Project Lab 1 – Techno-Modern Clock

Purpose

This project lab will give the student the experience of designing, writing, implementing, debugging and documenting their own program. The project will provide an understanding of how to solve almost any kind of project and/or problem from a written description. The project will also test the understanding of the PLC concepts and instructions learned to date.

Completing this lab the student will be able to

1. design, write, implement, debug and document a program from a written process description.

Instructions

1. It is best to sketch your ideas on paper first. Brainstorm some ideas on paper before going to the computer. Break the project down into small manageable sections. Test each section for correct functionality before moving on to the next part of the problem.
2. Fully document the program. Do not describe how the instructions work, describe what purpose and/or function they have in the solution to the project.
3. This project will be done in the LogixPro Simulator as a homework assignment.

Project Description

You have been hired by one of your clients to design what your client is calling a techno-modern clock. The client wants the clock to keep track of seconds, minutes, hours (in 12-hour increments) and provide a display of AM and PM. An indicator light should blink at a rate of ½-second on / ½-second off to know that the clock is running and that seconds are accumulating. The minutes of each hour should be displayed in binary using six indicator lights. The hour of each half day should be displayed in binary using four indicator lights. Another indicator should tell the user if it is AM or PM. The AM/PM indicator should be on for PM and off for AM. AM starts at 12:00 midnight and PM starts at 12:00 noon. The clock should run with no interaction from the user other than placing the PLC in run mode. The clock should be capable of being set to the correct minute and hour of the day. This can be done by entering the correct time in words in the data tables.

How to Approach Solving a Project

The project description above may seem over-whelming and you might be thinking that you'll never be able to solve this problem. Well…that's just not true. Whether you believe it or not, you currently have enough knowledge to solve this problem.

This is most likely the first time you are designing a project from scratch using just a written description as in the project description section above. Whenever a project of any kind or any size is to be done it is best to break the project down into small, manageable pieces and then solve each one of the smaller pieces. By the time each small piece has been solved and working, the pieces should have been assembled into a fully functional project. It's like building a house one block at a time until the home is complete or an electrical contractor bidding on the installation of service in a large industrial building and breaking down the project by floor or by rooms then taking all the individual data and assembling it into a full working quotation along with assigning workers to perform each task.

How a project is broken down is up to the individual solving the project problem. There are as many methods, styles and solutions as there are stars in the sky, therefore, just because you're approach is different than your classmate doesn't make either of you right or wrong.

Following is an example of how this project can be broken down into small manageable sections. You will be responsible for making each section work, but the steps given in this document should give you a good idea of how to tackle almost any project or problem.

Getting Started

Start by carefully reading the project description and extracting the details of the functionality to accomplish. As an example we'll show the project description again and underline the possible steps toward a solution:

You have been hired by one of your clients to design what your client is calling a techno-modern clock. The client wants the clock to keep track of seconds, minutes, hours (in 12-hour increments) and provide a display of AM and PM. <u>An indicator light should blink at a rate of ½-second on / ½-second off</u> to know that the clock is running and that seconds are accumulating. The <u>minutes of each hour should be displayed in binary using six indicator lights</u>. The <u>hour of each half day should be displayed in binary using four indicator lights</u>. <u>Another indicator should tell the user if it is AM or PM</u>. The <u>AM/PM indicator should be on for PM and off for AM. AM starts at 12:00 midnight and PM starts at 12:00 noon</u>. The clock should <u>run with no interaction from the user</u> other than placing the PLC in run mode. The clock should be <u>capable of being set to the correct minute and hour of the day</u>. This can be done by entering the correct time in words in the data tables.

Underlining is used here to emphasize what at first seems to be important information to solving the problem. Write out each one in a list that will start forming an activities list.

1. Indicator should blink ½ second ON and ½ second OFF.
2. Minutes need to be displayed in binary. This means the program must be capable of accurately timing or counting minutes and in a form that can be represented in binary. Six indicators are required because it will take 6-bits to represent the minutes from 1 to 59.
3. Hours need to be displayed in binary, (could be similar to the way minutes are being displayed). This also means that hours need to be accurately timed or counted. Only a half day (12-hours) needs to be timed or counted. Four indicators are required because it will take 4-bits to represent the hours from 1 to 12.
4. An indicator is required to display AM and PM. AM starts at 12:00 midnight and PM starts at 12:00 noon. This means a method must be derived to keep track of half days.
5. Clock needs to be set to the correct time.

The list now states five steps that can be implemented on-at-a-time to form a complete, fully functional project. Each operation seems to have been broken down in a manageable piece. Steps 2 and 3 could be further broken down if so desired as follows:

1. Indicator should blink ½ second ON and ½ second OFF.
2. Minutes need to be timed or counted based on the number of seconds.
3. Minutes need to be displayed in binary using six indicators. Six indicators are required because it will take 6-bits to represent the minutes from 1 to 59.
4. Hours need to be timed or counted based on the number of minutes.
5. Hours need to be displayed in binary, (could be similar to the way minutes are being displayed). Only a half day (12-hours) needs to be timed or counted. Four indicators are required because it will take 4-bits to represent the hours from 1 to 12.

6. An indicator is required to display AM and PM. AM starts at 12:00 midnight and PM starts at 12:00 noon. This means a method must be derived to keep track of half days.
7. Clock needs to be set to the correct time.

This new activity list might be more manageable because each step has less to solve. These activity lists are only a starting point and they are "living-breathing" document. It is perfectly normal and is usually very common to make modification to the activity list as the project is being solved. Like anything in life, as you do something you find better or different ways of performing the task. The idea is to have a fully functional project when finished.

Now arrange the activities list in a solution order that makes sense. The way these lists are arranged seems to make sense. Since the finished project is a clock we already know that there are 60-seconds in a minute, 60-minutes in an hour and 24-hours in a day. Therefore it makes sense that the solution should start by timing or counting seconds, then minutes, then hours.

After completing each step, download the code and run and test the operation of that step. Make sure that the solution for that step completely works before proceeding to the next step. Sometimes it might be necessary to make modifications to previously working steps to make another step and the project function correctly. This is normal and probable.

It's now time for you to start solving this project problem. You can use one of the activity lists we created or you can create one of your own. The bottom line is that the entire project should correctly and completely work when the project is finished.

Make sure to document the program. Documentation is something that most people do not like to do, but by doing it and doing it accurately you will thank yourself in the future if you ever have to revisit the project and other users will thank you for providing them with information about the functionality. One of the hardest tasks there is, especially for maintenance people, is reading program code or schematics and drawings that someone else wrote or created and there is no documentation to guide you. Get into the habit of providing that needed documentation. If you document things in a way that you would want someone else to document for you, you will probably be providing good documentation.

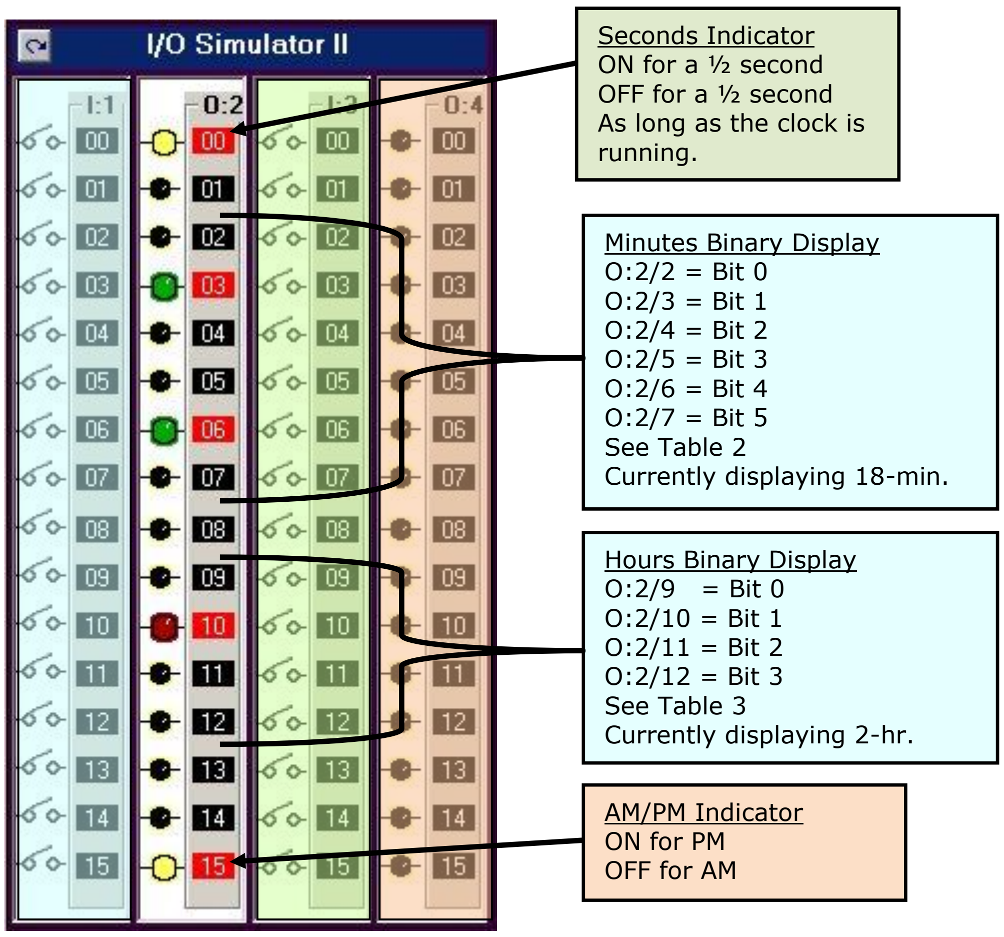
I/O For This Project

You will be using the LogixPro Simulator software to solve this project and using the I/O simulator that is built into LogixPro. The client as already assigned the I/O he wants you to use. Refer to Table 1 for a list of the I/O (Note that there are no input devices used because there is no user interaction except to switch the PLC from Program mode to Run mode and to set the hours and minutes).

Figure 1 shows the LogixPro Simulator with the outputs identified. Table 2 and Table 3 describes the minutes and hours binary count.

Table 1

I/O Address	Function
O:2/0	Seconds indictor
O:2/2	Bit 0 of the minutes binary count
O:2/3	Bit 1 of the minutes binary count
O:2/4	Bit 2 of the minutes binary count
O:2/5	Bit 3 of the minutes binary count
O:2/6	Bit 4 of the minutes binary count
O:2/7	Bit 5 of the minutes binary count
O:2/9	Bit 0 of the hours binary count
O:2/10	Bit 1 of the hours binary count
O:2/11	Bit 2 of the hours binary count
O:2/12	Bit 3 of the hours binary count
O:2/15	AM/PM Indicator



Seconds Indicator
ON for a ½ second
OFF for a ½ second
As long as the clock is running.

Minutes Binary Display
O:2/2 = Bit 0
O:2/3 = Bit 1
O:2/4 = Bit 2
O:2/5 = Bit 3
O:2/6 = Bit 4
O:2/7 = Bit 5
See Table 2
Currently displaying 18-min.

Hours Binary Display
O:2/9 = Bit 0
O:2/10 = Bit 1
O:2/11 = Bit 2
O:2/12 = Bit 3
See Table 3
Currently displaying 2-hr.

AM/PM Indicator
ON for PM
OFF for AM

Figure 1 – Time shown is 2:18 PM

Minutes Binary Display, (1 = Indicator ON, 0 = Indicator OFF)

Table 2

Minutes	O:2/7	O:2/6	O:2/5	O:2/4	O:2/3	O:2/2
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0
5	0	0	0	1	0	1
6	0	0	0	1	1	0
7	0	0	0	1	1	1
8	0	0	1	0	0	0
9	0	0	1	0	0	1
10 to 58	Continue the binary count from 10 to 58					
59	1	1	1	0	1	1

Hours Binary Display, (1 = Indicator ON, 0 = Indicator OFF)

Table 3

Hours	O:2/12	O:2/11	O:2/10	O:2/9
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

What You Should Be Submitting

You should be submitting a fully documented, fully functional techno-modern clock program.

Name the file using the naming structure: <lastname>_Techno-modern. As an example student John Doe would name his file: Doe_Techno-modern. LogixPro allows for saving a file as a LogixPro file or as an image. Be sure to submit the LogixPro program file and not an image file.

Submit the file to the appropriate assignment area in Blackboard.

HINT For Solving This Problem

1. Every address in the Allen Bradley memory structure can be addressed to word and to bit level.
2. Everything "under the hood" is already in binary.