GSG001 Rev. B

Getting Started Guide

MATH AND TRIG FUNCTIONS ON CP1E

Getting Started Guide

"Getting Started Guide" is a collection of information that helps to speed up learning and easily start working with Omron products. It is not to be intended as a substitution of manuals.

INDICE

1 INTRODUCTION

1.1 SCOPE OF THIS GUIDE

The present document is intended to help with the use of the trigonometric and math subroutine library for CP1E, that in fact extends its set of functions.

1.2 TECHNICAL NOTES

The CP1E PLC instruction set doesn't include trigonometric functions and some math functions such as the square root or the power raise.

The *CP1E_Math.cxp* file contains nine sections whose the first (Sezione1) is empty and must be used for the main program.
Each of the remaining eight sections contains the definition of one of these math or trigonometric functions:

- SIN: sine function;
- COS: cosine function;
- TAN: tangent function;
- ASIN: arcsine function;
- ATAN: arctangent function;
- SQRT: square root function;
- RAD: degree to radians conversion;
- REARG: conversion from a generic angle to a -2π ÷ +2π range angle.

2 SUBROUTINES

2.1 SIN

Calculates the sine of a radians-expressed angle, stored in the *ARG* global variable, and outputs the result value in the *RES* global variable.

Subroutine definition number: 2.

Operands

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-205887.4 ; +205881.1	Input angle (in radians)
RES	D2023	REAL	-1 ; +1	Result

Accuracy

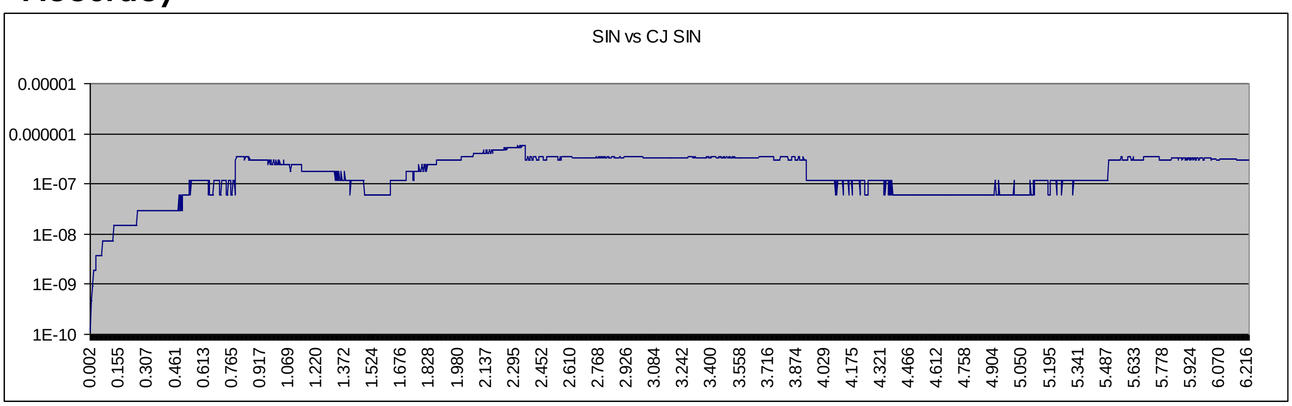

Figure 1: Measured error between SIN subroutine and standard SIN instruction in the 0 ÷ 2π range

Requested subroutines

- REARG.

2.2 COS

Calculates the cosine of a radians-expressed angle, stored in the *ARG* global variable, and outputs the result value in the *RES* global variable.

Subroutine definition number: 3.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-205887.4 ; +205881.1	Input angle (in radians)
RES	D2023	REAL	-1 ; +1	Result

Accuracy:

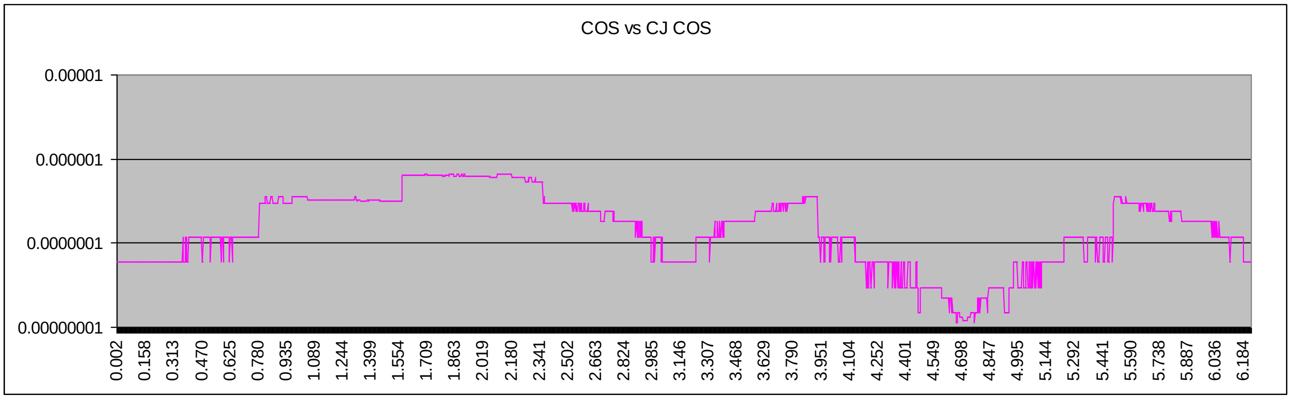

Figure 2: Measured error between COS subroutine and standard COS instruction in the 0 ÷ 2π range

Requested subroutines

- REARG.

2.3 TAN

Calculates the tangent of a radians-expressed angle, stored in the *ARG* global variable, and outputs the result value in the *RES* global variable.

Subroutine definition number: 4.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-205887.4 ; +205881.1	Input angle (in rad.)
RES	D2023	REAL	-3.402823e38 ; +3.402823e38	Result

Accuracy:

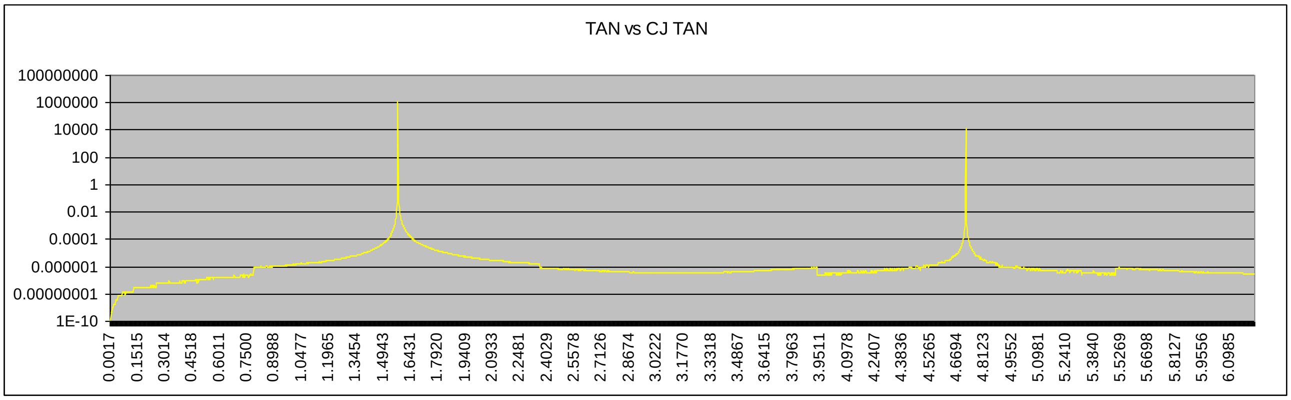

Figure 3: Measured error between TAN subroutine and standard TAN instruction in the 0 ÷ 2π range

Requested subroutines

- REARG;
- SIN;
- COS.

2.4 ASIN

Calculates the arcsine of a REAL value, stored in the *ARG* global variable, and outputs the result (in radians) in the *RES* global variable.

Subroutine definition number: 6.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-1 ; +1	Argument
RES	D2023	REAL	-3.141593 ; +3.141593	Result angle(in radians)

Accuracy:

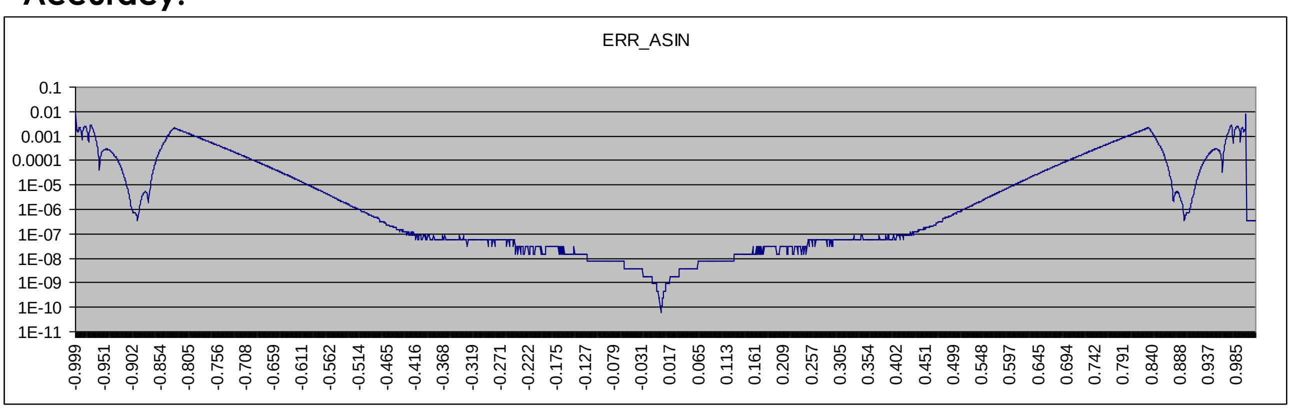

Figure 4: Measured error between ASIN subroutine and standard ASIN instruction in the -1 ÷ +1 range

Requested subroutines
None.

2.5 ATAN

Calculates the arctangent of a REAL value, stored in the *ARG* global variable, and outputs the result (in radians) in the *RES* global variable.

Subroutine definition number: 5.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-3.402823e38 ; +3.402823e38	Argument
RES	D2023	REAL	-3.141593 ; +3.141593	Result angle(in rad.)

Accuracy:

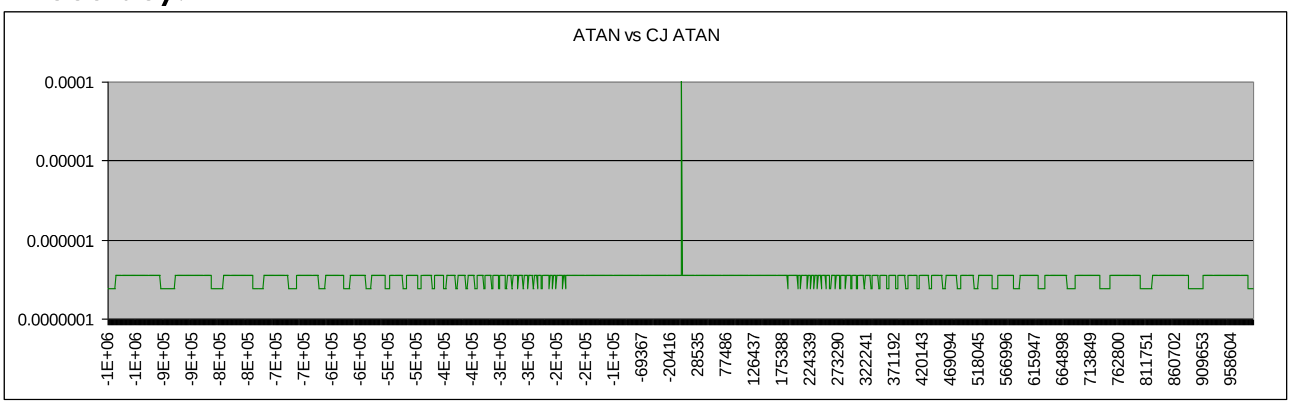

Figure 5: Measured error between ATAN subroutine and standard ATAN instruction in the $-10^6 \div +10^6$ range

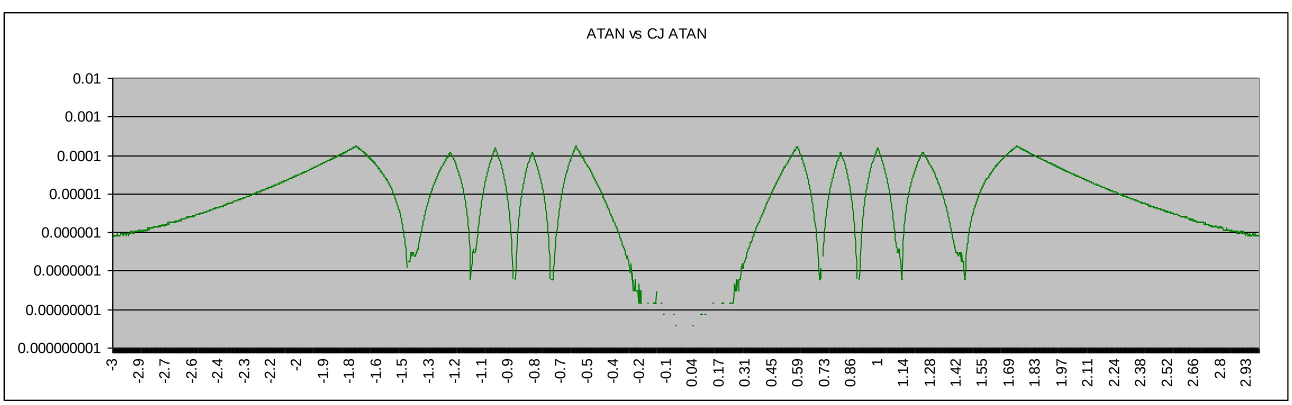

Figure 6: Measured error between ATAN subroutine and standard ATAN instruction in the $-3 \div +3$ range

Requested subroutines
None.

2.6 RAD

Converts a degree-expressed angle stored in the *ARG* global variable to a radians-expressed angle in the *RES* global variable.

Subroutine definition number: 1.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-3.402823e38 ; +3.402823e38	Degrees
RES	D2023	REAL	-5.93904e36 ; +5.93904e36	Radians

Accuracy:

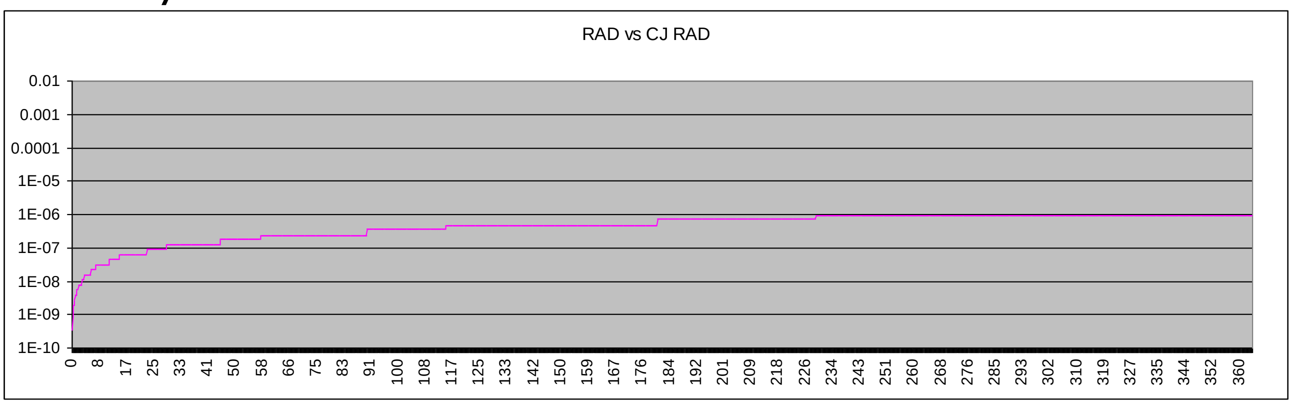

Figure 7: Measured error between RAD subroutine and standard RAD instruction in the 0° ÷ 360° range

Requested subroutines
None.

2.7 REARG

Calculates the -2π to +2π equivalent of a generic radians-expressed angle; furthermore it overwrites the output on the same *ARG* input global variable.

<u>Note</u>: REARG subroutine is for internal use in SIN, COS and TAN subroutines; no equivalent instruction in the standard instruction set.

Subroutine definition number: 8.

Operands:

Name	Address	Type	Valid range	Descrption
ARG	D2021	REAL	-205887.4 ; +205881.1	Input angle (in rad.)

Requested subroutines

None.

2.8 SQRT

Calculates the square root of a real value stored in the *ARG* global variable and outputs the resut in the *RES* global variable.

Subroutine definition number: 7.

Operands:

Name	Address	Type	Recommended range	Descrption
ARG	D2021	REAL	0 ; +13000	Input
RES	D2023	REAL	0 ; +114.0175	Result

Accuracy:

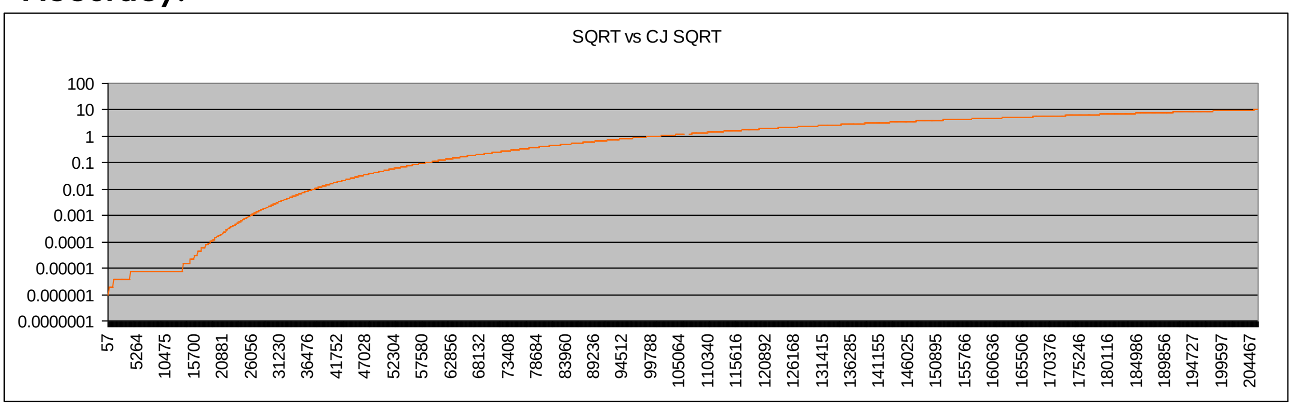

Figure 8: Measured error between SQRT subroutine and standard SQRT instruction in the 0 ÷ 2*10⁵ range

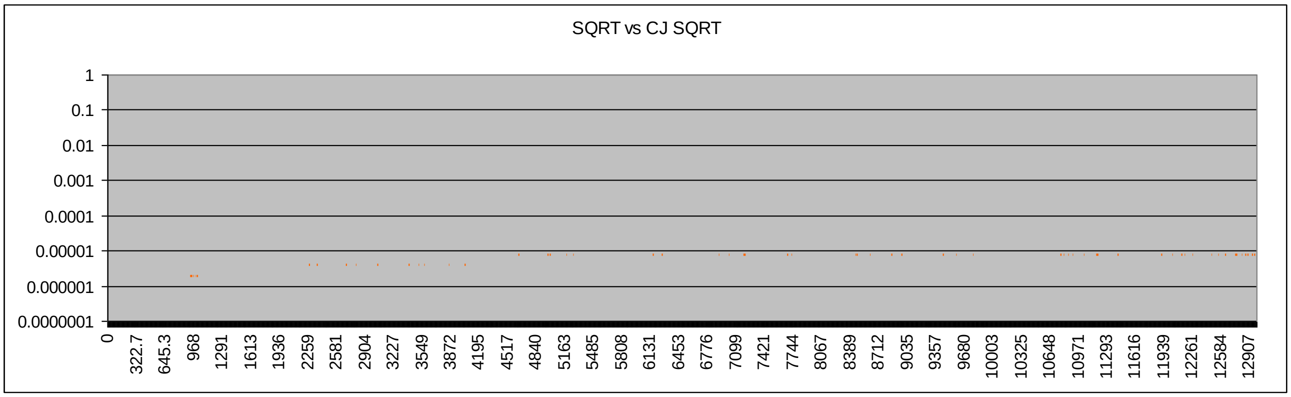

Figure 9: Measured error between SQRT subroutine and standard SQRT instruction in the 0 ÷ 13000 range (recommended range)

Requested subroutines

None.

NOTE:

Calculations with arguments out of the valid range of use will cause invalid results.

3 PROGRAMMING PROCEDURE

3.1 STEP BY STEP GUIDE

The sequent steps explain the way to insert the math/trig subroutines in a ladder program.

3.1.1 CREATION OF A NEW PROGRAM WITH MATH LIBRARY

Launch the *CP1E_Math.cxp* file, eventually renamed, and open the *Sezione1* section in the *Programma1* main program. If not renamed save it with another name.

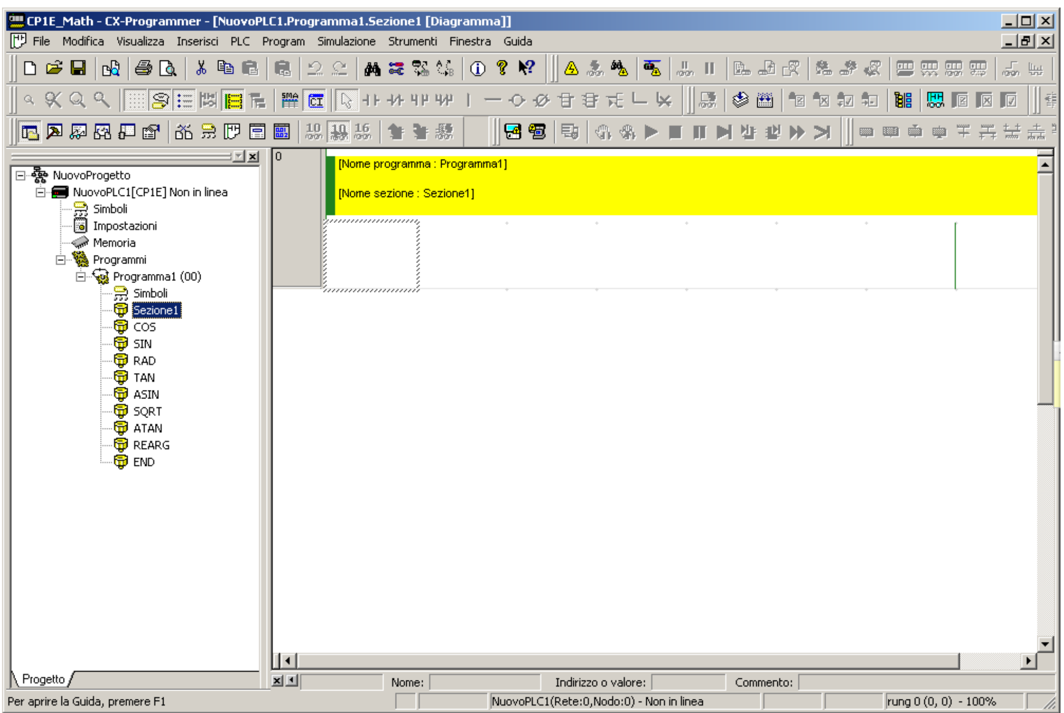

Figure 10: CP1E_Math program view in CX-Programmer

3.1.2 CALLING A SUBROUTINE IN THE MAIN PROGRAM

Before calling the subroutines in the program, you should define an input variable (the argument of the function) and an output variable to put the result.

In order to call one of the subroutines follow these steps:

1. Copy the input variable content to the *ARG* variable.*

2. Execute the SBS instruction with the name (or the definition number) of the desired subroutine to be called;
3. Copy the *RES* variable content to the defined output variable.*

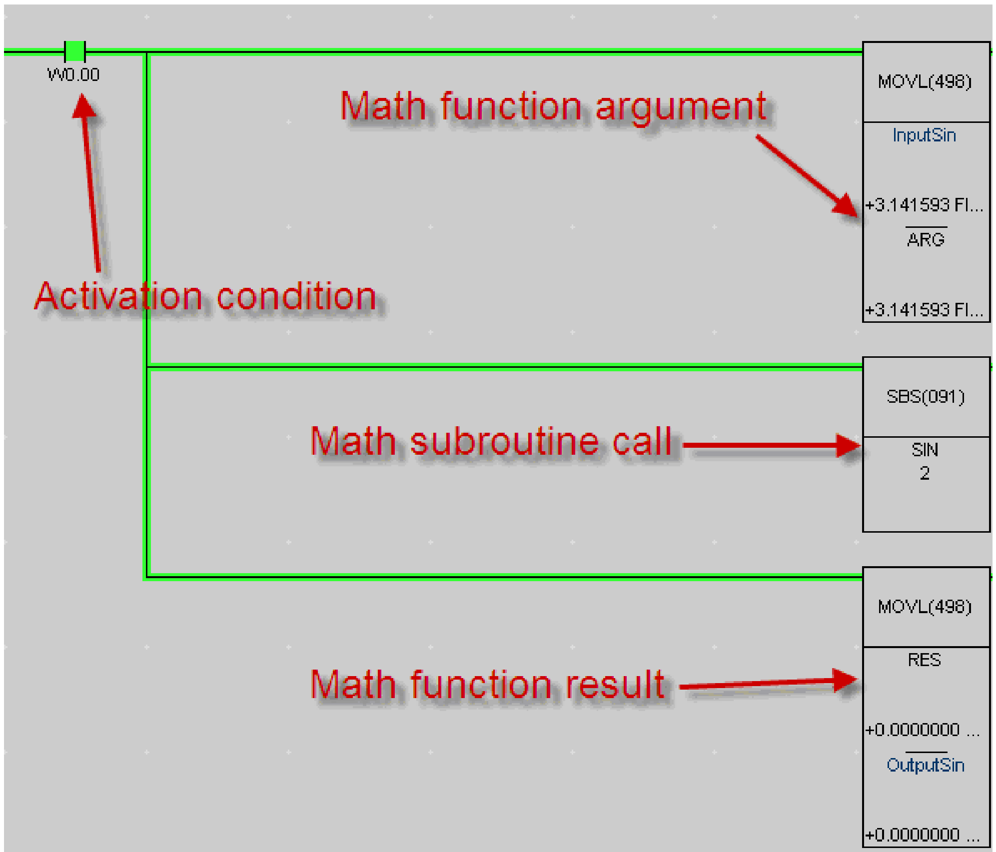

Figure 11: Calling a subroutine in the program

NOTE:
(*) Use MOVL instruction to copy values, because *ARG* and *RES* variables are REAL (32-bit).

3.1.3 PROGRAM ORGANISATION

Start programming the empty section *Sezione1*; do not modify the subroutines sections.
Other sections can be added, but they must be put before the first subroutine section.

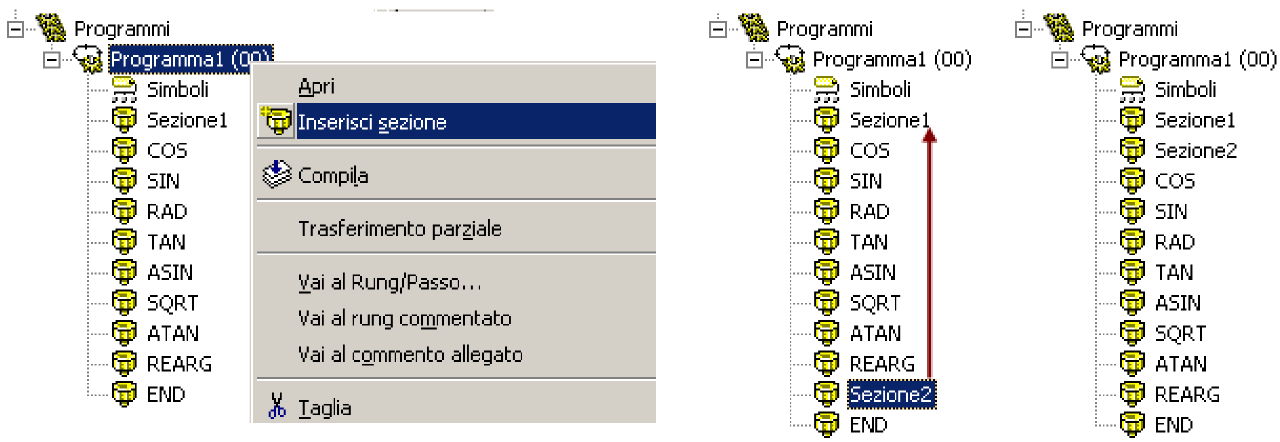

Figure 12: Adding a new section

3.2 RESERVED MEMORY AREAS

The memory channels reserved for the subroutines are:

- D2021 – D2022 : *ARG* data exchange variable from the main program to the subroutine;
- D2023 – D2024 : *RES* data exchange variable from the subroutine to the main program;
- D2025 – D2047 : Subroutines internal data;
- W90 : Subroutines internal data.

3.3 FREEING UP PROGRAM MEMORY

An empty program with all the subroutines described above occupy 1495 steps.

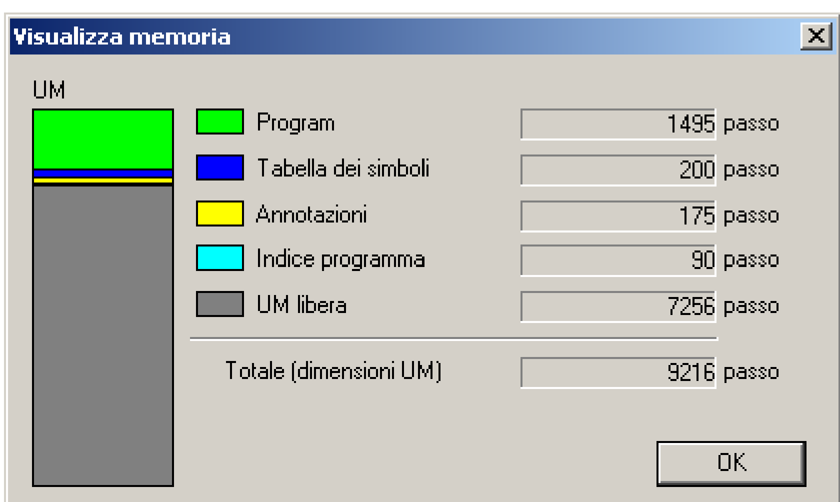

Figure 13: Program memory occupation for CP1E-NA

If some subroutines are unused, it is possible to free up memory by deleting the correspondent section.
Some of the subroutines, however, are called inside others, so before deleting a subroutine it is necessary to check the "*Requested subroutines*" field in the subroutine reference in paragraph 2, or the following table:

Subroutine name	Subroutines where it is called
SIN	TAN
COS	TAN
TAN	
ASIN	
ATAN	
RAD	
REARG	SIN, COS, TAN
SQRT	

Example: If in a program are used only TAN and COS subroutines, then the following can be removed: ASIN, ATAN, RAD, SQRT.

3.4 NOTE ABOUT THE EXECUTION TIME

The execution time of mathematical subroutines is generally greater than that of the same standard instructions, whereby a wide use of these subroutines can weigh on the program in terms of cycle time slowdown. It is advisable to check that the final program cycle time respects design specifications.

3.5 EXAMPLE

Calculate the cosine of *alpha_deg* degree-expressed angle:

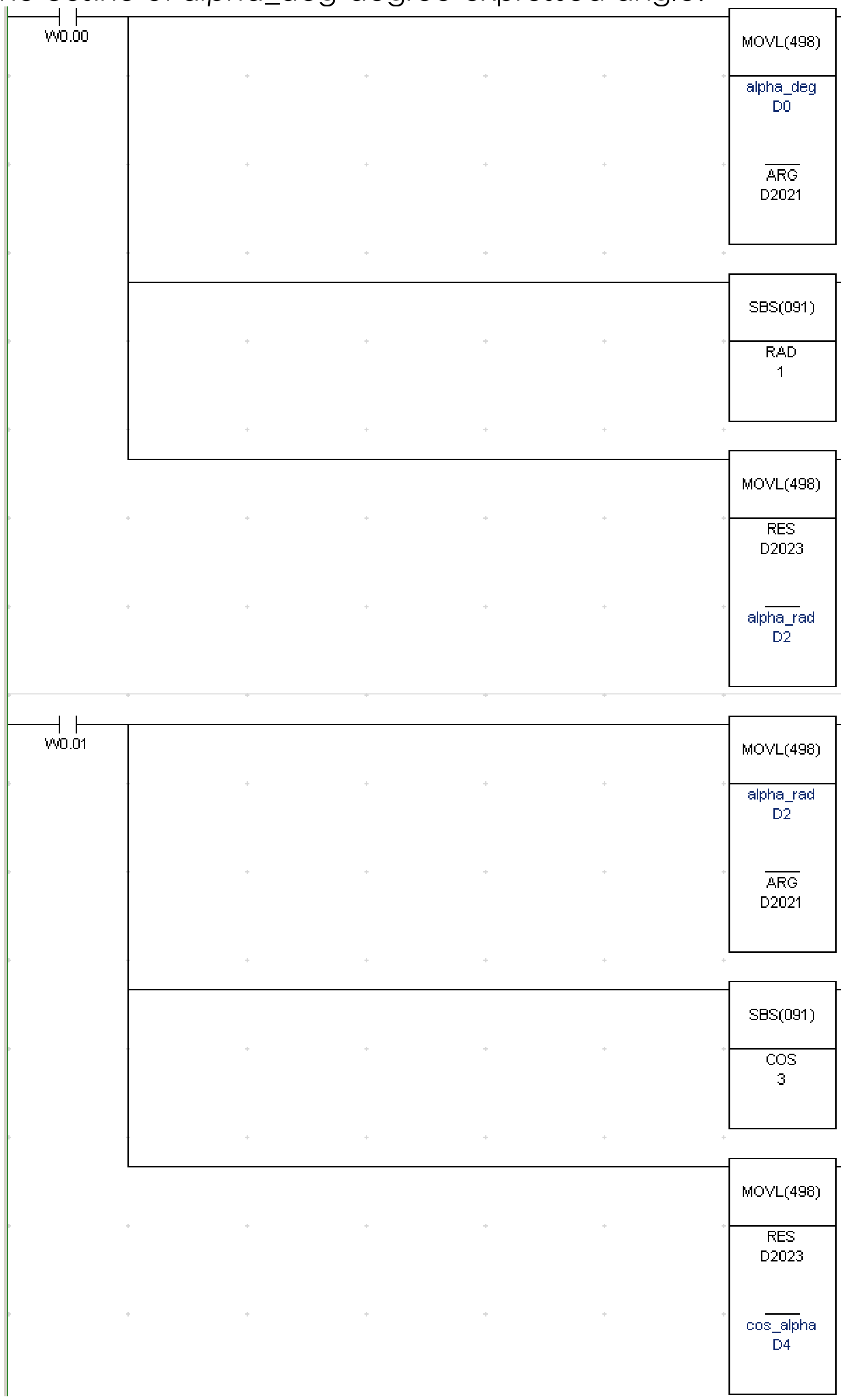

Figure 14: Programming example

In the first rung *alpha_deg* is converted from a degree-expressed to a radians-expressed angle through the *RAD* subroutine, then the result is pasted to an *alpha_rad* variable, while in the second rung the cosine of *alpha_rad* is calculated through the *COS* subroutine, then the result is pasted to the *cos_alpha* variable.

4 APPENDIX

You can find more documents related to this tool and interesting information visiting the web site www.myomron.com.