MuM-Anlage_22_05_1 / PLC_1 [CPU 1512SP F-1 PN] / Programmbausteine

LREAL2DINT [FC1]

LREAL2DINT Eigenschaften					
Allgemein					
Name	LREAL2DINT	**Nummer**	1	**Typ**	FC
Sprache	KOP	**Nummerierung**	automatisch		
Information					
Titel	LREAL_TO_DINT	**Autor**	CS2FA	**Kommentar**	---------- ---------- ----- Function/Subroutine name: "LREAL_TO_DINT" ---------- ---------- ----- Description: This function shall be used to convert a 64bit floating point number (IEEE 754 double precision) into a double integer value 32 bit wide. Since this is only for a limited range of values possible status information will be provided as well to indicate a probable loss of data ---------- ---------- ----- Parameters: Input: Name: Data Type Valid Values/ Unit Description/ Meaning IN ARRAY[1..2] OF DWORD [LREAL] Since the S7 CPU 300/400 doesn't support 64bit natively an array of 2 DWORDS must be passed in. Output: Name: Data Type Valid Values/ Unit Description/ Meaning RET_VAL DINT [+/-2147483647] returns the converted value in the maximum possible range STATUS BYTE [bitencoded] contains Status information .0=Overflow .1=Underflow . 2=Less than Zero . 3=Greater than Zero . 4=Lost Significance . 5=Reserved .6=Warning . 7=Error ---------- ---------- ----
Familie	CONVERT	**Version**	0.3	**Anwenderdefinierte ID**	LRL2DINT

LREAL2DINT				
Name	**Datentyp**	**Offset**	**Defaultwert**	**Kommentar**
▼ Input				
▼ IN	Array[1..2] of DWord			[LREAL] value split into 2x 32bit words
IN[1]	DWord			
IN[2]	DWord			
▼ Output				
STATUS	Byte			[bitencoded] contains information about the conversion
InOut				
▼ Temp				
StatusB	Byte	0.0		internal Status information
Sign	Bool	1.0		Sign Bit of the LREAL number
Exponent	Int	2.0		11 Bit Exponent of the LREAL number
Mantissa	DInt	4.0		32 bit out of the 52 bit Mantissa of the LREAL number
Value	DInt	8.0		32 bit Value of denormalized Mantissa
Bias	DInt	12.0		32 bit Bias for the hidden bit
ShiftExp	Int	16.0		number of position the Bias needs to be shifted
ShiftVal	Int	18.0		number of positions the Value needs to be shifted
Constant				
▼ Return				
LREAL2DINT	DInt			

Netzwerk 1: Initialize the Status

eng.: Initializing the Status information de.: Inizialisieung der Statusinformation

```
0001       L     B#16#0                // Initialze everything is OK
0002       T     #StatusB
0003
```

Symbol	Adresse	Typ	Kommentar
#StatusB		Byte	internal Status information

Netzwerk 2: Extraction of Sign Bit [LREAL.X63]

eng.: The Sign Bit is the most significant bit of the LREAL value. In order to access the MSB moving the less significant bits out to the right (SRD). Masking the signbit with a '1' makes sure only the sign value is provided, no matter what is being "pulled" in. de.: Das Vorzeichenbit ist das höchstwertige Bit des LREAL Wertes. An dieser Stelle wird das Vorzeichenbit lokalisiert und entsprechend seines Wertes der Status belegt.

```
0001       L     #IN[1]                // take all the more significant bits
0002       SRD   31                    // push exponent and mantissa out to the
     right
0003       L     1
0004       ==D
0005       =     #Sign
0006
0007       A     #Sign
0008       JC    STSN
0009 // If the LREAL is positive, set the 'Positive' status
0010       L     B#16#1                // Status is still OK
0011       JU    GEXP
0012 // None of the above applies, get the Exponent
0013 STSN: L     B#16#2                // Status is set to 'Less than Zero'
```

```
0014 GEXP: T     #StatusB
0015
0016
0017       L     #IN[1]              // take all the more significant bits
0018       L     DW#16#0
0019       ==D
0020       JCN   STS0
0021       L     B#16#0
0022       T     #StatusB
0023 STS0: NOP 0
0024
```

Symbol	Adresse	Typ	Kommentar
#IN[1]		DWord	[LREAL] value split into 2x 32bit words
#Sign		Bool	Sign Bit of the LREAL number
#StatusB		Byte	internal Status information

Netzwerk 3: Extraction of Exponent [LREAL.X[62 .. 52]]

eng.: select the exponent from the input de.: Auslesen des Exponenten

```
0001       L     #IN[1]              // Accessing only the Most Significant Bytes
0002       SRD   20                  // Only care for the highest 11 bits
0003       AD    DW#16#7FF
0004 // Mask the sign bit (eventually)
0005       T     #Exponent           // store it for later
0006
```

Symbol	Adresse	Typ	Kommentar
#Exponent		Int	11 Bit Exponent of the LREAL number
#IN[1]		DWord	[LREAL] value split into 2x 32bit words

Netzwerk 4: Extraction of the Mantissa [LREAL[51.. 0]]

eng.: In this IEE 754 double precision the Mantissa is stored in the remaining 52 bits. Converting this number the loss of significant digits is assumed The conversion will use a 32 bit internal representation for the mantissa using the most significant bits for it. These 32 bits are split up into the two #IN[] Dwords and will be assembled. This assembly uses OR for combining the numbers and shifting for positioning. Positioning the most significant bits of #IN[1] by moving them to the left by 12 bit positions. Positioning the less significant bits of #IN[2] by moving them to the right by 20 bit positions. Combining them using an OR instruction de.: Ein LREAL Wert besitzt eine 52 Bit große Mantisse. Diese Informationen sind in den Eingangsdoppelwörtern IN[1] und IN[2] enthalten. Nach der lokalisierung dieser Informationen werden sie in die 32 Bit Größe eines REAL Wertes umgewandelt.

```
0001       L     #IN[1]              // take the bits (incl. sign and exponent)
0002       SLD   12                  // push sign and exponent out to the left
0003
0004       L     #IN[2]              // take the less signififcant bits
0005       SRD   20                  // push the non-significant bist out
0006
0007       OD                        // combine the above
0008       T     #Mantissa           // to a new 32 bit mantissa
0009
```

Symbol	Adresse	Typ	Kommentar
#IN[1]		DWord	[LREAL] value split into 2x 32bit words
#IN[2]		DWord	[LREAL] value split into 2x 32bit words
#Mantissa		DInt	32 bit out of the 52 bit Mantissa of the LREAL number

Netzwerk 5: Lost Significance Check

eng.: The conversion will loose significant digits, since the number of bits used for the IEEE 754 double precision representation of the number is way higher than the S7 internally possible (52 bits for IEEE vs. 32 bits for S7) Only if the least significant 20 bitpositions are all 0, there is no loss of significant bits. If the lower 20 bits are all zero a mask with F_FFFF would result in zero (0) de.: Bei der Mantisse-Konvertierung von 52 auf 32 Bit gehen Informationen verloren. Die niederwertigsten 20 Bits werden daher auf 0 überprüft. Wenn dieser Vergleich posituv ist, gehen keine Informationen verloren. Andernfalls wird der Verlust von Informationen über das Statusbyte angezeigt.

```
0001       L     #IN[2]               // take all of the least significant bits
0002       AD    DW#16#FFFFF
0003 // Make only the lower 20 bits visible
0004       L     #StatusB
0005       JZ    NSIG
0006 // If the result of the masking is zero, no loss of significance
0007       OW    W#16#10              // otherwise indicate the loss
0008 NSIG: T     #StatusB             // set the status bit additionally
0009
```

Symbol	Adresse	Typ	Kommentar
#IN[2]		DWord	[LREAL] value split into 2x 32bit words
#StatusB		Byte	internal Status information

Netzwerk 6: Denormalization

Floating Point numbers are typically normalized, which means that the mantissa is always >=1 and <2 . In this case the denormalization realizes the partial removal of the exponent and thus the floating decimal dot. Since the representation is binary based multiplication ends up in shifting to the left (multiply) or to the right (divide). a number Z = (-1)^S * (1.0 + M/2^32)*2^(E-BIAS) a number Z = (-1)^S * (1.0*2^(E-BIAS) + M/(2^32) * 2^(E-BIAS) Z = (-1)^S * (2^(E-BIAS) + M^(E-BIAS-32)) or Z = (-1)^S * Bias + Value

```
0001       L     #Exponent            // Create the Exponent for the Bias
0002       +     -1023                // ExpB = Exponent - BIAS = Exponent - 1023
0003       T     #ShiftExp            // store for later use
0004       +     -32                  // the Value Exponent is another 32 Bits
     shifted
0005       T     #ShiftVal            // ExpV = Exponent - BIAS - 32 = Exponent -
     1023 - 32
0006
0007       L     0                    // If ExpV is negative a Division is required
0008       >I                         // which is represented by a shift to the
     right
0009       JCN   NVAL
0010 // Right Shift will be done starting at NVAL
0011
0012       L     #ShiftVal            // ExpV is non negative which means a multi-
     plication
0013       L     #Mantissa            // Value = Mantissa * 2^ExpV
0014       SLD
0015       JU    VAL
0016 NVAL: L     #ShiftVal            // ExpV is negative which means a division
     has to be done
0017       NEGI                       // by how many positions (negative positions
     can't be shifted)
0018       T     #ShiftVal
0019       L     #Mantissa            // Value = Mantissa / 2*ExpV
0020       SRD
0021 VAL:  T     #Value               // Value (first part of the number)
0022       L     #ShiftExp            // Is the ExpB a negative number, then a
     division is required
0023       L     0
```

```
0024        >I
0025        JCN    NEXP
0026  // the division will be executed starting at NEXP
0027        L      #ShiftExp            // ExpB is a non negative number which
      requires a multiplication
0028        L      DW#16#1              // Bias = 1 * 2^ExpB
0029        SLD
0030        JU     EXPO
0031
0032  NEXP: L      #ShiftExp            // ExpB is negative which requries a division
0033        NEGI                        // shifting by negative number of positions
      isn't possible
0034        T      #ShiftExp
0035        L      DW#16#1              // Bias = 1 / 2^ExpB
0036        SRD
0037
0038  EXPO: T      #Bias                // Bias (second part of the number)
0039
0040        L      #Value
0041        +D                          // Z = Value + Bias
0042        A      #Sign
0043  // Is the LREAL a negative number ?
0044        JCN    POS
0045  // If Sign is not set, LREAL was a positive number
0046        NEGD                        // Sign is set, Z needs to be negated (Z = -
      Z)
0047  POS:  T      #LREAL2DINT          // return Z
0048        L      #StatusB
0049        T      #STATUS              // return Status
```

Symbol	Adresse	Typ	Kommentar
#Bias		DInt	32 bit Bias for the hidden bit
#Exponent		Int	11 Bit Exponent of the LREAL number
#LREAL2DINT		DInt	
#Mantissa		DInt	32 bit out of the 52 bit Mantissa of the LREAL number
#ShiftExp		Int	number of position the Bias needs to be shifted
#ShiftVal		Int	number of positions the Value needs to be shifted
#Sign		Bool	Sign Bit of the LREAL number
#STATUS		Byte	[bitencoded] contains information about the conversion
#StatusB		Byte	internal Status information
#Value		DInt	32 bit Value of denormalized Mantissa