



Cadence Driver API Usage Guide

Product Version 1.0

April 2020

© 1996-2020 Cadence Design Systems, Inc. All rights reserved.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This document is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this document, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this document may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This document contains the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only in accordance with, a written agreement between Cadence and its customer.

Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this document subject to the following conditions:

1. This document may not be modified in any way.
2. Any authorized copy of this document or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
3. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF CADENCE. EXCEPT AS MAY BE EXPLICITLY SET FORTH IN A WRITTEN AGREEMENT BETWEEN CADENCE AND ITS CUSTOMER, CADENCE DOES NOT MAKE, AND EXPRESSLY DISCLAIMS, ANY REPRESENTATIONS OR WARRANTIES AS TO THE COMPLETENESS, ACCURACY OR USEFULNESS OF THE INFORMATION CONTAINED IN THIS DOCUMENT. CADENCE DOES NOT WARRANT THAT USE OF SUCH INFORMATION WILL NOT INFRINGE ANY THIRD PARTY RIGHTS, AND CADENCE DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. CADENCE DOES NOT ASSUME ANY LIABILITY FOR DAMAGES OR COSTS OF ANY KIND THAT MAY RESULT FROM USE OF SUCH INFORMATION. CADENCE CUSTOMER HAS COMPLETE CONTROL AND FINAL DECISION-MAKING AUTHORITY OVER ALL ASPECTS OF THE DEVELOPMENT, MANUFACTURE, SALE AND USE OF CUSTOMER'S PRODUCT, INCLUDING, BUT NOT LIMITED TO, ALL DECISIONS WITH REGARD TO DESIGN, PRODUCTION, TESTING, ASSEMBLY, QUALIFICATION, CERTIFICATION, INTEGRATION OF CADENCE PRODUCTS, INSTRUCTIONS FOR USE, LABELING AND DISTRIBUTION, AND CADENCE EXPRESSLY DISAVOWS ANY RESPONSIBILITY WITH REGARD TO ANY SUCH DECISIONS REGARDING CUSTOMER'S PRODUCT.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227- 14 and DFAR252.227-7013 et seq. or its successor.

1. API Usage Model

1.1. Introduction

The purpose of this document is to describe the two different usage models for the Cadence Core Driver API (Application Programming Interface). The Core Driver is the Cadence Software component that provides a programming abstraction to the Cadence IP.

The Cadence Core Driver provides two API styles:

- **Object-like API:**
With this method, the first step is to call a statically linked function which will return a C struct containing pointers to each API function. The user application then uses the appropriate pointer from this struct to call API functions.
- **Function Prototypes API:**
With this method, function prototypes for the API are defined in a header file. The user application can link statically to the necessary API functions, and calls them directly.

The choice of which method to use will depend on the use-cases for the driver.

- The Function Prototypes API method may be most suitable where performance or memory usage is an important factor. Because the API can be statically linked it may be possible to make significant optimizations, e.g. inlining of code, removal of API that are not accessed etc.
- The Object-like API may be most suitable during initial debug or bring-up of hardware. Many of the examples and reference code use this access method.

1.2. Details of the Object-like API method

The user application code must include the header file `IPNAME_obj_if.h` (where `IPNAME` is the appropriate prefix for your IP delivery.) This includes a type definition for a struct containing the function pointers, and the function which will return an instance of this struct e.g.

```
typedef struct IPNAME_OBJ_s
{
    ...
    uint32_t (*drvFunc1)(IPNAME_PrivateData* pD, uintptr_t* pReq);
    uint32_t (*drvFunc2)(IPNAME_PrivateData* pD);
    ...
} IPNAME_OBJ;

extern IPNAME_OBJ *IPNAME_GetInstance(void);
```

To access the APIs, a customer's application must call the `IPNAME_GetInstance()` global function to obtain a pointer to the driver object. Here is an example of how you would specify a call to `drvFunc1()`:

```
#include "IPNAMEobj_if.h"
int main(int argc, char **argv) {
    ...
    IPNAME_OBJ * driver_obj = IPNAME_GetInstance();
    driver_obj->drvFunc1(pD, &memReq);
    ...
}
```

1.3. Details of the Function Prototypes API method

The user application code must include the header file `IPNAME_if.h` (where `IPNAME` is the appropriate prefix for your IP delivery). The file contains prototypes for the driver's API functions, for example:

```
...
uint32_t IPNAME_DrvFunc1(IPNAME_PrivateData* pD, uintptr_t* pReq);
uint32_t IPNAME_DrvFunc2(IPNAME_PrivateData* pD);
...
```

To call the `IPNAME_DrvFunc1()` function, invoke the API as follows:

```
#include "IPNAME_if.h"
int main(int argc, char **argv) {
    ...
    IPNAME_DrvFunc1(pD, &memReq);
    ...
}
```