Issued by: Mitsubishi Electric Europe B.V. - UK Branch
Document issue: V1.00 June 2006
Produced by : - CTC
Travellers Lane, Hatfield, Herts, AL10 8XB, UK
Author: Matt Wills

Databank - **Technical Bulletin**

This data sheet covers using the MR-HENC Absolute encoder to generate a 360^0 Ring counter for use with Rotary indexing table type applications.

<u>MR-HENC Absolute Serial Encoder</u>

The MR-HENC Absolute Serial Encoder has a resolution of 16384 Pulses Per Revolution. (ppr). It's Maximum permissible speed is 4300 Revolutions Per Minute. (rpm)

This Encoder is designed to be used with the Serial Encoder input module detailed below:

86729	MR-HENC	Synchronous encoder unit
140584	Q172EX	Serial ABS encoder interface (MR-HENC) x 2 - Tracking inputs x 2

The Value of the external encoder can be monitored and used both in the Mechanical System editor and also the SFC programming environment, however the Virtual Mode Operating System (SV22) must be used.

More than one external encoder can be utilised, a maximum of 8 for the Q172(H)CPU and 12 for the Q173(H)CPU.

The Monitor Data for each encoder is shown:

- Detailes of each axis

Device No.	Signal name
D1120 + 10n D1121 + 10n	Current value
D1122 + 10n	Minor error code
D1123 + 10n	Major error code
D1124 + 10n D1125 + 10n	Unusable
D1126 + 10n D1127 + 10n	Current value after synchronous encoder axis main shaft's differential gear
D1128 + 10n	Error search output axis No.
D1129 + 10n	Unusable

(Note-1) : "n" in the above device No. shows the numerical value which correspond to axis No.

The Device we need to consider for the 360^0 Ring counter is D1120 (For the first encoder).

D1120 will store the current value of the Encoder as detailed overleaf:

Issued by: Mitsubishi Electric Europe B.V. - UK Branch
Document issue: V1.00 June 2006
Produced by : - CTC
Travellers Lane, Hatfield, Herts, AL10 8XB, UK
Author: Matt Wills

Databank - **Technical Bulletin**

4.2.5 Synchronous encoder axis monitor devices

(1) Current value storage register (D1120+10n, D1121+10n) .... Monitor device

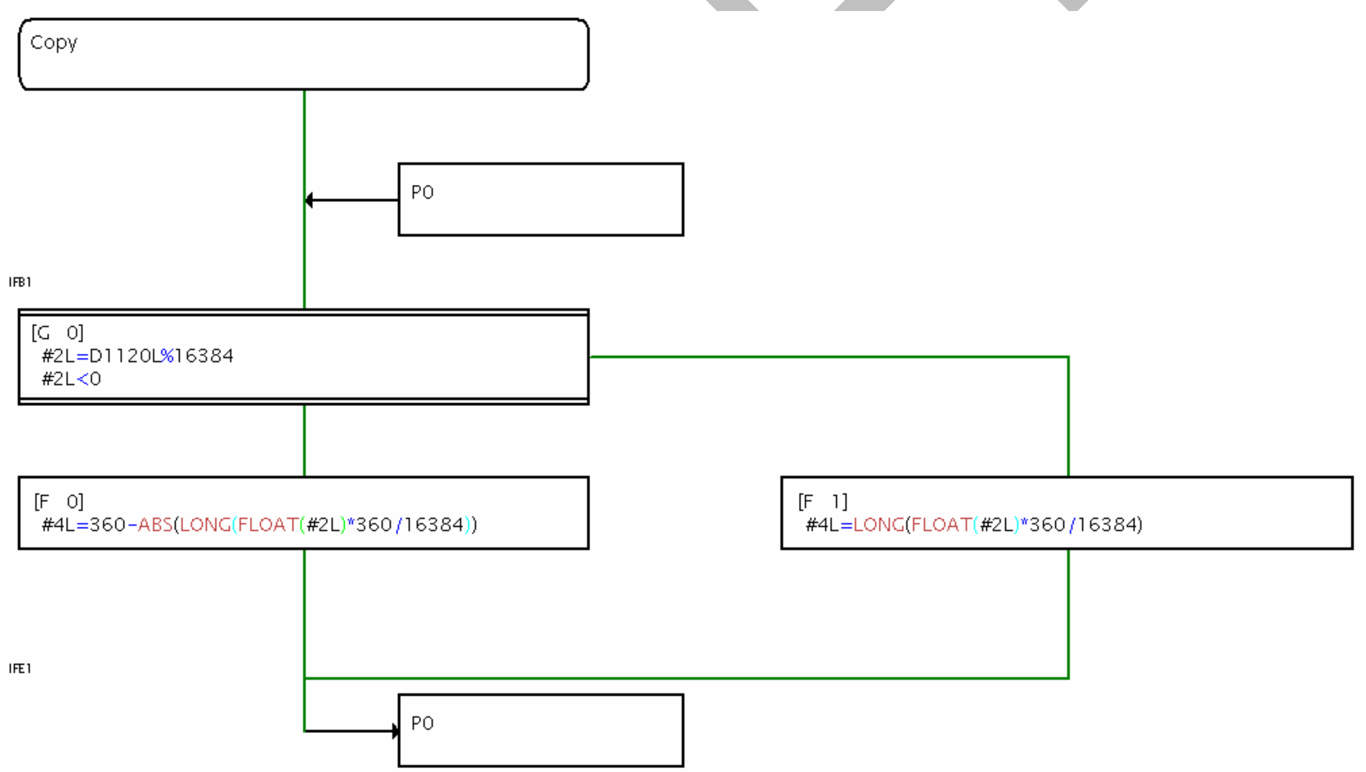
(a) This register stores the synchronous encoder current value of the drive module.

(b) Ring address is "-2147483648 (-2^{31}) to 2147483647 ($2^{31}-1$)" [PLS].

(c) The current value storage register data is also stored in a backup memory at the power supply off or resetting of the Multiple CPU system.

As can be seen, D1120 has a Ring value of -2147483648 to 2147483647 which will not give us a 0 to 360^0 value suitable for a rotary indexing table type application.

In order to overcome this we need to make some calculations in the Motion CPU using the SFC Programs:



If we break down the simple example program above into the 3 individual steps:

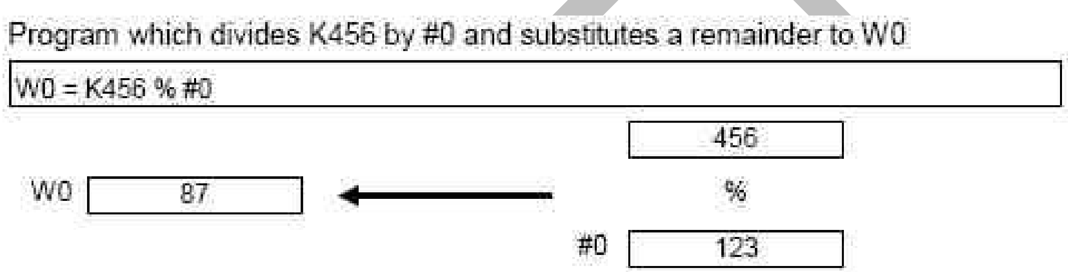
Issued by: Mitsubishi Electric Europe B.V. - UK Branch
Document issue: V1.00 June 2006
Produced by : - CTC
Travellers Lane, Hatfield, Herts, AL10 8XB, UK
Author: Matt Wills

Step 1:

```
[G 0]
 #2L=D1120L%16384
 #2L<0
```

Here the important instruction is the Remainder Instruction, %. This instruction divides the contents of D1120 (The Encoder Current Value) by the resolution of the Encoder, in this case 16384, and stores the **Remainder** in the Number Register 2 as a 32 bit word. (The L signifies Long, or 32 bit).

Another example of the remainder instruction taken from the SFC Programming manual:



The next process is a Y/N Transition. If the Contents of #2L are less then zero the Transition will shift downwards in the normal way, however if the contents of #2L are greater than zero the Transition will shift to the right hand side connection step:

Name	Symbol	Function
Shift Y/N transition	(Not completion of condition) Gn N (Completion of condition) Y	• When a transition condition set at Gn enables, execution shifts to the lower step. When that condition disables, execution shifts to the right-connected step. • Differences between "Shift Y/N" and "WAIT Y/N" are the same as those between "Shift" and "WAIT".
WAIT Y/N transition	(Not completion of condition) Gn N (Completion of condition) Y	

So, let us assume the value of #2L was greater than zero first, therefore the right hand side connected step will be executed:

Step 2:

```
[F  1]
 #4L=LONG(FLOAT(#2L)*360/16384)
```

Here the following instructions are used:

LONG()
FLOAT()
* (Multiplication)
/ (Division)

Let us start by explaining each instruction:

LONG(S)

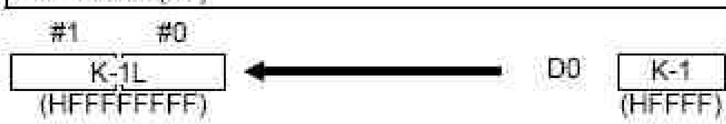
The Data specified with (S) is converted into a signed 32-bit Integer value. The Data range of (S) is -2147483648 to 2147483647.

When (S) is a 64-bit floating-point type, its fractional portion is rounded down before conversion is made.

Example taken from SFC Programming manual:

Program which converts the data of D0 into a signed 32-bit integer value and substitutes the result to #0L

```
#0L = LONG(D0)
```



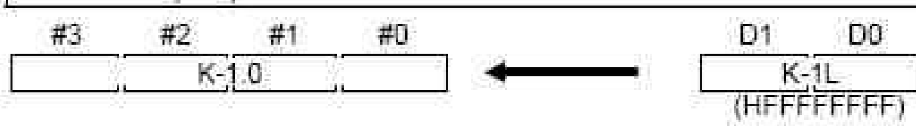
FLOAT(S)

The data specified with (S) is converted into a signed 64-bit floating-point value.

Example taken from SFC Programming manual:

Program which converts the data of D0L into a signed 64-bit floating-point value and substitutes the result to #0F

```
#0F = FLOAT(D0L)
```



(S1)*(S2) (Multiplication)

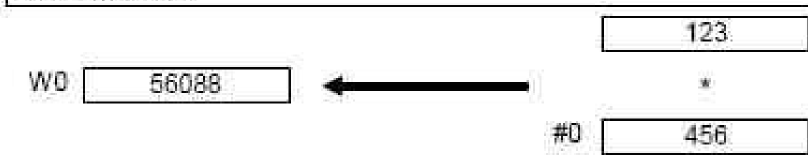
The data specified with (S1) is multiplied by the data specified with (S2).

When (S1) and (S2) differ in data type, the data of the smaller data type is converted into that of the greater type before operation is performed.

Examples taken from SFC Programming manual:

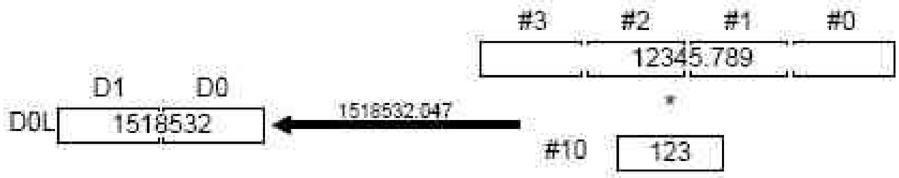
(1) Program which substitutes the result of multiplying K123 by #0 to W0

```
W0 = K123*#0
```



(2) Program which substitutes the result of multiplying #0F by #10 to D0L

```
D0L = #0F*#10
```



The 64-bit floating-point type data are used for multiplication, and the result is converted into the 32-bit integer type and then substituted.

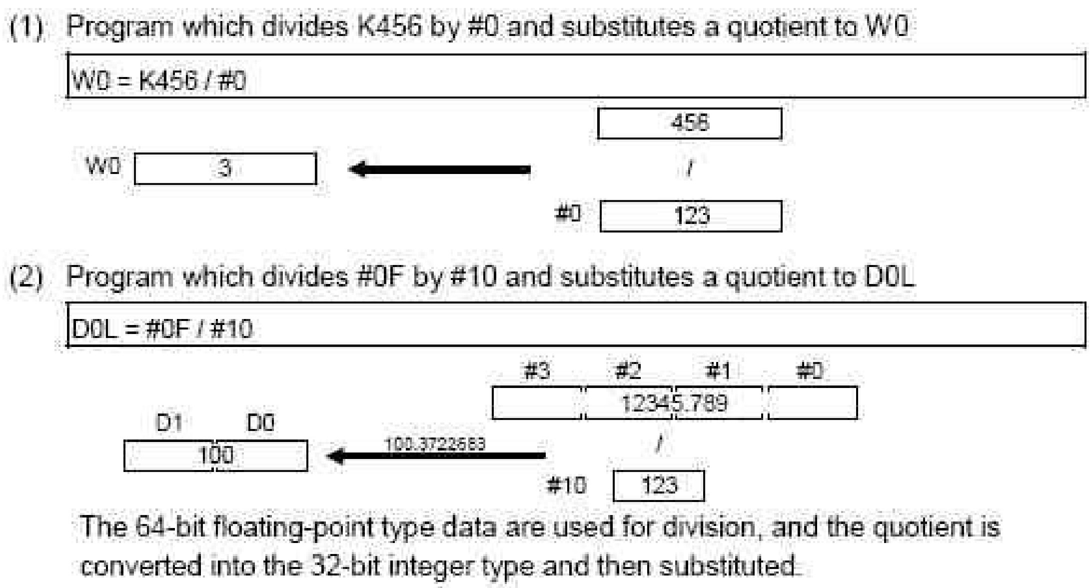
(S1)/(S2) (Division)

The data specified with (S1) is divided by the data specified with (S2).

When (S1) and (S2) differ in data type, the data of the smaller data type is converted into that of the greater type before operation is performed.

Issued by: Mitsubishi Electric Europe B.V. - UK Branch
Document issue: V1.00 June 2006
Produced by : - CTC
Travellers Lane, Hatfield, Herts, AL10 8XB, UK
Author: Matt Wills

Examples taken from SFC Programming manual:


(1) Program which divides K456 by #0 and substitutes a quotient to W0

W0 = K456 / #0

(2) Program which divides #0F by #10 and substitutes a quotient to D0L

D0L = #0F / #10

The 64-bit floating-point type data are used for division, and the quotient is converted into the 32-bit integer type and then substituted.

Therefore in our F1 Program the following occurs:

1. #2L is converted to a 64-bit floating point data type with the Float instruction.

2. #2L is Multiplied by 360 and the answer divided by 16384

3. The result of the above calculation is converted into a 32-bit Integer value and is stored in #4L

Now, let us consider the process if the value of #2L was less than zero, therefore the connected step below will be executed:

```
[F  O]
 #4L=360-ABS(LONG(FLOAT(#2L)*360/16384))
```

Here 2 addition instructions are used:

- (Subtraction)
ABS()

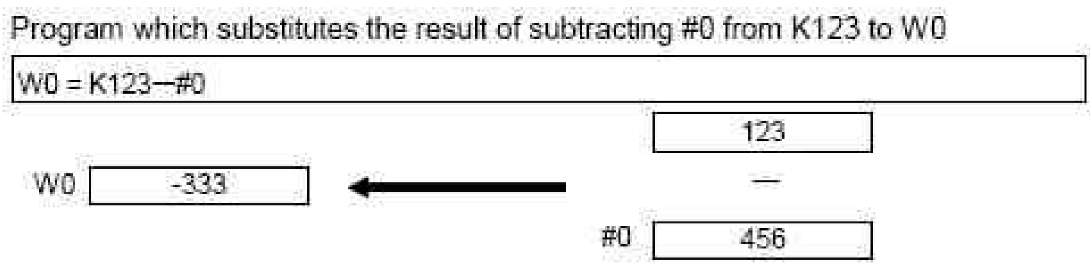
Let us start by explaining each instruction:

(S1)-(S2) (Subtraction)

The data specified with (S2) is subtracted from the data specified with (S1).

When (S1) and (S2) differ in data type, the data of the smaller data type is converted into that of the greater type before operation is performed.

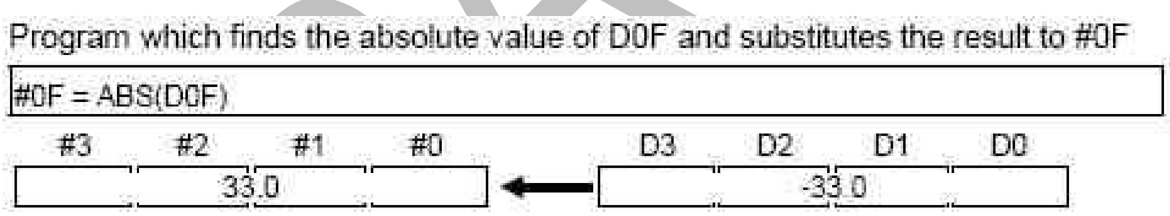
Example taken from SFC Programming manual:



ABS()

The absolute value of the data specified with (S) is found.

Example taken from SFC Programming manual:



Therefore in our F0 Program the following occurs:

1. #2L is converted to a 64-bit floating point data type with the Float instruction.

2. #2L is Multiplied by 360 and the answer divided by 16384

3. The result of the above calculation is converted into a 32-bit Integer value

4. The ABS instruction is executed on the 32-bit Integer value found above and its absolute value is determined

5. The Absolute Value determined above is subtracted from 360 and the result is stored in #4L

In both cases, irrespective of which branch is followed the Pointer PO then repeats the calculation continuously, resulting in a 360 degree ring counter being available in the Number Register **#4L.**