Application Note

AN000651

AS7341 Auto Gain & Optimization

AS7341 Using Auto Gain in GUI and suggestion for software algorithm

v0-01 • 2021-Jan-22

Content Guide

1 Introduction

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

2 Optimization in GUI

In the main Demo GUI for AS7341, by default, Auto Gain is ON. Optimization of the 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 enabled by checking “Max TINT” and mentioning the maximum value to consider in the drop-down list. The Gain and time should be checked for optimization.

Disable the “Optimized Gain Detection” after taking one measurement, as it keeps changing the parameter values for measurement - to achieve 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 sections. One part contains the Auto Gain, and the second part optimizes the derived Auto Gain. In the Auto Gain section, a gain between the maximum and minimum range is automatically calculated by the results of the test measurement. Therefore, the sensor's raw value is placed as close to the maximum as possible, without saturation.

3.1 Steps in algorithm - Search

1. 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);
```

2. 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;
    }
```

3. 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 are 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 the algorithm below.

```
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;

     }
```

4. 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 the algorithm below.

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

```
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 in chapter 3.1, now we have a gain, which reads raw values between the Saturation and Noise. The next step is to calculate an optimized gain in such a way that the raw values are closer to the maxRawVal limit.

1. The maximum value of the current raw value is taken from the measured raw values. The range of the maximum raw value and minimum raw value is calculated based on the parameter below.

```
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

2. The logarithmic value of maxRawVal divided by currentRawvalue is calculated with a base of two. This value is then rounded to the largest integer, less than or equal to that value.

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

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

   $30000 \text{ x } 2\text{^x} = 50000$

   $X = \log_2(maxRawvalue/CurrentRawValue)$

   $X = \log_2(50000/30000)$

4. When the currentGain added with the diffGain is greater than the maxGain (user given maximum gain for optimization), the diffGain is calculated as the difference between maxGain and currentGain.

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

5. 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 in gain is negative, the currentRawValue is decrement by a right shift of (-)diffGain times.

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

6. Current gain is added with the calculated diffGain.

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

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

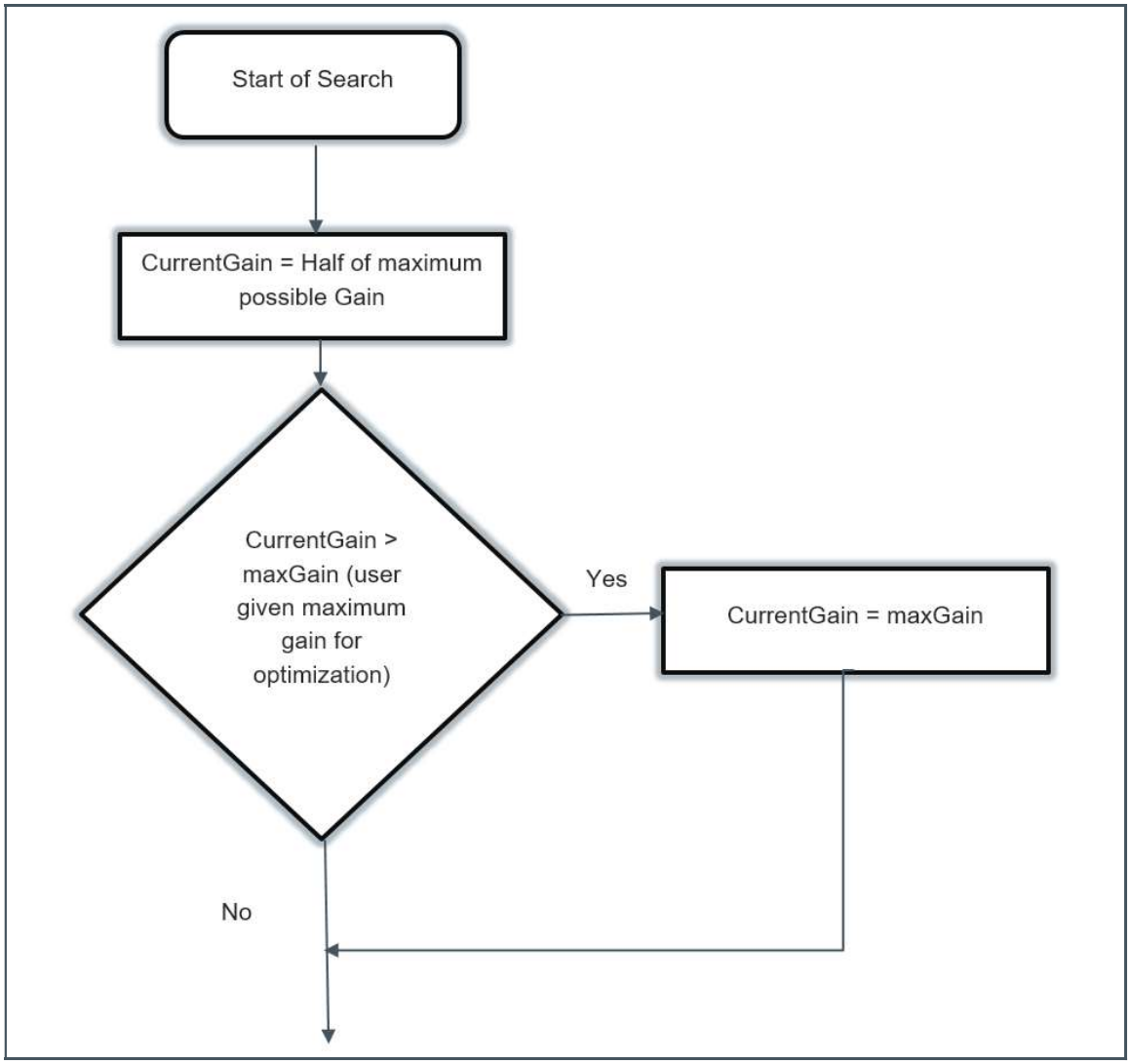
```
_sensor.setGain(currentGain);


// 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

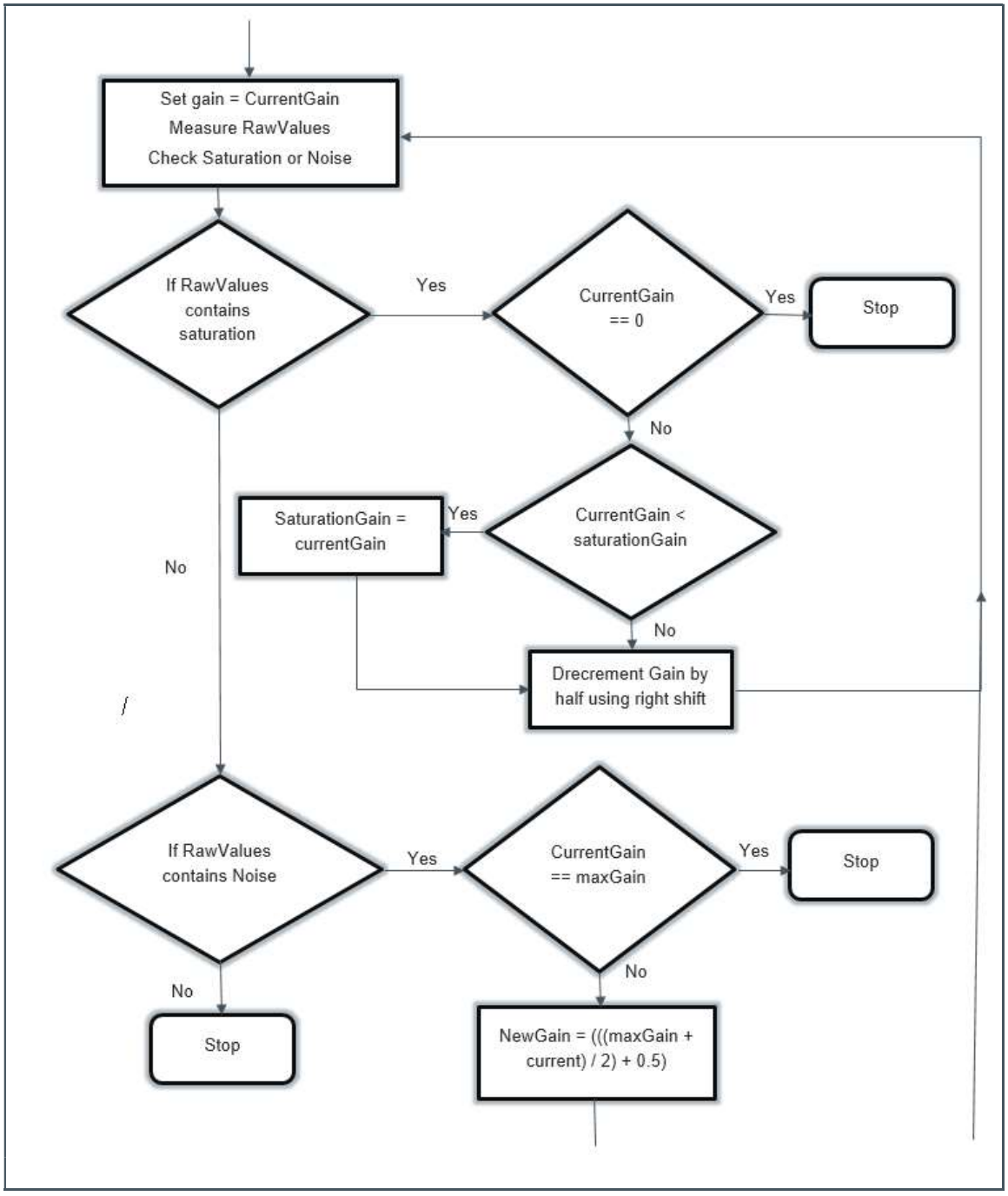
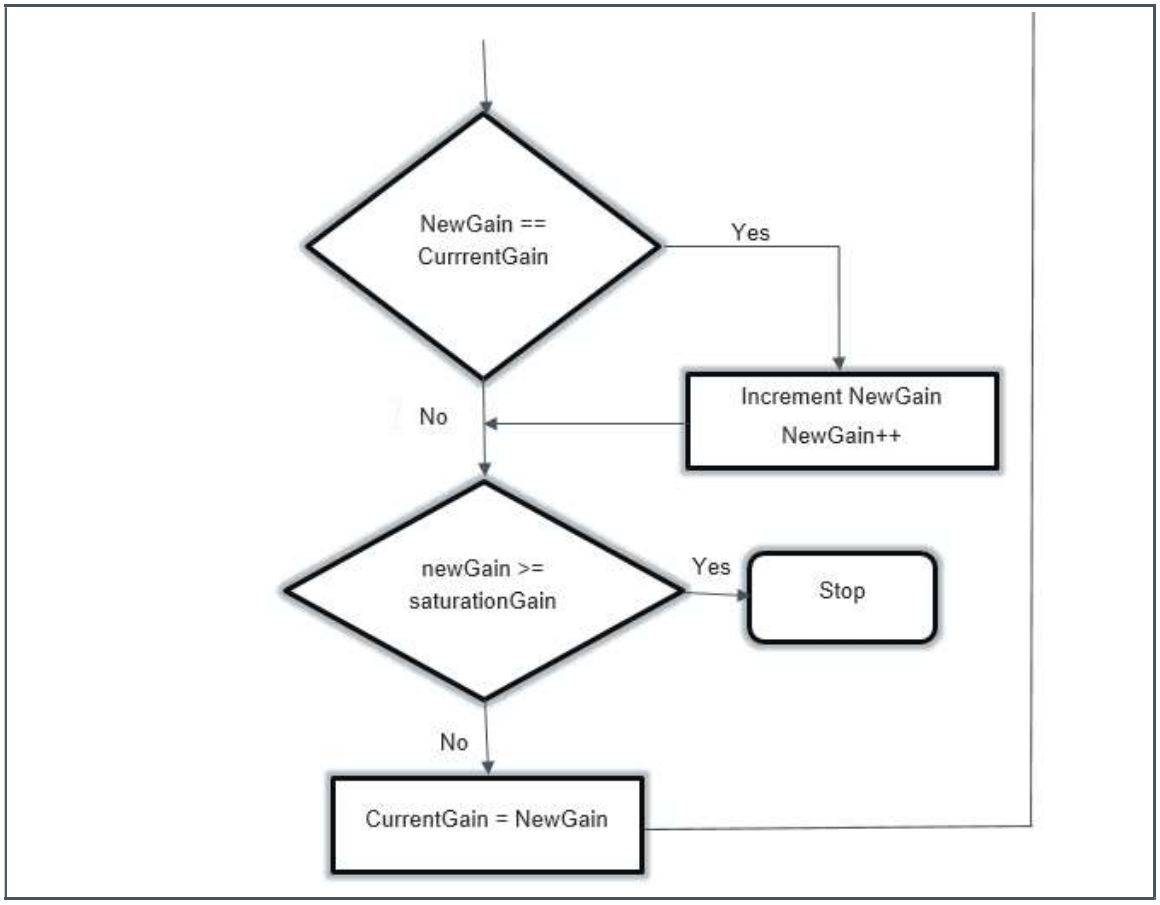


Figure 3:
While loop for searching Auto Gain – Part II



3.3.3 Calculation of optimized gain closer to maximum limit

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

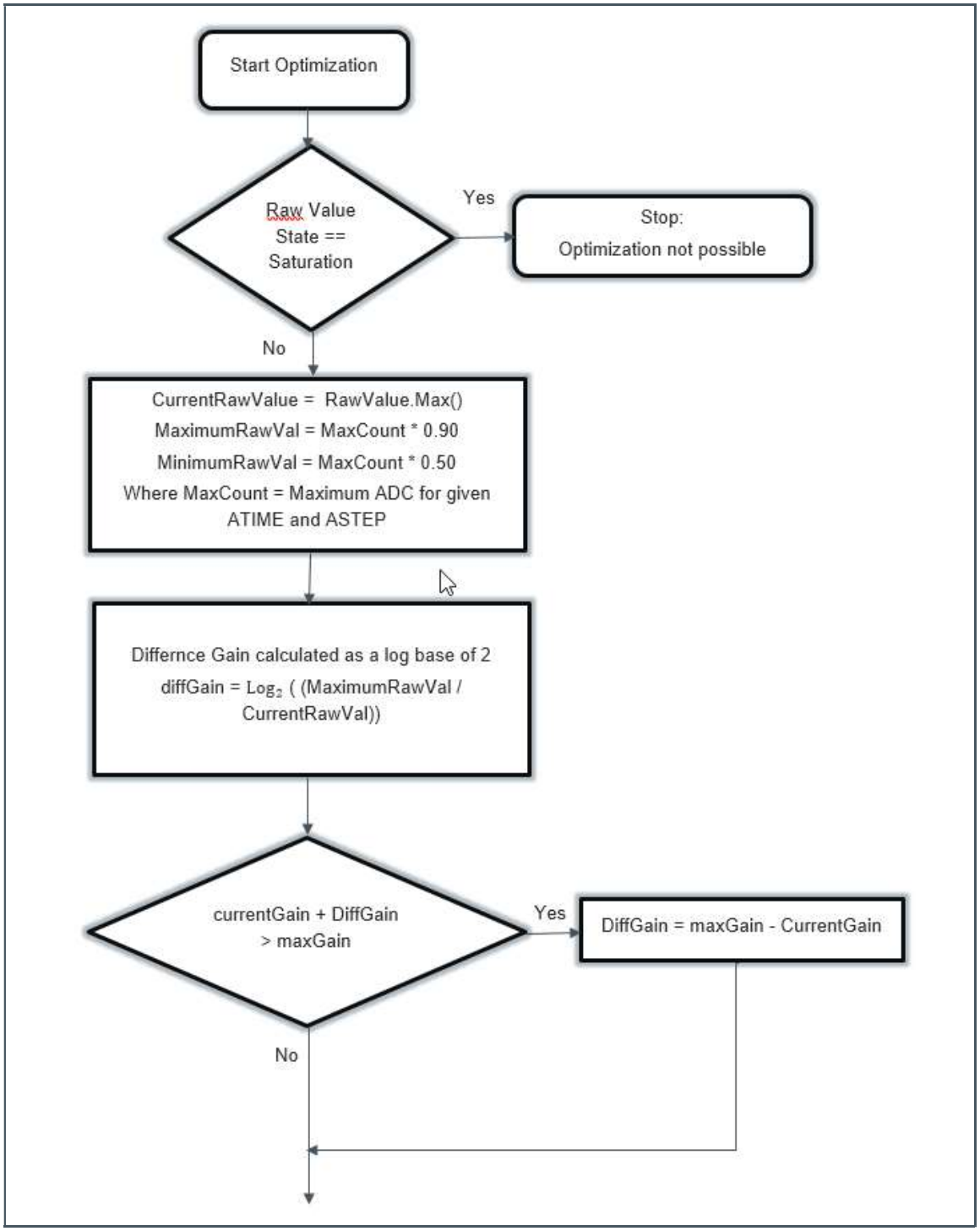
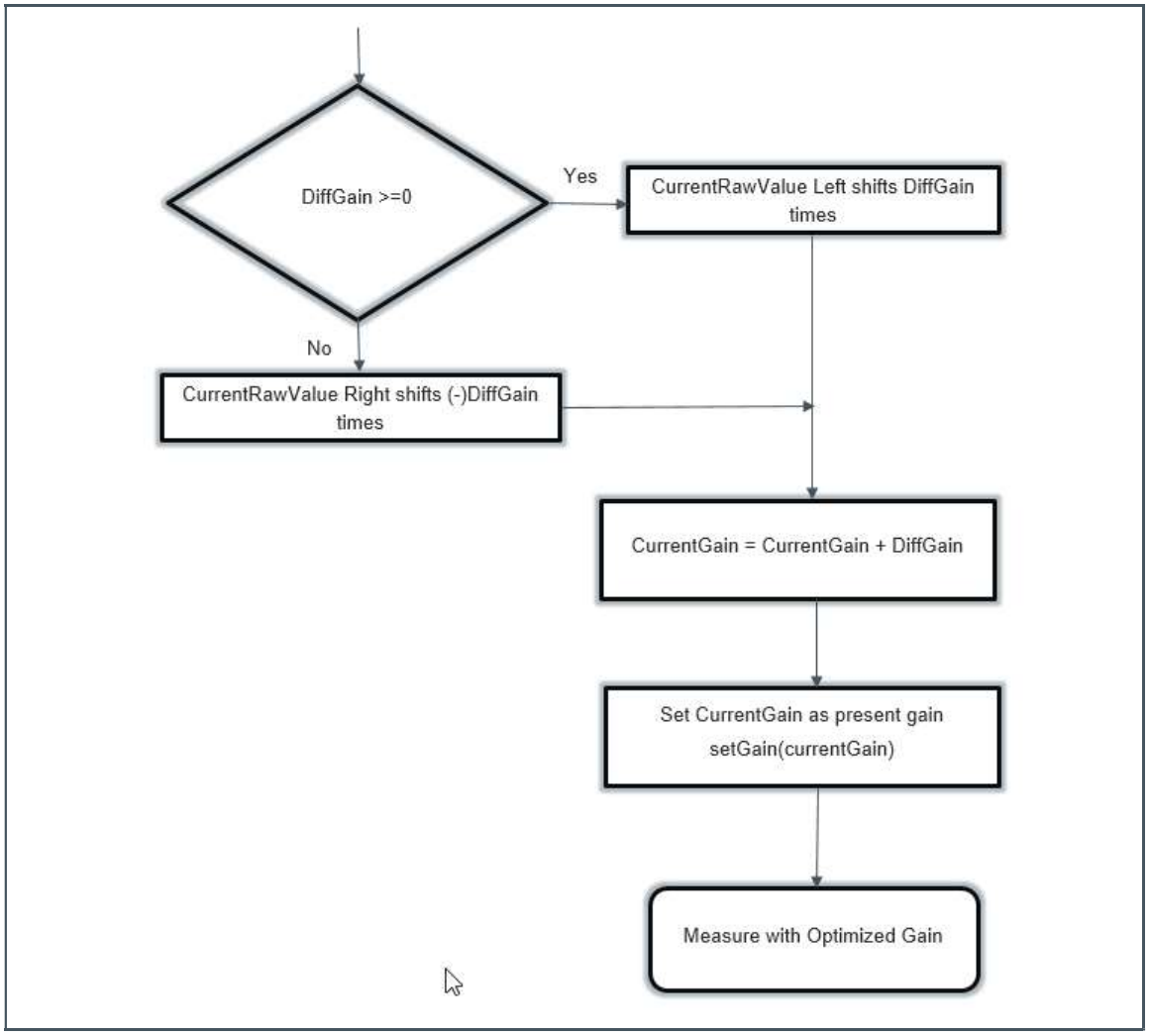


Figure 5:
Calculation of optimized gain closer to maximum limit – Part II



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

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