

21. FUNCTION BLOCK PROGRAMMING

Topics:

- The basic construction of FBDs
- The relationship between ST and FBDs
- Constructing function blocks with structured text
- Design case

Objectives:

- To be able to write simple FBD programs

21.1 INTRODUCTION

Function Block Diagrams (FBDs) are another part of the IEC 61131-3 standard. The primary concept behind a FBD is data flow. In these types of programs the values flow from the inputs to the outputs, through function blocks. A sample FBD is shown in Figure 299. In this program the inputs A and B are used to calculate a value $\sin(A) * \ln(B)$. The result of this calculation is compared to C . If the calculated value is less than C then the output X is turned on, otherwise it is turned off. Many readers will note the similarity of the program to block diagrams for control systems.

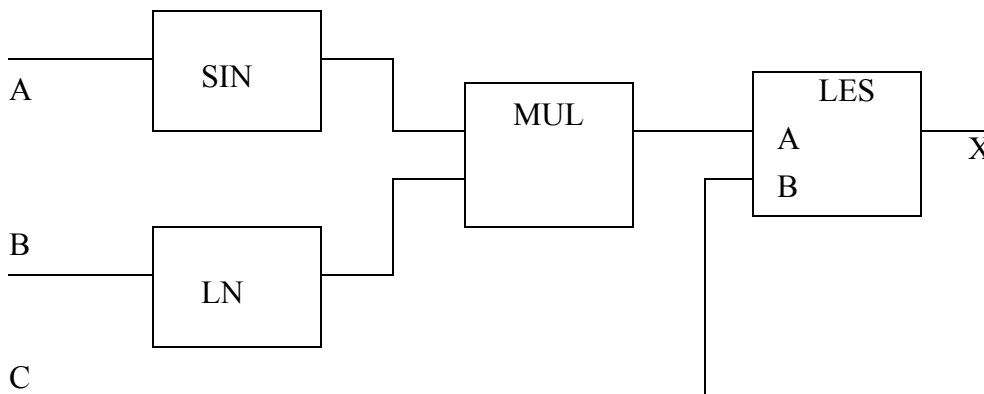


Figure 299 A Simple Calculation and Comparison Program

It is possible to disable part of the FBDs using enables. These are available for each function block but may not be displayed. Figure 300 shows an XOR calculation. Both of the Boolean AND functions have the enable inputs connected to 'enable'. If 'enable' is true, then the system works as expected and the output 'X' is the exclusive OR of 'A' and 'B'. However if 'enable' is off then the BAND functions will not operate. In this case the 'enable' input is not connected to the BOR function, but because it relies on the outputs from the BAND blocks, it will not function, and the output 'X' will not be changed.

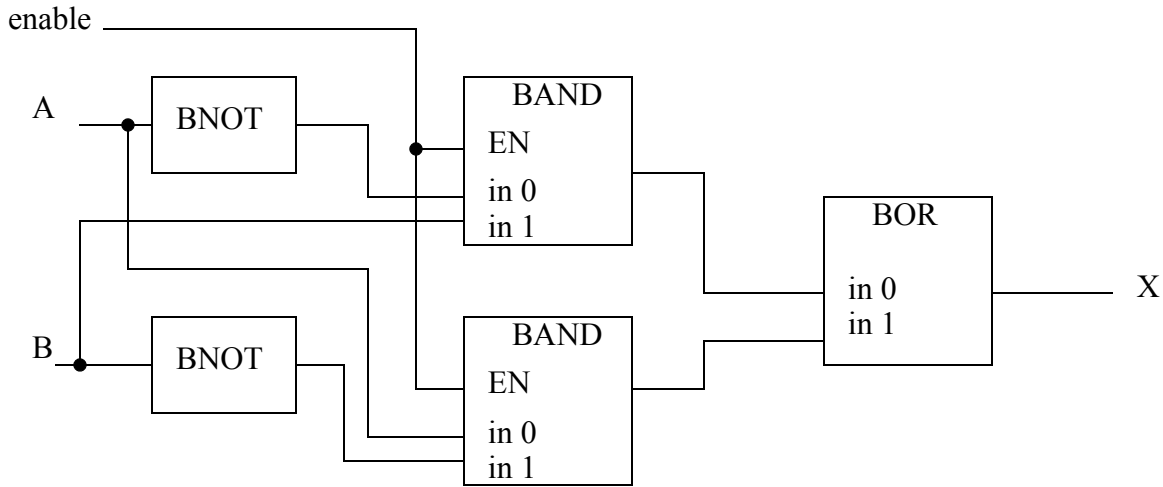


Figure 300 Using Enables in FBDs

A FBD program is constructed using function blocks that are connected together to define the data exchange. The connecting lines will have a data type that must be compatible on both ends. The inputs and outputs of function blocks can be inverted. This is normally shown with a small circle at the point where the line touches the function block, as shown in Figure 301. (Note: this is NOT available for Allen Bradley RSLogix, so BNOT functions should be used instead.)

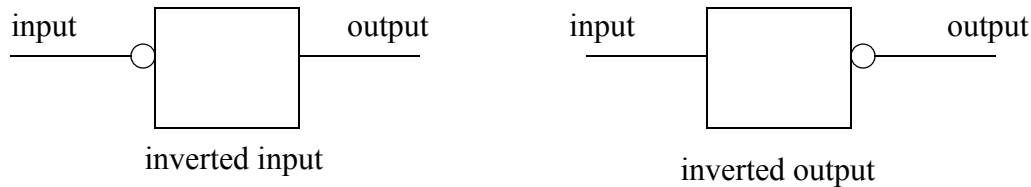


Figure 301 Inverting Inputs and Outputs on Function Blocks

The basic functions used in FBD programs are equivalent to the basic set used in Structured Text (ST) programs. Consider the basic addition function shown in Figure 302. The ST function on the left adds A and B , and stores the result in O . The function block on the right is equivalent. By convention the inputs are on the left of the function blocks, and the outputs on the right.

Structural Text Function	Function Block Equivalent
$O := \text{ADD}(A, B);$	

Figure 302 A Simple Function Block

Some functions allow a variable number of arguments. In Figure 303 there is a third value input to the *ADD* block. This is known as overloading.

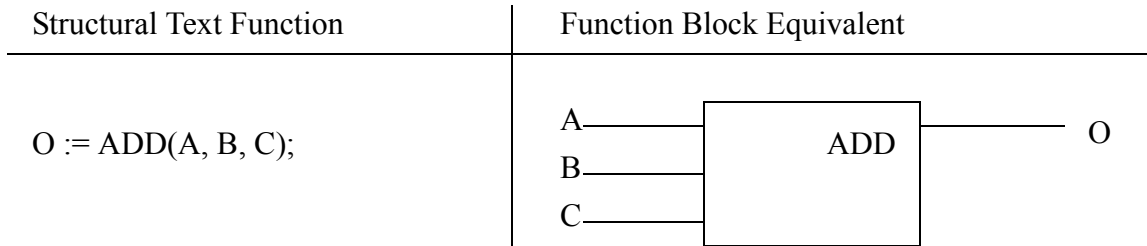


Figure 303 A Function with A Variable Argument List

The *ADD* function in the previous example will add all of the arguments in any order and get the same result, but other functions are more particular. Consider the circular limit function shown in Figure 304. In the first ST function the maximum *MX*, minimum *MN* and test *IN* values are all used. In the second function the *MX* value is not defined and will default to 0. Both of the ST functions relate directly to the function blocks on the right side of the figure.

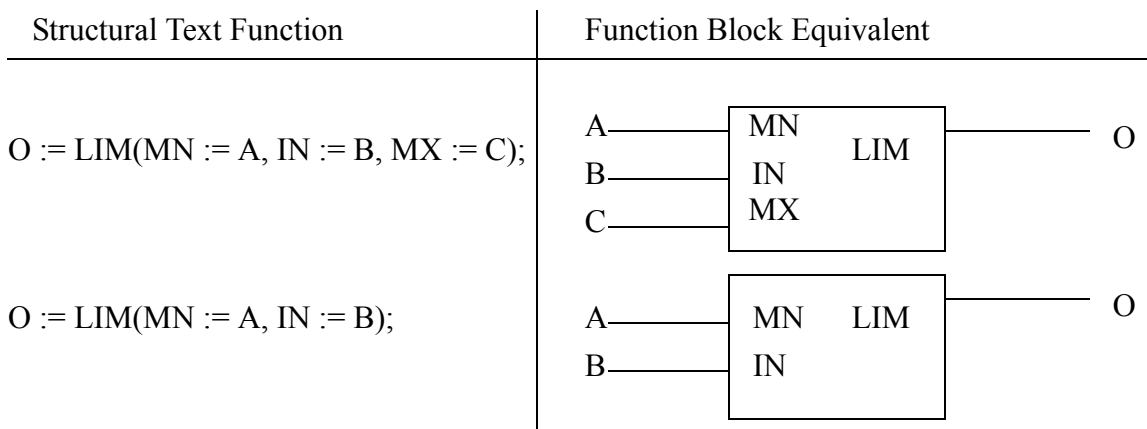


Figure 304 Function Argument Lists

21.2 CREATING FUNCTION BLOCKS

When developing a complex system it is desirable to create additional function blocks. This can be done with other FBDs, or using other IEC 61131-3 program types. Figure 305 shows a divide function block created using ST. In this example the first statement declares it as a *FUNCTION_BLOCK* called *divide*. The input variables *a* and *b*, and the output variable *c* are declared. In the function the

denominator is checked to make sure it is not 0. If not, the division will be performed, otherwise the output will be zero.

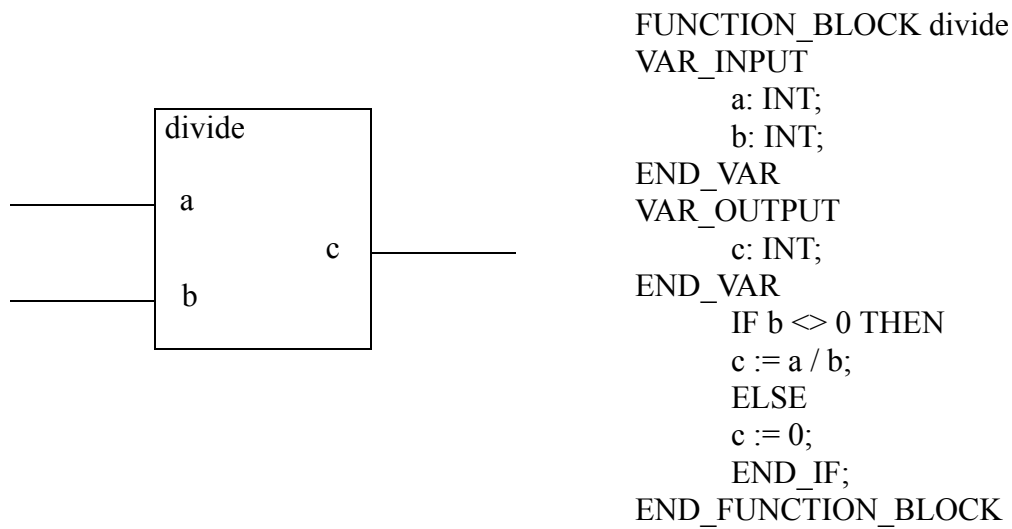


Figure 305 Function Block Equivalencies

21.3 DESIGN CASE

A simple state diagram is shown in Figure 306.

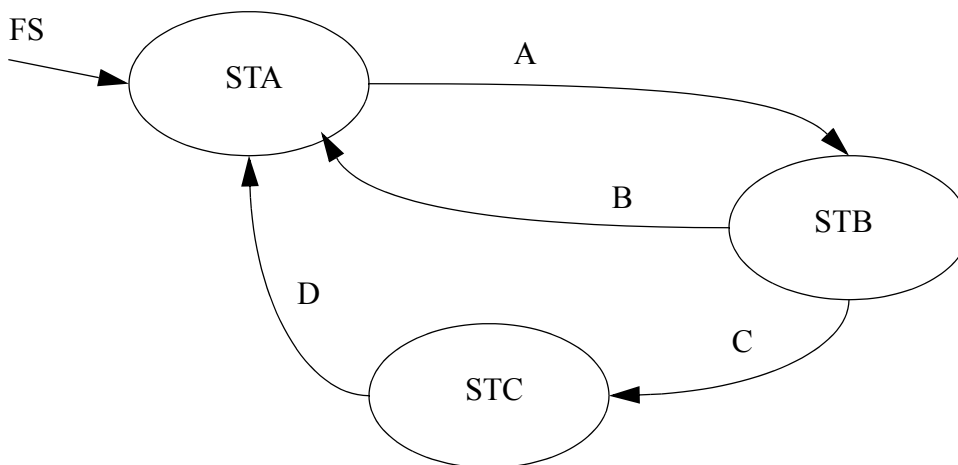


Figure 306 An Example State Diagram

The state diagram is implemented in FBD form in Figure 307. In this case the transition equations approach was used, although other methods are equally applicable. The transitions 'STA_TO_STB', 'STB_TO_STA', 'STB_TO_STC', and 'STC_TO_STA' are calculated first. These

are then used to update the states 'STA', 'STB', and 'STC'. Additional program steps could then be added to drive outputs.

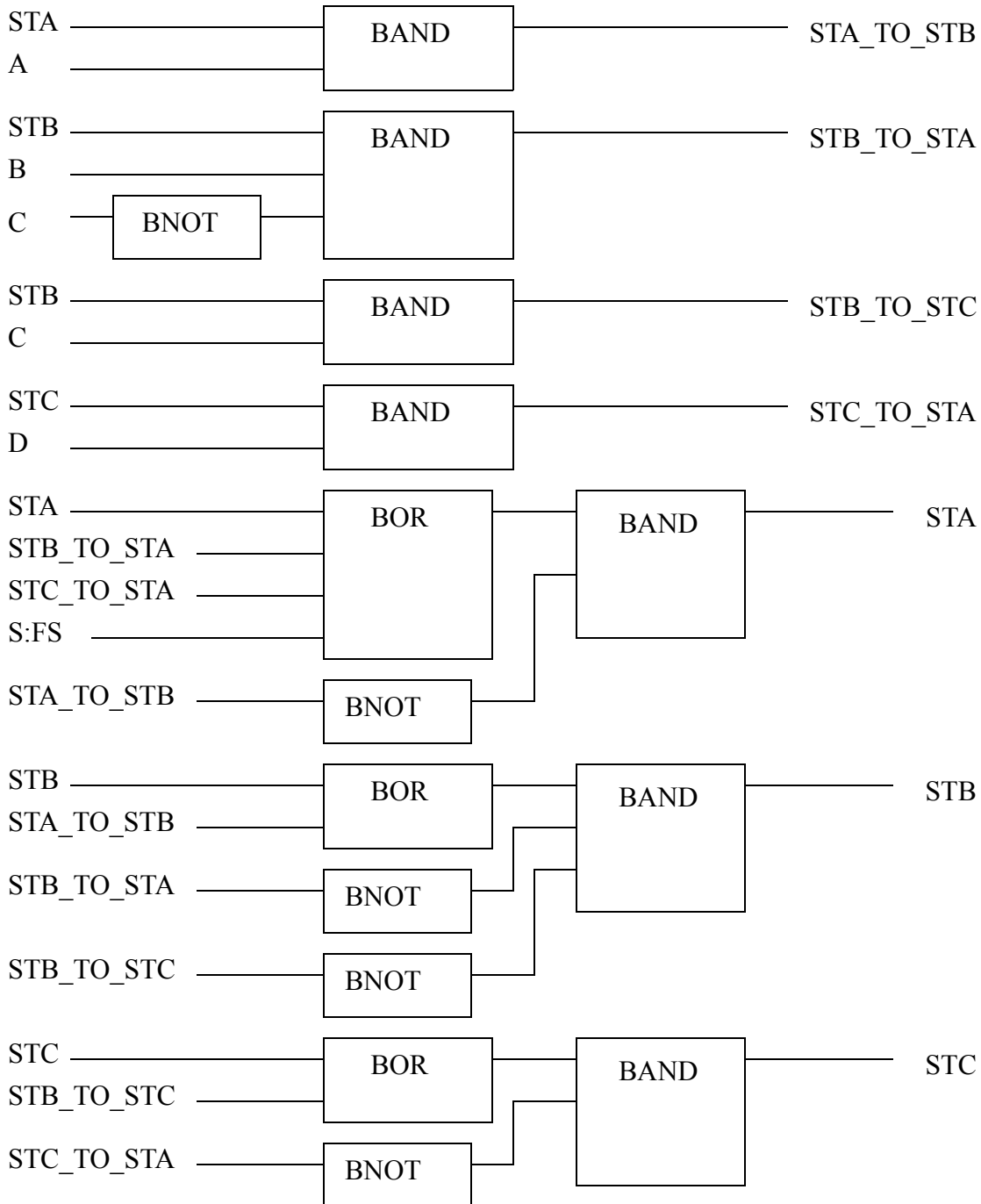


Figure 307 An FBD Implementation of a State Diagram Using Transition Equations

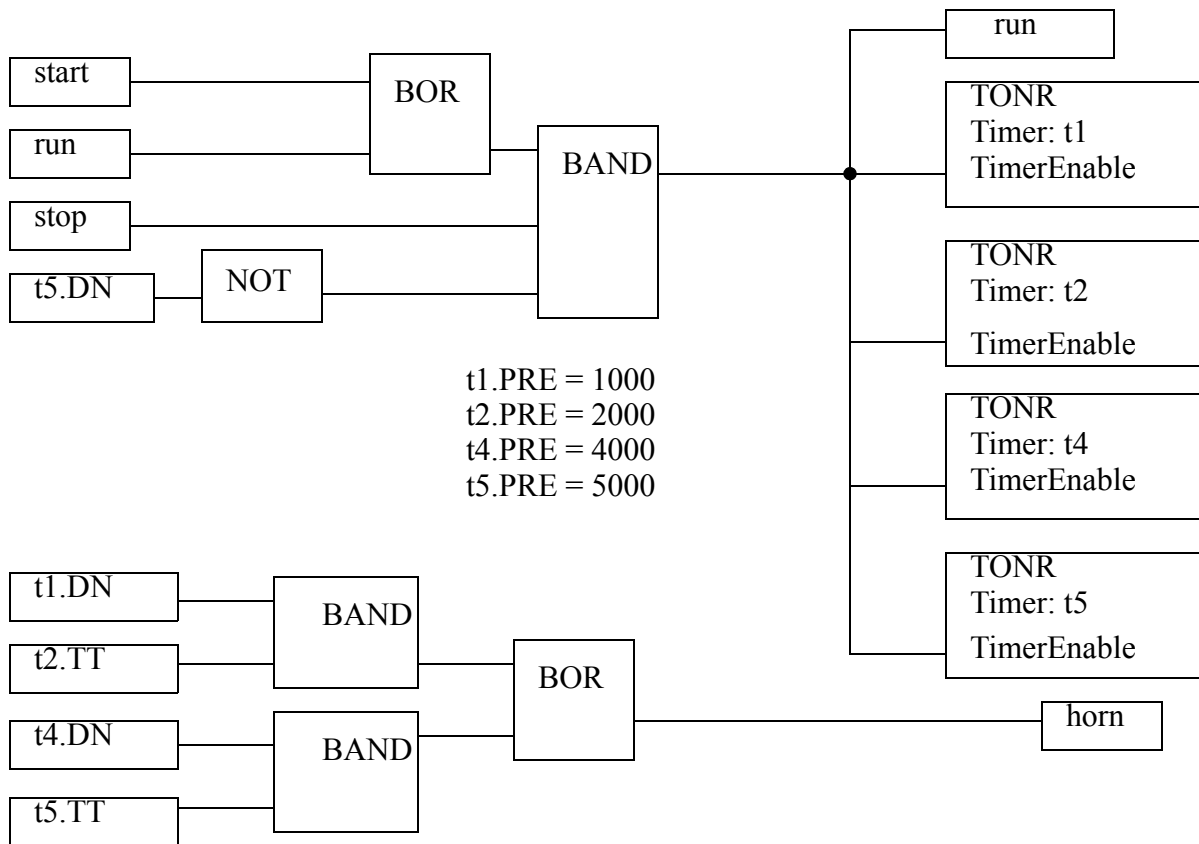
21.4 SUMMARY

- FBDs use data flow from left to right through function blocks

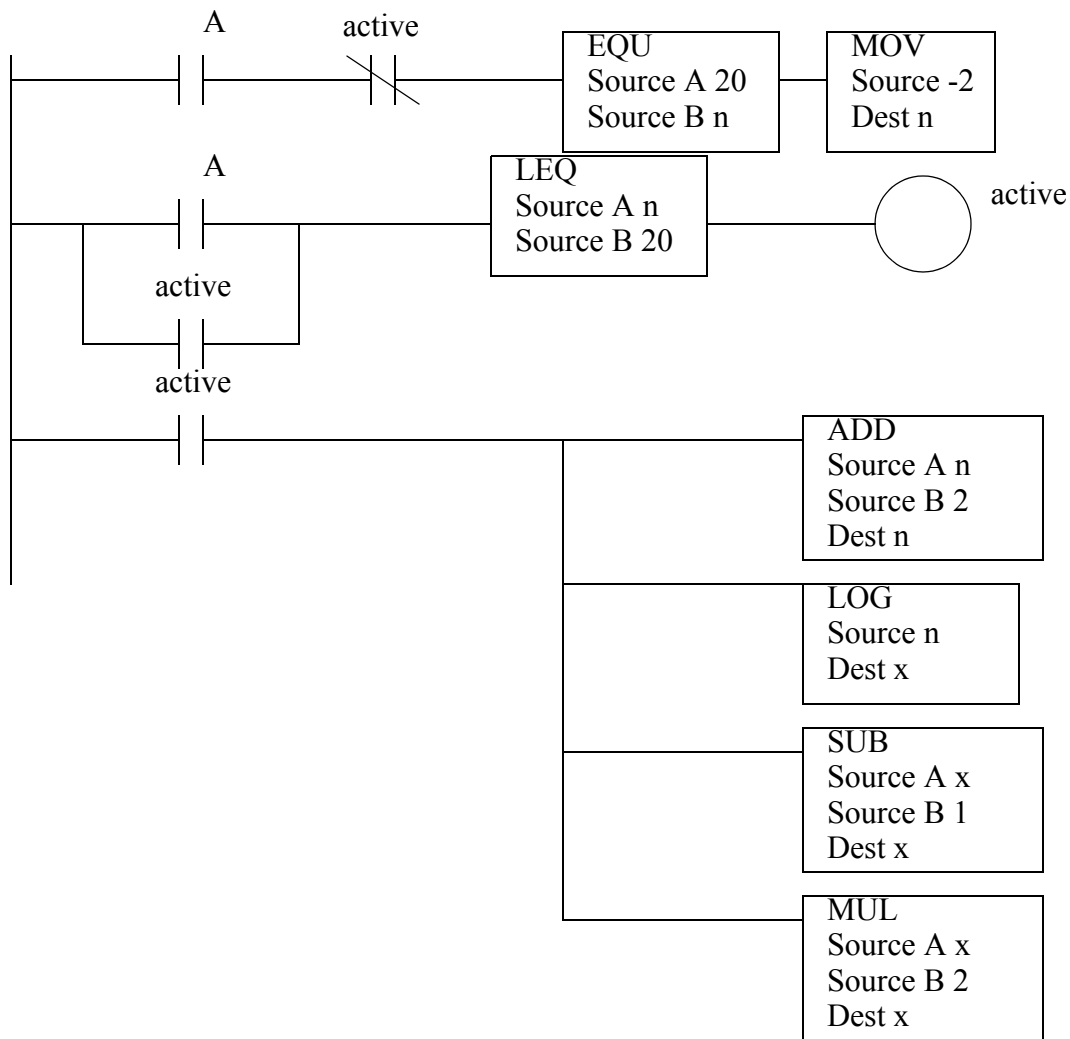
- Inputs and outputs can be inverted
- Function blocks can have variable argument list sizes
- When arguments are left off default values are used
- Function blocks can be created with ST

21.5 PRACTICE PROBLEMS

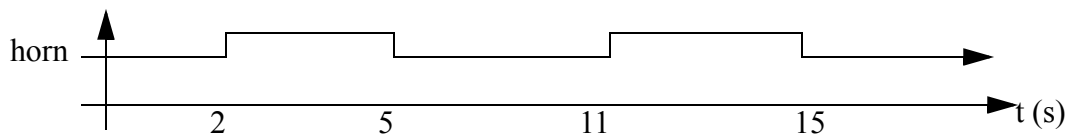
1. Draw a timing diagram for the following FBD program.



2. Write a function block diagram program that will replace the following ladder logic.

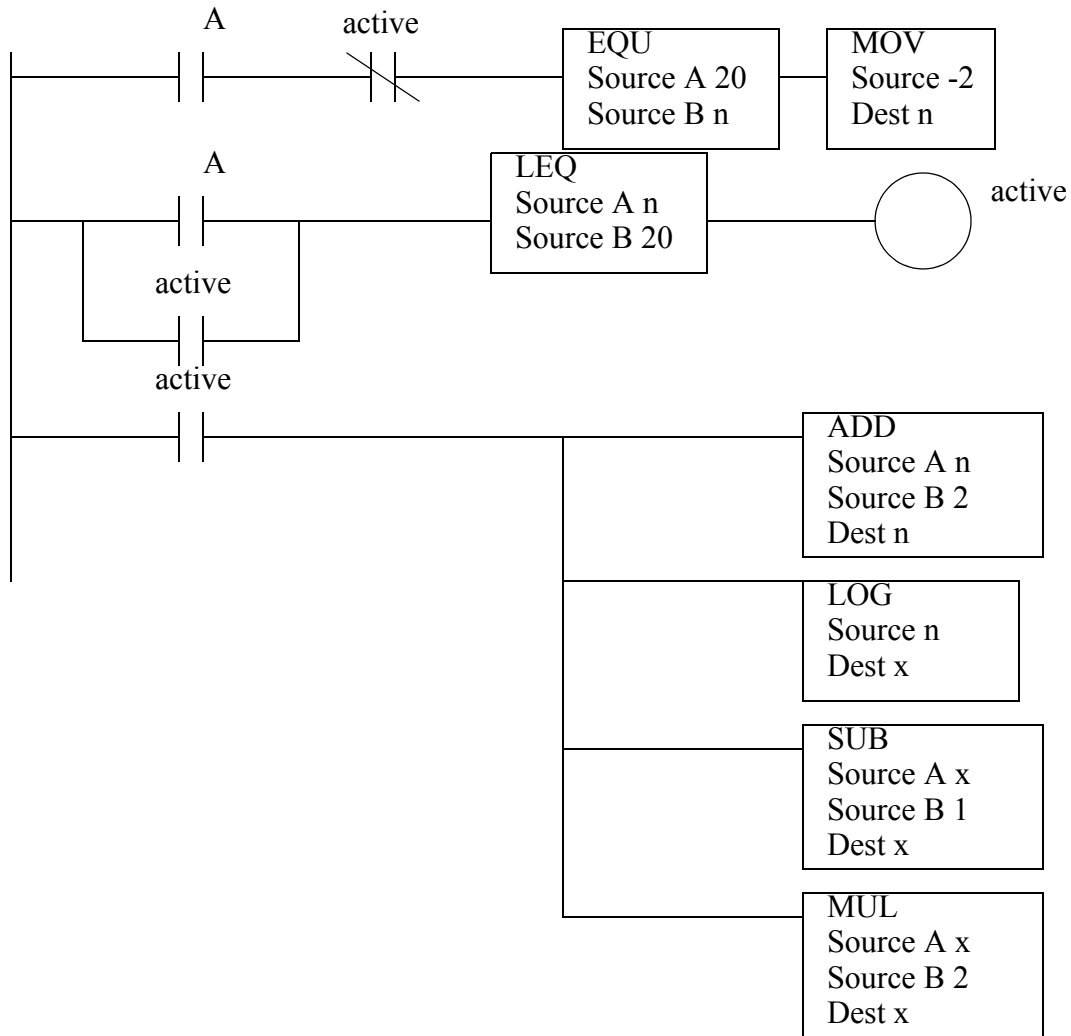


3. Write a Function Block Diagram program to implement the following timing diagram. The sequence should begin when a variable 'temp' rises above 80.



4. Develop a FBD for a system that will monitor a high temperature salt bath. The systems has *start* and *stop* buttons as normal. The temperature for the salt bath is available in *temp*. If the bath is above 250 C then the *heater* should be turned off. If the temperature is below 220 C then the *heater* should be turned on. Once the system has been in the acceptable range for 10 minutes the system should shut off.

5. Write a function block diagram program that will replace the following ladder logic.



6. Write a structured text program that reads inputs from 'channel 0'. An input string of 'CLEAR' will clear a storage array. Up to 100 real values with the format 'XXX.XX' will arrive on 'channel 0' and are to be stored in the array. If the string 'AVG' is received, the average of the array contents will be calculated and written out 'Channel 0'.

21.6 ASSIGNMENT PROBLEMS

1. Convert the following state diagram to a Function Block Diagram.

