Isochronous_MasterProg_V18_ThesisPics / Sec_PLC_ET200SP [CPU 1514SP-2 PN] / Program blocks / Iso-Fast_AnalogInputChannel

IsoFast_AnalogInputChannel_FB [FB10]

IsoFast_AnalogInputChannel_FB Properties							
General							
Name	IsoFast_AnalogInputChan-nel_FB	**Number**	10	**Type**	FB	**Language**	SCL
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

```
0001 REGION BLOCK_INFO_HEADER
0002 (/*
0003 ================================================================================
0004  HAHN GROUP / (c)Copyright 2023
0005  ------------------------------------------------------------------------------
0006  Title: IsoFast_AnalogInputChannel_FB
0007  Comment/Function: This function block does the preprocessing to the Analog input Raw samples.
0008  Operation such as scaling(Offset and gain), Filterring and bufferring is performed.
0009  Library/Family: IsochronousFast(IsoFast)/HighSpeed(HS)
0010  Author: Sunish Suresh(Master Thesis)
0011  Department: R&D
0012  Tested with: FunctionGenerator
0013  Engineering: TIA Portal V17/V18
0014  Restrictions: None
0015  Requirements: PLC (S7-1500 with HighSpeed Analog Input cards and Isochrnous mode
0016  and oversampling )
0017  ------------------------------------------------------------------------------
0018  Change log table:
0019  Version | Date | Engineer in charge | Changes applied
0020  ----------|------------|------------------------|-----------------------------
0021  01.00.00 | 20.09.2018 | Sunish Suresh | First released version
0022  01.00.02 | 18.10.2023 | Sunish Suresh | Insert documentation
0023  01.00.03 | 09.12.2023 | Sunish Suresh | Added cases for different filterring
0024 ================================================================================
0025 */)
0026 END_REGION Block info header
0027
0028 REGION DESCRIPTION
0029 (/*
0030 The function scales raw input values into engineering units using the formula:
0031
0032 Scaled Value = Gain * Raw Value + Offset
0033
0034 Based on the type of the raw input (`Word`, `Int`, `Real`), the function employs
0035 appropriate type conversion before scaling.
0036 */)
0037
0038 END_REGION DESCRIPTION
0039
0040 REGION SCALING // Article dependent paramerter
0041
0042   FOR #i := 0 TO ("oversampling") BY 1 DO
0043     #Interface.InterfaceInout.iCurrentRawArr[#i]:= WORD_TO_INT(#io_wRawArr[#i]);
0044     #Interface.InterfaceInout.rCurrentScaledArr[#i] := #Interface.parameters.rGain * INT_TO_REAL(#Interface.Interfa-
     ceInout.iCurrentRawArr[#i]) + #Interface.parameters.rOffset;
0045   END_FOR;
0046
0047 END_REGION SCALING
0048
0049 REGION FILTER
0050   // Extend the input array with previous values
0051   FOR #i := 0 TO MIN(IN1 := #Interface.parameters.usiWindowSize - 1, IN2 := "oversampling") DO
0052     #ExtendedArray[#i] := #LastValuesPrevCyc[#i];
0053   END_FOR;
0054   FOR #i := 0 TO "oversampling" DO
0055     #ExtendedArray[#i + #Interface.parameters.usiWindowSize] := #Interface.InterfaceInout.rCurrentScaledArr[#i];
0056   END_FOR;
0057
0058   CASE #Interface.parameters.usiFilterType OF
0059
0060     0: // No Filterring
0061       #Interface.InterfaceInout.rCurrentFilteredArr[#i] := #Interface.InterfaceInout.rCurrentScaledArr[#i];
0062
0063     1: // Simple Moving Average
0064        // Calculate the moving average
0065       FOR #i := 0 TO "oversampling" DO
0066         #sum := 0;
0067         FOR #j := #i TO MIN(IN1 := #i + #Interface.parameters.usiWindowSize - 1, IN2 := "oversampling" + 1 + #Inter-
     face.parameters.usiWindowSize - 1) DO
0068           #sum := #sum + #ExtendedArray[#j];
0069         END_FOR;
0070         #avg := #sum / (MIN(IN1 := #Interface.parameters.usiWindowSize, IN2 := "oversampling" + 1));
0071         #Interface.InterfaceInout.rCurrentFilteredArr[#i] := #avg;
```

```
0072        END_FOR;
0073
0074      2:  // Weighted Moving Average
0075        // Calculate the moving average
0076        FOR #i := 0 TO "oversampling" DO
0077          #sum := 0;
0078          #totalWeight := 0;
0079          FOR #j := #i TO MIN(IN1 := #i + #Interface.parameters.usiWindowSize - 1, IN2 := "oversampling" + #Interface.pa-
      rameters.usiWindowSize) DO
0080            #weight := (#j - #i) + 1;                          // Linearly increasing weight
0081            #sum := #sum + (#ExtendedArray[#j] * #weight);
0082            #totalWeight := #totalWeight + #weight;
0083          END_FOR;
0084          #Interface.InterfaceInout.rCurrentFilteredArr[#i] := #sum / #totalWeight;
0085        END_FOR;
0086
0087
0088    END_CASE;
0089
0090    // Update previous values for the next cycle
0091    FOR #i := 0 TO MIN(IN1 := #Interface.parameters.usiWindowSize - 1, IN2 := "oversampling") DO
0092      #LastValuesPrevCyc[#i] := #Interface.InterfaceInout.rCurrentScaledArr["oversampling" + 1 - #Interface.parame-
      ters.usiWindowSize + #i];
0093    END_FOR;
0094
0095  END_REGION FILTER
0096
```