DEEP SOUND™

Getting Started Manual

Deepsound Software Development Kit

INC, which shall not be liable for any errors or inaccuracies in the information contained in this guide. Existing illustrations or images that you wish to include in your project may be protected by copyright law. You must obtain permission from the copyright owner.
References to company names, company logos, and user names in sample materials or sample forms included in this manual and/or software are solely for its purpose and do not refer to any actual organization or person. DEEPSOUND is a registered trademark of Seongsan Research Institute in Korea and is the property of its owner.

Contact Information:
SEONGSAN LAB INC

P : 02-2039-5725
F: 02-2039-5726
www.seongsanlab.com
admin@seongsanlab.com

Quality Control:
All processes are to follow the quality standards of SEONGSAN LAB INC

Disclaimer

DSK Update Log:

Version	Change Notes
1.0	Release Version
	✓

Table of Contents

1 INTRODUCTION

This document explains how to use DSK to write software on NDT ultrasonic testing.
DSK's development environment is windows 10 based, using Visual Studio 2017 with .NET framework 5.0.

1.1 PREPARATION

DSK programs must be developed in its designated directory due to the various DLLs it requires to run. (e.g. c:/deepsound/dsktutorial/bin/release_x86)

Please note that Python must be installed inside CommonResource, which has to be two directories above the DSK program being written. (Check the User manual for reference)

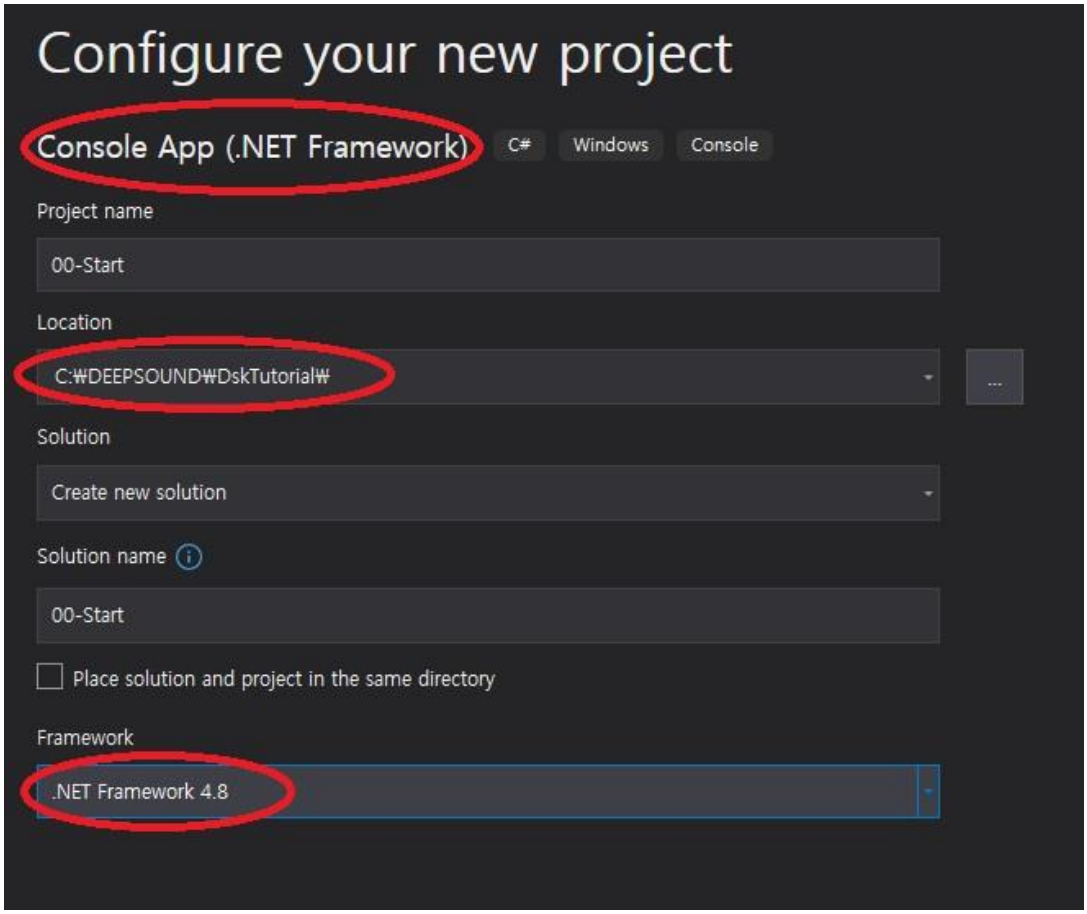
We recommend creating new projects in the dsktutorial directory.

1.2 GETTING STARTED

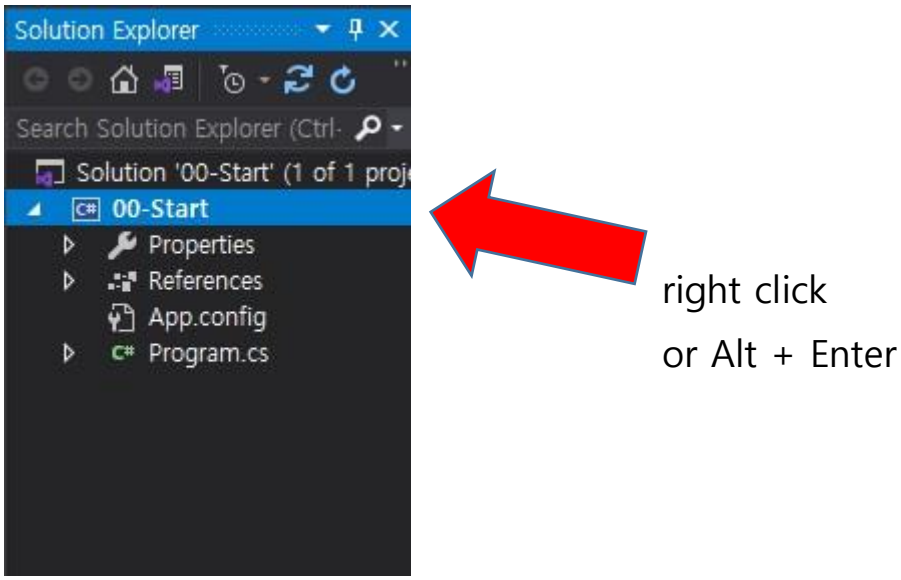
Open Visual Studio and create a New Solution/Project with the settings shown below.

After installing Tutorial, open DskTutorial.sln.

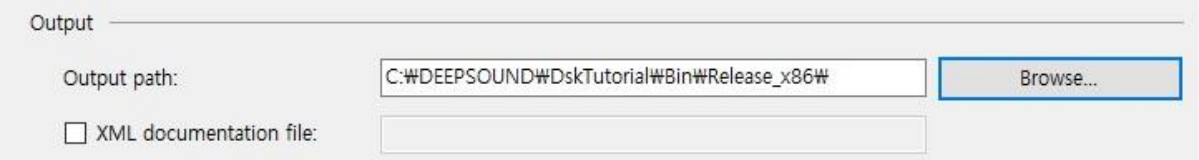
Select and create a new project (File->New->Project)by following the steps below.



After adding the “00-Start Project”, right click “00-Start” to open the project property window.

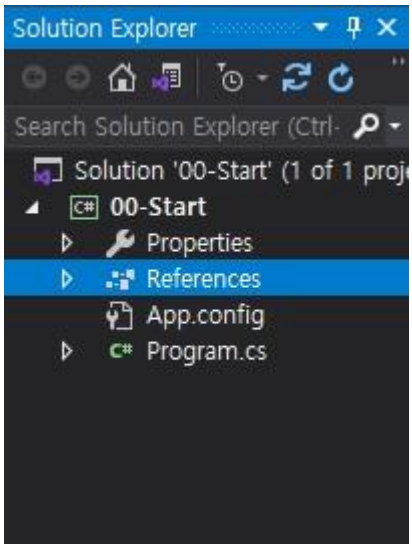


Select the “Build” tab and choose your output location/directory.



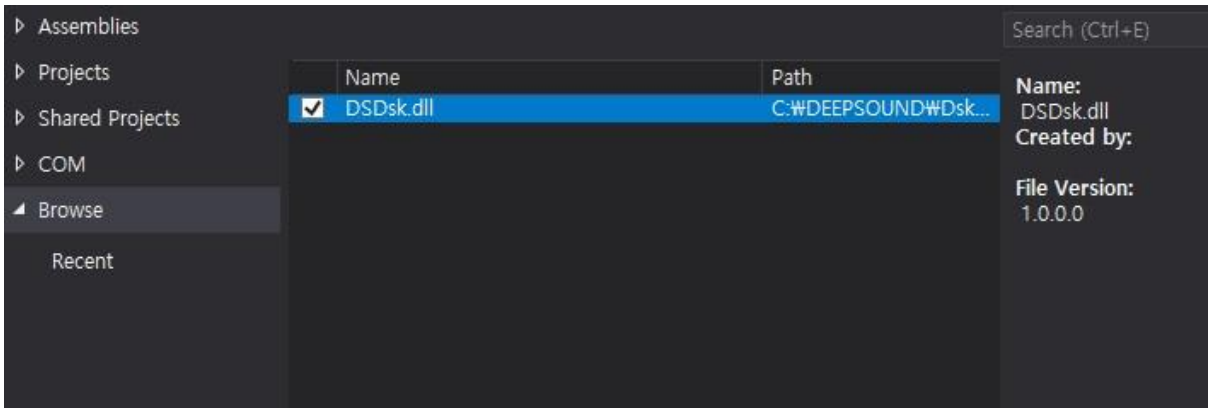
Adding References

DSK requires SVDSK.dll as a reference.

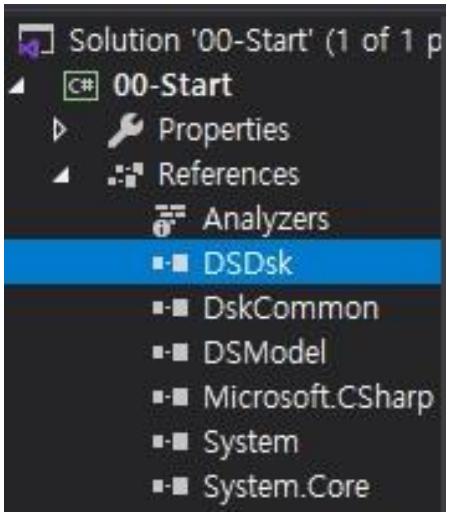


Right click “References” to select “Add reference…”

After opening the “Reference manager”, click on the “Browse” tab, select the same output directory “../Bin/Release_x86” set from the previous step, and choose DSDsk.dll.



It will be added to “Reference” as shown below.



Writing a Program

Open "Program.cs". Because DSK uses the DEEPSOUND namespace, you must always include "using DEEPSOUND;" in your code. Additionally, you must always call the **InitDsk method** before using DSK. The program below outputs the DSK version.

```
using System;
using DEEPSOUND;

namespace _00_Start
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello DSK!");

            var version = Dsk.GetVersion();
            Console.WriteLine($"DSK version={version}\n");
            Console.ReadKey();

        }
    }
}
```

Once you build and run 00-Start.exe you should see the following output on your console.

2 TUTORIALS

Section 2 is about the sample project included in the DSK.

The DSK sample project consists of the following contents.

Projects	**Explanation**
01 InitDevice	Initializes the hardware and demonstrates data transfer via DMA.
02 SScan	S Scan image 를 구현합니다.
03 AScan	A Scan image 를 구현합니다.
04 Inspection	Set up the encoder for inspection.
10 SimplePAUT	Use multiple components to implement a basic PAUT image.

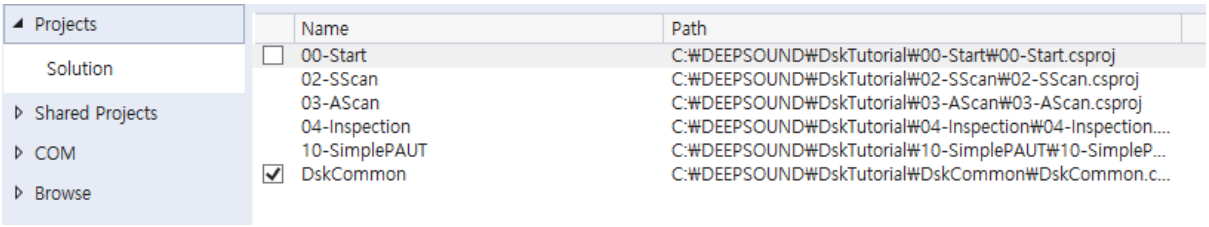
2.1 INITDEVICE

It is assumed that DSK is being run on NDT ultrasonic hardware.

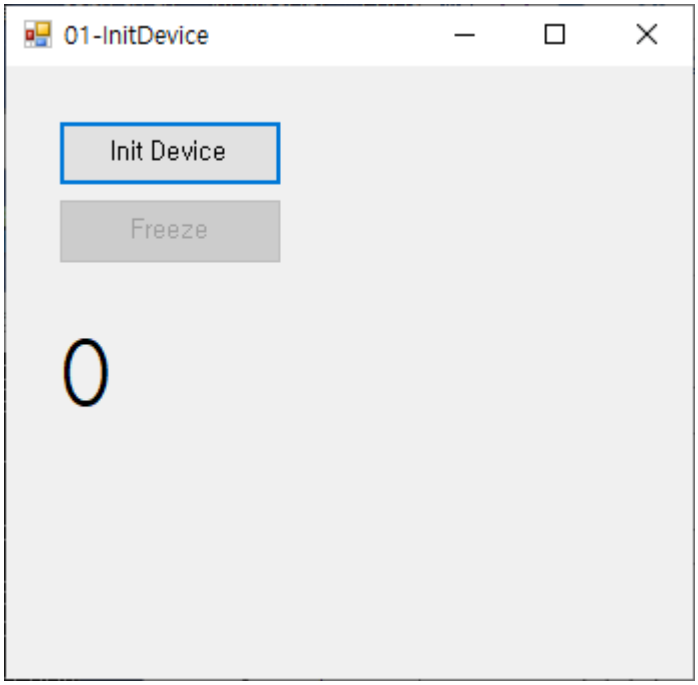
The InitDevice project is an example of receiving data through DMA after initializing the hardware.

Open the 01-InitDevice project. All DSK projects must include SVDSK.dll and the DskCommon project, which has the library that is used throughout all the examples as references.

Note that you can add DskCommon.dll manually just like SVDsk.dll, but adding it as a project works as well.



You will see the screen as shown below when you build and run the program.



The code for resetting is as follows. It is in the same format as the Start project.

```
public Form1()
{
    InitializeComponent();

    Dsk.InitDsk();

    Dsk.LogInfo($"DSK version => {Dsk.GetVersion()}");

    Load += Form1_Load;
    FormClosed += Form1_FormClosed;

}
```

The following code is for the “Init Device” button.

```
private void button1_Click(object sender, EventArgs e)
{
    Dsk.InitDevice();

    Dsk.SetCallbackFrame(DskCallback);

    button1.Enabled = false;
    button2.Enabled = true;
}
```

In the code above, InitDevice and InitDsk must be called together. When the InitDevice method is called, DSK resets the hardware, gets an image by using the **default setting**, and calls the “callback” function that is set in SetCallbackFrame.

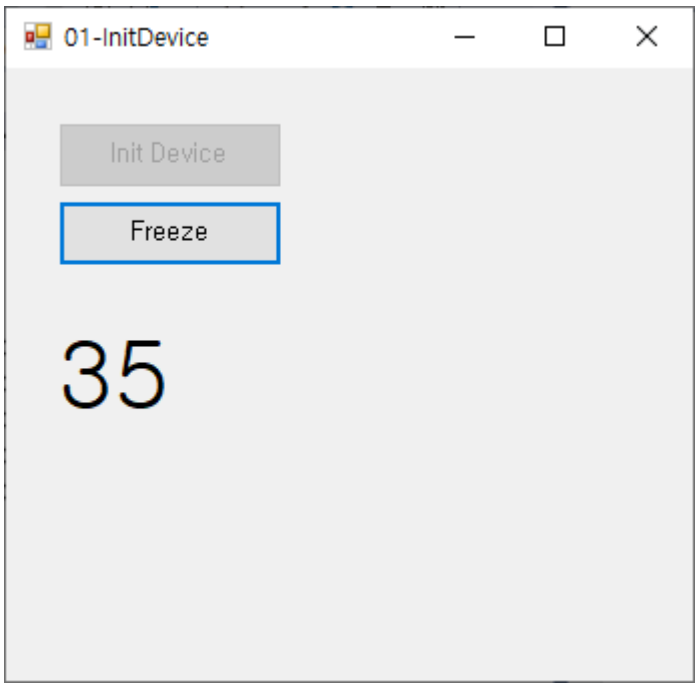
The ultrasonic data from the hardware is ultimately transferred to the Callback function via the USB DMA.

There are many types of Callback functions. Refer to the API reference for more information.

The Callback function in InitDevice is used to increment the counter and display it on the screen.

```
public void DskCallback(short[] frameData)
{
    ReceivedCount++;

    label1.InvokeIfNeeded(() =>
    {
        label1.Text = $"{ReceivedCount}";
    });
}
```

In addition, you can use Dsk.Freeze, Dsk.UnFreeze to stop or resume taking images. The screen below shows 00-InitDevice being run.



2.2 S SCAN

The SScan project is an example of taking the “rawdata” from the Callback function and translating it to a 2D image by using scan conversion.

While SScan uses “rawdata” for scan conversion, the dimensions of the 2D image are calculated automatically depending on the **settings in DSK.**

Therefore in order to run scan conversion with DSK, you must create the final image using SScan’s width and height from the DSK.

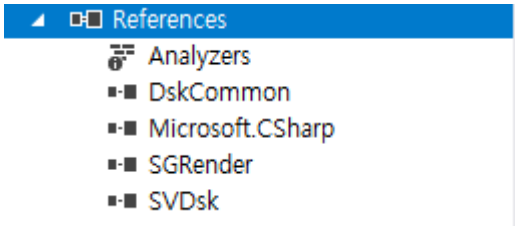
The following is the order in which SScan performs scan conversion.

- Get width and height from the functions GetSScanWidth and GetSScanHeight, respectively
- Use the GetVectorCount and GetSampleCountPerVector functions to collect information from “rawdata” being transferred into the Callback function.
- Use CalcSScanImage to get the 2D image created by “rawdata” through scan conversion
- Display the image on screen after completing color mapping

Reference Settings

The SScan project requires both SVDSK.dll and SGRender.dll to be added to references. Additionally, the SScan project is displayed by using openGL. But if you choose to use a different method or another openGL library, you do not have to include SGRender.dll in references.

Furthermore, in order to use color mapping, you must add the DskCommon project in references, which is included in the tutorial solution.



The Callback function first checks if it has all the transferred data, copies it to the buffer, then performs scan conversion through timer interrupt. Although this example simply copies the “rawdata” to the buffer, you must store the data in the queue in order to guarantee retention of all the data.

Callback Function

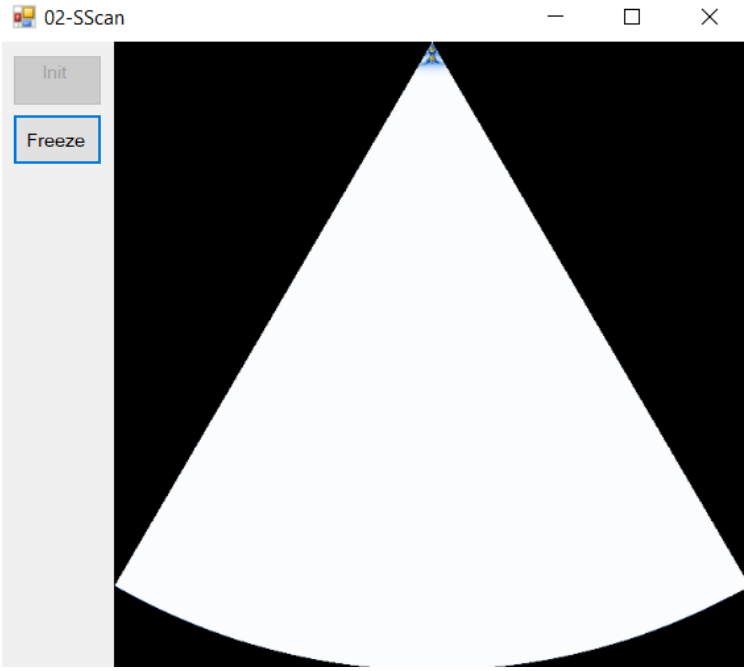
```
public void DskCallback(short[] frameData)
{
    ReceivedCount++;

    if (frameData.Length != VectorCount * SampleCountPerVector)
        return;

    Array.Copy(frameData, RawDataBuffer, VectorCount * SampleCountPerVector);
}
```

Scan Conversion Function

```
void DrawSScan(short[] rawdata)
{
    if (SScanImageWidth <= 0 || SScanImageHeight <= 0)
        return;

    if (rawdata == null)
        return;

    var _scimage = Dsk.CalcSScanImage(rawdata);
    if (_scimage == null)
    {
        Dsk.LogInfo("DrawSScan::SScanDisplayBuffer is null");
        return;
    }

    var SScanDisplayBuffer = ColorMap.SScanColorMapping(SScanImageWidth, SScanImageHeight, _scimage);

    GLGraphic graphicSscan = GLContextHelper.Instance.GetGraphic(panelSScan);
    var bounds = new Rectangle(0, 0, panelSScan.Width, panelSScan.Height);

    SScanTexture.Load(SScanDisplayBuffer, SScanImageWidth, SScanImageHeight, GLTextureType.RGB);
    graphicSscan.DrawTexture(SScanTexture, 0, 0, bounds.Width, bounds.Height);
    SScanTexture.Dispose();
    graphicSscan.ReleaseGraphic();
}
```

The code above shows SGRender.dll being used after getting the value of SScanDisplayBuffer. As mentioned before, you can use different ways to display this on the screen.

The following image shows the SScan project being run.



2.3 A SCAN

The AScan project is an example of creating an AScan image.

While SScan copies rawdata from the Callback function, AScan generates an image from timer interrupt by copying the AScan data from the AScan vector.

```
public void DskCallback(short[] frameData)
{
    ReceivedCount++;

    if (frameData.Length != VectorCount * SampleCountPerVector)
        return;

    Array.Copy(frameData, RawDataBuffer, VectorCount * SampleCountPerVector);
    Array.Copy(frameData, AScanVectorIndex * SampleCountPerVector, AScanBuffer, 0, SampleCountPerVector);
}

protected override void OnPaint(PaintEventArgs e)
{
    if (!IsInitDevice)
    {
        return;
    }
    DrawSScan(RawDataBuffer);
    DrawAScan(AScanBuffer);
}
```

The DrawAscan function displays the AScan image generated by bitmap instead of openGL.

```
void DrawAScan(short[] data)
{
    var width = AScanImage.Width;
    var height = AScanImage.Height;

    bool rotate = false;
    bool rfwave = false;

    var bmp = DSBitmap.MakeAScanBitmap(data, SampleCountPerVector, width, height, rotate, rfwave);
    AScanImage.Image = bmp;
}
```

2.4 INSPECTION

NDT ultrasonic testing uses the encoder to locate positions or generate an image from a set position.
The Inspection project is an example of using the encoder.

```
void SetParameters()
{
    SScanImageWidth = Dsk.GetSScanWidth();
    SScanImageHeight = Dsk.GetSScanHeight();

    VectorCount = Dsk.GetVectorCount();
    SampleCountPerVector = Dsk.GetSampleCountPerVector();

    // set encoder
    double resolution = 100;   // steps / mm
    Dsk.SetScanEncoder(0, 100, 1, resolution, false);
}
```

The code shown above is an example of using the SetScanEncoder function to initialize the scan encoder. The order of the SetScanEncoder parameters are as follows.

- Start position (mm)
- Stop position (mm)
- Step (mm)
- Encoder pulse count per mm
- Reverse flag

The code above gets an image from 0mm to 100mm by 1mm increments, by using an encoder that generates 100 pulse counts per mm.
The SetScanEncoder function initializes the encoder. To actually use the encoder, inspection mode must be set up.

- UploadInspectionModeStart : Starts inspection mode. **The Callback function calls the image when the encoder is at the predetermined position.**
- UploadInspectionModeStop : **Generates an image** and calls the Callback function regardless of the encoder's position.

2.5 SIMPLE PAUT

This project is an example of a simple PAUT equipment implemented by DSK functions.
DSK functions are largely divided into three categories: Get, Set, and Upload. Please refer to the API reference manual for more information. Simply put, Get is for obtaining DSK values, Set is for setting DSK values. Set only sets the value and is not implemented directly in the image. To actually implement it into the image, an Upload function must be called. There is UploadGain which only uploads certain parameters, and UploadAllParameters which uploads all parameters including all variables set by the Set function.

- GetXX
- SetXXX
- UploadXXX
- UploadAllParameters

In the Simple PAUT program, the UploadParameter() function is called when a user event occurs. The parameters are passed on as follows.

```
private bool UploadParameter()
{
    lock (SyncParameter)
    {
        Gui2Param();
        Gui2Gate();
        CgParam.Param2Dsk();

        Dsk.UploadAllParameters();

        CgParam.Dsk2Param();
        Param2Gui();

        ///Update Rule
        UpdateRulers();
    }

    return true;
}
```

The value for Gui2Param is set by the variable within the **GUI component.** After setting the value with DSK's Set function from Param2Dsk, it is implemented in the image by calling UploadAllParameters.
When **Upload** is called, DSK makes changes to certain values. These values are updated with DSK's Get function within the Dsk2Param function. Finally, Param2Gui updates the screen and the user event is fully processed.

B/C Scan

The Simple PAUT program utilizes the encoder settings used in the Inspection project to process B scans and C scans.

Use the ProcessBScanImage and ProcessCScanImage to generate their respective images. You can find an example of generating a CScan image in the DSKCommon project.

Load/Save Setup

While DSK only defines its functions' API, the user is free to load/save the parameters however they want to. The Simple PAUT program has an example of writing in and reading a file using the SimpleCGParam class defined in DskCommon.

The following shows how a Load is performed.

```
CgParam = Dsk.JsonFile2Object<SimpleCGParam>(fname);
CgParam.Param2Dsk();
Dsk.UploadAllParameters();

// update gui
CgParam.Dsk2Param();
Param2Gui();
Param2GuiGate();

UpdateRulers();
```

The DSK function, JsonFile2Object, uses json for serialization, so the user can use their own format. It is in the order of reading from the file, performing Set and Upload in DSK, and updating the GUI.