

# Sailboat Regatta Starting Aid (SRSA) REQUIREMENTS DOCUMENT

CENG 4534

Digital System Design

Final Project

University of Houston- Clear Lake

Instructor: Dr. E. Husband, Ph.D., P.E.

April 22, 2005

Robert D. Hunkins



## Table of Contents

BACKGROUND.....	5
OBJECTIVE .....	6
SYSTEM DESCRIPTION .....	6
USER INTERFACE.....	6
BLOCK DIAGRAM .....	7
One-Pulse Circuit .....	8
Sequence Controller .....	9
Horn Driver .....	12
Timer .....	13
BCD to Seven segment Decoders.....	18
TIMING CONSTRAINTS .....	19
EQUIPMENT REQUIREMENTS .....	20
TIME SCHEDULE .....	20
TESTING .....	20
FUTURE WORK .....	21
COMMENTS AND CONCLUSIONS .....	22
REFERENCES .....	23
ACKNOWLEDGEMENTS .....	23
Appendix A- VHDL Code for the SRSA .....	1
-- Horn Driver VHDL module .....	1
-- One pulse Circuit VHDL module .....	1
--Timer Circuit VHDL module.....	3
--BCD to seven Segment Decoder.....	4
-- Sailboat Race Starting Aid VHDL module .....	6

## Table of Figures

Figure 1:User Interface.....	7
Figure 2: System Block Diagram.....	8
Figure 3: One pulse circuit State Graph .....	9
Figure 4: Logic diagram for one-pulse circuit .....	9
Figure 5 : Sequence Controller State Graph .....	11
Figure 6: Sequence Controller Logic Diagram.....	12
Figure 7: Horn driver.....	12
Figure 8: Horn Driver Logic diagram.....	13
Figure 9: Timer LOAD signal ASM CHART .....	15
Figure 10: RESET Signal ASM CHART .....	16
Figure 11: Enable signal ASM chart .....	17
Figure 12: Horn and Alert Decoder.....	18
Figure 13: Numeric LED Display Internal Circuit Diagram.....	19
Figure 14: Prototype System using CPLD and LED displays .....	20

## Table of Tables

Table 1: One pulse circuit State Transition table.....	9
Table 2: Sequence Controller State Transition Table.....	11
Table 3 State Transition table for the Horn driver.....	13



## BACKGROUND

A regatta is a set of sailboat races. These races are conducted according to a set of rules laid out by the International Sailing Federation (ISAF), the international governing body of the sport. A subset of these rules governs the way the races are started. The races are started by a group of people aboard a powerboat at one end of the starting line. These people are referred to as the Race Committee. They are responsible for the conduct of the race and for recording the finishes of the boats in the race.

The Race Committee provides precisely timed signals to the sailboats in the race via International Code Flags and sound signals, usually a horn. According to the rules set forth by the ISAF, races are to be started using a five-minute time sequence. This sequence is started at five minutes before the race by raising a flag that is associated with a particular fleet of boats. This flag, called the class flag, is accompanied by a sound signal to draw attention to the raising of the flag. At exactly four minutes prior to the start, a second flag, called the preparatory flag, is raised and is also accompanied by a sound signal. At one minute prior to the start, the preparatory flag is lowered with a sound signal. Finally, at the start, the class flag is lowered, also accompanied by a sound signal.

There may be several fleets that race, and they are started in succession. Often, the lowering of the starting signal for one fleet is simultaneous with the raising of the class flag for the next fleet. Races with multiple fleets are conducted in a series of these starting sequences.

The Race Committee also has other signals, which may be made to communicate with the sailboats. These signals are:

- Postponement - a postponement flag is raised with two sounds. Races that have not already started are postponed. The race committee uses this in the event of an error in the starting sequence. For example, if the wind changes direction drastically, the starting line may be repositioned, requiring a postponement. Lowering the postponement flag, concurrent with one sound signal, ends the postponement period. The starting sequence will begin exactly one minute after the postponement flag is lowered.
- Individual Recall – one or more sailboats are over the starting line early. Those boats must return. Subsequent starting sequences are not affected. The individual recall flag is raised with one sound.
- General Recall – the fleet last started is recalled. All boats in that fleet must return to the starting area. The general recall flag is raised, with two sounds. The general recall period is ended similarly to the postponement period, by lowering the flag with one sound. The starting sequence will resume in one minute.
- Abandonment – All races started are abandoned, or cancelled. The abandonment flag is raised with three sounds. The starting sequence may or may not resume. If the sequence is to resume, the flag is lowered with one sound indicating that the starting sequence will begin in one minute.

- Come within Hail –The Race committee wishes to communicate with the Captains of the sailboats verbally. The come within hail flag is raised with one sound.

## OBJECTIVE

The objective of this project is to design a system that will aid a Race Committee in the execution of their duties. This system will display a count down timer, be able to handle a range of fleets (1-9), allow the race committee to signal Postponement and General recall and also allow them to manually signal with the horn for Abandonment, Come within Hail, and Individual recall.

This system is envisioned to be powered by either internal batteries or from power derived from the race committee boat battery. The user interface is via a seven-segment LED display and pushbuttons. An Electrical interface to a horn system with a separate power supply is envisioned. A piezoelectric buzzer, or an electronic chime is used to alert the race committee that an event is imminent.

## SYSTEM DESCRIPTION

The system is composed of a timer, a sequence controller, four BCD to seven segment display drivers four seven segment displays, and a horn driver (refer to Figure 1)

### USER INTERFACE

The user interfaces with the system by means of four pushbuttons, four seven segment numeric LED displays, an alert signal and a horn signal.

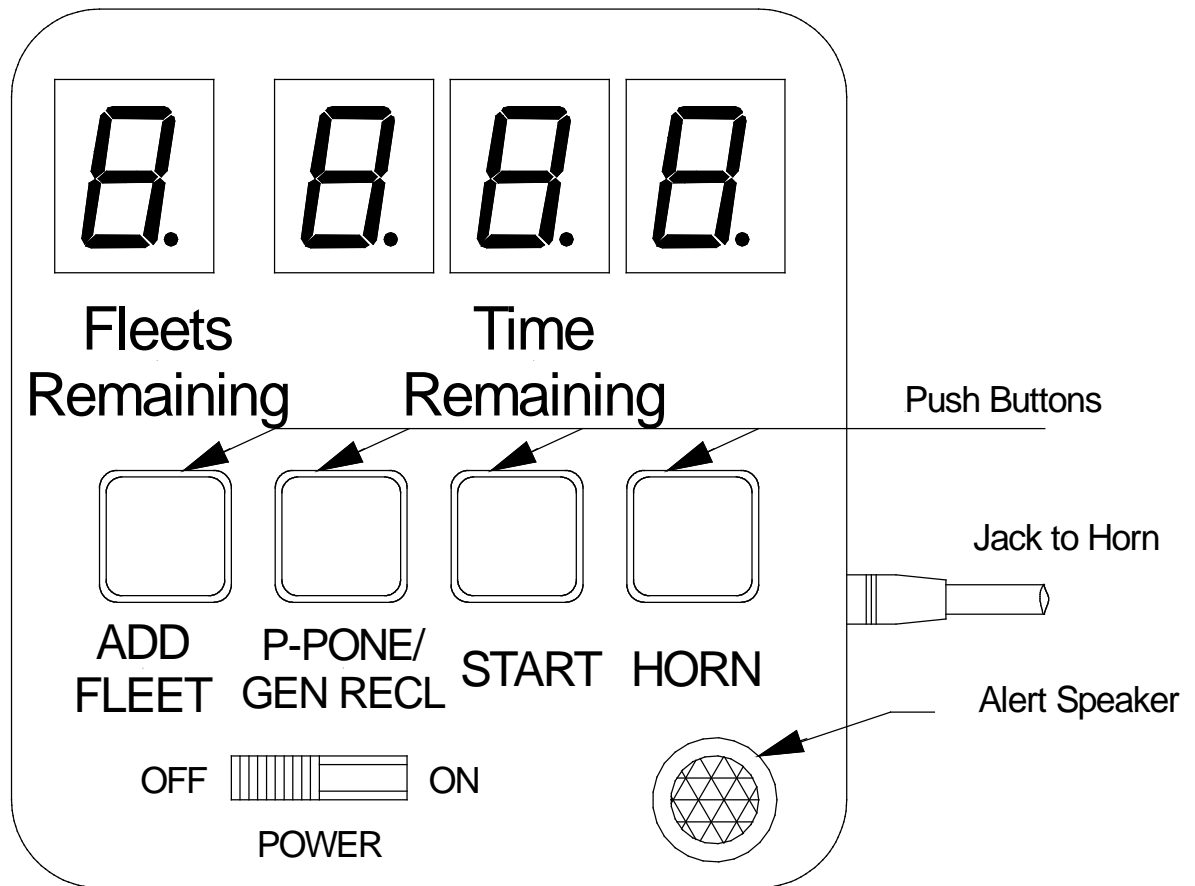
The four buttons perform the following functions:

- **Add Fleet:** Increment the fleet counter by one. This is used to enter the desired number of fleets that will be started. Between one and nine fleets may be entered and is displayed on the “FLEETS” LED display. Fleets may be added at any time and will not stop a sequence underway.
- **Postpone/General Recall:** This button is used to signal a general recall or postponement. Since the audible signals and sequencing are identical for both cases, either signal may be sent using this button, the difference being the flag that is raised. These two cases are therefore transparent to the SRSA system. When this signal is sent, the timer resets to 6:05 seconds. This will allow the Race committee five seconds to synchronize the flag signal with the audible signal when the start is pressed.
- **Start:** This button is used to begin sequencing. The system will begin counting from the time displayed and will not stop until the time has expired and the fleet counter is zero, or a postponement/general recall signal is sent. This switch is a low level input switch on the PLDT-3 board, and must be inverted for use in this application.
- **Horn:** The Individual Recall, Come within Hail and Abandonment signals may be made at any time in the starting sequence, and may not necessarily stop the sequence. To allow the Race committee the ability to make these signals, this button provides a direct control of the horn. The horn signal is sent for as long as the button is depressed.

The display is comprised of four digits. The first digit shows the number of fleets to be started. The next digit shows the minutes remaining in the current fleet starting sequence.

The two audible signals are Race Committee Alert and Horn. For the Prototype device, both are simulated using LED's. The Race Committee alert signal is an intermittent signal made during the five-second period before a flag signal is to be made. The horn signal is intended to

be a much louder signal, audible to the fleet and sent at the appropriate times.



**Figure 1:User Interface**

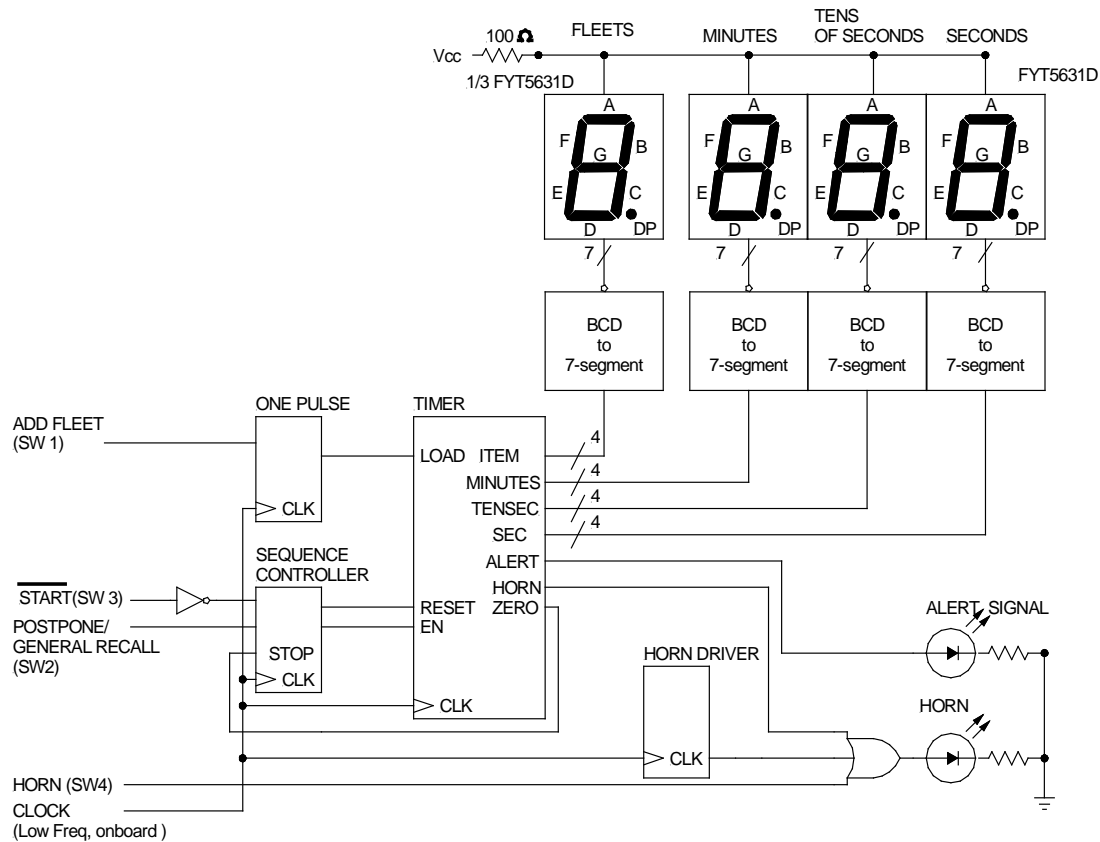
### *BLOCK DIAGRAM*

The Block Diagram for the prototype system using LED's as the outputs for the Horn and Race Committee Alert signals is presented in Figure 2: The system is composed of a timer circuit, a sequence control circuit, a horn driver, a one-pulse circuit, four BCD to seven segment decoders with low level output and four seven segment LED displays.

The timer circuit, sequence controller, horn driver, BCD-to-Seven segment decoders, and the one-pulse circuits are implemented using VHDL and programmed into the RSR electronics PLDT-3 CPLD board. For testing purposes, the onboard low frequency clock is used to provide a clock signal at approximately 2 Hz. The external LED displays are two Ningbo Foryard Optoelectronics Company's FYT 5631D three digit numeric LED displays. Only one digit of the second display unit is utilized.

The output to the horn is from a three-input OR gate. The horn can receive a signal from three separate sources. The first and simplest is the manual horn signal from the pushbutton. This signal may be any duration and remains high as long as the button is depressed.

The second is from the timer circuit and is a single horn signal, one clock cycle in duration. The third is from the horn driver, which processes the postponement/general recall signal. The horn driver will output two horn signals, each one clock period in length separated by one clock period of silence.



**Figure 2: System Block Diagram**

*One-Pulse Circuit*

This circuit is used to process the pushbutton switch input when the user request to add a fleet. This unit is a Mealy Machine, designed to detect a button press, debounce the switch, synchronize it to the clock and output a pulse of one clock cycle duration.

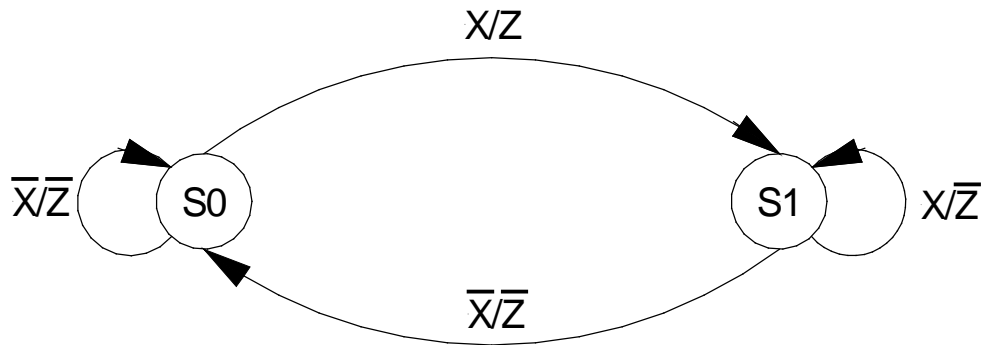
The State Graph for this machine is presented in Figure 3. The State table and assignments are presented in Table 1.

The data flow model was chosen to implement this circuit in VHDL. The equations were determined as follows:

$$Q = X$$

$$Z = X\overline{Q}, \text{ for a Total Input Cost of 2 and a total Gate cost of 1.}$$

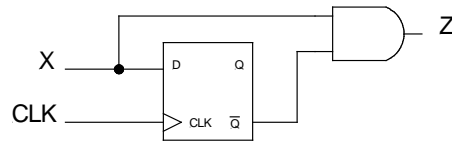
If a hardware circuit were to be built, the components required are shown in Figure 4.



**Figure 3: One pulse circuit State Graph**

**Table 1: One pulse circuit State Transition table**

State Assignment	Present State	Next State		Output Z	
		X=0	X=1		
0	S0	S0	S1	0	1
1	S1	S0	S1	0	0



**Figure 4: Logic diagram for one-pulse circuit**

### Sequence Controller

The purpose of the sequence controller is to debounce switch inputs and to respond to those switch inputs as well a signal from the timer.

The INPUTS to the sequence controller are:

PP (Postpone/General Recall button)

ST(Start, an inverted signal from the start button)

STOP(from the ZERO output in the timer)

CLK (1 Hz clock for the field unit, 2 Hz from the PLDT-3 low freq clock for testing purposes)

Based on these inputs, the sequence controller controls two outputs: an enable counting signal (EN), and a reset signal (RESET). Both are sent to the timer circuit.

The controller has three states, as seen in Figure 5. State S0 is the stopped state. In this state, neither enable nor the reset signals are sent. If a PP signal is received, the sequence controller will transition to state S1, outputting a RESET signal for one clock cycle. If the ST signal is received and the STOP signal is low, the controller will transition to state S2 setting the EN signal high.

If the STOP signal from the timer is high, indicating that the system is clear, and all zeros are displayed, then the START signal will have no effect. The Add Fleet button must be used to

increment the fleet counter. If the PP signal is sent while the STOP signal is high, the timer treats this as a general recall situation and will set fleet counter to 1 without an input from the Add Fleet button.

State S1 is the state the controller goes to when the PP signal is received. When in this state, the RESET signal has already been sent to the timer circuit for only one clock pulse. If the PP signal continues to be received, indicating the pushbutton is still being depressed then the controller will remain in S1. When the button is released, the controller reverts to state S0. Subsequent presses of the Postpone button will signal cause the controller to transition to state S1 again, but unless the START signal is sent in the meantime, no change in the timer will occur.

State S2 is the counting state. In this state, the EN signal to the timer is set, and the controller remains in this state unless the timer signals with a STOP signal or a PP signal is received. If the PP signal is received, the controller transitions to state S1, disables the EN signal and send the RESET signal. The following clock cycle, the controller returns to State S0 and awaits another ST signal. If the timer sends a STOP signal, the controller transitions to State S0, and disables the timer by disabling the EN signal.

The State Transition table for the controller is presented in Table 2. The State assignment was chosen to minimize the number of inputs and gates required. The controller was implemented in VHDL using a data flow model. Using the Logic Aid tool, the following flip flop excitation equations were generated:

Simplification Routine: PI Chart

$$D(Q1) = (ST + Q1)(STOP')(PP')(Q0')$$

$$\text{Input Cost} = 6 \quad \text{Gate Cost} = 2$$

$$D(Q0) = (PP)(ST' + STOP' + Q1')$$

$$\text{Input Cost} = 5 \quad \text{Gate Cost} = 2$$

$$EN = (ST + Q1)(STOP')(PP')(Q0')$$

$$\text{Input Cost} = 6 \quad \text{Gate Cost} = 2$$

$$RESET = (PP)(Q0')(ST' + STOP' + Q1')$$

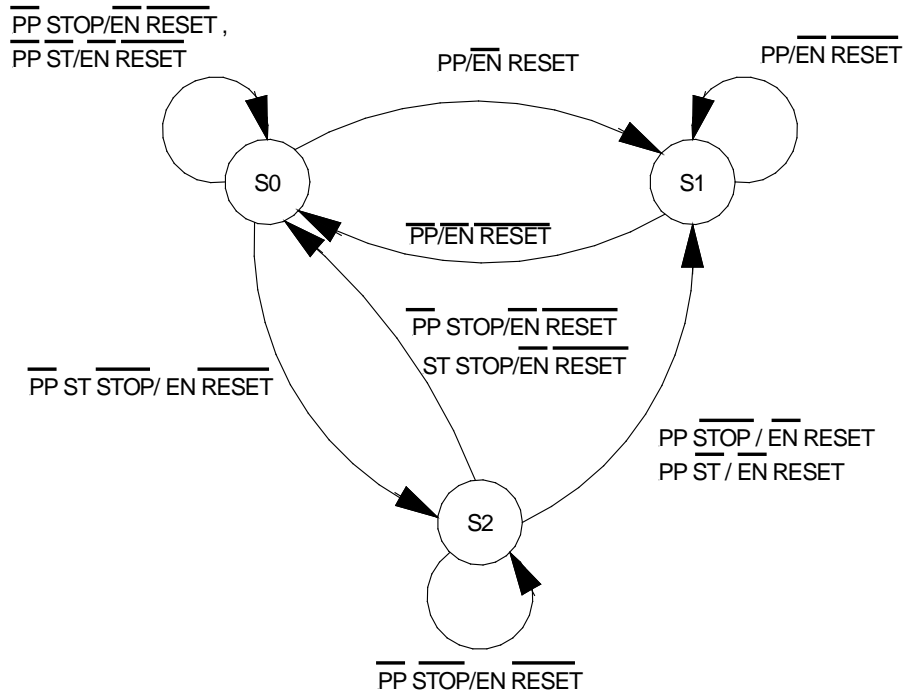
$$\text{Input Cost} = 6 \quad \text{Gate Cost} = 2$$

$$***\text{Total Input Cost} = 23***$$

$$***\text{Total Gate Cost} = 8***$$

Logic Aid indicated that using a Product of Sum form would use less gates and inputs than the Sum of Products form.

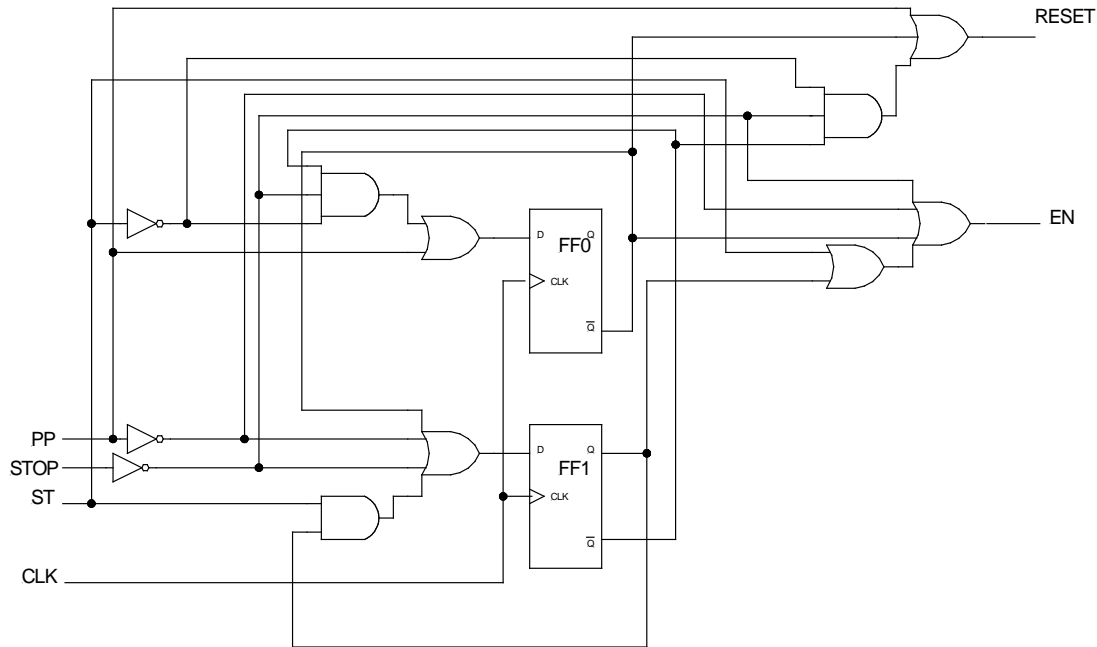
The circuit to implement this controller is presented in Figure 6.



**Figure 5 : Sequence Controller State Graph**

**Table 2: Sequence Controller State Transition Table**

State Assignment	Present State	INPUTS	Next State								Outputs							
											EN RESET							
		PP ST STOP	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
00	S0		S0	S0	S2	S0	S1	S1	S1	S1	00	00	10	00	01	01	01	01
01	S1		S0	S0	S0	S0	S1	S1	S1	S1	00	00	00	00	00	00	00	00
10	S2		S2	S0	S2	S0	S1	S1	S1	S0	10	00	10	00	01	01	01	00

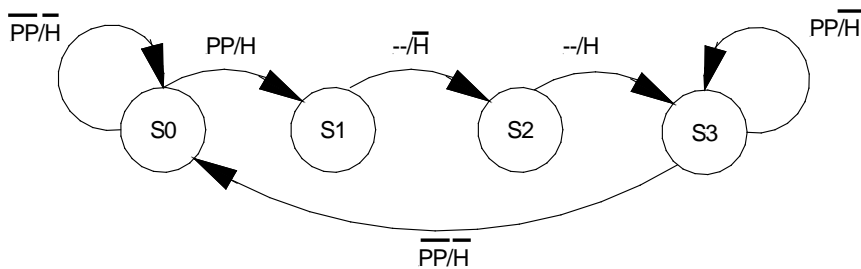


**Figure 6: Sequence Controller Logic Diagram**

*Horn Driver*

The horn driver circuit is a sequential state machine that has one input and one output, not including the clock input. The input is the signal from the pushbutton signaling a Postponement or a General Recall. The output is two pulses, each one clock length in duration separated by a one clock pulse of silence. Given a 1 HZ clock, this will give two one second long horn signals separated by one second of silence.

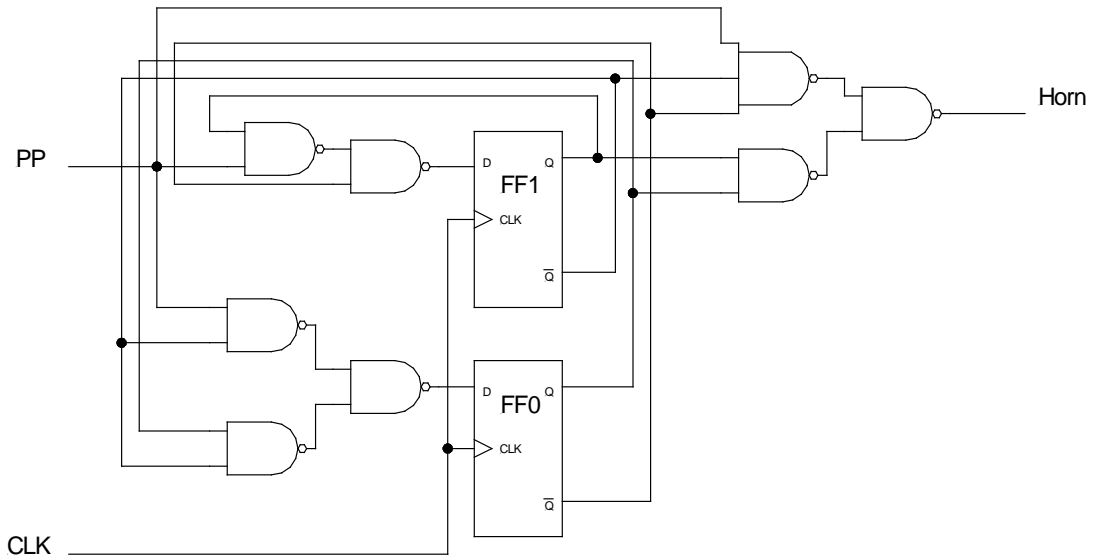
The state graph for this circuit is shown in Figure 7, and the state table in Table 3. A Mealy Machine was chosen for this system over a Moore Machine as it was found that a Moore Machine had a delay of one clock cycle from the press of the button compared to the Mealy Machine. The logic diagram for this circuit is shown in Figure 8.



**Figure 7: Horn driver**

**Table 3 State Transition table for the Horn driver**

State Assignment	Present State	Next State Input X		Output Horn	
		PP=0	PP=1	PP=0	PP=1
00	S0	S0	S1	0	1
01	S1	S2	S2	0	0
11	S2	S3	S3	1	1
10	S3	S0	S3	0	0



**Figure 8: Horn Driver Logic diagram**

*Timer*

The timer is a behavioral model unit, designed with four inputs and seven outputs.

The inputs are:

- LOAD

The load signal will increment the fleet count by one for each clock cycle it is high. When the fleet count is at nine it will roll over to 0. This allows for a maximum of nine fleets. This signal is intended to receive an input one clock cycle in duration, otherwise it will continue to increase the fleet count. This is accomplished by sending signal from a one-pulse circuit generated by the Add Fleet button.

- RESET

The RESET input will set the time to 6:05. This is done in response to a postponement or a general recall signal from the user. The sequence controller sends the RESET signal for one clock cycle. If the fleet count indicated by the ITEM output is zero, it will be incremented by 1. This assumes the signal was sent to recall the last fleet to be started and at least one fleet is required for the start button to be active.

- EN

The EN signal enables counting in the timer. As long as the EN signal is high, the counter will decrement, until all digits are zero. The EN signal also enables the output of the ALERT and

- CLK

The CLK signal is the 1Hz clock used by all other components in this system. For testing, the 2 Hz low frequency clock on the PLDT-3 is used

The outputs for the timer are:

- ITEM

This output is a four-bit standard logic vector that is sent to a BCD to seven-segment converter. This output represents the number of fleets waiting to start.

- MIN

This output is a four-bit standard logic vector that is sent to a BCD to seven-segment converter. This output represents the number of minutes remaining in the current starting sequence. The MIN signal is also used to determine if the ALERT and HORN signals are output.

- SECTEN

This output is a four-bit standard logic vector that is sent to a BCD to seven-segment converter. This output represents the tens digit of the number of seconds remaining. This signal is used to determine the ALERT and HORN signals.

- SECOND

This output is a four-bit standard logic vector that is sent to a BCD to seven-segment converter. This output represents the ones digit of the seconds remaining. This signal is used to determine the ALERT and HORN signals.

- ALERT

The purpose of the ALERT signal is to warn the Race committee that an event is imminent and they must be prepared to raise or lower a flag. The signal is sent during the five-second period before the flag is to be raised or lowered. It is an intermittent signal, high for one half of a clock cycle and low the other half. It is sent during the last five seconds of minutes 6,5,4,1 and 0 in the sequence.

- HORN

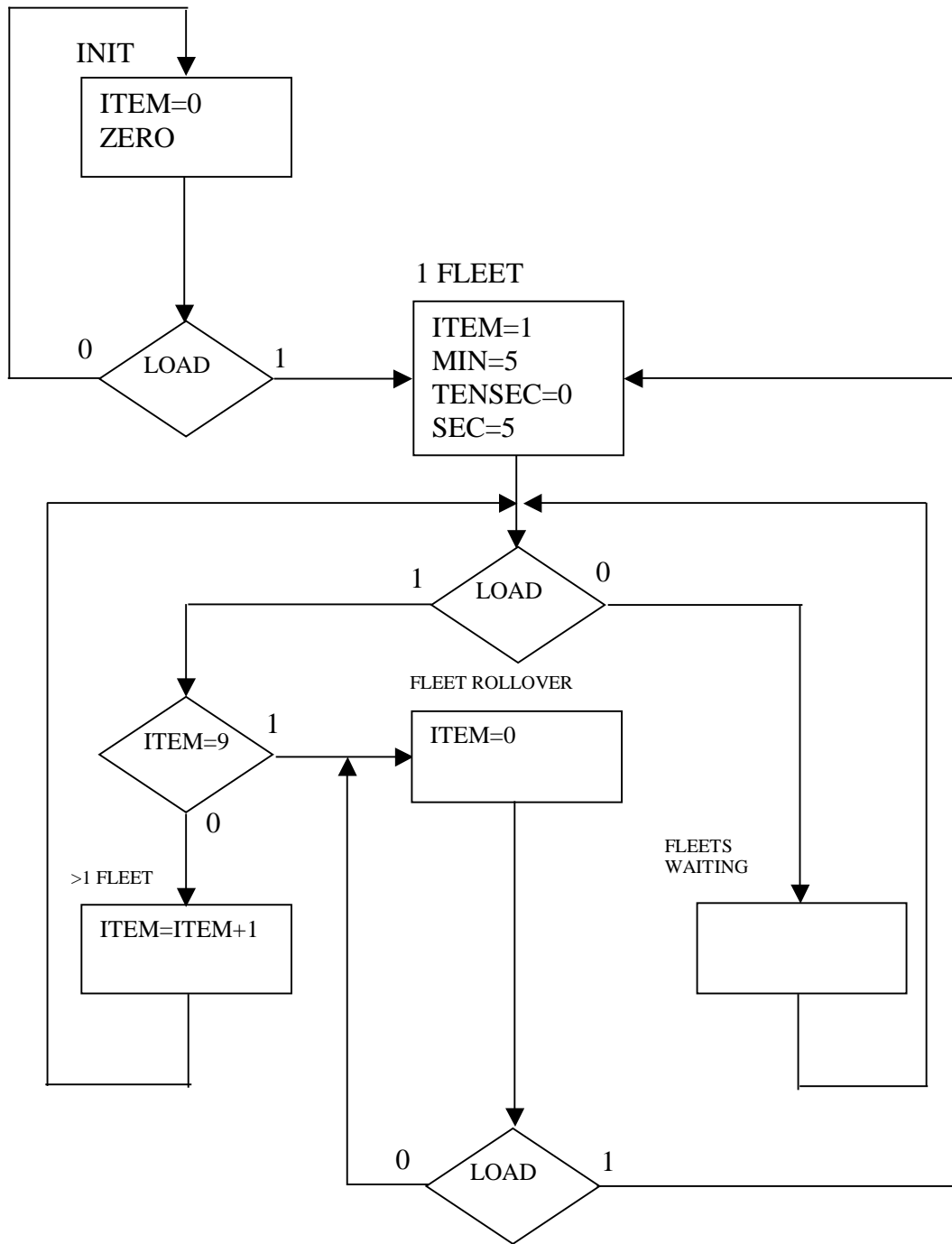
The horn signal generated from the timer the end of minutes 6,5,4,1, and 0 The horn signal is to be sent to the control system of a an air horn or other loud audible device. The horn signal coincides with the visual flag signals manually made by the race committee.

The equations and logic diagram to decode the signals for the ALERT and HORN signals are presented in Figure 12.

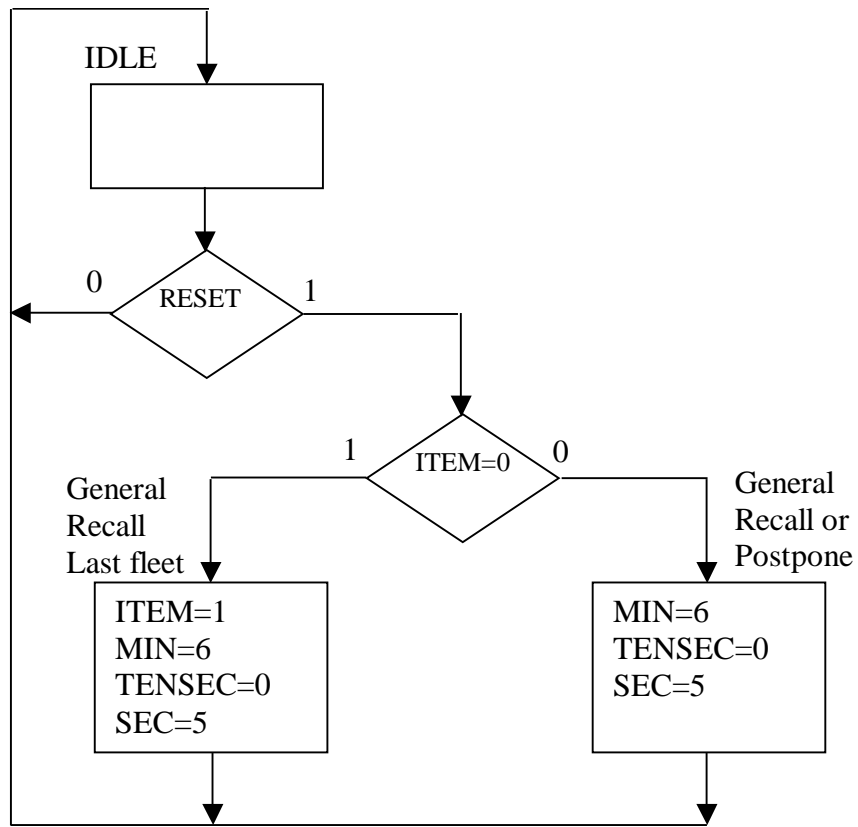
- ZERO

The ZERO signal is sent when the system has completed all the timing and there are no more fleets remaining. This signal is sent to the sequence control unit, which then inhibits the enable signal to the timer.

The ASM charts for the LOAD, RESET, and ENABLE signals are presented in Figure 9, Figure 10 and Figure 11 respectively.



**Figure 9: Timer LOAD signal ASM CHART**



**Figure 10: RESET Signal ASM CHART**

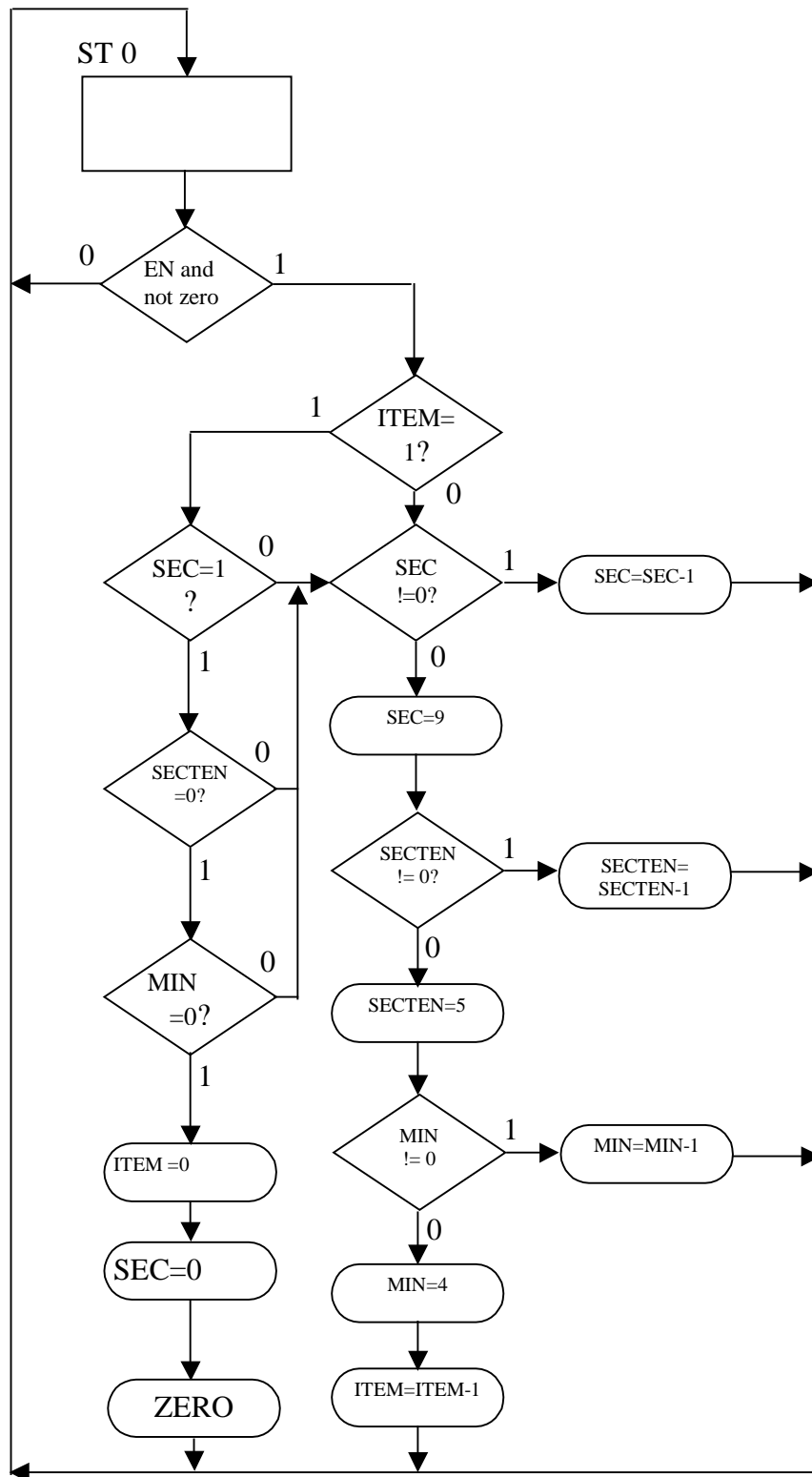
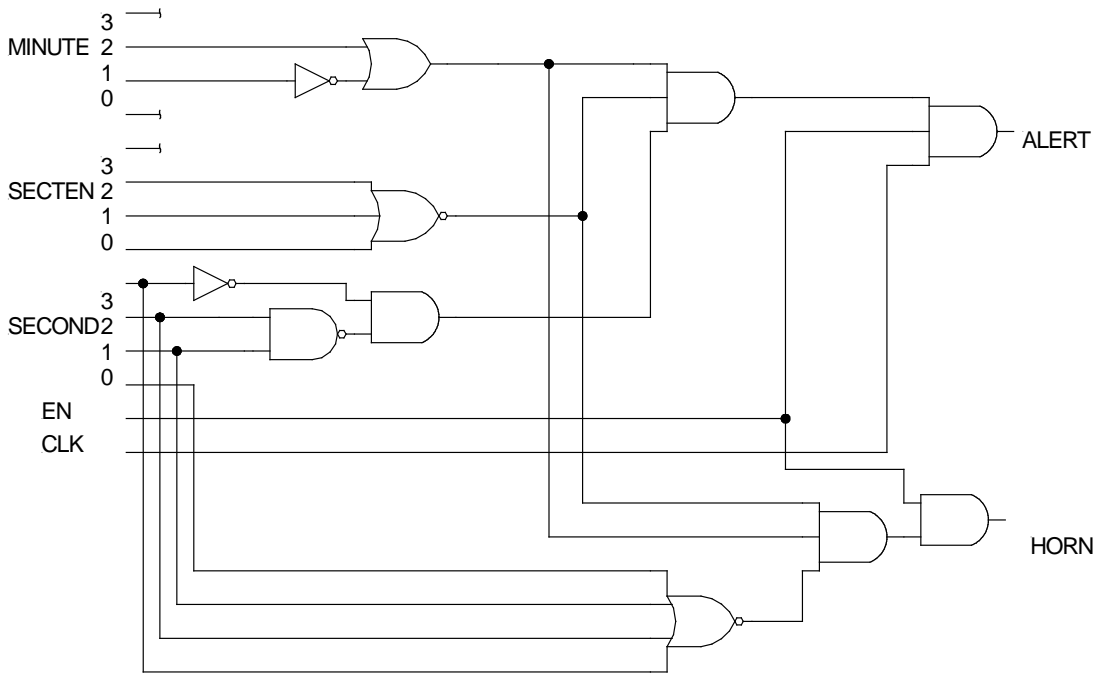


Figure 11: Enable signal ASM chart



$$ALERT = (\overline{MINUTE1} + MINUTE2)$$

$$(\overline{SECTEN2 + SECTEN1 + SECTEN0})(\overline{SECOND2 + SECOND1})\overline{SECOND3}(EN)(CLK)$$

$$HORN = (\overline{MINUTE1} + MINUTE2)$$

$$(\overline{SECTEN2 + SECTEN1 + SECTEN0})(\overline{SECOND3 + SECOND2 + SECOND1 + SECOND0})(EN)$$

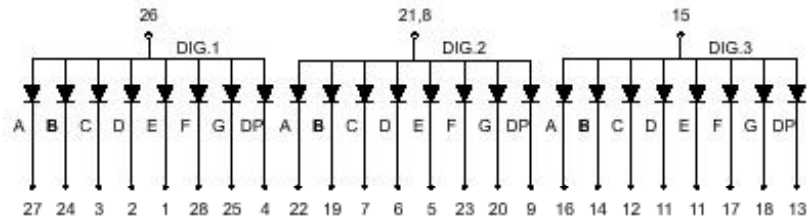
**Figure 12: Horn and Alert Decoder**

*BCD to Seven segment Decoders*

The BCD to seven segment decoders are active-low converters. This was necessary to provide low signals to the numeric displays. The anodes of the segments in each display are connected to a common pin, and to illuminate the segments, active low signals must be applied to the cathodes. The internal wiring of the numeric displays is presented in Figure 13.

# INTERNAL CIRCUIT DIAGRAM

FYT-5631Dx



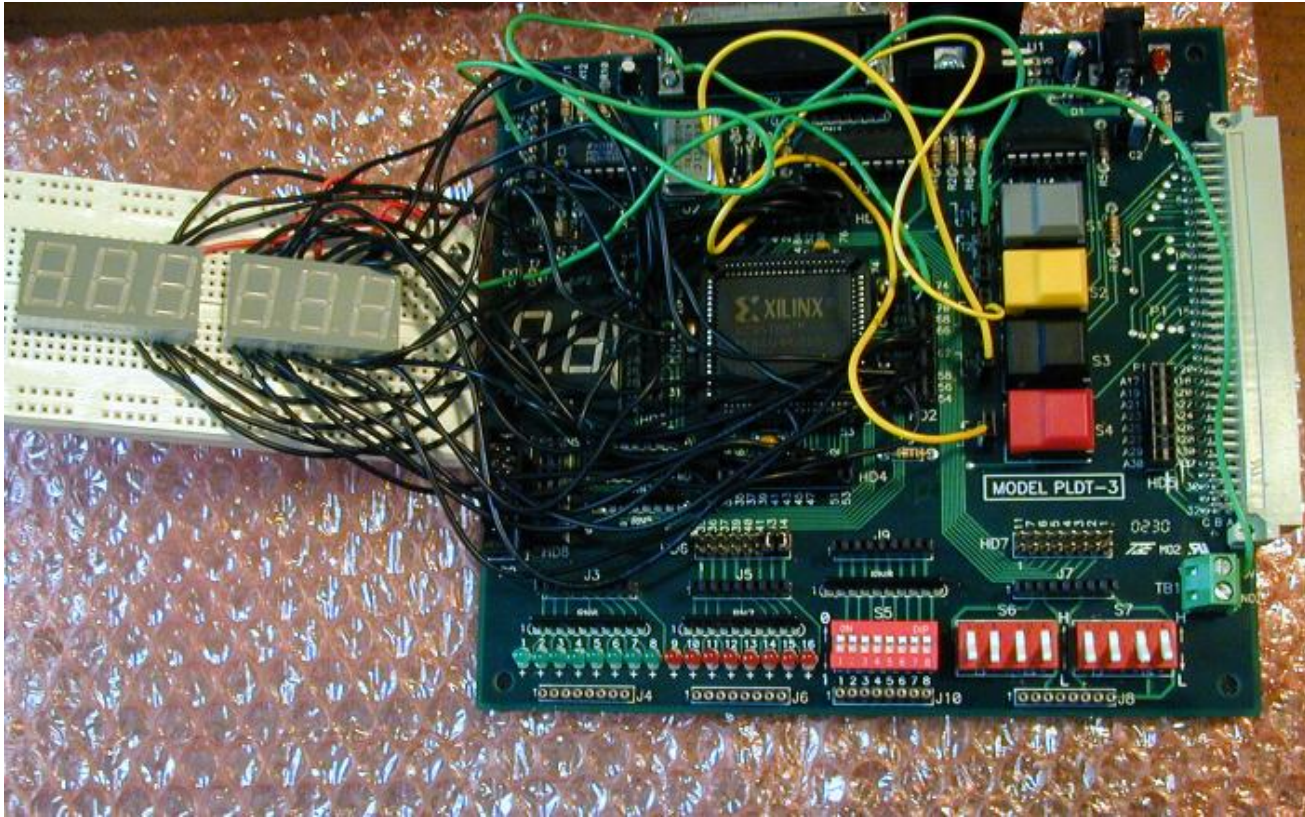
**Figure 13: Numeric LED Display Internal Circuit Diagram**

## TIMING CONSTRAINTS

The system is to operate on a 1 Hz clock, which must provide the same accuracy as today's digital watches. When a postponement signal is sent, the sequence controller must send the RESET signal for one clock cycle and inhibit the enable signal simultaneously to prevent the timer from continuing to count from the 6:05 time point.

## EQUIPMENT REQUIREMENTS

For this prototype, a RSR Inc. PLDT-3 CPLD trainer board is used, augmented with two Ningbo Foryard Optoelectronics Company's FYT 5631D three digit numeric LED displays on an external breadboard. A photograph of the system is shown in Figure 14.



**Figure 14: Prototype System using CPLD and LED displays**

## TIME SCHEDULE

SPECIFICATION - 3/30/2005

PROTOTYPE - 4/19/2005

DEMONSTRATION- 4/21/2005

FINAL PRESENTATION. 5-5/2005

## TESTING

Testing was performed using the low frequency clock on the PLDT-3 board. This provided an approximately 2Hz signal which sped testing.

The first stage of testing was to simulate the behavior of each module using the SIM VHDL tool from Green Mountain Computing. After debugging and testing of each module was completed, tests of the One pulse, sequence controller, timer and horn driver modules in conjunction with one another were conducted, also using SIM VHDL. Finally, testing of all modules connected, including the BCD-to-seven segment decoders was conducted by loading all modules into the fully augmented CPLD, as shown in Figure 14.

Testing in the CPLD was done by trying to start the system with no fleets, by starting with one fleet and allowing it to count down to zero, and then by starting the system with two or more fleets. Tests of the Postponement/General Recall function was done by sending that signal at different times in the sequence and after the sequence was complete. Although the manual horn button is a trivial one to implement pressing it at differing times to ensure operation tested its functionality. After debugging, the system seemed to operate properly and was demonstrated. However, there may be other errors or bugs in the system. A full test at the real 1Hz clock speed under conditions similar those expected to be encountered should be conducted. After this, a field unit could be fabricated and testing of the system using audible signals should be done. After completing this testing, the system could be placed into use starting with low importance regattas, in the local area. As the system matures and other errors and bugs are worked out, the follow on versions of this system could eventually be used in almost any fleet regatta in the world.

#### FUTURE WORK

One additional change to this system would be the ability to suspend sequencing between fleets. In other words, instead of the system continuing to count when it reaches zero, it would stop until the start button is pressed again. This would require an additional toggle switch, but would allow the race committee to select between so-called “rolling starts” which is what this system implements, or “non-rolling” starts.

A second modification would be to allow different intervals of time between the four-minute preparatory signal and the one-minute signal. For example, the Clear Lake Racing Association during their Wednesday night races uses a three-minute period between the warning signal and the start.

I hope to someday build this system using SSI and MSI components and to test it on the water. To do this I would need an accurate clock signal to drive the system, and the appropriate IC's, or a system with a CPLD and that can be used on the field.

A power supply would need to be selected. If batteries were to be used, an analysis of the power consumption would be required to correctly size the batteries. It may also be necessary to use lower power IC's to extend the time that the system can function. Lower power display systems may also be selected instead of the LED displays. Another possible power source is 12V DC power available on many boats. If this were to be used, a dc-dc converter circuit to condition the power would be necessary.

Design of the printed circuit board will be necessary as well as a design for the housing to resist the corrosive effects of the marine environment.

Selection of a suitable buzzer or sound-producing device for the ALERT signal would need to be selected.

For the HORN signal, two options are apparent at this time. The first would be to design a dedicated horn system with its own power source and a switching device controlled by this system to operate the horn. The second option would be to find a means to interface the system with an existing horn system integral to the boat. This might require amplifying or conditioning the horn signal from this

system.

One area of concern that has arisen in my mind is the performance of the buttons when using a 1Hz clock. It is possible that when driving the system at that frequency the pushbuttons may not respond quickly enough for the user. The user may have to depress the button longer than he or she may feel natural. If this proves true, I think that introducing a second faster clock (perhaps 60 Hz) for the Sequence controller, and one pulse circuit may solve that problem, however it would be necessary to ensure the outputs of those circuits are synchronous with the timer circuit. This would involve more investigation, design and testing.

#### COMMENTS AND CONCLUSIONS

This project was challenging first from that standpoint that I had to consider what how best to make use of the available resources on the PLDT-3. At first I was concerned about the number of outputs needed to go to the Numeric display, but realized that I could use the pins that drive the two numeric LED displays as well. I had originally thought of using the two onboard digits, but when I realized that the off board displays available required low -level inputs I thought this would complicate matters and I decided to go entirely off the PLDT-3 for the numeric display. If I had been forced to use the on-board display as well as an off board display, I could have redesigned the BCD-to-seven segment decoder to have both high level and low level outputs and to use the desired output for the digit in question.

During design, I was glad to be able to exercise the techniques learned this and the previous semester to design the combinational and sequential circuits required for this project. Additionally, using VHDL allowed me to cement my knowledge of that language more firmly in my mind.

Testing went more easily than I had expected. I believe this was in part to testing the individual modules separately before instantiating them in one structural module and performing an integrated test.

Another issue I encountered was errors I received during the debugging process stating I received a "bad synchronous description" error. I learned that CLK'events must be in the top most level of an "If" statement in a process, and cannot be embedded in "if" statements at lower levels.

Overall, this project was an excellent opportunity to fully exercise the skills and knowledge I've learned this semester.

## REFERENCES

C.H. Roth, Jr., *Fundamentals of Logic Design, 5<sup>th</sup> Ed.*, Belmont ,CA: Brooks/Cole-Thomson Learning, 2004

M.M.Mano and C.R. Kime, *Logic and Computer Design Fundamentals, 3<sup>rd</sup> Ed.* Upper Saddle River, NJ: Pearson Prentice Hall, 2004

RSR Electronics, Inc. *PLDT-3 Trainer Manual*, 2004

Texas Instruments, *TTL Logic Data Book, 1988 edition*, Dallas, Texas 2004

Ningbo Foryard Optoelectronics, Co, Ltd. *Three digit display data sheet*

[http://www.foryard.com/fy\\_products/PDF\\_FILE/30.PDF](http://www.foryard.com/fy_products/PDF_FILE/30.PDF)

Xilinx Answers Data base:

[http://www.xilinx.com/xlnx/xil\\_ans\\_display.jsp?iLanguageID=1&iCountryID=1&getPagePath=14047](http://www.xilinx.com/xlnx/xil_ans_display.jsp?iLanguageID=1&iCountryID=1&getPagePath=14047)

Brain, M : *How Digital Clocks work*: <http://home.howstuffworks.com/digital-clock.htm>

Rose, D. et. al., *The Racing Rules of Sailing 2005-2008 Including US Sailing Prescriptions*, United States Sailing Association, Portsmouth, RI, 2005

## ACKNOWLEDGEMENTS

I wish to acknowledge Mr. Dan Morgan for sharing with me the source location on the Internet for the three-digit display data sheet for the FYT 5631D display and also for reminding me about the Xilinx answers database.

I also would like to thank Dr. Eldon Husband, Ph.D., P.E. for his efforts in instructing me in Digital System Design this semester.



## Appendix A- VHDL Code for the SRSA

*-- Horn Driver VHDL module*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity horndr is
    Port ( CLK : in std_logic;
          PP : in std_logic;
          horn : out std_logic);
end horndr;

architecture Dataflow of horndr is

    signal Q1: std_logic := '0';
    signal Q0: std_logic := '0';
begin
    process(CLK)
    begin
        if (CLK'event and CLK = '1') then
            Q1 <= Q0 or (PP and Q1) ;
            Q0 <= (PP and not(Q1)) or ((not Q1) and Q0);
            horn <= (PP and (not Q1) and (not Q0)) or (Q1 and Q0);

        end if;
    end process;
end Dataflow;
```

*-- One pulse Circuit VHDL module*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity one_pulse is
    Port ( CLK : in std_logic;
          X: in std_logic;
          Z: out std_logic);
end one_pulse;
```

```

architecture dataflow of one_pulse is
signal Qint: std_logic :='0';

begin
  process(CLK)
  begin
    if (CLK'event and CLK='1') then
      Qint <= X;
      Z <= X and not Qint;
    end if;
  end process;
end dataflow;

--Sequence Controller
se IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity seqctrl is
  Port ( CLK : in std_logic;
        PP: in std_logic;
        ST: in std_logic;
        STOP: in std_logic;
        EN: out std_logic;
        RESET: out std_logic);
end seqctrl;

architecture dataflow of seqctrl is
signal Q1: std_logic :='0';
signal Q0: std_logic :='0';

begin
  process(CLK)
  begin
    if (CLK'event and CLK='1') then
      Q1<= (ST or Q1 )and (not STOP)and (not PP)and (not Q0);

      Q0<= (PP)and ((not ST) or (not STOP) or (not Q1));
      EN <= (ST or Q1 )and (not STOP)and (not PP)and (not Q0) ;

      RESET <= (PP)and (not Q0) and ((not ST) or (not STOP)or (not
Q1)) ;
    end if;
  end process;
end dataflow;

```

*--Timer Circuit VHDL module*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity timer is
    Port ( CLK : in std_logic;
          EN : in std_logic;
          LOAD : in std_logic;
          RESET: in std_logic;
          ITEM : out std_logic_vector(3 downto 0);
          MIN : out std_logic_vector(3 downto 0);
          TENSEC : out std_logic_vector(3 downto 0);
          SEC : out std_logic_vector(3 downto 0);
          ALERT : out std_logic;
          HORN : out std_logic;
          ZERO : out std_logic);
end timer;

architecture Behavioral of timer is

    signal MINint : std_logic_vector(3 downto 0) := "0000";
    signal TSECint : std_logic_vector(3 downto 0) := "0000";
    signal SECint : std_logic_vector(3 downto 0) := "0000";
    signal ITEMint : std_logic_vector(3 downto 0) := "0000";
    signal ZVECT: std_logic_vector(3 downto 0);
    signal Zint : std_logic;
begin

    process(CLK, RESET)
    begin
        if (CLK'event and CLK = '1') then
            if LOAD = '1' then
                if ITEMint = "1001" then
                    ITEMint<="0000";
                elsif ITEMint = "0000" then ITEMint
                    <= ITEMint + "0001";
                    MINint <="0101";
                    TSECint <="0000";
                    SECint <="0101";
                    else ITEMint<=ITEMint +"0001";
                    end if;
            end if;
            if (EN = '1' and Zint = '0') then
                if ITEMint="0001" and SECint = "0001"
                    and TSECint<="0000" and MINint<="0000" then
```

```

ITEMint<="0000"; SECINT<="0000";
else
  if SECint /= "0000" then SECint <=
SECint - "0001";
  else
    SECint <= "1001";
    if TSECint /= "0000" then
TSECint <=TSECint - "0001";
      else TSECint <= "0101";
        if MINint /= "0000"
          else MINint
            end if;
        end if;
      end if;
    end if;
  end if;
end if;

if (RESET = '1') then
  MINint <="0110";
  TSECint <="0000";
  SECint <="0101";
  if ITEMint = "0000" then
    ITEMint <= ITEMint + "0001";
  end if;
end if;
end if;
end process;

ITEM<=ITEMint;
MIN<=MINint;
TENSEC<=TSECint;
SEC<=SECint;
ALERT <= CLK when EN='1' and (((MINint >= "0100" and MINint
<= "0110")
  or (MINint <= "0001")) and TSECint =
"0000" and SECint <="0101")) else '0';
HORN <= '1' when EN='1' and (((MINint >= "0100" and MINint
<= "0110")
  or (MINint <= "0001")) and TSECint =
"0000" and SECint = "0000")) else '0';
ZVECT<= (not ITEMint) and (not MINint) and (not TSECint) and
(not SECint);
Zint <= ZVECT(3) and ZVECT(2) and ZVECT(1) and ZVECT(0);
ZERO<=Zint;
end Behavioral;

```

--BCD to seven Segment Decoder

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

```

```

use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bcdto7seg is
    Port ( D : in std_logic_vector(3 downto 0);
          Q : out std_logic_vector(1 to 7));
end bcdto7seg;

architecture Behavioral of bcdto7seg is
    signal Qint : std_logic_vector(1 to 7);
begin
    Qint <=
        "1111110" when D ="0000"
        else "0110000" when D ="0001"
        else "1101101" when D ="0010"
        else "1111001" when D ="0011"
        else "0110011" when D ="0100"
        else "1011011" when D ="0101"
        else "1011111" when D ="0110"
        else "1110000" when D ="0111"
        else "1111111" when D ="1000"
        else "1111011" when D ="1001"
        else "1110111" when D ="1010"
        else "0011111" when D ="1011"
        else "1001110" when D ="1100"
        else "0111101" when D ="1101"
        else "1001111" when D ="1110"
        else "1000111" when D ="1111";
    Q <= not Qint;
end Behavioral;

```

```

-- Sailboat Race Starting Aid VHDL module
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following lines to use the declarations that are
-- provided for instantiating Xilinx primitive components.
--library UNISIM;
--use UNISIM.VComponents.all;

entity SSRS is
    Port ( PP : in std_logic;
          CLK : in std_logic;
          ADD : in std_logic;
          START : in std_logic;
          MHORN : in std_logic;
          FLEET : out std_logic_vector(1 to 7);
          MINUTE : out std_logic_vector(1 to 7);
          SECTENS : out std_logic_vector(1 to 7);
          SECONDS : out std_logic_vector(1 to 7);
          ALERT : out std_logic;
          HORN : out std_logic);
end SSRS;

architecture Structural of SSRS is
    component timer is
        Port ( CLK : in std_logic;
              EN : in std_logic;
              LOAD : in std_logic;
              RESET: in std_logic;
              ITEM : out std_logic_vector(3 downto 0);
              MIN : out std_logic_vector(3 downto 0);
              TENSEC : out std_logic_vector(3 downto 0);
              SEC : out std_logic_vector(3 downto 0);
              ALERT : out std_logic;
              HORN : out std_logic;
              ZERO : out std_logic);
    end component;

    component bcdto7seg is
        Port ( D : in std_logic_vector(3 downto 0);
              Q : out std_logic_vector(1 to 7));
    end component;

    component horndr is
        Port ( CLK : in std_logic;
              PP : in std_logic;
              horn : out std_logic);
    end component;

    component seqctrl is

```

```

    Port ( CLK : in std_logic;
           PP: in std_logic;
           ST: in std_logic;
           STOP: in std_logic;
           EN: out std_logic;
           RESET: out std_logic);
end component;

component one_pulse is
    Port ( CLK : in std_logic;
          X: in std_logic;
          Z: out std_logic);
end component;

signal FLTINT, MININT, STINT, SECINT : std_logic_vector(3 downto 0);
signal STOPSIG, CNTSIG, PPSIG, FLTPLSIG, HSIG1, HSIG2, STINV:
std_logic;
begin
STINV <= not START; --start is on S3, inverted pb switch

CTR1: timer port map(CLK, CNTSIG, FLTPLSIG, PPSIG, FLTINT, MININT,
STINT, SECINT, ALERT, HSIG2, STOPSIG);
PUL1: one_pulse port map (CLK, ADD, FLTPLSIG);
SEQ1: seqctrl port map(CLK, PP, STINV, STOPSIG, CNTSIG, PPSIG);
HORNDR1: horndr port map(CLK, PPSIG, HSIG1);
DISPFLT: bcdto7seg port map(FLTINT, FLEET);
DISPMIN: bcdto7seg port map(MININT,MINUTE);
DISPSEC10: bcdto7seg port map(STINT, SECTENS);
DISPSEC: bcdto7seg port map(SECINT,SECONDS);
HORN <= HSIG1 or HSIG2 or MHORN;
end Structural;

```