HUMBER

ROBOTICS

Advanced Programming - Part II

Programming Flow & IF Conditions

Type of Commands / Statements

- Process statements (MOVE, Delay,)
- Input statements (wait For…)
- Output statement (set / Reset output)

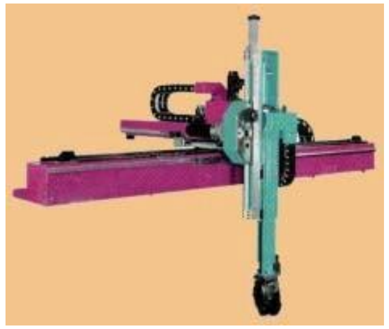
$\Big\}$ **Process and Action Statements**

- Conditional statement (IF - ELSE- ENDIF)
- Jump statements
- Sub Procedures
- Looping

$\Big\}$ **Flow Statement**

Program Flow Control

So far our robot programs have been structured that they start at a home or ready position and run through the commands and programmed points in the order they were programmed. *(Example point 4 is ran after point 3)*. In many robotic applications the program flow is changed within the program.



Program Flow: The sequence or order that the robot runs the commands in, but not necessarily the order that the commands are in the program.

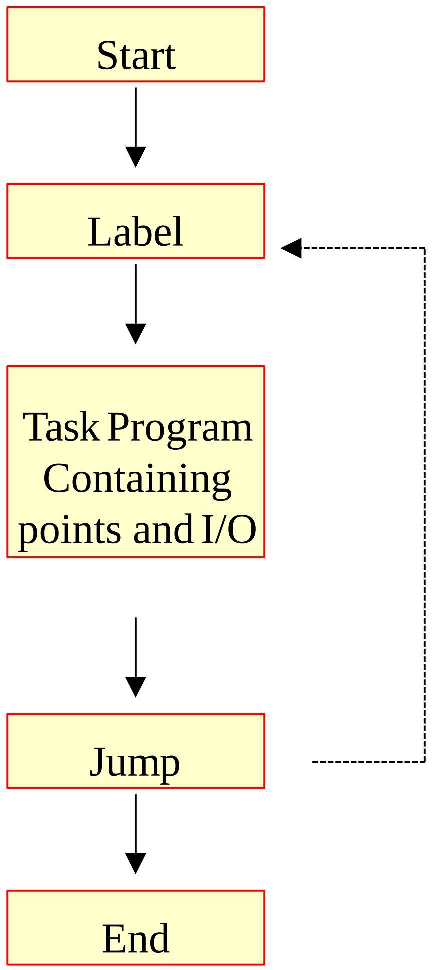
In this module we are going to introduce the concept of Program Flow Control. These concepts will be applied and practiced in Robotics 2. As with concepts discussed in previous modules most robots have the ability to perform flow control but the way that it is programmed and implemented will differ with robotic systems.

Advanced Programming

PROGRAM FLOW

- LABEL / JUMPS
- SUBPROGRAM OR SUBROUTINE
- NESTED SUBROUTINE
- IF Condition
- NESTED IF
- ELSE IF CLAUSE

Jump / Label



The most basic type of program flow control is the the Jump and Label commands. These commands are used as a pair or are complimentary. Normally identified with a number.

Jump (1) or Label (1)

The Jump / Label commands are used to control the order that a program will run in.

The robot runs its program one line at a time. When it reaches a *Jump* line it will perform a jump function. The scan or flow of the program will jump to the matching label command and continue running the program from that point.

A program can have multiple *jump / label* commands each identified with a different number.

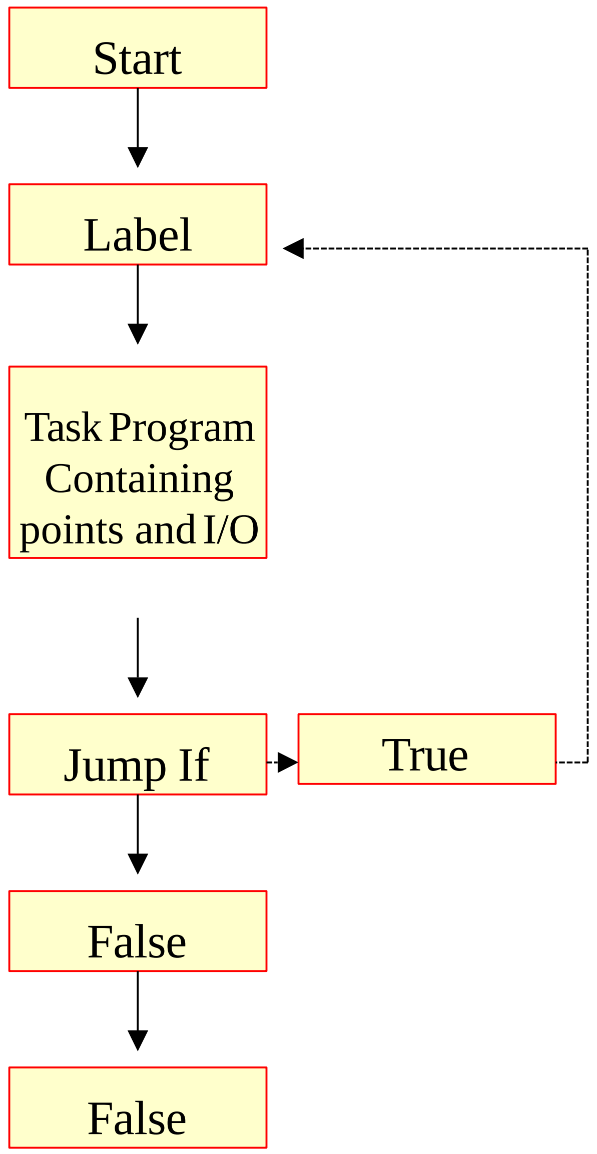
Jump (1) or Label (1)

Jump (2) or Label (2)

The label command is ignored if program flow is not commanded to it from a jump.

You can have multiple jumps to one label but not the other way around.

Conditional Jump



The Jump / Label command is very limited unless we have additional control. This additional control is gained be adding a condition on the jump command.

This means the jump will only be performed if the conditions are met. If the jump conditions are not true the jump command will be ignored.

If the programmer wanted the robot to pickup a part at one of two locations depending on what location had a part the inputs from the station could be used to determine the what pickup sequence to jump to within the program.

With all jump command care should be taken to not accidentally get into a situation of a endless loop. This is a situation where the robot starts a loop and never ends the loop.

Control Vs. Task Programs

Due to the costs involved with building and installing robotic equipment it is rare that a robot performs only one task. Typically robotic applications perform many different tasks. We need to setup a program flow structure where we can jump to the different tasks and still keep things manageable.

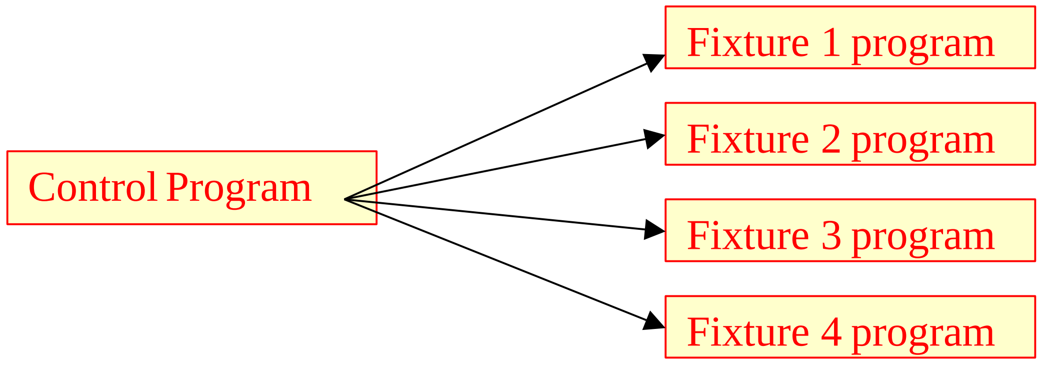
You can imagine that if we setup a series of conditional jumps it would make for a large and confusing program that is almost impossible to interoperate if you did not program it.

A better situation would be if we had a program for each task and one control program that determined what task program should run.

Example:

A robot works at 4 different fixtures and can move between fixtures in any order.

The correct task program to call is typically conditional based on input signals



Control Vs. Task Programs

This system of control and task program is so popular that CSA (Canadian Standards Association) has defined these terms within the Robotic Safety Standards.

Control Program:

(CSA) The inherent set of instructions that defines the capabilities, actions and responses of the robotic system. This program is not normally intended to be modified by the user.

This is a program that causes the program flow to jump or run the different task programs. This program normally runs continuously and uses conditions based on inputs to determine the task to run.

Task Program:

(CSA) set of motions and auxiliary functions that defines the specific intended task of the robotic system.

This is the program that does the work and is called by the control program if the conditions are correct.

Subroutine (subprograms):

A program ran or called from a program. When one program calls another the program being called is normally considered a subroutine. The subroutine is normally a task program

Call, Subroutines and Return

The best way to get into or out of the different task programs is with the Call and Return commands. The definitions are below along with an explanation for the command. The next page shows a flow charted view of how this concept works.

Call:

The call command has different names for it depending on the robotic system. The function of the command is very similar for all robots.

The call command is used to exit one program and start running a subroutine.

It can be conditional or non-conditional.

Return:

At the end of a subroutine the return command is used to return the program flow back to the place that the subroutine was called from.

An example of this is on the Fanuc Robot when you call the subroutine “opengrip” or “closegrip”. You call a small program to function the gripper and then return the original program at the end of the gripper subroutine.

If we had a control program calling your task program for lab 2A and called “opengrip” in the task program this is nested subroutines. See slide 9.

Subprogram / Subroutines

- Subprograms and subroutines are used to make program simple, readable, efficient and easy to modify
- Subprograms and subroutines have set of instructions that need to be used again and again in the main program
- Subprogram is a separate program which is called inside the main program
- Subroutines is the set of codes or program instructions which is written inside the main program under the name of a subroutine.
- Subprograms and Subroutines can be called many time in the program
- All subroutines must have return statment

Difference between Subprogram and Subroutine

Subprogram is an outside external program.

Subroutine is the part of the program, normally coded at the end of program.

All subroutines end at the Return statement.

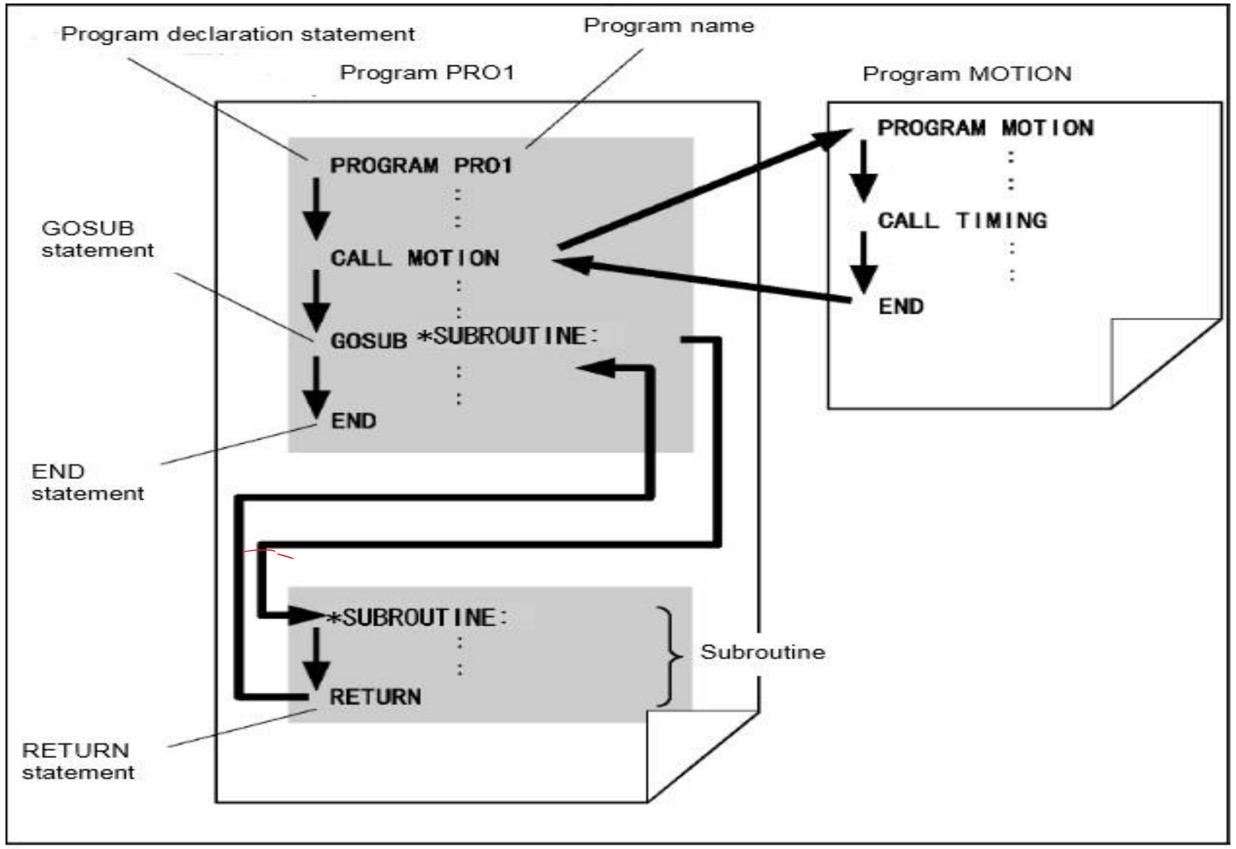
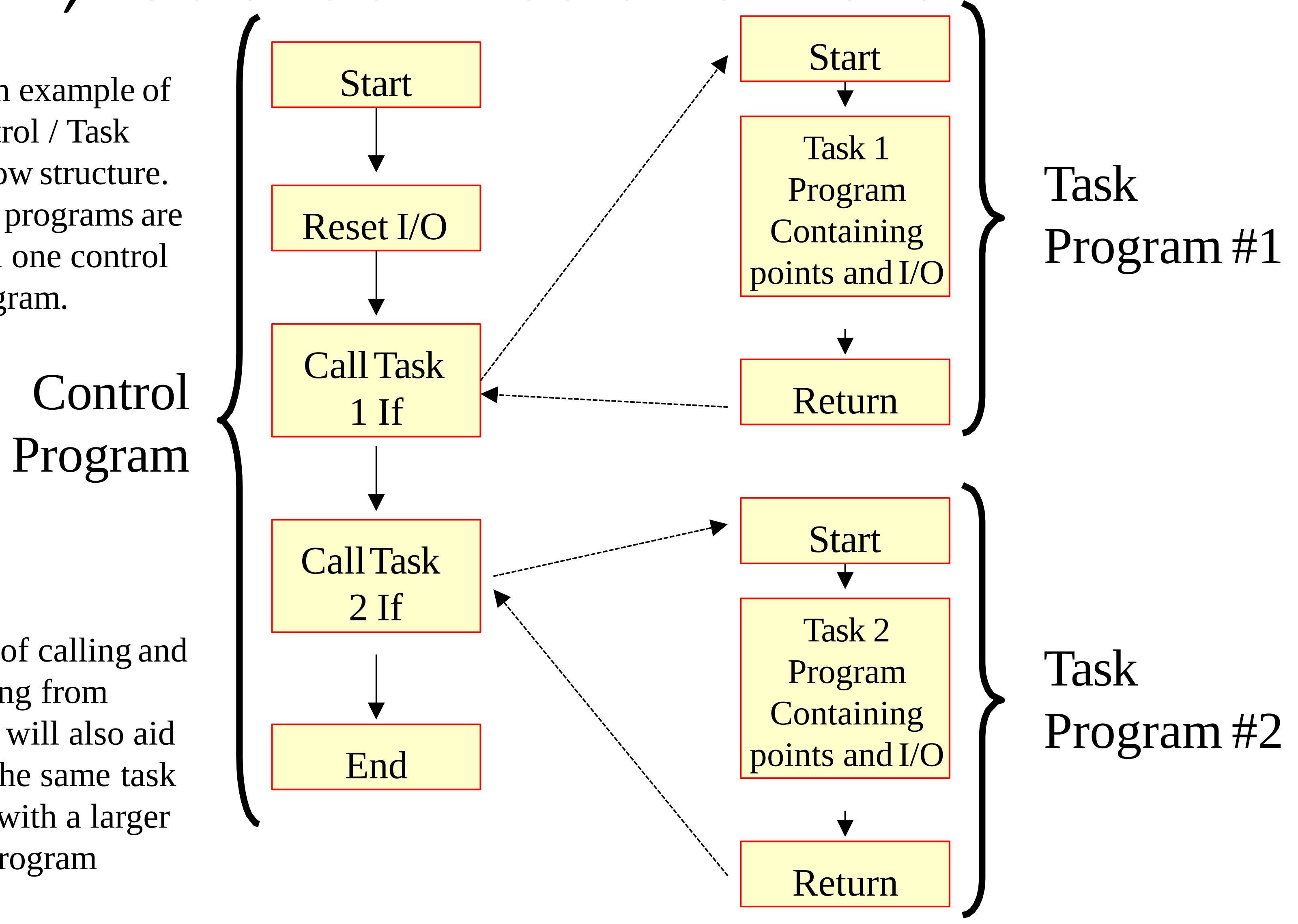


Fig. 2-1 Difference Between Calling a Program and Calling a Subroutine

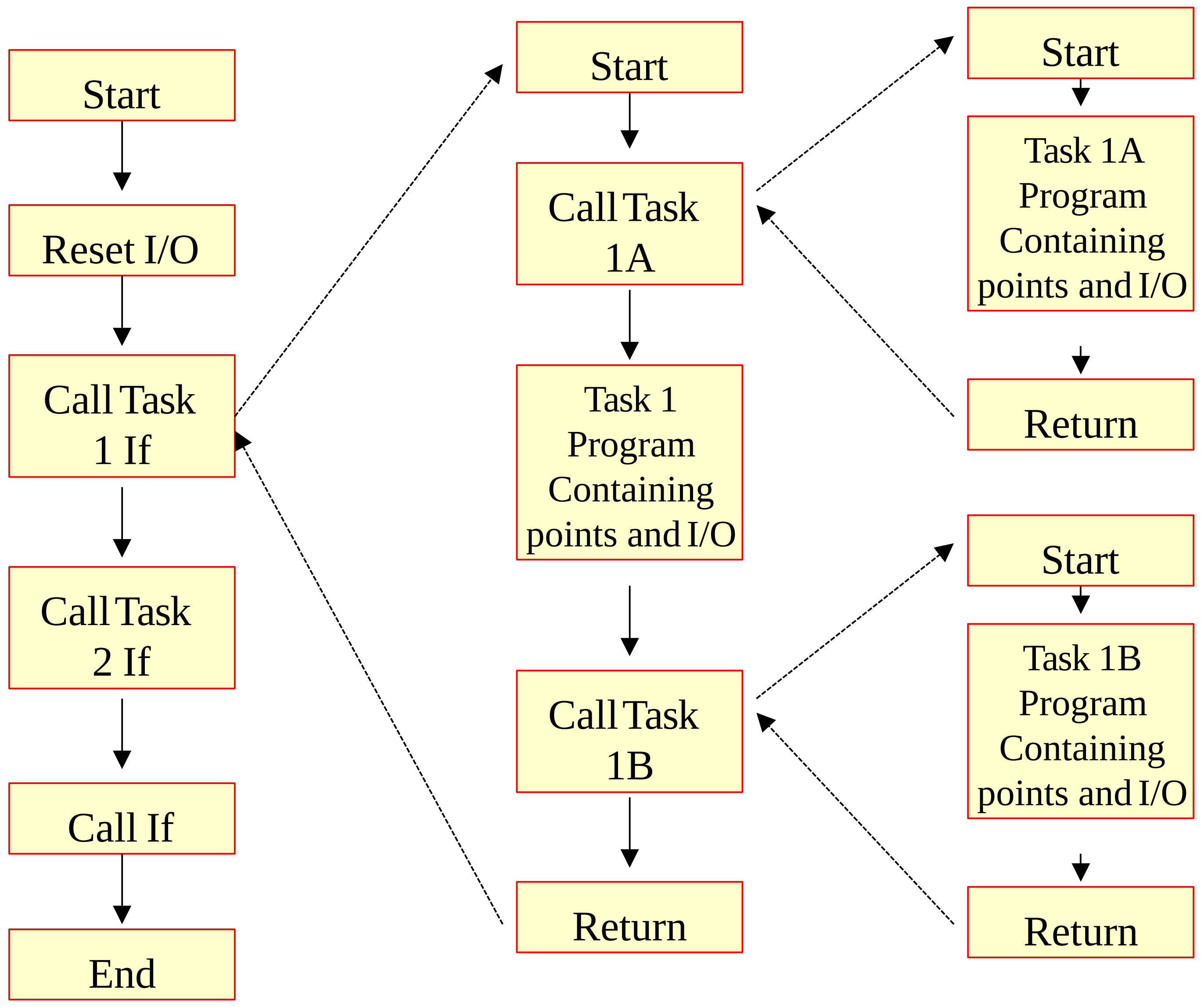
Call, Subroutines and Return

Shown is an example of the Control / Task program flow structure. Various task programs are called from one control program.



This system of calling and returning from subroutines will also aid in running the same task repeatedly with a larger task program

Nested Subroutines


Start
Reset I/O
Call Task 1 If
Call Task 2 If
Call If
End
Start
Call Task 1A
Task 1 Program Containing points and I/O
Call Task 1B
Return
Start
Task 1A Program Containing points and I/O
Return
Start
Task 1B Program Containing points and I/O
Return

Subroutines can be nested, or subroutines can call subroutines. This is a common practice.

There is a limit to the number of nested sub you can run. This maximum number of subs will change with different robotic systems.

You should never call out of a subroutine. Faults and errors will often occur if each sub does not have a return command.

Lab Robots with support of Sub-procedures and Subroutines

Robots	Sub Procedure Support	Sub Routines Support
KUKA	Yes	Yes
FANUC	Yes	No
ABB	No	Yes
WITTMANN	No	Yes
MOTOMAN	Yes	No
DENSO	Yes	Yes

IF - Conditional Statement

IF condition is used to control the flow of the program. IF condition has 04 major parts

1. IF
2. Condition
3. Action
4. End IF

Action will be processed if the condition is true, otherwise nothing will be executed.

Example : Single IF Block – General IF Block Instruction

```
IF <Condition> then
Action
End IF
```

Example:

```
IF Switch A =ON then
CALL Proc_1
End IF
```

IF Conditional Statement

Example : Double IF Block (Two Conditions & Two Actions)

```
IF
Condition1 then      First IF
Action1              Block
End IF
IF
Condition2 then      Second
Action1              IF Block
End IF
```

Example:

```
IF Switch A =ON then
CALL Proc_1
End IF
IF Switch B =ON then
CALL Proc_2
End IF
```

IF ELSE - Conditional Statement

IF ELSE condition is used to control the flow of the program with alternative option. IF ELSE condition has 06 major parts

1. IF
2. Condition
3. Action1
4. Else
5. Action 2
6. End IF

Example : Single IF Block – General IF ELSE Block Instruction

```
IF <Condition> then
Action1
Else
Action2
End IF
```

Action1 will be executed if the condition is true, Action2 will be executed if the condition is false.

Example:

```
IF Switch A =ON then
CALL Proc_1
Else
CALL Proc_2
End IF
```

IF Block Joining Two Conditions For One Action

IF Block can be used to check more than one condition by adding “and” statement.

General Statement

```
IF Condition 01 and condition 02 then
Action
End IF
```

Example

```
IF switch A = ON and switch B= OFF then
Call Task A
End IF
```

Note: Many Robots allow to join 02 or more conditions in IF statement, but some robots do not allow to join two conditions for example Panasonic

IF Block with Two Conditions Other Options

IF Block can be used for more than one condition by using Nested IF or Jump and label command “and” command.

Example (Joining two conditions with Jump & Labels)

```
IF Condition 01 then
Jump Label 1
End IF
Label 1 : IF Condition 02 then
Call Task A
End IF
```

Example (Joining two conditions with and)

```
IF condition 1 and condition 2 then
Call Task A
End IF
```

Nested IF (Joining two conditions with Nested IF)

```
IF Condition 01 then
  IF condition 02 then
                 call Task A
 End IF
End IF
```

IF ELSE IF Clause with many Independent Conditions & Actions

IF ELSE IF clause can be used to check as many as possible independent conditions and execute the actions accordingly.

General Example

```
IF condition 1 then
Action 1
Else IF condition 2
Action 2Else IF
condition 3
Action 3
Else IF condition 4
Action 04
Else
Action 05
End IF
```