Application Note

AN000651

AS7341 Auto Gain & Optimization

AS7341 Using Auto Gain in GUI and suggestion for software algorithm

v0-01 • 2019-Oct-21

Content Guide

1 Introduction

Auto Gain & optimization will automatically find the best (maximum possible) parameters options for Gain to get the maximum optimized raw in a defined range. Optimization analyzes sensor output and gain parameter setup to find an optimal ADC value based various calculation of gain.

2 Optimization in GUI

In main Demo GUI for AS7341 by default, Auto Gain is ON. Optimization of gain is enabled by checking “Max AGAIN” and denoting the maximum gain value to be considered in the list box besides the checkbox. Similarly optimization of Integration time is enable by checking Max TINT and mentioning the maximum value to consider in up down list. The Gain and time should be checked for optimization.

It is suggested to disable the “Optimized Gain Detection” after taking one measurement as it keep changing the parameter values for measurement to achieving an optimized raw value in each measurement if they are enabled

3 Algorithm and flow chart of optimization used in GUI

The steps in Auto Gain optimization is divided into two section. A part where the Auto Gain is found and the other part optimizes the derived Auto Gain. In Auto Gain section, a gain between the maximum and minimum range is calculated automatically by the results of a test measurement. Therefore, sensors raw value is placed closer to the maximum as possible without saturation.

3.1 Steps in algorithm - Search

- Middle Gain value is calculated from the maximum possible gain for optimization and taken as the currentGain

```
currentGain = (byte)(cobMaxGain.Items.Count / 2.0 + 0.5);
```

- If currentGain is greater than the given maxGain(user given Maximum gain for optimization). Then, currentGain will be equal to the given maxGain.

```
if (currentGain > maxGain)
    {
        currentGain = maxGain;
    }
```

- Inside the while loop, the gain of the device is set to currentGain. Then, reads out the raw measurement and checks the saturation or noise state of rawValue measurements. If any of the conditions is true, it will enter the corresponding loop.

  If the raw value is above the maximum range of the raw values(RawValueStates.Saturation) gain correction is made by reducing the gain by half of the currentGain using below algorithm.

```
while (true)

{

_sensor.setGain(currentGain);

rawValueState = CheckRawValues(ref checkState, ref rawVal, ref basicVal, ref
corrVal);

measureCount++;



    if (rawValueState == RawValueStates.Saturation)

        {

if (currentGain == 0)

              {

                break;

             }

             // Set the saturation gain flag

              if (currentGain < saturationGain)

             {

             saturationGain = currentGain

             }

              // set new gain value by reducing to half of current gain by right
shift method

            currentGain >>= 1;

     }
```

- Otherwise, if the raw value is below the minimum range of raw values (RawValueStates. Noise). The gain correction is made by increasing the gain when it is in noise state is shown in below algorithm.

```
else if (rawValueState == RawValueStates.Noise)

        {

// in case of low gain value use the middle between max and current gain

            if (currentGain == maxGain)

            {

                break;

            }

            newGain = (byte)((maxGain + currentGain) / 2.0 + 0.5);

            if (newGain == currentGain)

            {

                newGain++;

            }

            // check if new gain value greater Tan saturation gain

            if (newGain >= saturationGain)

            {

                break;

            }

// currentGain takes the newGain

            currentGain = newGain;

        }
```

If still the rawValueState is in saturation, the optimization is not possible. The error is handled in below method.

```
if (rawValueState == RawValueStates.Saturation)

                    {

                        lblOptimizationError.ForeColor = Color.Red;
```

```
                        lblOptimizationError.Text = "Optimization not possible due
to saturation";

                        return errorcode;

                    }

                }
```

3.2 Steps in algorithm - Calculation

As discussed before now we have a gain, which reads raw values between the Saturation and Noise. Next step is to calculate an optimized gain such a way that raw values is closer to the maxRawVal limit.

Maximum value of current raw value is taken from the measured raw values. Range of Maximum Raw value and minimum raw value is calculated based on below parameter.

```
UInt16 currentRawVal = rawVal.Max();

    double maxRawVal = _sensor.MaxCounts * _maximumAdcRange;

    double minRawVal = _sensor.MaxCounts * _minimumAdcRange;
```

where,

_sensor.MaxCounts : the maximum number of counts for the current ATIME and ASTEP value

_maximumAdcRange: 0.90

_minimumAdcRange: 0.50

The logarithmic value of maxRawVal divided by currentRawvalue is calculated with a base of 2. Then this value rounded to the largest integer Less than or equal to that value.

```
  int diffGain = (int)Math.Floor(Math.Log(maxRawVal / currentRawVal, 2));
```

Considering an example of this calculation, maximumRaw value is 50000 and currentRawvalue is 30000. Idea behind above step is to scale 30000 so to reach closer 50000. Since gain in AS7341 is a multiple of 2, the base of calculation 2.

30000 x 2^x = 50000

$X = \log_2(maxRawvalue/CurrentRawValue)$

$X = \log_2(50000/30000)$

When the currentGain added with the diffGain is greated than the maxGain (user given maximum gain for optimization). Then, the diffGain is calclated as the difference of maxGain and currentGain.

```
if (currentGain + diffGain > maxGain)
{
    diffGain = maxGain - currentGain;
}
```

If the diffGain is greater than or equal to zero, the currentRawValue is incremented by a left shift of diffGain times. On the other hand, if the difference gain is negative, the currentRawValue is decrement by a right shift of (-)diffGain times.

```
currentRawVal = (UInt16)(diffGain >= 0 ? currentRawVal << diffGain : currentRawVal >> -diffGain);
```

Current gain is added with the calculated diffGain

```
currentGain = (byte)((int)currentGain + diffGain);
```

Optimized Gain is set as the Gain. Finally the raw value measurement with optimized gain is made.

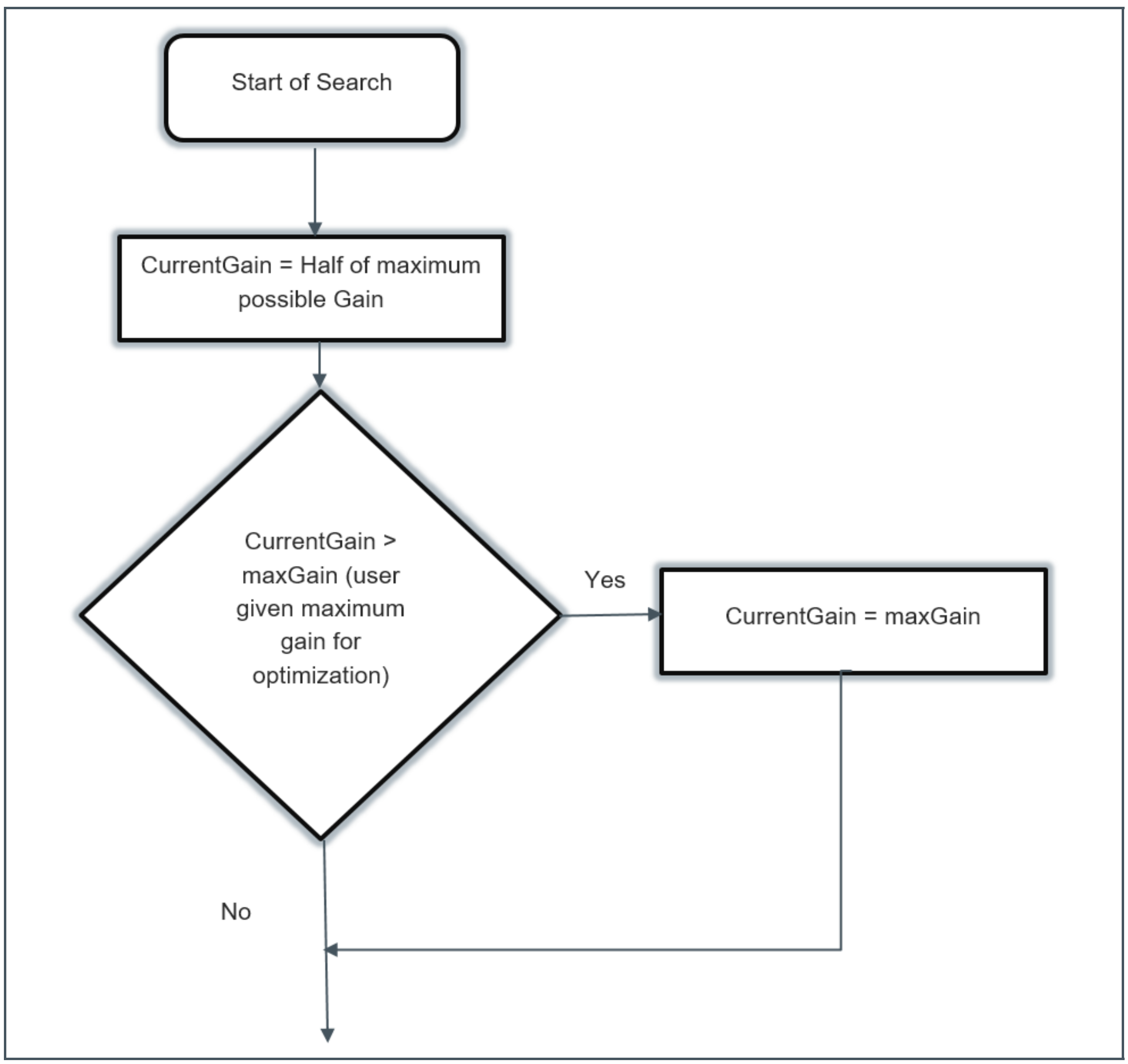
```
_sensor.setGain(currentGain);
```

```
// measurement with optimized values
getMeasurementValues(ref checkState, ref rawVal, ref basicVal, ref corrVal);
```

3.3 FlowChart of algorithm

3.3.1 Search for Gain between Saturation and Noise

Figure 1:
Search for Gain between Saturation and Noise



3.3.2 While loop for searching Auto Gain

Figure 2:
While loop for searching Auto Gain – Part I

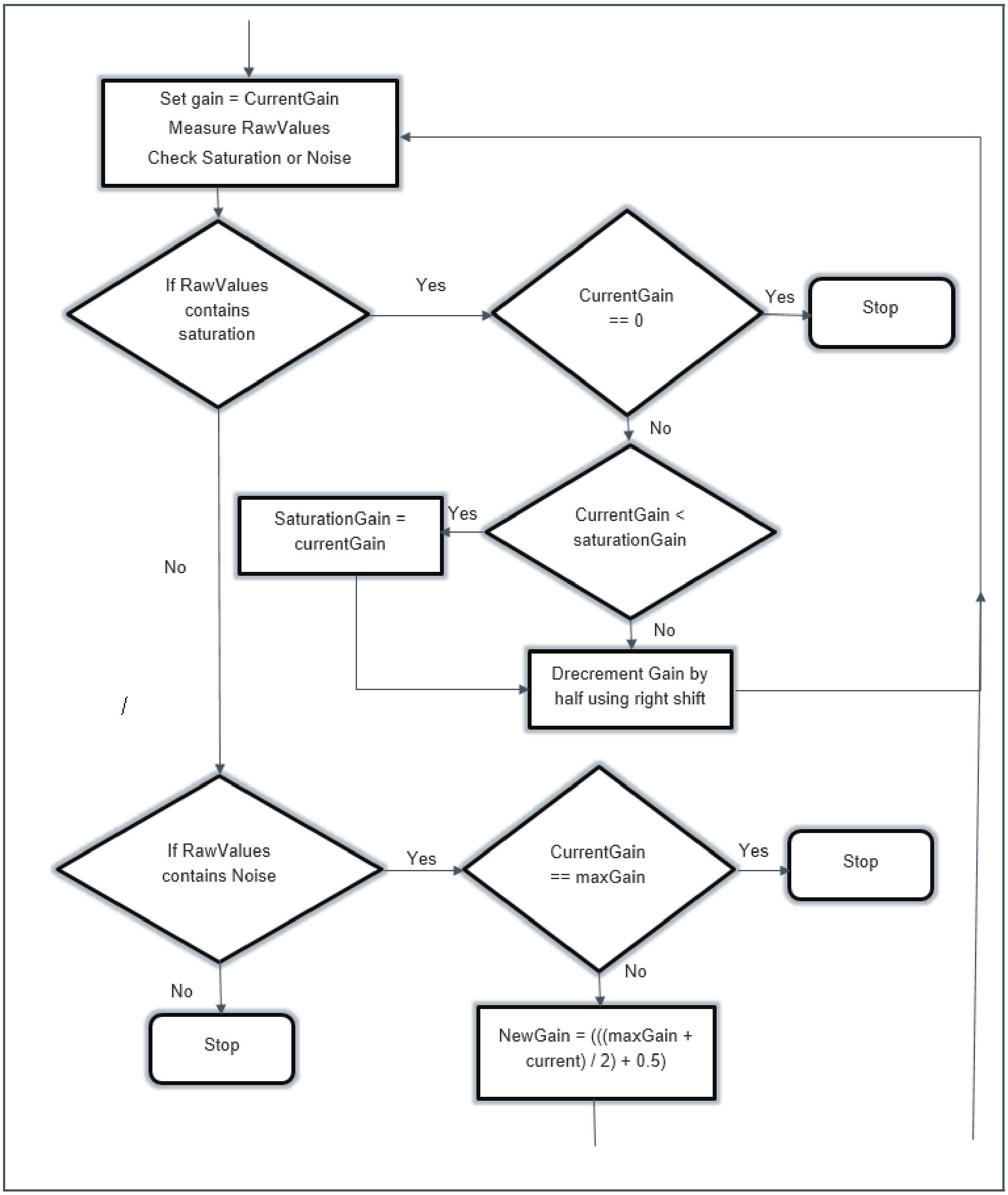
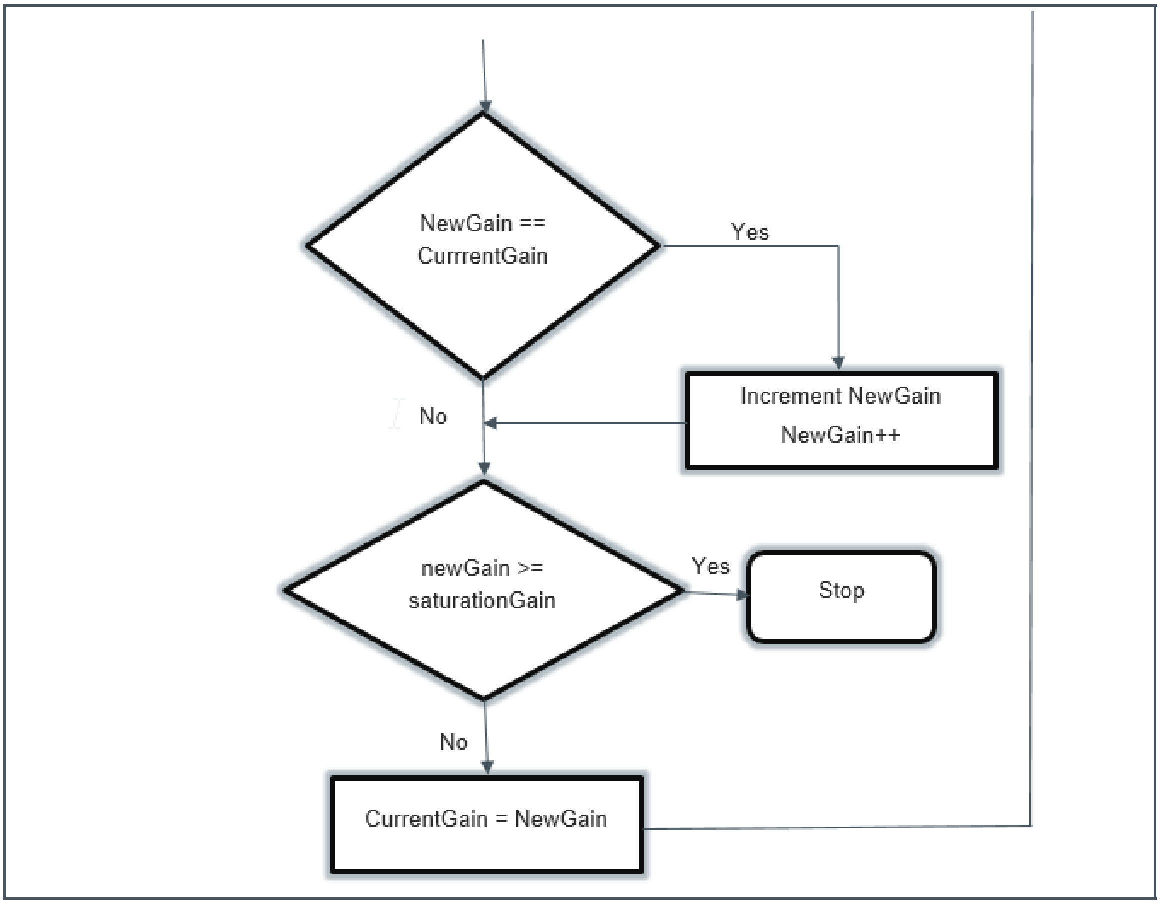


Figure 3:
While loop for searching Auto Gain – Part II



3.3.3 Calculation of optimized gain closer to maximum limit

Figure 4:
Calculation of optimized gain closer to maximum limit – Part I

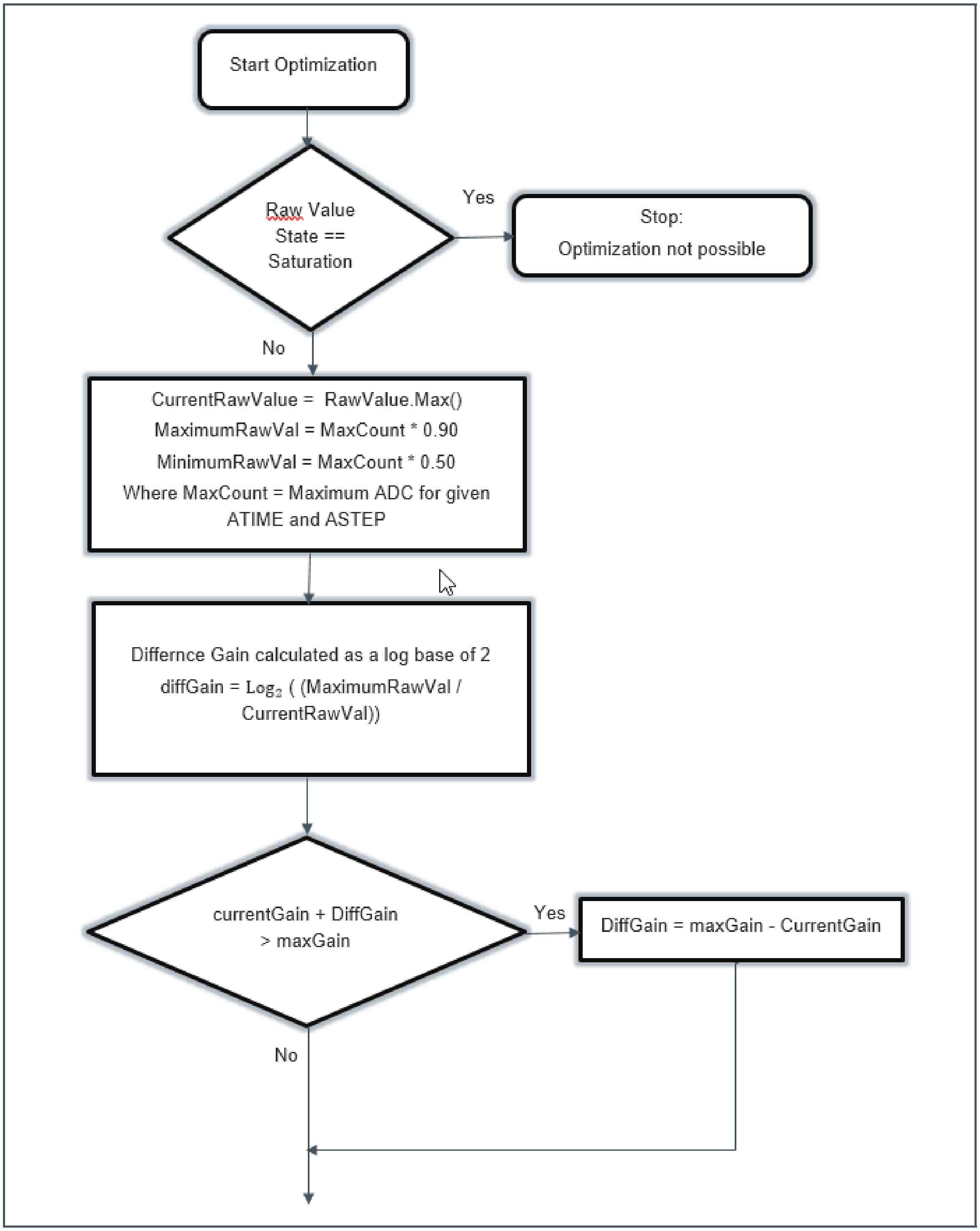
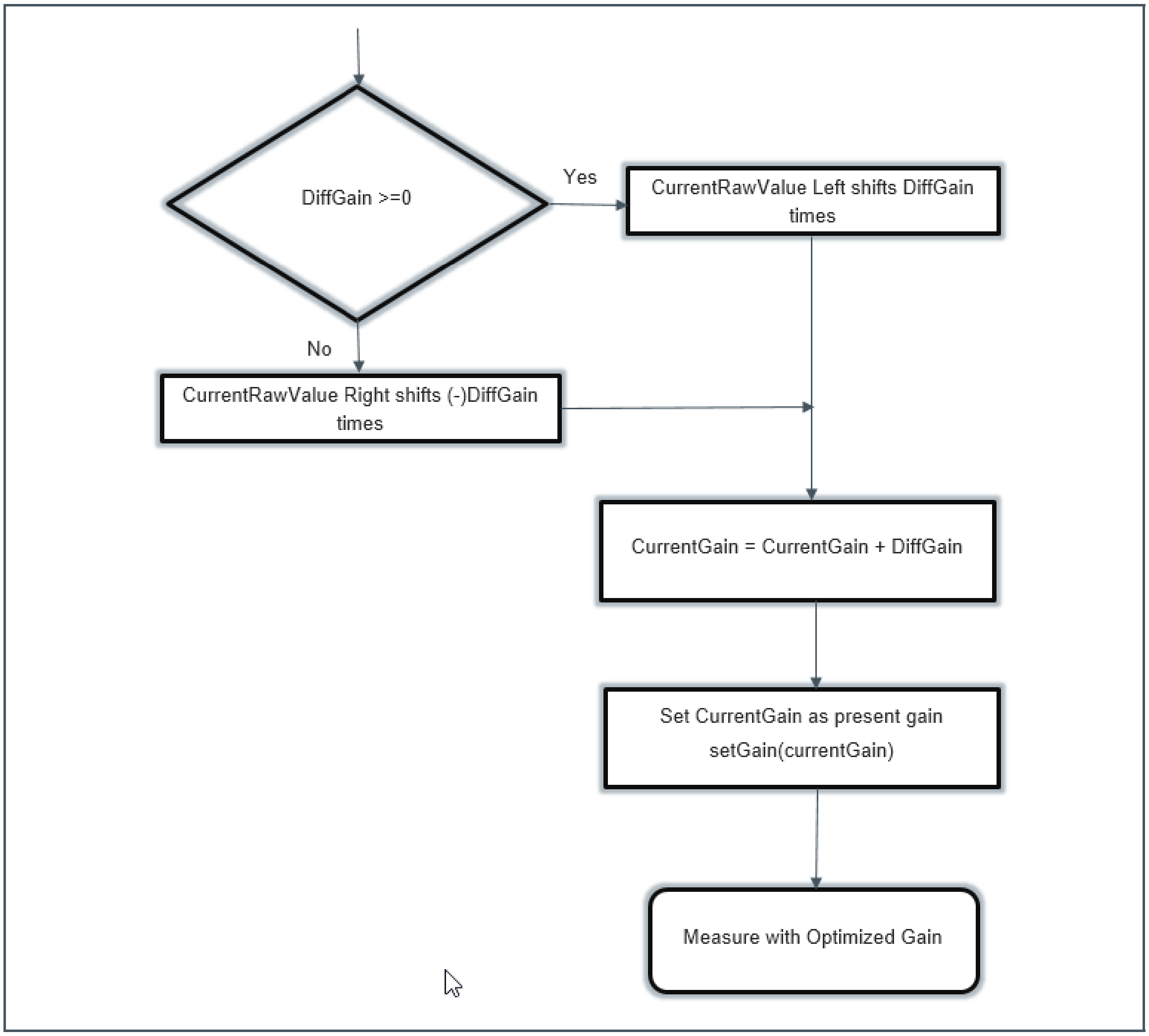


Figure 5:
Calculation of optimized gain closer to maximum limit – Part II



4 Example Code of Automatic Gain optimization Algorithm

```
private byte getOptimizedMeasurementValues(ref bool checkState, out bool
optimizedValuesDetected, ref UInt16[] rawVal, ref double[] basicVal, ref double[]
corrVal)

{

    RawValueStates rawValueState = RawValueStates.Saturation;

    // start with middle gain value

    currentGain = (byte)(cobMaxGain.Items.Count / 2.0 + 0.5);

    if (currentGain > maxGain)

    {

        currentGain = maxGain;

    }

    byte newGain;

      byte saturationGain = (byte)(cobMaxGain.Items.Count + 1);

    while (true)

    {

        // set gain value

        errorcode = _sensor.setGain(currentGain);

        if (errorcode != (byte)As7341Errorcodes.OK)

        {

            throw new Exception("Error from setGain: " +
((As7341Errorcodes)errorcode).ToString());

        }

        // measure and check raw vlues
```

```
        rawValueState = CheckRawValues(ref checkState, ref rawVal, ref basicVal,
ref corrVal);

        measureCount++;

        if (rawValueState == RawValueStates.Saturation)

        {

            BaseFunctions.DebugOut(true, "Saturation gain: " +
currentGain.ToString());

            // in case of saturation have the gain value

            if (currentGain == 0)

            {

                break;

            }

     // current gain less than saturation gain, then setting saturationGain equals
currentGain

             if (currentGain < saturationGain)

             {

                 saturationGain = currentGain

             }

            // set new gain value

            currentGain >>= 1;

        }

        else if (rawValueState == RawValueStates.Noise)

        {

            BaseFunctions.DebugOut(true, "Noise gain: " + currentGain.ToString() +
" Raw: " + rawVal.Max().ToString());

            // in case of low gain value use the middle between max and current
gain

            if (currentGain == maxGain)
```

```
                {
                    break;
                }
                newGain = (byte)((maxGain + currentGain) / 2.0 + 0.5);
                if (newGain == currentGain)
                {
                    newGain++;
                }

// check if new gain value is greater than saturationGain flag
                if (newGain >= saturationGain)
                {
                    break;
                }
                currentGain = newGain;
            }
            else
            {
BaseFunctions.DebugOut(true, "Ok gain: " + currentGain.ToString() + " Raw: " +
rawVal.Max().ToString());
                break;
            }
        }

        // check for saturation
        if (rawValueState == RawValueStates.Saturation)
```

```
    {
        lblOptimizationError.ForeColor = Color.Red;
        lblOptimizationError.Text = "Optimization not possible due to saturation";
        return errorcode;
    }

    // set values for optimization
    UInt16 currentRawVal = rawVal.Max();
    double maxRawVal = _sensor.MaxCounts * _maximumAdcRange;
    double minRawVal = _sensor.MaxCounts * _minimumAdcRange;

    // optimize gain
    int diffGain = (int)Math.Floor(Math.Log(maxRawVal / currentRawVal, 2));
    if (currentGain + diffGain > maxGain)
    {
        diffGain = maxGain - currentGain;
    }
currentRawVal = (UInt16)(diffGain >= 0 ? currentRawVal << diffGain : currentRawVal
>> -diffGain);
currentGain = (byte)((int)currentGain + diffGain);

    // set gain value
    errorcode = _sensor.setGain(currentGain);
    if (errorcode != (byte)As7341Errorcodes.OK)
    {
```

```
        throw new Exception("Error from setGain: " +
(As7341Errorcodes)errorcode).ToString());

    }


    // show current values

    cobAgain.SelectedIndex = currentGain;

    lblResultAgain.Text = _sensor.calculateGain(currentGain) + "x";


    // measurement with optimized values

    errorcode = getMeasurementValues(ref checkState, ref rawVal, ref basicVal, ref
corrVal);

    lblOptimizationMessage.ForeColor = Color.Black;

    lblOptimizationMessage.Text = "Optimization measurements: " + ++measureCount;

    return errorcode;

}
```

5 Revision Information

Changes from previous version to current revision v0-01	Page

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

6 Legal Information

RoHS Compliant & ams Green Statement

RoHS Compliant: The term RoHS compliant means that ams AG products fully comply with current RoHS directives. Our semiconductor products do not contain any chemicals for all 6 substance categories, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes.

ams Green (RoHS compliant and no Sb/Br): ams Green defines that in addition to RoHS compliance, our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

Important Information: The information provided in this statement represents ams AG knowledge and belief as of the date that it is provided. ams AG bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ams AG has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ams AG and ams AG suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

Headquarters

ams AG

Tobelbader Strasse 30

8141 Premstaetten

Austria, Europe

Tel: +43 (0) 3136 500 0

Please visit our website at www.ams.com

Buy our products or get free samples online at www.ams.com/Products

Technical Support is available at www.ams.com/Technical-Support

Provide feedback about this document at www.ams.com/Document-Feedback

For sales offices, distributors and representatives go to www.ams.com/Contact

For further information and requests, e-mail us at ams_sales@ams.com