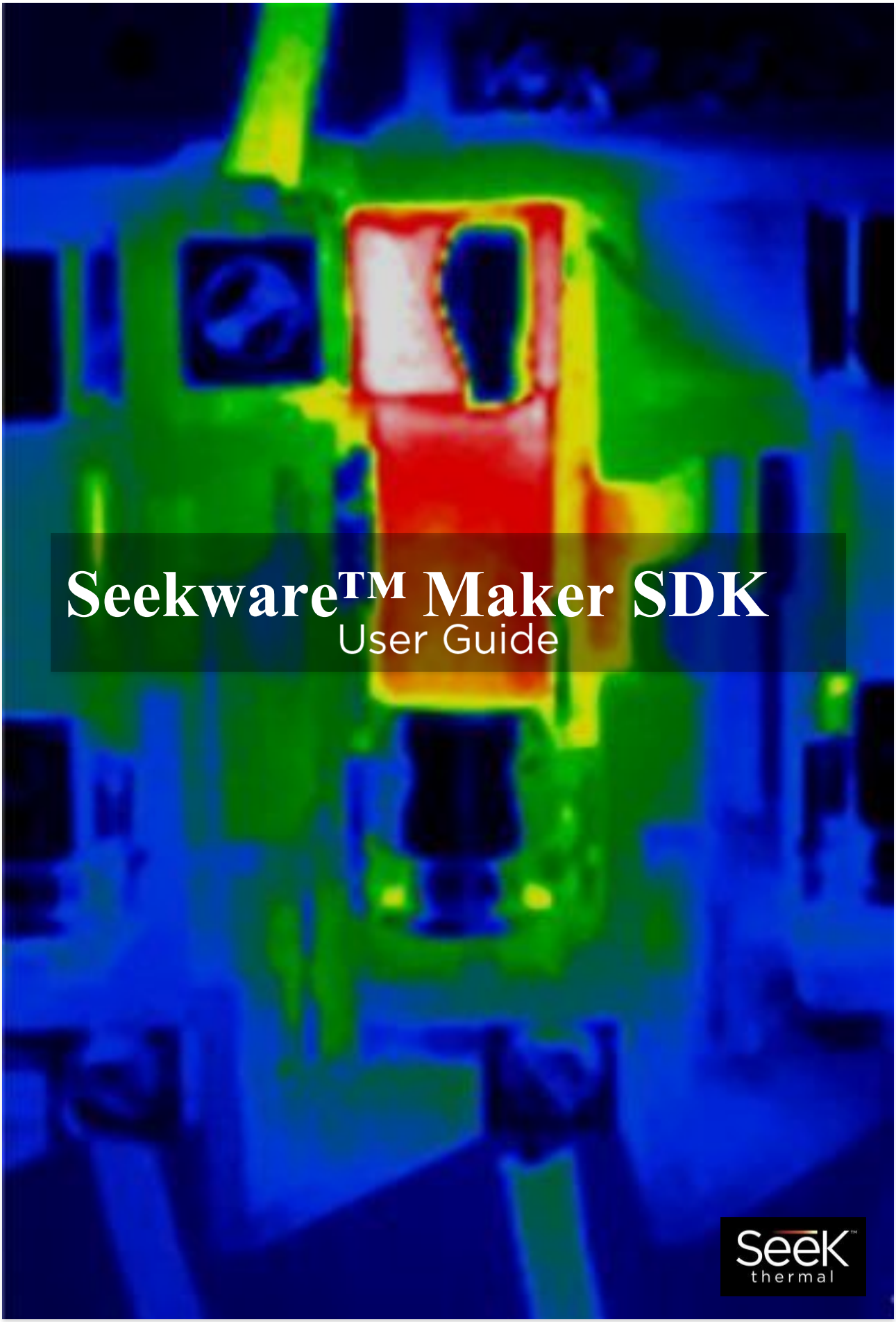

Seekware™ Maker SDK
User Guide
Seek thermal

Seekware™ Maker SDK

Seek Thermal, Inc.

111 Castilian Drive | Santa Barbara, CA 93117 | thermal.com

Table of Contents

WELCOME 4

ABOUT THE SEEKWARE SDK 5
- SUPPORTES CAMERAS 5
- THE SEEKWARE™ MAKER API 6
- THE SEEKWARE™ MAKER LIBRARIES 7

THE SEEKWARE API 7
- ENUMERATIONS 8
- USER DEFINED TYPES 9

FUNCTIONS 10
- SEEKWARE_FIND 11
- SEEKWARE_OPEN 11
- SEEKWARE_CLOSE 12
- SEEKWARE_GETSDKINFO 12
- SEEKWARE_GETIMAGE 13
- SEEKWARE_GETSETTING 13
- SEEKWARE_SETSETTING 14
- ANALYSIS_GETPALETTE 14
- ANALYSIS_GETTHERMOGRAPHYMINMAX 15

THE SEEKWARE™ MAKER LIBRARIES 16
- WINDOWS 16

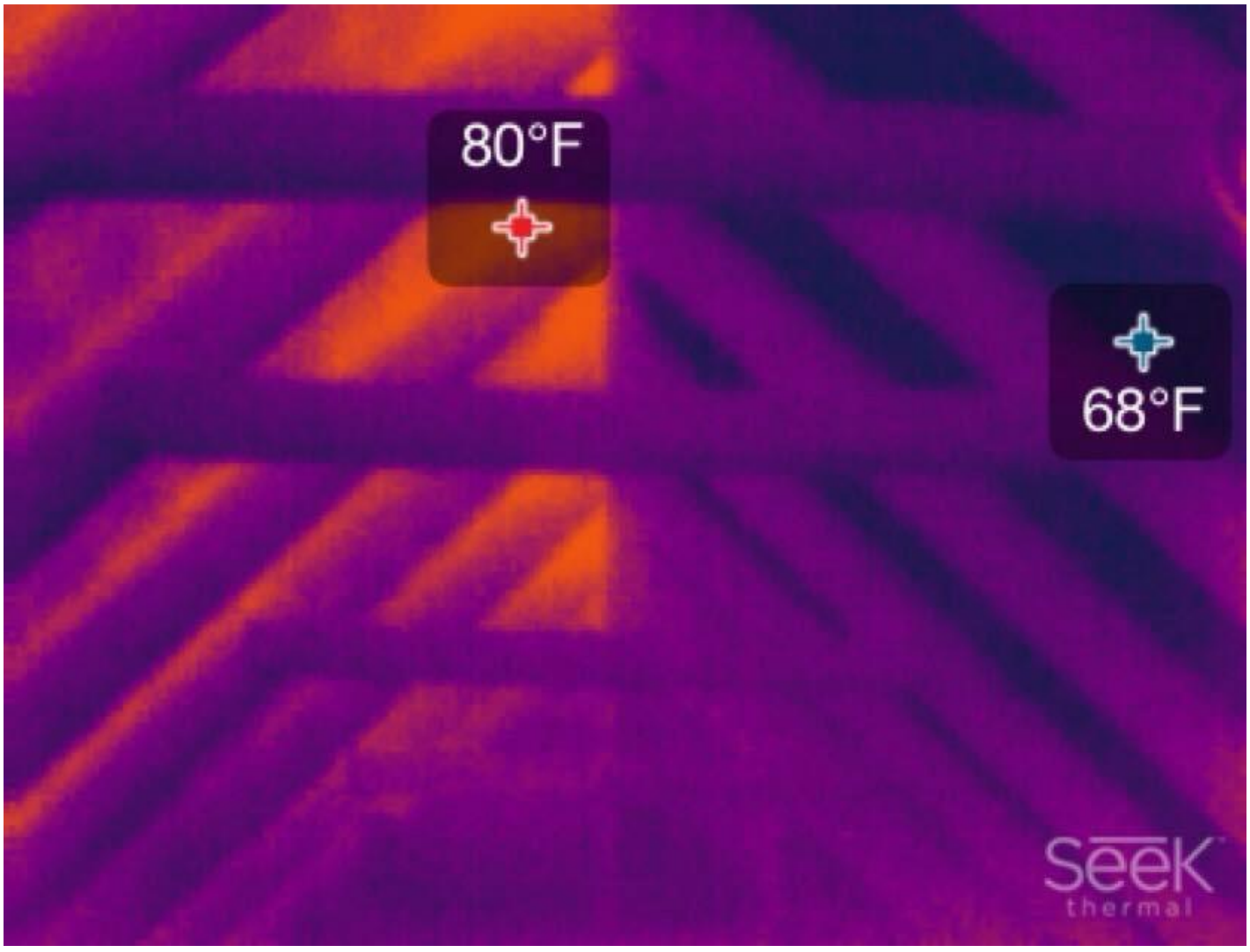
SAMPLE APPLICATIONS 17
- WINDOWS SAMPLE APP 17

SUPPORT 20

ATTRIBUTIONS 21

Welcome

The maker movement has an artisan spirit in which the methods of digital fabrication—previously held in the exclusive domain of institutions—have become accessible on a personal scale. Following a logical and economic progression similar to the transition from minicomputers to personal computers, a full-fledged industry based on the growing number of do-it-yourselfers who want to build something rather than buy it has arrived. Welcome to Seekware.



Compact Series

About the Seekware SDK

The Seekware™ Maker SDK was created for developers who want to use Seek Thermal cameras in their own projects. The SDK is designed to be simple to use while also providing access to key capabilities of the camera. We offer the Seekware™ Maker SDK for multiple operating systems and the Maker API is the same for them all.

Supported Cameras

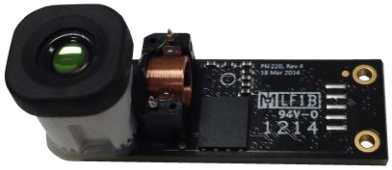
The Seekware™ Maker SDK has been designed to support various models of Seek Thermal cameras. Currently supported are the Compact, the CompactXR, the Compact Pro, and J1 , J2 and J3 platform cores. The software recognizes the type of camera and operates accordingly.

Supported Cameras

Camera		Image	Speed	SDK OS
Compact	PIR-206	206 x 156	<9Hz	Windows
CompactXR	PIR-206	206 x 156	<9Hz	Windows
Compact Pro	PIR-320	320 x 240	<9Hz	Windows
J1 Platform Core	PIR-206	206 x 156	<9Hz	Windows
J2 Platform Core	PIR-206	206 x 156	<9Hz	Windows
J3 Platform Core	PIR-320	320 x 240	<9Hz	Windows

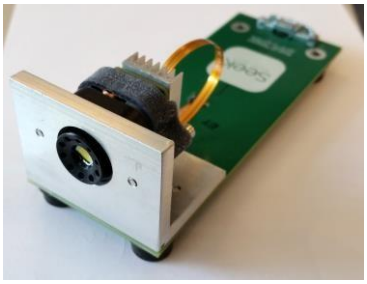
NOTE: Special requests can be made for Starter Kits that run < 18Hz.

CompactPro



J1 Platform Core

J2 or J3 Platform Core



J3 Maker Kit

The SDK consists of the following elements:

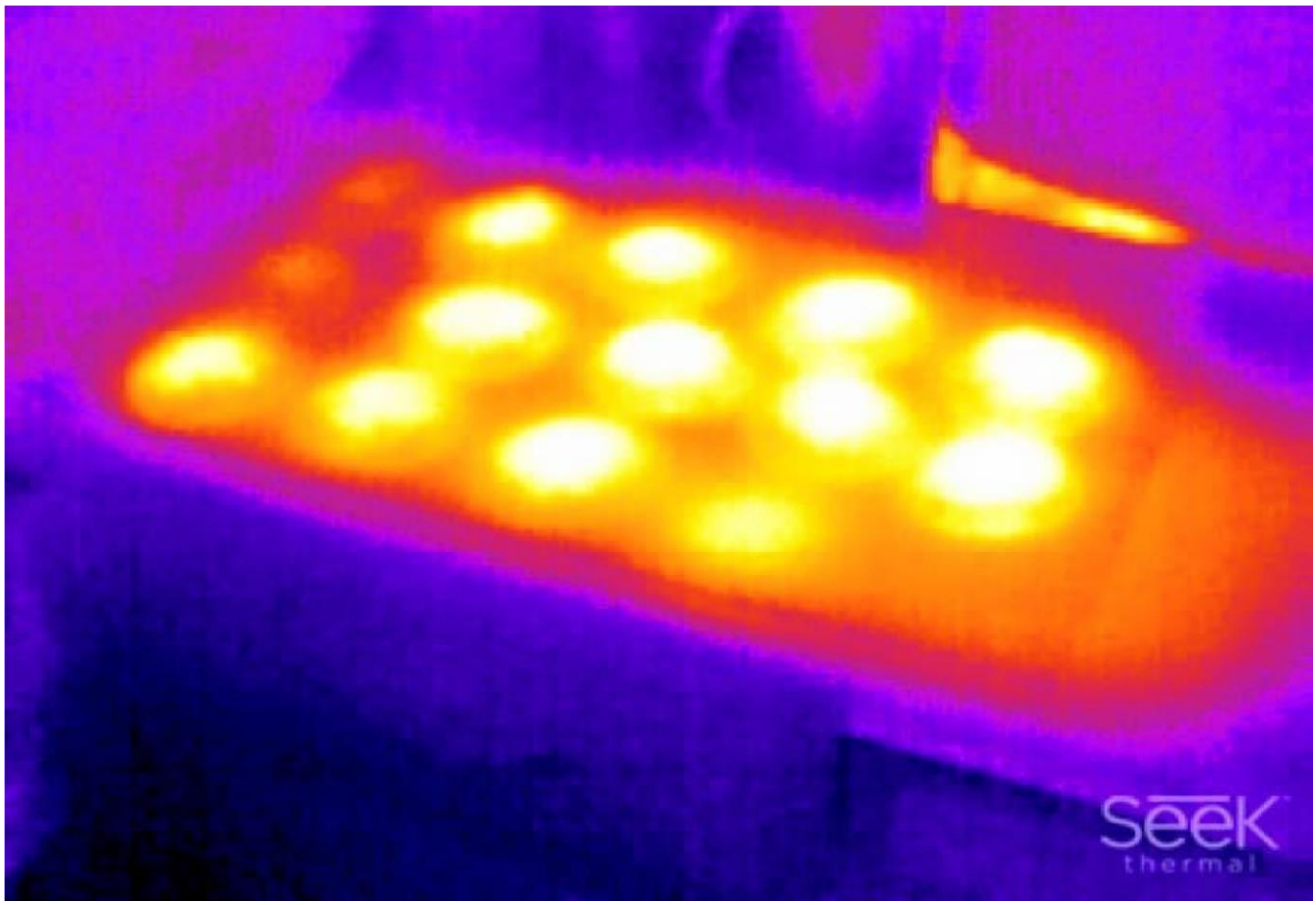
The Seekware™ Maker API

This is the definition of all the data structures and callable routines that are available to the maker. The API is based on the C programming language and specifies how software components should interact.

The Seekware™ Maker Libraries

The Seekware™ Maker Libraries currently supports the Windows Desktop platform. They are designed to be utilized during the build process to produce a custom application with thermal camera capabilities.

The Seekware API



Enumerations and User Defined Types

Enumerations

For this API, constant values are given as enumerations rather than simple constants in order to help avoid invalid input values and to clearly define all valid output values. In the C programming language, enumerations are declared with the keyword 'enum'.

Settings

This enumeration is used as the second parameter to the **Seekware_GetSetting** and **Seekware_SetSetting** functions to select the desired setting to read or modify.

```
typedef enum sw_settings {
      SETTING_ACTIVE_LUT,              // Selects the active LUT
      SETTING_TEMP_UNITS,              // Selects reported temp units
      SETTING_TIMEOUT                  //
} sw_settings;
```

Display LUTs

When calling the **Seekware_GetSetting** and **Seekware_SetSetting** functions with SETTING_ACTIVE_LUT as the second parameter, this enumeration is used as the third parameter to identify the current or desired LUT.

```
typedef enum sw_display_lut {
      SW_LUT_WHITE,                    // White Hot
      SW_LUT_BLACK,                    // Black Hot
      SW_LUT_IRON,                     // Classic hot iron
      SW_LUT_COOL,                     // Blue oriented
      SW_LUT_AMBER,                    //
      SW_LUT_INDIGO,                   //
      SW_LUT_TYRIAN,                   //
      SW_LUT_GLORY,                    // Red, white and blue
      SW_LUT_ENVY,                     // Green
      SW_LUT_NEWWHITE,                 // White Hot
      SW_LUT_NEWBLACK,                 // Black Hot
      SW_LUT_NEWIRON,                  // Classic hot iron
      SW_LUT_HI,                       //
```

```
      SW_LUT_HILO,                    //
      SW_LUT_NEWAMBER,                //
      SW_LUT_TYRIAN,                  //
      SW_LUT_SPECTRA,                 // Rainbow
      SW_LUT_SPECTRA2                 // High Contrast Rainbow
} sw_display_lut;
```

Temperature Units

When calling the **Seekware_GetSetting** and **Seekware_SetSetting** functions with **SETTING_TEMP_UNITS** as the second parameter, this enum is used as the third parameter to identify the current or desired temperature units.

```
typedef enum sw_temp_units {
      SW_TEMP_F,                      // Fahrenheit
      SW_TEMP_C,                      // Celsius
      SW_TEMP_K                       // Kelvin
} sw_temp_units;
```

Return Code

This enum contains all the function return values. Each function will have a particular set of valid return values.

```
typedef enum sw_retcode {
      SW_RETCODE_NONE,                // No error has been detected
      SW_RETCODE_OPENEX,              // Device is already opened
      SW_RETCODE_BPARAM               // Bad parameter
} sw_retcode;
```

User Defined Types

Device structure

This structure contains camera specific information to describe attached devices. It also contains OS specific information that is used to manage devices and their use.

```
typedef struct sw {                          // Device information
      unsigned char model[16];
      unsigned char serialNumber[12];
       uint8 fw_version_major;
       uint8 fw_version_minor;
       uint8 fw_build_major;
       uint8 fw_build_minor;
       uint16 frame_rows;
       uint16 frame_cols;

#ifdef defined _WIN32 || defined _WIN64

       // Windows Fields
       char * win_dev_path;
       FILE *
win_dev_handle;
#endif
     sw_retcode retcode;                     // Latest return code
       void *sdkPrivate;                     // SDK Private Data
} sw, *psw
```

SDK Information

This structure contains data necessary to uniquely identify the SDK and internal components.

```
typedef struct sw_sdk_info {
       uint8 sdk_version_major,              // SDK version number
       uint8 sdk_version_minor,
       uint8 sdk_build_major,
       uint8 sdk_build_minor,
       uint8 lib_version_major,              // Library version number
       uint8 lib_version_minor,
       uint8 lib_build_major,
       uint8 lib_build_minor
} sw_sdk_info;
```

Functions

All functions return an error code of the type **sw_retcode**.

```
typedef enum sw_retcode {
```

```
      SW_RETCODE_NONE,    // No error has been detected.
SW_RETCODE_OPENEX,  // Device is already opened exclusively
SW_RETCODE_BPARAM   // Bad parameter.
} sw_retcode;
```

Seekware_Find

```
sw_retcode Seekware_Find (
psw *pswlist[],   int
length,
      int *numfound
)
```

Description

Search the target environment for all connected devices. Then, starting at index zero-fill the **psarray** with a pointer to a device structure for each connected device up to **length**, then set **numfound** to the number of devices found. If there are more than **length** devices connected, fill the array, set **numfound** to length and return **SW_ERR_FIND_OFLOW**.

This function reads camera parameters and fills the **sw struct** members to allow selecting devices based on values in the **struct**.

If a camera is already open when this function is called, then the **sw_retcode** field will be set to **SW_RETCODE_OPENEX** and any fields which require that the device to be open to query shall be left in their default value.

Parameter(s)

pswlist[]	A pointer to an array of **psw** pointers allocated and supplied by the caller.
length	The length of the caller-supplied pointer array.
numfound	The number of devices found in the target environment.

Seekware_Open

```
sw_retcode Seekware_Open (
      psw id
)
```

Description

Open the device for use, populate internal elements in **`*sw`**, allocate any memory or events necessary for operation then return. Open devices are available exclusively to the instance of the SDK that opened them. A call to this function on a device that is already open should return **`SW_RETCODE_OPENEX`**.

Parameter(s) `id` A pointer to a Seekware device structure.

Seekware_Close

```
sw_retcode Seekware_Close (
      psw id
)
```

Description

Close the device, release any memory and terminate any events associated with the device.

Parameter(s)

`id` A pointer to a Seekware device structure.

Seekware_GetSdkInfo

```
sw_retcode Seekware_GetSdkInfo (
sw_sdk_info *info
)
```

Description

Returns a structure containing information about the SDK.

Parameter(s)

A pointer to a sw_sdk_info structure (see

info

Seekware_GetImage

```
sw_retcode Seekware_GetImage (
      psw id,
       uint16_t *binary,
float *temperature,
       uint32_t *display
)
```

Description

Grabs the next available image from the camera in one or more of the available formats. If any of the output formats is not necessary, the caller may supply a NULL pointer for that format and that parameter shall be ignored.

If a buffer is supplied, then it must be appropriately sized based on the data type and the number of pixels in the image. If a buffer is not supplied, then the computations and memory allocation required for that function shall not be performed.

Parameter(s)

id	A pointer to a Seekware device structure.
binary	A pointer to the destination image buffer.
temperature	A pointer to a buffer to hold temperature data.
display	A pointer to a buffer to hold color image data.

Seekware_GetSetting

```
sw_retcode Seekware_GetSetting (
      psw id,
      sw_settings index,
      int *value
)
```

Description

Gets the value of the specified setting.

Parameter(s)

id	A pointer to a Seekware device structure.
index	The setting index.
value	A pointer to the variable into which to place the value.

Seekware_SetSetting

```
sw_retcode Seekware_SetSetting (
      psw id,
      sw_settings index,
      int value
)
```

Description

Sets the value of the specified setting.

Parameter(s)

id	A pointer to a Seekware device structure.
index	The setting index (see Settings).
value	The setting value.

Analysis_GetPalette

```
sw_retcode Analysis_GetPalette (
      void *context,
      PALETTE_INDEX index,
      PALETTE *palette
)
```

Description

Gets the palette

Parameter(s)

context	A pointer to a Seekware device structure
index	The palette index
palette	A pointer to the location of the palette

Analysis_GetThermographyMinMax

```
sw_retcode Analysis_GetThermographyMinMax (
      void *context,
      float *thermographyData,
      float *min,
      float *max
)
```

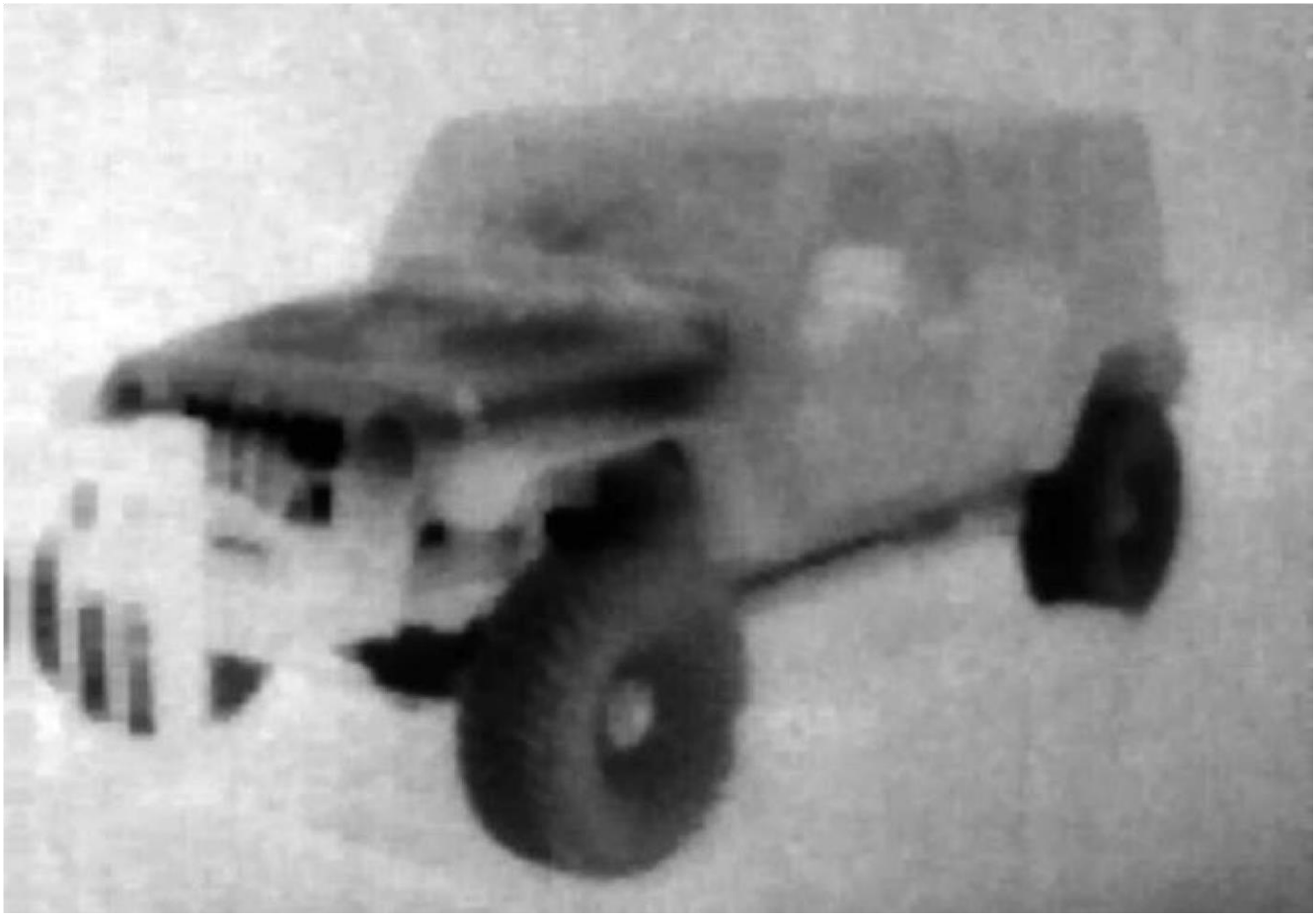
Description

Gets the min and max thermography data

Parameter(s)

context	A pointer to a Seekware device structure
thermographyData	A pointer to a buffer to hold thermography data
min	A pointer to the location of the min value
max	A pointer to the location of the max value

The Seekware Library



Desktop Platform Applications

Windows

Versions:

Windows 7+ (32 & 64-bit), Visual Studio® 2013+

Files:

The files included for this library are:

Seekware.dll - This is the Windows DLL that enables communication with the Seek Thermal camera.

Sample Applications

Windows Sample App

The windows sample app included with the SeekWare SDK is distributed as a Visual Studio project. The DLL libraries are included separately for x86 and x64 configurations.

User Setup:

Driver installation:

NOTE: Perform the following steps before connecting a Seek Thermal camera.

1) Unpack the provided zip file and navigate to the *Driver_version#* folder.

2) Right click on the *SeekWare.inf* file and select install.

3) Answer the security prompt and if driver installation is successful, Windows will return an *Operation Completed Successfully* dialogue box.

Begin Imaging:

1) Open the "SampleViewer.sln" file under the SampleApp directory in Visual Studio.

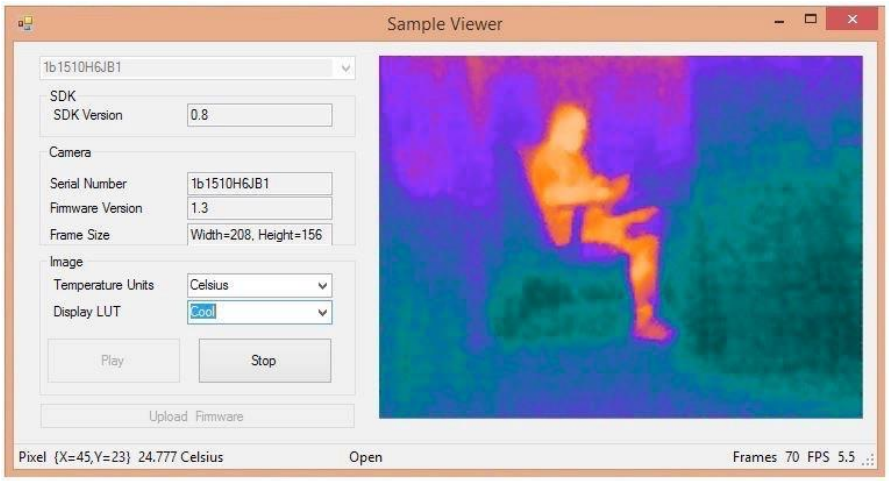
2) On the upper control bar select "release" and your system architecture (x86 or x64).

3) Click Start and the SampleViewer app will launch.

App Features:

Within the GUI, the *Camera Selection* drop-down menu shows the serial number(s) of the cameras currently plugged in to the USB ports. Drop down the selector and click on the serial number of the desired camera to begin. Click the play button to start live imaging.

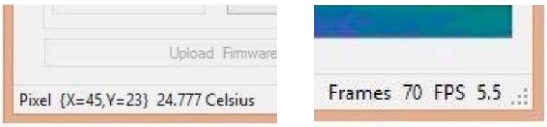
NOTE: If the camera is not available in the list or fails to image when the *Play* button is selected, the driver is probably not installed correctly. See known issues section on pg.19 for resolution.



Controlling Display Settings:

In the *Image Section* of the display (bottom left), the user can change the temperature units between *Fahrenheit*, *Celsius* and *Kelvin*. The *Display LUT* can be changed to display the image with several preloaded Display LUTs

Reading Temperature and Framerate:



Using the cursor the user can monitor the output temperature of a specific pixel, which is shown in the lower left of the window. The *Frame* count since clicking play, and frames per second (*FPS*) of the display output is shown on the bottom right.

Known Issues:

1) Driver is incorrectly installed, camera not recognized

   Resolution:

   If Seek Thermal camera is not showing up in Device Manager as: 

   (or PIR-206) then the device driver supplied with the SDK should be deleted and reinstalled. To reinstall the driver:

   1) Right click on the camera in device manger, navigate to the Driver tab and select uninstall.

   2) Check the box that says *"Delete the driver software for this device"*, click ok.

   3) Unplug the Seek Thermal camera and repeat the steps for driver installation above.

SUPPORT

While there is not currently a formal support program, there is an SDK Support team that will continue to make revisions and enhancements. We are interested in your experiences, whether good or bad. We strive to make the Maker experience a good one.

Please refer to www.thermal.com/SDK-Support for the Frequently Asked Questions (FAQ), Release Notes and email addresses for access to the SDK Support team.

ATTRIBUTIONS

1. **Windows** and **Visual Studio** are registered trademarks of Microsoft Corporation in the United States and/or other countries.
2. **ARM** is a registered trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. **ARM6** (ARMv6) and **ARM7** (ARMv7) are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
3. **Eclipse** is a trademark of Eclipse Foundation, Inc.
4. **Git** is a trademark of the Git Project. The Git Project is a member project of Software Freedom Conservancy ("Conservancy"). Conservancy holds rights in the Marks on behalf of the Git Project in accordance with its non-profit charitable mission.
5. **BeagleBone** is a trademark licensed under a Creative Commons Attribution-Share Alike 3.0 license.
6. **Compact**, **CompactXR**, and **Compact Pro** are trademarks of Seek Thermal, Inc.