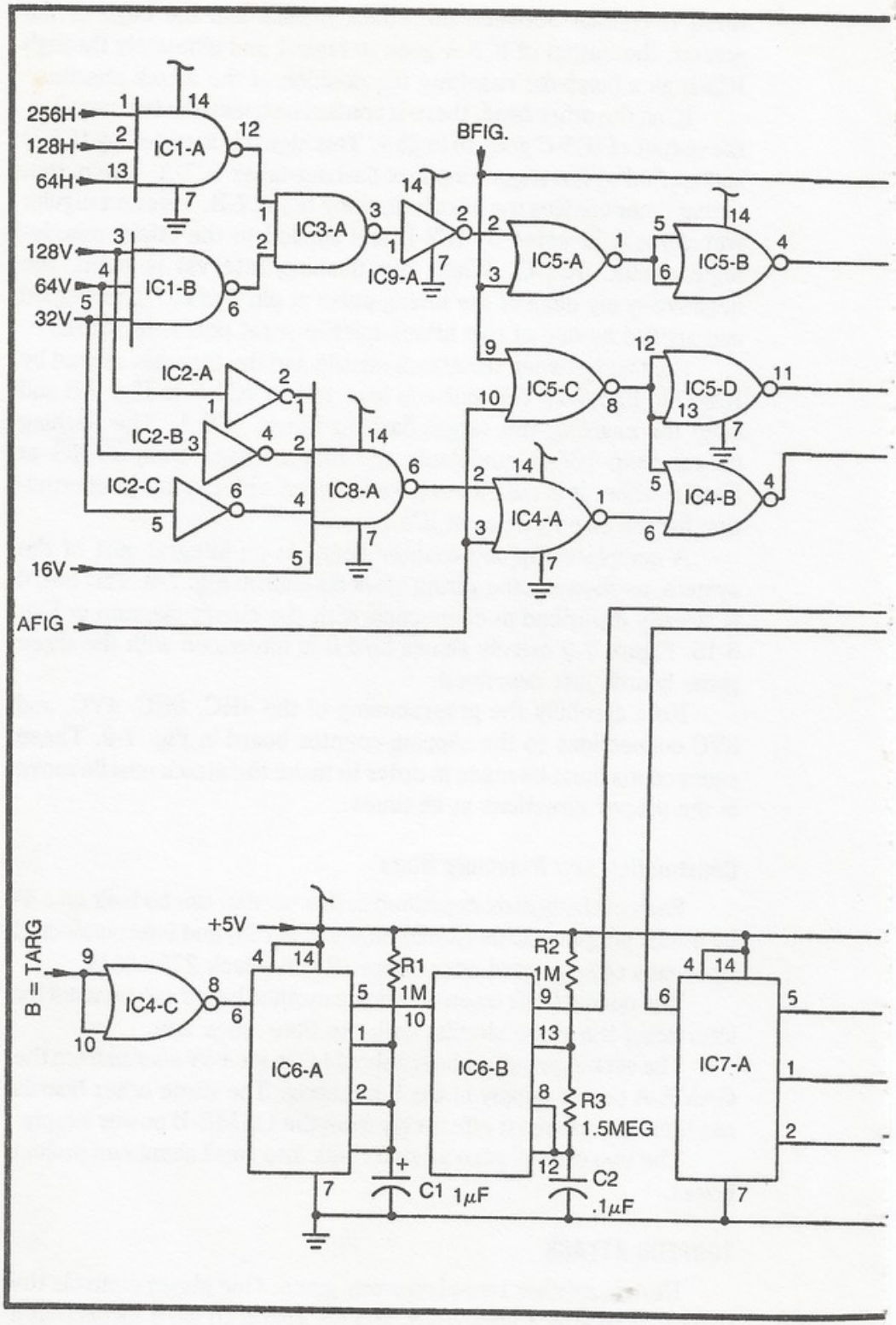
The connections on an additional mother board can be used for interfacing the game circuits with the Sourcebox unit.

The slipping-counter board should take its +5V source from the GAME-A power supply in the Sourcebox. The three other boards can then operate most effectively from the GAME-B power supply.

The two control panels can be built into small aluminum project boxes.

TORPEDO ATTACK

Here is another two-player war game. One player controls the motion of an attack craft (AC), and his goal is to lob a torpedo at a


256H
128H
64H
IC1-A
128V
64V
32V
IC1-B
IC3-A
IC9-A
BFIG.
IC5-A
IC5-B
IC5-C
IC5-D
IC2-A
IC2-B
IC2-C
IC8-A
IC4-A
IC4-B
16V
AFIG
B = TARG
IC4-C
+5V
R1
1M
R2
1M
IC6-A
IC6-B
IC7-A
R3
1.5MEG
C1 1μF
C2
.1μF

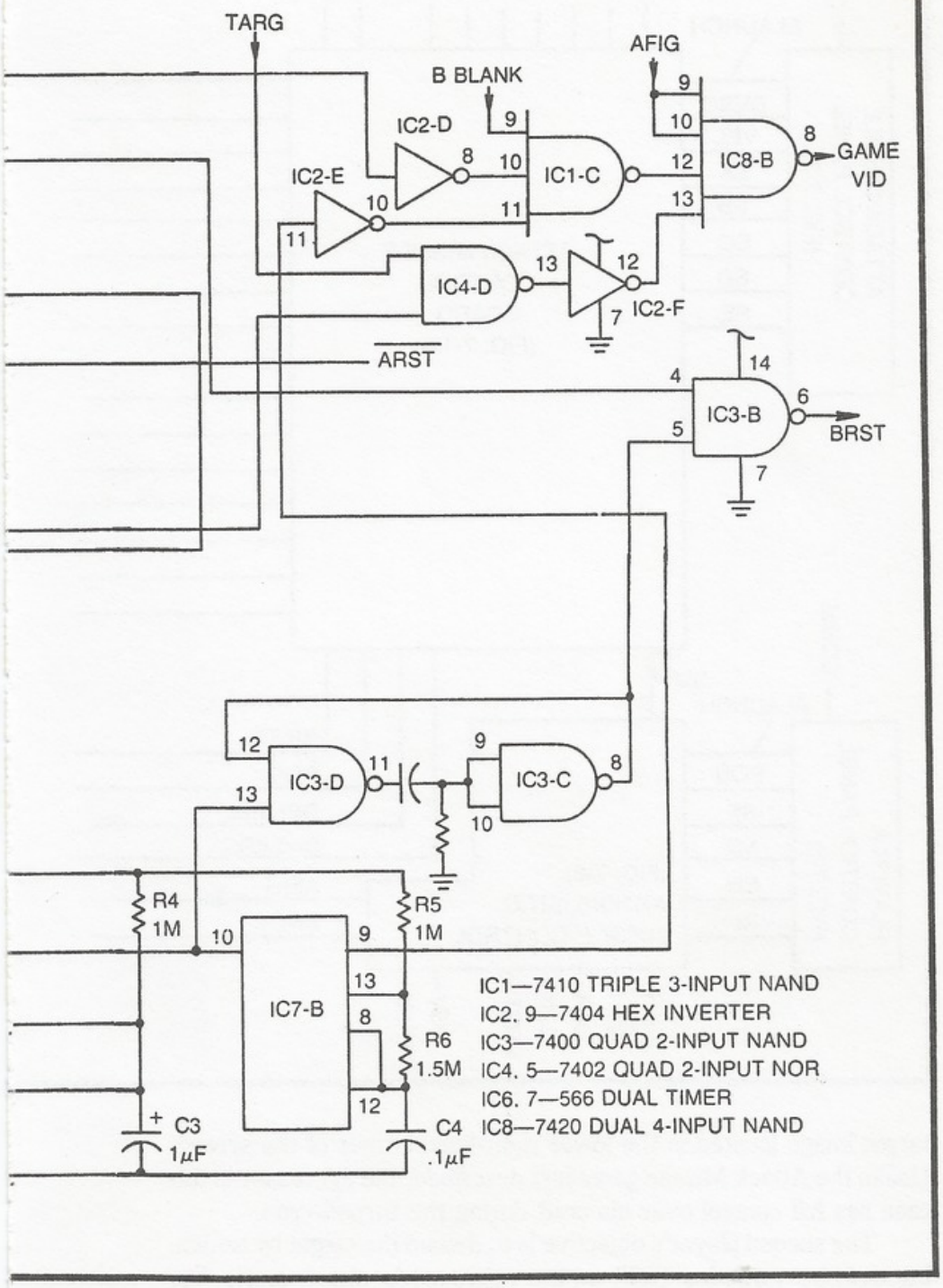
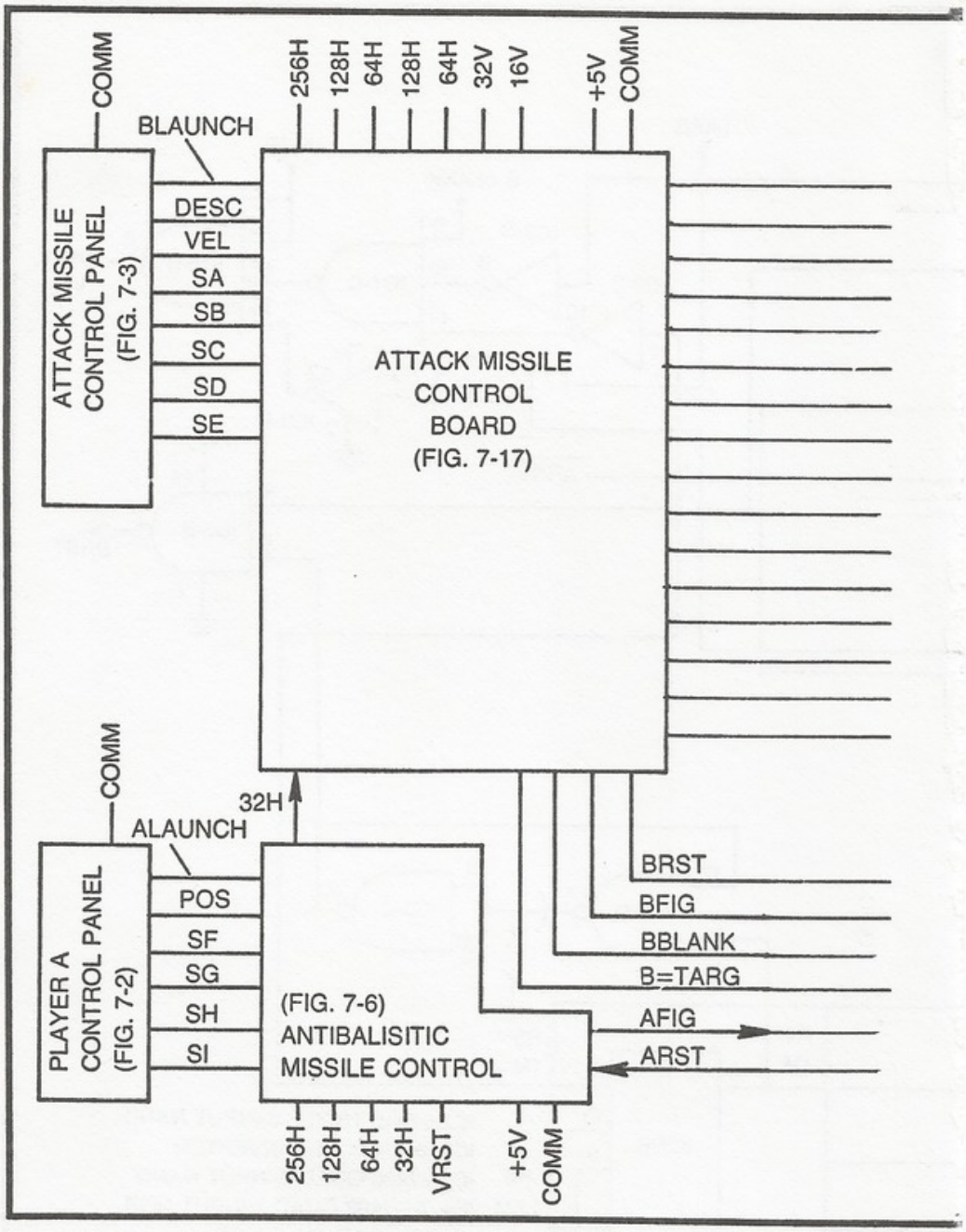


Fig. 7-8. Schematic diagram for the game logic circuit board.



target image located in the lower right-hand corner of the screen. Unlike the Attack Missile game just described, the aggressor in this case has full control over his craft during the torpedo run.

The second player's objective is to defend the target by launching defense torpedos (DT) at the aggressor's attack craft. The horizontal positions of the four defense torpedoes are fixed, but the

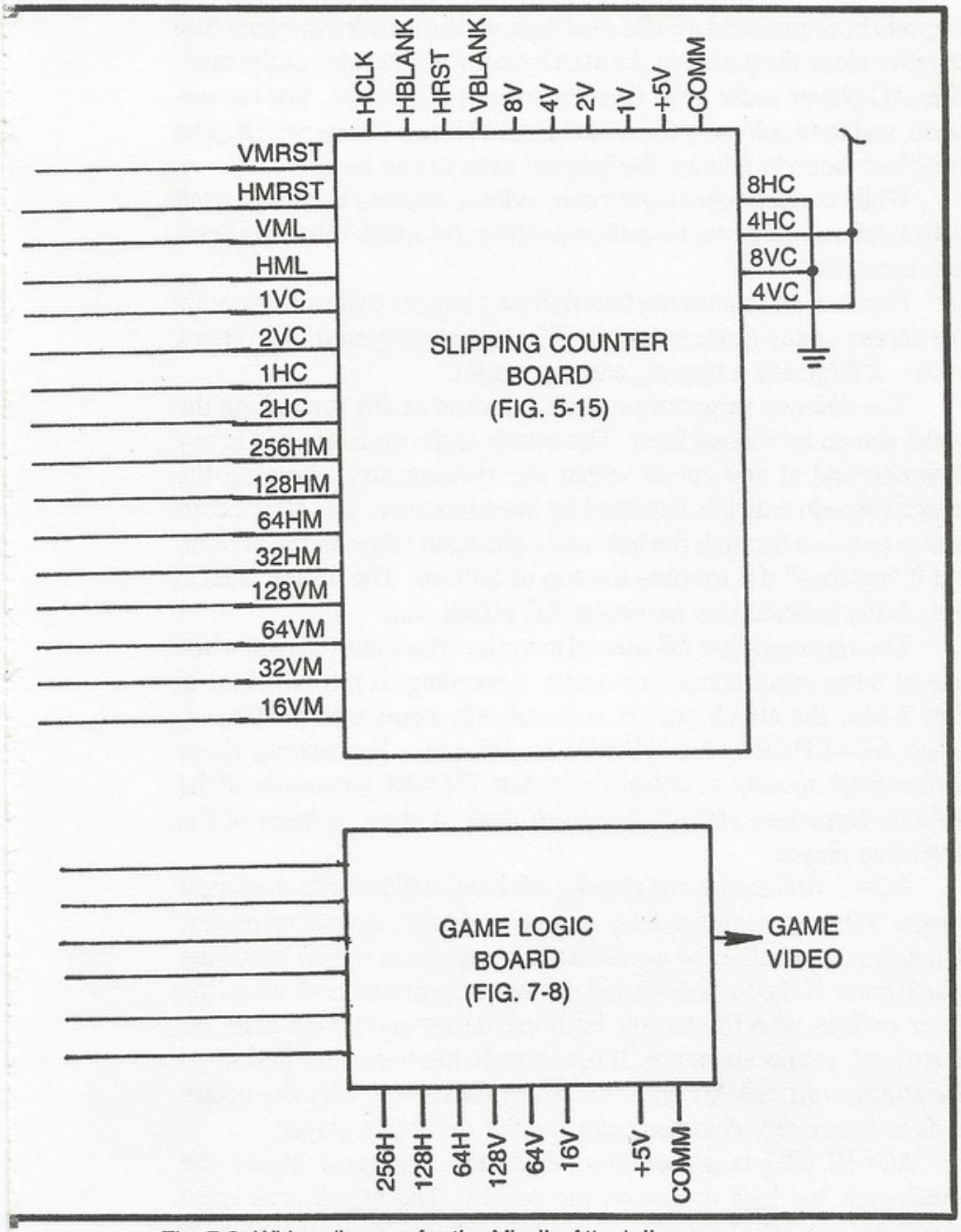


Fig. 7-9. Wiring diagram for the Missile Attack II game.

DT player can launch them one at a time in any sequence and any number of times.

The really unique feature of this Torpedo Attack game, however, is that it marks our first application of the tagalong feature described briefly in Chapter 6. The attack craft carries the attack torpedo in the tagalong fashion. When the aggressor launches his

torpedo by depressing a FIRE pushbutton, the attack torpedo is free to move along the path that the attack craft was following at the time. The AC player must head directly toward the target, fire his torpedo, and then pull away from the target before colliding with it. The launched torpedo follows the original path to the target.

While the defensive player can, indeed, destroy the attack craft with a defense torpedo, he cannot destroy the attack torpedo after it has been fired.

Figure 7-10c shows the four different images that can appear on the screen at any given moment: DT (defense torpedo), AC (attack craft), AT (attack torpedo), and the target.

The defense torpedoes can be launched at any time along the paths shown by dashed lines. The attack craft can be moved in any direction and at any speed within the viewing area, carrying the attack torpedo until it is launched by the aggressor. The attack craft is free to move through the left- and right-hand edges of the screen, but it "crashes" if it touches the top or bottom. The dotted lines in Fig. 7-10c indicate one particular AC attack run.

The aggressor has full control over the attack craft except when one of three reset conditions occur. According to the flowchart in Fig. 7-10a, the attack craft is automatically reset (and destroyed) when AC=EDGE1, AC=TARG, or DT=AC. Translating these choice-point operations into plain English, DT=AC means one of the defense torpedoes strikes the attack craft, a score in favor of the defensive player.

AC=TARG means the attack craft itself collides with the target image. This represents a draw or a score for the defensive player, depending on whether or not the attack torpedo is still on board the attack craft. If the torpedo is still on board the attack craft when the latter collides with the target, both the target and attack craft are destroyed, a draw sequence. If the torpedo has been launched when the attack craft collides with the target, however, only the attack craft is destroyed. Another point for the defensive player.

AC=EDGE1 is a condition where the aggressor steers the attack craft too high or low on the screen. The attack craft must remain between the top and bottom of the screen. Touching either of these two boundaries destroys the attack craft and represents a default score for the defensive player.

So according to Fig. 7-10a, the attack craft is flashed on and off, and reset to a starting position whenever it is hit by one of the defense torpedoes, hits the top or bottom of the screen, or collides with the target. The AC reset position, by the way, is at the left-hand

edge of the screen, about halfway between the top and bottom. The attack player has no control over the reset position and timing.

Now suppose the defensive player launches a defense torpedo. Once launched, a defense torpedo continues its rather rapid upward motion until one of two events occur: the DT strikes the attack craft (DT=AC) or the DT images reaches the top of the screen (DT=TOP). In either case, the DT image is reset immediately and the defensive player is ready to launch another one.

Figure 7-10b is the flow chart for the attack torpedo. One of two things must happen after the aggressor launches a torpedo. The torpedo can strike the target (AT=TARG), causing the target image to flash and then returning the torpedo to the attack craft. The other thing that can happen after launching the attack torpedo is that it misses the target and moves to the top, bottom, or right-hand edge of the screen. These three edges, designated EDGE2, represent the limits of the AT's travel. Hitting any one of them immediately resets the position of the AT to the attack craft. There is no score for either player.

Torpedo Attack Game Panels

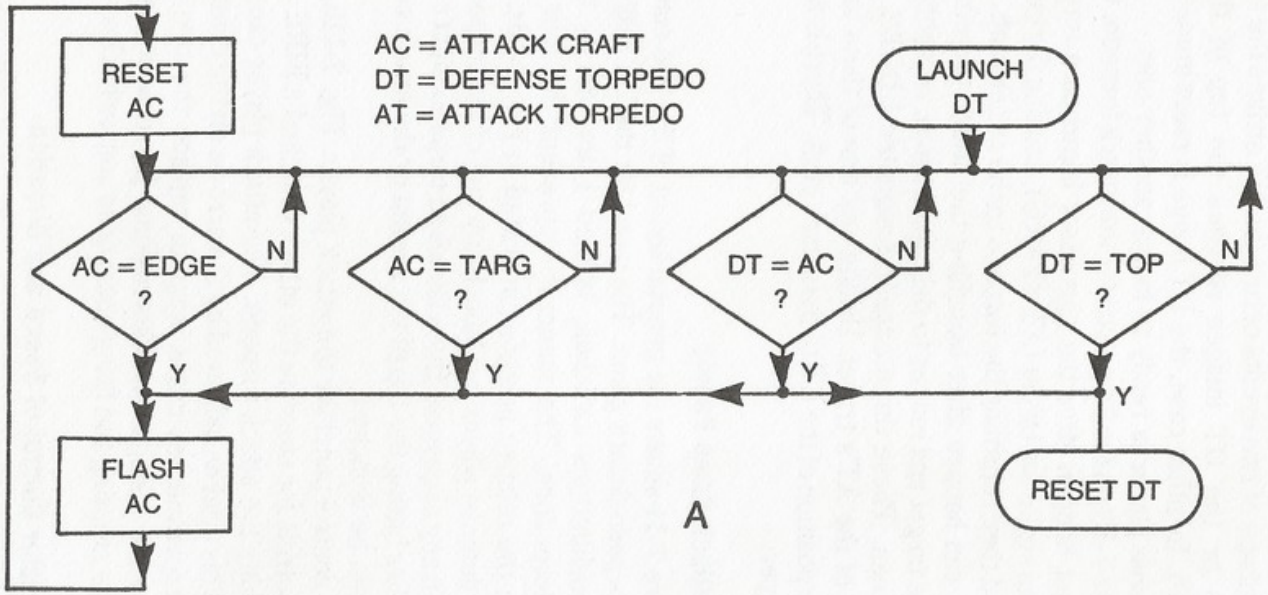
Figure 7-11 shows the general layout of the two control panels for the Torpedo Attack game. The panel for the DT player consists of four pushbuttons switches, labeled 1 through 4, that launch defense torpedoes. The numerals represent the four torpedoes located in the relative positions indicated in Fig. 7-10c.

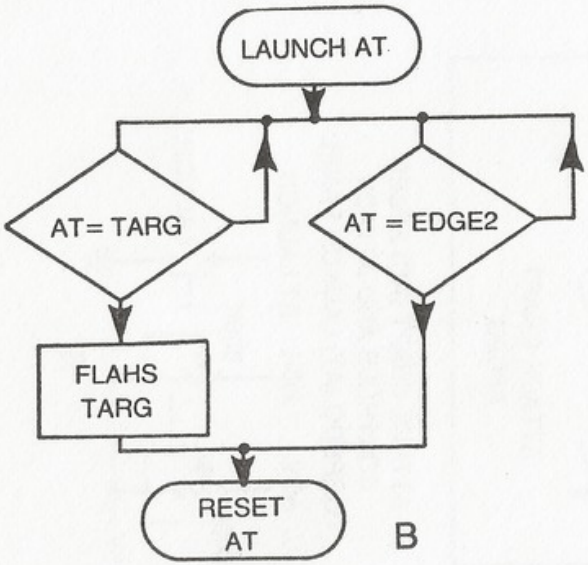
The defense player can launch any one of the torpedoes at any time and in any sequence. But a launched torpedo must complete its cycle (either hitting the top of the screen or the attack craft) before another can be launched.

The control panel for the attack player, Fig. 7-11b, includes a pair of controls for steering the attack craft and a FIRE pushbutton for launching the attack torpedo. The attack player can control the motion of his craft as indicated by the arrows on the control panel. A serious experimenter might want to replace the two AC motion controls with a single two-dimensional joystick control, thereby making the steering and firing operations somewhat less awkward.

Potentiometer Control of Speed and Direction

All of the slipping-counter speed and direction controls described thus far call for inserting a 4-bit binary number that determines the direction and speed of the apparent motion. The Missile Attack II game, for example, lets the player controlling the attack


AC = ATTACK CRAFT
DT = DEFENSE TORPEDO
AT = ATTACK TORPEDO
RESET AC
LAUNCH DT
AC = EDGE ?
AC = TARG ?
DT = AC ?
DT = TOP ?
N
N
N
N
Y
Y
Y
Y
FLASH AC
RESET DT
A



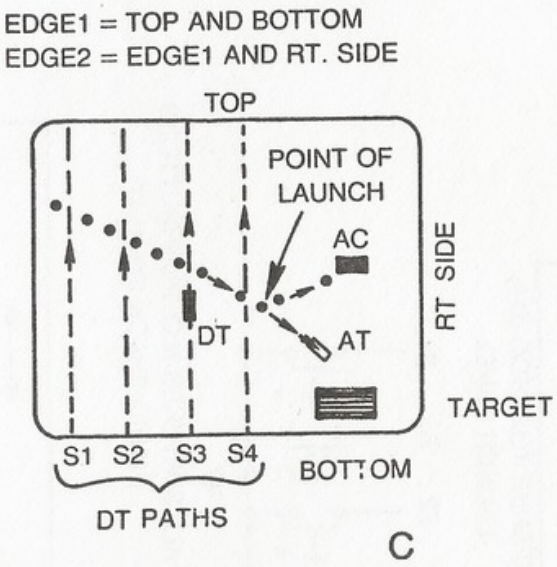


Fig. 7-10. Flowcharts and definitions for Torpedo Attack, (a) Defense torpedo and attack craft flowchart. (b) Attack craft flowchart. (c) Game images as they can appear on the screen (white and black reversed for clarity).

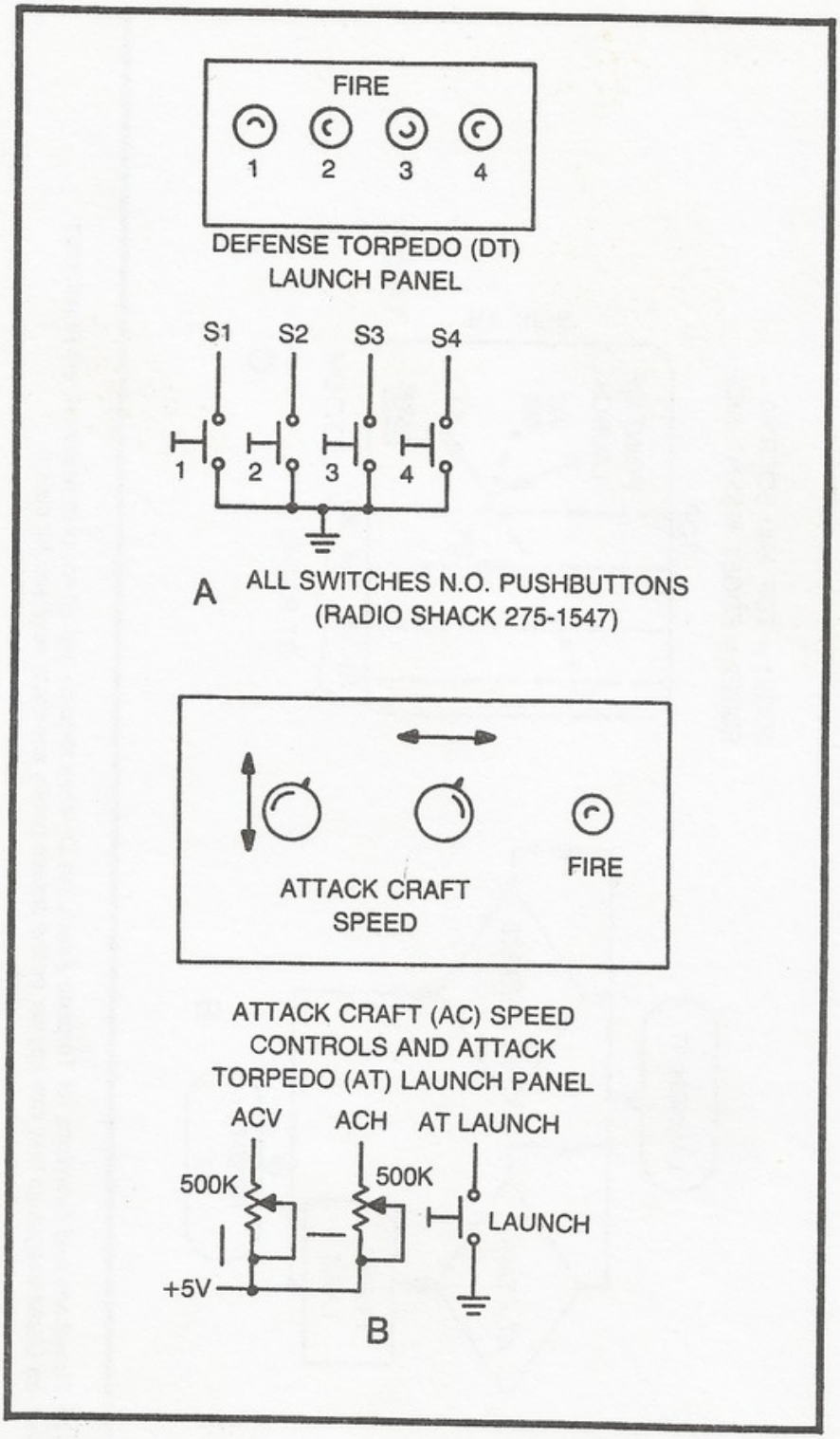


Fig. 7-11. Control panels for Torpedo Attack. (a) Defense torpedo panel and schematic. (b) Attack system panel and schematic diagram.

missile's path enter motion data via a set of switches. The control input in this case is purely digital.

Digital speed and direction controls, however, are not always the most appropriate. It is often more helpful and realistic to control the motion of a figure by means of dials—potentiometers, to be more specific.

Figure 7-12 shows a relatively simple circuit that converts the digital motion-control input into an analog format. In essence, this circuit converts the setting of a potentiometer into a 4-bit binary number for controlling the count length of a slipping counter. Using this scheme, a player controls the speed and direction of an image by turning the shaft of the variable resistor, R2.

This elementary sort of A/D converter includes a 4-bit binary counter (IC4), a monostable multivibrator having an adjustable output timing interval (IC3), a clock-pulse gating circuit (IC2-B), and a synchronizing-pulse generator (IC2-A and IC1-B). The inputs to the circuit are a 128V count pulse from the Sourcebox, a 2V count from Sourcebox, and the main control resistor, R2. The output is a 4-bit binary word that ultimately determines the speed and direction of motion of a figure on the screen. The 4-bit output is the one required for setting the count length of a slipping-counter circuit.

Two such circuits are required where the player is to have control over both the horizontal and vertical components of motion. The horizontal control includes the components shown in the main schematic in Fig. 7-12. IC5, shown in the insert in Fig. 7-12, must be added for potentiometer control of the vertical-motion component.

The waveforms in Fig. 7-12 illustrate the operation of the circuit. Whenever 128V makes a transition from logic 0 to logic 1 (near the middle of the screen), the pulse generator (IC2-A and IC1-B) generates a negative-going pulse, designated $\overline{128P}$ throughout the remainder of this book.

The monostable multivibrator is programmed such that its output timing interval is initiated on the trailing edge of the $\overline{128P}$ pulse. See the second waveform in Fig. 7-12. With the pin-3 output of the monostable circuit thus set to logic 1, 2V pulses at pin 5 of IC2-B are allowed to pass to the clocking input of the counter circuit, IC4. The counter then increments at the 2V rate until the timing interval is over.

When the timing interval is completed, the counter holds its last 4-bit output count until another $\overline{128P}$ pulse occurs. That brief pulse clears the output of the counter to zero, letting the next counting interval begin from zero.

The timing interval of the monostable multivibrator is determined by the values of R2, R3, and C3. Normally the controls are adjusted so that the count reaches any number between 5 and 15, the normal operating range for the binary numbers fed to the speed and direction control inputs of the slipping counters.

So let's suppose the player wants to stop the motion of a figure on the screen. All he has to do is adjust the value of R2 so that the counter increments to the binary equivalent of 9 (the stop code) during the monostable's timing interval. When the $\overline{128P}$ pulse occurs, then, the counter is cleared to zero, then allowed to count at the 2V rate until the timing interval is over. To stop the motion of the figure, the counting interval should be terminated with the counter showing an output of 1001 (decimal 9).

The counter then holds that number until $\overline{128P}$ occurs once again. And if the player has not changed the position of the control, the counter repeats its count-to-9 sequence.

If the player then wants to move the figure to the right at a relatively high speed, he adjusts the control so that the monostable's output timing is a bit shorter, short enough to stop the counting operation at a number such as 0101 (decimal 5).

If, on the other hand, the player wants to move the figure to the left, he adjusts R2 for a slightly longer timing interval from IC3, letting the counter run to perhaps 1101 (decimal 13).

Recall that the horizontal-slipping counter samples its speed and direction codes during the vertical-blanking interval. This circuit has its 4-bit output established before vertical blanking occurs, and it holds that number through blanking and, in fact, all the way to the end of $\overline{128V}$.

The vertical-slipping counter samples its 4-bit control input continuously. And since the output of IC4 in Fig. 7-12 spends some of its time counting, it yields some undesirable and confusing vertical-motion effects on the screen. It is thus necessary to load the output of IC4 into a temporary memory circuit when it is being used in conjunction with a vertical-slipping-counter circuit. IC5 shown in the insert in Fig. 7-12 is a quad D latch that performs this function.

The outputs of IC4 are connected to the corresponding inputs of IC5 for vertical-motion control. The latch is normally in its memory mode, keeping the four VC outputs stable through the counter's up-counting functions. The count output is then updated whenever the VRST pulse occurs.

The TRIM potentiometer in Fig. 7-12 is used for adjusting the control range of the main control potentiometer. First set the main

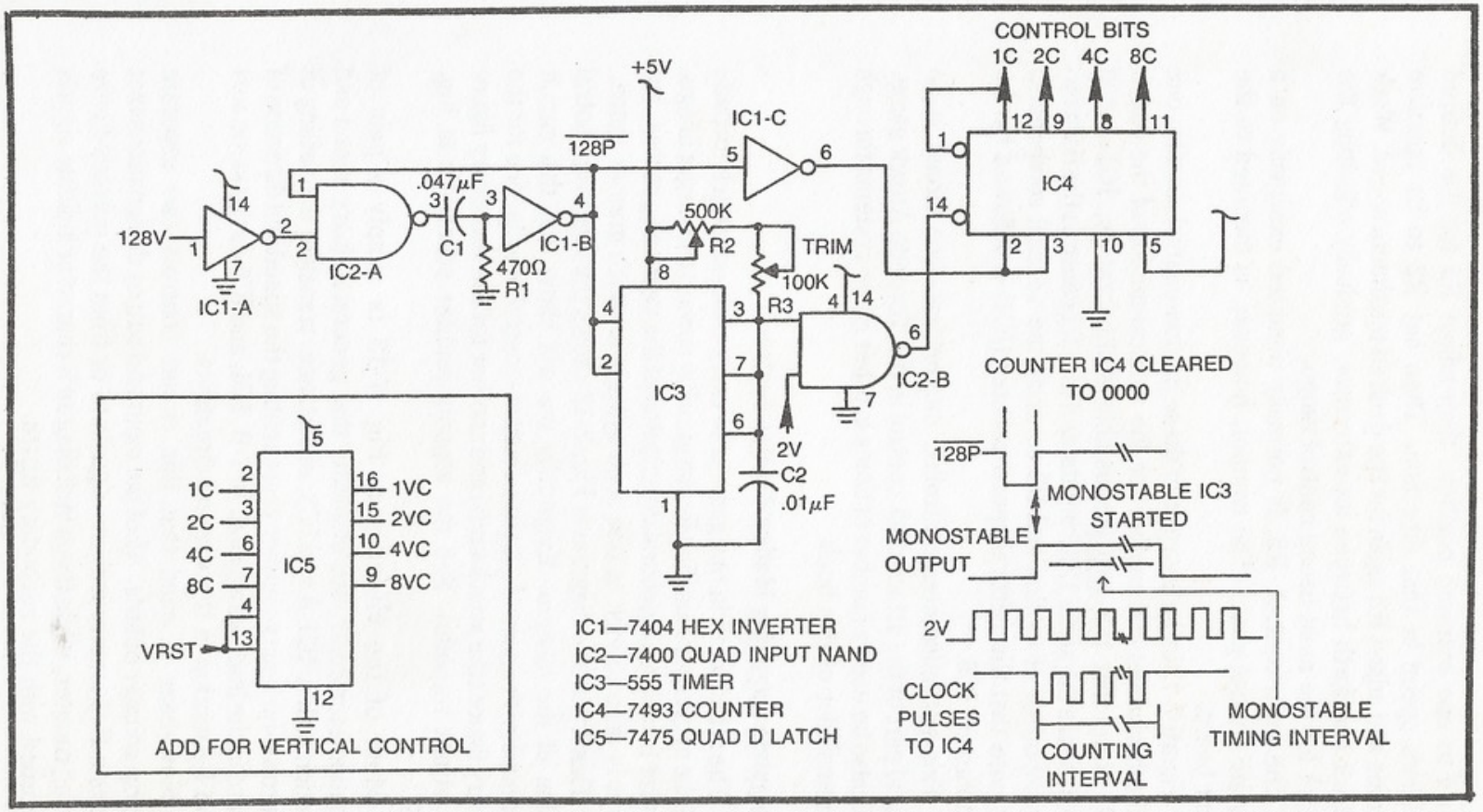


Fig. 7-12. Circuits for potentiometer control of speed and direction.

control to one extreme position, then adjust R3 for the desired maximum speed in that direction. Then set R2 to its opposite extreme and adjust R3 again for the desired maximum speed. Work R2 back and forth between its extremes, gradually adjusting the trimpot for the most useful control range.

The main control, R2, is normally mounted externally on a separate control panel. The trimpot, however, is mounted to the circuit board.

Figure 7-13 includes a pair of these motion-control circuits: one for the horizontal and another for the vertical motion of the attack craft. Note that the two monostable multivibrators, IC1-A and IC1-B, share the same $\overline{128P}$ inputs, and that R1 controls the horizontal component of motion and R3 controls the vertical component. Also note that the vertical-speed counter (IC4) is followed by the latch circuit, IC5.

This potentiometer-controlled motion interface circuit is an integral part of the attack craft control in the Torpedo Attack game. It will also be used a number of times in other game systems through the remainder of this book.

A Complete Tagalong Motion Control System

The attack craft in this game is to carry along an attack torpedo until the aggressor launches it, presumably toward the target image. Chapter 6 includes a general description of the tagalong process, but now it is time to look at one such system in much greater detail.

The schematic diagram in Fig. 7-13 shows the tagalong-control portion of the system. Eventually we will show that this circuit interfaces with a pair of slipping-counter-control boards, one for the primary figure (the attack craft) and another for the secondary figure (the attack torpedo). See the slipping-counter schematic in Fig. 5-15.

Most of the circuitry in Fig. 7-13 is simply a pair of potentiometer-controlled elements that generate binary speed and direction codes. IC1-A and IC3, of instance, translate the setting of R1 into a 4-bit binary number representing the speed and direction of a figure in the horizontal plane. IC1-B, IC4, and IC5 do the same sort of job for motion in the vertical directions.

Now bear in mind that this circuit controls two separate slipping-counter boards. All of the input and output designations that include a P character indicate signals to or from the primary-figure slipping counter, while those including an S character indicate signals concerned with the secondary figure.

Suppose the secondary figure (the attack torpedo in this case) is supposed to be taging along with the primary figure (the attack craft). In this instance, the $\overline{\text{FIRE}}$ input to IC6, IC7, and IC9 is at logic 1. This particular logic level sets memory latches IC6 and IC7 to their "read" modes—they are passing any 4-bit words at their inputs directly to their outputs. In other words, any change in the 4-bit words from IC3 and IC5 appears immediately at the outputs of IC6 and IC7 respectively.

The output designations from IC6 and IC7 imply they are 4-bit motion-control words for the horizontal- and vertical-slipping counters on the secondary-figure slipping-counter-motion board. 1SHC through 8SHC from IC6, for instance, are connected to the 1HC through 8HC control terminals of the horizontal counter, while 1SVC through 8SVC go to their respective vertical-counting-control inputs, 1VC through 8VC.

The two 4-bit motion-control words for the primary-figure slipping-counter board are taken ahead of ICs 6 and 7. These outputs are designated 1PVC through 8PVC and 1PHC through 8PHC.

As long as $\overline{\text{FIRE}}$ is at logic 1, then, both motion-control boards see the same sets of motion-control codes, and whatever changes in motion are prescribed for the primary figure are likewise delivered to the secondary figure.

The 2:1 multiplexer, IC9, also works under the control of the $\overline{\text{FIRE}}$ signal. As long as $\overline{\text{FIRE}}$=1, slipping-counter synchronizing pulses for the primary and secondary figure are identical: $\overline{\text{PHML}}$=$\overline{\text{SHML}}$, $\overline{\text{PVML}}$=$\overline{\text{SVML}}$. The primary and secondary figures thus appear at the same place on the screen.

As long as $\overline{\text{FIRE}}$=1, the player has complete control over the motion of both the primary and secondary figures by means of controls R1 and R3. And what's more, the secondary figure is always superimposed on the primary one. The latter follows the former, wherever it might go.

The picture changes completely, however, when $\overline{\text{FIRE}}$ is set to logic 0. The player still has complete control over the primary figure as before, but now latches IC6 and IC7 are set to their memory modes. The 4-bit motion-control words appearing at their outputs are fixed at the values present the moment $\overline{\text{FIRE}}$ changed from 1 to 0.

The result is that the secondary figure continues moving in the direction and at a speed specified at the moment $\overline{\text{FIRE}}$ is changed from 1 to 0. The player has no control over the motion of the secondary figure then.

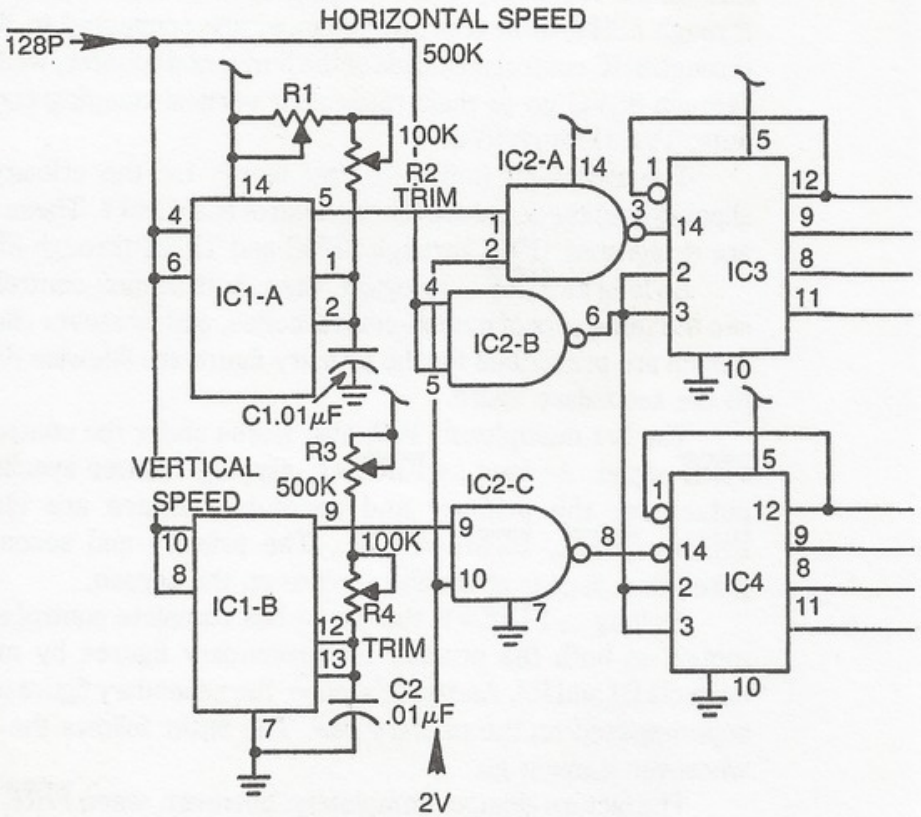
TAGALONG CONTROL BOARD

NOTE: CONTROLS R1 AND R3 ARE NORMALLY LOCATED ON A SEPARATE CONTROL PANEL (FIG. 7-10b, FOR EXAMPLE)

JUMPER WIRES = THAT CAN BE EASILY REMOVED FOR OTHER GAME APPLICATIONS

= +5V

= COMM



IC1—556 DUAL TIMER
IC2—7400 QUAD 2-INPUT NAND
IC3, 4—7493 4-BIT COUNTER
IC5, 6, 7—7475 QUAD D LATCH
IC8, 9—74157 QUAD 2:1 MULTIPLEXER

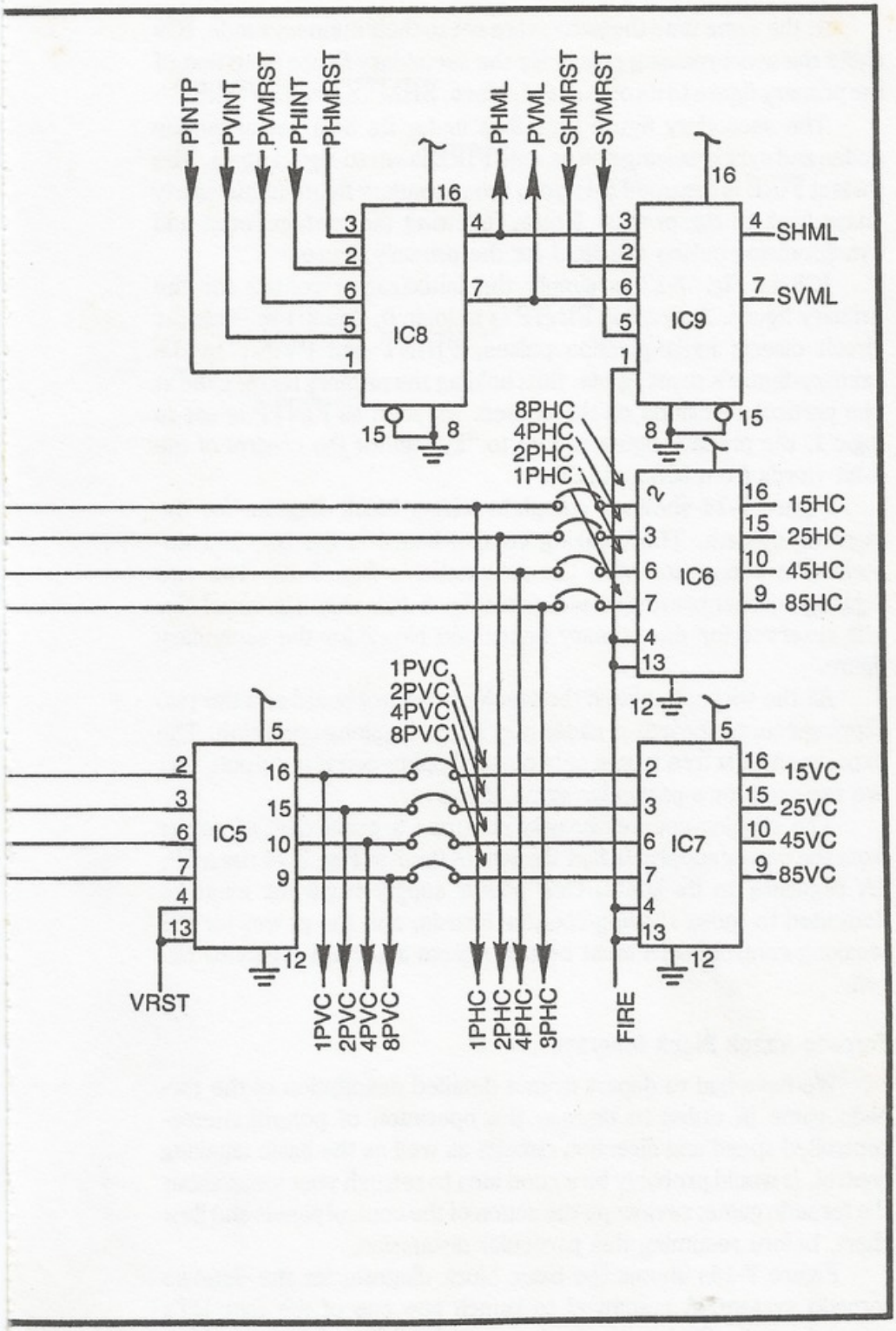


Fig. 7-13. Schematic diagram for a complete tagalong control circuit board.

At the same time the latches are set to their memory mode, IC9 shifts the synchronizing pulses for the secondary figure from that of the primary figure to its own set of pulses, $\overline{\text{SHMRST}}$ and $\overline{\text{SVMRST}}$.

The secondary figure thus flies under its own set of motion codes and synchronizing pulses until $\overline{\text{FIRE}}$ is set to logic 1 again. The instant $\overline{\text{FIRE}}$ is returned to logic 1, the secondary figure immediately snaps back to the primary figure, following the motion codes and synchronizing pulses specified for the primary figure.

IC8 in Fig. 7-13 is simply the initialization control for the primary figure. As long as $\overline{\text{PINTP}}$ is at logic 0, this 2:1 multiplexer circuit directs initial-position pulses, $\overline{\text{PHINT}}$ and $\overline{\text{PVINT}}$ to the primary-figure's reset inputs, thus holding the primary figure fixed at one particular position on the screen. As soon as $\overline{\text{PINTP}}$ is set to logic 1, the primary figure is free to "fly" under the control of the 4-bit words from IC3 and IC5.

Figure 7-14 shows a complete wiring block diagram for the tagalong system. The tagalong control board is the one just described in connection with the schematic in Fig. 7-13. The two slipping-counter boards are identical (Fig. 5-15), although board No. 1 is reserved for the primary figure and No. 2 for the secondary figure.

All the wiring between the tagalong-control board and the two slipping-counter boards is necessary in any tagalong operation. The experimenter is free to use only those slipping-counter outputs that are required for a particular game, however.

The slipping-counter boards consume a great deal of power from the power supply. In fact the two of them in Fig. 7-14 run a 5V, 1A regulator to its limits. One power supply regulator must be dedicated to these slipping-counter boards, and the power for the tagalong-control board must be taken from a second regulator circuit.

Torpedo Attack Block Diagram

We have had to depart from a detailed description of the torpedo game in order to discuss the operation of potentiometer-controlled speed and direction circuits as well as the basic tagalong control. It would probably be a good idea to refresh your ideas about the torpedo game, reviewing the action of the control panels and flow chart, before resuming this particular discussion.

Figure 7-15a shows the basic block diagram for the defense torpedo system. A command to launch any one of the four DTs comes from the DT launch panel goes to the DT horizontal position

logic circuit, which determines the DT figure to be displayed. The same firing pulse from the DT launch panel goes to a simple pulse generator and releases the initialization operation on a vertical-slipping counter. The vertical component of the selected DT figure thus begins moving upward from its initial position at the bottom of the screen. The rate of motion is internally fixed at a moderately high speed.

The DT horizontal-position-logic circuit thus determines which one of the four DTs are fired, while the DT initialization logic and vertical-slipping counter determine when and how rapidly the DT figure moves up the screen. The horizontal and vertical components of the selected DT figure are combined in the DT figure logic block to form a complete image.

The DT figure continues moving up the screen until a negative-going $\overline{\text{DTRST}}$ pulse appears at the initialization-logic block. At that moment, the DT figure is blanked from the screen and the whole DT system is re-initialized until the defensive player launches another one.

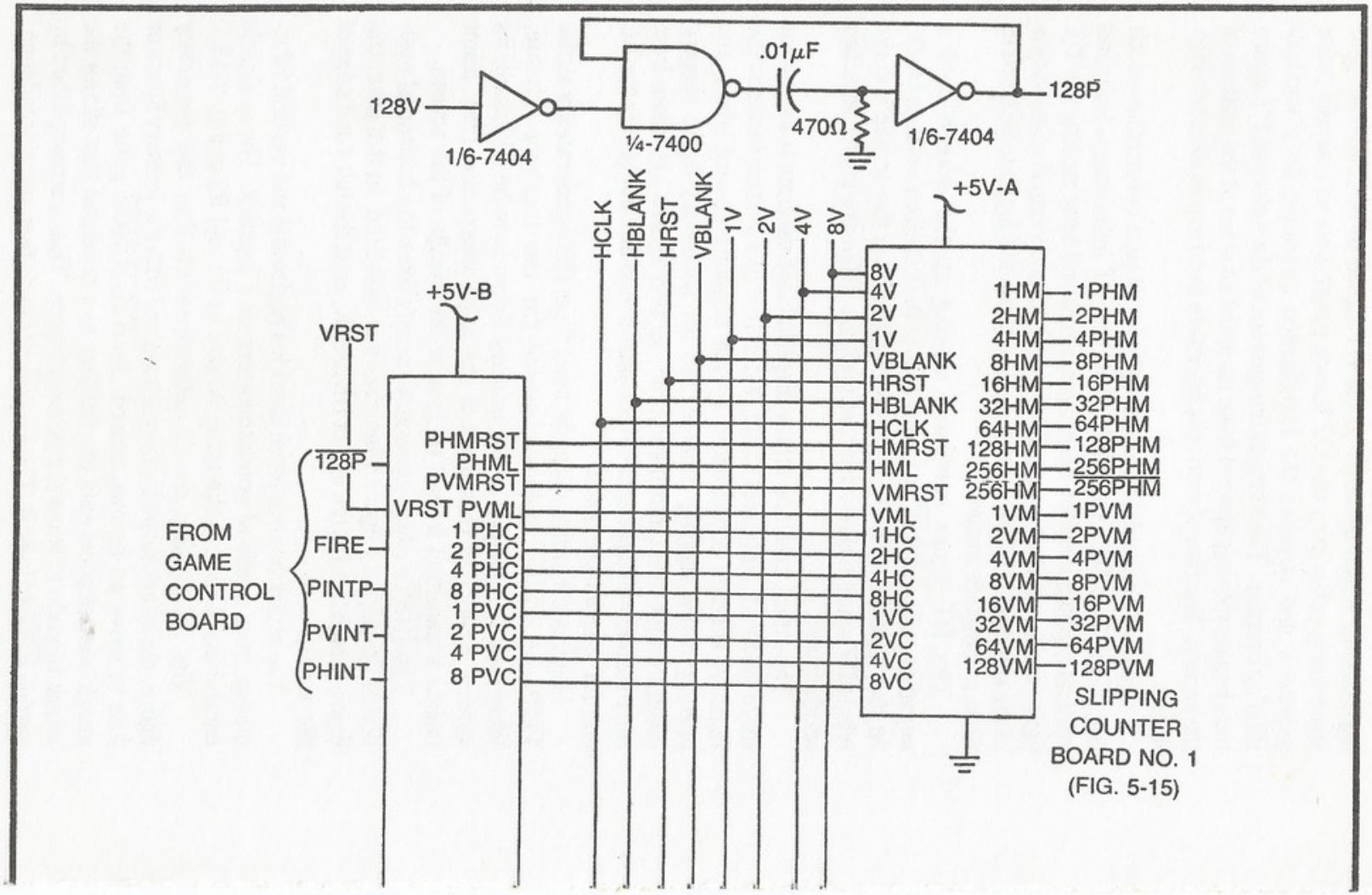
The attack craft and attack torpedo block diagram is somewhat more involved. Note first in Fig. 7-15b that this system includes the tagalong scheme represented by the tagalong control board, the primary figure slipping counter, and the secondary figure slipping counter. The video information for the two figures are taken from their respective slipping counters and formed into images by the AC and AT figure logic blocks.

A pulse generator taking its input from 128V generates both the $\overline{\text{128P}}$ and $\overline{\text{PVINT}}$ pulses required for the tagalong scheme. Whenever the position of the primary figure is to be initialized, its vertical position is thus fixed at the point where the 128V count makes a transition from 0 to 1, near the middle of the screen.

The $\overline{\text{HINTP}}$ pulse generator merely fixes the horizontal position of the primary figure whenever it is initialized. In this case, the figure is initialized at the end of HBLANK, or at the left-hand edge of the screen.

The attack control panel provides horizontal and vertical information from a pair of potenitometers or a joystick. These signals control the motion in the same fashion as R1 and R3 in Fig. 7-13.

The AT flip-flop circuit determines whether the secondary figure, the attack torpedo, is synchronized with the primary figure or free to move on its own accord. An $\overline{\text{ATLAUNCH}}$ pulse from the attack-control panel sets this flip-flop to a condition that allows the attack torpedo to leave the primary figure. This is accomplished by setting $\overline{\text{FIRE}}$ to logic 0. The secondary figure then remains indepen-


128V
1/6-7404
1/4-7400
.01μF
470Ω
1/6-7404
128P
+5V-A
+5V-B
VRST
HCLK
HBLANK
HRST
VBLANK
1V
2V
4V
8V
FROM GAME CONTROL BOARD
128P
FIRE
PINTP
PVINT
PHINT
PHMRST
PHML
PVMRST
VRST PVML
1 PHC
2 PHC
4 PHC
8 PHC
1 PVC
2 PVC
4 PVC
8 PVC
8V
4V
2V
1V
VBLANK
HRST
HBLANK
HCLK
HMRST
HML
VMRST
VML
1HC
2HC
4HC
8HC
1VC
2VC
4VC
8VC
1HM
2HM
4HM
8HM
16HM
32HM
64HM
128HM
256HM
256HM
1VM
2VM
4VM
8VM
16VM
32VM
64VM
128VM
1PHM
2PHM
4PHM
8PHM
16PHM
32PHM
64PHM
128PHM
256PHM
256PHM
1PVM
2PVM
4PVM
8PVM
16PVM
32PVM
64PVM
128PVM
SLIPPING COUNTER BOARD NO. 1 (FIG. 5-15)

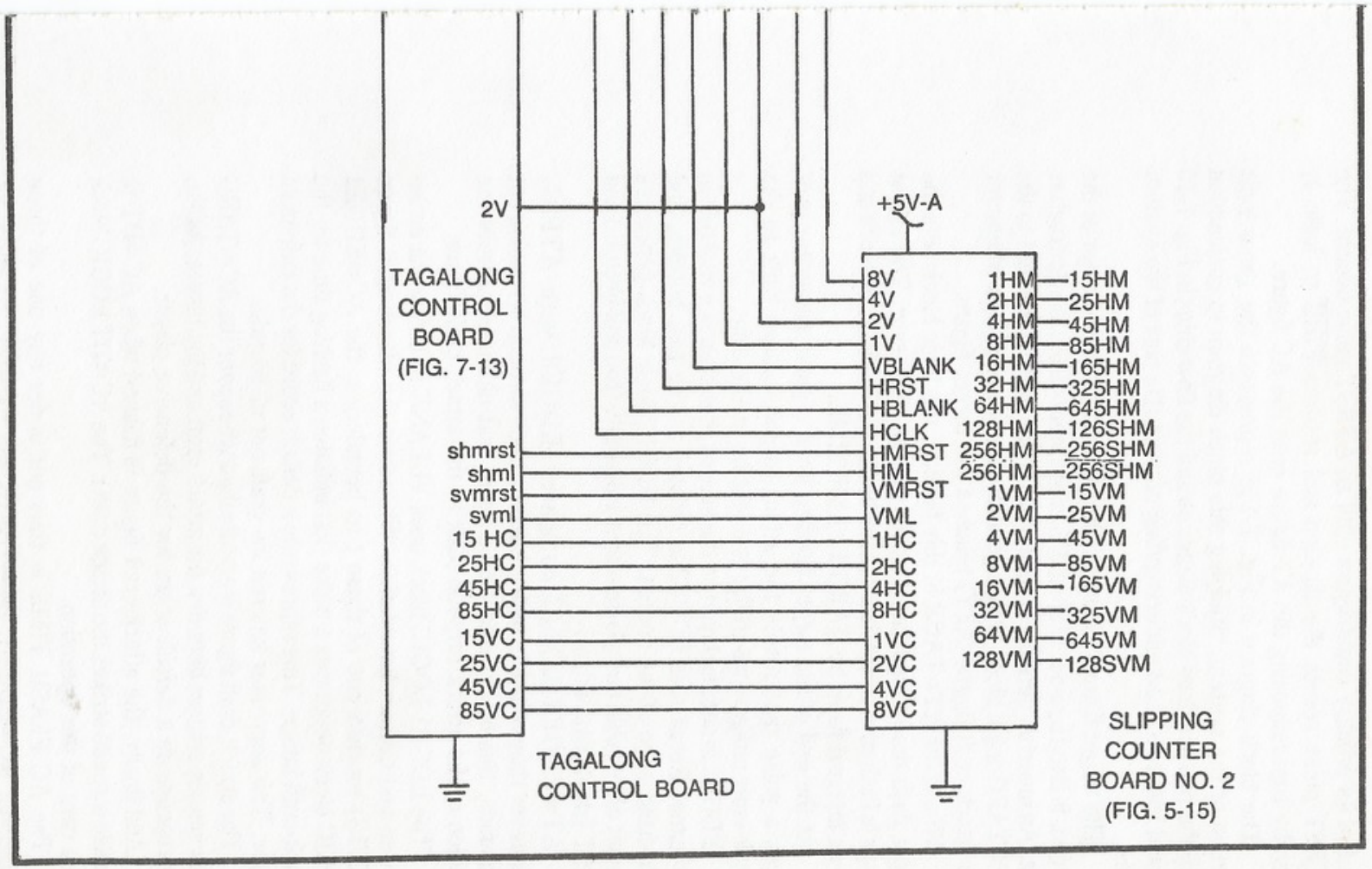


Fig. 7-14. Wiring diagram for a complete tagalong system.

dent of its primary counterpart until an $\overline{\text{ATRST}}$ pulse occurs. The ATRST pulse resets the flip-flop and returns $\overline{\text{FIRE}}$ to logic 1, thereby superimposing the AT figure onto the AC figure.

The block diagram in Fig. 7-15c represents the game logic portion of the system. Studying this block diagram in conjunction with the two previous block diagrams and the flow chart in Fig. 7-10 should lead to a good understanding of what this part of the system does.

The target figure, located in the lower right-hand corner of the screen, is fixed by a set of H- and V-count inputs from the Sourcebox unit. An inverted version of the target image is then directed to the AT=TARG logic block which generates a set of pulses whenever the attack craft figure (AFT) touches the target figure.

Whenever AT=TAGR in this fashion, the logic block sets the target flash time to indicate a score for the aggressor. The target image is flashed at a rate determined by the TARG FLASH block and for an interval fixed by TARG FLASH TIME.

At the end of the target flashing time, a pulse generator produces a pulse that resets the attack-torpedo image back to the attack-craft image—wherever it might be at the time.

EDGE2 is not displayed on the screen, but it plays a vital role in the game. Recall that EDGE2 is defined as the top, bottom, and right-hand edge of the screen. The EDGE2 logic block generates this set of invisible boundaries from VBLANK (top and bottom) and HRST (right-hand edge).

AT=EDGE2 LOGIC compares EDGE2 with ATFIG. Whenever they coincide, indicating the attack torpedo is running out of bounds, this block generates a pulse that ultimately resets the position of the attack torpedo back to the attack-craft figure.

The EDGE1 LOGIC block uses VBLANK to define the invisible top and bottom boundaries. Whenever the attack-craft figure (ACFIG) touches one of these two boundaries, the AC=EDGE1 LOGIC block generates a pulse that initiates a flashing time for the attack-craft image. This represents a default score for the defensive player. The aggressor has run his craft out of bounds.

The attack-craft figure is also flashed whenever the AC=TARG block senses contact between the attack craft and the target. Again, this represents a default score for the defensive player.

And finally, the attack-craft figure is flashed when AC=DT (a defense torpedo strikes the attack craft). The AC=DT LOGIC block takes care of this operation.

The AC FLASH TIME is thus set under any one of three conditions: AC=EDGE1, AC=TARG, and AC=DT. All three in-

stances can represent a score for the defensive player because they lead to a destruction of the attack craft.

A defense torpedo that has been fired is reset under either of two conditions: AC=DT or DT=TOP. In the first case, the defensive player has successfully stopped an attack by hitting the attack craft with a torpedo. If a defense torpedo misses its target, however, it continues its steady upward motion until it reaches the top of the screen as defined by DT=TOP LOGIC. The TOP in this case is determined by VBLANK and a TOP PULSE GENERATOR.

The four figures to be displayed (the target, attack craft, attack torpedo, and defense torpedo) are combined in the GAME VID OUTPUT LOGIC block to yield a composite game-video signal.

Torpedo Attack Schematics

The complete Torpedo Attack game system requires three circuit boards for the tagalong feature, three special game-control boards, and two player-control panels. As described earlier in this section, the tagalong feature is made up of a pair of identical slipping-counter boards (Fig. 5-15) and a tagalong-control board (Fig. 7-13). Figure 7-14 shows the complete wiring detail for these three tagalong boards.

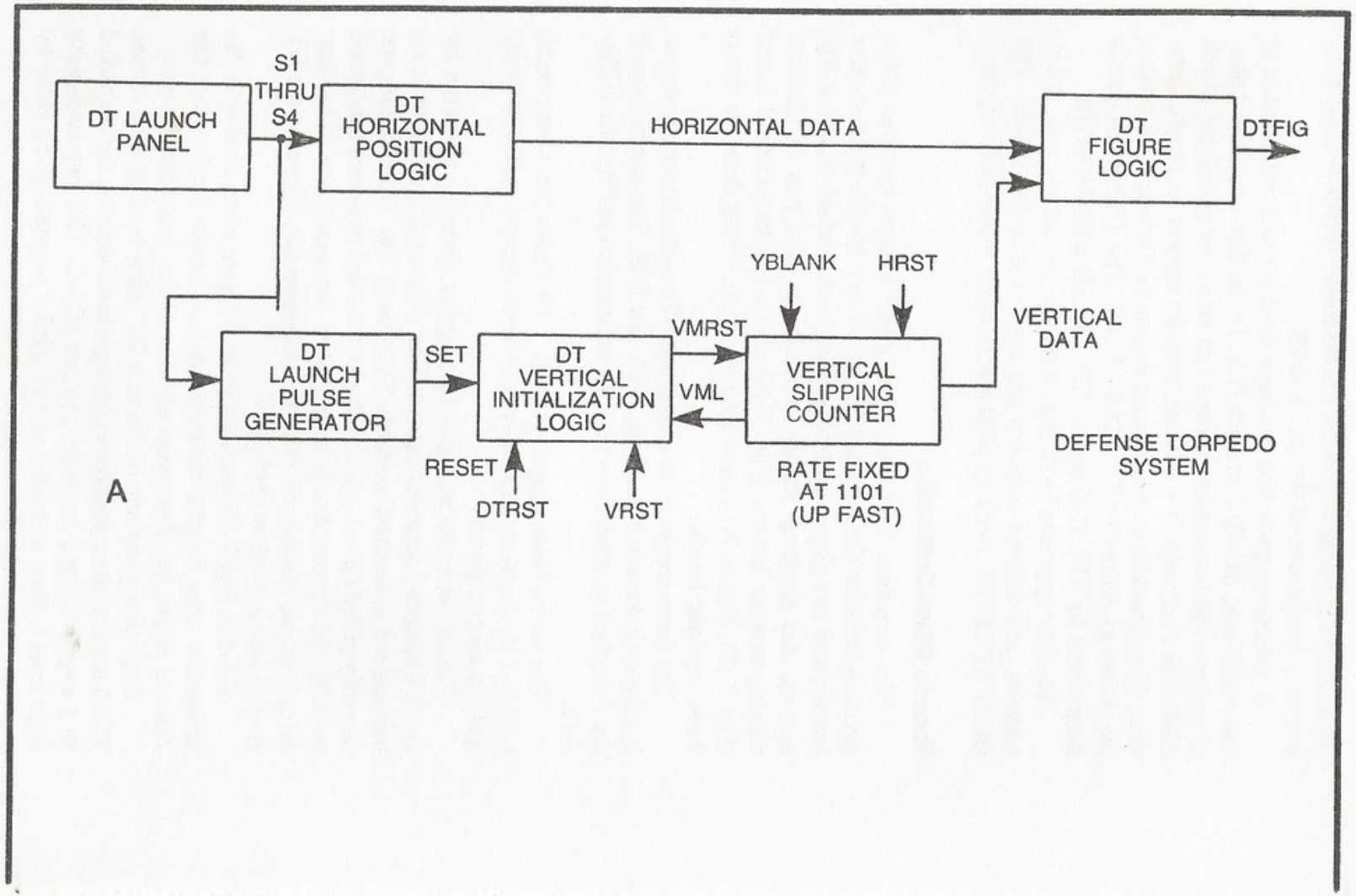
The three special control boards for Torpedo Attack are shown in schematic form in Figs. 7-16, 7-17, and 7-18. The control panels are described in connection with the schematics and layouts in Fig. 7-11.

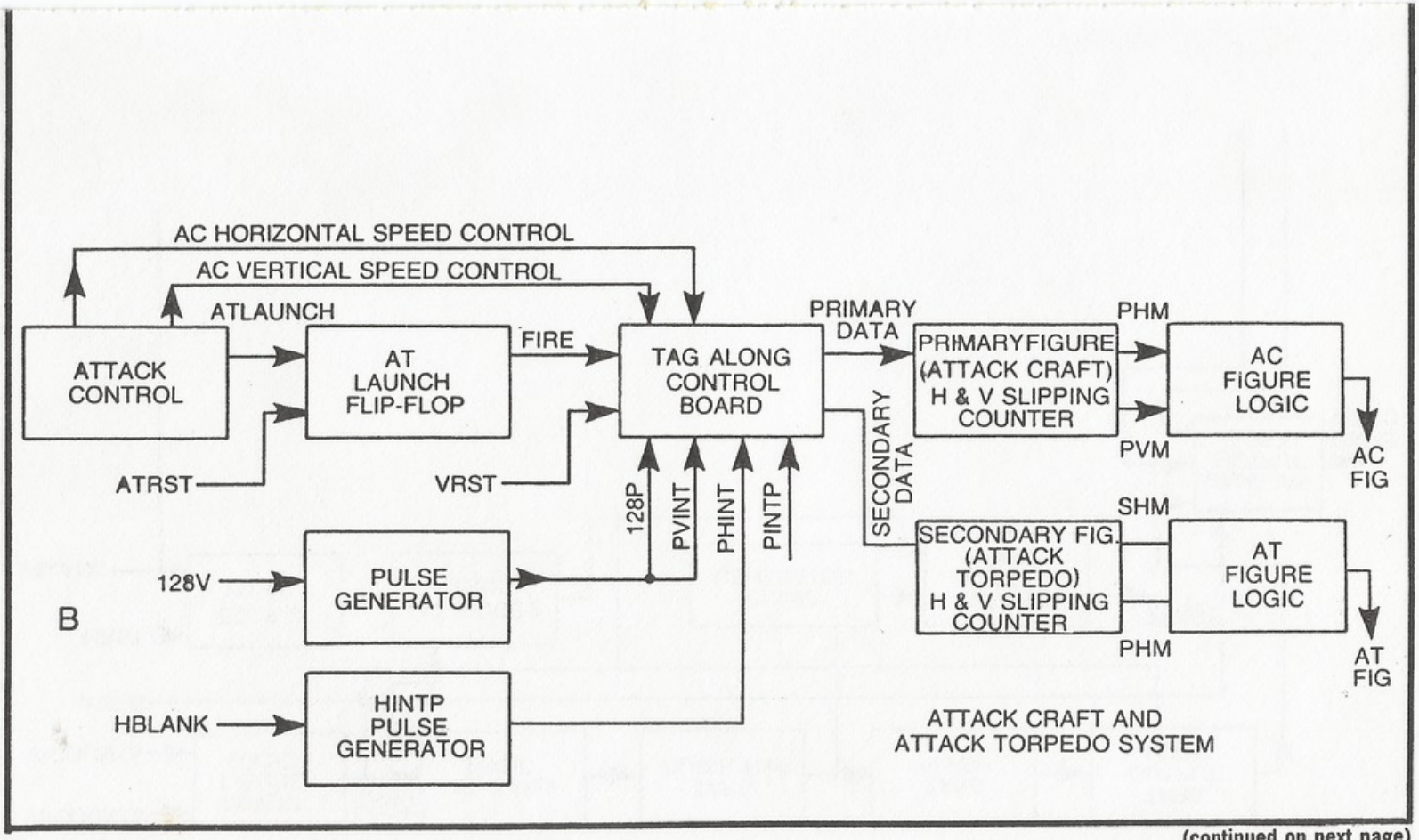
The wiring block diagram in Fig. 7-19 shows the wiring detail between the tagalong assembly, the three special control boards, and the control panels.

The circuit board in Fig. 7-16 contains all the components for the DT (defense torpedo) figures. The defense player's four firing buttons are connected to inputs S1 through S4. These inputs are normally pulled up to logic 1 by the four 2.2-kΩ resistors connected to +5V. Whenever the defense player depresses one of his four firing buttons, the logic level at the corresponding S input to the DT control board is forced to logic 0.

IC4-A in Fig. 7-16 functions as an OR gate when used with this active-low input format. Its main job is to sense the fact that the defense player has depressed any one of the four firing buttons.

Depressing any one of the four DT firing buttons thus causes IC4-A to generate a positive-going logic level which is then inverted to a negative-going level by inverter IC5-C. This negative-going logic level is then transformed into a brief negative-going pulse by

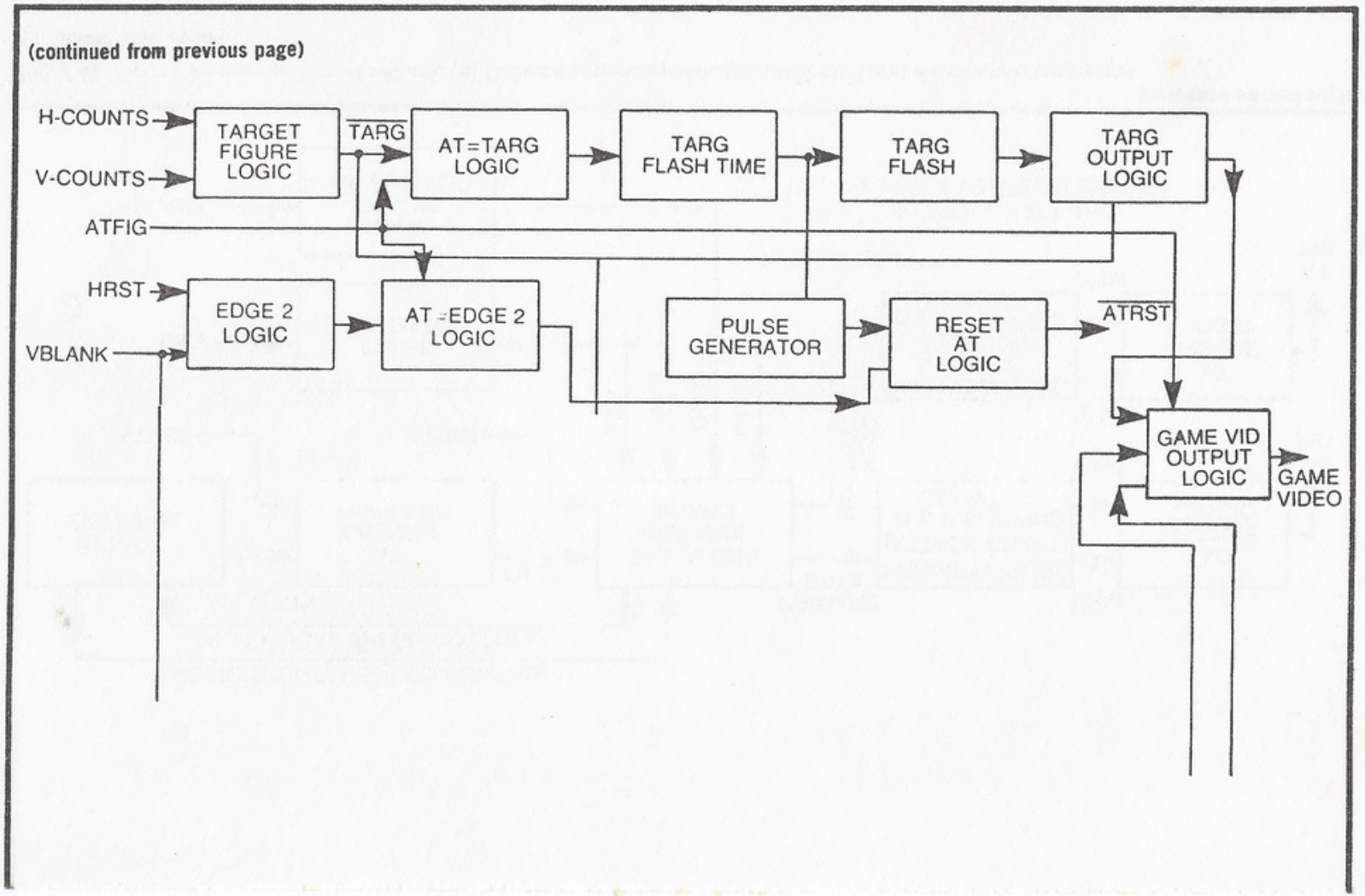

DT LAUNCH PANEL
S1 THRU S4
DT HORIZONTAL POSITION LOGIC
HORIZONTAL DATA
DT FIGURE LOGIC
DTFIG
YBLANK
HRST
VERTICAL DATA
VMRST
DT LAUNCH PULSE GENERATOR
SET
DT VERTICAL INITIALIZATION LOGIC
VML
VERTICAL SLIPPING COUNTER
DEFENSE TORPEDO SYSTEM
RESET
DTRST
VRST
RATE FIXED AT 1101 (UP FAST)
A

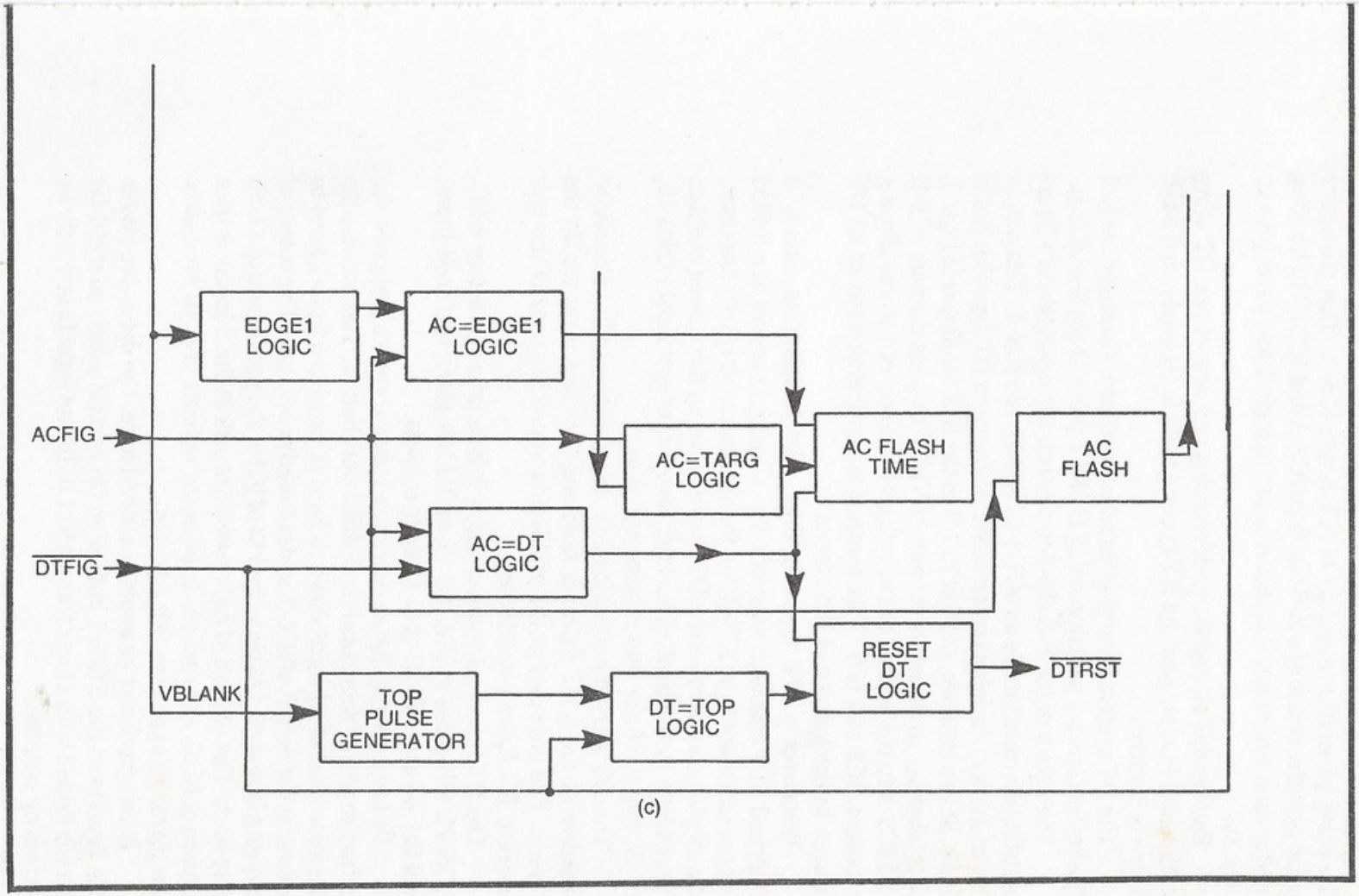


(continued on next page)

Fig. 7-15. Block diagrams for Torpedo Attack. (a) Defense torpedo system. (b) Attack craft and attack torpedo systems. (c) Game logic system.

(continued from previous page)




EDGE1
LOGIC
AC=EDGE1
LOGIC
ACFIG
AC=TARG
LOGIC
AC FLASH
TIME
AC
FLASH
AC=DT
LOGIC
DTFIG
RESET
DT
LOGIC
DTRST
VBLANK
TOP
PULSE
GENERATOR
DT=TOP
LOGIC

(c)

the pulse generator, made up of IC1-A and IC5-A. That particular pulse sets the status of a $\overline{R}$-$\overline{S}$ flip-flop (IC2-A and IC2-B) to its firing mode—one where pin 3 of IC2-A is set to logic 1 and pin 6 goes to logic 0.

Putting this all together, depressing any one of the DT firing switches ultimately sets the DT system to its firing mode, and a DT figure is launched.

The DT system is reset to its initial condition whenever the $\overline{R}$-$\overline{S}$ flip-flop receives a negative-going DTRST pulse at pin 5 of IC2-B.

The output of the $\overline{R}$-$\overline{S}$ flip-flop controls the operation of a figure initialization circuit made up of IC2-C, IC2-D, and IC3-A. This part of the circuit is in the initialization mode whenever a DT figures is not in flight. In this instance, pin 6 of IC2-B in the $\overline{R}$-$\overline{S}$ flip-flop is at logic 1, thus allowing an inverted version of VRST to pass through IC2-D and IC3-A to the loading or reset inputs of a pair of vertical-slipping counters, IC8 and IC9. This forces the vertical positon of all DT figures to the bottom of the screen.

Whenever a DT figure is fired, the initialization circuit is switched to a condition whereby the slipping counters are loaded with reset pulses from IC3-B. Under this particular set of conditions, the slipping counters let the DT figure move up the screen at a rate determined by the hard-wired logic levels at preset inputs (pins 15, 1, 10, and 9) of the two slipping counters.

The DT circuitry described to this point merely determined whether or not a DT figure is fired and how fast it moves up the screen. The next part of the problem is to see how one of the four possible DT figures is selected.

The DT figure selection circuitry is built around the four EXCLUSIVE OR gates in IC6, the three NAND gates that feed them, and the 4-input NAND gate at their outputs.

To see how this figure-selection scheme works, suppose the defensive player depressed the FIRE 1 pushbutton. This action pulls the logic level of S1 input down to logic 0, causing a logic-1 level to appear at the output of IC4-A as described previously. The output of this gate is returned to pins 4 and 13 of IC10, the gate inputs of a 4-bit data latch. The pattern of logic levels present at the S inputs is thus latched in IC10, and remains fixed at the outputs of IC10 even after the player releases the firing button.

In this particular example described here, the defensive player has depressed the FIRE 1 button on his control panel, so the 4-bit latch is loaded with a logic 0 at pin 9 of IC10 and logic 1s at the three remaining outputs.

NAND gates IC1-C and IC1-D now see logic-0 levels at one of their two inputs, thereby guaranteeing logic-1 levels from those two gates. The output of IC1-B remains at logic 0 because its two inputs from the latch are still fixed at logic 1.

The EXCLUSIVE OR gates in IC6 then produce a pattern of inverted and noninverted H-count signals. IC6-B and IC6-C both yield inverted versions of their respective H-count inputs, while IC6-A produces a noninverted version of 128H. Since IC6-D is permanently wired to invert its 32H input, it follows that the EXCLUSIVE OR configuration is producing H-count signals of $\overline{256H}$, 128H, $\overline{64H}$, and $\overline{32H}$. NANDing these four signals together in IC4-B produces a DT image having its horizontal position determined by the four H-count parameters just described. That's true if the player depresses the FIRE 1 button.

The following list summarizes the H-count parameters from this DT selection circuit whenever the defensive player fires any one of the four defense torpedoes:

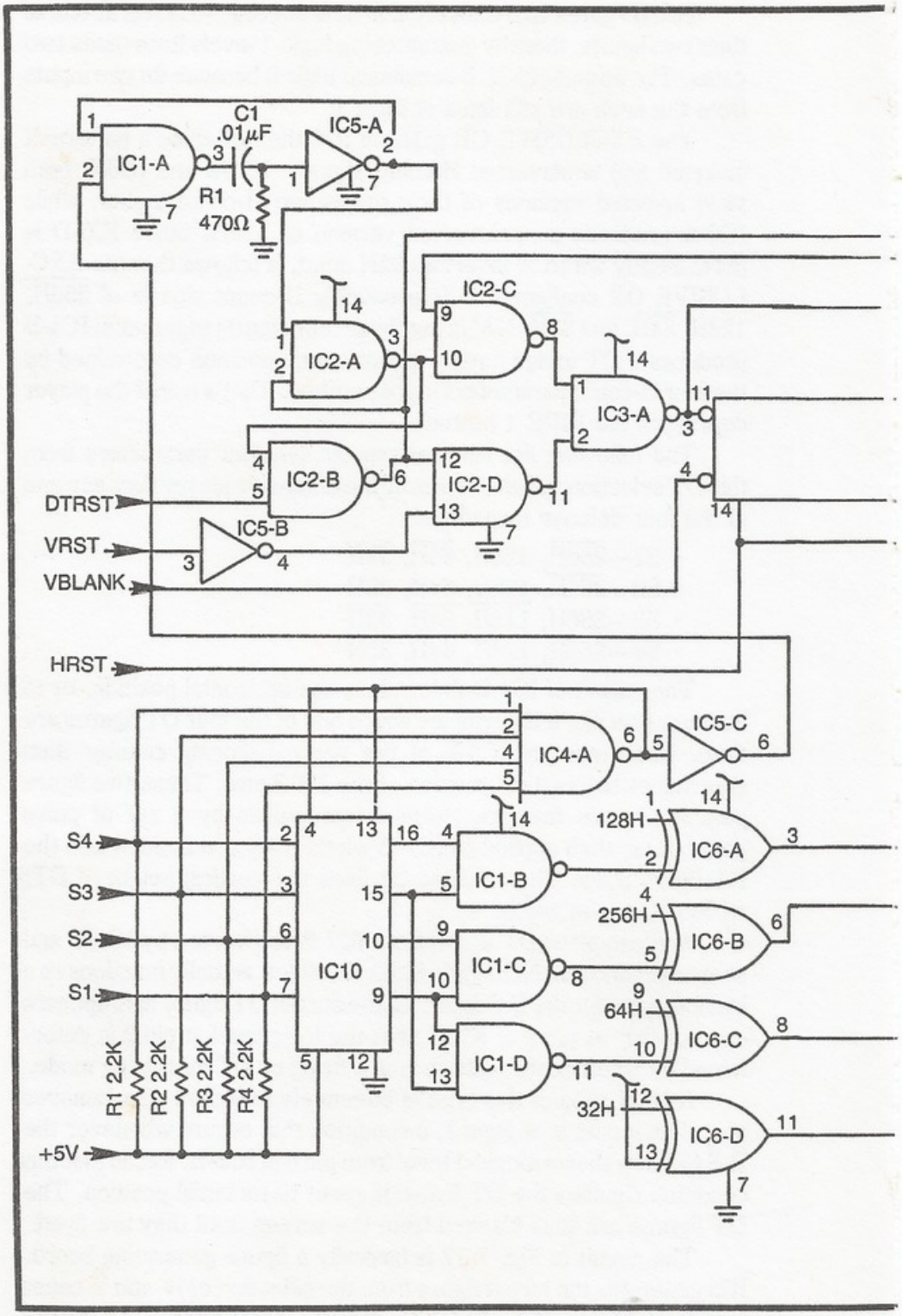
- S1—$\overline{256H}$, 128H, $\overline{64H}$, $\overline{32H}$
- S2—$\overline{256H}$, $\overline{128H}$, $\overline{64H}$, 32H
- S3—256H, $\overline{128H}$, $\overline{64H}$, $\overline{32H}$
- S4—256H, $\overline{128H}$, 64H, $\overline{32H}$

The output of IC4-B determines the horizontal position, or to be more specific, it determines which one of the four DT figures are fired. The output of IC3-B at the vertical-slipping counter then determines the vertical position of the DT figure. These two figure parameters are first transformed into pulses by a set of pulse generators, then applied to IC7-B where they are molded into the DT figure itself. The value of C2 fixes the vertical height of DT, while C3 sets in width.

The complete DT figure from IC7-B is inverted by IC7-C and sent to yet another NOR gate which, in effect, actually functions as a blanking gate for the DT figure. Some sort of DT figure data appears continuously at pin 3 of IC7-A, but the logic level at pin 2 is determined by whether the system is in a firing or DT initializing mode.

A NOR gate of this type is effectively switched off whenever one of its inputs is at logic 1, a condition that occurs whenever the $\overline{R}$-$\overline{S}$ flip-flop shows a logic-1 level from pin 6 of IC2-B. Recall that this condition signifies the DT figure is reset to its initial position. The DT figures are thus blanked from the screen until they are fired.

The circuit in Fig. 7-17 is basically a figure generating board. IC8 generates the target figure from the selection of H- and V-count signals shown at its inputs. The ACFIG signal from IC3-A is the


C1
.01μF
IC1-A
IC5-A
R1
470Ω
IC2-C
IC2-A
IC2-B
IC2-D
IC3-A
IC5-B
DTRST
VRST
VBLANK
HRST
IC4-A
IC5-C
S4
S3
S2
S1
IC10
IC1-B
IC1-C
IC1-D
128H
256H
64H
32H
IC6-A
IC6-B
IC6-C
IC6-D
R1 2.2K
R2 2.2K
R3 2.2K
R4 2.2K
+5V

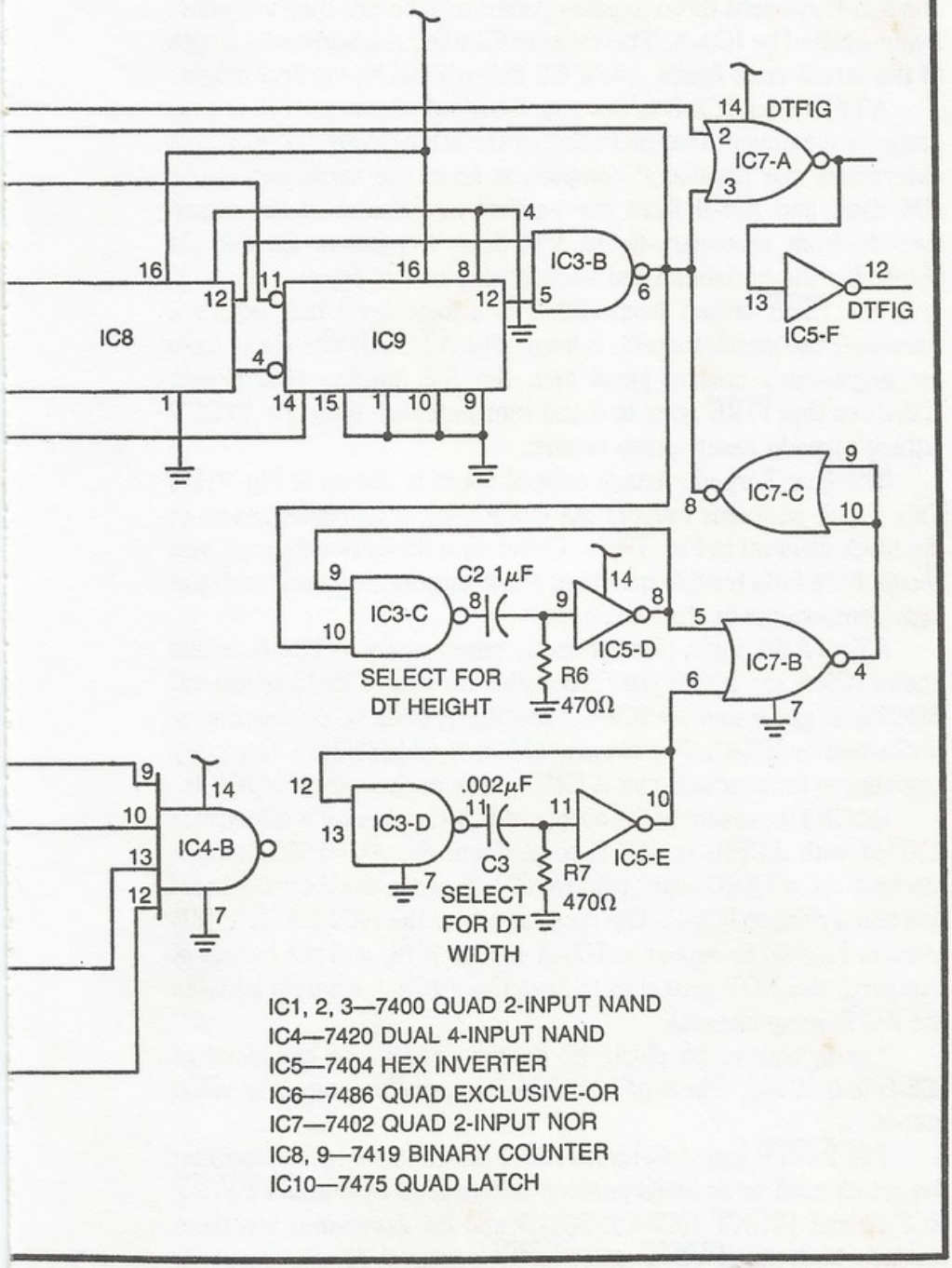


Fig. 7-16. Schematic diagram for the defense torpedo circuit board.

composite image for the attack craft. The horizontal component of this movable figure is determined by the primary-figure HM inputs to IC5-A. And in a similar fashion, the vertical component comes from the primary-figure VM inputs to IC5-B. The signals from IC5-A and IC5-B are sent through pulse generators before they are effectively ANDed by IC3-A. The value of C1 fixes the horizontal length of the attack-craft figure, while C2 determines its vertical height.

ATFIG from IC3-B is the attack-torpedo figure, which is generated in a manner identical to that of the attack-craft figure. IC6-A determines the horizontal component from the secondary-figure HM data, and IC6-B fixes the vertical component of the attack torpedo from secondary-figure VM data. Capacitors C3 and C4 determine the horizontal and vertical size of this figure.

The $\overline{\text{FIRE}}$ output from IC2-C is a logic level that equals 0 whenever the attack torpedo is fired. The $\overline{\text{ATLAUNCH}}$ signal from the aggressor's control panel sets the $\overline{\text{R}}$-$\overline{\text{S}}$ flip-flop (IC2-B and IC2-C) so that $\overline{\text{FIRE}}$ goes to 0 and remains there until the ATRST (attack torpedo reset) pulse occurs.

The final Torpedo Attack control board is shown in Fig. 7-18. This circuit performs most of the game-control functions shown in the block diagram in Fig. 7-15c. Comparing these two figures, you should have little trouble matching block diagram functions with the logic components in the schematic.

AT=TARG logic, for instance, takes place at IC1-B, while timers IC9-A and IC9-B are responsible for target-flashing effects. EDGE2 is generated by IC4-D, and the AT=EDGE2 function is performed by IC5-C. The outputs of IC9-A or IC5-C are ultimately responsible for producing an ATRST pulse at the output of IC7-B.

EDGE1 is present at the output of IC6-C, where it is effectively ANDed with ACFIG in IC3-D to perform the AC=EDGE1 logic function. AC=TARG takes place at IC4-A, while the AC=DT function takes place at IC3-C. The three inputs to the AC FLASH TIME block in Fig. 7-15c appear at IC5-A in Fig. 7-18, and the output of that particular NOR gate goes to flash timer IC8-A where it initiates the AC flashing interval.

The figures to be displayed on the screen are combined at IC5-B and IC1-C. Pin 8 of IC1-C is the game's composite video output.

The $\overline{\text{PINTP}}$ logic level from IC6-E is responsible for resetting the attack craft to an initial position determined by outputs $\overline{\text{PVINT}}$ (IC7-F) and $\overline{\text{PHINT}}$ (IC7-C). IC1-D and its associated inverters produce both the $\overline{\text{PVINT}}$ pulse $\overline{\text{128P}}$ required for the tagalong system.

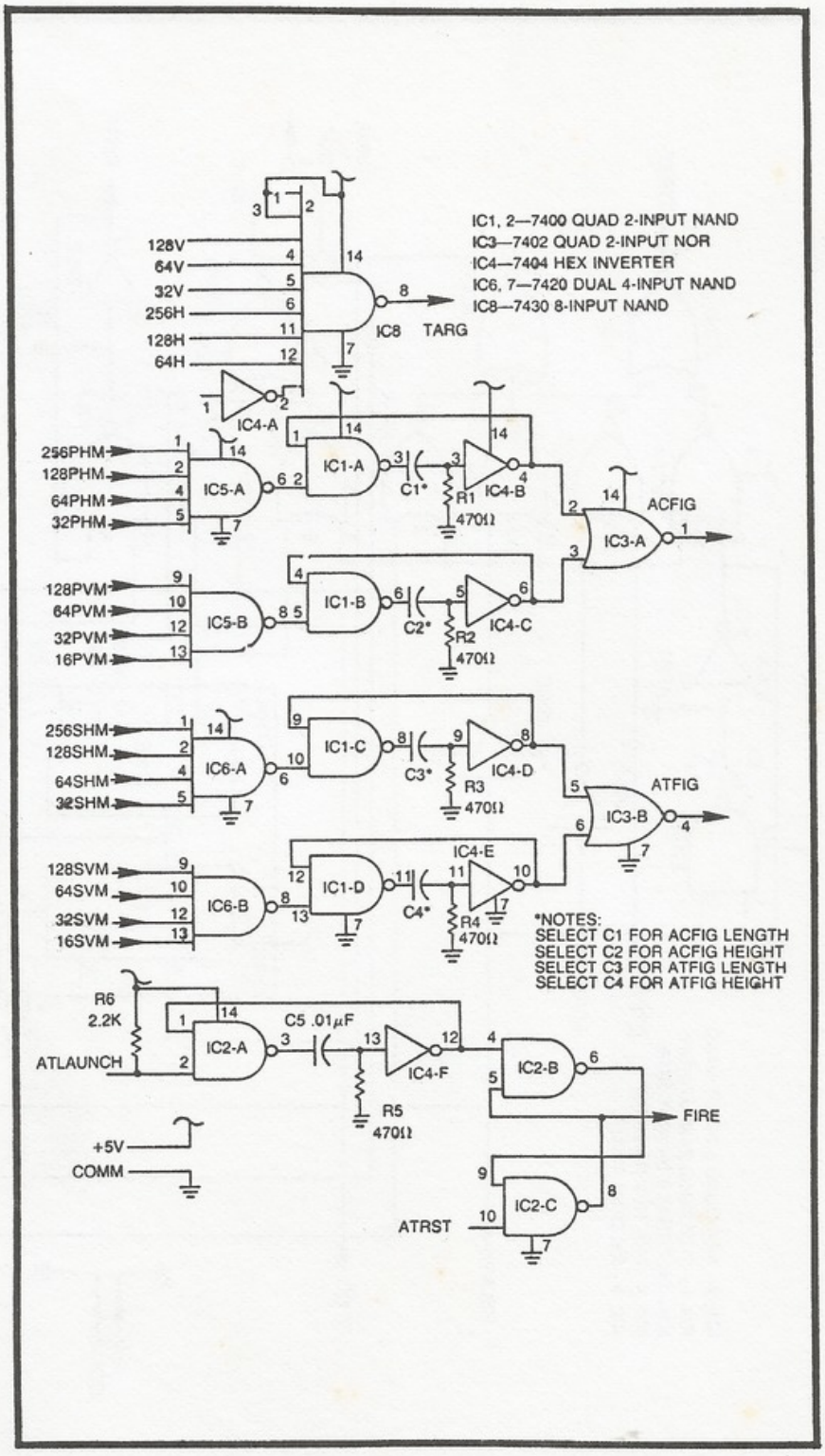
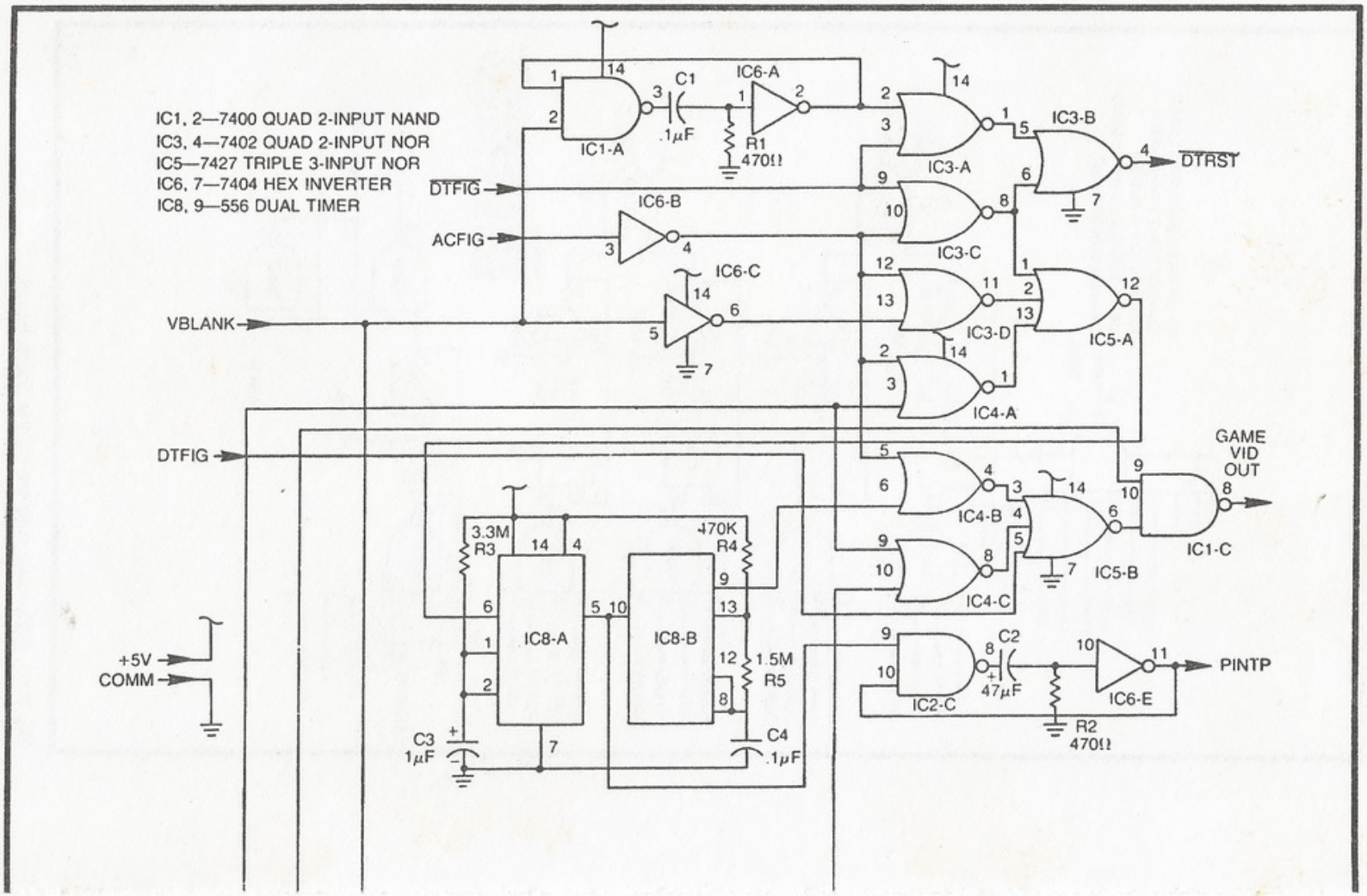


Fig. 7-17. Schematic diagram for the figure board.


IC1, 2—7400 QUAD 2-INPUT NAND
IC3, 4—7402 QUAD 2-INPUT NOR
IC5—7427 TRIPLE 3-INPUT NOR
IC6, 7—7404 HEX INVERTER
IC8, 9—556 DUAL TIMER
IC1-A
IC6-A
C1
.1μF
R1
470Ω
IC3-A
IC3-B
DTRST
DTFIG
IC6-B
ACFIG
IC3-C
IC6-C
VBLANK
IC3-D
IC5-A
IC4-A
DTFIG
GAME VID OUT
IC4-B
IC1-C
IC5-B
IC4-C
3.3M R3
470K R4
IC8-A
IC8-B
1.5M R5
+5V
COMM
C3 1μF
C4 .1μF
IC2-C
C2 47μF
IC6-E
PINTP
R2 470Ω

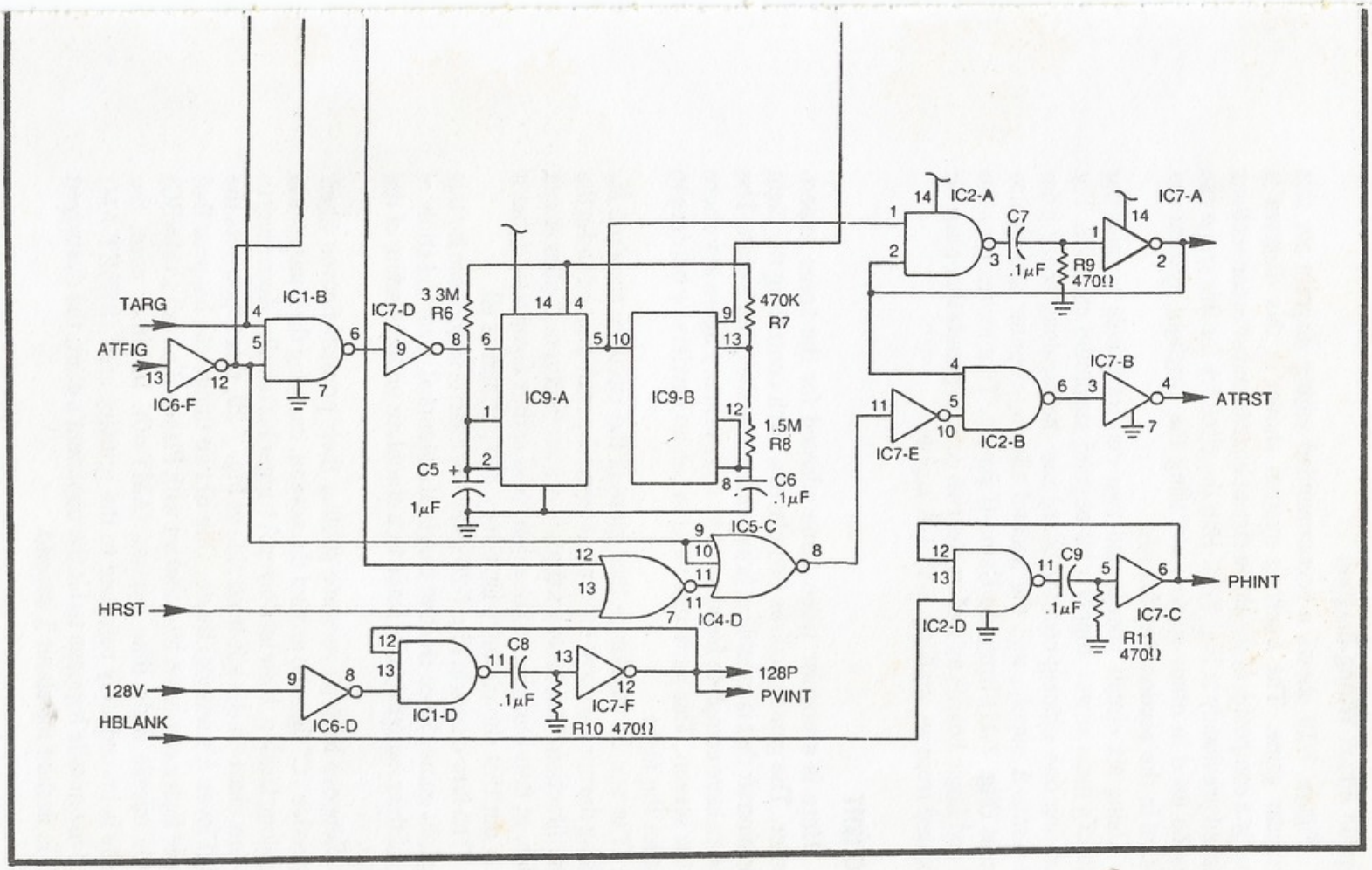


Fig. 7-18. Schematic diagram for the logic board.

Torpedo Attack Wiring Diagram

Figure 7-19 shows a recommended wiring diagram for this particular game. The tagalong system shown in this diagram is actually a composite of the three circuit boards and interconnections detailed previously in Fig. 7-14. Half the circuitry for this game can thus be used in other games, including the Dogfight system described in the section that follows.

Using six circuit boards and two control panels pushes the system's main power supply a bit beyond maximum capacity. Try operating one slipping-counter board and the tagalong board from the Game-A supply, and the second slipping-counter and Figure Boards (Fig. 7-17) from the Game-B supply. That leaves only the DT and Logic boards as well as the two control panels that must be operated from an auxiliary 5-V, 1-A supply.

DOGFIGHT

Here is a popular video game adapted for the home experimenter. The game requires two players, each controlling the flight of an aircraft that is capable of firing a missile at the other's craft. The players have complete freedom to fly their primary figure anywhere on the screen. This particular version has no barriers or borders to restrict the flight.

The special wrinkle in this game is the circuitry required for making the missile leave the craft at twice the craft's speed and in the same direction. This involves the use of a 2× vector multiplier circuit which, at first thought, might seem to be rather complicated, but it turns out that the circuitry isn't very complicated at all.

The flow charts in Fig. 7-20 show the control sequences for this Dogfight game. Since the two charts are identical, a careful study of one of them automatically leads to a complete understanding of the other.

Suppose both players are piloting their primary figures (their respective "C" figures) around the screen, carrying their missiles in a tagalong fashion. Now let Player A trigger his missile. According to the flow chart on the left-hand side of Fig. 7-20, this action sets the AM (Player A's missile) mode. One of two things can happen: that missile figure can come into contact with Player B's craft (AM=BC) or A's missile flight time expires (AMT=0). In either case, the missile is immediately returned to the primary craft (RESET AM) but if the missile happens to hit the opponent's craft, the destroyed craft is flashed for about 1 second.

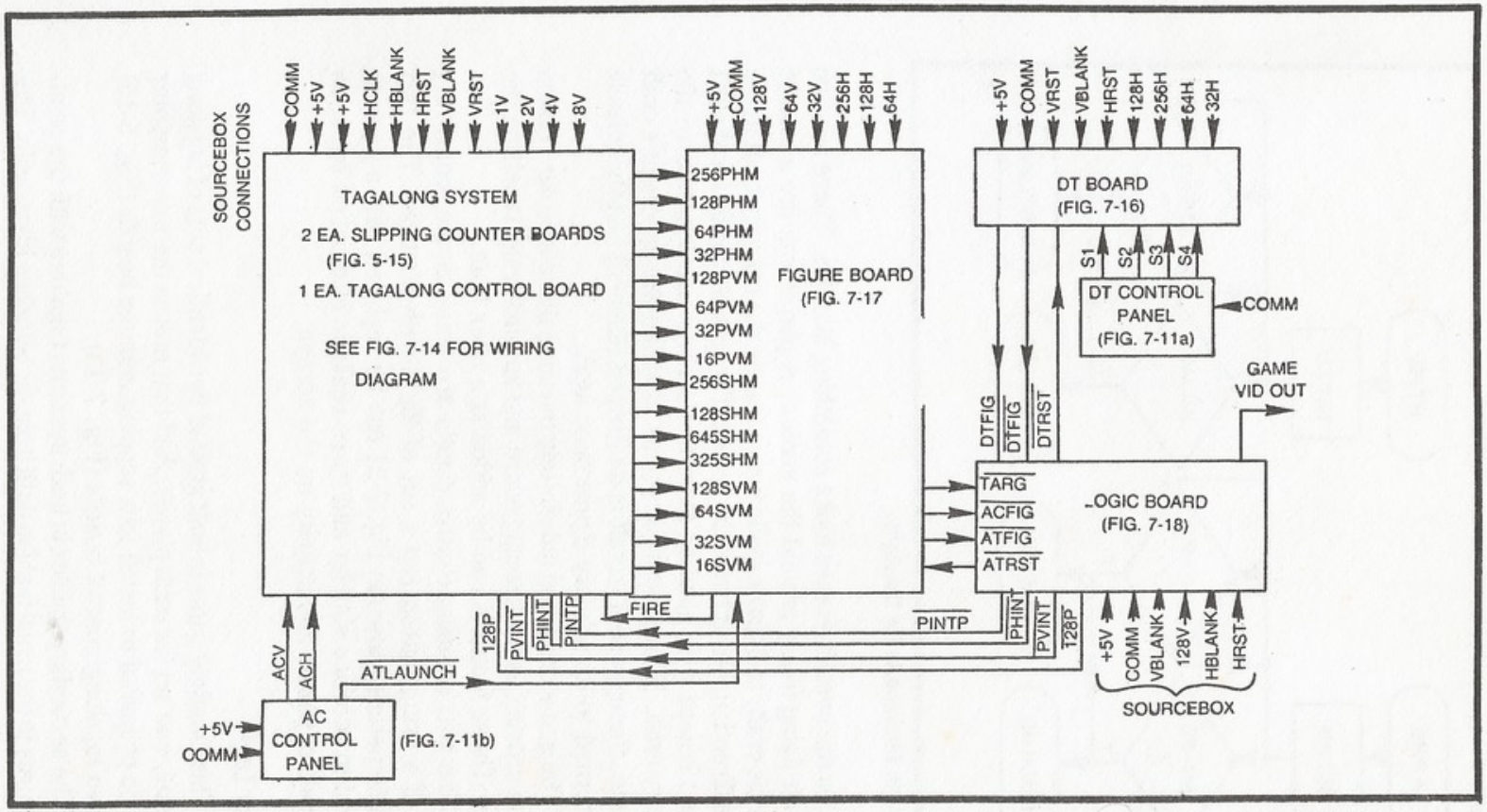


Fig. 7-19. Wiring diagram for Torpedo Attack.

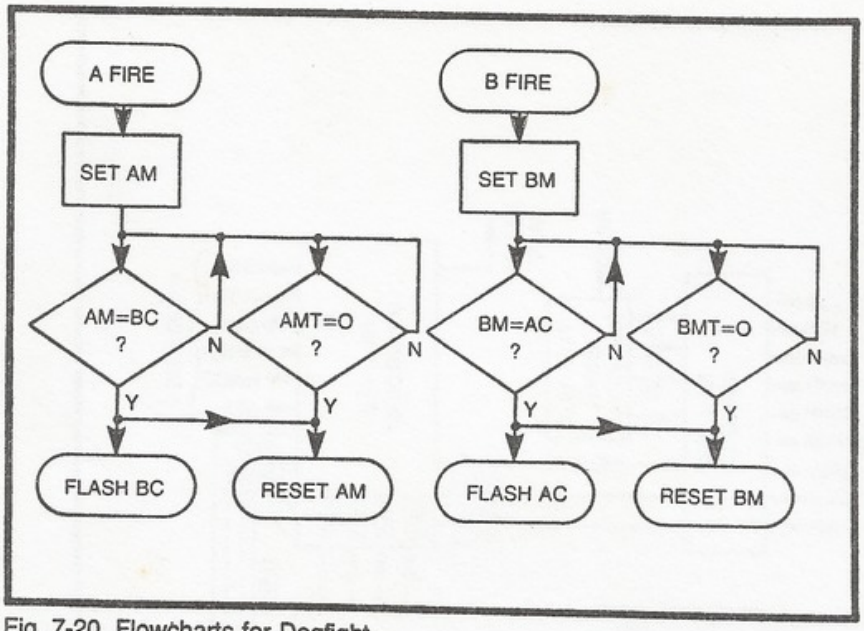


Fig. 7-20. Flowcharts for Dogfight.

So the overall picture looks something like this. There are two aircraft flying freely around the screen. A player can fire a missile from his craft, and that missile figure leaves the primary craft in the same direction, but at twice the speed. The missile can fly for about 1 second before it is blanked from the screen and returned to the primary craft. If that missile happens to strike the opponent's craft enroute, the opponent's craft is destroyed (flashed) and the missile is returned to its primary figure once again.

The game can go on indefinitely because this particular version has no provisions for keeping score and automatically resetting the game. These features can be added at a later time.

The two identical control panels are quite simple, consisting only of a firing button and a set of flight-path controls. The two potentiometers shown in Fig. 7-21 can be replaced with a joystick control to produce simpler and more realistic interaction between the control panels and activity on the screen.

Vector Logic

This Dogfight game is built around two identical sets of tagalong systems, one set for each player. And that means the experimenter must be prepared to install four slipping-counter boards (Fig. 5-15) and two tagalong-control boards (Fig. 7-13).

The secondary figures in both systems tagalong with the primary figures in the usual fashion until a player launches his missile. The

secondary figure then leaves the primary figure, taking on a speed and direction that is dictated by the set of control data present the moment the missile is fired. In the case of the Missile and Torpedo Attack games described earlier in this chapter, the missile leaves the primary figure with the same speed and direction the primary figure had at the moment of launching. In this case, however, the missile leaves in the same direction, but at a faster speed. And that means the secondary-figure controls must be loaded with speed and direction data that is entirely different from that of its primary figure.

Figure 7-22 shows a complete analysis of control data that is entered into any of the slipping-counter schemes described thus far. There are two relevant mathematical equations that show how fast a figure moves across the screen and, alternately, how long it takes to move across the screen.

The velocity (v) is expressed in rather unusual units of screens per second. It is possible to use other, more conventional units of speed, such as inches per second, but such units vary with the size of the experimenter's TV screen. The screens-per-second unit of speed on the screen.

The velocity of a figure, in either the horizontal or vertical direction, is determined by the first equation in Fig. 7-22. Note that

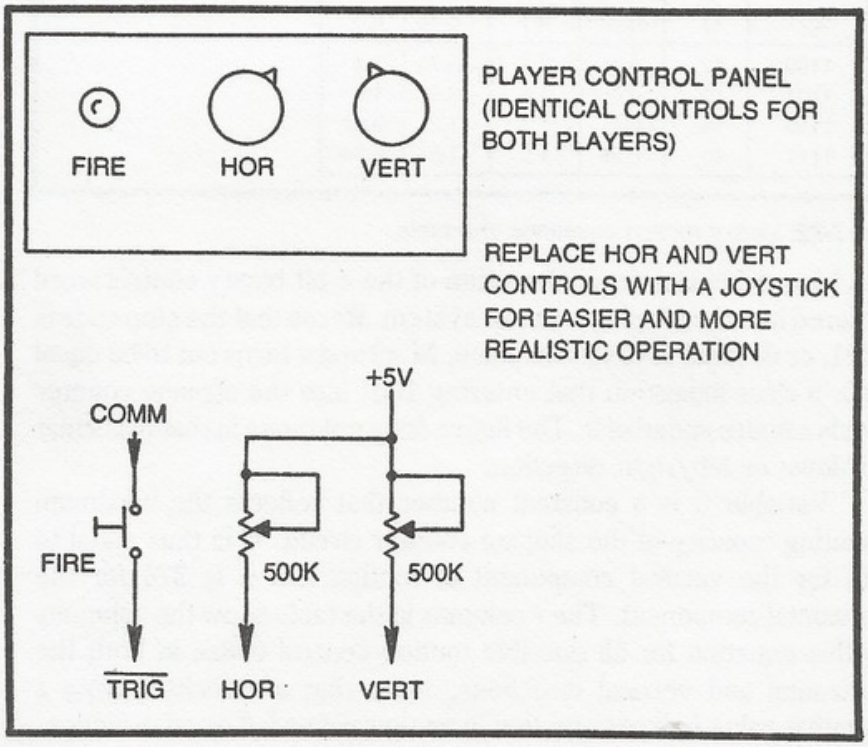


Fig. 7-21. Control panel and schematic for each player.

$$v = \frac{60(9-M)}{C} \qquad t = \left|1/v\right|$$

v = FIGURE VELOCITY IN SCREENS/SECOND
M = DECIMAL VALUE OF SPEED CODE ENTERED INTO THE SLIPPING COUNTERS
C = HORIZONTAL OR VERTICAL COUNT TOTAL
C = 245 FOR VERTICAL MOTION
C = 374 FOR HORIZONTAL MOTION
t = TIME REQUIRED FOR FIGURE TO CROSS THE SCREEN IN A HORIZONTAL OR VERTICAL DIRECTION IN SECONDS

CODE	M	VERTICAL		HORIZONTAL	
		v	t	v	t
0000	0	1.4	0.7	2.2	0.45
0001	1	1.3	0.8	1.9	0.51
0010	2	1.2	0.9	1.7	0.58
0011	3	0.96	1.0	1.5	0.68
0100	4	0.8	1.2	1.2	0.82
0101	5	0.64	1.6	0.98	1.0
0110	6	0.48	2.1	0.73	1.4
0111	7	0.32	3.1	0.49	2.0
1000	8	0.16	6.2	0.24	4.1
1001	9	0.0	00	0	00
1010	10	−0.16	6.2	−0.24	4.1
1011	11	−0.32	3.1	−0.49	2.0
1100	12	−0.48	2.1	−0.73	1.4
1101	13	−0.64	1.6	−0.98	1.0
1110	14	−0.8	1.2	−1.2	0.82
1111	15	−0.96	1.0	−1.5	0.68

v > 0 = MOTION DOWN OR RIGHT
v < 0 = MOTION UP OR LEFT

Fig. 7-22. Vector motion equations and table.

the M variable is a decimal version of the 4-bit binary control word entered into the slipping-counter system. Recall that the stop code is 1001, or decimal 9. In this instance, M=9 and v turns out to be equal to 0, a clear indication that entering 1001 into the slipping counter yields a figure speed of 0. The figure does not move in that particular up/down or left/right direction.

Variable C is a constant number that reflects the maximum counting capacity of the slipping-counter circuit. C is thus equal to 245 for the vertical component of motion and it is 374 for the horizontal component. The v columns in the table show the solutions to this equation for all possible motion control codes in both the horizontal and vertical directions. Note that a velocity having a negative value indicates motion in an upward or left-hand direction. Positive values of v indicate motion down the screen or to the right.

So if a figure happens to be moving with M=6 in the horizontal direction, it is moving downward and to the left. And since the downward velocity is 0.48 screens/second and the left velocity is −0.73 screens/second, it follows that it is moving at a relatively sharp downward angle. If you are familiar with the mathematical procedures for drawing vectors and solving them, you can determine the exact angle and speed.

It is, in fact, quite tempting to digress from the main topic and indulge in some vector analyses of the data in Fig. 7-22. The results could be quite useful, but the matter is better left to reader's who have the knowledge and initiative for doing the job.

Returning to the matter of multiplying the speed components of the secondary figure, look at the table and circuit in Fig. 7-23. The left-hand side of the table shows 10 different slipping-counter controls that might be present at the control inputs of the primary-figure-motion-control board. There are actually as many as 16 possible motion-control codes, but many of them are invalid in the context of our 2× vector logic system. The 8 valid codes are for M values from 5 through 12. The values of 4 and 13 are shown on the table to illustrate the nature of the invalid conditions, but the list also should include 13 through 15 and 0 through 3.

PRIMARY FIGURE					SECONDARY FIGURE					
M	8P	4P	2P	1P	VP	2VP	$8P^1$	$4P^1$	$2P^1$	$1P^1$	M^1
13	1	1	0	1	1.6	3.2	INVALID				
12	1	1	0	0	−0.48	−0.96	1	1	1	1	15
11	1	0	1	1	−0.32	−0.64	1	1	0	1	13
10	1	0	1	0	−0.16	−0.32	1	0	1	1	11
9	1	0	0	1	0.0	0.0	1	0	0	1	9
8	1	0	0	0	0.16	0.32	0	1	1	1	7
7	0	1	1	1	0.32	0.64	0	1	0	1	5
6	0	1	1	0	0.48	0.96	0	0	1	1	3
5	0	1	0	1	0.64	1.28	0	0	0	1	1
4	0	1	0	0	0.8	1.6	INVALID				

2 × SPEED AND DIRECTION VECTOR MULTIPLIER (IDENTICAL FOR BOTH HORIZONTAL AND VERTICAL MOTION)

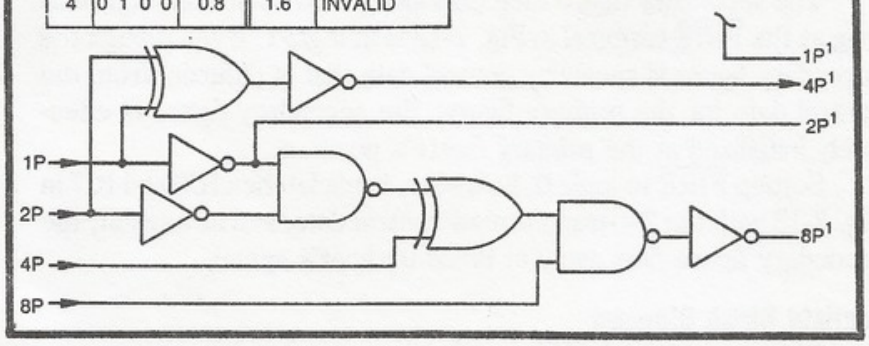


Fig. 7-23. Vector table and circuit for achieving launch velocities twice that of the primary figure.

The primary-figure control codes are translated into velocities in the VP column. This data is taken directly from the table in Fig. 7-22. The 2VP column then shows the VP figure multiplied by 2. Multiplying a velocity vector changes only the speed and not the direction, so the 4-bit binary words on the right-hand side of the table in Fig. 7-23 show the control codes that suit the 2VP velocity figures. Again, this data is taken directly from Fig. 7-22. The M' column merely translates the control codes into their corresponding decimal values.

What this table is saying is that when any primary figure is carrying along a secondary figure with one velocity component of M=12, the secondary figure should be launched with a velocity component of M=15. The secondary figure will move away in the same direction, but with twice the speed of the primary figure.

Primary-figure M figures greater than 12 or less than 3 are considered invalid because the 2× transformation calls for M' values that are greater or less than a 4-bit binary format allows. A figure cannot go any faster than 1111 or 0000.

The circuit in Fig. 7-23 shows how the primary-figure data is translated into a 2× format for its secondary figure. As long as the primary control data stays within its valid operating range of M=5 through M=12, the circuit performs the prescribed transformations.

Interfacing this 2× vector circuit with the tagalong control system is a matter of removing the jumpers specified in Fig. 7-13 and connecting the PVC outputs of that circuit to the corresponding P inputs of the 2× vector circuit. The four outputs of the vector circuit are then connected to their respective PVC' connections in Fig. 7-13.

A second 2× vector circuit can then be interfaced with the PH connections in a similar fashion.

The secondary figure then tags along with its primary figure as long as the $\overline{\text{FIRE}}$ terminal in Fig. 7-13 is at logic 1. Even though the secondary figure is receiving control data that is different from the control data for the primary figure, the secondary figure is effectively initialized at the primary figure's position.

Setting $\overline{\text{FIRE}}$ to logic 0, however, loads latches IC6 and IC7 in Fig. 7-13 with the 2×-transformed-control data, and as a result, the secondary figure flies away at twice its host's speed.

Dogfight Block Diagram

The block diagram in Fig. 7-24 represents the Dogfight game described in this section. The PLAYER A CONTROLS block gener-

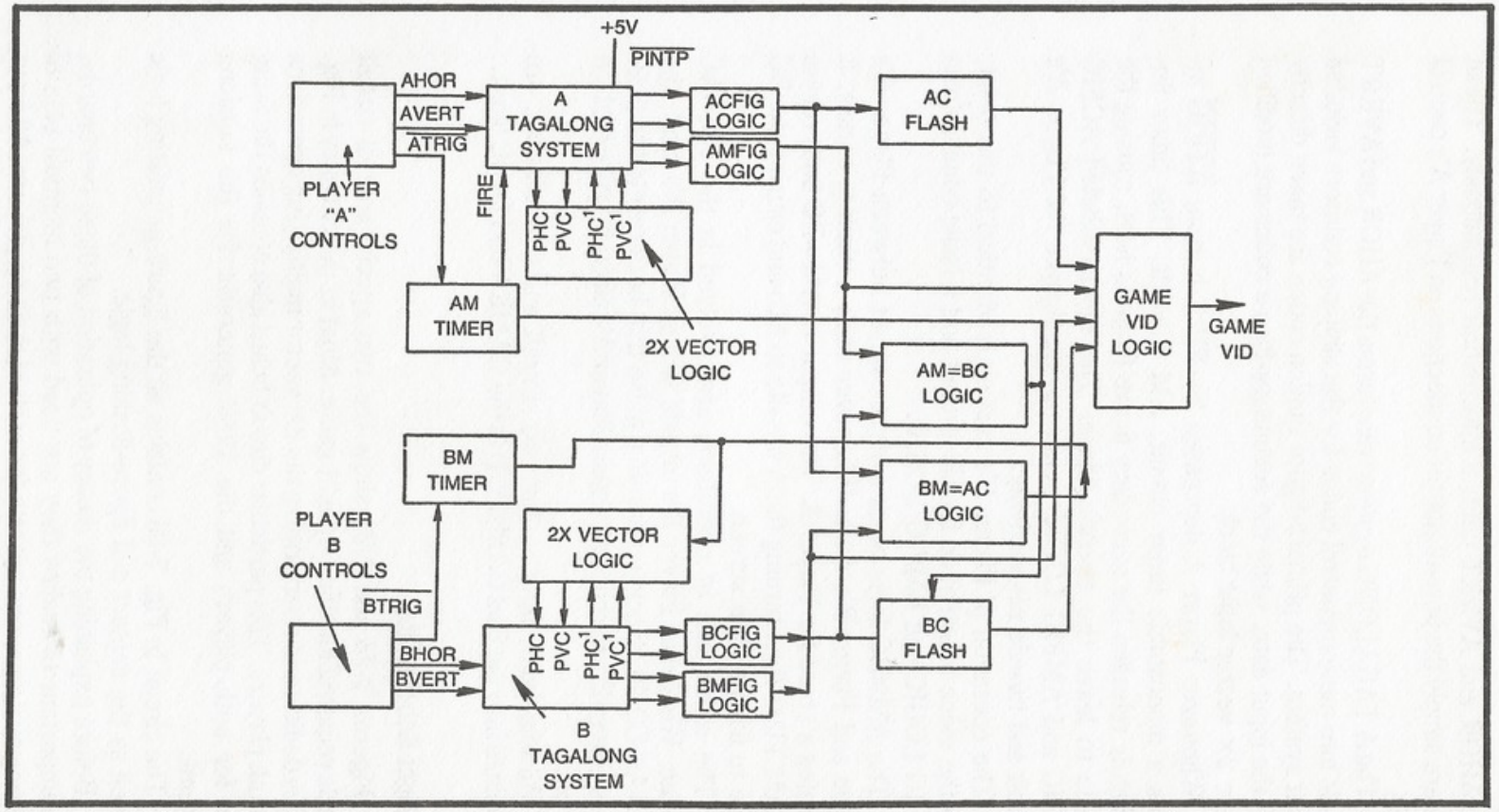


Fig. 7-24. Block diagram of the Dogfight system.

ates AHOR and AVERT control information continuously. These lines are simply the potentiometer connections on Player A's control panel.

The A TAGALONG system transforms the AHOR and AVERT signals into motion-control codes for the slipping counters included in that system. The primary-figure motion codes are taken directly from the input data, while the secondary-figure codes are modified by the 2× vector logic block.

Whenever Player A depresses his FIRE button, $\overline{\text{ATRIG}}$ initiates a monostable timer circuit, AM TIMER. This timer immediately releases the secondary figure from its host, causing the missile to leave the aircraft figure. The blocks labeled ACFIG LOGIC and AMFIG LOGIC generate the figures for Player A's aircraft and missile respectively.

The operation of Player B's system is identical to this point, with the video information for his aircraft and missile coming from BCFIG LOGIC and BMFIG LOGIC.

The AM=BC logic block senses contact between Player A's missile and Player B's craft. Whenever such a contact occurs, it indicates a score for Player A. The output of AM=BC both resets the AM TIMER (returning the A missile to AC) and causing the BC figure to flash on the screen.

The same sort of operations are involved in the BM=AC scheme. Whenever Player B's missile strikes Player A's craft, the BM=AC LOGIC block generates a pulse that both resets the position of Player B's missile and makes Player A's aircraft figure flash on and off.

All four game figures, the two aircraft and two missiles, are combined into the final GAME VID in the GAME VID LOGIC block.

Dogfight Schematics

Figures 7-25 and 7-26 show the two special control circuit boards required for this Dogfight game. Most of the circuitry in Fig. 7-25 is dedicated to performing the 2× vector multiplying operations for both players. This particular circuit board also contains the firing logic for both players and the 128P generator for the tagalong systems.

The circuit in Fig. 7-26 contains all the figure-generating logic as well as the contact and figure-flashing logic.

Before explaining the theory of operation of these two boards, it is important to realize they are used with two identical sets of tagalong systems, one system for each of the two players. All input

and output designations carrying an "A" prefix denote connections to Player A's systems, while those carrying a "B" prefix indicate connections in Player B's system.

It might be helpful at this point to look ahead a bit to the wiring block diagram in Fig. 7-27. You can see the two special Dogfight boards, the vector and figure boards, servicing two complete tagalong systems.

Now notice that the vector board in Fig. 7-25 has four of the 2× vector multiplier circuits described previously in Fig. 7-23. Each receives a set of four primary-figure-motion-control bits from the tagalong control boards.

The first of these four 2× vector circuits accepts bits A1PHC through A8PHC. These are Player A's primary figure horizontal-motion codes. The outputs from this same circuit, designated A1PHC' through A8PHC', are the 2×-corrected motion codes for Player A's secondary figure horizontal-slipping counter.

The vector circuit having inputs A1PVC through A8PVC gets its data from Player A's primary figure vertical-motion control circuit; and the outputs (A1PVC' through A8PVC') go to his secondary figure vertical-control circuit.

The two remaining vector circuits in Fig. 7-25 perform exactly the same operation on Player B's motion-control codes.

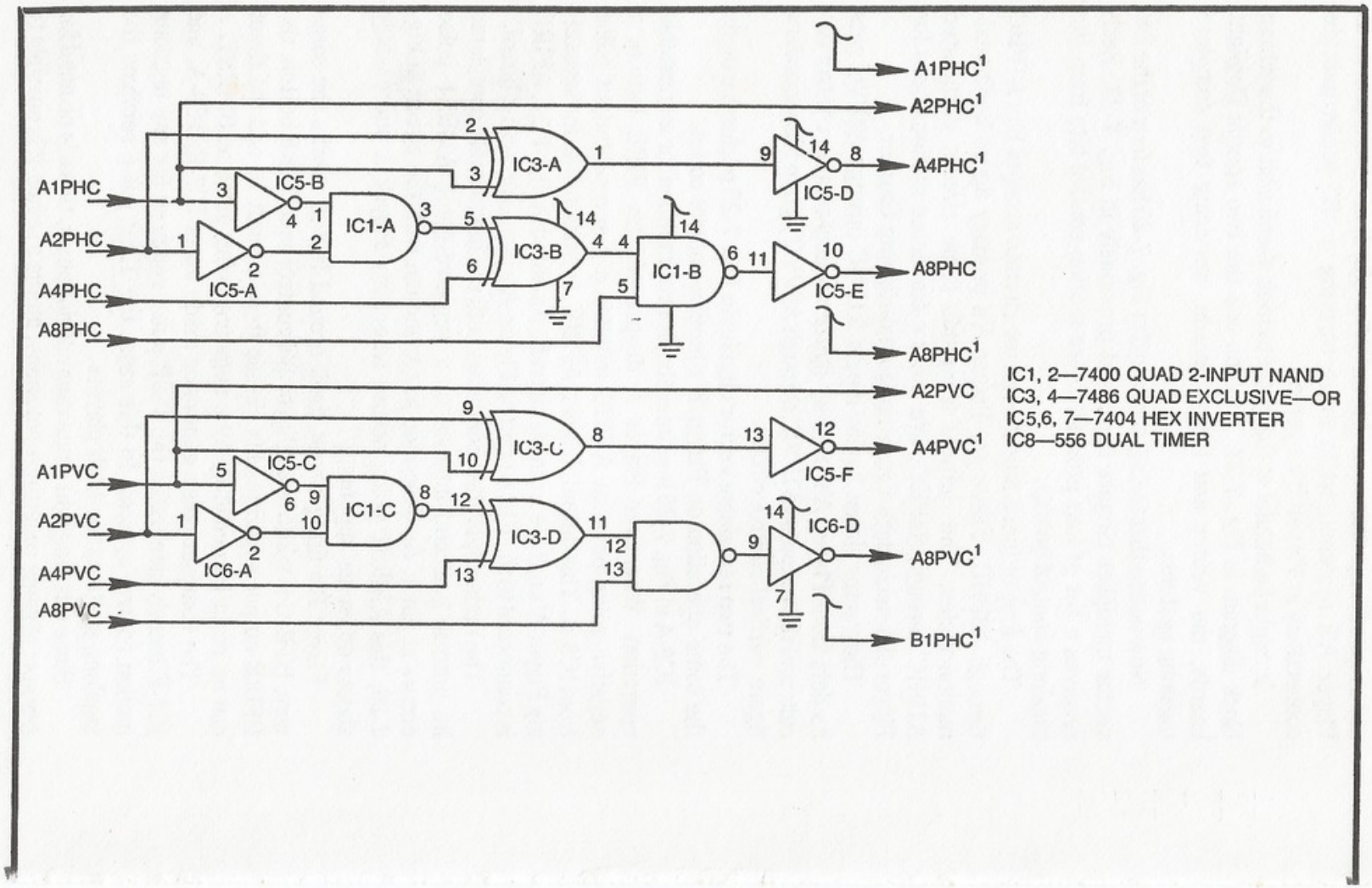
IC8-A in Fig. 7-25 is a timer that is programmed for monostable operation. Whenever Player A despresses his FIRE button, a negative-going pulse at ATRIG initiates a 1-second output pulse from IC8-A. This output, labeled AFIRE, is responsible for separating Player A's missile from his aircraft. As noted in Fig. 7-27, AFIRE is connected to the FIRE input on Player A's tagalong control board.

The timing operation continues until the monostable completes its normal 1-second interval or a negative-going AMRST pulse occurs at pin 4. As described in connection with the circuit in Fig. 7-26, this AMRST pulse occurs whenever Player A successfully shoots down his opponent.

Player B's firing circuit, built around IC8-B, works the same way. BTRIG is taken from Player B's control panel FIRE button, the BFIRE output separates his missile from aircraft B, and the timer can be reset immediately by a negative-going pulse at BMRST.

The compact pulse generator made up of IC7-E, IC9-A, and IC7-F merely generates the 128P pulse required for any tagalong motion control system. In this case, the 128P pulse services the tagalong systems for both players.

Since this Dogfight game runs continuously, there is no need for any sort of game resetting operations, and as a result, it is possible to



IC1, 2—7400 QUAD 2-INPUT NAND
IC3, 4—7486 QUAD EXCLUSIVE—OR
IC5,6, 7—7404 HEX INVERTER
IC8—556 DUAL TIMER

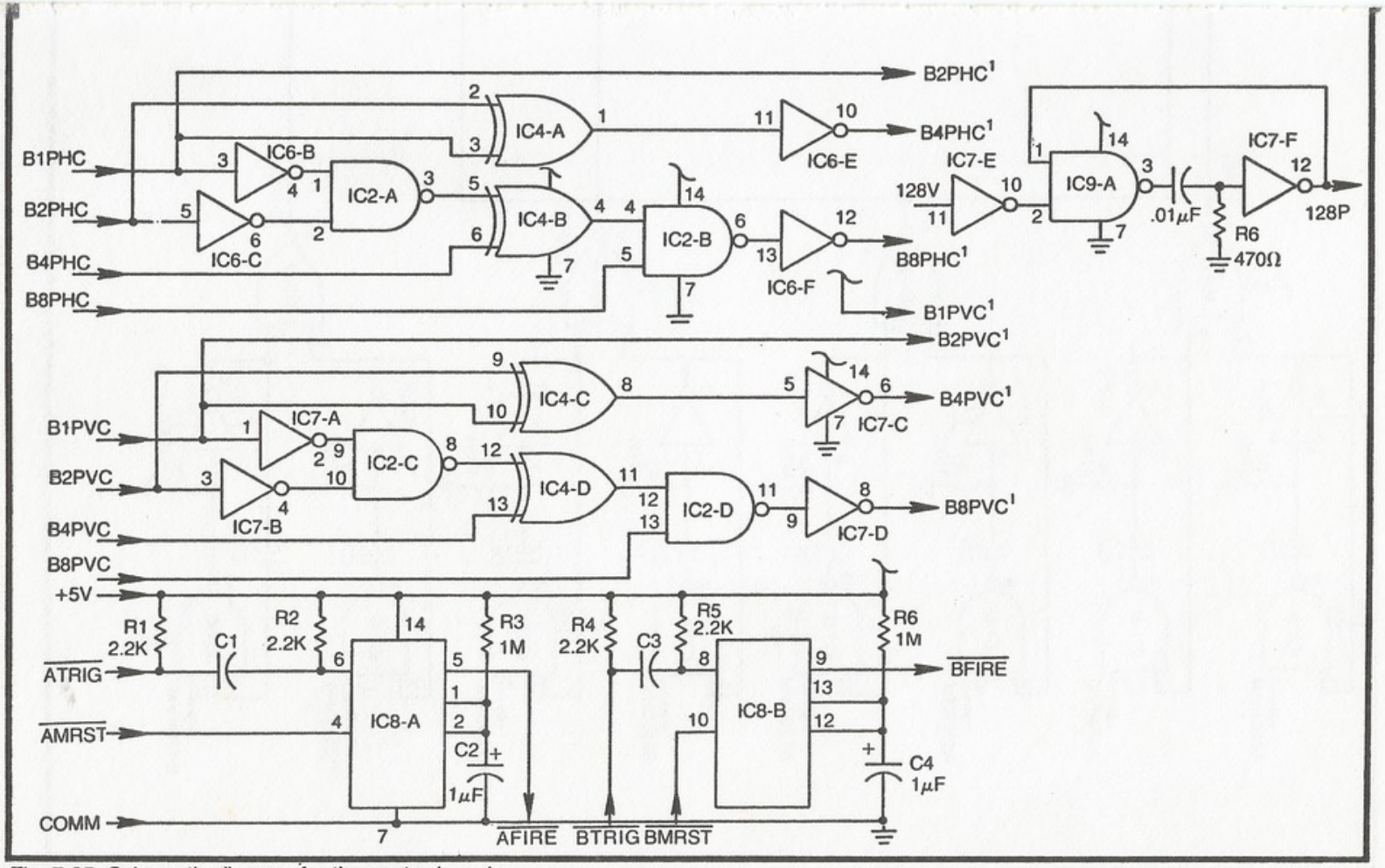
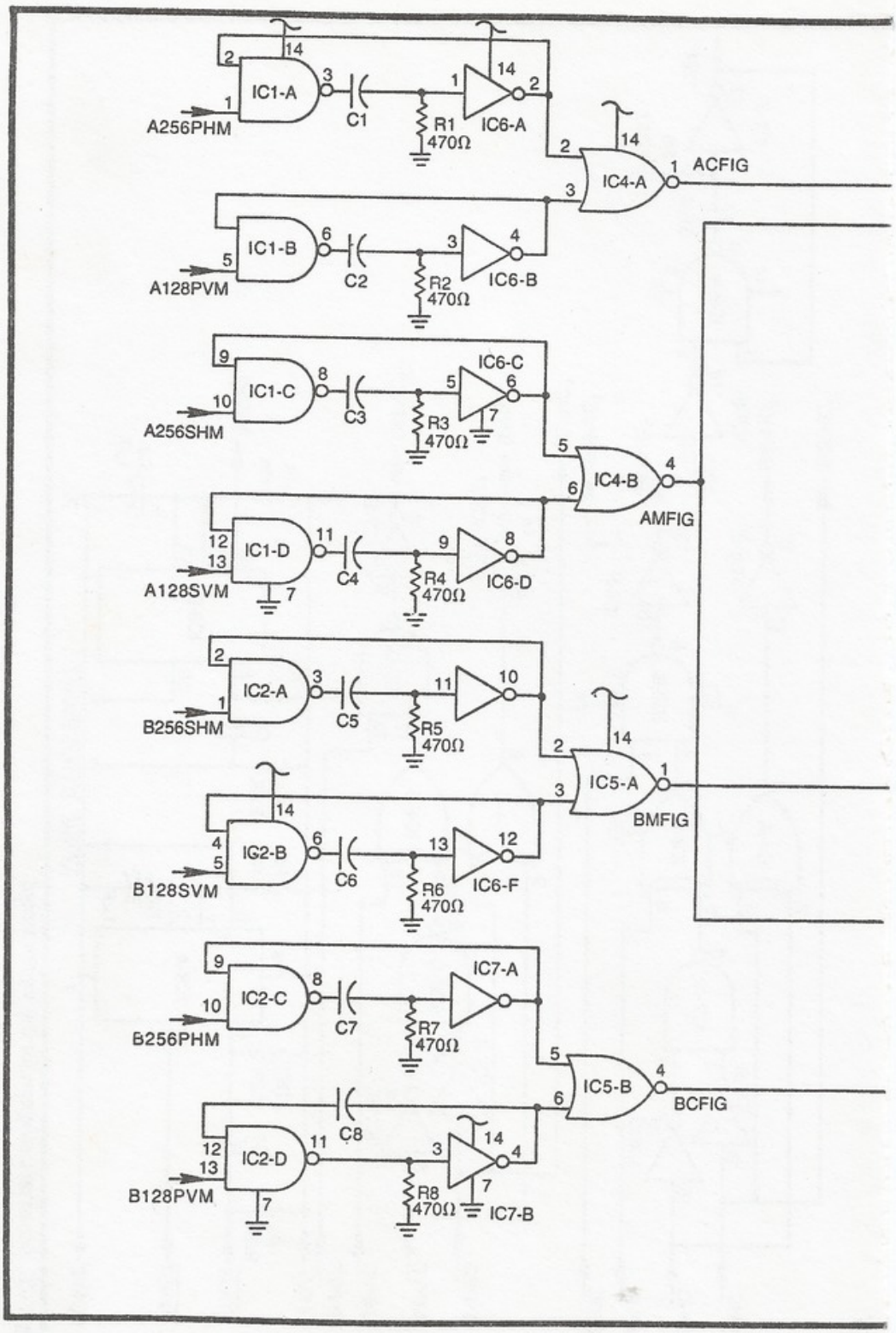
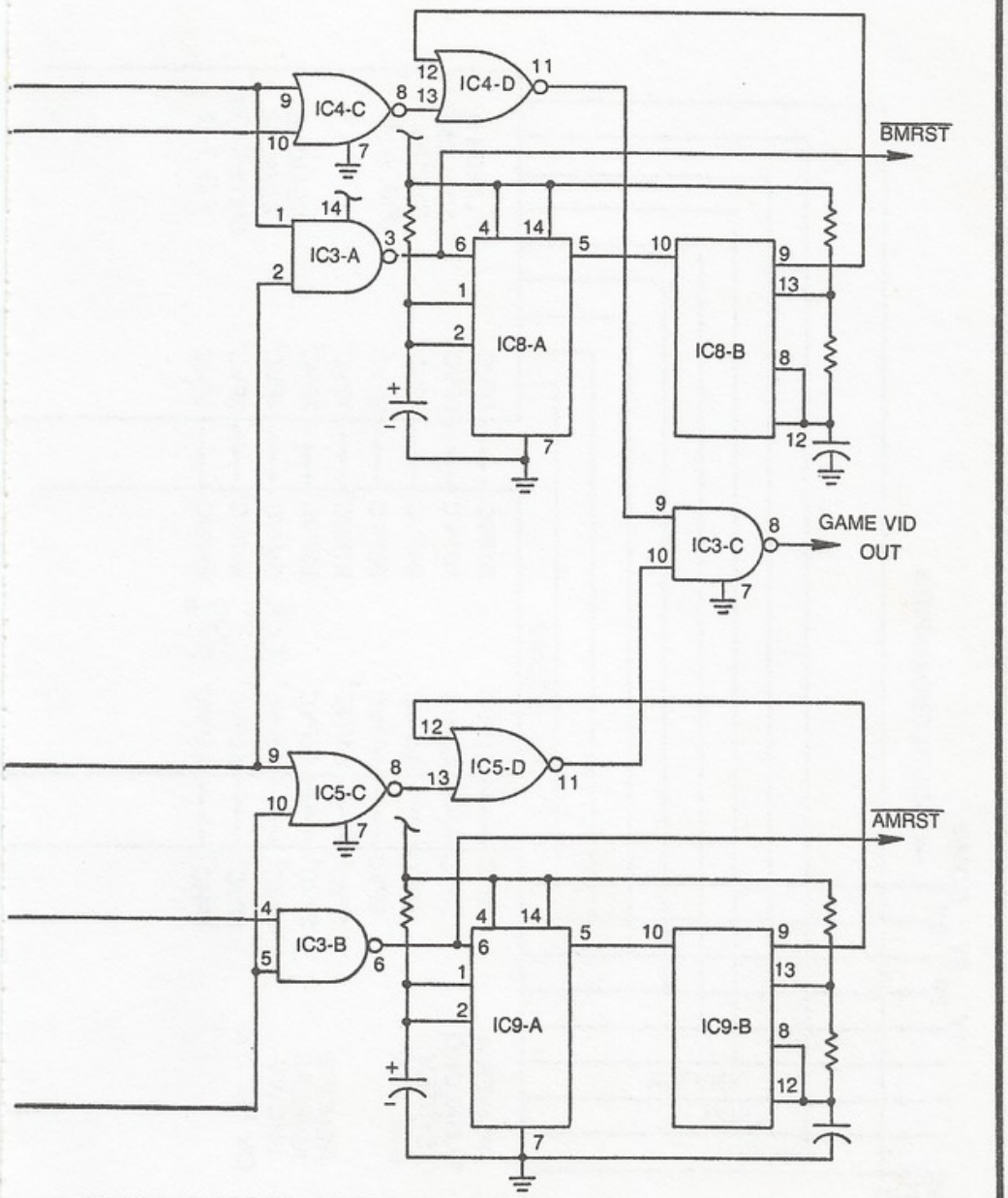


Fig. 7-25. Schematic diagram for the vector board.

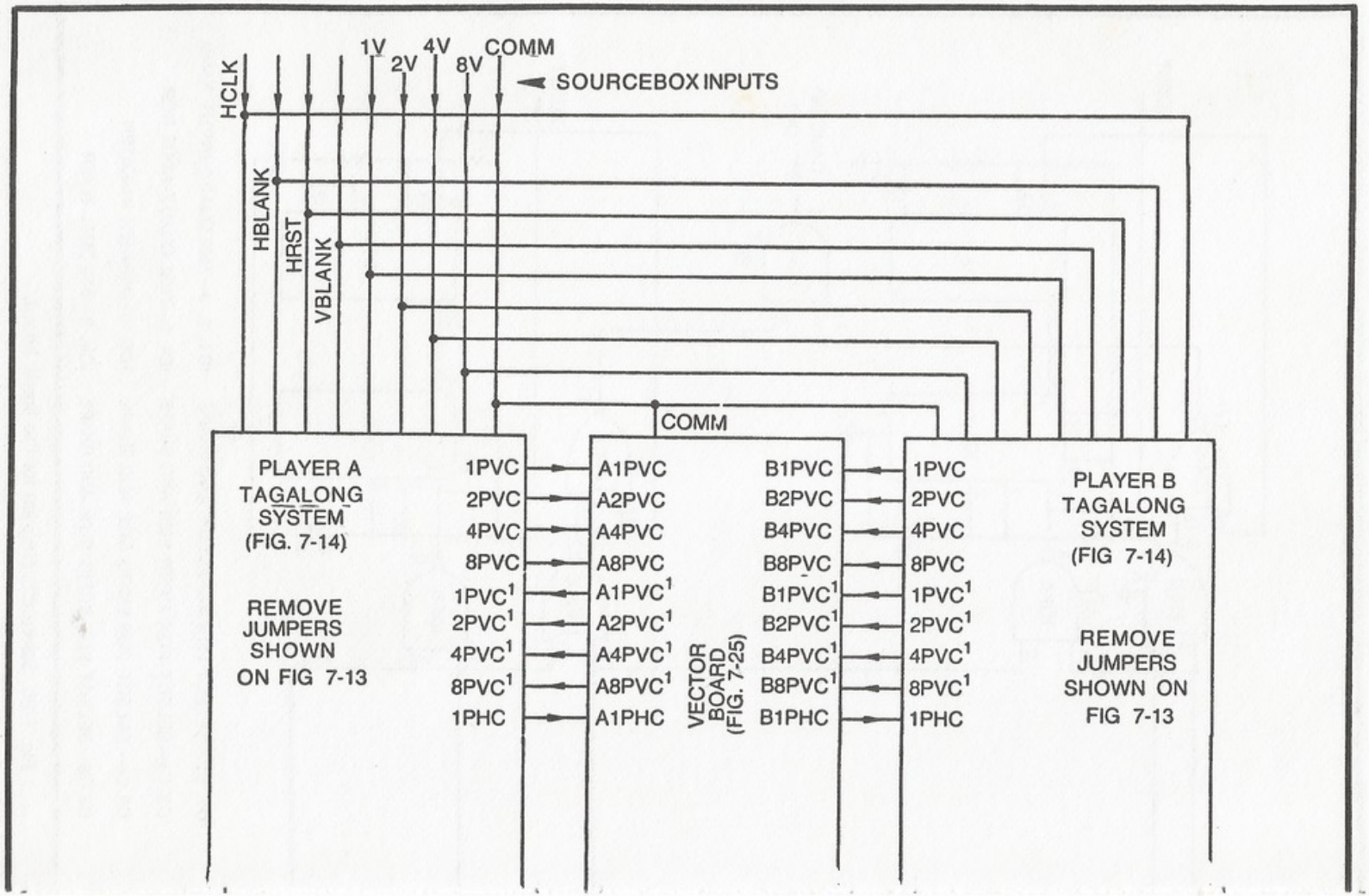

A256PHM
IC1-A
C1
R1
470Ω
IC6-A
IC4-A
ACFIG
A128PVM
IC1-B
C2
R2
470Ω
IC6-B
A256SHM
IC1-C
C3
R3
470Ω
IC6-C
IC4-B
AMFIG
A128SVM
IC1-D
C4
R4
470Ω
IC6-D
B256SHM
IC2-A
C5
R5
470Ω
IC5-A
BMFIG
B128SVM
IG2-B
C6
R6
470Ω
IC6-F
B256PHM
IC2-C
C7
R7
470Ω
IC7-A
IC5-B
BCFIG
C8
B128PVM
IC2-D
R8
470Ω
IC7-B



C1, C2—SELECT FOR ACFIG SIZE AND SHAPE
C3, C4—SELECT FOR AMFIG SIZE AND SHAPE
C5, C6—SELECT FOR BMFIG SIZE AND SHAPE
C7,C8— SELECT FOR BCFIG SIZE AND SHAPE

IC1, 2, 3—7400 QUAD 2-INPUT NAND
IC4, 5—7402 QUAD 2-INPUT NOR
IC6, 7—7404HEX INVERTER
IC8, 9—556 DUAL TIMER

Fig. 7-26. Schematic diagram for the figure board.


HCLK
1V
2V
4V
8V
COMM
SOURCEBOX INPUTS
HBLANK
HRST
VBLANK
COMM
PLAYER A TAGALONG SYSTEM (FIG. 7-14)
REMOVE JUMPERS SHOWN ON FIG 7-13
1PVC
2PVC
4PVC
8PVC
1PVC1
2PVC1
4PVC1
8PVC1
1PHC
A1PVC
A2PVC
A4PVC
A8PVC
A1PVC1
A2PVC1
A4PVC1
A8PVC1
A1PHC
VECTOR BOARD (FIG. 7-25)
B1PVC
B2PVC
B4PVC
B8PVC
B1PVC1
B2PVC1
B4PVC1
B8PVC1
B1PHC
1PVC
2PVC
4PVC
8PVC
1PVC1
2PVC1
4PVC1
8PVC1
1PHC
PLAYER B TAGALONG SYSTEM (FIG 7-14)
REMOVE JUMPERS SHOWN ON FIG 7-13

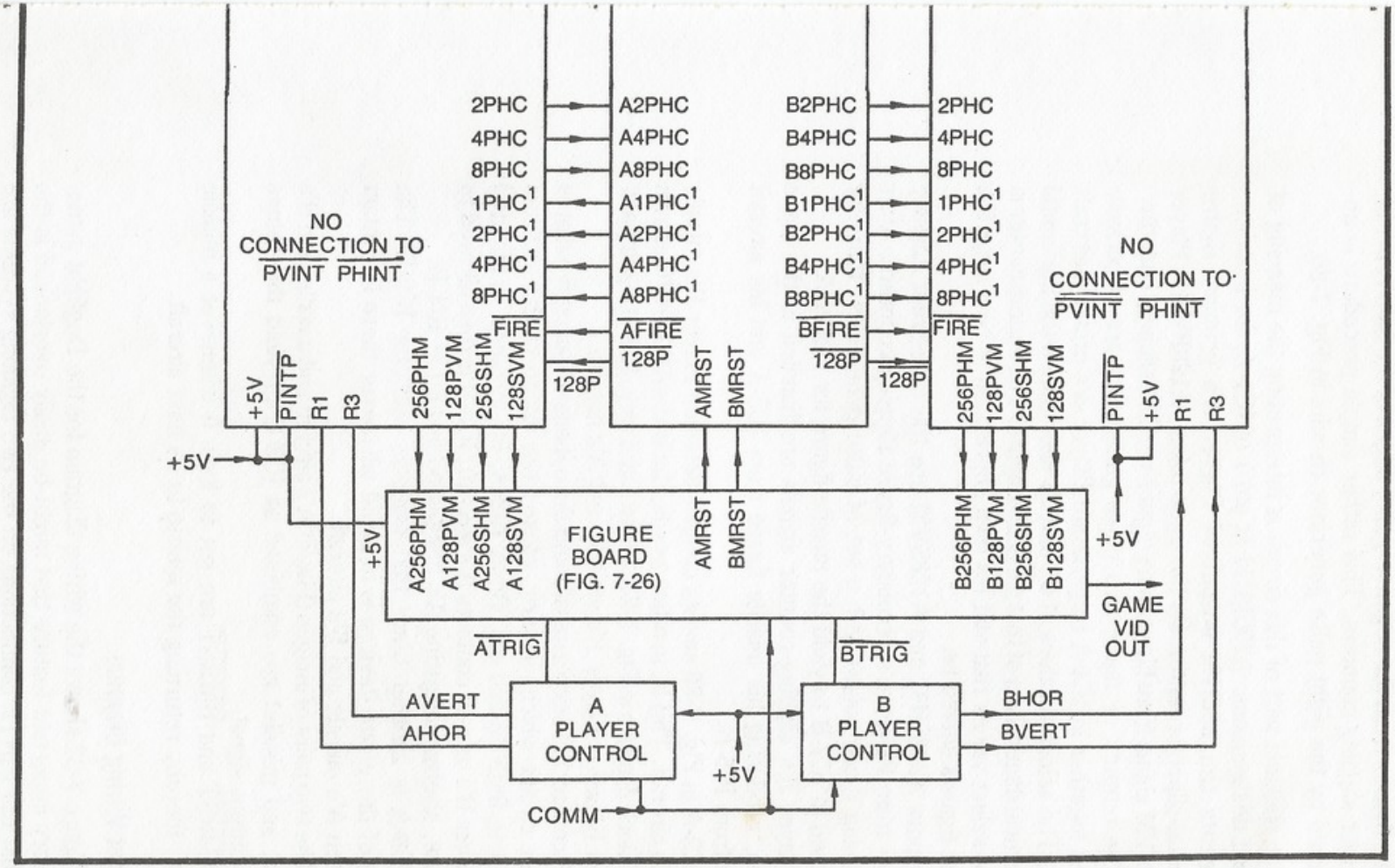


Fig 7-27 Wiring diagram for the Dogfight system.

build the figures from the negative-going edges of the high-order bit on all four slipping counters. This rather simple procedure is implemented by the eight pulse generator circuits in Fig. 7-26.

The trickiest part of this circuit is interpreting the meaning of the input designations. A256PHM at pin 1 of IC1-A, for instance, represents the 256HM output of Player A's primary figure horizontal-slippng counter. By the same token, A128PVM is Player A's 128VM signal from his primary-figure vertical-slipping counter.

The outputs of these two pulse generators are effectively ANDed together in IC4-A to generate Player A's primary-aircraft figure. The selected values of C1 and C2 determined the horizontal and vertical dimensions of that particular figure. The experimenter is free to select values that suit his own impression of how large the aircraft figures should be.

Inputs A256SHM and A128SVM are the high-order counter outputs from Player A's secondary-figure slipping counters. After shortening these pulses with a set of pulse generators, they are combined in IC4-B to yield the missile figure for Player A.

Player B's slipping-counter signals are handled in the same fashion, producing his missile figure from IC5-A and his aircraft figure from IC5-B.

IC3-A in Fig. 7-26 senses contact between Player B's missile and A's aircraft. This is, in other words, the BM=AC LOGIC shown in the block diagram in Fig. 7-24. In the same way, IC3-B signals any contact between Player B's aircraft and A's missile.

A contact between a missile and the opponent's aircraft initiates a timer circuit which, in turn, causes the image of the stricken aircraft to flash on and off. Timer IC8-A, for instance, is initiated whenever B's missile contacts A's aircraft. IC8-B is then allowed to oscillate, alternately gating A's images on and off at IC4-D.

IC9-A is another timer that controls oscillator IC9-B. The action of these two devices is initiated whenever there is contact between A's missile and B's aircraft.

The two pairs of images (Player A's aircraft and missile, and B's aircraft and missile) are combined at IC3-C to yield the games video-output signal.

$\overline{\text{AMRST}}$ and $\overline{\text{BMRST}}$ are set to logic 0 whenever a missile scores, thereby returning the missle to its host aircraft.

Dogfight Wiring Diagram

Figure 7-27 shows the wiring diagram for the Dogfight game. One very important feature that might be easily overlooked is the fact that the PINTP connections on the two tagalong systems are

connected directly to logic 1, or +5V. This connection disables the primary-figure-initializing circuit so that the game procedes continuously. There is never any condition that calls for initializing the positions of either primary figure.

This system uses a total of eight circuit boards and two control panels: two slipping counter and one tagalong control for each player, as well as a vector and figure board. The tagalong system for Player A should be operated from the +5V source for Game A, and the tagalong system for Player B should operate from the Game-B supply. The vector board, figure board, and two control panels must then be powered from an auxiliary +5-V, 1-A supply. Of course all COMM connections should be connected together.