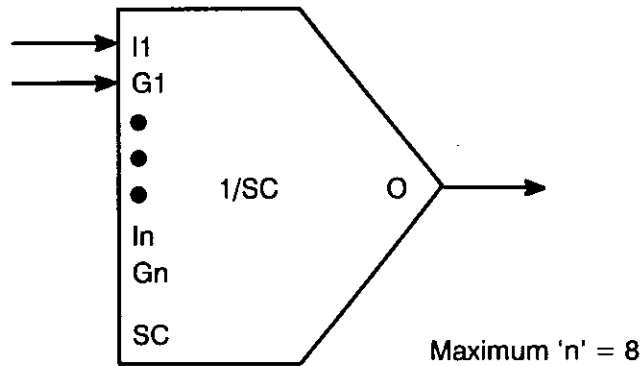


7.0 AMPLIFIER

This function can be used in AutoMax Control Block tasks and UDC Control Block tasks.



Function

$$\text{OUTPUT} = (\text{INPUT1} * \text{GAIN1} + \dots + \text{INPUTn} * \text{GAINn}) / \text{SCALE}$$

Program Statement

```
CALL AMPLIFIER(INPUT1=input1%,           &
                GAIN1 = gain1%,...,      &
                INPUTn = inputn%,        &
                GAINn = gainn%,          &
                SCALE = scale%,          &
                OUTPUT = output%)        &
```

Inputs

- I1 (INPUT) =
INTEGER signal type 1. This parameter must be specified.
- G1 (GAIN) =
INTEGER gain 1. This parameter must be specified.
- In (INPUTn) =
INTEGER signal input n. This is an optional parameter. A maximum of 8 input/gain pairs can be specified.
- Gn (GAINn) =
INTEGER gain n. This is an optional parameter; however, a GAIN must be specified for each INPUT.
- SC (SCALE) =
INTEGER scale factor. The default for this parameter is 10000.

Outputs

- O (OUTPUT) =
INTEGER signal output. This parameter must be specified.

Notes

1. The order in which (INPUTn) and (GAINn) pairs are entered is not important. However, it is required that, for "m" channels used, channels 1 through "m" be programmed. In other words, the channels used must be contiguous beginning with channel 1. A compilation error will occur if this requirement is not met.

The following is a correct statement:

```
CALL AMPLIFIER(                                &
  INPUT1 = INA%,GAIN1 = GAINA%,                &
  INPUT2 = INB%,GAIN2 = GAINB%,                &
  INPUT3 = INC%,GAIN3 = GAINC%,                &
  INPUT4 = IND%,GAIN4 = GAIND%,                &
  SCALE = scale%,OUTPUT = output%)
```

The following is also correct, illustrating that the order of entry is not important:

```
CALL AMPLIFIER(                                &
  INPUT2 = INB%,GAIN2 = GAINB%,                &
  INPUT4 = IND%,GAIN4 = GAIND%,                &
  INPUT3 = INC%,GAIN3 = GAINC%,                &
  INPUT1 = INA%,GAIN1 = GAINA%,                &
  SCALE = scale%,OUTPUT = output%)
```

The following is an incorrect statement because the channels are not contiguous:

```
CALL AMPLIFIER(                                &
  INPUT1 = INA%,GAIN1 = GAINA%,                &
  INPUT3 = INC%,GAIN3 = GAINC%,                &
  INPUT4 = IND%,GAIN4 = GAIND%,                &
  SCALE = scale%,OUTPUT = output%)
```

2. Overflow handling: During block execution, (INPUTn) and (GAINn) pairs are read and processed in sequential order beginning with (INPUT1) and (GAIN1) until a null pair is found. As the sum of the products from (INPUTn*GAINn) is computed and an overflow of 32 bits occurs (sum exceeds +2147483647 or -2147483648), the sum will be clamped to +2147483647 or -2147483648, an error will be logged, and the OUTPUT will be clamped to +32767 or -32768. Once a 32-bit overflow occurs, all remaining input channels will be ignored.

When all consecutive sequential input channels are processed, the 32-bit sum is divided by SCALE, producing OUTPUT as a 16-bit result. If an overflow occurs on the divide, that is, the result exceeds 16 bits +32767 or -32768, an error will be logged and the result will be clamped to +32767 or -32768, producing OUTPUT.

3. Multiplying a positive input by -1 will result in that input being subtracted from the other inputs.
4. Setting SCALE <0 will effectively invert the sign of the output.