cadence

Cadence Driver API Usage Guide

Product Version 1.0

April 2020

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

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 wouldspecify a callto 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);
   ...
}
```