

Volume

7

A-WIT TECHNOLOGIES INC.

... a passion for execution ...

Digital Logic Reference Guide Manual

Version 1.0

A-WIT TECHNOLOGIES INC.

Digital Logic Reference Guide Manual (Digital Logic Design 101)

© A-WIT Technologies Inc.
Phone (800) 985-AWIT • Fax (800) 985-2948

Table of Contents

Audience	1	C Stamp Logic Personality 2	60
Registering Your C Stamp or C Stamp		C Stamp Logic Personality 3	61
Related Product	1	C Stamp Logic Personality 4	62
Introduction to the C Stamp	2	C Stamp Logic Personality 5	63
Introduction to the CS31X000 Board of Learning (BOL)	3	Terms and Conditions	66
Introduction to the CS21X003 Digital Logic Design Training Kits	4		
Notices	10		
Getting Support	11		
Installing the Software	11		
Creating your First Logic Design	12		
Downloading the Logic Personality	12		
Verifying Your Design	14		
Using the BOL Breadboard	14		
Developing Your Own Projects	14		
Binary Signals	15		
Hexadecimal Numbers	16		
Complements	16		
Binary Logic	17		
Exclusive-OR Function	21		
Combinational Circuit Design	23		
Combinational Logic Project 1	31		
Combinational Logic Project 2	32		
Combinational Logic Project 3	39		
Synchronous Clocked Sequential Circuits	41		
Analysis of Clocked Sequential Circuits	43		
Design of Clocked Sequential Circuits	46		
Sequential Logic Project 1	57		
C Stamp Logic Personality 1	59		

Digital Logic Design 101

This text covers the basics of Digital Logic Design through a series of experimental activities using the A-WIT Technologies, Inc.'s Digital Logic Trainer based on its C Stamp™ microcomputer module. The designs described in this text expose users to the fundamentals of Digital Logic Design by using the exposition of the theory followed by the construction of Digital Logic circuits to achieve a given design goal or set of goals. The activities in the text are projects that introduce a variety of concepts in digital logic design, electricity and electronics, mathematics, and physics. All activities are highly illustrated hands-on presentation of design practices used by engineers in the design of modern digital equipment.

Audience

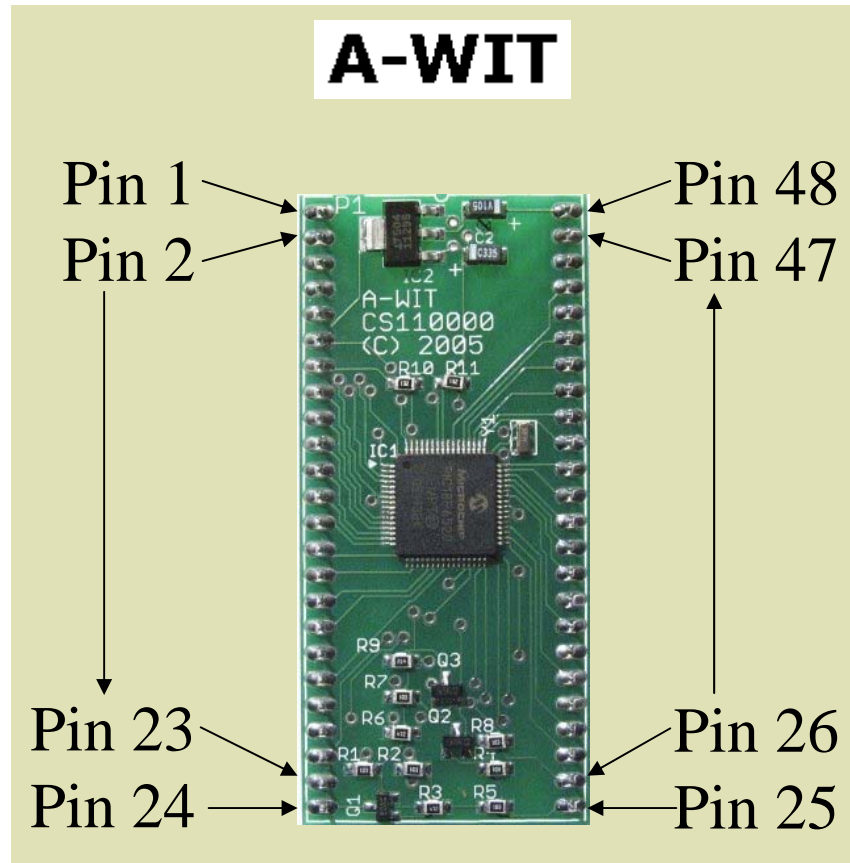
This text is organized so that it can be used by pre-engineering and engineering students, as well as independent learners. Comprehension and problem-solving skills can be tested with the questions, exercises, projects, and solutions provided at the end of each chapter.

Registering Your C Stamp or C Stamp Related Product

At A-WIT Technologies we respect your privacy; however, we do ask you to register your C Stamp or C Stamp related product, so you can receive free of charge product updates. The registration procedure is simple. Just send an e-mail to tech_support@awit.com with the word "REGISTRATION x" in the subject line, where "x" is the product number that you purchased. If you purchased more than one product, send an e-mail for each different product.

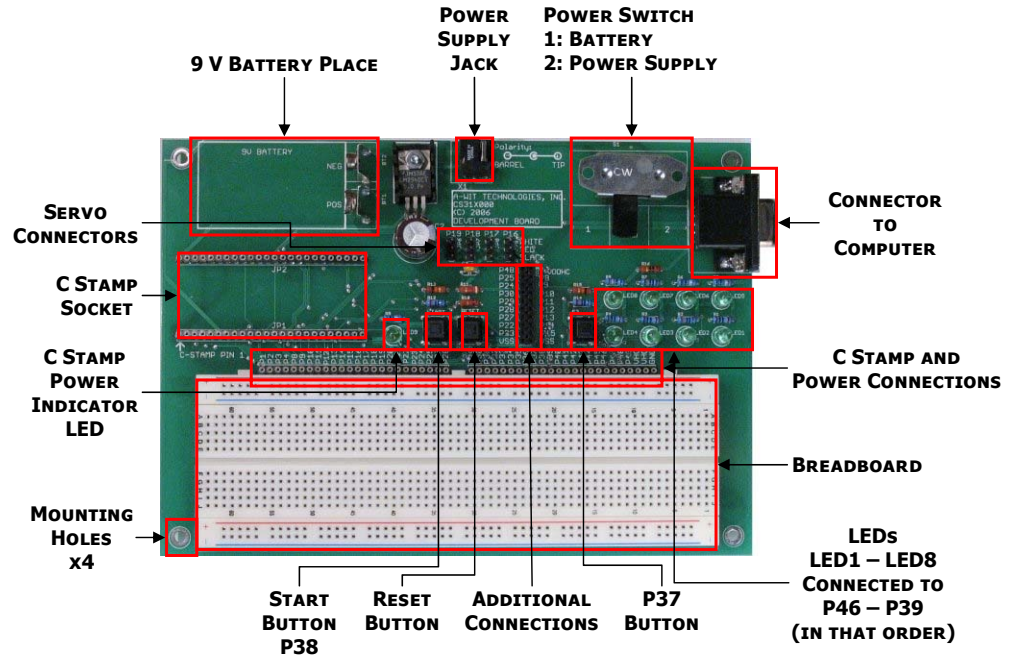
Introduction to the C Stamp

The figure below shows how the pins of the CS110000 C Stamp module are numbered.



Introduction to the CS31X000 Board of Learning (BOL)

The figure below shows the CS310X00 BOL with all features highlighted.



The START Button is pre-connected to Pin 38. When the Button is pushed, Pin 38 is pulled LOW; otherwise, Pin 38 is HIGH. For the Utility Button pre-connected to Pin 37, when the Button is pushed, Pin 37 is pulled LOW; otherwise, Pin 37 is HIGH. Each one of the LED1 - LED8 LED group is pre-connected to Pins 46 - 39 respectively. When a pin is set HIGH, the corresponding LED lights up, and when the pin is set LOW, the corresponding LED turns off. If the user wants to use any of the pre-connected pins for a different purpose, any of the pre-connected components can be ignored.

The C Stamp should be inserted in its corresponding socket with the pins 1 of the C Stamp and the socket aligned. Pin 1 is at the bottom-left corner of the socket in the figure above. Since the C Stamp has so many pins, enough pressure needs to be applied to the C Stamp to push it into the socket until it snaps into place. Press at several points of the C Stamp, but be gentle and careful at the same time. If you ever need to remove the C Stamp from the socket, first make sure that the C Stamp is powered off. Then, use a dull prying or lever like object to pry consecutively the C Stamp incrementally from both sides (left and right in the figure above) until the C

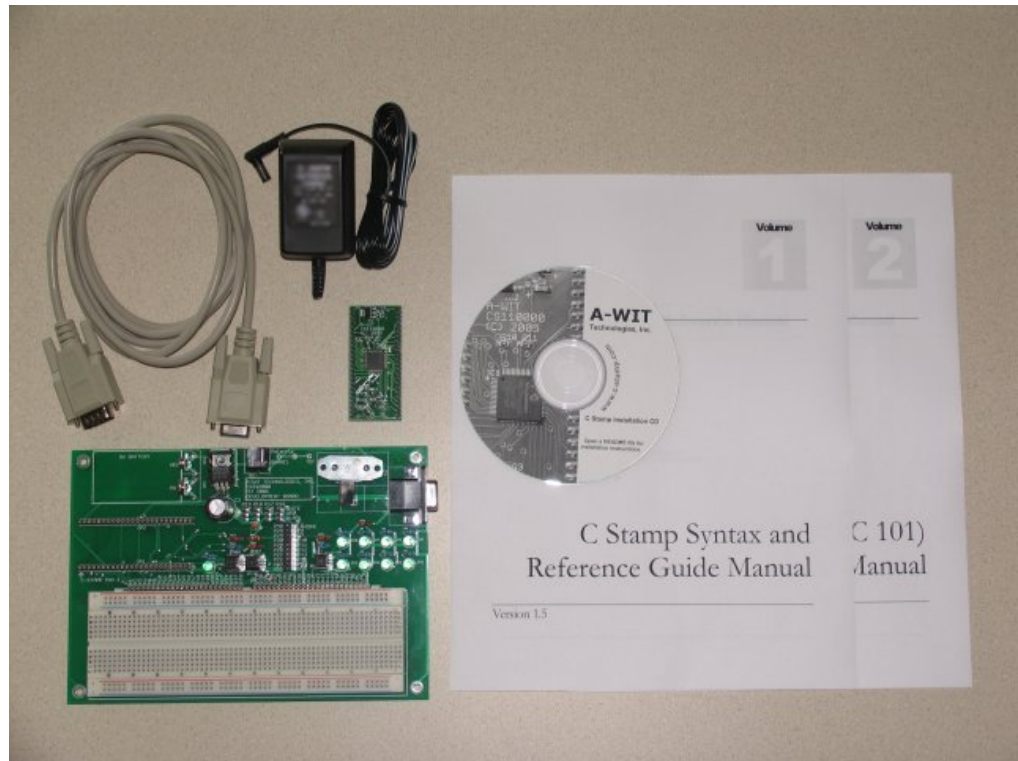
Stamp is released from the socket. Since the C Stamp has so many pins, prying all the way from one side only will likely bend some pins on the opposite side from where the prying is being done.

Introduction to the CS21X003 Digital Logic Design Training Kits

Digital Logic Design! This kit is a powerful Digital Logic Design learning aide. The Kit includes the C Stamp module, the Board of Learning, PC cable, and power supply. All necessary documentation, projects, and instructions written by College professors are included in the Kit CD. Simply download a Logic Personality to the board, and you are ready to implement your digital logic design.

This Digital Logic Design Kit is unique in the sense that students and autodidactic users alike just download the necessary logic personality to the board to implement their design. When a different design calls for a different logic personality all that is necessary is a new download, and a new design can be implemented. Then, if the student or the independent learner is wants to go on to program microcontrollers, the Parts Kit for the Microcontroller Activity Kits (part number CS218000) can be purchased, and a slew of fun and instructive microcontroller programming learning activities can be tackled. That is all that is necessary to move on the next level! This is how the power and versatility of the C Stamp guarantees that learners can expand upon their logic design knowledge with their own ideas, projects, and imagination. Well thought out and pedagogically sound examples developed by College professors make sure that learning Digital Logic Design with this Kit is a powerful and enjoyable experience. The following figures and table show the contents of the different Digital Logic Design Kits.

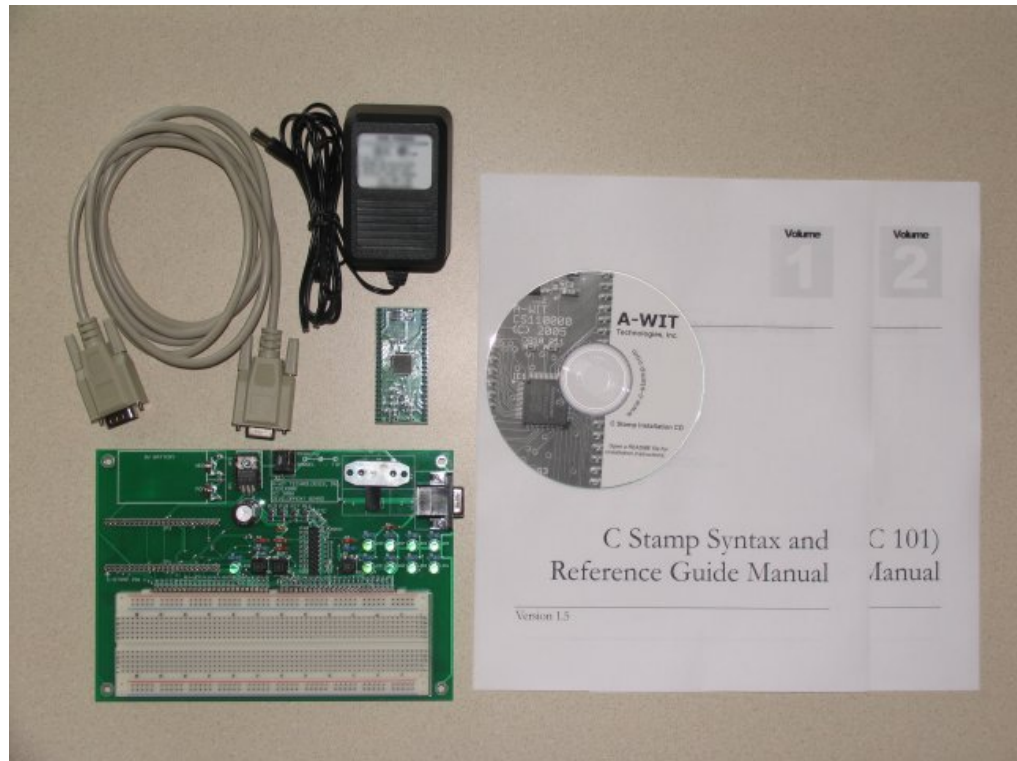
Digital Logic Design Training Kit - Serial Connector - 120V Power Supply (US, etc.):



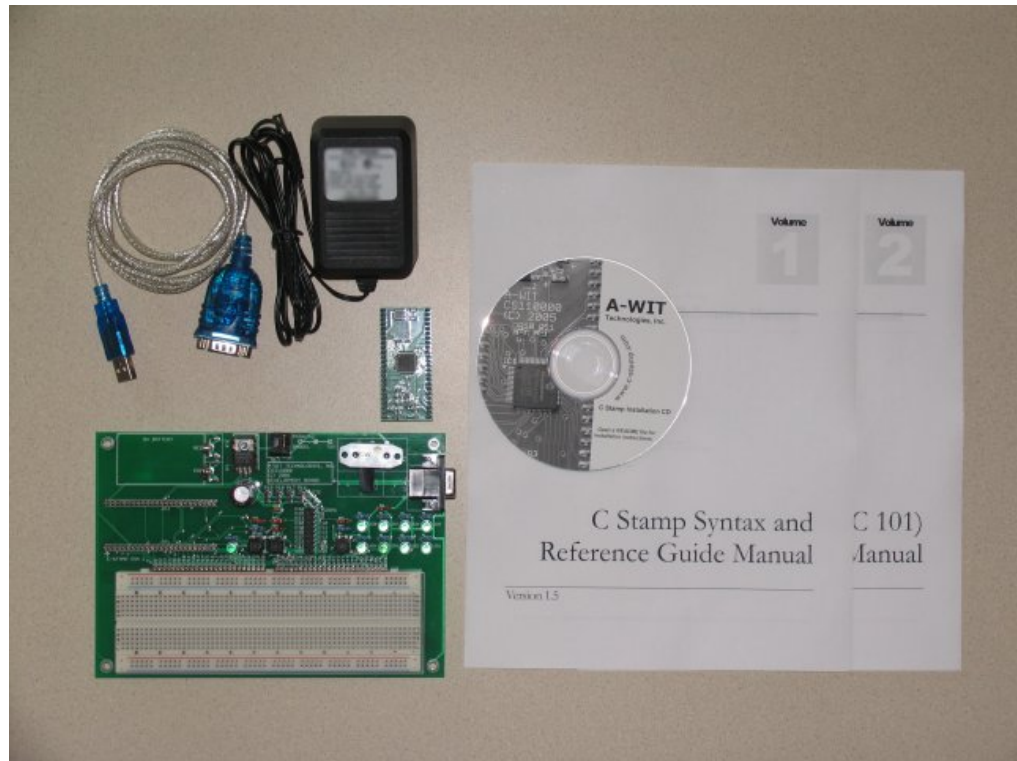
Digital Logic Design Training Kit - USB Connector - 120V Power Supply (US, etc.):



Digital Logic Design Training Kit - Serial Connector - 230V Power Supply (Europe, etc.):



Digital Logic Design Training Kit - USB Connector - 230V Power Supply (Europe, etc.):



Contents description for each kit:

<i>Contents Description</i>	<i>Minimum Quantity</i>
CS110000 C-Stamp	1
Board of Learning (BOL)	1
Power supply for BOL (120V or 230V)	1
PC cable (Serial or USB)	1
Software installation and documentation CD	1
DIP switches	4
7 Segment LED Digit Display (0.56 inch)	1
287 OHM 1/4W $\pm 1\%$ Metal Film Resistor	7
4 cm hook-up wire	20

<i>Contents Description</i>	<i>Minimum Quantity</i>
10 cm hook-up wire	7
17 cm hook-up wire	10

Getting Started

This chapter is a quick start guide to connecting the C Stamp to the PC and downloading Logic Personalities to it. At this point, you should be able to complete the tutorial in this chapter and complete your first logic circuit. The tutorial assumes you have a C Stamp logic design kit and an appropriate connection kit or development board. You will also need a programming cable, power supply, PC running Windows® 2000/XP/Media/Vista, with a quantity of RAM recommended for the OS, sufficient free hard disk drive space for the software installations, CD-ROM drive, Internet access (recommended only), and available port compatible with your programming cable.

Notices

CSTAMP™ and CSTAMP™ Related Hardware Products, Software Products and Documentation are developed and distributed by A-WIT Technologies, Inc. All rights reserved by A-WIT Technologies, Inc. A-WIT SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL A-WIT BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

MPLAB C-18 and MPLAB C-18 Users Guide is reproduced and distributed by A-WIT Technologies, Inc. under license from Microchip Technology Inc. All rights reserved by Microchip Technology Inc. MICROCHIP SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR ANY CLAIM,

DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

Getting Support

If possible, please check the C Stamp website www.c-stamp.com under SUPPORT for any updates to documentation, changes, or notices that may have become available since your Installation CD was produced. If you continue to have any issues for which a solution is not found in the aforementioned website, please e-mail tech_support@a-wit.com for help.

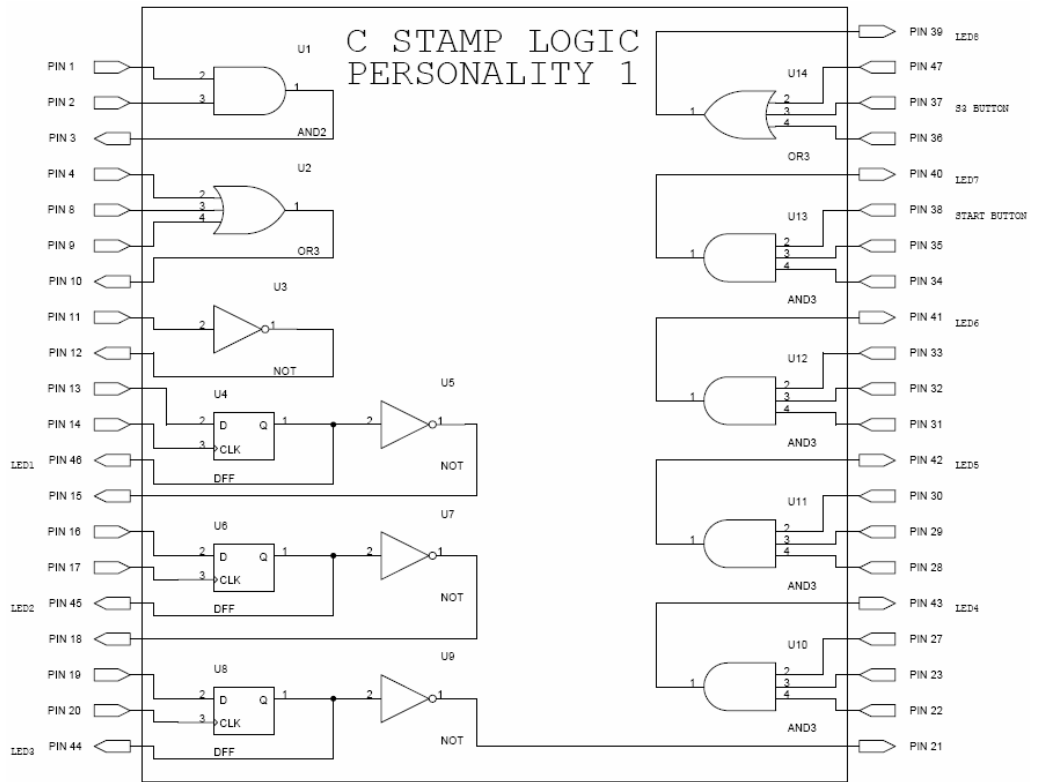
Installing the Software

The first step is to install the logic personalities and the software that you will use to download the logic personalities to the logic trainer board.

Insert your A-WIT provided Installation CD in your CD drive. An installation wizard should run automatically. If the wizard does not run, go to the CD and double click to run the autorun application in the CD. When the wizard application runs, you should see its first screen. Perform the installations in steps 1, 4, 5, 6, and 7 as needed. We also recommend that you perform steps 9 and 10 as well.

Creating your First Logic Design

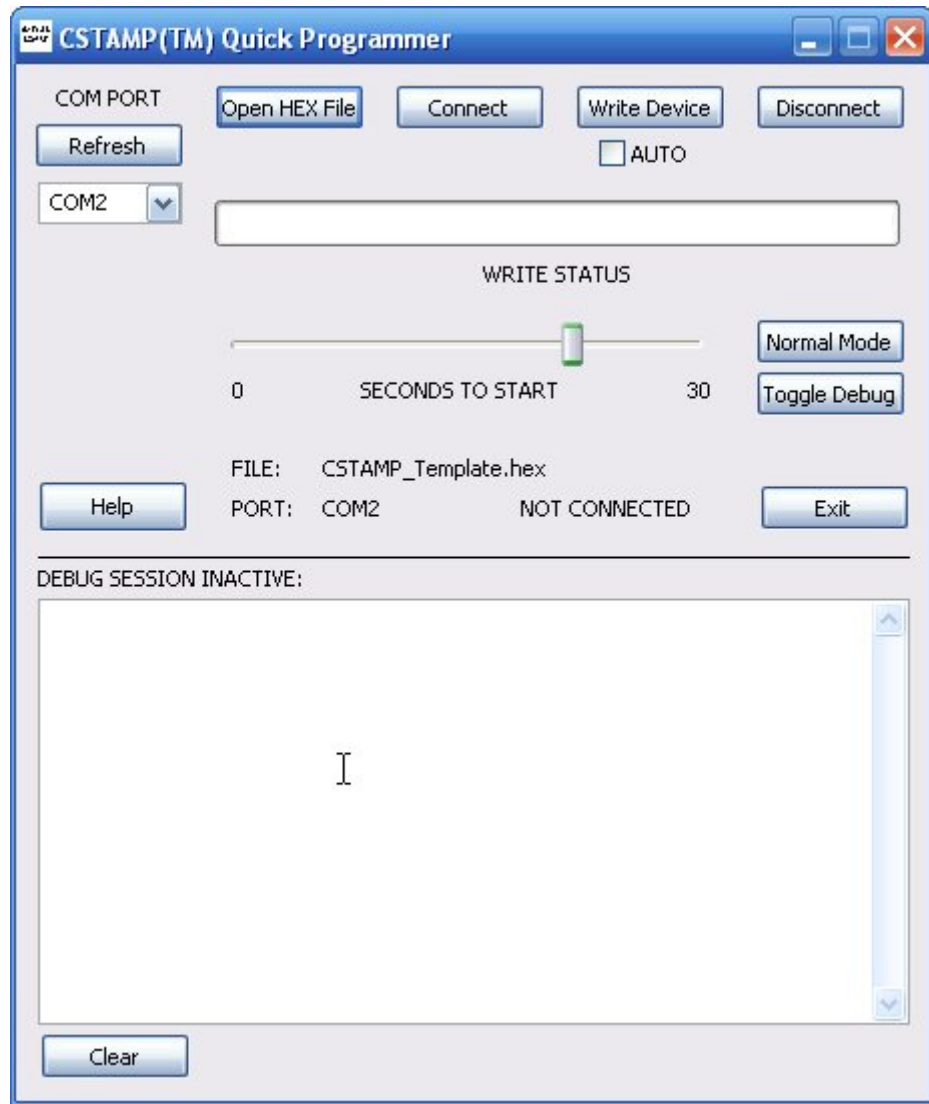
Suppose you want to light up an LED when any one of three inputs to your circuit is a logic HIGH. For this simple example, we will connect two inputs of an OR gate to LOW or GND, and the other input is going to be connected to push button S3. S3 provides a HIGH when it is not pressed and a LOW when pressed, so the output of the OR gate will be HIGH when S3 is not pressed and LOW when it is. The output of the gate will be connected to LED8 in the BOL. Close observation of the C Stamp Logic Personality 1 reveals that this personality has an OR gate that meets the conditions necessary for this design. The logic gate is labeled U14. The connections to S3 and LED8 are already present on the BOL. All we have to do is to connect pins 36 and 47 of the C Stamp to GND. The figure below shows a picture of the C Stamp Logic Personality 1.



Downloading the Logic Personality

Insert the C Stamp in the BOL, power up your BOL, and connect the BOL to the PC with the provided cable. Upon power up, the C Stamp will be in RESET/BOOT/DOWNLOAD mode. To go back to this mode at any time, just

push and let go of the RESET button. Then open the A-WIT C Stamp Quick Programmer application shown in the next figure.



The first step is to choose the serial port that you are using from the drop-down menu. Then click on “Refresh”, so that the program registers your selection. Your selection should show in the status area of the program next to “PORT”. Then click on “Open HEX File” and load/select the HEX file LOGIC_PERSONALITY_01 in the directory C:\A-WIT\CSTAMP_Logic. The status should indicate that the file has been loaded successfully. This is what will be downloaded to the C Stamp. Then click on “Connect”, and the PC will be connected to the BOL, and the status area should indicate so. To download the logic personality to the C Stamp, just click on “Write Device”, and you should see the progress bar after a few seconds, as the HEX file is

downloaded. At this point, you can click on “Disconnect” to disconnect the PC from the BOL, disconnect the serial cable from both the PC and the BOL, and start the logic manually at the BOL. To do this just push and let go of the RESET button while pushing the START button. Then you can let go of the START button. Alternatively, you can click on “Normal Mode” to start the logic from the PC. This will also disconnect the program/PC from the BOL. Then you can disconnect the serial cable from the PC and the BOL. You can also instruct the CSTAMP™ Quick Programmer to wait several seconds before starting the logic from the PC and disconnecting by adjusting the “SECONDS TO START” slide. This feature is useful in case you want to keep the PC connected with the serial cable, but need time to manually set up something in a circuit that you have built. If this is not the case it can just be left at the default of “0”, and the logic will be started from the PC right away. After you click “Normal Mode” and the logic is started, the CSTAMP™ Quick Programmer will not be communicating with the C Stamp any longer, so if you want to reconnect, you must click on “Connect” again.

Verifying Your Design

Now that you have the logic operational, you should connect pins 36 and 47 to GND with wires. Then you will be able to demonstrate that when S3 is not pressed, LED8 is on, and when S3 is pressed, LED8 is off.

Using the BOL Breadboard

To build circuits that require additional components, the μ C101 BOL provides a spacious full size solderless breadboard. Breadboards are simply a matrix of inserts that allow you to connect electrical circuits a simple manner with the supplied jumper wires without the need of solder. At the top and bottom of each panel there are rows of connections that are used for power. These are denoted with a solid red or blue line, and each row corresponding to a color has all its inserts connected together. The rest of the inserts are labeled in rows (A-J) and columns (1-63). For each column, each group of rows (A-E) or (F-J) are connected together inside the breadboard, and are used to connect different components by inserting terminals into the inserts of that group. If you run out of inserts in a group, then jump a connection to an unused group and keep connecting components.

Developing Your Own Projects

Now that you have successfully developed your design, it is easy to move on to more complex and elaborate projects and circuits of your own or the ones described in this manual: Digital Logic Reference Guide Manual (Digital Logic Design 101).

Combinational Logic

This chapter describes the Combinational Logic Design projects and activities that can be done with the C Stamp and the μ C101 BOL. The material in this chapter is written at a very fundamental level, so depending on the present level of expertise, not all the material will have to be reviewed by all readers. This chapter also provides background in combinational logic design that leads to the successful completion and understanding of the design projects.

Binary Signals

We, humans, use an intuitive number system based on the number 10. This system is intuitive to us. It is speculated by anthropologists that the base 10 as our counting system came from the fact that we have 10 fingers, and people first started to count with them. The earlier roman numerals have a base of 5; perhaps because people were counting with only one hand first, before they realized that they could use both hands. Whatever the reason, the fact is that number systems need to be based on the feasibility of the entity using them. For digital systems and circuits this base is the number 2, so they are called binary signals or number. This stems from the fact that it is easy, reliable, and convenient to use two voltage levels to signal two unique binary numbers: the numbers 1 and 0. Therefore, we say that computers, which are sophisticated digital systems, use 1's and 0's. Let us draw a parallel between our familiar base 10 number system (decimal) and binary. Consider the decimal number 7392. This is

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0,$$

but only the coefficients are written.

In general, a number is written as

$$\cdots a_5 a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \cdots$$

The a 's are the coefficients and the subscripts are the powers of an implied number system base r . For decimal $r = 10$, and for binary, $r = 2$. Then the value of the number is given by the following sum.

$$\cdots + a_3 \times r^3 + a_2 \times r^2 + a_1 \times r^1 + a_0 \times r^0 + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + a_{-3} \times r^{-3} + \cdots$$

Then, what is the value of the binary number 11010.11? Remember that only 1's and 0's are used in the binary number representation. Let us calculate it using the concepts outlined above. Notice that we denote the base of the number as a subscript.

$$\begin{aligned} (11010.11)_2 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ (11010.11)_2 &= 16 + 8 + 0 + 2 + 0 + 0.5 + 0.25 \\ (11010.11)_2 &= 26.75 \end{aligned}$$

At this point of the numerical discussion, we should mention that quantities in digital systems are often predicated with Kilo, Mega, or Giga; as in 1 KB (one Kilo Byte). The prefixes K, M, and G have the following values.

$$\begin{aligned} K &= 2^{10} = 1,024 \\ M &= 2^{20} = 1,048,576 \\ G &= 2^{30} = 1,073,741,824 \end{aligned}$$

This is obviously not the usual thousand, million, or billion we normally use. Therefore

$$\begin{aligned} 2^n &= 2^{n-10} \quad K & | \quad n > 10 \\ &= 2^{n-20} \quad M & | \quad n > 20. \\ &= 2^{n-30} \quad G & | \quad n > 30 \end{aligned}$$

Hexadecimal Numbers

Hexadecimal (hex) and binary numbers are predominant in digital systems. The hex number system has a base of 16. One hex digit is equal to one Nibble, which are four binary bits. A bit is a binary number (1 or 0). Two hex digits form one Byte, which is 8 bits. There is full correspondence between all the hex digits, 0 – 9 and A – F. For example you can easily convert the following hex number to binary by writing the 4-bit number that corresponds to each hex digit starting from the radix point going outward: $(2C6B.F2)_{16} = (10\ 1100\ 0110\ 1011.1111\ 001)_2$. Notice that 0's at each end are omitted, because they are numerically meaningless. Conversely, if we group the binary bits of a binary number in groups of four starting from the binary point moving outward, we can convert the binary number to hex by writing the hex digits that corresponds to each 4-bit group.

Complements

Binary bits in digital systems have no inherent meaning. Your project and the interpretation of the bits is what give them meaning. This is especially true when bits

are interpreted as numbers. Let us say that we are using bytes. Then if we regard the bytes as unsigned numbers, we can represent numbers from 0 to 255. In general, we can represent unsigned or positive or unsigned numbers with n bits from 0 to $2^n - 1$. For bytes $n = 8$.

However, what if we wanted to represent signed numbers with bytes? Then the range of numbers would be -128 to 127. In general, for signed numbers, n bits can represent -2^{n-1} to $2^{n-1} - 1$. In addition, whereas the most significant bit of an unsigned number of n bits has a numerical weight of 2^{n-1} , the weight of the most significant bit in a signed number is -2^{n-1} . We also call the most significant bit of signed numbers the sign bit. If this bit is 0, then the number is positive, and if it is 1, the number is negative.

When we deal with signed binary numbers (positive and negatives), the operation to obtain the negative of a number, or the equivalent of multiplying by -1, is called taking the 2's complement of that number. The first step for obtaining the 2's complement of a number is to invert all the bits; i.e. if the bit is 0, then it gets set to 1, and vice versa. This is also called taking the 1's complement. Once we have the 1's complement, the final step to obtain the 2's complement, or the negative of the number, is to add 1. The following is an example.

Let us say that we want to find the negative of 01011000; this is 88 in decimal.

First invert the bits: 10100111

Then add 1. Binary addition is just like the addition that we normally use, but we have to keep in mind that if the result goes above the base 2, then a carry is generated.

CARRY					1	1	1		
	+	1	0	1	0	0	1	1	1
		0	0	0	0	0	0	0	1
RESULT		1	0	1	0	1	0	0	0

The result represents -88.

In digital systems, we are always subject to a finite storage length of n bits, so any carries beyond the n bits are discarded. If the result cannot fit in the fixed storage, then we have what is called an overflow condition. Finally, in digital systems, to do subtraction, we add the number being subtracted from with the negative of the number we are subtracting. An remember, whether the number is signed or not is always subject to the user's interpretation as required by the project at hand.

Binary Logic

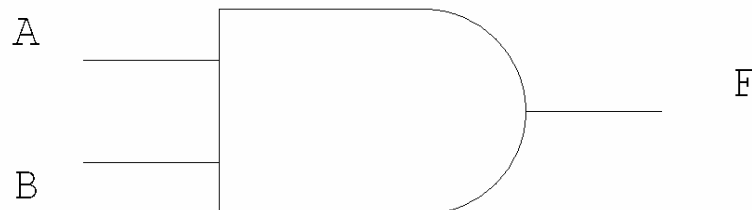
In digital systems, where we work with binary numbers 1 and 0, the numbers are represented by digital voltage levels; we call these binary signals. Usually a 0 is

represented by 0V and 1 is represented by a positive voltage greater than 0.9V. In this logic training kit, 1 is represented by 5V.

Additionally, the expressions that describe the functionality of the system are represented in what is called Boolean algebra. This branch of mathematics used to describe logic and therefore digital systems is very simple to understand. All the rules of regular algebra, such as the distributive law and the associative law apply to Boolean algebra as well. Boolean algebra has three primitive operations and operators, which are carried out by primitive logic circuits or logic gates. The three primary logic operations are the AND function, the OR function, and the NOT function. The AND and OR gates operate on at least two inputs, and the NOT gate is a unary operator, which operates on a single input. The figures and tables below show the symbols of these gates, and describe their functionality. A Truth Table is a way to completely describe how these logic operators act on all possible combination values of their inputs or operands.

<i>Binary Logic Operations</i>	
<i>Function</i>	<i>Operator</i>
AND	• (dot)
OR	+ (plus sign)
NOT	– (over bar)

Two input AND gate, operator, and Truth Table:

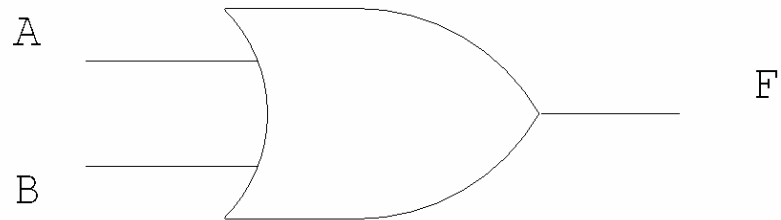


Two-input AND gate

$$F = A \bullet B$$

<i>AND Truth Table</i>	
A B	F
0 0	0
0 1	0
1 0	0
1 1	1

Two input OR gate, operator, and Truth Table:

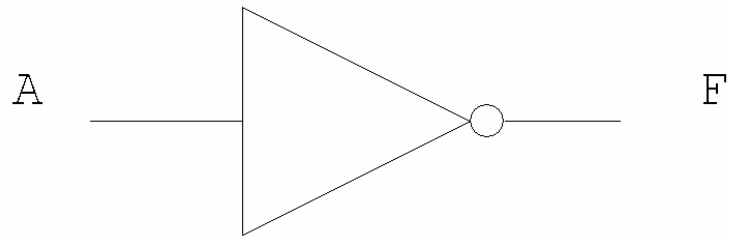


Two-input OR gate

$$F = A + B$$

<i>OR Truth Table</i>	
A B	F
0 0	0
0 1	1
1 0	1
1 1	1

Inverter, operator, and Truth Table:

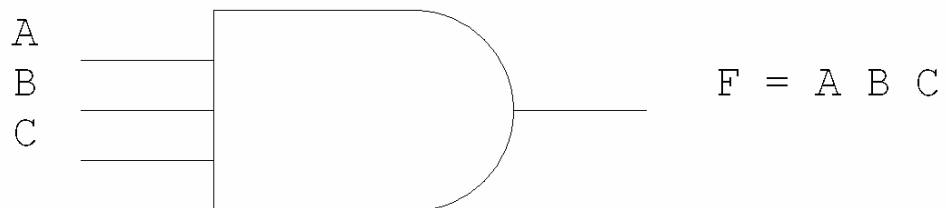


NOT gate or inverter

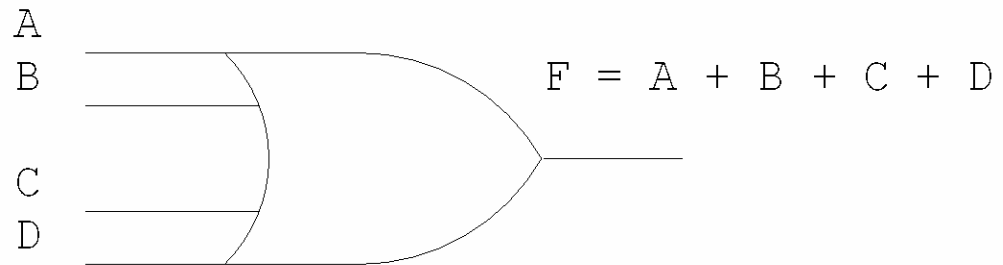
$$F = \bar{A}$$

<i>Inverter Truth Table</i>	
A	F
0	1
1	0

Since the associative law applies to Boolean algebra, it is possible to have AND and OR gates with any number of inputs, as shown in the figures below. Note that the AND dot operator can be dropped when writing AND terms. When variables are written next to each other, it is understood that they are being “ANDed”.

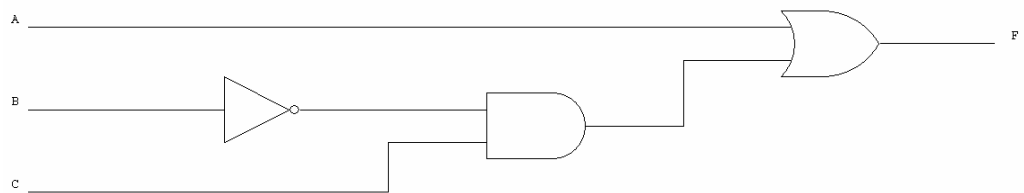


Three-input AND gate



Four-input OR gate

The figure below shows an example schematic of the implementation of the Boolean function $F = A + \bar{B} C$.



Exclusive-OR Function

In this section, we introduce the Exclusive-OR Boolean function. The symbol for this operation is an encircled plus sign: \oplus . The logic gate that implements the Exclusive-OR function is also referred to as the XOR gate, and of course as the Exclusive-OR gate. Specifically, this gate implements the Boolean function below.

$$x \oplus y = x \bar{y} + \bar{x} y$$

This gate is very important, as it implements a one-bit binary addition. The logic of its functionality can be described in words as “either or, but not both”. This means that the outcome of the XOR gate is going to be TRUE or a “one” when the inputs are different and it will be a FALSE or a “zero” when the inputs are the same. Therefore, this gate can be used to detect a difference in the inputs.

The complement, or the inverted version, of the XOR gate is the Exclusive-NOR gate, and it implements the Boolean function below.

$$\overline{x \oplus y} = x y + \bar{x} \bar{y}$$

The Exclusive-NOR gate is the equivalence gate. Its output is TRUE when the inputs are the same and FALSE when the inputs are different. Therefore, this gate is useful to detect equality of its inputs.

Several useful XOR identities are listed below.

$$x \oplus 0 = x$$

$$x \oplus 1 = \bar{x}$$

$$x \oplus x = 0$$

$$x \oplus \bar{x} = 1$$

$$x \oplus \bar{y} = \bar{x} \oplus y = \overline{x \oplus y}$$

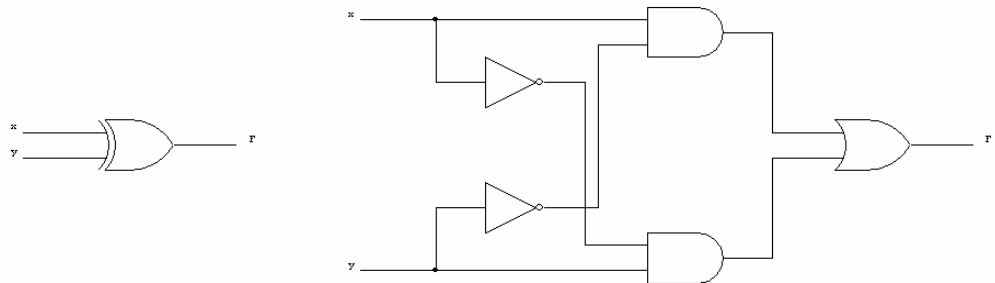
From the first two identities, we can see that the XOR gate can act as a conditional inverter. If we designate one input as the control input, it passes the value of x unchanged when the control is 0, but it inverts it when the control is 1.

Other properties, or algebraic laws, that apply to the XOR function are the commutative and associative laws, as shown below.

$$\text{Commutative Law: } A \oplus B = B \oplus A$$

$$\text{Associative Law: } (A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

The figures below show the symbol of the XOR gate and a schematic of the implementation of the function $F = x \oplus y$ with primitive gates; that is AND, OR, and inverters.



In addition and in general, an n -input XOR gate will output a 1 if the number of 1's in the n -bit input binary pattern is odd; therefore it is also called the "odd" function. In

contrast, an n-input Exclusive-NOR gate will output a 1 when the number of 1's in the n-bit input binary pattern is even, so it called the "even" function.

Combinational Circuit Design

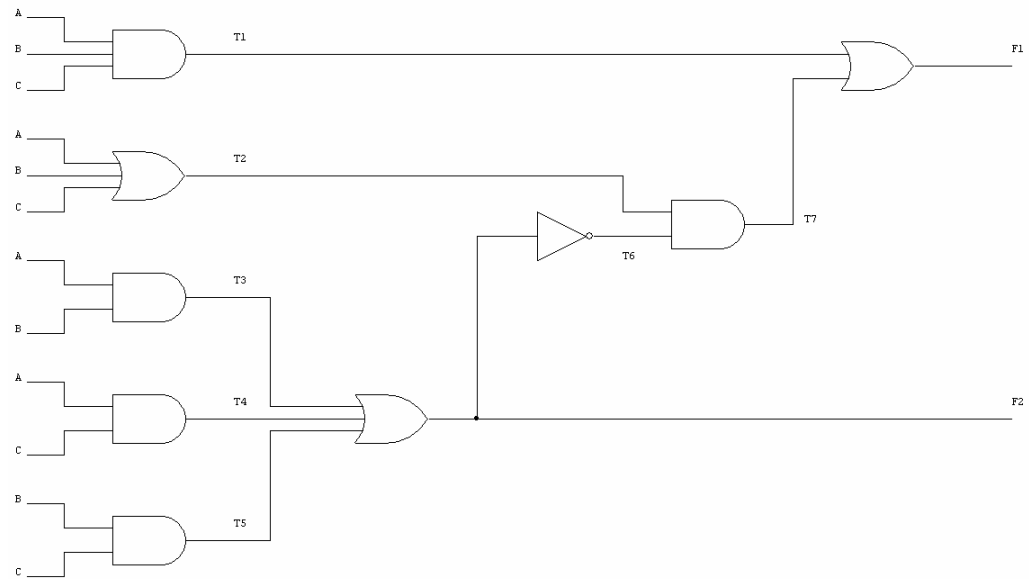
Logic circuits them employ logic gates like the ones we have examined thus far are called "Combinational Circuits". The chief characteristic of combinational logic circuits is that they are memory-less. In other words, they are simply comprised of gates connected without feedback producing m outputs, where each output is a function of any n inputs. In this context, feedback refers to a connection in the circuit, such that an output is function of itself.

Before we establish a formal procedure for designing combinational circuits, we must articulate a procedure for analyzing these circuits. Analysis differs from the design in the following way: when you analyze a circuit, you are extracting the functionality of a circuit from its schematic. By contrast, when you do design, you are synthesizing a circuit according to some set of requirements or specifications. However, analyzing circuits is an important part of the design loop, because it helps you verify that the circuit you designed indeed meets the initial requirements. The analysis procedures are presented below.

ALGEBRAIC ANALYSIS PROCEDURE:

1. Label all gate outputs that not produce a circuit output with arbitrary symbols.
2. Find the Boolean function of each gate output.
3. By repeated substitution, find the output Boolean functions in terms of the input variables of the circuits.

An example of this simple procedure is illustrated below.



Then we find the Boolean function of each gate output:

$$T1 = A B C$$

$$T2 = A + B + C$$

$$T3 = A B$$

$$T4 = A C$$

$$T5 = B C$$

$$T6 = \overline{F2}$$

$$T7 = T2 T6$$

$$F1 = T1 + T7$$

$$F2 = T3 + T4 + T5$$

Finally, by repeated substitution, we can find the outputs F1 and F2. Since we need F2 to find F1, we will find it first.

$$F2 = A B + A C + B C \text{ (RESULT)}$$

$$F1 = A B C + T2 T6$$

$$F1 = A B C + (A + B + C) \overline{F2}$$

$$F1 = A B C + (A + B + C) \overline{A B + A C + B C} \text{ (RESULT)}$$

Another analysis procedure that does not rely on Boolean algebra uses a Truth Table.

TRUTH TABLE ANALYSIS PROCEDURE:

1. Determine the input variables in the circuit. For n inputs, form the 2^n possible input combinations, and list them as binary number from 0 to 2^n-1 in a table.
2. Label all gate outputs that not produce a circuit output with arbitrary symbols.
3. Add these intermediate symbols and the circuit outputs to the table.
4. Obtain the Truth Table results for intermediate values; **AND**
5. Obtain the table results for output values.

Using the previous schematic and labels, the Truth Table result would be the following.

A B C	T1	T2	T3	T4	T5	T6	T7	F1	F2
0 0 0	0	0	0	0	0	1	0	0	0
0 0 1	0	1	0	0	0	1	1	1	0
0 1 0	0	1	0	0	0	1	1	1	0
0 1 1	0	1	0	0	1	0	0	0	1
1 0 0	0	1	0	0	0	1	1	1	0
1 0 1	0	1	0	1	0	0	0	0	1
1 1 0	0	1	1	0	0	0	0	0	1
1 1 1	1	1	1	1	1	0	0	1	1

DESIGN PROCEDURE:

1. From the specifications of the circuit, determine the required number of inputs and outputs, and assign a symbol to each.
2. Derive the Truth Table that defines the required relationship between inputs and outputs.
3. Obtain the simplified Boolean equation for each output as a function of the input variables.
4. Draw the logic diagram.
5. Verify the correctness of the design.

DESIGN PROCEDURE EXAMPLE:

Design a circuit that takes a 4-bit binary number from 0 to 9 inclusive, and outputs the sum of the input plus three.

INPUTS	OUTPUTS
A B C D	W X Y Z
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	X X X X
1 0 1 1	X X X X

INPUTS	OUTPUTS
A B C D	W X Y Z
1 1 0 0	X X X X
1 1 0 1	X X X X
1 1 1 0	X X X X
1 1 1 1	X X X X

At this point, we can use the Logic Minimizer software to enter the Truth Table results for each of the outputs and obtain the logic diagram for each of the output variables. To do this, run the Logic Minimizer software and enter the information for an output variable by clicking on the “Truth table” button. After you have entered the Truth Table outcomes, click on the “Solve” button to minimize the logic and obtain the Boolean equation for that particular output variable. Then you can click on the “Circuit” button and the Circuit dialog will appear. Click on the “Draw” button, and you will see the logic diagram for the output variable with which you are working. You have to do this process for each output variables at a time. The results for the output variables W, X, Y, Z in this example are shown in the following figures in that order.

Logic Minimizer

LM Ed Tools Help

Number of variables: 4

Solve on the fly

Show Cell Address

File Solve Fill with... Truth table Sets Formula Circuit Clear

	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Circuit

Gates: Gate type: AND/OR Gate system: Conventional

Drawing: Line thickness: Medium Size: Medium

Colors: Foreground: Background:

Draw Save Clear Close

Click on a term to highlight it on the map:

BD
A
BC

Solution form: Sum of Product **BD+A+BC**

XOR Optimized:

NUM CAPS

Logic Minimizer

Number of variables: 4

Solve on the fly

Show Cell Address

File Solve Fill with... Truth table Sets Formula Circuit Clear

	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

Circuit

Gates: Gate type: AND/OR Gate system: Conventional

Drawing: Line thickness: Medium Size: Medium

Colors: Foreground: Background:

Draw Save Clear Close

B
C'
D'

B'
D

B'
C

Fn

Click on a term to highlight it on the map:
BC'D'
BD
B'C

Solution form: Sum of Product **BC'D'+B'D+B'C**
XOR Optimized

NUM CAPS

Logic Minimizer

LM Edit Tools Help

Number of variables: 4

Solve on the fly

Show Cell Address

File Solve Fill with... Truth table Sets Formula Circuit Clear

	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

Circuit

Gates: Gate type: AND/OR Gate system: Conventional

Drawing: Line thickness: Medium Size: Medium

Colors: Foreground: Background:

Draw Save Clear Close

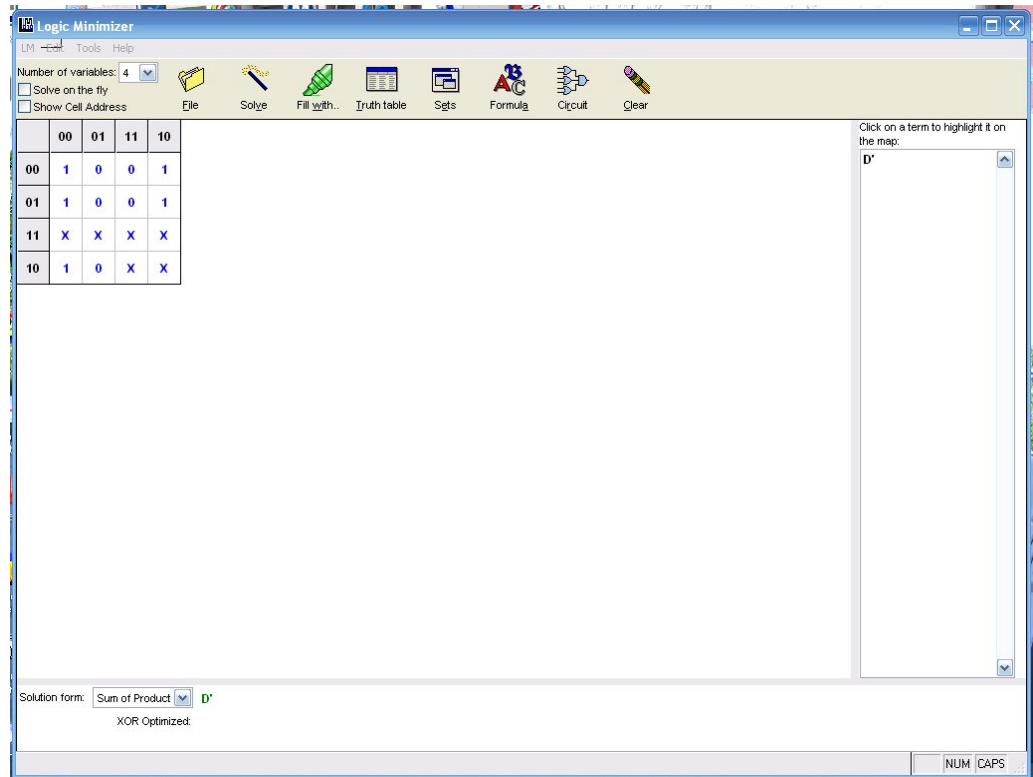
Click on a term to highlight it on the map:

C'D'
CD

Solution form: Sum of Product **C'D'+CD**

XOR Optimized: **(C#D)'**

NUM CAPS



Now you are ready to attempt your own combinational logic design projects. After the logic circuits have been obtained from Logic Minimizer, you can wire up and verify your design by choosing the most appropriate LOGIC PERSONALITY, downloading the same to the C Stamp, wiring the circuit, and making sure it works as required. The process of downloading a LOGIC PERSONALITY was outlined in Chapter 2, and diagrams of all logic personalities are presented in Chapter 5.

Combinational Logic Project 1

In this project, you will study digital design using the C Stamp. The digital design will be evaluated using a C Stamp LOGIC PERSONALITY. Additionally, you will become familiar with the electronic digital design components in the different C Stamp logic personalities and their specifications.

Design a combinational circuit that counts the coins placed into an automatic toll coin collector. The toll is \$0.10 and the machine gives no change. When at least \$0.10 are received the go-signal is turned on and the money is collected (no change is given). Otherwise, the stoplight remains on.

Implement the design with the C Stamp.

What would be your overall toll collector scheme including any mechanical components?

What is your logic schematic?

How would you test your design?

Did the design work once you downloaded the appropriate C Stamp LOGIC PERSONALITY and wired your schematic?

Combinational Logic Project 2

Design a logic circuit that has a 4-bit binary input from 0 to 9 inclusive, and has outputs to display that number in a 7-segment LED display. The digital design will be evaluated using a C Stamp LOGIC PERSONALITY.

Implement the design with the C Stamp.

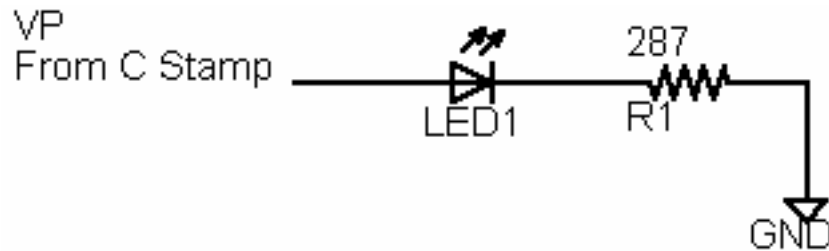
What are your logic schematics for each output?

How would you test your design?

Did the design work once you downloaded the appropriate C Stamp LOGIC PERSONALITY and wired your schematic?

USING A 7 SEGMENT LED DIGIT DISPLAY:

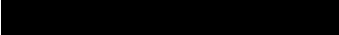









Each segment of the 7-segment LED display is represented by the LED in the figure below. The outputs of your design will drive the LEDs with a 1 to light the light emitting diodes (LEDs).



Each segment circuit is comprised by an LED and a limiting resistor R1. In general, diodes are electronic devices that (conduct) allow current to flow) in one direction from a high voltage point in the circuit to a low voltage point, while dropping a constant voltage across it. Ground or GND is always at 0 V. The symbol similar to an arrow denotes the direction of the flow of current, and when the diode is conducting current, we say that it is forward biased. The left side of the symbol as drawn above is called the anode.

To forward bias a diode, the anode must be at a higher voltage than the cathode by the constant voltage that the diode drops across it. When forward biased, the voltage across a diode is constant. LEDs are a special type of diode that emits light when forward biased. Depending on the semiconductor material used to fabricate the LED, the color (wave length) of the light emitted is different, so any given LED will emit light of a given color.

The other component in the circuit is R1. Resistors are electronic components that allow certain amount of current to flow through them depending on the voltage across it, and the value of its resistance. The resistance value depends on the material and how much of this material is used to fabricate the resistor. The unit of resistance is ohms, and it is symbolized by the capital Greek letter omega (Ω). In many resistors, the resistance is color coded with three, four, or five color stripes. If the last stripe is either gold or silver, it represents the tolerance of the resistance (i.e. how much the value of the resistance varies, since no component will have an exact value over all possible manufactured pieces). A gold band means a $\pm 5\%$ tolerance, silver means $\pm 10\%$, and the absence of a gold or silver band means $\pm 20\%$. For the other color bands, the first stripe represents the first digit, the second the second digit, and the third a power of 10 multiplier. If an additional non gold or silver band is present, the third stripe represents the third digit and the fourth the power of 10 multiplier. The numerical meaning of the non gold or silver color bands are shown in the table below. For example, a resistor with color bands yellow, violet, and brown has a resistance of 470Ω . 4 for the first digit, 7 for the second and a 1 power of ten multiplier $10^1 = 10$ ($47 \times 10 = 470$).

<i>Color</i>		<i>Meaning</i>
	BLACK	0
	BROWN	1
	RED	2
	ORANGE	3
	YELLOW	4
	GREEN	5
	BLUE	6
	VIOLET	7
	GRAY	8
	WHITE	9

Now that we know how to bias LEDs, we can proceed to using a 7 Segment LED Digit Display.

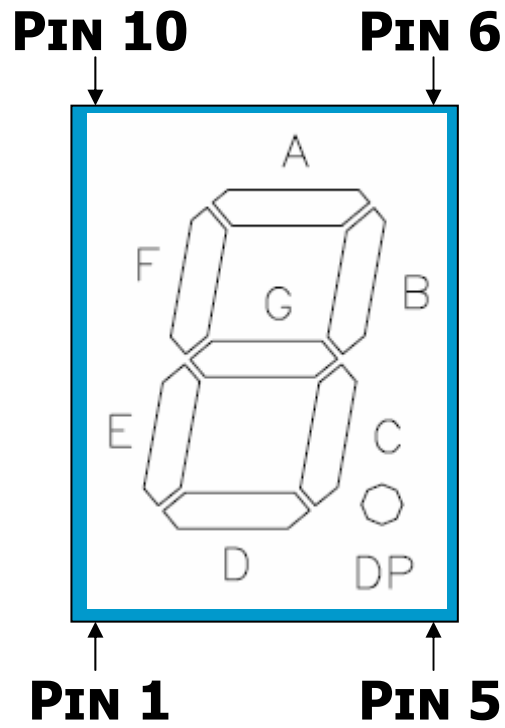
The circuit for this project is a 7 Segment Display properly biased and connected to the C Stamp in the μ C101 BOL. The A-WIT Part# CS480000 is a suitable 7 Segment LED Digit Display that can be used for this project. This display is also provided with some A-WIT KITS.

The CS480000 is a 0.56 inch (14.22 mm) digit height single-digit display. This device uses GREEN LED chips. The display has gray face and green segments.

Technical Specifications

- 0.56 inch (14.22 mm) digit height
- Excellent segment uniformity
- Low power requirement
- High brightness and high contrast
- Wide viewing angle
- Solid state reliability
- High luminous intensity
- Common anode
- Current limiting resistor per segment $R_L = 287 \Omega$
- Additional right decimal point
- Breadboard friendly

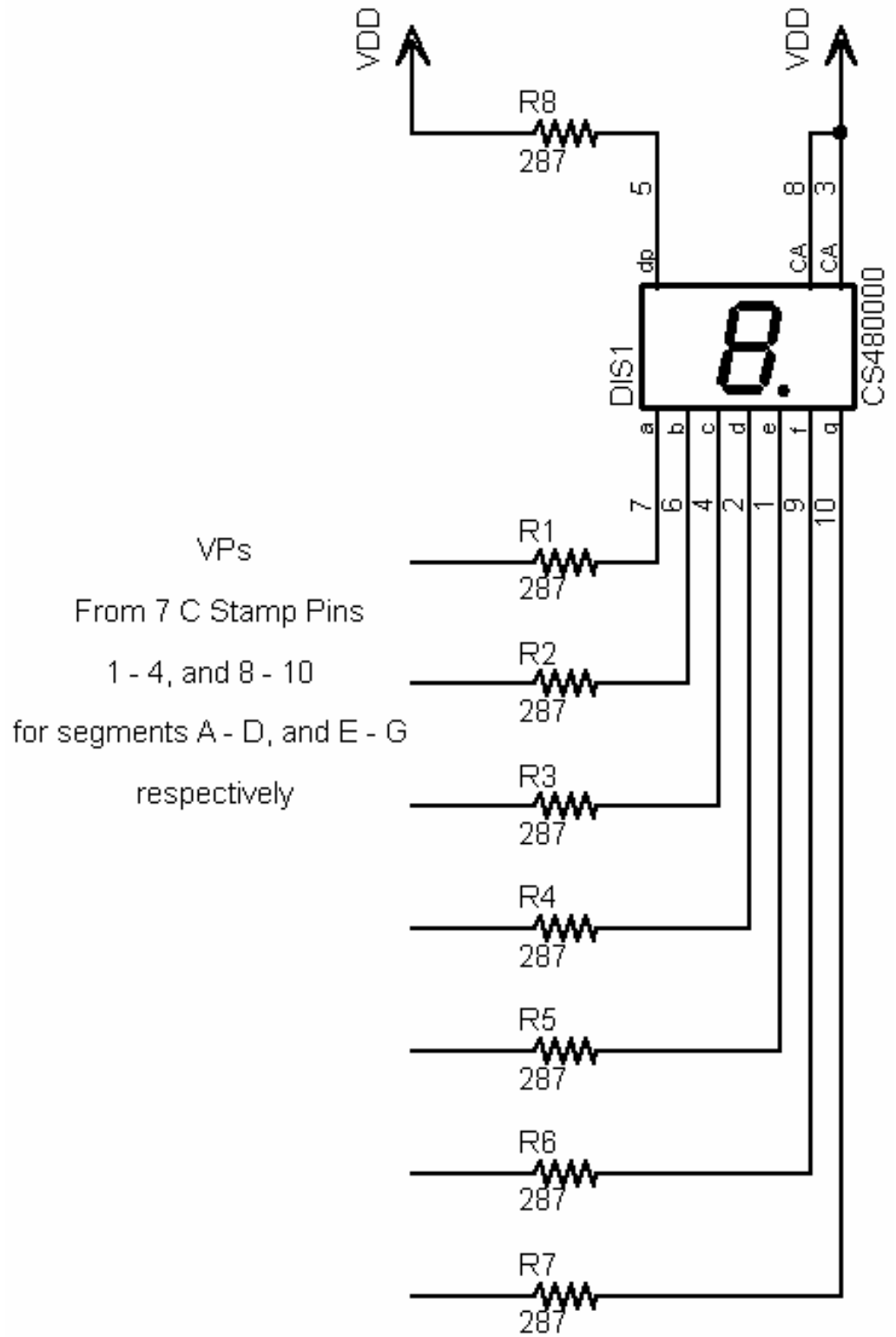
The figure below shows the pins orientation of the display and the segment notations; and the following table shows the pin to segment mapping.

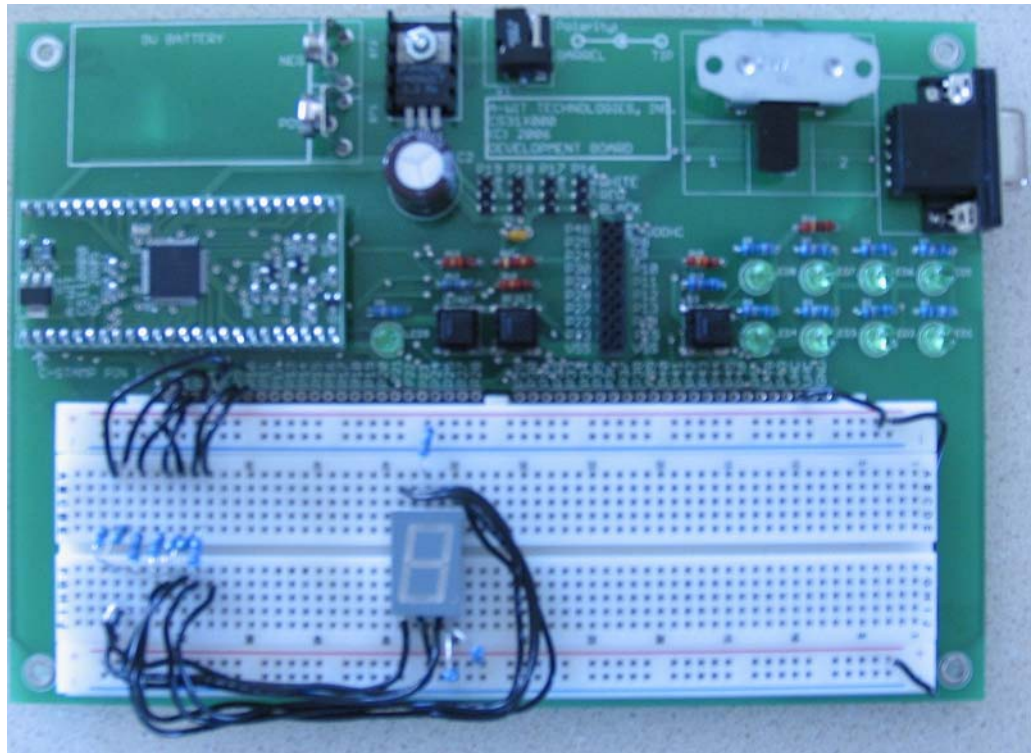


<i>Pin Number</i>	<i>Connection</i>
1	Cathode E
2	Cathode D
3	Common Anode
4	Cathode C
5	Cathode DP
6	Cathode B
7	Cathode A
8	Common Anode
9	Cathode F

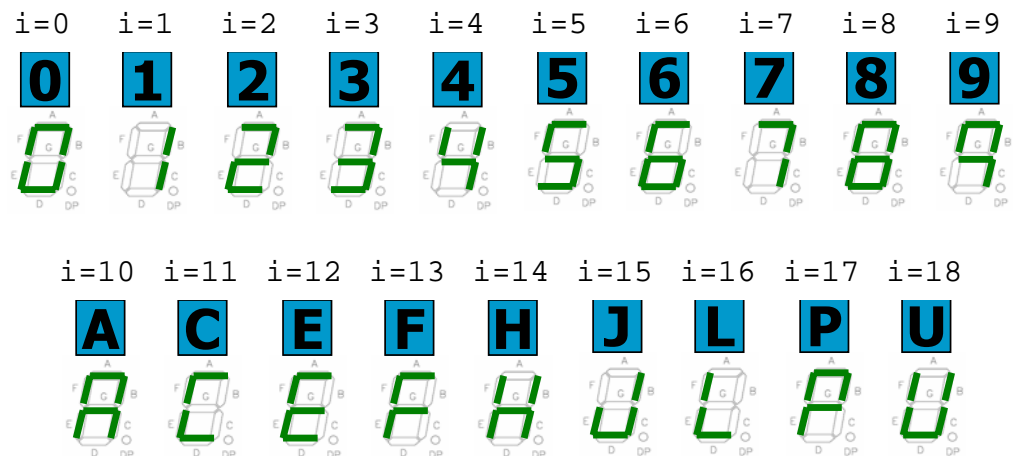
<i>Pin Number</i>	<i>Connection</i>
10	Cathode G

Since the display has a common anode, which in LEDs has to be connected to a higher voltage than the cathode for the LED to be lit, we know that we have to connect Pin 3 or Pin 8 (or both) to VHC. Then we control whether each segment is lit by providing a voltage VP from the C Stamp. A LOW will light a segment, and a HIGH will keep the segment off. From the specifications of the display, we also know that each segment needs a current limiting resistor $R_L = 287 \Omega$. Additionally, since the decimal point (DP) segment will not be used, we connect it to VHC through a limiting resistor. The circuit schematic and the wired circuit in the μ C101 BOL are shown in the following two figures. We use C Stamp pins 1 – 4, and 8 – 10 in the example schematic, but you will have to use the appropriate pins for your design according to what is available in the LOGIC PERSONALITY you choose to implement and test your design.





To implement your design, the first step is to determine which segments are needed to be on to display the numbers 0 – 9 without ambiguity. The figure below shows this for the numbers, and even for some letters, although you will only be implementing the numbers 0 through 9.

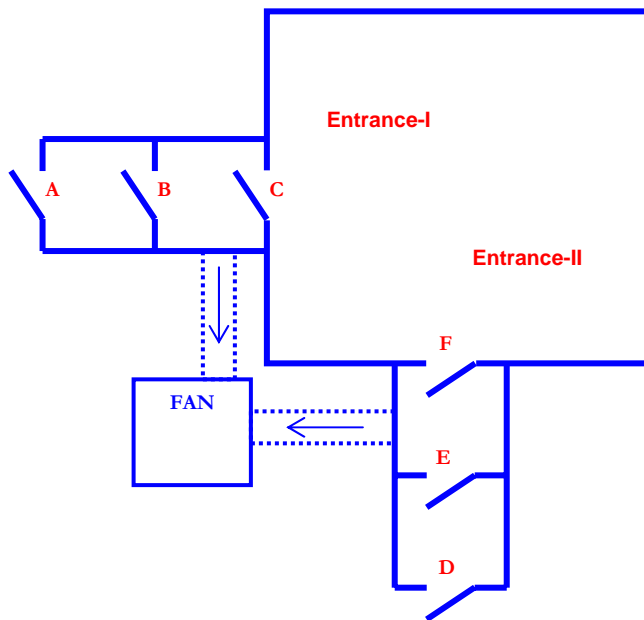


Combinational Logic Project 3

A layout diagram of an operating room at a hospital is shown below. There are two entrances, each with 3 doors that enclose two intermediate rooms. The purpose of the intermediate rooms is to keep dirt and germs out of the operating room, which is a very good thing! Each entrance is identical in design and operation. As an example, let us look at the operation of Entrance #1. The inner Door C is enabled to be opened (actually opened automatically so no one needs to touch it) only when both door A and door B are fully closed (i.e. no new germs/dirt can come in!). Also, if either of the intermediate doors (B or E) are open then a high speed bio-fan (fan plus bio-filter) must be turned on, ensuring that any left-over beasts get sucked out!

We will only be modeling doors A, B, C and E. Let switches S1, S2, and S3 correspond to doors A, B, and E, respectively. Let an Open switch represent an open door and a Closed switch represent a closed door. Your mission is to build a circuit that accomplishes this very important function of the operating room doors. The surgeons really need this room operational by the due date. [Life lesson: always keep medical personnel happy; you never know when you will need them!] Let us have you turn on an LED when both doors A and B are closed (i.e. - you have enabled door C to open with a motor). Similarly, let us light another LED only when the bio-fan is ON. [Obviously, the LED is OFF when the bio-fan is off; the hospital is world-famous for low energy consumption so turning devices off when not needed is a common constraint.]

HAVE FUN!

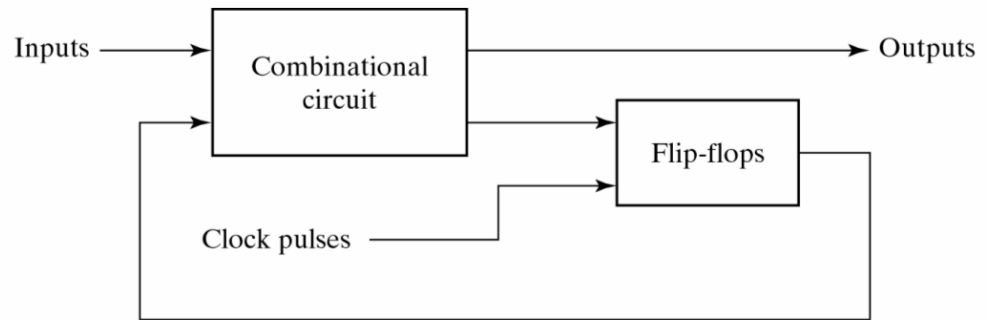


Sequential Logic

This chapter describes the Sequential Logic Design projects and activities that can be done with the C Stamp and the μ C101 BOL. The material in this chapter is written at a very fundamental level, so depending on the present level of expertise, not all the material will have to be reviewed by all readers. This chapter also provides background in sequential logic design that leads to the successful completion and understanding of the design projects.

Synchronous Clocked Sequential Circuits

Synchronous clocked sequential circuits are characterized by the addition of memory elements to the combinational logic with which we are already familiar. This is shown in the figure below. The memory elements are called D-Flip-Flops. These flip-flops do not change the output when the input changes, but the output becomes the logic level of the input at the instant of the rising (low to high) edge of the clock. The flip-flops may also have asynchronous reset and preset inputs to force the output to a zero or a one at any time regardless of the clock, so you can start your sequential circuit in a known state. This is not always necessary, and is mostly used at the beginning of the circuit operation. The D flip-flops present in the logic personalities for the C Stamp are self resetting; meaning that they power up or start at an initial set of a zero output, so you can always start your sequential circuit at a known state.

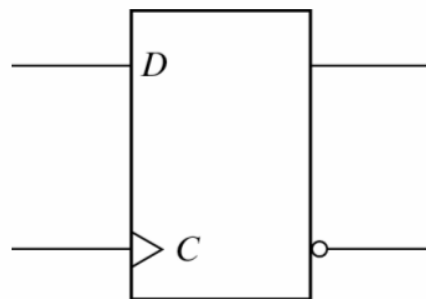


(a) Block diagram

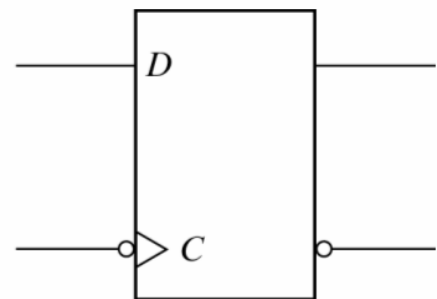


(b) Timing diagram of clock pulses

The figure below shows symbols for D flip-flops. D is the input, and the outputs on the right are the Q outputs. These examples have two complementary outputs: Q is the non-bubbled output and \bar{Q} is the bubbled output. \bar{Q} is always the complement, or inverted version, of Q. The clock inputs can respond to the rising, or positive edge (low to high), or to the falling, or negative edge (high to low). Flip-flops that are negative edge triggered have a bubble in the clock input of the symbol.



(a) Positive-edge



(a) Negative-edge

CHARACTERISTIC EQUATION:

The equation that describes how the output of the flip-flop changes in response to the input at the trigger edge is called the Characteristic Equation of the flip-flop. For the D flip-flop, the characteristic equation is simply:

$$Q(t+1) = D$$

This equation is read as follows: The output Q of the flip-flop after the trigger edge will be the value of the input D.

Analysis of Clocked Sequential Circuits

Before we learn how to design sequential circuits, we must learn the tools of how to analyze existing circuits. This Section introduces the background for analyzing Clocked Sequential Circuits.

STATE EQUATIONS:

The behavior of a clocked sequential circuit can be described by state equations. These are also called transition equations, because these Boolean equations describe how the circuit transitions from state to state, as well as how the outputs react to the inputs and the present state.

We define the following terms as follows:

In general, the NEXT STATE is defined by a Boolean function of the present STATE and the present INPUT.

$$NEXT\ STATE \equiv STATE(t+1) \equiv f(STATE(t), INPUT(t))$$

The present OUTPUT is defined by a Boolean function of the present STATE and the present INPUT.

$$OUTPUT \equiv OUTPUT(t) \equiv g(STATE(t), INPUT(t))$$

For simplicity, we write:

$$STATE(t+1) = f(STATE, INPUT)$$

and

$$OUTPUT = g(STATE, INPUT)$$

where

$$\begin{aligned} STATE(t+1) &= NEXT STATE \\ STATE &= CURRENT STATE \\ INPUT &= CURRENT INPUT \\ OUTPUT &= CURRENT OUTPUT \end{aligned}$$

STATE TABLE:

State tables are a way to express how the state and outputs change in response to inputs. There are two ways to model a State Table. The first one has all possible combinations of states and inputs tabulated vertically as in a Truth Table. The table below is an example of this. This model of a state table works well for any number of inputs.

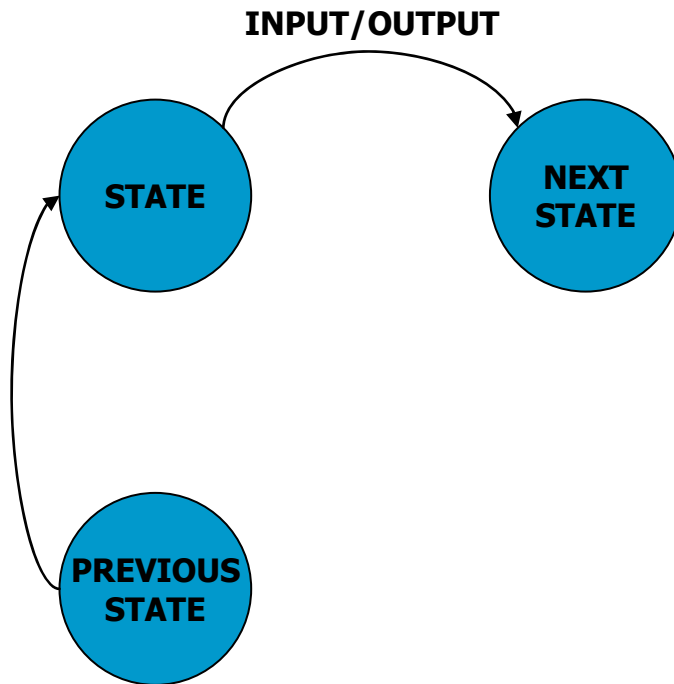
<i>PRESENT STATE</i>	<i>INPUT</i>	<i>NEXT STATE</i>	<i>OUTPUT</i>
A ...	X ...	A ...	Y ...

The next model of a state table has all the possible combinations of the inputs horizontally for each state and the output. This model works well only when the number of inputs is less than three. Of course, if the outputs do not depend on the input, it is not necessary to express all the combinations of inputs under the output column, because the outputs depend only on the present state. The following table is an example of this model.

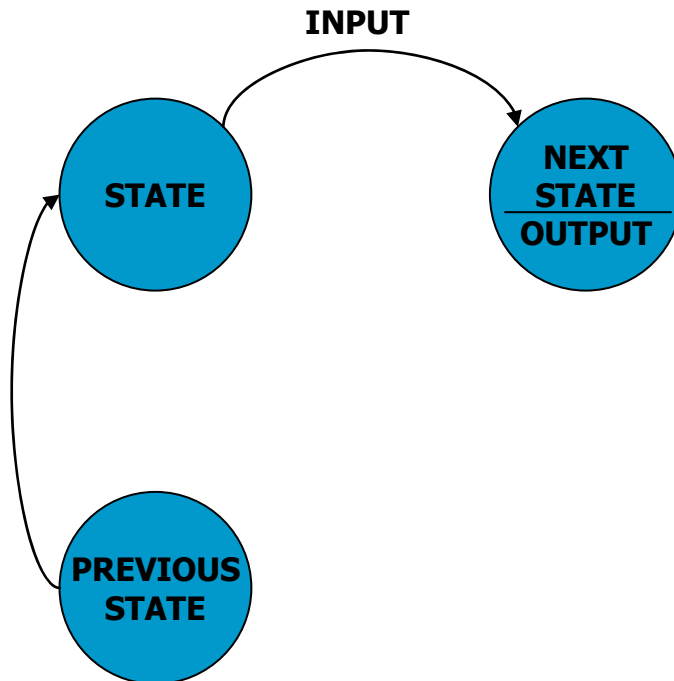
<i>PRESENT STATE</i>	<i>NEXT STATE</i>			<i>OUTPUT</i>		
	<i>X = 0</i>	<i>X = 1</i>	<i>...</i>	<i>X = 0</i>	<i>X = 1</i>	<i>...</i>
A ...	A ...	A ...		Y ...	Y ...	

STATE DIAGRAM:

A State Diagram expresses the same information as a State Table, but in a graphical form. In general, it looks like the figure below; where the outputs are a function of the present state and the inputs. This type of Clocked Sequential Circuit is called a Mealy State Machine.



If the outputs were only a function of the present state, we would put the output inside the state on which it depends. This is shown in the figure below. This type of Clocked Sequential Circuit is called a Moore State Machine.



Design of Clocked Sequential Circuits

Now that we are familiar with some tools to analyze state machines, we can go on to learning how to designing. However, before that, we must learn an additional tool that is going to help us simplify the design process. This procedure has to do with state reduction.

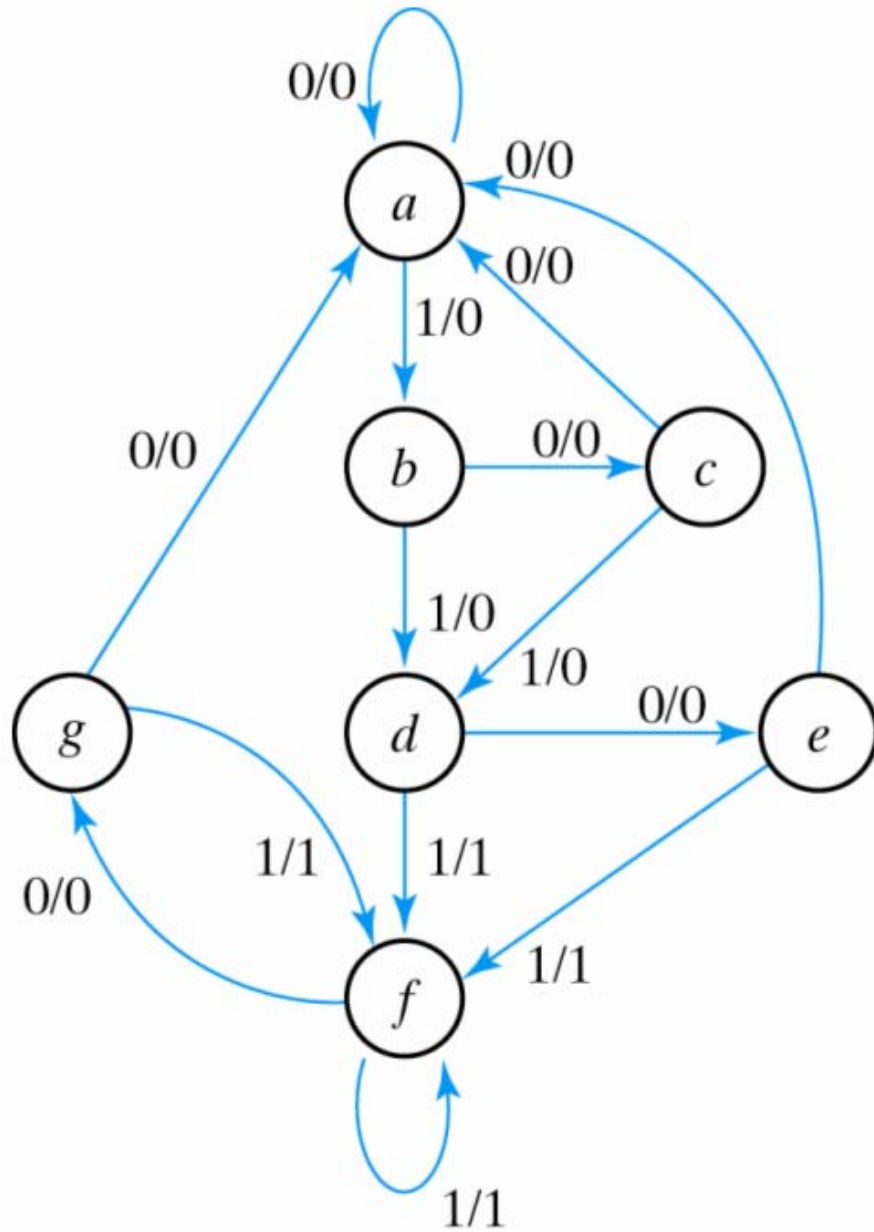
STATE REDUCTION:

In a State Table, states are equivalent if, and only if:

- For the same inputs, they produce the same output.
- For the same inputs, they send the circuit to the same state or to an equivalent state.

This procedure should be applied in its entirety to a State Table until there is no further state reduction.

Let us present an example with the following State Diagram.



The table below shows the State Table that results from the State Diagram:

<i>PRESENT STATE</i>	<i>NEXT STATE</i>		<i>OUTPUT</i>	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	a	f	0	1
g	a	f	0	1

<i>PRESENT STATE</i>	<i>NEXT STATE</i>		<i>OUTPUT</i>	
	<i>X = 0</i>	<i>X = 1</i>	<i>X = 0</i>	<i>X = 1</i>
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

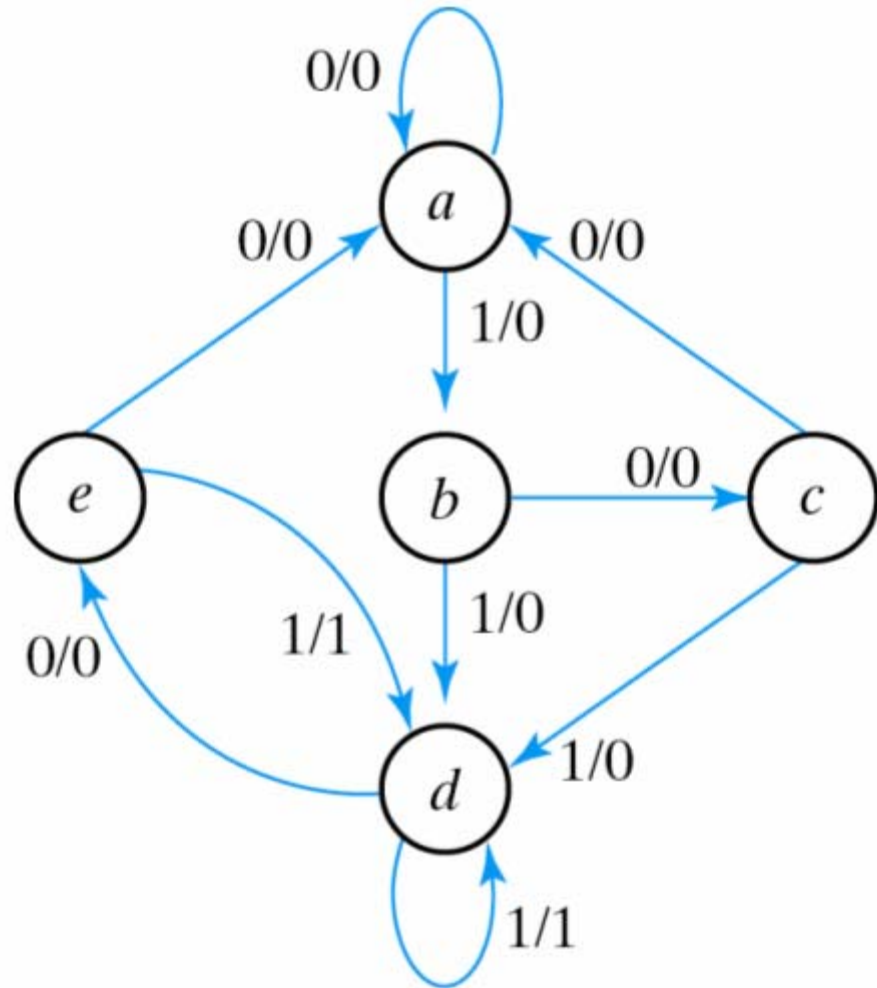
We can see from the table above that states e and g are equivalent, so we eliminate state g and write in e wherever g is present in the remaining rows. This results in the following table.

<i>PRESENT STATE</i>	<i>NEXT STATE</i>		<i>OUTPUT</i>	
	<i>X = 0</i>	<i>X = 1</i>	<i>X = 0</i>	<i>X = 1</i>
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

We can see from the table above that states d and f are now equivalent, so we eliminate state f and write in d wherever f is present in the remaining rows. This results in the following table.

<i>PRESENT STATE</i>	<i>NEXT STATE</i>		<i>OUTPUT</i>	
	<i>X = 0</i>	<i>X = 1</i>	<i>X = 0</i>	<i>X = 1</i>
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Now if we obtain a State Diagram from the reduce State Table, we can see that this diagram is simpler than the original one, which will result in a simpler and more economical circuit. The reduced State Diagram is shown in the figure below.

***DESIGN PROCEDURE:***

The following is the procedure for the design of clocked sequential logic circuits, or state machines.

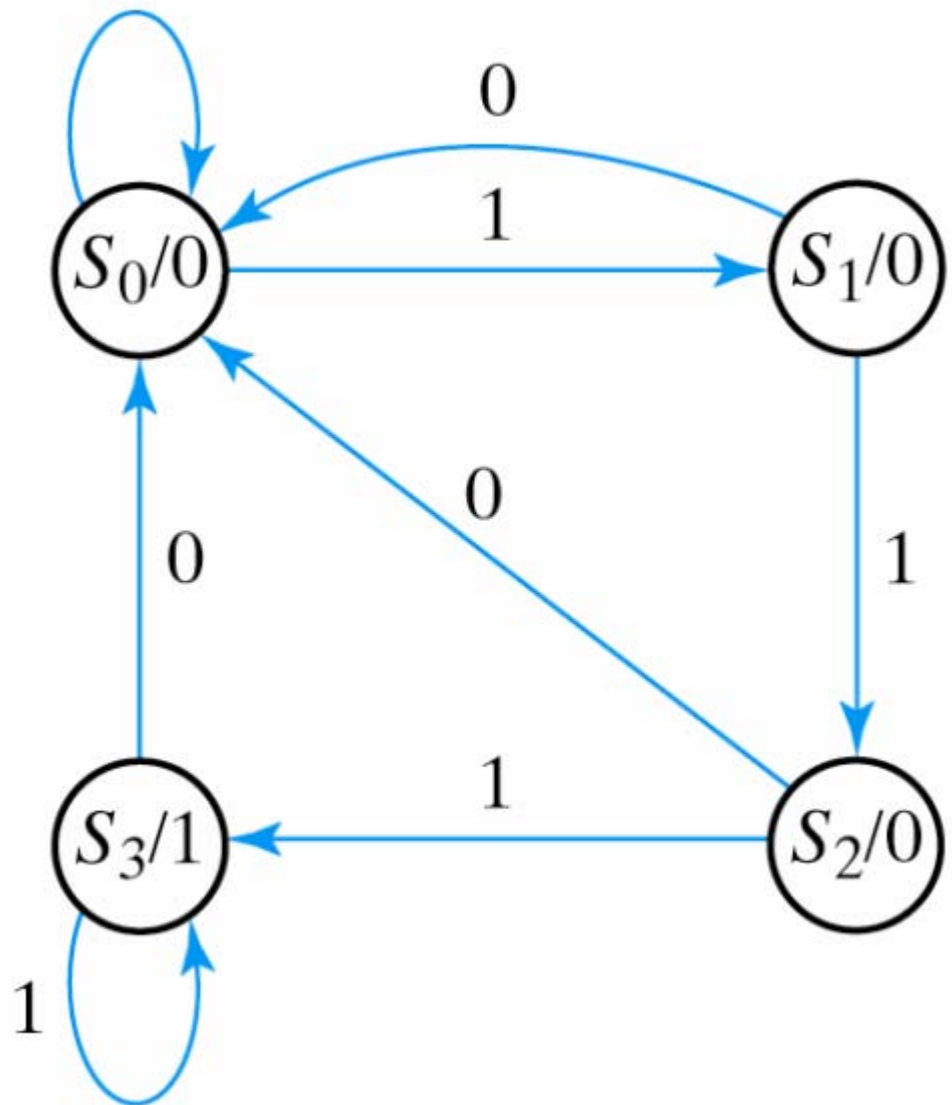
1. From the word description of the problem to be solved or the specification of the design, derive a State Diagram.
2. From the State Diagram, derive a State Table.
3. Reduce the number of states in the State table if possible.
4. Number the states with binary numbers.

5. Code State Table with state assignments.
6. Derive truth tables from the State Table for the outputs of the next state logic and the outputs of the state machine.
7. Use Logic Minimizer to obtain simplified Boolean equations and logic diagrams for the outputs of the next state logic and the outputs of the state machine.
8. Use the Logic Minimizer results to draw the complete State Machine, along with the appropriate number of D flip-flops, which correspond to each bit in the state number. The Q outputs of the flip-flops are the Present State, and the D inputs of the flip-flops are the Next State.

DESIGN PROCEDURE EXAMPLE:

Design a State Machine that detects a serial input of three consecutive 1's. The sequence can overlap, and the output is 0 when the sequence has not been detected and 1 when the sequence has been detected.

The State Diagram for the solution of this problem is shown in the following figure. Note that it meets the specification requirements in that the output is 0 until three 1's have been detected at the input. At that point, the output becomes 1, and it remains a 1, as long as 1's keep being present at the input. This occurs because the sequence of three 1's can overlap. If a 0 is detected at the input at any time, the State Machine returns to the initial state to wait for the beginning of a new sequence. Also, note that the output of the State Machine depends only on the state, so it is a Moore State Machine.



The State Table that is derived from this State Diagram is shown below. Note that no state reduction is possible.

<i>PRESENT STATE</i>	<i>INPUT</i>	<i>NEXT STATE</i>	<i>OUTPUT</i>
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	0
S_2	1	S_3	0
S_3	0	S_0	1
S_3	1	S_3	1

We will number states S_0 , S_0 , S_0 , and S_0 with binary numbers 00, 01, 10, and 11 respectively. Since our state number has 2 bits, we need two D flip-flops in this State Machine, and these state number bits will correspond to the outputs of the D flip-flops. The next state bits correspond to the D inputs of the flip-flops. Then we can code the State Table with these state assignments.

<i>PRESENT STATE</i>		<i>INPUT</i>	<i>NEXT STATE</i>		<i>OUTPUT</i>
	$Q_1 Q_0$	X		$D_1 D_0$	Y
S_0	00	0	S_0	00	0
S_0	00	1	S_1	01	0
S_1	01	0	S_0	00	0
S_1	01	1	S_2	10	0
S_2	10	0	S_0	00	0
S_2	10	1	S_3	11	0
S_3	11	0	S_0	00	1
S_3	11	1	S_3	11	1

From this coded State Table, we can derive truth tables for D_1 and D_0 as functions of Q_1 , Q_0 , and X . These truth tables are shown below. Then we can use these truth tables in the Logic Minimizer software to get the circuits for D_1 and D_0 with input Q_1 , Q_0 , and X each.

Since the output Y depends only on the present state, we can observe from the coded State Table that the Boolean function for Y is:

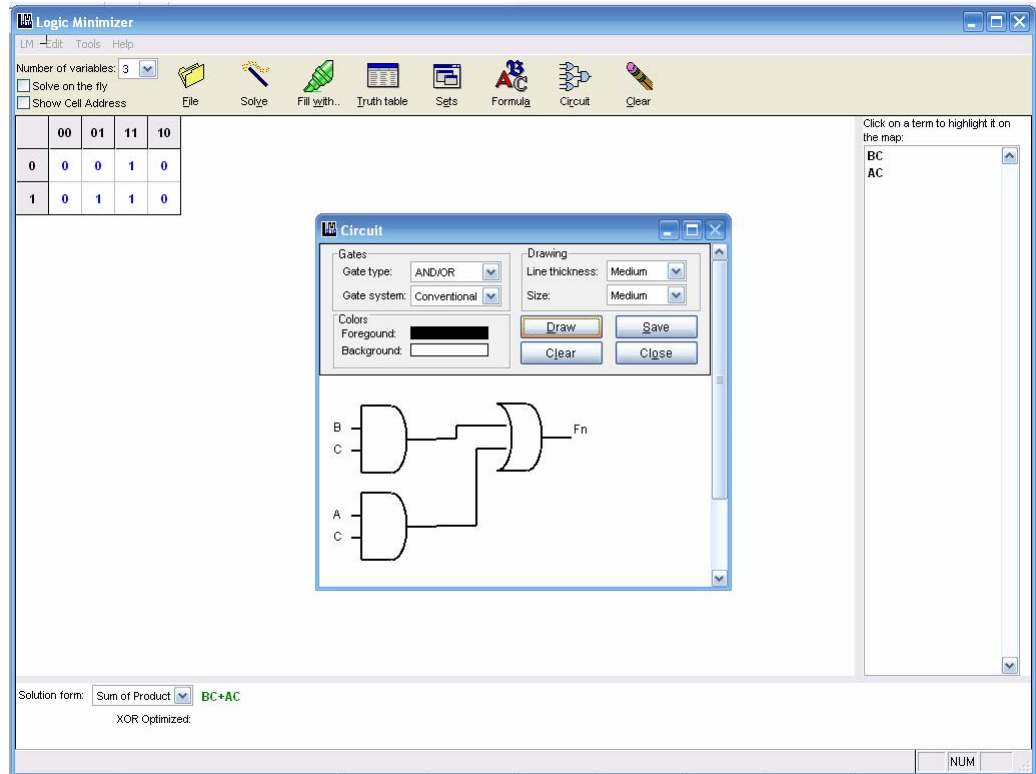
$$Y = Q_1 Q_0$$

<i>TRUTH TABLE FORD₁</i>		
$Q_1 Q_0$	X	D_1
00	0	0
00	1	0
01	0	0
01	1	1
10	0	0
10	1	1
11	0	0
11	1	1

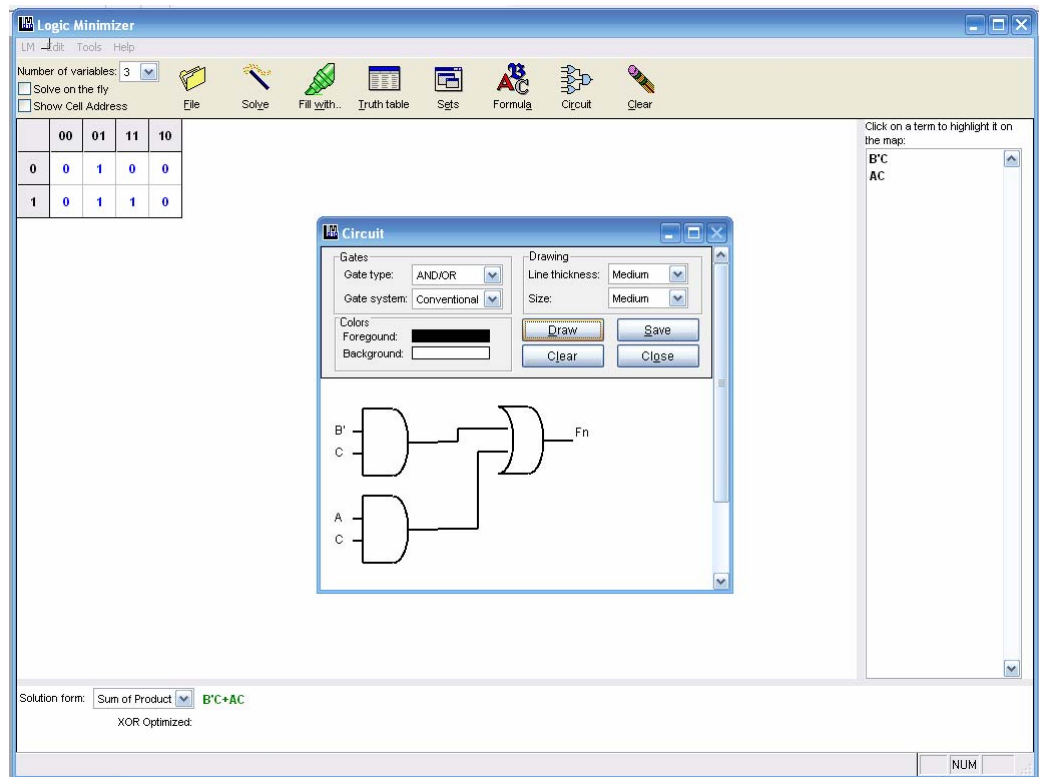
<i>TRUTH TABLE FORD₀</i>		
$Q_1 Q_0$	X	D_0
00	0	0
00	1	1
01	0	0
01	1	0
10	0	0
10	1	1
11	0	0
11	1	1

The circuits that result from solving these truth tables with Logic Minimizer are shown below.

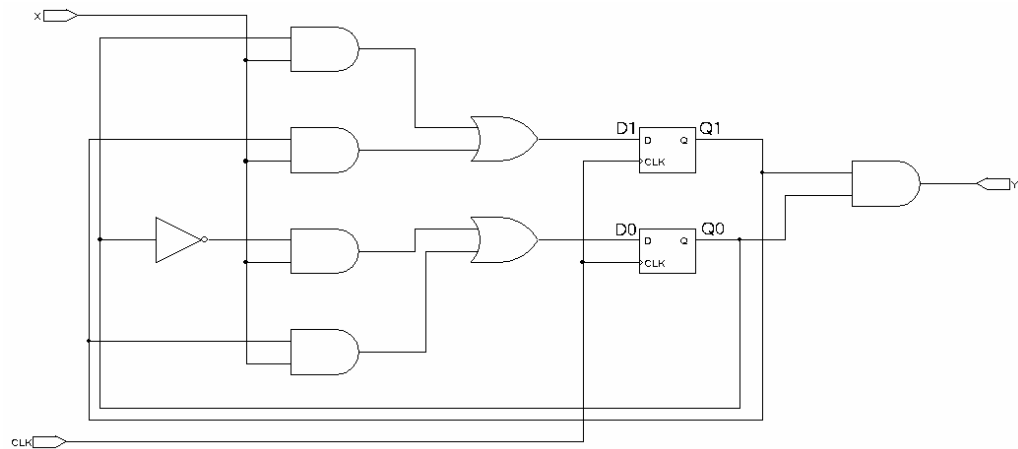
In this figure, the inputs A, B, and C correspond to Q_1 , Q_0 , and X respectively; and the output F_n corresponds to D_1 .



In this figure, the inputs A, B, and C correspond to Q_1 , Q_0 , and X respectively; and the output F_n corresponds to D_0 .



The figure below shows the completed circuit for the State Machine.



Sequential Logic Project 1

In this project, you will obtain experience with sequential logic design, and study digital design using the C Stamp logic personalities. This project builds on the previous ones, and it is assumed that you have performed the previous projects. If this material is not

clear, you should go back and repeat the previous projects. The previous State Machine design example will prove to be valuable as well.

Design a controller that detects the four-bit, binary MSB first serial sequence of the last digits of your Social Security Numbers. The input bit should be setup before the clock pulse.

Implement the design using one of the C Stamp logic personalities.

What is the digit that you are trying to detect?

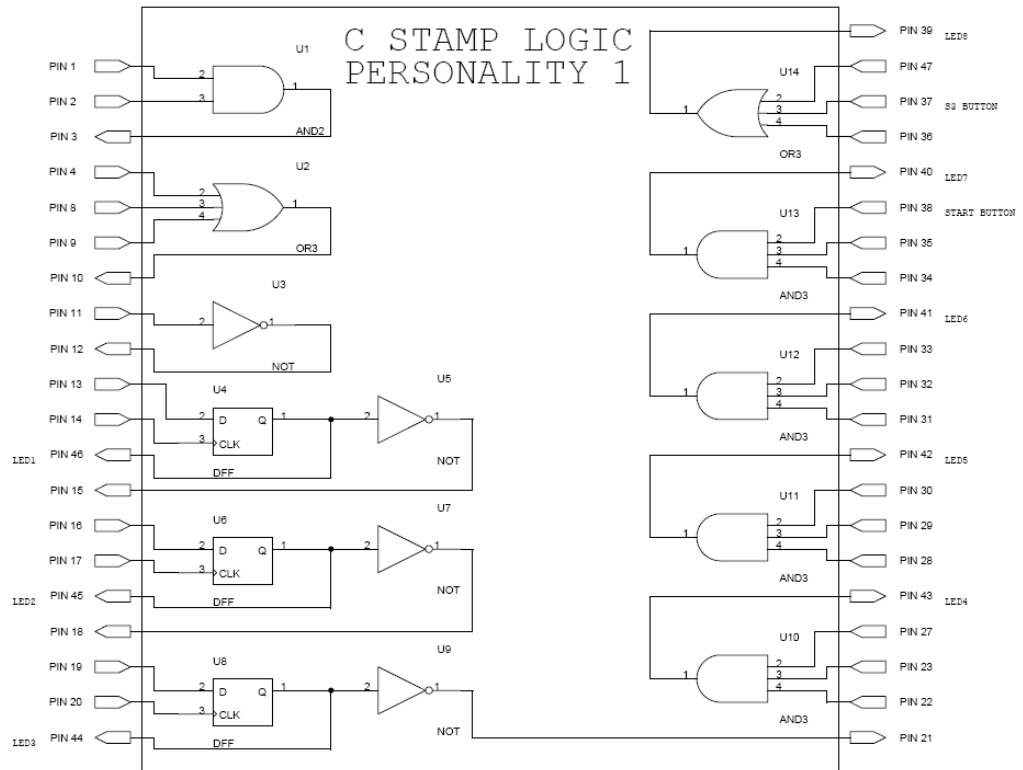
What is the resulting schematic of your completed design?

What C Stamp LOGIC PERSONALITY did you use and what connections you made to implement and test your design?

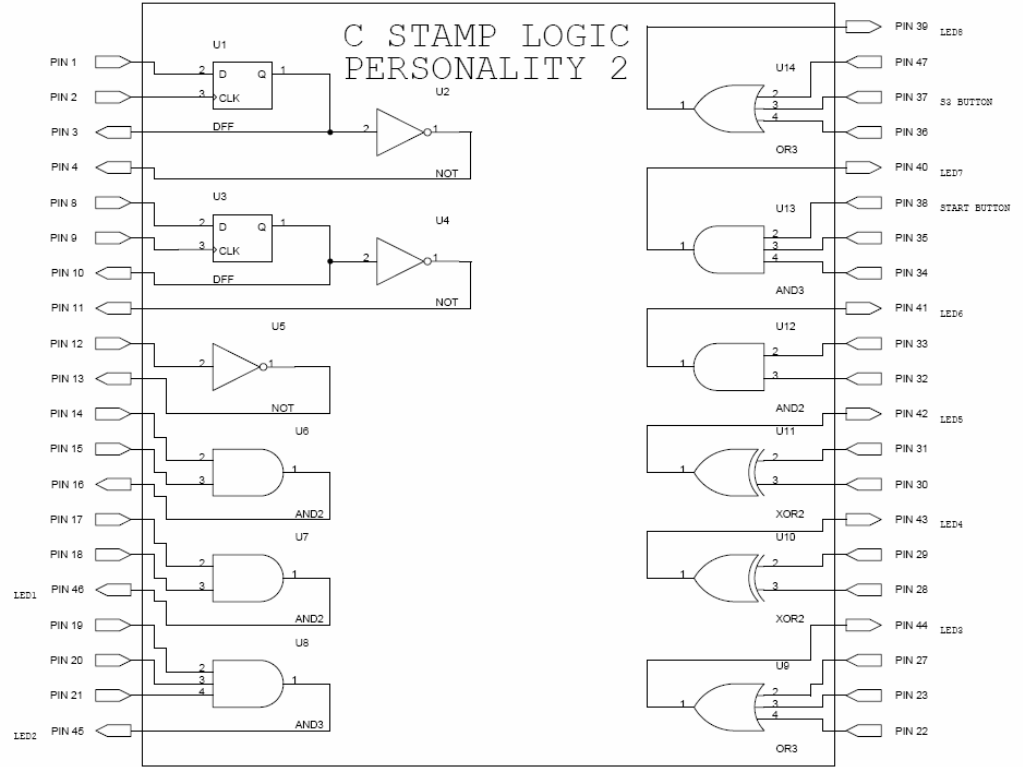
C Stamp Logic Personalities

This chapter describes the different logic personalities that come available you're your C Stamp Logic Training Kit. You download the appropriate LOGIC PERSONALITY to the C Stamp according with the design that you want to implement. You have to recognize and make sure that the LOGIC PERSONALITY has all the components that you need to implement your design available in it.

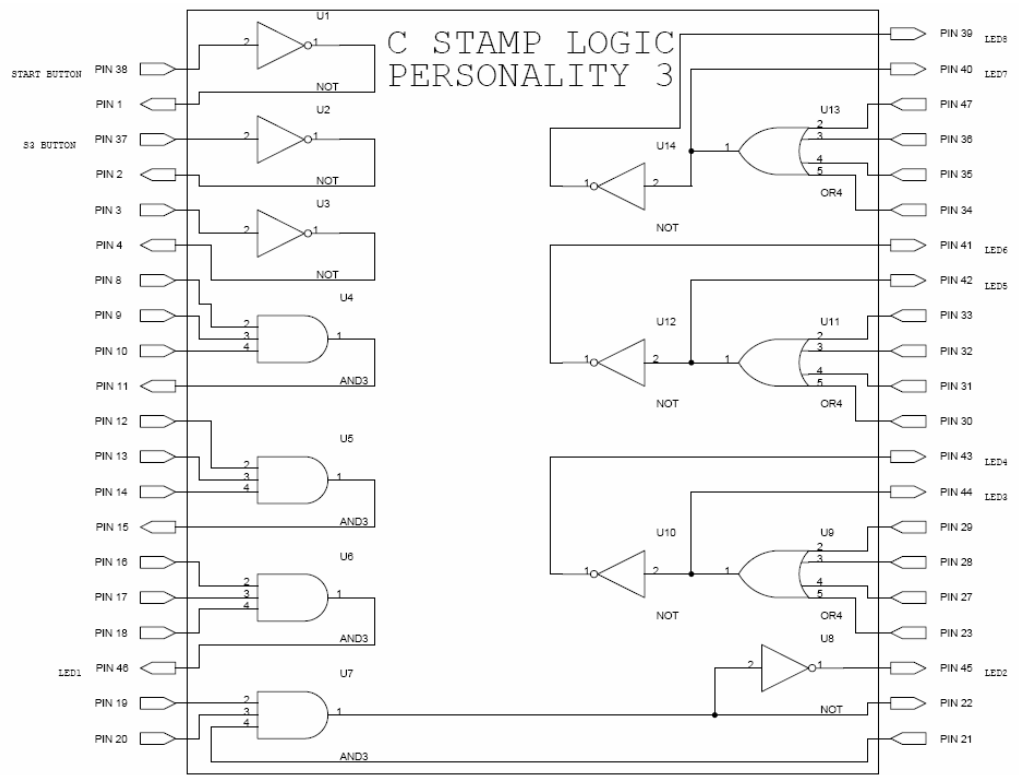
C Stamp Logic Personality 1



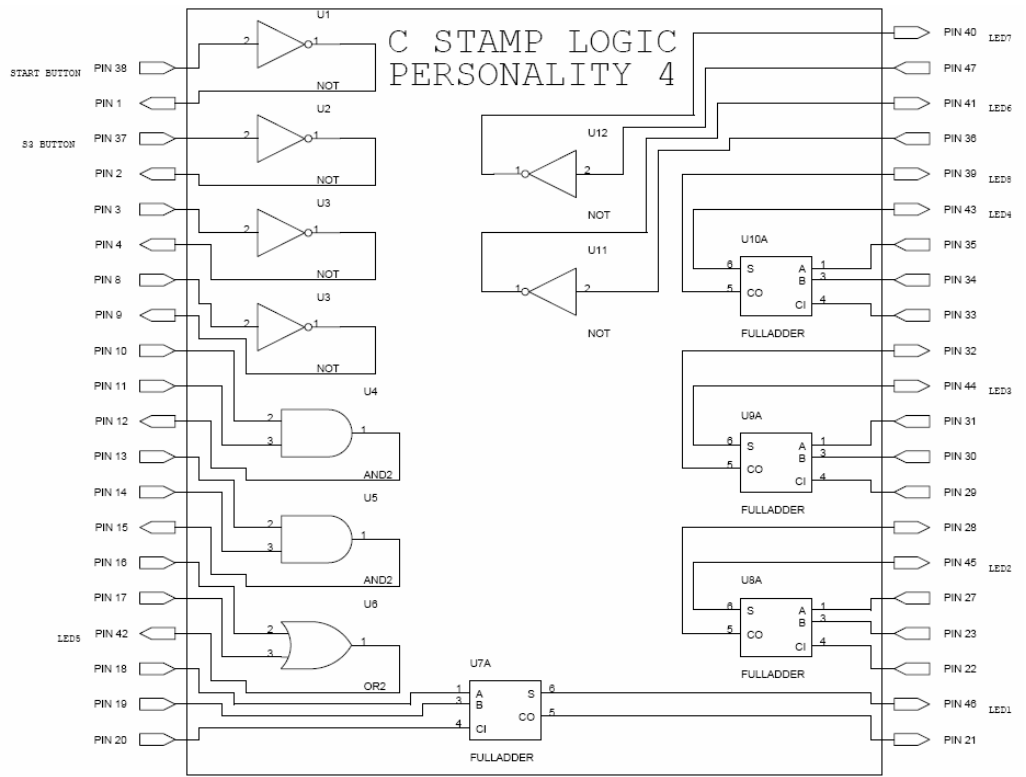
C Stamp Logic Personality 2



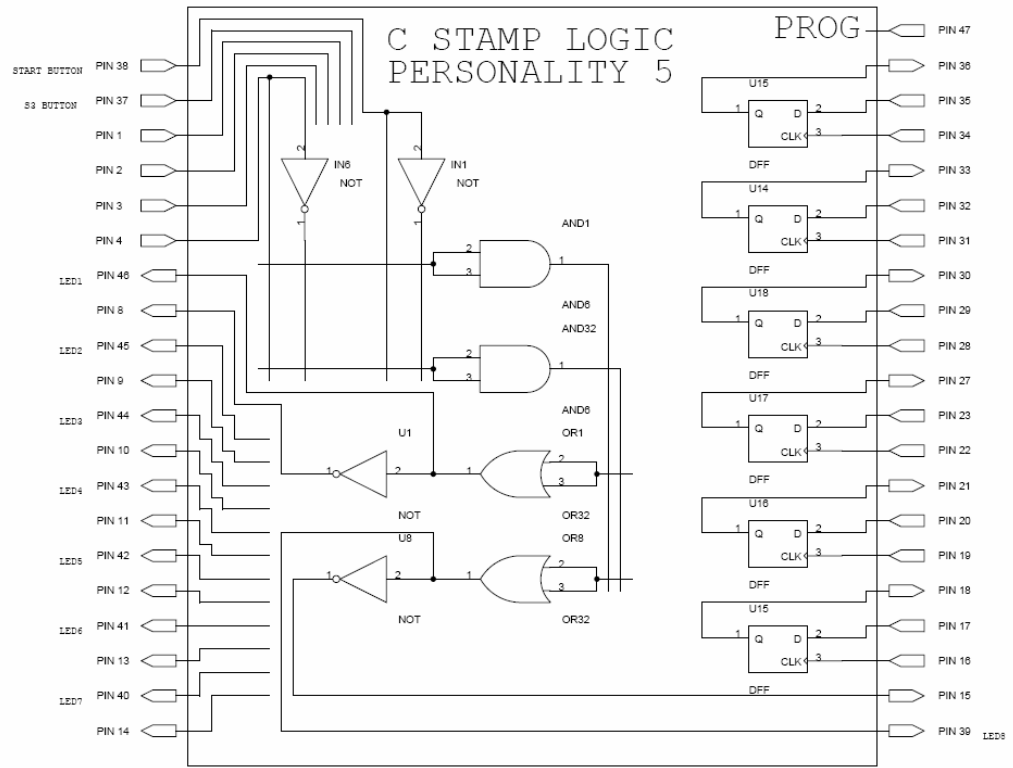
C Stamp Logic Personality 3



C Stamp Logic Personality 4



C Stamp Logic Personality 5



In LOGIC PERSONALITY 5, the C Stamp is a Programmable Logic Array (PLA) with 6 inputs and 8 complementary outputs. It also has 6 additional D flip-flops. The AND plane consists of 32 6-inputs AND gates, and each AND gate can AND the non-inverted or inverted version of each of the PLA inputs. The OR plane consists of 8 32-inputs OR gates that can OR the outputs of the 32 AND gates in the AND plane to generate the 8 outputs of the PLA. The template file to designate the AND plane connections and the OR plane connections is shown below.

```
# C STAMP LOGIC PERSONALITY 5 PROGRAMMING FILE
```

```
# PROJECT :
```

```
# AND PLANE
```

```
# PUT A 1 WHERE YOU WANT A CONNECTION
```

```
000000000000
```

```
000000000000
```

```
000000000000
```

```
000000000000
```

```
000000000000
```

```
000000000000
```


A - W I T T E C H N O L O G I E S I N C .

is in the same directory where the logic personalities are; in C:\A-
WIT\CSTAMP_Logic.

Terms and Conditions

Quality Assurance

A-WIT has stringent quality control procedures in place to insure the best quality products.

90-Day Limited Warranty

A-WIT Technologies, Inc warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, A-WIT Technologies, Inc. will, at its option, repair, replace, or refund the purchase price. After 90 days, products can still be sent in for repair or replacement, but there will be a \$10.00USD minimum inspection/labor/repair fee (not including return shipping and handling charges).

14-Day Money-Back Guarantee

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a refund. A-WIT will refund the purchase price of the product in the form of a check, excluding shipping/handling costs, once the product is received. This refund does not apply if the product has been altered or damaged. If you decide to return the products after the 14-day evaluation period, a 20% restocking fee will be charged against a credit.

Disclaimer

Warranty does not apply if the product has been altered, modified, or damaged. A-WIT makes no other warranty of any kind, expressed or implied, including any warranty of merchantability, fitness of the product for any particular purpose even if that purpose is known to A-WIT, or any warranty relating to patents, trademarks, copyrights or other intellectual property. A-WIT shall not be liable for any injury, loss, damage, or loss of profits resulting from the handling or use of the product shipped.

How to Return a Product

When returning, you must first e-mail sales@a-wit.com for a Return Merchandise Authorization number. No packages will be accepted without the RMA number clearly marked on the outside of the package. After inspecting and testing, we will return your product, or its replacement using the same shipping method used to ship the product to A-WIT within 30 days. In your package, please include a daytime telephone number and a brief explanation of the problem.

Please contact our Sales Department at sales@a-wit.com if you have any questions regarding our warranty policy or if you are requesting an RMA number.

