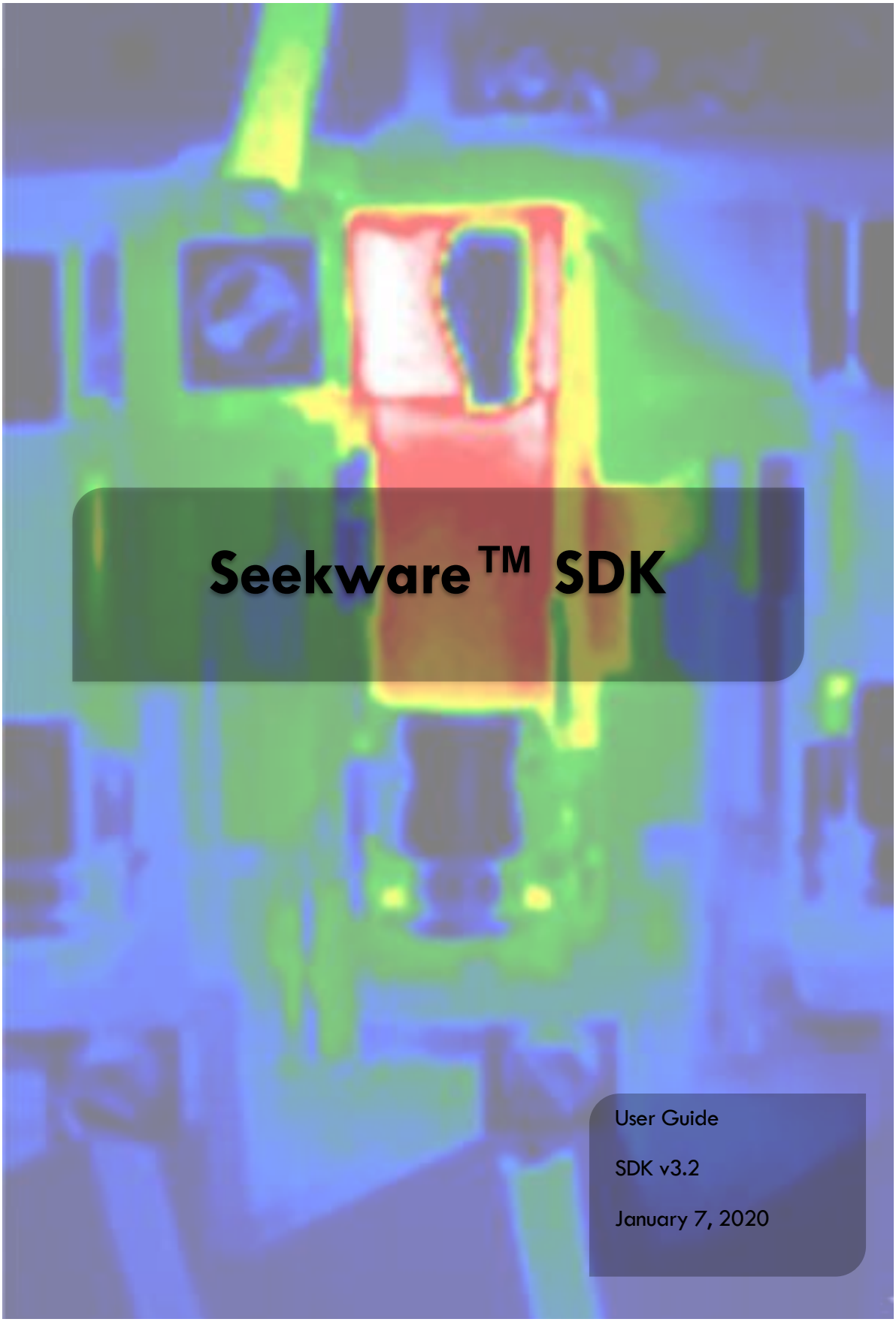


Seekware™ SDK

User Guide

SDK v3.2

January 7, 2020

Seekware™ SDK

Table of Contents

Welcome

The Seekware™ 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™ SDK for multiple platforms with a common API.

Supported Cameras

Compact

CompactPRO

J2/J3/C2x/C3x
Platform Core

J2/J3/SK2x/SK3x
Starter Kit

Camera		Image	Speed
Compact	PIR-206	206 x 156	<9Hz
CompactXR	PIR-206	206 x 156	<9Hz
CompactPRO	PIR-320	320 x 240	<9Hz
J2/C2x/SK2x	PIR-206	206 x 156	<9Hz
J3/C3x/SK3x	PIR-320	320 x 240	<9Hz

NOTE: Starter Kits that run at higher frame rates are available on special request.

Supported Platforms

Linux

	glibc	uclibc	musl libc
x86_64	x86_64-none-linux-gnu		
i686	i686-none-linux-gnu		
armv7 • **ABI: softfp eabi** • **FPU: neon**	arm-none-linux-gnueabi arm-linaro-linux-gnueabi	arm-none-linux-uclibcgnueabi	arm-none-linux-muslgnueabi
armv7 • **ABI: hard eabi** • **FPU: neon**	arm-none-linux-gnueabihf arm-linaro-linux-gnueabihf	arm-none-linux-uclibcgnueabihf	arm-none-linux-muslgnueabihf
armv8 • **ABI: aarch64 eabi** • **FPU: neon**	aarch64-none-linux-gnu		aarch64-none-linux-muslgnu

Windows

	Windows 10	Windows 8	Windows 7
x86	x86_windows		
x64	x64_windows		

Dependencies

Below is a list of all dependencies of the Seekware SDK library. All GNU libraries are reported as GCC version symbols unless otherwise noted.

x86

	x86_64-none-linux-gnu	i686-none-linux-gnu	x86-windows	xX64-windows
libc	GLIBC_2.3.2+	GLIBC_2.3.2+	-	-
libm	GLIBC_2.2.5+	GLIBC_2.1+	-	-
libstdc++	GLIBCXX_3.4.19+	GLIBCXX_3.4.19+	-	-
libgcc	-	GLIBC_2.0+	-	-
kernel32	-	-	Windows 7+	Windows7+
MSVCR120	-	-	Visual C++ 2013	Visual C++ 2013
MSVCP120	-	-	Visual C++ 2013	Visual C++ 2013
libusb	1.0.22+	1.0.22+	1.0.22+	1.0.22+

ARMV8

	aarch64-none-linux-gnu	aarch64-none-linux-muslgnu
libc	GLIBC_2.17+	musl 1.1.16+
libm	GLIBC_2.17+	-
libstdc++	GLIBCXX_3.4.19+	GLIBCXX_3.4.19+
libgcc	-	-
libusb	1.0.22+	1.0.22+

ARMV7 (HARD FLOAT ABI)

	arm-none-linux-gnueabihf	arm-linaro-linux-gnueabihf	arm-none-linux-uclibcgnueabihf	arm-none-linux-muslgnueabihf
libc	GLIBC_2.4+	GLIBC_2.4+	uClibc 0.9.33.2+	musl 1.1.16+
libm	GLIBC_2.4+	GLIBC_2.4+	uClibc 0.9.33.2+	-
libstdc++	GLIBCXX_3.4.19+	GLIBCXX_3.4.11+	GLIBCXX_3.4.19+	GLIBCXX_3.4.19+
libgcc	GCC_3.5+	GCC_3.5+	GCC_3.5+	GCC_3.5+
libusb	1.0.22+	1.0.22+	1.0.22+	1.0.22+

ARMV7 (SOFT FLOAT ABI)

	arm-none-linux-gnueabi	arm-linaro-linux-gnueabi	arm-none-linux-uclibcgnueabi	arm-none-linux-muslgnueabi
libc	GLIBC_2.4+	GLIBC_2.4+	uClibc 0.9.33.2+	musl 1.1.16+
libm	GLIBC_2.4+	GLIBC_2.4+	uClibc 0.9.33.2+	-
libstdc++	GLIBCXX_3.4.19+	GLIBCXX_3.4.11+	GLIBCXX_3.4.19+	GLIBCXX_3.4.19+
libgcc	GCC_3.5+	GCC_3.5+	GCC_3.5+	GCC_3.5+
libusb	1.0.22+	1.0.22+	1.0.22+	1.0.22+

The Seekware™ API

This is the definition of all the data structures and callable routines that are available to the developer. The API is based on the C programming language.

Return Codes

All functions return with a value of the type `sw_retcode`.

Return Code	Meaning
SW_RETCODE_NONE	No error has been detected
SW_RETCODE_NOTOPENED	Device is not opened
SW_RETCODE_OPENEX	Device is already opened exclusively
SW_RETCODE_BPARAM	Bad parameter
SW_RETCODE_NOFRAME	Frame processing error
SW_RETCODE_ERROR	Generic error
SW_RETCODE_OFLOW	Buffer overflow
SW_RETCODE_USBERR	USB error; camera restart required
SW_RETCODE_SETONLY	Setting is write only
SW_RETCODE_GETONLY	Setting is read only
SW_RETCODE_NOTSUPPORTED	Setting is not supported by version of camera firmware
SW_RETCODE_INVALIDSETUP	Another setting needs to be configured differently
SW_RETCODE_DISCONNECTED	Device is disconnected from the host

Table 2 - API Return Codes

Error Recovery

After finding a camera and opening it for use, if a function returns a value other than `SW_RETCODE_NONE`, the software should call `Seekware_Close`, then `Seekware_Find`, then `Seekware_Open` to properly recover the connection to the camera. This SDK has been designed to gracefully recover from errors when this process is employed.

If the SDK returns `SW_RETCODE_INVALIDSETUP` when accessing `SETTING_TRIGGER_SHUTTER`, be sure to enable `SETTING_AUTOSHUTTER` first.

SW Structure Definition

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
    uint16_t model;
    char serialNumber[13];
    char modelNumber[17];
    char manufactureDate[33];
    uint8_t fw_version_major;
    uint8_t fw_version_minor;
    uint8_t fw_build_major;
    uint8_t fw_build_minor;
    uint16_t frame_rows;
    uint16_t frame_cols;
#if defined (__linux__) || defined  (__APPLE__)
    struct libusb_device_handle * lusb_dev_handle;
    enum libusb_transfer_status * lusb_status;
#elif defined (_WIN32) || !defined(_WIN64)
    char * win_dev_path;
    FILE * win_dev_handle;
#else
#error "Platform was not defined."
#endif

    /// Latest return code
    sw_retcode retcode;

    /// Private SDK Context
    void* seekware_context;

    uint16_t rawframe_rows;
    uint16_t rawframe_cols;

#if defined (__linux__) || defined  (__APPLE__)
    struct libusb_device * libusb_device;
#endif
} sw, * psw;
```

The `model` field is of the sw_model type given below:

```
typedef enum sw_model {
    SEEK_MODEL_206_WFOV = 0,
    SEEK_MODEL_206_WFOV_FF,
    SEEK_MODEL_206_NFOV,
    SEEK_MODEL_206_NFOV_FF,
    SEEK_MODEL_320_WFOV,
    SEEK_MODEL_320_WFOV_FF,
    SEEK_MODEL_320_NFOV,
    SEEK_MODEL_320_NFOV_FF
} sw_model;
```

The `serialNumber` field contains a null terminated string with the12-digit camera serial number.

The `modelNumber` field contains a null terminated string with camera model number.

The `fw_version/build` fields report the camera firmware version and build numbers.

The `frame_rows` and `frame_cols` fields report the image data rows and columns.

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_RETCODE_OFLOW`. The `sw` structure contains `frame_rows` and `frame_cols` fields which indicate the rows and columns of the attached camera.

Parameter(s)

`pswlist[]`	A pointer to an array of `psw` pointers allocated 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

Opens the device for use, allocates memory, and begins processing thermal data. Open devices are available exclusively to the instance of the SDK that opened them. Only a single device can be opened at once in the current process. Returns `SW_RETCODE_OPENEX` if already open.

Parameter(s)

`id`	A pointer to a Seekware device structure.

Seekware_Start

```
sw_retcode Seekware_Start (psw id)
```

Description

Starts background frame processing and wakes up the camera from low power mode (if it is sleeping). Calling Seekware_Start on a camera that is already started, has no effect.

Parameter(s)

`id`	A pointer to the Seekware device structure of an open device.

Seekware_Stop

```
sw_retcode Seekware_Stop(psw id)
```

Description

Stops background frame processing and puts the camera into low power mode (if supported). Calling Seekware_Stop on a camera that is already stopped has no effect.

Parameter(s)

`id`	A pointer to a Seekware device structure.

Seekware_Close

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

Description

Closes the device, releases memory, and puts the device into a low power state.

Parameter(s)

`id`	A pointer to a Seekware device structure.

Seekware_GetSdkInfo

```
sw_retcode Seekware_GetSdkInfo(psw id, sw_sdk_info *info)
```

Description

Returns a structure containing information about the SDK. This function must be called after a successful call to `Seekware_Open`.

Parameter(s)

`id`	A pointer to a Seekware device structure. You may pass NULL in order to access the SDK version before opening a device.
`info`	A pointer to a `sw_sdk_info` structure.

SDK Info Structure

The `sw_sdk_info` structure contains data necessary to uniquely identify the SDK and internal components.

```
typedef struct sw_sdk_info {
      uint8_t sdk_version_major,   // SDK version number
      uint8_t sdk_version_minor,
      uint8_t sdk_build_major,
      uint8_t sdk_build_minor,
      uint8_t lib_version_major,   // Image Processing version number
      uint8_t lib_version_minor,
      uint8_t lib_build_major,
      uint8_t lib_build_minor
} sw_sdk_info;
```

Seekware_UploadFirmware

```
sw_retcode Seekware_UploadFirmware(psw id, const char* filename)
```

Description

Loads firmware into Seekware device. For use with firmware files provided separately by Seek. Contact Seek for details.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`filename`	The firmware file to be uploaded to Seekware device.

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 (see Table 4 – GetSettingEx/SetSettingEx). Must be less than or equal to `SETTING_THERMOGRAPHY_VERSION`.
`value`	A pointer to the location to write the setting value.

Seekware_GetSettingEx

```
sw_retcode Seekware_GetSettingEx (
      psw id, sw_settings index, void *value, uint32_t bytes
)
```

Description

Writes the requested setting into value.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`index`	The setting index (see Table 4 – GetSettingEx/SetSettingEx).
`value`	A pointer to the storage location of value
`bytes`	The size of value in bytes.

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 Table 4 – GetSettingEx/SetSettingEx). Must be less than or equal to `SETTING_THERMOGRAPHY_VERSION`.
`value`	The setting value.

Seekware_SetSettingEx

```
sw_retcode Seekware_SetSettingEx(
      psw id, sw_settings setting, void *value, uint32_t bytes
)
```

Description

Sets the value(s) of the requested setting using the provided value(s).

Parameter(s)

`id`	A pointer to a Seekware device structure.
`setting`	The setting index (see Table 4 – GetSettingEx/SetSettingEx).
`value`	A pointer to the setting value(s).
`bytes`	The size of value in bytes.

SW_SETTINGS

Various SDK settings can be queried and changed by calling the `Seekware_GetSetting/Ex` and `Seekware_SetSetting/Ex` functions respectively. The settings are selected by the `index` parameter. The following tables provide descriptions for each setting in the `sw_settings` enum. Further explanation of all settings can be found after Table 4.

The settings listed in Table 3 can be accessed via any of the `Seekware` settings functions, while the settings listed in Table 4 can only be accessed via `Seekware_GetSettingEx` and `Seekware_SetSettingEx`.

Setting	Index	Set/Get	Description	Value Size in Bytes	Type
SETTING_ACTIVE_LUT	0	Set/Get	The active display LUT	4	int
SETTING_TEMP_UNITS	1	Set/Get	Temperature units	4	int
SETTING_TIMEOUT	2	Set/Get	Communications timeout	4	int
SETTING_CONTROL	3	Set/Get	Control settings	4	int
SETTING_EMISSIVITY	4	Set/Get	Emissivity	4	int
SETTING_BACKGROUND	5	Set/Get	Background temperature	4	int
SETTING_THERMOGRAPHY_VERSION	6	Get	Thermography version	4	int

Table 3 - GetSetting/Ex and SetSetting/Ex Settings

Setting	Index	Set/Get	Description	Value Size in Bytes	Type
SETTING_GLOBAL_THERM_ADJUST	10	Set	Global temperature offset	4	sw_global_ therm_adjust_t
SETTING_SCENE_THERM_ADJUST	11	Set	Temperature offset for a scene	8	sw_scene_ therm_adjust_t
SETTING_ENVIRONMENT_ THERM_ADJUST	12	Set	Temperature offset for an environment	8	sw_environment_ therm_adjust_t
SETTING_SPECIFIC_THERM_ADJUST	13	Set	Temperature offset for a scene and an environment	12	sw_specific_ therm_adjust_t
SETTING_TRANSIENT_ CORRECTION_ENABLE	14	Set/Get	Transient correction	4	uint32_t
SETTING_TRANSIENT_ CORRECTION_PARAMS	15	Set/Get	Amplitude and decay for transient correction	8	sw_transient_ adjust_t
SETTING_SMOOTHING	16	Set/Get	Image smoothing	4	uint32_t
SETTING_AUTOSHUTTER	17	Set/Get	Auto shutter	4	uint32_t
SETTING_MINMAX	18	Get	Min/Max with coordinates	24	sw_minmax_t
SETTING_SHARPENING	19	Set/Get	Image sharpening	4	uint32_t
SETTING_ENABLE_TIMESTAMP	20	Set/Get	Enables timestamp counter	4	uint32_t
SETTING_RESET_TIMESTAMP	21	Set	Resets timestamp counter	4	uint32_t
SETTING_TRIGGER_SHUTTER	22	Set	Triggers a camera shutter	4	uint32_t
SETTING_AGC_MODE	23	Set/Get	AGC Mode	4	uint32_t
SETTING_HISTEQ_BIN_COUNT	24	Get	Number of bins	4	uint32_t
SETTING_HISTEQ_INPUT_BIT_DEPTH	25	Get	Number of input bins before performing AGC calculation	4	uint32_t

SETTING_HISTEQ_OUTPUT_ BIT_DEPTH	26	Get	Number of output bins after performing AGC calculation	4	uint32_t
SETTING_HISTEQ_HIST_ WIDTH_COUNTS	27	Get	Width of active histogram	4	uint32_t
SETTING_HISTEQ_PLATEAU_VALUE	28	Set/Get	Max percentage of total pixels that can be assigned to a single bin	4	float
SETTING_HISTEQ_GAIN_LIMIT	29	Set/Get	Max percentage of output colors per sensor count	4	float
SETTING_HISTEQ_GAIN_LIMIT_ FACTOR_ENABLE	30	Set/Get	Enables *GainLimitFactor*	4	uint32_t
SETTING_HISTEQ_GAIN_LIMIT_ FACTOR	31	Get	*GainLimitFactor* value	4	float
SETTING_HISTEQ_GAIN_LIMIT_ FACTOR_XMAX	32	Set/Get	Max histogram width with which *GainLimitFactor* will affect the total gain	4	uint32_t
SETTING_HISTEQ_GAIN_LIMIT_ FACTOR_YMIN	33	Set/Get	Min *GainLimitFactor* value	4	float
SETTING_HISTEQ_ALPHA_TIME	34	Set/Get	Number of seconds to blend current frame and previous frame's histogram	4	float
SETTING_HISTEQ_TRIM_LEFT	35	Set/Get	Percentage of outliers to trim from left side of histogram	4	float
SETTING_HISTEQ_TRIM_RIGHT	36	Set/Get	Percentage of outliers to trim from right side of histogram	4	float
SETTING_LINMINMAX_MODE	37	Set/Get	Linear min/max mode	4	uint32_t
SETTING_LINMINMAX_MIN_LOCK	38	Set/Get	Lower bound for linear min/max	4	uint32_t
SETTING_LINMINMAX_MAX_LOCK	39	Set/Get	Upper bound for linear min/max	4	uint32_t

SETTING_LINMINMAX_ACTIVE_MIN_VALUE	40	Get	Min sensor count value in the last scene	4	uint32_t
SETTING_LINMINMAX_ACTIVE_MAX_VALUE	41	Get	Max sensor count value in the last scene	4	uint32_t
FEATURE_OEM	42	Set/Get	Generic passthrough function for access to OEM features Size in bytes and type dependent on setting. Please contact Seek for details.		

Table 4 - GetSettingEx/SetSettingEx Settings

SETTING_ACTIVE_LUT

For this setting, `Seekware_GetSetting` returns a value in the following table and `Seekware_SetSetting`, with a `value` parameter set to one of the following values, will change the LUT used to generate `display` imagery.

Setting Value	Description
SW_LUT_WHITE_NEW	White hot
SW_LUT_BLACK_NEW	Black hot
SW_LUT_SPECTRA	Rainbow
SW_LUT_PRISM	Modified rainbow
SW_LUT_TYRIAN_NEW	Purple
SW_LUT_AMBER_NEW	Yellow-orange
SW_LUT_IRON_NEW	Classic Hot Iron
SW_LUT_HI	High temp highlight
SW_LUT_HILO	High and low temp highlight

Table 5 - LUT Values

SETTING_TEMP_UNITS

For this setting, `Seekware_GetSetting` returns a value in the following table and `Seekware_SetSetting`, with an `index` parameter set to one of the following values, will change temperature units used for temperature data returned from `Seekware_GetImage` and `Seekware_GetImageEx`.

Setting Value	Temperature Units
SW_TEMP_F	Fahrenheit
SW_TEMP_C	Celsius
SW_TEMP_K	Kelvin

Table 6 – Temperature Unit Values

SETTING_TIMEOUT

The USB transaction timeout is determined by the value of this setting. The `value` field is an int containing the timeout period in milliseconds. Thus, a value of 3000 would cause the USB timeout to be 3 seconds. The default timeout is 500.

SETTING_CONTROL (deprecated)
(replaced with SETTING_SMOOTHING and SETTING_AUTOSHUTTER)

This setting allows control over certain camera binary settings as described in the following table. This setting is a bitmask of the values in the table. To set the setting, the software must bitwise OR the setting on the value sent. To get the setting, the software must mask the value returned to see if the bit is set or not.

Setting Value	Bit Description
SEEKWARE_CTRL_SMOOTHING	Read/Write. Sets/indicates that image smoothing is enabled.
SEEKWARE_CTRL_AUTOSHUTTER	Write only. Mask off to disable auto-shutter.

Table 7 – Control Settings

SETTING_EMISSIVITY

This setting determines the assumed surface emissivity for temperature readings. Since the required value is a floating-point number but the setting must be an integer, this value is set with the intended value x 100. Thus, for a desired emissivity setting of 0.97, this function should be called with a value of 97. Reading returns a number in the same x 100 format. The default is 0.97.

SETTING_BACKGROUND

This setting determines the assumed background temperature for temperature readings. This value must be given and shall be reported in whole degrees Celsius regardless of the temperature units setting. The default background temperature is 25°C.

SETTING_THERMOGRAPHY_VERSION

This is a read-only setting that reports the version of the thermography code used by the calibration test station when the connected camera was calibrated.

Note: The following four settings are used to make thermography adjustments based on a device's scene and operating temperature. Before applying any of these settings, the device's thermography values are unadjusted, as shown in the thermography adjustment table below:

Operating (Environment) Temperature

Scene Temperature

BBActual	10	15	20	25	30	35	40	45	50
-15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0
150	0	0	0	0	0	0	0	0	0
175	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0	0	0
350	0	0	0	0	0	0	0	0	0
425	0	0	0	0	0	0	0	0	0

Table 8 - Unadjusted Thermography Table

SETTING_GLOBAL_THERM_ADJUST

This is a write-only setting that allows the user to set a global temperature offset. When applying a `SETTING_GLOBAL_THERM_ADJUST` of 5 degrees, all thermography values are adjusted by 5 degrees.

Operating (Environment) Temperature

Scene Temperature

BBActual	10	15	20	25	30	35	40	45	50
-15	5	5	5	5	5	5	5	5	5
0	5	5	5	5	5	5	5	5	5
15	5	5	5	5	5	5	5	5	5
30	5	5	5	5	5	5	5	5	5
45	5	5	5	5	5	5	5	5	5
60	5	5	5	5	5	5	5	5	5
80	5	5	5	5	5	5	5	5	5
100	5	5	5	5	5	5	5	5	5
125	5	5	5	5	5	5	5	5	5
150	5	5	5	5	5	5	5	5	5
175	5	5	5	5	5	5	5	5	5
200	5	5	5	5	5	5	5	5	5
250	5	5	5	5	5	5	5	5	5
300	5	5	5	5	5	5	5	5	5
350	5	5	5	5	5	5	5	5	5
425	5	5	5	5	5	5	5	5	5

Table 9 - Global Offset Thermography Table

SETTING_SCENE_THERM_ADJUST

This is a write-only setting that allows the user to set a temperature offset for a specific scene. When applying a `SETTING_SCENE_THERM_ADJUST`, the two calibrated scene temperatures bounded by the adjustment are offset at all camera operating temperatures.

Operating (Environment) Temperature

Scene Temperature

BBActual	10	15	20	25	30	35	40	45	50
-15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
15	6.09	6.52	7.11	7.4	7.2	7.18	6.92	5.53	5.21
30	2.46	2.85	2.96	3.03	2.6	3.17	2.22	3.19	2.11
45	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0
150	0	0	0	0	0	0	0	0	0
175	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0	0	0
350	0	0	0	0	0	0	0	0	0
425	0	0	0	0	0	0	0	0	0

Table 10 - Scene Offset Thermography Table

SETTING_ENVIRONMENT_THERM_ADJUST

This is a write-only setting that allows the user to set a temperature offset for a specific environment. When applying a `SETTING_ENVIRONMENT_THERM_ADJUST`, all calibrated scene temperatures are offset at the camera operating temperatures bounded by the adjustment.

Operating (Environment) Temperature

Scene Temperature

BBActual	10	15	20	25	30	35	40	45	50
-15	0	0	0	0	2.46	6.09	0	0	0
0	0	0	0	0	2.64	6.52	0	0	0
15	0	0	0	0	2.85	7.11	0	0	0
30	0	0	0	0	2.96	7.4	0	0	0
45	0	0	0	0	2.87	7.2	0	0	0
60	0	0	0	0	2.87	7.18	0	0	0
80	0	0	0	0	2.88	7.13	0	0	0
100	0	0	0	0	3.03	7.55	0	0	0
125	0	0	0	0	3.31	8.33	0	0	0
150	0	0	0	0	2.75	6.92	0	0	0
175	0	0	0	0	2.6	6.43	0	0	0
200	0	0	0	0	2.12	5.27	0	0	0
250	0	0	0	0	3.17	7.91	0	0	0
300	0	0	0	0	2.22	5.53	0	0	0
350	0	0	0	0	3.19	7.87	0	0	0
425	0	0	0	0	2.11	5.21	0	0	0

Table 11 - Environment Offset Thermography Table

SETTING_SPECIFIC_THERM_ADJUST

This is a write-only setting that allows the user to set a temperature offset for a specific scene and environment. When applying a `SETTING_SPECIFIC_THERM_ADJUST`, only calibrated temperature bounded by the correct scene and environment temperatures are offset.

Operating (Environment) Temperature

Scene Temperature

BBActual	10	15	20	25	30	35	40	45	50
-15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	2.85	7.15	0	0	0
30	0	0	0	0	4.46	5.85	0	0	0
45	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0
150	0	0	0	0	0	0	0	0	0
175	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0	0	0
350	0	0	0	0	0	0	0	0	0
425	0	0	0	0	0	0	0	0	0

Table 12 - Specific Offset Thermography Table

Note: Applying a temperature correction is a 2-step process:

1) Use one of the following typedefs in seekware.h to define the struct that holds the desired adjustment parameters:
   - `sw_global_therm_adjust_t`
   - `sw_scene_therm_adjust_t`
   - `sw_environment_therm_adjust_t`
   - `sw_specific_therm_adjust_t`
2) Call `Seekware_SetSettingEx` with one of the following settings defined in `sw_settings` and pass a pointer to the struct defined in (1):
   - SETTING_GLOBAL_THERM_ADJUST
   - SETTING_SCENE_THERM_ADJUST
   - SETTING_ENVIRONMENT_THERM_ADJUST
   - SETTING_SPECIFIC_THERM_ADJUST

At least one successful call to `Seekware_GetImage` or `Seekware_GetImageEx` must precede `Seekware_SetSettingEx` when applying a thermography adjustment, otherwise `Seekware_SetSettingEx` will fail and return `SW_RETCODE_ERROR`. To avoid this, it is recommended to wrap the calls to `Seekware_SetSettingEx` for thermography adjustments like the example code below:

Example Code: Apply a +5 degree offset to a 25 degree scene

```
if(Seekware_GetImageEx(dev, NULL, NULL, NULL) == SW_RETCODE_NONE){
        /*Apply temperature adjustments as needed*/
        sw_scene_therm_adjust_t scene_adjust;
        scene_adjust.scene_temp = 25.0f;
        scene_adjust.offset = 5.0f;
        Seekware_SetSettingEx(dev, SETTING_SCENE_THERM_ADJUST, &scene_adjust, sizeof(scene_adjust));
}
```

SETTING_TRANSIENT_CORRECTION_ENABLE

This setting enables a global transient correction of temperature readings. The correction is determined by the 'Offset' value shown in the equation below.

SETTING_TRANSIENT_CORRECTION_PARAMS

This setting determines the amplitude and decay used for transient correction. The offset is calculated based on the following equation where *t* is run time in seconds, and amplitude A and decay d are set by the user using the `sw_transient_corr_params_t` in Seekware.h. The parameters are not stored in camera memory and much be set after each device is opened.

$$\text{Offset} = -A\, e^{-\frac{t}{d}}$$

Note: On cameras with a QVGA sensor, time (t) resets to zero only when the camera is powered off. On cameras with a 206 sensor, time resets to 0 each time `Seekware_Open` is called.

SETTING_SMOOTHING

This setting smooths the display image. Passing `Seekware_SetSettingEx` a value > 0 for this setting enables smoothing.

SETTING_AUTOSHUTTER

This setting enables/disables the auto shutter. Passing `Seekware_SetSettingEx` a value > 0 for this setting enables autoshutter. Passing 0 disables the shutter. By default, this setting is enabled.

SETTING_MINMAX

This feature returns the min/max temperature values of a device's scene and the x,y coordinates of these values. A min/max value of "nan" instead of a temperature value means that the camera does not support min/max calculations. Use this feature as a parameter of `Seekware_GetSettingEx`. This feature must be used after getting a thermal image buffer, therefore it cannot be used with `Seekware_GetDisplayImage`.

SETTING_SHARPENING

This setting sharpens the display image. Passing `Seekware_SetSettingEx` a value > 0 for this setting enables sharpening.

SETTING_ENABLE_TIMESTAMP

This setting enables/disables timestamping. Passing `Seekware_SetSettingEx` a value > 0 for this setting enables timestamping. Passing 0 disables timestamping. By default, this setting is disabled.

SETTING_RESET_TIMESTAMP

This setting resets the timestamp counter. Passing `Seekware_SetSettingEx` any value for this setting resets the timestamp counter. Do not call `Seekware_SetSettingEx` for this setting if you do not want to reset the timestamp counter.

SETTING_TRIGGER_SHUTTER

This setting triggers the camera shutter. Passing `Seekware_SetSettingEx` a value > 0 for this setting triggers the shutter. Passing 0 does not trigger the shutter.

The following settings configure the Seekware image processing pipeline to create thermal data (pixel intensity) utilizing multiple Automatic Gain Control (AGC) algorithms. The three AGC algorithms provided are Histogram Equalization (HistEQ), Linear Min/Max, and Legacy HistEQ (all HistEQ implementations prior to Seekware SDK v2.15.0 use Legacy HistEQ). HistEQ is a non-linear AGC transformation, meaning that thermal data created by it does not correlate to temperature. Linear Min/Max is a linear AGC transformation, meaning that thermal data created by it does correlate to temperature.

SETTING_AGC_MODE

This setting configures the AGC mode used when creating thermal data for an image. The Seekware SDK allows you to select between three AGC algorithms. Passing `Seekware_SetSettingEx` a value of 0 sets the AGC mode to Legacy HistEQ. A value of 1 sets the AGC mode to Linear Min/Max. A value of 2 sets the AGC mode to HistEQ.

Figure 1 - Linear Min/Max

Figure 2 - Histogram Equalization

The figures above depict the same scene. Figure 1 is displayed using Linear Min/Max and Figure 2 is displayed using HistEQ. In the scene, there is a table with a stapler at room temperature, a hot coffee cup, a cold bottle of water, and a person in the background. Notice in the Linear Min/Max image that the color information clearly shows that the coffee is substantially hotter than the person. In the HistEQ image, the content of the table and background is enhanced such that you now see information that is not visible in the Linear Min/Max scene. Pay close attention to the color of the cup of coffee, as its temperature information is lost since the person is colored using the same "hot" colors (white).

In order to utilize the following SETTING_HISTEQ settings, AGC mode must equal a value of 2:

SETTING_HISTEQ_BIN_COUNT

This setting returns the number of bins used by the HistEQ algorithm. 2^x determines the exact bin count, where x is the number returned by this setting.

SETTING_HISTEQ_INPUT_BIT_DEPTH

This setting returns the number of input bits before performing an AGC calculation. The default is 16 bits.

SETTING_HISTEQ_OUTPUT_BIT_DEPTH

This setting returns the number of output bits after performing an AGC calculation. The default is 8 bits.

SETTING_HISTEQ_HIST_WIDTH_COUNTS

This setting returns the width of the active histogram used by the HistEQ algorithm. This value is useful for gathering information and tuning other stretch limit parameters.

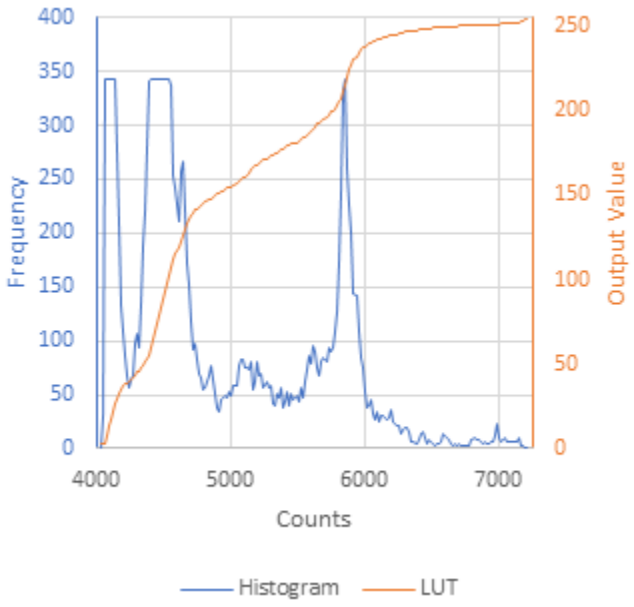
SETTING_HISTEQ_PLATEAU_VALUE

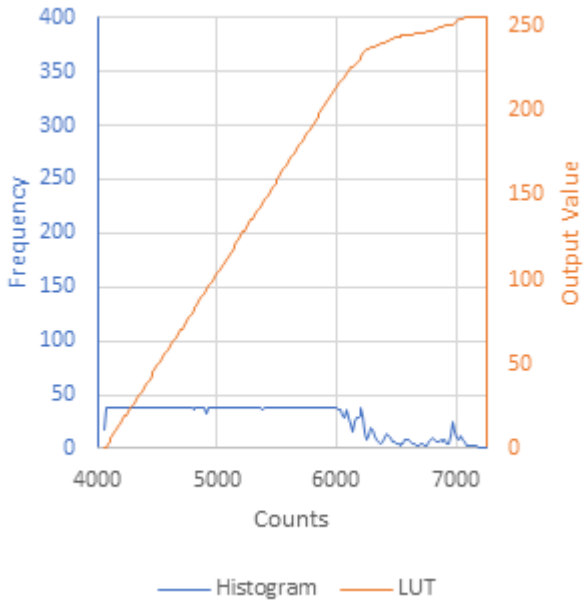
This setting limits the number of pixels that can be assigned to a single histogram bin. This is necessary when a large area of the scene, such as the sky, is very uniform and should be assigned a small number of colors. The default value of 0.05 (5%) means that a maximum of 5% of the total number of pixels can be assigned to a single bin. Bins are capped at this height and clipped pixels are not re-distributed in the histogram. See figures below for further explanation:

Figure 3 is a high-contrast scene with an optimally-tuned *Plateau Value*, as plotted in Figure 5. Figure 6 is a plot with a very small *Plateau Value*; note that the LUT is very linear, which results in an image (Figure 4) that loses some detail within each object (flatter coffee cup, water bottle, and background).

Figure 3 - Optimally tuned Plateau Value

Figure 4 - Very low Plateau Value





Gain controls the maximum number of output values that may be allocated for each count of input data. Typically, gain is limited so that intense and concentrated data resulting in histogram peaks do not use up an extraordinary amount of output count values, reducing the contrast of the remainder of the image. A side-effect of limiting gain, however, is that it may cause HistEQ to not assign each of the 256 output values. This may be a good thing if, for example, the scene itself is very uniform (has little contrast) and assigning too many colors would only increase noise. Conversely, a scene with a lot of content that is gain-limited may appear very washed out when only a subset of the 256 available colors are used. In this type of scene, a better image could be created by reducing the gain-limiting (increasing gain) without an adverse effect on noise in the image.

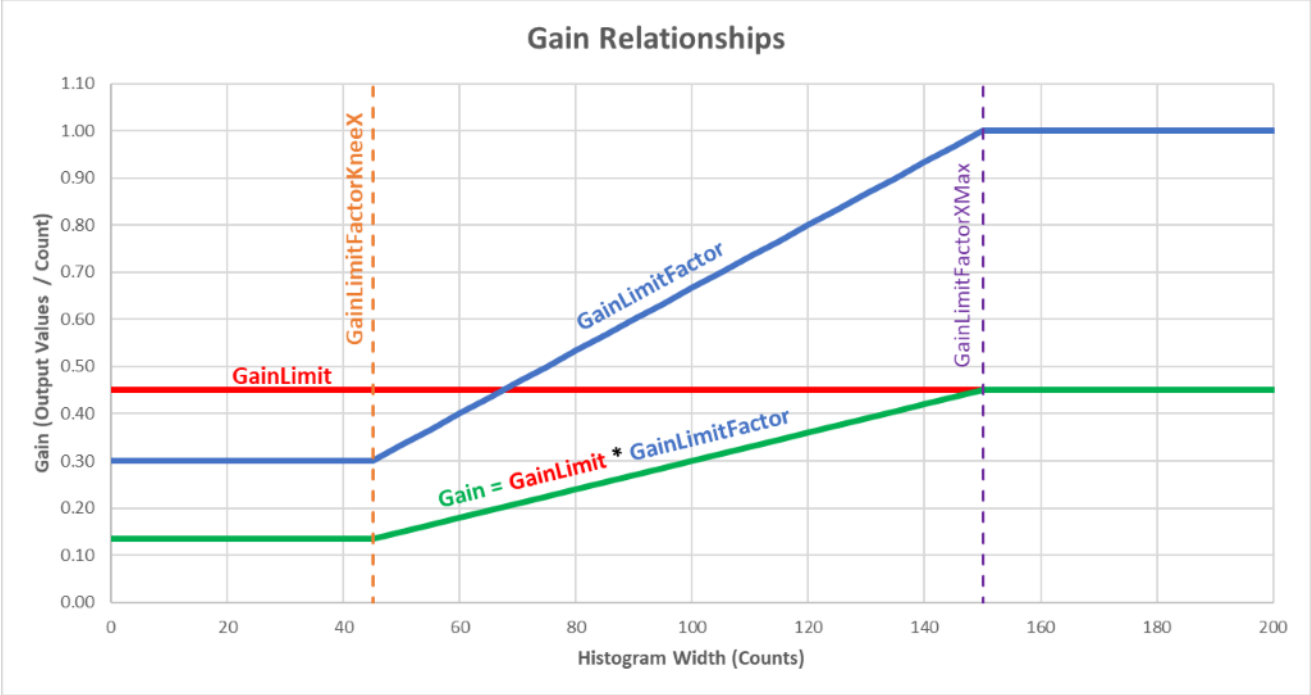


Figure 7 - GainLimit and GainLimitFactor

Gain control itself is separated into two pieces: *GainLimit* and *GainLimitFactor*. *GainLimit* is a single setting that controls the maximum gain (slope) of output values per count. *GainLimitFactor* adds additional flexibility in automatically reducing gain as a function of the histogram width. This additional control allows the user to tune the *GainLimit* value for a particular application, and then allow the algorithm to reduce gain as the scene content decreases. This helps to not magnify sensor noise in low-contrast scenes. The *GainLimitFactor* only has an effect in low-contrast scenes where noise begins to adversely effect image quality. All scenes with enough content are gain-limited only by the *GainLimit* value itself.

$$Gain = (GainLimit * GainLimitFactor)$$

$$GainLimitFactor = \begin{cases} GainLimitFactorYMin & HistWidthCounts \leq GainKneeX \\ \left(\frac{HistWidthCounts}{GainLimitFactorXMax}\right) & GainKneeX < HistWidthCounts < GainLimitFactorXMax \\ 1.0 & HistWidthCounts \geq GainLimitFactorXMax \end{cases}$$

$$GainLimitFactorKneeX = (GainLimitFactorYMin * GainLimitFactorXMax)$$

SETTING_HISTEQ_GAIN_LIMIT

This setting configures the max percentage of output colors per sensor count. The default value of 0.45 means that no more than 45% of the available 256 color output values may be assigned to a single sensor count. Setting an appropriate *GainLimit* will ensure that highly concentrated parts of scenes (peaks in the histogram) are not assigned too many output values, thus ensuring that plenty of output values are reserved for other parts of the scene. Increasing the *GainLimit* can increase contrast (and noise) within a scene, while decreasing the *GainLimit* can reduce the contrast of a scene. Decreasing the *GainLimit* can cause the HistEQ algorithm to not utilize all output (and color) values. If the *GainLimit* is too low in a high contrast scene, the image will appear washed out.

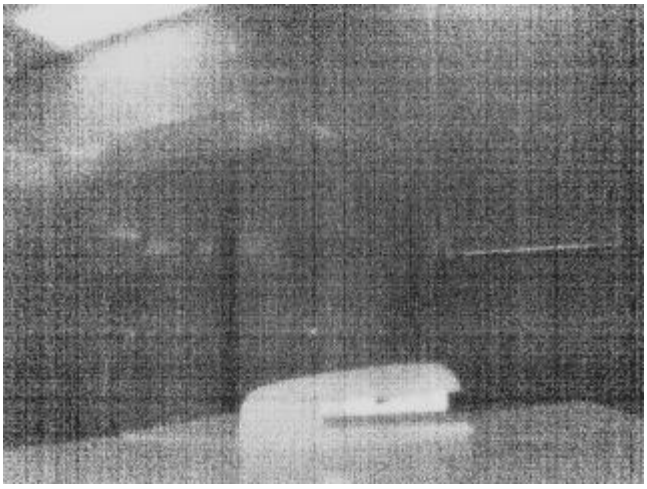


Figure 8 - High GainLimit

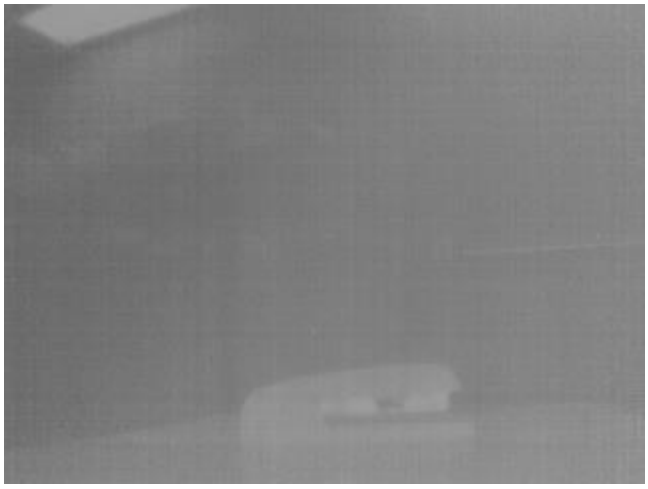


Figure 9 - Low GainLimit

The images above illustrate the difference between high and low *GainLimit* settings in a low-contrast scene. The optimal setting for this scene would likely be a *GainLimit* setting slightly higher than Figure 9.

Figure 10 - High GainLimit

Figure 11 - Low GainLimit

The images above illustrate the difference between high and low *GainLimit* settings in a high-contrast scene. Note that Figure 11 appears somewhat washed out, while in Figure 10, the background wall and light have more contrast.

SETTING_HISTEQ_GAIN_LIMIT_FACTOR_ENABLE

This setting determines whether *GainLimitFactor* is used to determine total Gain. When enabled, Gain = *GainLimit* * *GainLimitFactor*. When disabled, Gain = *GainLimit*. Passing `Seekware_SetSettingEx` a value > 0 enables this setting.

GainLimitFactor is a mechanism to alter the transfer function gain for particularly low-contrast scenes where noise may adversely affect the image. *GainLimitFactor* is controlled through a set of four control settings. In addition to those control settings, there are a handful of read-only settings that are useful when tuning *GainLimit* and *GainLimitFactor*.

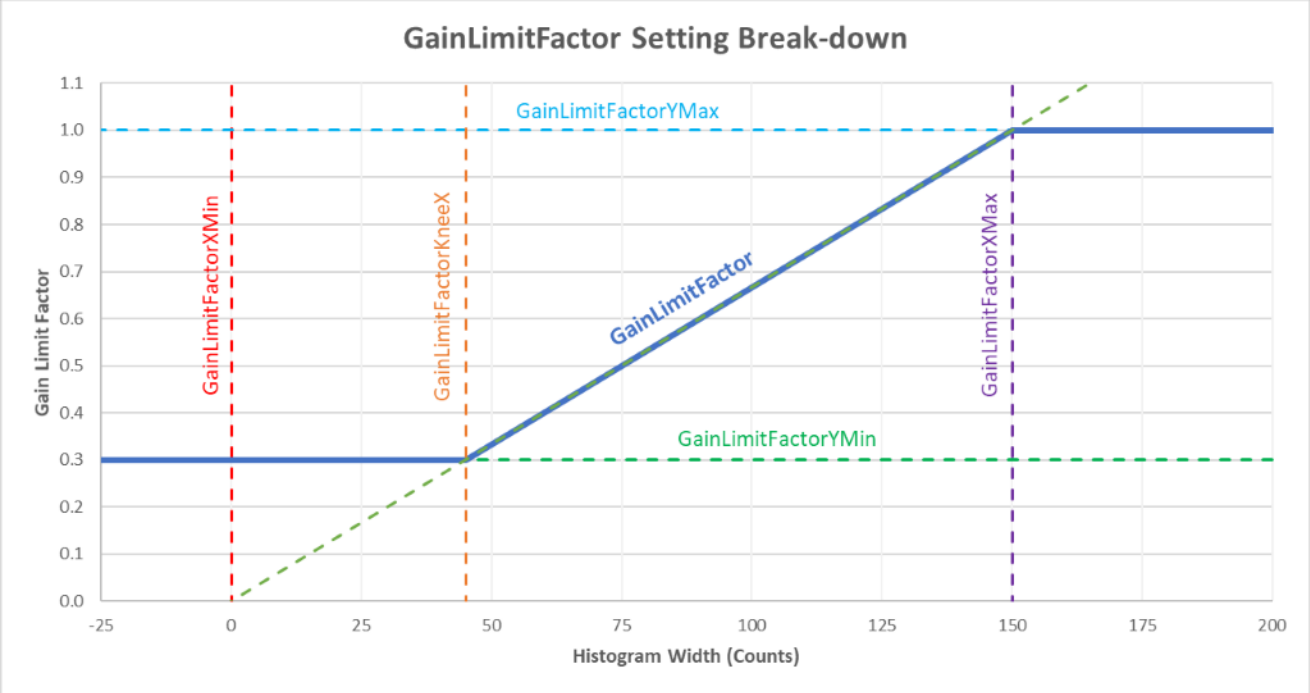


Figure 12 - GainLimitFactor Parameters

Typically, low-contrast scenes, such as a wall or isothermal room, are relatively uniform and would be assigned too many output values resulting in a very noisy image. When a scene has very little content it results in a narrow histogram. The *GainLimitFactor* collection of settings will smoothly reduce gain, thus reducing the number of output values used. This helps to make low-contrast scenes appear flat while maintaining the flexibility of having the *GainLimit* parameter set to a higher value to maintain higher contrast in non-flat scenes.

A simple example is if you cover the sensor lens entirely with your hand. The scene is extremely uniform and has almost no contrast (perhaps less than a degree). If HistEQ was not gain-limited (the *GainLimit* value is high enough that all 256 output values are used), then the scene would appear very noisy as HistEQ attempts to maximize contrast in a scene with very little content. If instead, *GainLimitFactor* is enabled, then the *GainLimit* is reduced as a result of the narrow histogram and only a fraction of the output values are used. For example, perhaps 10 output values are used instead of the full 256 resulting in a relatively flat image with minimal noise. Note that in scenes that are gain-limited, HistEQ will center the values within the output range (around 128).

SETTING_HISTEQ_GAIN_LIMIT_FACTOR

This setting returns the *GainLimitFactor* value used to calculate total Gain. This setting returns a floating-point value between 0 and 1. This setting is most effective in low-contrast scenes where noise begins to adversely affect image quality.

SETTING_HISTEQ_GAIN_LIMIT_FACTOR_XMAX

This setting sets the width of the histogram where the *GainLimitFactor* will begin to kick in. The default value of 150 counts means that for scenes where the histogram content is all contained within 150 counts or less, *GainLimit* will be scaled down. The amount of scaling is proportional to the width of the histogram such that a histogram width of *GainLimitFactorXMax* (150 by default) or higher will result in no scaling, and a histogram width of anything below *GainLimitFactorKneeX* would result in a *GainLimitFactor* of *GainLimitFactorYMin* (0.3 by default).

SETTING_HISTEQ_GAIN_LIMIT_FACTOR_YMIN

This setting sets the minimum value for *GainLimitFactor*. The default value of 0.30 means that the minimum value of *GainLimitFactor* is 30% by default.

SETTING_HISTEQ_ALPHA_TIME

When scene content rapidly changes, the HistEQ output can flash and distract the user. This setting fixes this problem by blending the current frame's histogram with the previous frame's histogram. The setting controls the length of one time-constant, where three time-constants typically result in the new scene being completely blended in. Therefore, the default setting of 1/3 will blend one second of previous frames, such that if a completely new scene is presented to HistEQ, it will take approximately 1 second for the histogram to fully incorporate the new scene. A value of zero effectively disables alpha blending.

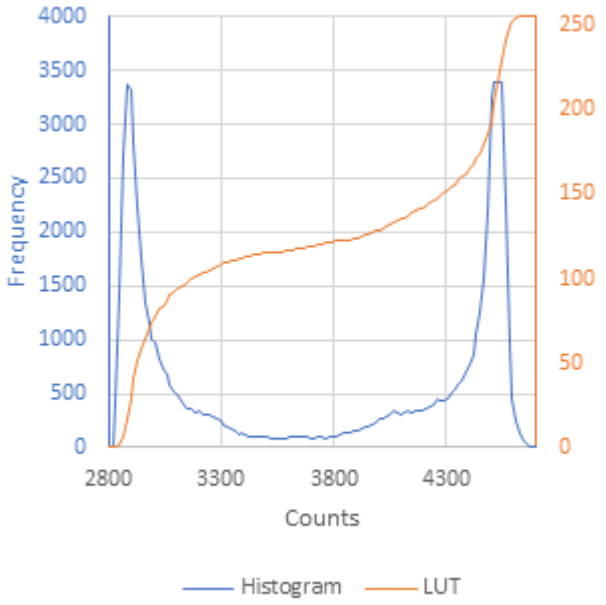


Figure 13 - Outdoor Scene with Default Trim

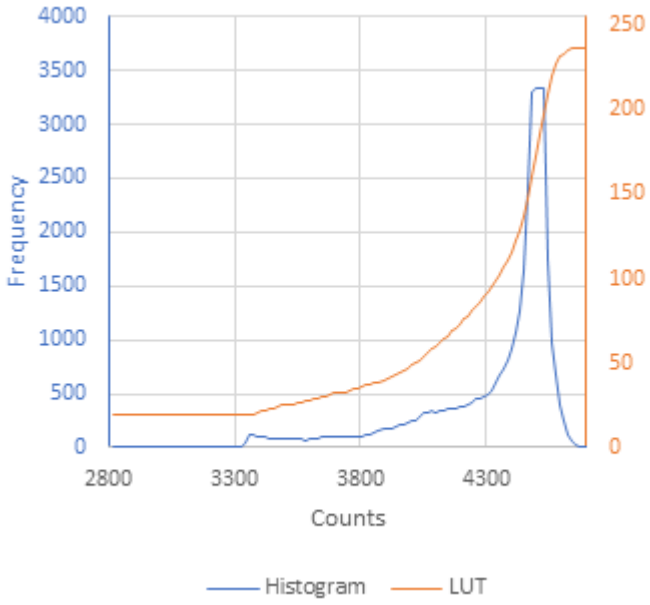


Figure 14 - Outdoor Scene with 40% Left Trim

SETTING_HISTEQ_TRIM_LEFT

This setting removes outliers from the left side of the input scene's histogram. The default of 0.001 trims the left 0.1 % of the scene histogram and leaves the remaining 99.9 % of image content to feed HistEQ. Setting this parameter helps to prevent situations where a small number of outlier pixels are adversely affecting the scene's contrast.

SETTING_HISTEQ_TRIM_RIGHT

This setting removes outliers from the right side of the input scene's histogram. The default of 0.001 trims the right 0.1 % of the scene histogram and leaves the remaining 99.9 % of image content to feed HistEQ. Setting this parameter helps to prevent situations where a small number of outlier pixels are adversely affecting the scene's contrast.

In order to utilize the following SETTING_HISTEQ settings, SETTING_AGC_MODE must equal a value of 2:

SETTING_LINMINMAX_MODE

This setting configures the Linear Min/Max mode. Passing Seekware_SetSettingEx a value of 0 sets the Linear Min/Max mode to Auto. A value of 1 sets the mode to Manual. A value of 2 sets the minimum to Manual and the maximum to Auto. A value of 3 sets the minimum to Auto and the maximum to Manual.

The Linear Min/Max AGC mode operates in two distinct configurations (or a combination of the two). In Auto mode, the minimum and maximum values of the scene content are found, then the 256 output values are evenly stretched between those bounding values. In Manual mode, the minimum and maximum values are user-specified in terms of counts and the 256 output values are evenly stretched between those bounding values. The minimum and/or maximum may also be user-specified, creating a hybrid mode where one setting can be manual and the other automatic.

The Linear AGC is useful when it is important that colder scene content remain lower-valued than hotter scene content. This ensures that an output values of 0 is always colder than 1, which is always colder than 2, all the way up to 255 which represents the hottest part of the scene. If you need a particular scene object to always have the same output value (and therefore be colored the same), then you must use Manual mode. When in Manual mode, any scene content lower than the user-specified minimum will have an output value of 0. Likewise, any scene content higher than the user-specified maximum will have an output value of 255.

SETTING_LINMINMAX_MIN_LOCK

This setting specifies the lower bound of the Linear AGC. If the Linear Min/Max Mode is set to either 0 (full Auto) or 3 (minimum Auto), this setting is ignored and the minimum value is set by the lowest scene value.

SETTING_LINMINMAX_MAX_LOCK

This setting specifies the upper bound of the Linear AGC. If the Linear Min/Max Mode is set to either 0 (full Auto) or 2 (maximum Auto), this setting is ignored and the maximum value is set by the highest scene value.

SETTING_LINMINMAX_ACTIVE_MIN_VALUE

This is a read-only setting that returns the minimum count value from the last scene. This is useful for characterizing and tuning.

SETTING_LINMINMAX_ACTIVE_MAX_VALUE

This is a read-only setting that returns the maximum count value from the last scene. This is useful for characterizing and tuning.

FEATURE_OEM

When calling either `Seekware_GetSettingEx` or `Seekware_SetSettingEx`, structure the index/setting parameter as (`FEATURE_OEM` + `SeekProvidedSetting`), where `SeekProvidedSetting` will be provided after contacting Seek.

Seekware_GetSpot

```
sw_retcode Seekware_GetSpot(psw id, float *temp, float *min, float *max)
```

Description

Retrieve the spot thermography for the center 6x6 pixels of the most recently processed image frame. A spot value of "nan" instead of an actual temperature value means that the camera does not support spot temperature readings. If this function is called without specifying a temperature buffer in `Seekware_GetImage` or `Seekware_GetImageEx`, `Seekware_GetSpot` will only return spot temperature, not min/max.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`temp`	A float pointer to the memory location to put the spot temperature.
`min`	A float pointer to the memory location to put the minimum temperature.
`max`	A float pointer to the memory location to put the maximum temperature.

Seekware_SetUserLUT

```
sw_retcode Seekware_SetUserLUT(
      psw id, uint32_t lut_index, uint32_t *lut_data, uint32_t length
)
```

Description

Loads user LUT data to a USER LUT. Only supports ARGB32 look-up tables.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`lut_index`	Index of the user LUT to set. SW_LUT_USER0 <= lut_index <= SW_LUT_USER4.
`lut_data`	A uint32_t array containing the LUT data to write. Should be NR_LUTCOLORS elements.
`length`	Number of LUT elements to write. Should be NR_LUTCOLORS.

Seekware_GetImage

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

Description

This function 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 a buffer to hold filtered 16 bit greyscale image data.
`temperature`	A pointer to a buffer to hold full frame temperature data.
`display`	A pointer to a buffer to hold ARGB colored display data.

Seekware_GetImageEx

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

Description

This function is identical to `Seekware_GetImage` except that it appends telemetry data to an extra row immediately following the image data. **Therefore, for this function the `binary` buffer must be sized to include one additional row.** The extra row contains telemetry data supplied by the Seekware Library for each frame. The telemetry data definition follows:

Telemetry Row Index	Value	Description
0	Field Count	The field count is the index of each frame that comes from the sensor. It continues to increment even during shutter closures therefore, it can be used to detect shutter closures because shutter closures cause a discontinuity in the field count sequence.
1		
2	Temperature Diode	The temp diode count value is an uncalibrated, raw sampling of the temperature diode voltage.
3	EnvTemp	EnvTemp is the estimated environment temperature based on FPA.
4		
5	Timestamp	The timestamp value displays a timer counter of the current frame. This value is controlled by both the `SETTING_ENABLE_TIMESTAMP` and `SETTING_RESET_TIMESTAMP` settings.
6		
7		
8		
9+	Reserved	

Table 13 – Telemetry Data Definition

Parameter(s)

`id` A pointer to a Seekware device structure.

`binary` A pointer to a buffer to hold filtered16 bit greyscale image data + 1 telemetry line.

`temperature` A pointer to a buffer to hold full frame temperature data.

`display` A pointer to a buffer to hold ARGB colored display data.

Seekware_GetThermographyImage

```
sw_retcode Seekware_GetThermographyImage(psw id, uint16_t* thermography,
uint32_t num_elements)
```

Description

Returns a frame of fixed point uint16_t thermography values. To get temperature, apply the following formula: $\text{Temperature} = \left(\frac{counts}{64}\right) - 40$. Temperature units are controlled by `SETTING_TEMP_UNITS`. This data can be used with ProcessDisplayImage to perform AGC, zoom, colorization.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`thermography`	A pointer to a uint16_t frame buffer.
`num_elements`	Number of elements in the display buffer.

Seekware_GetDisplayImage

```
sw_retcode Seekware_GetDisplayImage(
      psw id, uint32_t* display , uint32_t num_elements
)
```

Description

Returns a frame of ARGB8888 display values with auto gain control enabled. This function does not calculate thermography data for Spot, Min, and Max.

Parameter(s)

`id`	A pointer to a Seekware device structure.
`display`	A pointer to a uint32_t frame buffer.
`num_elements`	Number of elements in the display buffer.

Sample Applications

seekware-simple

Grabs frames from the camera and prints thermography information to stdout. Save full frame thermography data to a csv file every 10th frame.

seekware-sdl

Draws a display image using SDL2 on Linux and Windows

Running the seekware-sdl app: `seekware-sdl`

Available command line options:

`-h | --help` Print usage information and exit.

`-lut | --lut <l>` Sets the given LUT for RGB image.

Valid LUTs are reported with –h.

seekware-fbdev (Linux only)

Linux only. Draws an image directly to the Linux frame buffer. Demonstrates how to avoid image tearing with the FBIO_WAITFORVSYNC and FBIOPAN_DISPLAY ioctls.

Running the seekware-fbdev app: `seekware-fbdev`

Available command line options:

`-h | --help` Print usage information and exit.

`--device <dev>` The name of the framebuffer device. Its default is "/dev/fb0".

`-d | --double` Doubles the size of the displayed rectangle(s) both in horizontal and vertical directions.

`-lut | --lut <l>` Sets the given LUT for RGB image.

Valid LUTs are reported with –h.

seekware-upgrade

Shows usage of Seekware_Upload that performs a firmware upgrade on compatible devices. Firmware upgrade files are provided by Seek Thermal Engineering.

Running the seekware-upgrade app: `seekware-upgrade <upgrade-file>`

Available command line options:

`-h | --help` Print usage information and exit.

FAQ

How do I update firmware on a Seek camera?

1) Install the Linux SDK on an Ubuntu or Debian based PC. See page 7.
2) Navigate to the SDK installation directory and build the included seekware-upgrade sample app. The source code for this tool is provided which demonstrates usage of the Seekware_UploadFirmware API.

```
cd /usr/src/seekware-upgrade
make
>> g++ -o seekware-upgrade objs/seekware-upgrade.o  -lseekware -lstdc++ -lpthread
-ldl `pkg-config --libs libusb-1.0`
ls
>> include  Makefile  objs  seekware-upgrade  src
```

3) Attach a camera and run seekware-upgrade with the path to the upgrade file provided by Seek as the first command line parameter:

```
./seekware-upgrade COMPACT-4.8.1.9.bin
```

On Success:

```
seekware-upload-firmware - uploads new firmware to a Seek Device
Found 1 camera devices...
Opening device...
Current Firmware Version: 4.8.0.9
Updating Firmware...please wait...
Firmware Upgrade Successful!
New Firmware Version: 4.8.1.9
Closing device...
```

On Failure:

```
seekware-upload-firmware - uploads new firmware to a Seek Device
Found 1 camera devices...
Opening device...
Current Firmware Version: 4.8.0.9
Updating Firmware...please wait...
Firmware Upgrade Failed!
New Firmware Version: 4.8.0.9
Closing device...
```

How do I apply a temperature adjustment using the new THERM_ADJUST Settings?

See the THERM_ADJUST settings in the SW_SETTINGS section in this document to view an example of how to apply a temperature adjustment.

How do I apply a transient temperature correction?

See the SETTING_TRANSIENT_CORRECTION_ENABLE and SETTING_TRANSIENT_CORRECTION_PARAMS settings for information on how the transient offset is calculated. Use the `Seekware_GetSettingEx`/ `Seekware_SetSettingEx` functions to apply these settings.

How do I apply a custom color LUT?

This is a 2 step process. First, call `Seekware_SetUserLUT`, specifying the lut_index parameter to be any of the Seek defined SW_LUT_USER enum values. After performing this call, the desired user LUT can now be accessed in the SETTING_ACTIVE_LUT setting via a `Seekware_SetSetting` or `Seekware_SetSettingEx` function call.

Where do I find the Makefiles for each sample app?

Assuming that the user installs the SDK package to the default /usr location, the location of the Makefiles can be found in the /usr/src directory.

Example :

```
user@user-linuxpc:/usr/src$ ls seekware-*

seekware-simple:

include  Makefile  objs  seekware-simple  src


seekware-sdl:

include  Makefile  objs  seekware-sdl  src


seekware-fbdev:

include  Makefile  objs  seekware-fbdev  src


seekware-upgrade:

include  Makefile  objs  seekware-upgrade  src
```

Why does my SDK installation abort when running this command?

```
~/Downloads/Seekware-SDK-2.15.2/x86_64-linux-gnu$ ./install.sh
Enter target directory for SDK (default: /usr):
You are about to install the SDK to "/usr". Proceed[Y/n]?
Installation aborted!
```

The Seekware SDK installation script must run as root in order to install the Seek Thermal udev rules.

```
~/Downloads/Seekware-SDK-2.15.2/x86_64-linux-gnu$ sudo ./install.sh
```

How do I apply timestamping using the provided Seekware™ SDK settings?

In order to utilize timestamping, the user must first determine if they are using a version of camera firmware that supports timestamping. Currently, CompactPro/J3 firmware versions of 4.9.1.14 and 4.18.1.14 or greater support timestamping. Compact/J2 firmware does not support timestamping currently.

There are two SDK settings that control timestamping: SETTING_ENABLE_TIMESTAMP and SETTING_RESET_TIMESTAMP. The below example shows how to correctly utilize these settings. Each setting only needs to be enabled once before calling `Seekware_GetImageEx`.

Example Code: Enable timestamping

```
int enableTimestamp = 1;
int resetTimestamp = 1;

// Reset Timestamp – passing any value for resetTimestamp will enable this setting
if(Seekware_SetSettingEx(dev, SETTING_RESET_TIMESTAMP, &resetTimestamp, sizeof(resetTimestamp)) != SW_RETCODE_NONE) {
    printf("Reset Timestamp SetSettingEx Failed!\n");
    return NULL;
}

// Enable Timestamp – passing a value of greater than 0 for enableTimestamp will enable this setting
if(Seekware_SetSettingEx(dev, SETTING_ENABLE_TIMESTAMP, &enableTimestamp, sizeof(enableTimestamp)) != SW_RETCODE_NONE) {
    printf("Enable Timestamp SetSettingEx Failed!\n");
    return NULL;
}
```

The below example shows how to view timestamping data:

Example Code: View timestamping output (in seekware-simple.c sample app)

```
// Timestamp displayed in microseconds
uint64_t timestampUS = (uint64_t) frame_raw[dev->frame_rows*dev->frame_cols+5] + ((uint64_t) frame_raw[dev->frame_rows*dev->frame_cols+6] << 16) + ((uint64_t) frame_raw[dev->frame_rows*dev->frame_cols+7] << 32) + ((uint64_t) frame_raw[dev->frame_rows*dev->frame_cols+8] << 48);

// Timestamp displayed in seconds
float timestampS = (float) timestampUS / 1.0e6f;
```

Release Notes

v3.2

- Resolves an issue where Seekware_Open could hang on Windows hosts.
- Adds support for C2X Mosaic cores

v3.1

- Added new API functions (Seekware_Stop and Seekware_Start) that suspend or resume background frame processing. When frame processing it stopped, the camera will enter low power mode, but remain open.
- Fixed an issue that prevented selecting a user defined color lut (SW_LUT_USER0 - SW_LUT_USER4) with both Seekware_SetSetting and Seekware_SetSettingEx.
- Added support for 2850 J-series cores.

v3.0

- Added cross platform support for Windows. Moving forward, Linux and Windows builds of the Seekware SDK will maintain feature parity by sharing the common API outlined in this document.

v2.21

- Fixed an issue where certain J-series cores would report inaccurate thermography for the first 5 frames.

v2.20

- Added support for 2808 J-series cores.

v2.19

- Fixed an issue with error reporting during certain firmware upgrade failures.
- libseekware.so no longer requires librt.so for arm-linaro-linux-gnueabi and arm-linaro-linux-gnueabihf targets. For more detail on the dependencies of libseekware.so for each build variant see pages 6-7.

v2.18

- Improved reliability during firmware upgrades.
- Minor improvements to the seekware-upgrade sample app.
- Added the ability to call Seekware_GetSdkInfo with id=NULL for accessing the Seekware SDK version without a connected device.
- SETTING_AGC_MODE will default to Legacy HistEQ for all J series cores.

v2.17

- Added new build variants for armv7 and armv8 targets that support uclibc and musl libc.
- Added a new build variant, arm-linaro-linux-gnueabihf that lowers the minimum requirement for libstdc++ to GLIBCXX_3.4.11+ from GLIBCXX_3.4.19+
- Added support for 2470 J3 cores.

v2.16

- Fixed a bug where the Seek Device could not reopen after previously closing.

v2.15

- Added new settings for HistEQ and Linear Min/Max (SETTING_AGC_MODE, SETTING_HISTEQ_BIN_COUNT, SETTING_HISTEQ_INPUT_BIT_DEPTH, SETTING_HISTEQ_OUTPUT_BIT_DEPTH, SETTING_HISTEQ_HIST_WIDTH_COUNTS, SETTING_HISTEQ_PLATEAU_VALUE, SETTING_HISTEQ_GAIN_LIMIT, SETTING_HISTEQ_GAIN_LIMIT_FACTOR_ENABLE, SETTING_HISTEQ_GAIN_LIMIT_FACTOR, SETTING_HISTEQ_GAIN_LIMIT_FACTOR_XMAX, SETTING_HISTEQ_GAIN_LIMIT_FACTOR_YMIN, SETTING_HISTEQ_ALPHA_TIME, SETTING_HISTEQ_TRIM_LEFT, SETTING_HISTEQ_TRIM_RIGHT, SETTING_LINMINMAX_MODE, SETTING_LINMINMAX_MIN_LOCK, SETTING_LINMINMAX_MAX_LOCK, SETTING_LINMINMAX_ACTIVE_MIN_VALUE, SETTING_LINMINMAX_ACTIVE_MAX_VALUE)
- Performance improvements to the Seekware imaging pipeline.
- Image processing improvements for automotive products.
- Added support for Microcore Starter Kits.
- Added new sample apps seekware-sdl and seekware-fbdev, which replace seekware-test.
- Added new `sw_retcode` value (`SW_RETCODE_DISCONNECTED`)
- Added support for soft-float ARM7 targets.

v2.14

- Added new settings (SETTING_SHARPENING, SETTING_ENABLE_TIMESTAMP, SETTING_RESET_TIMESTAMP, and SETTING_TRIGGER_SHUTTER) that are used as parameters of `Seekware_GetSettingEx` and `Seekware_SetSettingEx`.
- Added `Seekware_GetImageEx` telemetry lines to display timestamp value.
- Fixed a bug where EnvTemp was not updating correctly.
- Fixed a bug where SETTING_SMOOTHING was not being properly enabled.
- Removed ProcessDisplayImage function documentation and set/get features to control it. A future release will include AGC control for the `Seekware_GetImage` display buffer.
- Added new `sw_retcode` values (`SW_RETCODE_NOTSUPPORTED` and `SW_RETCODE_INVALIDSETUP`).
- Renamed FEATURE_MINMAX to SETTING_MINMAX.

v2.13

- Improved transient correction functionality.

v2.12

- Added ProcessDisplayImage function and set/get features to control it.

v2.10

- Fixed a bug where some seek devices would report a usb error during `Seekware_Open`
- Fixed a bug where some seek devices would report incorrect min and max temperature values.

v2.9

- Added new API functions (`Seekware_GetThermographyImage` and `Seekware_GetDisplayImage`) that each return a frame of either fixed point thermography values or ARGB display values.
- Added new features (FEATURE_MINMAX and FEATURE_OEM) that are used as parameters of `Seekware_GetSettingEx` and `Seekware_SetSettingEx`.
- Deprecated API functions (`Seekware_GetSetting` and `Seekware_SetSetting`). For v3.0+, `Seekware_SetSettingEx` and `Seekware_GetSettingEx` will be renamed to `Seekware_SetFeature` and `Seekware_GetFeature`.

Attributions

1. **Linux Standard Base** is a trademark of The Linux Foundation.

2. **Linux** is a registered trademark of Linus Torvalds.

3. **Ubuntu** is a trusted open source product. Canonical owns and manages certain intellectual property rights in Ubuntu and other associated intellectual property (Canonical IP) and licenses the use of these rights to enterprises, individuals and members of the Ubuntu community in accordance with their IP rights Policy. Canonical's IP rights Policy. This policy is published by Canonical Limited (Canonical, we, us and our) under the Creative Commons CC-BY-SA version 3.0 UK license.

4. **Windows** and **Visual Studio** are registered trademarks of Microsoft Corporation in the United States and/or other countries.

5. **ARM** is a registered trademark of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. **ARM6** (ARMv6), **ARM7** (ARMv7), and **ARM8** (ARMv8) are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

6. **Compact, CompactXR** and **CompactPRO** are trademarks of Seek Thermal, Inc.