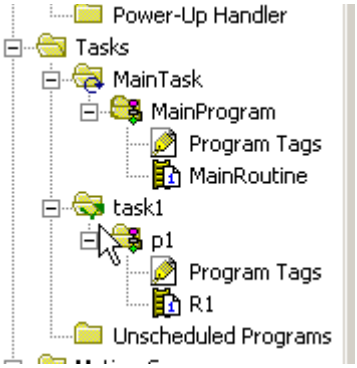
Module 6



Routines

Student Materials

Module 6:
Routines

Introduction:

When working with a CompactLogix / ControlLogix PLCs, a download takes a project file stored on the programming panel (computer) and puts it in the PLC. A controller can hold one project at a time. A common term many people used when describing a project is a program. At times it can be confusing when using the term program in place of the term project, since a project is made-up of task(s), program(s) and routine(s).

A project must have at least one Task, one Program and one Routine. Routines are organized within Programs. Routine locations are shown in the Controller Organizer window.

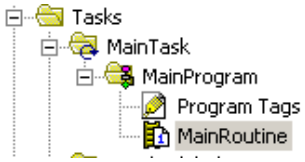


Figure 1-A MainRoutine within MainProgram

There is no hard limit as to the number of Routines per Program. There are three types of routines in Logix5000 processors, Main, Fault and subroutines.

By default a project will have one Task (MainTask - continuous), one Program (MainProgram) and one Routine (MainRoutine). Routines contain the processor instructions. By default Routines are programmed using Ladder Logic.

Note: Besides Ladder Logic, Routines can also programmed in Structured Text, Function Block Diagrams or Sequential Function Chart.

Tasks, programs and routines can be renamed. Tasks, programs and routines have properties. The Properties window can be opened by right clicking the routine name and choosing Properties.

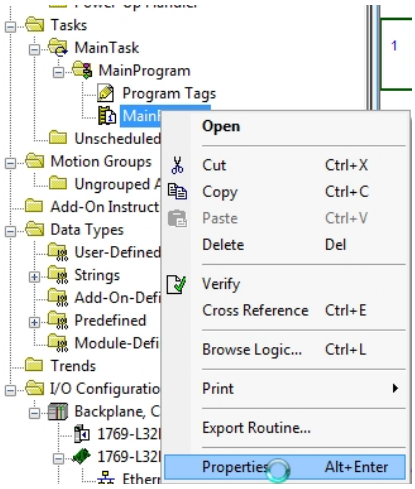


Figure 2-A- Selecting Routine Properties

The Properties window for Routines show in following information:
Name
Description
Type: how the Routine is programmed

-  Icons for Ladder Diagram Routines

In Program: the name of the Program that contains the Routine
Number of Rungs: if the Routine is programmed in Ladder Logic (Ladder Diagram)

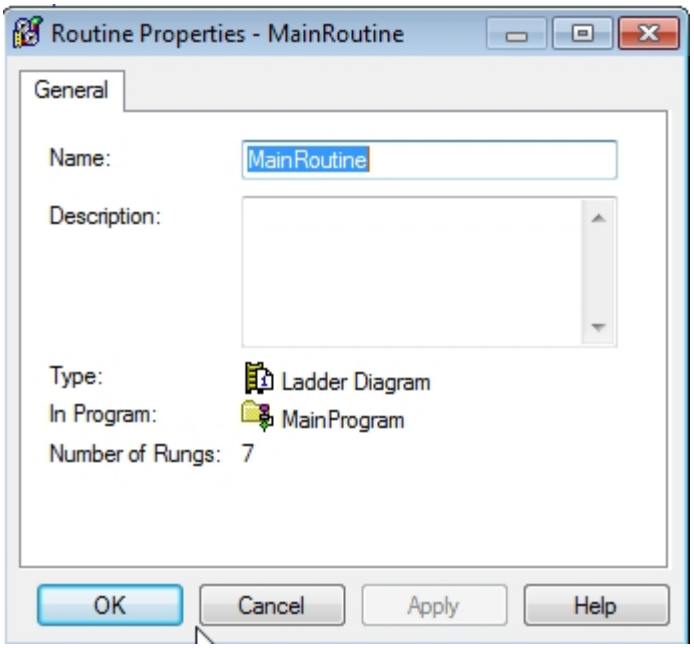


Figure 3-A – Routine Properties window for Ladder Logic

Note: If a Routine is programmed in a different format the Number of Rungs appears as:

Number of Steps: If the Routine is programmed with Sequential Function Charts

 Icon for Sequential Function Charts

Number of Sheets: If the Routine is programmed with Function Blocks Diagram

Icon for Function Block Diagram

Number of Lines: If the Routine is programmed with Structured Text

Icon for Structured Text

Creating Routines:

In the Controller Organizer window, right mouse click the Program in which the Routine will be added.

Click the New Routine selection

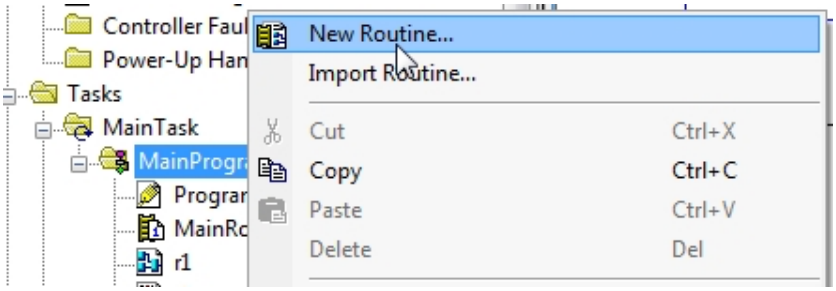


Figure 4-A – Create New Routine

In the New Routine window:
Name: name of Routine
Type: Choose how the routine will be programmed

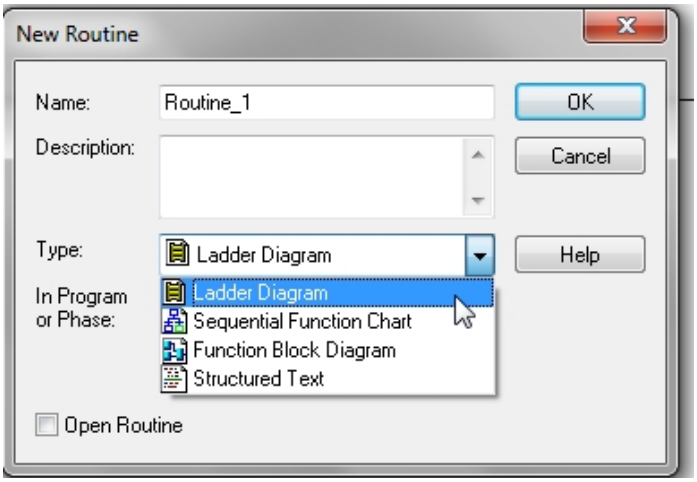


Figure 5-A – Name and Type of New Routine

In Program
or Phase: Select Program in which the Routine will be located

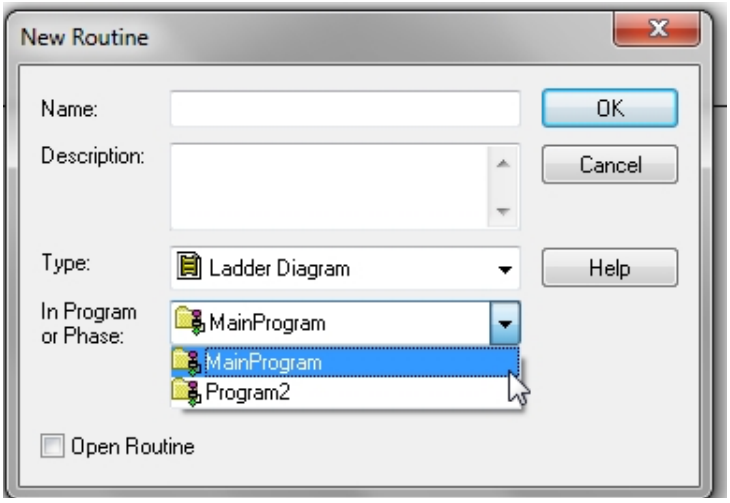


Figure 6-A – Choose Routine Location

Click the OK button to create the Routine.

Types of Routines

There are three types of Ladder Logic (Diagram) Routines

1. Main – first Routine executed (scanned) within a Program – required
   One per Program

 - Icon for Main routine

2. Fault – a Routine that will execute if a recoverable Major fault occurs within any of the routines within a Program – optional
   One per Program

 - Icon for Fault routine

3. Subroutine – a Routine that will execute when called using a JSR (Jump to Subroutine) instruction - optional
   No limit restrictions

 - Icon for Subroutine

To configure a Routine - Right click on the Program that contains a Routine

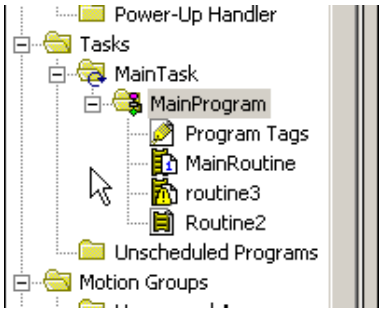


Figure 7-A. Right click on the MainProgram.

From the selection menu, choose Properties

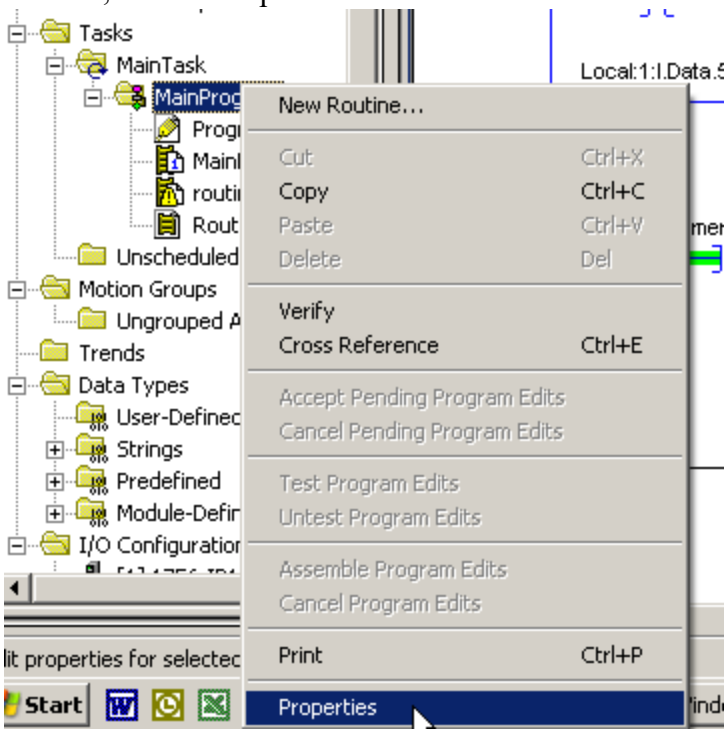


Figure 8-A. Choose Properties of the MainProgram.

From the Program's Properties window, choose the Configuration tab.
See Figure 9-A.

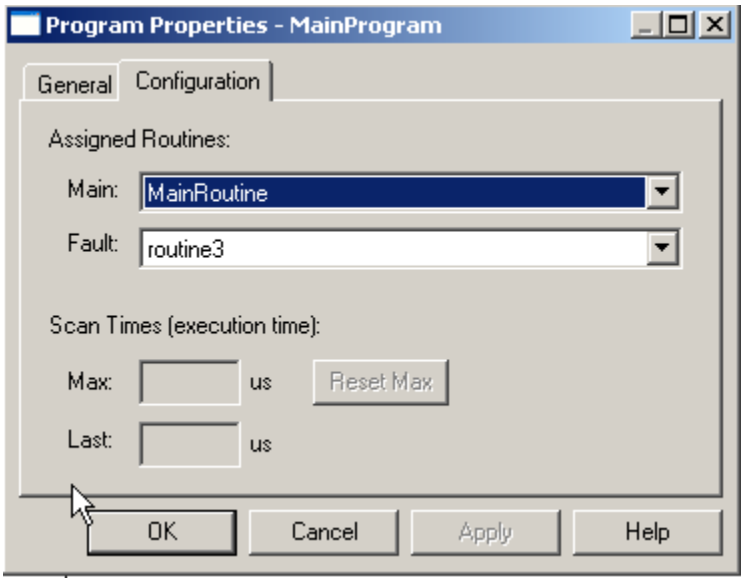


Figure 9-A – Program Configuration tab

In the Configuration tab:
Use the Main Selection box to assign the Main Routine.
Use the Fault Selection box to assign a Fault Routine
Any routines not configured in the Selection boxes are Subroutines

Note: Routine names alone do not configure the routine type.

From the Controller Organizer window a Main routine is shown as:

Figure 10-A – Main Routine Icon

A Fault routine is shown as:

Figure 11-A – Fault Routine Icon

A Main routine is the first routine that runs within a Program.
A Fault routine is the last routine that is run within a Program if a recoverable Major fault has occurred within the Program's routines.

Note: Each Program must have a Main Routine
Each Program can have its own Fault Routine

Other routines are accessed by the use of the JSR (Jump To Subroutine) instruction.

Subroutines:

A subroutine is executed by a JSR instruction.

The JSR instruction is located in the Program Control tab on the Language Toolbar.

Figure 12-A – JSR Instruction - Language Toolbar

The JSR instruction rung can have conditions (input instructions) or the rung can be unconditional (no input Instructions).

When the JSR instruction is True the processor jumps to Rung 0 of the routine named in the instruction (subroutine) and begins execution (scanning) of that routine.

Once the subroutine scan is completed, the processor returns to the JSR instruction. The PLC continues executing (scanning) rungs after the JSR rung in the routine that contains the JSR instruction.

The END Rung of the subroutine signals the processor to return to the JSR instruction.

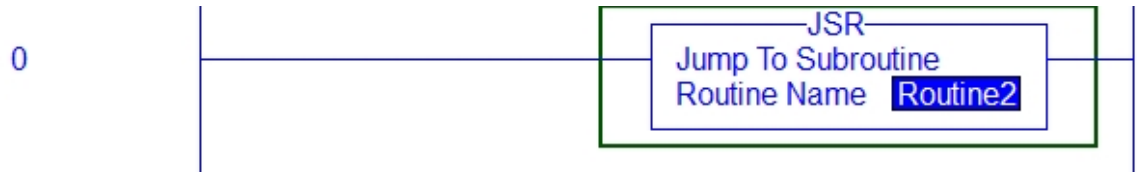


Figure 13-A – JSR Instruction Ladder Rung

In its simplest format that is the scanning process of the PLC and a subroutine.

Two additional instructions that maybe used in subroutines are:
SBR - Subroutine
RET – Return from Subroutine

The SBR and RET instructions are located in the Program Control tab on the Language Toolbar.

Figure 14-A – RET and SBR Instructions - Language Toolbar

When a JSR instruction is configured not to pass input and / or return parameters between routines SBR and RET instructions are optional in a subroutine.

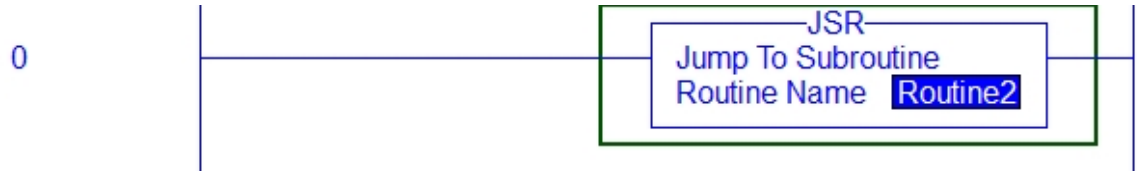


Figure 15-A – JSR Instruction configured not to pass parameters

If a SBR instruction is used, the instruction must be the first (left most) instruction on Rung 0 of the subroutine. The processor evaluates the SBR as True.

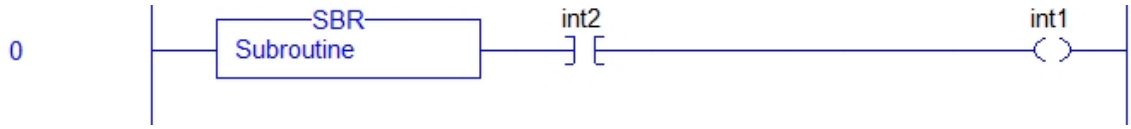


Figure 16-A
SBR Instruction configured not to pass parameters

A RET instruction, if used, signals the processor to exit the subroutine and Return to the JSR instuction.

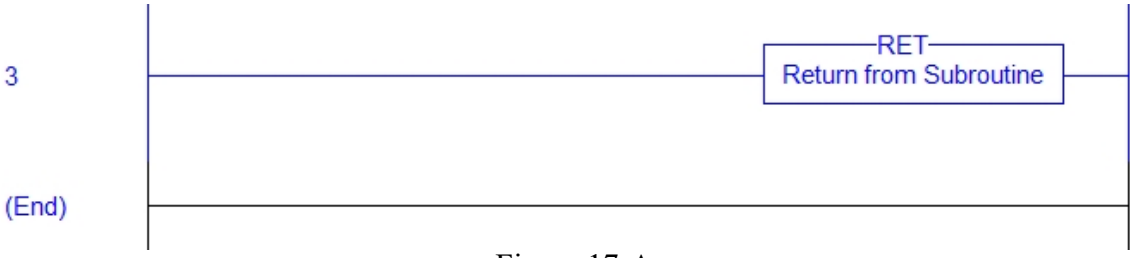


Figure 17-A
RET Instruction configured not to pass parameters, no conditions

A Rung with an unconditional RET instruction performs the same operation as a subroutine END Rung.

Ladder Diagram subroutines can have more than one RET Rung.
Rungs can be configured with conditional instructions that cause the processor to exit the subroutine before the entire subroutine is executed if certain conditions are True.

Figure 18-A shows a conditional RET Rung.

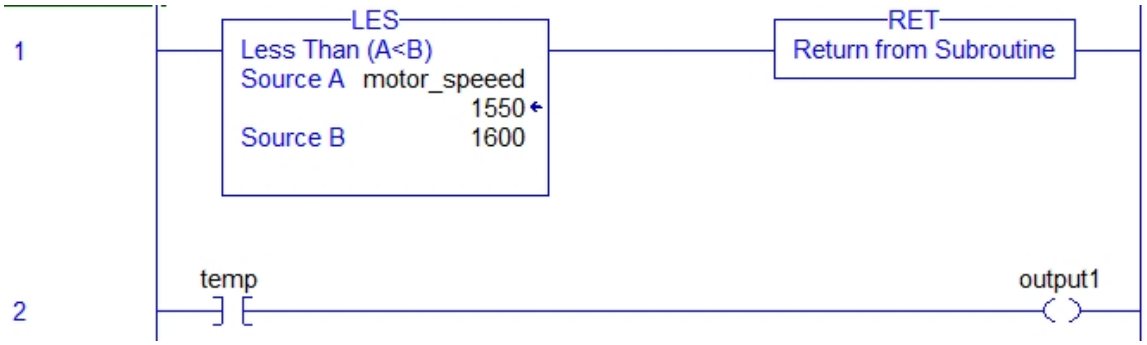


Figure 18-A
RET Instruction configured not to pass parameters, with conditions

.

In Figure 18-A, if the value of the tag motor_speed is Less Than 1600, then the processor exits the subroutine at that Rung.

In Figure 18-A, if the value of the tag motor_speed is greater than (or equal) to1600, the processor continues scanning the subroutine until it reaches a RET Rung that is True or until the processor reaches the END Rung.

JSR with Parameter Passing

JSR instruction can be configured to pass between routines.

If a JSR is configured to pass input parameters a SBR instruction must be used in the subroutine.

If a JSR is configured to receive return parameters from a subroutine a RET instruction must be used in the subroutine.

Figure 19-A shows a JSR instruction configured to pass two input parameters (Input Par) to Routine2 subroutine.

Figure 19-A shows a JSR instruction configured to receive one return parameter (Return Par) from Routine2 subroutine.

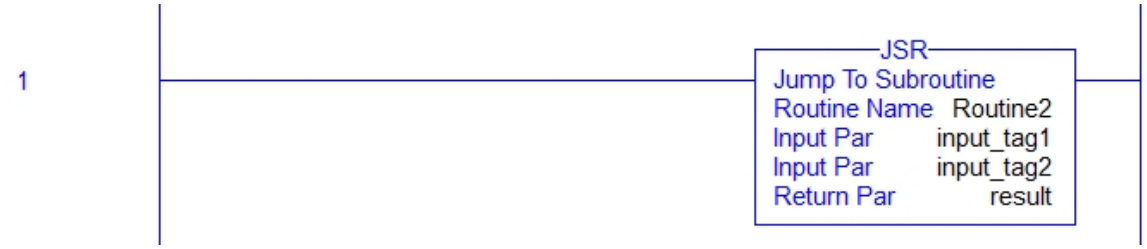


Figure 19-A. An JSR instruction with one return parameter.

Figure 20-A shows a SBR instruction configured to receive two input parameters (Input Par) from a JSR instruction.

If the SBR instruction in Figure 20-A were receiving parameters from the JSR instruction in Figure 19-A, the value of the input_tag1 in the JSR will be passed (moved / sent) to the Parameter1 tag in the SBR instruction.

The value of the input_tag2 in the JSR will be passed (moved / sent) to the Parameter2 tag in the SBR instruction.

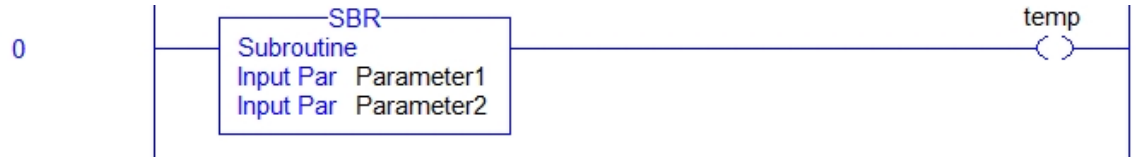


Figure 20-A. An SBR instruction with two input parameters.

Note: The number of Input Parameters (Input Par) in the JSR instruction must be equal to or greater than the number of Input Parameters (Input Par) in the SBR instruction.

The tag's Data Types should match i.e.
input_tag1's Data Type should be the same as Parameter 1's Data Type.
input_tag2's Data Type should be the same as Parameter 2's Data Type.

input_tag1's Data Type can be different than input_tag2's Data Type.

Figure 21-A shows a RET instruction configured to return one parameters (Return Par) to a JSR instruction.

If the RET instruction in Figure 21-A were returning parameters to the JSR instruction in Figure 19-A, the value of the Calc_value tag in the RET will be passed (moved / returned) to the result tag in the JSR instruction.

Figure 21-A. An RET instruction with one return parameter.

Note: The number of Return Parameters (Return Par) in the RET instruction must be equal to or greater than the number of Return Parameters (Return Par) in the JSR instruction.

The tag's Data Types should match i.e.
Calc_Value's Data Type should be the same as result's Data Type.

To add or remove an Input Par parameter from a JSR or SBR instruction, right mouse click inside the instruction box and select Add Input Parameter or Remove Instruction Parameter from the context menu.

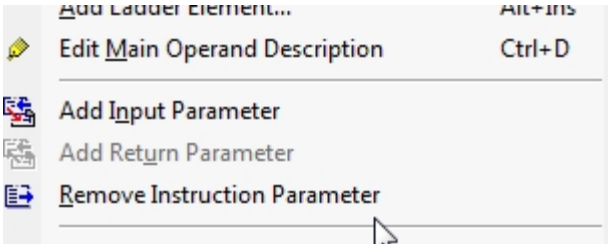


Figure 22-A. Adding an input parameter.

To add or remove a Return Par parameter from a JSR or RET instruction, right mouse click inside the instruction box and select Add Return Parameter or Remove Instruction Parameter from the context menu.

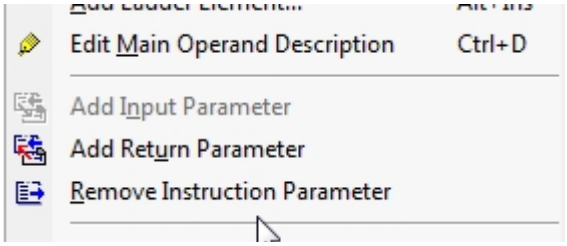


Figure 23-A. Removing an instruction parameter.

JSR instructions can be located in Main routines, Fault routines or in subroutine.
Do not configure a JSR to jump to a Main routine – Rung will not verify.

The following conditions will cause a Major Fault in the processor

1. RET instruction in a Main Routine
2. JSR instruction has fewer Input Par than a corresponding SBR
3. RET instruction has fewer Return Par than a corresponding JSR
4. JSR to a Fault routine – This will cause a User-defined Fault.

Review Questions

1. T **F** **A Program can contains only one Routine**

2. **Types of Routines are**

a) Subroutine

b) Main Routine

c) Fault Routines

d) All the above

3. **How many routines can be in a Program?**

a) 32

b) 100

c) 3

d) No set number

4. **Only one Routine per Program can be a _____**

a) subroutine

b) Fault

c) Main

d) Event

5. **The first routine run within a Program is:**

a) Fault

b) Continuous

c) Main

d) JSR.

6. T F **A project can have only one routine.**

7. A JSR instruction can pass Input parameters to a _____ instruction:

a) JSR

b) SBR

c) RET

d) All the above

8. A RET instruction can pass Return parameters to a _____ instruction

a) JSR

b) SBR

c) RET

d) All the above

9. All Programs should have a Main Routine:

a) True

b) False

10. All Programs must have a Fault Routine

a) True

b) False

11. T F **All subroutine must have a RET instruction**

Review Question Answers

1. T
2. d
3. d
4. b and c
5. c
6. T
7. b
8. a
9. a
10. b
11. F