

Intel® Converged Security Engine Firmware Integrated Clock Controller (ICC) Tool

Tools User Guide

March 2019

Revision 1.0

Intel Confidential



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2018, Intel Corporation. All rights reserved.



Contents

1	Introduction	6
1.1	Terminology	6
1.2	Reference Documents	7
2	Intel® Clock Commander (CCT) Tool	8
2.1	Command Line Interface	8
2.2	Detail on Supported Commands	9
2.2.1	? Or h - Help Menu	9
2.2.2	gcc – Get Clock Capabilities	9
2.2.3	gp – Get ICC Profile	9
2.2.4	sp – Set ICC Profile	10
2.2.5	gc – Get ICC Clock Settings	10
2.2.6	sc – Set ICC Clock Settings	10
2.2.7	rm – read from MPHY table	11
2.2.8	sm – Set mPHY Settings	11
2.2.9	Rvrt - Revert FW to defaults	12
2.3	Examples – How to Use CCT Commands	13
2.3.1	Help Menu	13
2.3.2	Get Clock Capabilities	13
2.3.3	Get Profiles	14
2.3.4	Set Profile	14
2.3.5	Get Clock Settings	14
2.3.6	Set Clock Settings	15
2.3.7	Read mPHY Table	15
2.3.8	Set mPHY Table	15
2.3.9	Revert FW to Default	15
2.4	Error and Status Messages	16
2.4.1	Clock Commander Tool Error and Status Messages	16
3	Intel® ICCS SDK	17
3.1.1	Intel® ICCS SDK	17
3.1.2	Intel® ICCS Control Library SDK Features	17
3.2	Intel® Integrated Clock Controller Service Data Structures	17
3.2.1	ICC_HECI_CLOCK_ID Type	18
3.2.2	ErrorCodes Type	18
3.2.3	ICC_GET_CLOCK_SETTINGSEx Type	18
3.2.4	ICC_SET_CLOCK_SETTINGSEx Type	19
3.3	Intel® Integrated Clock Controller Service API – Clock Manipulation Interface	20
3.3.1	Get Current Clock Settings Wrapper	21
3.3.2	Set Current Clock Settings Wrapper	21
4	Intel® Watchdog Timer Driver (Intel® WDT Driver)	22
4.1	Communication with the Driver	22
4.2	Device Functionally - IOCTLs	22
4.2.1	Acquire Intel® Watchdog Timer Driver Control	23
4.2.2	Intel® Watchdog Timer Driver Control	23
4.2.3	Reload and Start Intel® Watchdog Timer Driver	24
4.2.4	Stop Intel Watchdog Timer Driver	25
4.2.5	Get Supported Countdown Data	26



4.2.6	Get Intel® Watchdog Timer Driver Status.....	27
4.2.7	Start Intel® Watchdog Timer Driver on Next OS Load	29
4.3	Driver Access.....	30
4.4	Usage Flows Example.....	30
4.4.1	Normal Flow	31
4.4.2	Start on Next Boot Flow	32



Revision History

Revision Number	Description	Revision Date
0.6	<ul style="list-style-type: none">• Initial release.	August 2017
0.7	<ul style="list-style-type: none">• Added Set/Get Register commands• Added Set/Get mPHY commands• Removed Set/Get Record Commands• Removed Set/Get chipset Init commands• Listed examples for newly added commands	February 2018
0.8	<ul style="list-style-type: none">• Added ICC SDK guide	May 2018
0.9	<ul style="list-style-type: none">• Added new command to read from Chipset Init file	June 2018
1.0	<ul style="list-style-type: none">• Reformatted all Tools.	March 2019

§ §



1 Introduction

The purpose of the document is to provide guidance on the usage of the tools provided for Intel® Converged Security Engine (Intel® CSE) Firmware Integrated Clock Controller (ICC) included within the Intel firmware kit.

This document covers the usage of the CCT tool available in the **Tools\System_Tools\ICC Tools\CCT** directory And the ICC SDK available in the **Tools\System_Tools\ICC Tools\ICC SDK** directory within the Intel® CSE FW kit.

1.1 Terminology

Acronym or Term	Definition
API	Application Programming Interface
BIOS	Basic Input Output System
CCT	Clock Commander Tool
CCTwin	Windows* command line version of the Clock Commander Tool.
CPU	Central Processing Unit
DLL	Dynamic Link Library
Intel® FIT	Intel® Flash Image Tool
FW	Firmware
Intel® ICCS	Intel® Integrated Clock Controller Services
Intel® CSE	Intel® Converged Security Engine
Intel® MEI	Intel® Management Engine Interface (formerly HECI)
PCH	Platform Controller Hub
Permanent UOB	UOB that is applied on every boot.
UOB	Update on Boot. A record of ICC registers setting that is applied on the next platform boot.



1.2 Reference Documents

Document	Document No./Location
Lake Field Platform Controller Hub (LKF PCH) SPI Programming Guide	FW release kit
Lake Field Platform Controller Hub (LKF PCH) Intel® Management Engine Firmware Bring Up Guide	FW release kit
Lake Field Platform Controller Hub (LKF PCH) External Design Specification (EDS) Vol 1 & Vol 2	CDI#: Vol 1 → 574211 Vol 2 → 575200





2 Intel® Clock Commander (CCT) Tool

Intel® Clock Commander Tool (CCT) tool is mainly used to collect register programming and clock details of Intel® Converged Security Engine (Intel® CSE) Firmware Integrated Clock Controller (ICC) module.

2.1 Command Line Interface

CCT.exe supports the following command line options.

The command syntax for the CCT tool is CCT [-**Verbose options**] -**command** [-**arguments**].

The available CCT commands are:

CTT Commands	
? Or h	Help Menu
gcc	Get ICC Clocks Capabilities
gp	Get ICC Profile
sp	Set ICC Profile
gc	Get ICC Clock Settings
sc	Set ICC Clock Settings
rm	Read from mPHY table
sm	Set mPHY Settings
rvrt	Revert FW to defaults or nominal

Note: Refer the next section to get detail of usage, syntax and supported/expected argument options with each command.



2.2 Detail on Supported Commands

2.2.1 ? Or h - Help Menu

Usage: This is command to display help menu. It lists available commands supported by CCT tool.

Syntax:

CCT.exe -? OR CCT.exe -h

Arguments:

No Arguments.

2.2.2 gcc – Get Clock Capabilities

Usage: This command displays ICC FW version info and ICC HW SKU information.

Syntax:

CCT.exe -gcc

Arguments:

No arguments.

2.2.3 gp – Get ICC Profile

Usage: This command displays detail related to the ICC profile, OEM Profile Setup Parameters, Runtime profile selection, and total number of available profiles from Full SPI image flashed on the platform.

Syntax:

CCT.exe -gp

Arguments:

No arguments



2.2.4 sp – Set ICC Profile

Usage: This command allows user to set the ICC profile to the number specified in the profile number argument. Up to 16 ICC profiles can be added via Intel® Flash Image Tool but only one profile out of all can be used as a boot profile. This command allows user to change ICC boot profile. After this command is used, Intel® CSE FW executes global reset to apply the change.

This command will not work after the BIOS sends the End of Post Intel® MEI message.

Syntax:

```
CCT.exe -sp -p [Index]
```

Arguments:

[Index] - requested ICC profile number.

2.2.5 gc – Get ICC Clock Settings

Usage: This command displays details of current ICC clock settings.

Syntax:

```
CCT.exe -gc -clock [Clock ID] -get-type [setting type]
```

Arguments:

- [Clock ID] – specifies clock for which ICC settings should be displayed.

Valid Argument Options for Clock ID	Usage
BCLK	Displays ICC settings for BCLK clock

- [Setting type] – specifies which ICC record to be used to display clock settings mentioned by clock ID.

Valid Argument Options for Clock ID	Usage
Current	Displays current ICC settings for clock id mentioned by clock ID argument
Permanent	Displays Persistent ICC settings for clock id mentioned by clock ID argument

2.2.6 sc – Set ICC Clock Settings

Usage: This command allows user to set ICC Clock

**Syntax:**

CCT.exe -sc -clock [clock ID] -freq [frequency] -ssc [ssc percent] -set-type [setting type]

Arguments:

- [clock ID] – specifies clock for which ICC settings should be changed.

Valid Argument Options for Clockid	Usage
BCLK	Sets ICC settings for BCLK clock

- [Frequency] – User defined frequency.
- [SSC percent] – Specifies percentage of the frequency range to spread.
Example: a value of 50 indicates 0.50%.
- [setting type] –Set Clock Setting Type, Permanent or Dynamic

2.2.7 rm – read from MPHY table

Usage: This command is used to read data from Chipset Init file.

This command will not work after the BIOS sends the End of Post Intel® MEI message

Syntax:

CCT.exe -rm -offset [start register offset] -length [end register offset] -file [mphy file name]

Arguments:

[start register offset] – Start address

[end register offset] – End address

[mphy file name]- Output file name

2.2.8 sm – Set mPHY Settings

Usage: This command will update the mphy table with content from input.bin file

This command will not work after the BIOS sends the End of Post Intel® MEI message

Syntax:

CCT.exe -sm -file [inputFile.bin]

Arguments:

[inputFile.bin]- In



2.2.9 Rvrt - Revert FW to defaults

Usage: This command reverts FW to its default configuration.

Syntax:

```
CCT.exe rvrt -type [SettingType]
```

Arguments:

[settingType] – nominal , factory



2.3 Examples – How to Use CCT Commands

Below are the examples on how to use CCT tool supported commands. Please note that arguments used in the below examples may differ in value depending on the platform the commands executed on.

2.3.1 Help Menu

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -?
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

cct.exe [-exp] [-h|?] [help] [-ver] [-VERBOSE] [-PAGE] [-sp|set-profile]
        [-gp|get-profile] [-gr|get-register] [-gcc|get-clock-capabilities]
        [-rm|read-from-mphy] [-gc|get-clock-settings] [-sc|set-clock-settings]
        [-gm|get-mphy] [-sm|set-mphy] [-rvrt|revert-to-default] [-ep|endpoint]
        [-offset] [-length] [-p|profile] [-clock] [-freq|frequency] [-ssc]
        [-forcepowerflow|force-power-flow] [-settodefault|set-to-default] [-get-type] [-set-type] [-file]
        [-type] [-dump]

-exp [arg_name]                Display example usage of this tool
-h|?|help                      Display help screen
-ver                           Display version information
-VERBOSE [filename]           Display the debug information of the tool
-PAGE                          Pause after each screenful of information
-sp|set-profile                Set ICC Profile
-gp|get-profile                Get ICC Profile
-gr|get-register               Get ICC Register
-gcc|get-clock-capabilities    Get ICC Clocks Capabilities
-rm|read-from-mphy             Read data from Chipset Init file
-gc|get-clock-settings         Get ICC Clock Settings
-sc|set-clock-settings         Set ICC Clock Settings
-gm|get-mphy                   Get Chipset Init Settings
-sm|set-mphy                   Set Chipset Init Settings
-rvrt|revert-to-default        Revert FW to defaults
-ep|endpoint <endpoint>       Side Band End Point
-offset <offset>               Register Offset
-length <offset>               Register Offset
-p|profile <profile index>     ICC Profile ID
-clock <clock: bclk|pcie>      Clock
-freq|frequency <frequency>   Clock Frequency
-ssc <ssc>                     Clock SSC Percent
-forcepowerflow|force-power-flow Force Power Flow
-settodefault|set-to-default   Set To Default
-get-type <setting type: current|permanent|oem> Get Clock Setting Type
-set-type <setting type: permanent|dynamic>      Set Clock Setting Type
-file <mphy File name>         Chipsetinit File
-type <setting type: nominal|factory>             Chipsetinit File
-dump                          Dump Mphy Hex Data To Screen
PS C:\Users\Administrator\Desktop\CCT 1058>
```

2.3.2 Get Clock Capabilities

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -gcc
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Capabilities:
    HW Product Family: ICP-LP
    FW Version:        13.0.0.1059
    ICC HW SKU:         ENHANCED

Status: SUCCESS

PS C:\Users\Administrator\Desktop\CCT 1058>
```



2.3.3 Get ICC Profiles

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -gp
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Profile:
    Number Of Profiles:          1
    Failsafe Profile ID:         0
    Profile Changeable at Runtime: Yes
    Current Profile ID:          0
    Profile0 Name: Profile 0, Base: Adaptive

Status: SUCCESS
```

2.3.4 Set ICC Profile.

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -sp -p 0
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Set Profile. Profile ID: 0

Status: SUCCESS
```

2.3.5 Get ICC Clock Settings

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -gc -clock bclk -get-type current
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Clock Settings:
    Clock:          BCLK
    Setting Type:    CURRENT
    Frequency:       100000000 HZ
    User Frequency:  100000000 HZ
    Max Frequency:   100000000 HZ
    Min Frequency:   97500000 HZ
    SSC Mode:        2
    SSC Percent:     43
    Max SSC %:       50
    Current Flags:    0x00000000
    Power Cycle Pending: No
    Support Flags:    0x0010
    Support Down Spread: No
    Support Up Spread: No
    Support Center Spread: No
    Support Change Allowed: Yes

Status: SUCCESS
```



2.3.6 Set ICC Clock Settings

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -sc -clock bclk -freq 99800000 -ssc 3 -set-type permanent
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

Set Clock Settings:      Clock:      BCLK
Setting Type:    PERMANENT
Frequency:      (99800000) | SSC: 3
Set To Default: No | Force Power Flow: No
Frequency: 99800000

Status: SUCCESS

PS C:\Users\Administrator\Desktop\CCT 1058>
```

2.3.7 Read From mPHY Table

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -rm -offset 0x0 -length 0x0ff -file txt.bin
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Read From Mphy:      Offset:      0x0
Length:      0xFF
Output File:      txt.bin
Total Mphy Table Size: 5728 | Bytes Read: 255
```

2.3.8 Set mPHY Settings

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -sm -file .\1.bin
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

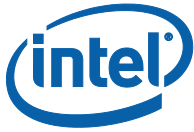
ICC Set Mphy Settings. Input File:.\1.bin
Status: SUCCESS
```

2.3.9 Revert FW to Default

```
PS C:\Users\Administrator\Desktop\CCT 1058> .\CCT.exe -rvrt -type factory
Intel (R) CCT Version: 13.0.0.1058
Copyright (C) 2005 - 2019, Intel Corporation. All rights reserved.

ICC Revert To Default. Revert Type: FACTORY
Status: SUCCESS

PS C:\Users\Administrator\Desktop\CCT 1058>
```



2.4 Error and Status Messages

2.4.1 Clock Commander Tool Error and Status Messages

When a command is executed the Clock Commander Tool will display status and error messages to indicate the result of the operations.



3 Intel® ICCS SDK

3.1 Intel® ICCS SDK

Intel® Integrated Clock Controller Services (Intel® ICCS) provides a lot of flexibility for OS applications. To ease OS application development and to avoid erratic programming of ICC, Intel provides an ICC Control Library and abstracts ICC hardware from clock tuning applications such as BIOS.

3.2 Intel® ICCS Control Library SDK Features

ICC HW is only accessible to Intel® Management Engine (Intel® ME) and is accessible indirectly to the host software through a set of Intel® Management Engine Interface (Intel® MEI) APIs that is known to the Intel® Integrated Clock Controller Service. Intel does not expose this Intel MEI APIs and does not recommend OS applications to use them to keep platform stability.

Example of application that may call Intel® Integrated Clock Controller Service:

- Intel® Extreme Tuning Utility (Intel® XTU). This application can overclock or underclock platform BCLK (Processor clock).

3.3 Intel® Integrated Clock Controller Service Data Structures

Intel® Integrated Clock Controller Service provides a simplified ICC data structures and APIs for clock manipulation. The new data structure has significantly been reduced and simplified compared to previous generation of ICC control library. The ICC data structure is described in this section.

Table 2. Intel® Integrated Clock Controller Service API Data Structures

Name	Type	Description
ICC_HECI_CLOCK_ID	Enum	Defines the clock id for applicable clocks
ErrorCodes	UINT32	Returns error codes from Intel® Integrated Clock Controller Service API calls
ICC_GET_CLOCK_SETTI NGSEx	Struct	Contains the current clock setting
ICC_SET_CLOCK_SETTI NGSEx	Struct	Contains updatable clock setting



3.3.1 ICC_HECI_CLOCK_ID Type

The ICC_HECI_CLOCK_ID data structure provides the applicable clock to be selected with the following structure.

```
typedef enum
{
    ICC_HECI_PCIE_CLOCK_ID = 0,
    ICC_HECI_BCLK_CLOCK_ID = 1,
    ICC_HECI_WMPHY_CLOCK_ID = 2
}ICC_HECI_CLOCK_ID;
```

Table 3. Intel ICC_HECI_CLOCK_ID type

Name	Description
ICC_HECI_PCIE_CLOCK_ID	PCIe Clock (CPUBCLK Signal to CPU)
ICC_HECI_BCLK_CLOCK_ID	BCLK Clock (CPUBCLK Signal to CPU)
ICC_HECI_WMPHY_CLOCK_ID	White Mountain PLL

3.3.2 ErrorCodes Type

The ErrorCodes data structure provides description of the return error code from the Intel® Integrated Clock Controller Service.

The returned values are represented in UINT32 the following function is required to parse the values into char type:

```
const char* GetErrorStringByCode(const UINT32 errorCode);
```

3.3.3 ICC_GET_CLOCK_SETTINGSEx Type

The ICC_GET_CLOCK_SETTINGSEx structure provides all the clock settings details as the following:

```
typedef struct _ICC_GET_CLOCK_SETTINGSEx
{
    UINT32 Frequency;
    UINT32 UserFrequency;
    UINT32 MaxFrequency;
    UINT32 MinFrequency;
    UINT8 SscMode;
    UINT8 SscPercent;
    UINT8 MaxSscPercent;
    UINT16 CurrentFlags;
```



```
    UINT16 SupportFlags;  
} ICC_GET_CLOCK_SETTINGSEx;
```

3.3.4 ICC_SET_CLOCK_SETTINGSEx Type

The ICC_SET_CLOCK_SETTINGSEx structure provides all the updatable clock settings as the following:

```
typedef struct _ICC_SET_CLOCK_SETTINGSEx  
{  
    UINT32 UserFrequency;  
    UINT8  SscPercent;    // encoding example: 1.28% -> SSC_SPREAD value is  
128  
    BOOL   SetToDefault;  
    BOOL   ForcePowerFlow;  
} ICC_SET_CLOCK_SETTINGSEx;
```



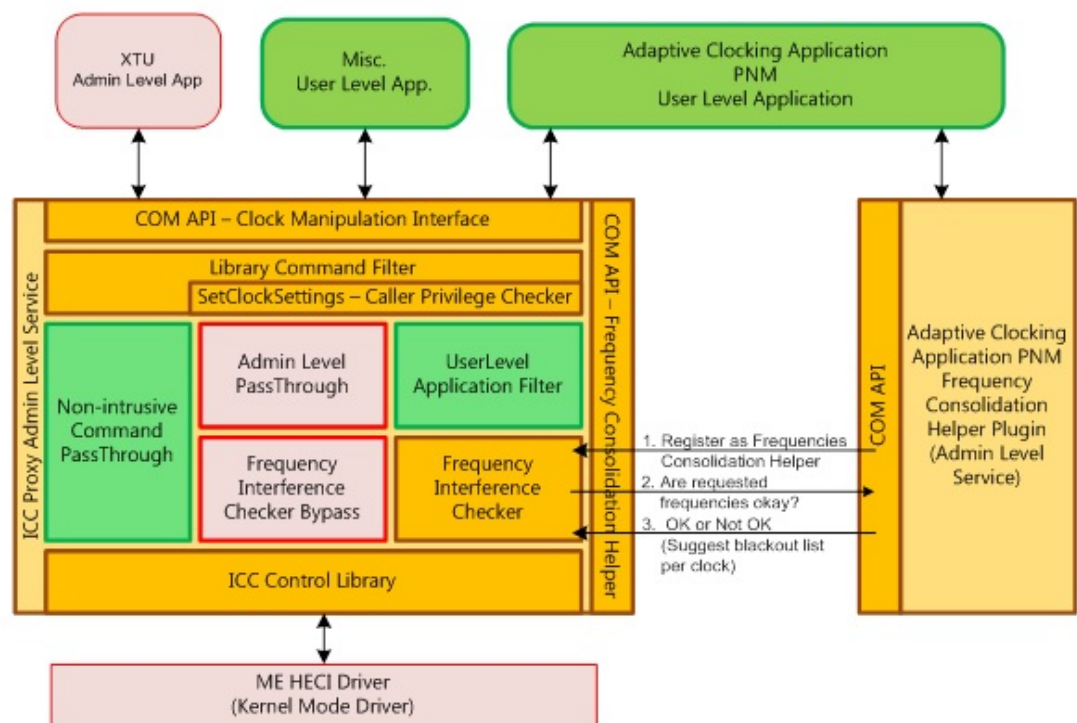
3.4 Intel® Integrated Clock Controller Service API – Clock Manipulation Interface

The Intel® Integrated Clock Controller Service provides a new set of simplified APIs in form of COM. The API exposed by the Intel® Integrated Clock Controller Service is used for communication between Intel® Integrated Clock Controller Service and client applications.

It is assumed that an application would only configure ICC clocks that “belong” to it. There is no provision in the library to lock a specific clock from a specific application.

Each API in the following section provides detail about Input and Output parameters for each API, as well as function prototype. For this guide, example in the following programming language will be provided: IDL and C++. Other programming languages are supported by the API, but an example will not be provided by this SDK user guide.

Note: Please note that Intel MEI Driver installation is required. Figure 1. Intel® Integrated Clock Controller Service Architecture





3.4.1 Get Current Clock Settings Wrapper

Through this function, the host application can get the runtime settings of the ICC clock identified by ICC_HECI_PCIE_CLOCK_ID parameter. The request returns the runtime settings of the clock.

Definition:

```
UINT32 IccLibGetCurrentClockSettingsWrapper(const ICC_HECI_CLOCK_ID
clockId, ICC_GET_CLOCK_SETTINGSEX * const clockSettings);
```

Table 4. Get Clock Runtime Settings Parameters

Type	Field	Description
Input	clockId	Clock identifier
Output	clockSettings	Runtime settings of the clock. Meaningful only if successful status is received.

3.4.2 Set Current Clock Settings Wrapper

Host application calls this function to change the settings of one of the ICC clocks.

For host application (with Admin level) call, the request will be verified against the ICC Clock Range Definition Record.

For host application (with user level) call, the request will be verified against the ICC enhanced SKU Clock Range definition.

Note: If the requested frequency is not supported by the HW, it will be automatically rounded by Intel® Integrated Clock Controller Service to the nearest valid frequency. If there are two nearest valid frequencies (up and down), the lower value will be chosen.

Definition:

```
UINT32 IccLibSetCurrentClockSettingsWrapper(const ICC_HECI_CLOCK_ID
clockId, ICC_SET_CLOCK_SETTINGSEX * clockSettings);
```

Table 5. Set Clock Runtime Settings Parameters

Type	Field	Description
Input	clockId	Clock identifier
Input	clockSettings	Runtime configuration for the clock.



4 **Intel® Watchdog Timer Driver (Intel® WDT Driver)**

Intel® Watchdog Timer Driver is a kernel driver (KMDF) that responsible for claiming the ACPI Intel WDT Device which is created by the BIOS and attaching to it. It is responsible for directly communicating with the hardware via I/O and should be aware of the hardware interface details necessary to communicate with the PCH. The Driver supports the following OS:

- Windows* 7 (32-bit and 64-bit Editions)
- Windows* 8 (32-bit and 64-bit Editions)
- Windows* 8.1 (32-bit and 64-bit Editions)
- Windows* Threshold (32-bit and 64-bit Editions)
- Windows* Threshold2 (32-bit and 64-bit Editions)
- Windows* 10 RS1(32-bit and 64-bit Editions)

4.1 **Communication with the Driver**

Communication with the Intel WDT driver is like any other kernel driver: When it's needed - use the standard Device/Files API (CreateFile, ReadFile, WriteFile, DeviceIoControl).

The device is identified by the following GUID (that is defined in ICCWDT_Interface.h):

```
// {C2E625A9-8693-4dea-BAC4-B15CA98F9EE9}  
  
DEFINE_GUID(GUID_DEVINTERFACE_ICCWDT,  
0xc2e625a9, 0x8693, 0x4dea, 0xba, 0xc4, 0xb1, 0x5c, 0xa9, 0x8f, 0x9e, 0xe9);
```

Connection example code can be found in Microsoft* Windows* Driver Kit
<http://www.microsoft.com/whdc/devtools/wdk/default.aspx>.

4.2 **Device Functionally - IOCTLs**

The following list describes the device IOCTLs:

**Table 6. Device Functionality - IOCTLs**

IOCTL	Description
ICCWDT_AQUIRE_WDT	Acquire Intel Watchdog Timer Driver Control
ICCWDT_RELEASE_WDT	Release Intel Watchdog Timer Driver Control
ICCWDT_RELOADANDSTART_WDT	Reload and Start Intel Watchdog Timer Driver
ICCWDT_STOP_WDT	Stop Intel Watchdog Timer Driver
ICCWDT_GET_SUPPORTED_CD_DATA	Get Supported Countdown Data
ICCWDT_GET_WDT_STATUS	Get Intel Watchdog Timer Driver Status
ICCWDT_START_WDT_ON_NEXT_OSBOOT	Start Intel Watchdog Timer Driver on Next OS Load

4.2.1 Acquire Intel® Watchdog Timer Driver Control

The Acquire Intel Watchdog Timer Driver (Intel® WDT Driver) Control method allows a single application to be able to gain control of the Intel WDT Driver. Due to the timing sensitive nature of the Intel WDT Driver, associated timeouts and reloads, only a single application can be allowed to control the hardware at a time.

Table 7. Acquire Intel® Watchdog Timer Driver Control

IOCTL Name	ICCWDT_AQUIRE_WDT
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x800, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)
Input	None
Output	ULONG Key – Watchdog Acquiring key
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR – Intel Watchdog Timer Driver already been acquired
	HARDWARE_LOCKED_ERROR – Intel Watchdog Timer Driver hardware registers are lock, can't be modified.
	FATAL_ERROR - Unexpected error – Fatal Error.

4.2.2 Intel® Watchdog Timer Driver Control

The Release Intel Watchdog Timer Driver Control IOCTL is used to release control of the Intel WDT Driver from the application that currently controls it. This



communicates to the driver that the current application is done with its needs for the Intel WDT Driver and that it can be allocated to another application if necessary.

Table 8. Release Intel® Watchdog Timer Driver Control

IOCTL Name	ICCWDT_RELEASE_WDT
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x802, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)
Input	ULONG Key – Intel Watchdog Timer Driver Acquiring key, retrieved from ICCWDT_AQUIRE_WDT
Output	None
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR – Intel Watchdog Timer Driver already been released
	FATAL_ERROR - Unexpected error – Fatal Error.

4.2.3 Reload and Start Intel® Watchdog Timer Driver

The Reload and Start Intel Watchdog Timer Driver IOCTL is responsible for loading the appropriate countdown timer into the Intel WDT Driver, reloading the countdown timer, and enabling the Intel WDT Driver countdown. When the Intel WDT Driver counter reaches 0 then the Intel WDT Driver will signal a Global Reset. If this happens it is assumed that the platform is no longer responsive, and the reset action is required to return the platform to a usable state. The countdown value can vary between 1 sec and 1024 sec (~17Min).

**Table 9. Reload and Start Intel® Watchdog Timer Driver**

IOCTL Name	ICCWDT_RELOADANDSTART_WDT
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x803, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)
Input	<pre>typedef struct _ICCWDT_RELOADANDSTART_DATA{ ULONG Key; UINT16 CountdownVal; }ICCWDT_RELOADANDSTART_DATA, *PICCWDT_RELOADANDSTART_DATA;</pre> <p>ULONG Key – Intel Watchdog Timer Driver Acquiring key, retrieved from ICCWDT_AQUIRE_WDT</p> <p>UINT16 CountdownVal – Intel Watchdog Timer Driver Countdown Value. Can be set between 1d and 1024d</p>
Output	None
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR - Intel Watchdog Timer Driver was not been acquired or wrong acquiring key.
	INVALID_PARAMETER_ERROR - Bad Countdown value.
	FATAL_ERROR - Unexpected error – Fatal Error.

4.2.4 Stop Intel Watchdog Timer Driver

The Stop Intel® Watchdog Timer Driver IOCTL is responsible for stopping the Intel WDT Driver countdown. When the Intel WDT Driver counter has been stopped it will never signal a Global Reset. This method does not infer that a reload of the countdown timer has occurred. It merely disabled the Global Reset output. To restart the Intel WDT Driver, the user can just call the Reload and Start Intel Watchdog Timer Driver IOCTL.



Table 10. Stop Intel® Watchdog Timer Driver

IOCTL Name	ICCWDT_STOP_WDT
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x804, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)
Input	ULONG Key – Intel Watchdog Timer Driver Acquiring key, retrieved from ICCWDT_AQUIRE_WDT
Output	None
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR – Intel Watchdog Timer Driver was not being acquired or wrong acquiring key.
	FATAL_ERROR - Unexpected error – Fatal Error.

4.2.5 Get Supported Countdown Data

The Get Supported Countdown Data IOCTL is responsible for providing the calling application with the information necessary to understand what timeout settings are available for the Intel WDT Driver. This IOCTL can be called at any point while the caller has control over the Intel WDT Driver.

**Table 11. Get Supported Countdown Data**

IOCTL Name	ICCWDT_GET_SUPPORTED_CD_DATA
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x805, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)
Input	ULONG Key – Intel Watchdog Timer Driver Acquiring key, retrieved from ICCWDT_AQUIRE_WDT
Output	<pre>typedef struct _ICCWDT_SUPPORTED_CD_DATA{ UINT16 MinimumTimeoutPeriod; UINT16 MaximumTimeoutPeriod; UINT16 TimeoutResolution; }ICCWDT_SUPPORTED_CD_DATA, *PICCWDT_SUPPORTED_CD_DATA;</pre> <p> UINT16 MinimumTimeoutPeriod - Minimum Intel Watchdog Timer Driver Timeout Period UINT16 MaximumTimeoutPeriod - Maximum Intel Watchdog Timer Driver Timeout Period UINT16 TimeoutResolution – Intel Watchdog Timer Driver Timeout Resolution </p> <p>Note: all values are in seconds.</p>
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR – Intel Watchdog Timer Driver was not acquired or wrong acquiring key.
	FATAL_ERROR - Unexpected error – Fatal Error.

4.2.6 Get Intel® Watchdog Timer Driver Status

This IOCTL is used to report the status of the previous boot to the application calling the Intel WDT Driver. The intent is to let the controlling application know whether the previous boot had resulted in a failed POST. This includes either an Intel WDT Driver Timeout or an unexpected reboot while the Intel WDT Driver was running. Either situation will result in a timeout and communication of that failure through this method.



Table 12. Get Intel® Watchdog Timer Driver Status

IOCTL Name	ICCWDT_GET_WDT_STATUS						
IOCTL Definition	CTL_CODE(FILE_DEVICE_ACPI, 0x806, METHOD_BUFFERED, FILE_READ_ACCESS FILE_WRITE_ACCESS)						
Input	ULONG Key – Watchdog Acquiring key, retrieved from ICCWDT_AQUIRE_WDT						
Output	<pre>typedef enum _ICCWDT_TIMER_STATUS_TYPE { FAIL, PASS }ICCWDT_TIMER_STATUS_TYPE;</pre> <pre>typedef enum _ICCWDT_TIMER_STATE_TYPE { RUNNING, STOPPED }ICCWDT_TIMER_STATE_TYPE;</pre> <pre>typedef struct _ICCWDT_GET_WDT_STATUS_DATA{ ICCWDT_TIMER_STATUS_ENUM WDTTimerStatus; ICCWDT_TIMER_STATE_TYPE WDTTimerState; UINT16 WDTCountdownPeriod; }ICCWDT_GET_WDT_STATUS_DATA, *PICCWDGET_WDT_STATUS_DATA;</pre> <p>ICCWDT_TIMER_STATUS_TYPE WDTTimerStatus - Watchdog Timer Status ICCWDT_TIMER_STATE_TYPE WDTTimerState - Watchdog Timer State UINT16 WDTCountdownPeriod - Watchdog Timer Countdown Period</p> <table><tr><th>ICC WDT STATUS</th><th>Description</th></tr><tr><td>Fail</td><td><ul style="list-style-type: none">• Previous boot occurred during Intel WDT Driver was running• Intel WDT Driver expired</td></tr><tr><td>Pass</td><td><ul style="list-style-type: none">• Previous boot occurred when Intel WDT Driver was stopped</td></tr></table>	ICC WDT STATUS	Description	Fail	<ul style="list-style-type: none">• Previous boot occurred during Intel WDT Driver was running• Intel WDT Driver expired	Pass	<ul style="list-style-type: none">• Previous boot occurred when Intel WDT Driver was stopped
ICC WDT STATUS	Description						
Fail	<ul style="list-style-type: none">• Previous boot occurred during Intel WDT Driver was running• Intel WDT Driver expired						
Pass	<ul style="list-style-type: none">• Previous boot occurred when Intel WDT Driver was stopped						



	ICC WDT STATE	Description
	Running	<ul style="list-style-type: none"> Intel WDT Driver is running
	Stopped	<ul style="list-style-type: none"> Intel WDT Driver is stopped
Error Code	SUCCESS - No Error – Successful	
	ACCESS_DENIED_ERROR - Watchdog Timer was not acquired or wrong acquiring key.	
	FATAL_ERROR - Unexpected error – Fatal Error.	

4.2.7 Start Intel® Watchdog Timer Driver on Next OS Load

This IOCTL is used to communicate to the BIOS that on the next boot attempt, the Intel WDT Driver should be turned on after the POST process has been completed. This allows for instability coverage after POST has completed, yet before drivers can be loaded by the OS. It is a mechanism that allows for automatically recovering from system instability issues while applying a setting that requires a reboot. The given value is between 0 and 1008. 0 is for disabling Starting Intel Watchdog Timer Driver on Next OS Load. The values will be rounded to 16sec intervals (e.g. 1-16 will be set to 16 secs; 17-32 will be set to 32sec etc....)

It is required that any application which requests the Intel WDT Driver be started during the next OS load must automatically load during the next OS boot and call the Check Intel Watchdog Timer Driver Status IOCTL and handle the result appropriately.



Table 13. Start Intel® Watchdog Timer Driver on Next OS Load

IOCTL Name	ICCWDT_START_WDT_ON_NEXT_OSBOOT
IOCTL Definition	<i>CTL_CODE</i> (<i>FILE_DEVICE_ACPI</i> , 0x807, <i>METHOD_BUFFERED</i> , <i>FILE_READ_ACCESS</i> <i>FILE_WRITE_ACCESS</i>)
Input	<pre>typedef struct _ICCWDT_START_WDT_ON_NEXT_OSBOOT_DATA{ ULONG Key; UINT16 TimeoutValue; }ICCWDT_START_WDT_ON_NEXT_OSBOOT_DATA, PICCWD T_START_WDT_ON_NEXT_OSBOOT_DATA; ULONG Key - Watchdog Acquiring key, retrieved from ICCWDT_AQUIRE_WDT UINT16 TimeoutValue - Watchdog Timer Timeout Value</pre>
Output	None
Error Code	SUCCESS - No Error – Successful
	ACCESS_DENIED_ERROR – Watchdog Timer was not acquired or wrong acquiring key.
	INVALID_PARAMETER_ERROR - Invalid Countdown Value.
	COUNTDOWN_ERROR - Enable to Start Intel WDT Driver ON Next Load.
	FATAL_ERROR - Unexpected error – Fatal Error.

4.3 Driver Access

By default, the access to the driver is limited for “Local System” accounts; this means that only Services will be able to access the driver. For development purposes there is a version of the driver called “Debug/Development Mode” this driver has all user access to the driver after installation.

Note: Intel Watchdog Timer Driver Debug Driver will reload and start the timer every 0.5 seconds after the first ReloadAndStart command from the User.

Note: Development’ mode driver should only be used in development, and not in production.

4.4 Usage Flows Example

Examples below use Pseudo Code.



4.4.1 Normal Flow

```
#include <windows.h>
#include "ICCWDT_Interface.h"
...
ULONG key = 0;

void MyIccWdtThread()
{
    // Boilerplate code from MSFT to getting the Device Path from Device Interface.
    string devicePath = GetDevicePath(GUID_DEVINTERFACE_ICCWDT);

    HANDLE deviceHandle = CreateFile(devicePath, ...);

    // Acquire Timer
    DeviceControl(deviceHandle, ICCWDT_AQUIRE_WDT, NULL, key);

    while(!MyApp_Stop)
    {
        ICCWDT_RELOADANDSTART_DATA data;
        data.key = key;
        data.CountdownVal = MY_WDT_VALUE;

        // Start the Timer
        DeviceControl(deviceHandle, ICCWDT_RELOADANDSTART_WDT, data, NULL);

        // Sleep for Time < ICC WDT Timer Value (MY_WDT_VALUE)
        Sleep(MY_WDT_SLEEP)
    }

    // Stop Timer
    DeviceControl(deviceHandle, ICCWDT_STOP_WDT, key, NULL);

    // Release Timer
    DeviceControl(deviceHandle, ICCWDT_RELEASE_WDT, key, NULL);

    CloseHandle(deviceHandle);
}
```



4.4.2 Start on Next Boot Flow

```
#include <windows.h>
#include "ICCWDT_Interface.h"
...

ULONG key = 0;

void StartOnNextBoot()
{
    // Boilerplate code from MSFT to getting the Device Path from Device Interface.
    string devicePath = GetDevicePath(GUID_DEVINTERFACE_ICCWDT);

    HANDLE deviceHandle = CreateFile(devicePath, ...);

    ULONG key;

    // Acquire Timer (if not already acquired)
    DeviceControl(deviceHandle, ICCWDT_AQUIRE_WDT, NULL, key);

    ICCWDT_START_WDT_ON_NEXT_OSBOOT_DATA data;
    data.key = key;
    data.TimeoutValue= MY_WDT_VALUE;

    // Start the Timer
    DeviceControl(deviceHandle, ICCWDT_START_WDT_ON_NEXT_OSBOOT, data, NULL);

    // Release Timer (if not releasing in another place)
    DeviceControl(deviceHandle, ICCWDT_RELEASE_WDT ,key, NULL);

    CloseHandle(deviceHandle);

    // Reboot System or prompt the user to reboot.
    RebootSystem();
}
```