

GLOBAL_ADC

Home

Apps

Here is a list of all modules:

- License Terms and Copyright Information
- Abbreviations and Definitions
- Overview
- Architecture Description
- APP Configuration Parameters
- Enumerations
- Data structures
- Methods
- Release History

GLOBAL_ADC

Home

License Terms and Copyright Information

License Terms and Copyright Information

Copyright (c) 2015, Infineon Technologies AG All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

To improve the quality of the software, users are encouraged to share modifications, enhancements or bug fixes with Infineon Technologies AG (dave@infineon.com).

GLOBAL_ADC

Home

Abbreviations and Definitions

Abbreviations and Definitions

Abbreviations:	
DAVE™	Digital Application Virtual Engineer
APP	DAVE™ Application
API	Application Programming Interface
GUI	Graphical User Interface
MCU	Microcontroller Unit
SW	Software
HW	Hardware
LLD	Low Level Driver
IO	Input Output
ADC	Analog to Digital Conversion
VADC	Versatile Analog to Digital Converter

Definitions:	
Singleton	Only single instance of the APP is permitted
Sharable	Resource sharing with other APPs is permitted
initProvider	Provides the initialization routine
Physical connectivity	Hardware inter/intra peripheral (constant) signal connection
Conditional connectivity	Constrained hardware inter/intra peripheral signal connection
Aggregation	Indicates consumption of low level (dependent) DAVE APPs



GLOBAL_ADC

Home

Overview

Overview

GLOBAL_ADC is a basic APP which configures VADC Global Registers. It configures the VADC clocks and other functions.

*The **GLOBAL_ADC** APP provides the following functionalities to configure the VADC peripheral:*

1. Configure the Analog clock.
2. Configure the digital clock (arbitration clock).
3. Startup calibration
4. Configures post calibration for each group.
5. Configures arbiter behaviour for each group.

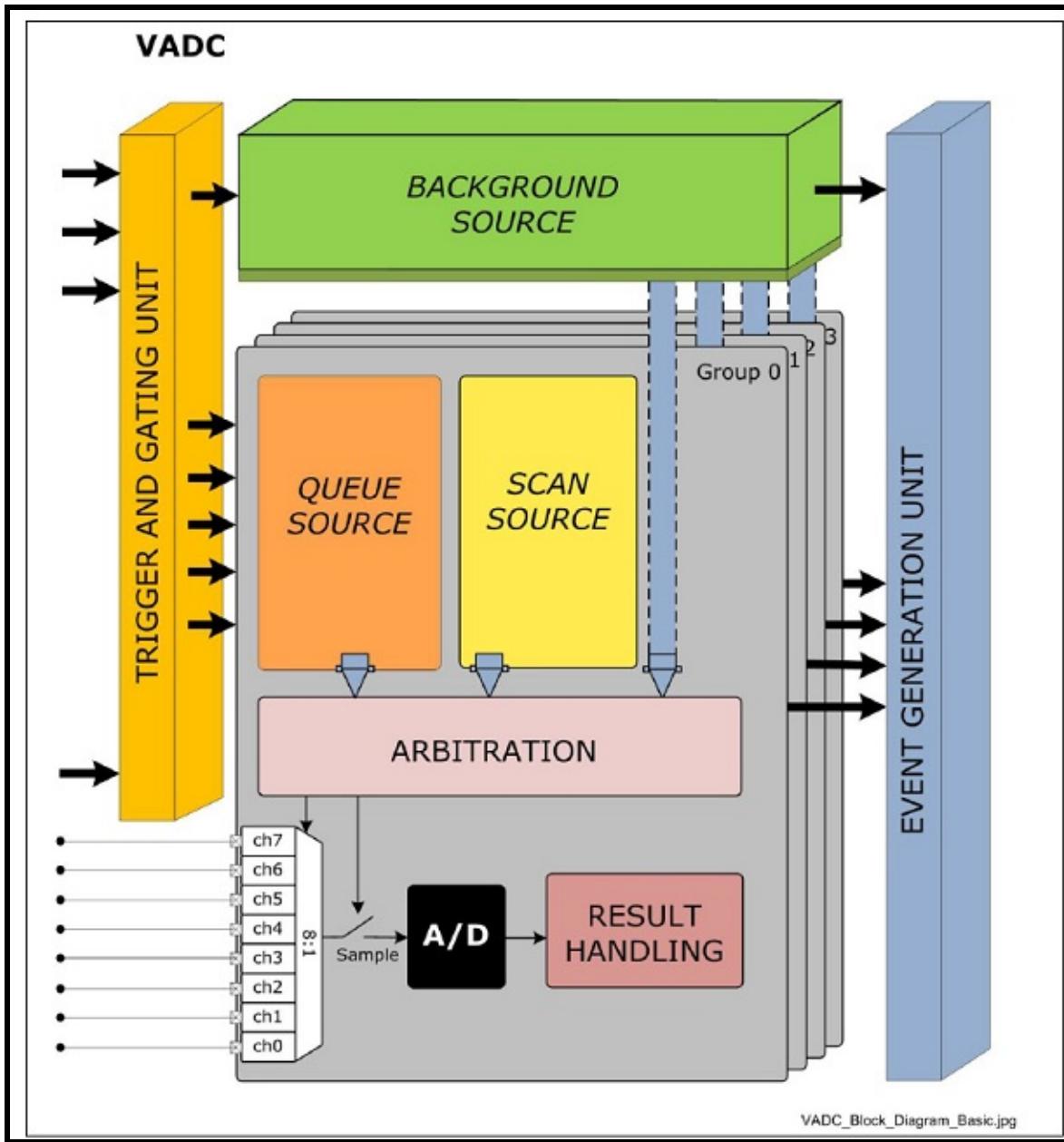


Figure 1 : Overview of VADC peripheral

Figure 1 shows the overview of the VADC peripheral. In this the **GLOBAL_ADC** APP would configure the clock dividers the entire module. It would also do all the necessary initialization for the module.

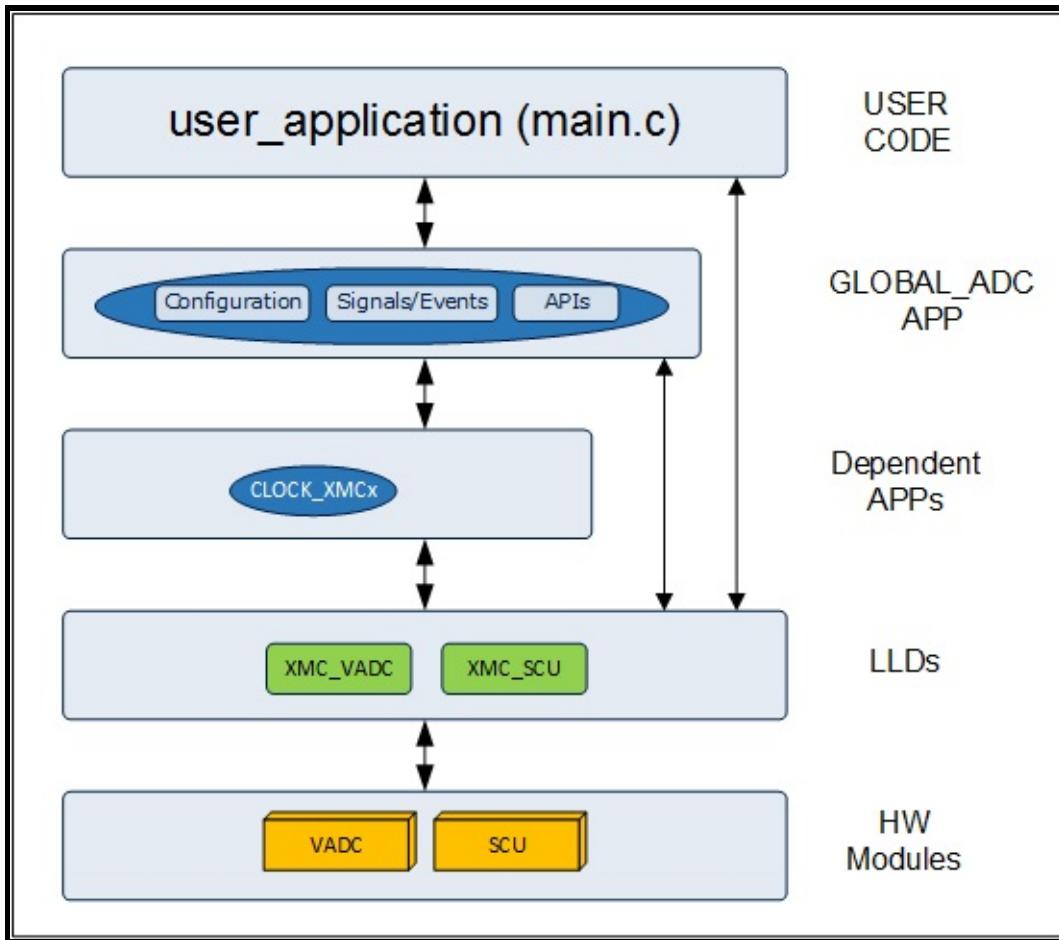


Figure 2 : Hardware and Software connectivity of **GLOBAL_ADC APP**

Figure 2 , shows how the APP is structured in DAVE. XMC controllers provide the VADC module be used for analog to digital conversion. The LLD layer provides abstraction for these hardware modules. The **GLOBAL_ADC APP** uses VADC and SCU LLDs and other dependent APPs like CLOCK_XMCx for the functionality.

Supported Devices

The APP supports below devices:

1. XMC4500 Series
2. XMC4400 Series
3. XMC4300 Series
4. XMC4200 / XMC4100 Series
5. XMC1300 Series
6. XMC1200 Series

7. XMC1100 Series

Reference

1. XMC4500 Reference Manual
 2. XMC4400 Reference Manual
 3. XMC4300 Reference Manual
 4. XMC4200 / XMC4100 Reference Manual
 5. XMC1300 Reference Manual
 6. XMC1200 Reference Manual
 7. XMC1100 Reference Manual
-



GLOBAL_ADC

Home

Architecture Description

Architecture Description

Figure 1 explains the architecture of the APP:
This would pictorially represent the internal workings of the
GLOBAL_ADC. This shows the hardware resources that are
consumed, the depended APPs and the various signals that would be
exported out.

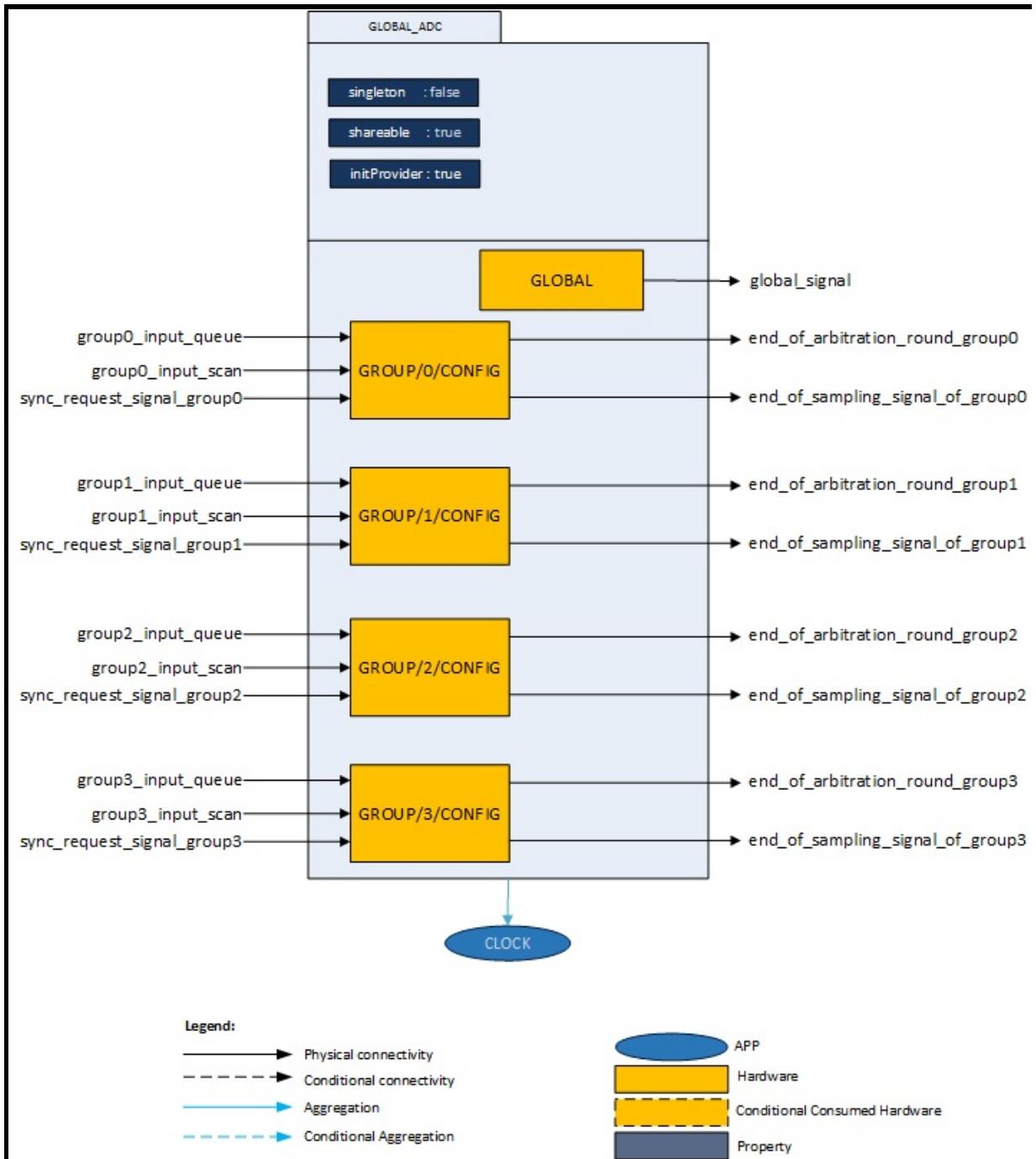


Figure 1 : Architecture of **GLOBAL_ADC APP**

The diagram above represents the internal software architecture of the **GLOBAL_ADC** APP. A **GLOBAL_ADC** APP instance exists in a DAVE-4.0 project with fixed attributes as shown and configures the VADC peripheral. This in addition requires the consumption of the **CLOCK** APPS for its configuration and functioning. The **GLOBAL_ADC**

APP also provides output signals for inter-peripheral connections.

Signals:

The following table presents the signals provided by the APP for connection. It also gives the flexibility to configure and extend the connectivity to other APPs.

Table 1: APP IO signals

Signal Name	Input/Output	Availability	Description
global_signal	Output	Always	Global connection signal: Should connect the API consumer GLOB APP (By default)
end_of_arbitration_round_group0	Output	Always	End of arbitration round signal from group 0: Can be connected to the VAI module's gating edge trigger various sources: An end of arbitration

		round robin arbitration round signal which can also be given to peripheral
end_of_sampling_signal_of_group0	Output	Always End of sampling phase from group 0. Can be connected internally to the VADC input for gating various trigger sources. Generates signal via sampling signal in complete.
end_of_arbitration_round_group1	Output	Always End of arbitration round signal from group 1. Can be connected to the VAI module via the gating output trigger various sources. An end of arbitration round signal is generated.

			arbitrat round r signal w can als given to peripher
end_of_sampling_signal_of_group1	Output	Always	End of samplir phase : from gr Can be connec intern VADC i for gati trigger various sources: Genera signal v samplir signal i comple
end_of_arbitration_round_group2	Output	Always	End of arbitrat round s from gr Can be connec the VAI module gating trigger various sources:

		An end arbitration round signal which can also be given to peripheral
end_of_sampling_signal_of_group2	Output	Always End of sampling phase from group 2. Can be connected internally to VADC for gating trigger various sources. Generates signal via sampling signal if complete.
end_of_arbitration_round_group3	Output	Always End of arbitration round signal from group 3. Can be connected to the VAI module for gating trigger various

			sources: An end-of-sampling signal can also be given to peripherals.
end_of_sampling_signal_of_group3	Output	Always	End of sampling phase from group 3. Can be connected internally to VADC input for gating trigger various sources. Generated signal via sampling signal is complete.
group0_input_queue	Input	Always	Queue selection. Group 0. If a ADC_Q[0] = 1 APP is consumer the group which triggered Queue.

			has to be forced by group-0 connection to be made between ADC_S and APP
group0_input_scan	Input	Always	Scan signal for Group-0. If a slave ADC_S and APP is connected to the group, which then belongs to be forced by group-0 connection to be made between ADC_S and APP
sync_request_signal_group0	Input	Always	Sync request signal for Group-0. If a sync operation is needed by the slave, must be connected to group-0 connection to be made
		Queue	

			selection Group- If a ADC_C APP is consum the gro which t Queue has to l forced group- connec be mac ADC_C APP
group1_input_queue	Input	Always	Scan s for Gro If a ADC_S APP is consum the gro which t belong be forc group- connec be mac ADC_S APP
group1_input_scan	Input	Always	Sync re signal f Group- If a syn operati

sync_request_signal_group1	Input	Always	needed the slave must be to group connection be made
----------------------------	-------	--------	---

group2_input_queue	Input	Always	Queue selection Group- If a ADC_C APP is consumed the gro which t Queue has to b forced group-2 connec be mad ADC_C APP
--------------------	-------	--------	---

group2_input_scan	Input	Always	Scan s for Gro If a ADC_S APP is consumed the gro which t belong be forc group- connec
-------------------	-------	--------	---

			be mac ADC_S APP
sync_request_signal_group2	Input	Always	Sync re signal f Group- If a syn operati needed the slav must be to grou connec be mac
group3_input_queue	Input	Always	Queue selectio Group- If a ADC_C APP is consun the gro which t Queue has to l forced group- connec be mac ADC_C APP
			Scan s for Gro If a ADC_S

group3_input_scan	Input	Always	APP is consuming the group which the slave belongs to. It will be forced to group-3 connection to be made between ADC_S and APP.
sync_request_signal_group3	Input	Always	Sync request signal for Group-3. If a sync operation is needed by the slave, it must be connected to group-3 connection to be made.

GLOBAL_ADC

Home

APP Configuration Parameters

App Configuration Parameters

General Settings Advanced Settings

Clock Settings

Peripheral bus clock [MHz]:	120
Desired analog clock [MHz]:	30
Actual analog clock [MHz]:	30
Digital clock:	fADC ▾
Actual digital clock [MHz]:	120

Enable start up calibration

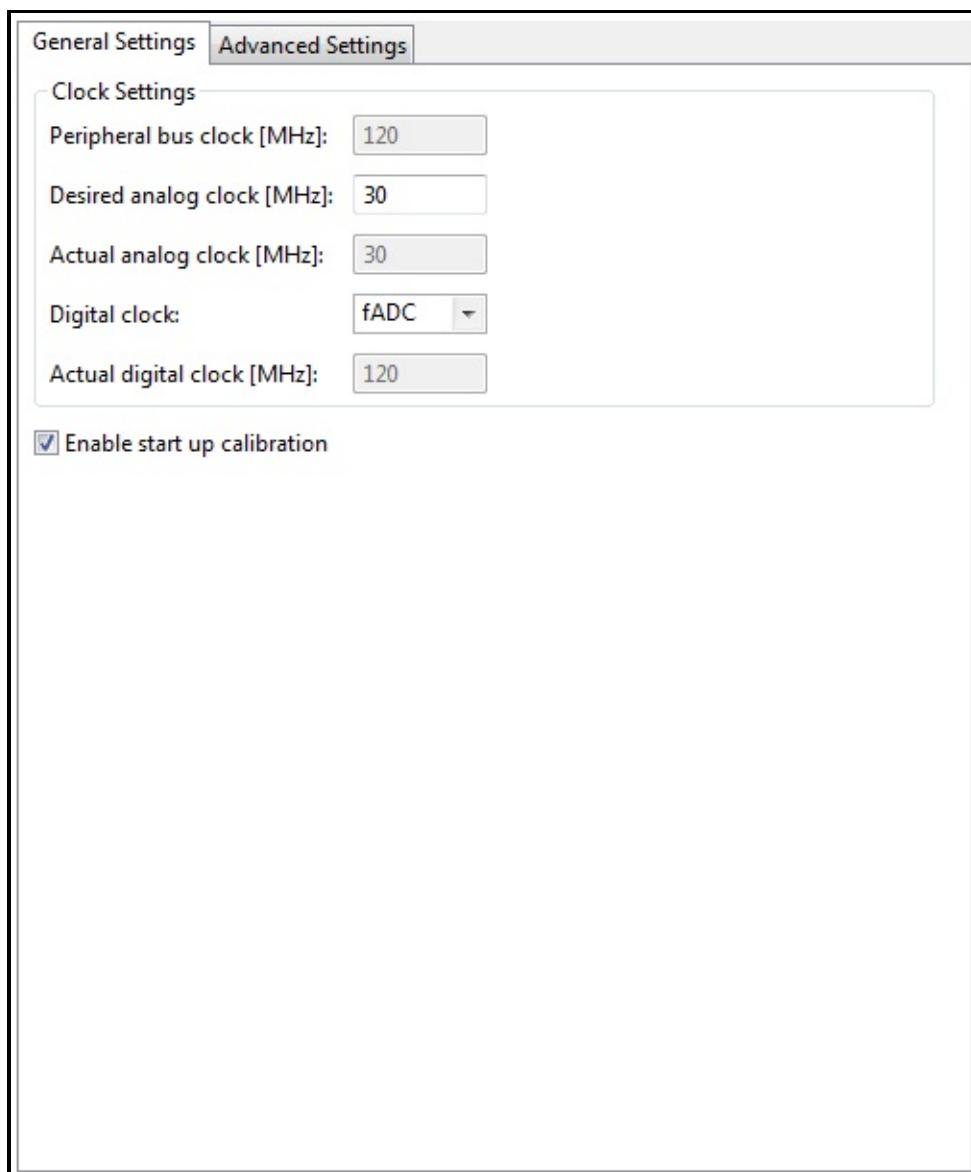


Figure 1: General Settings

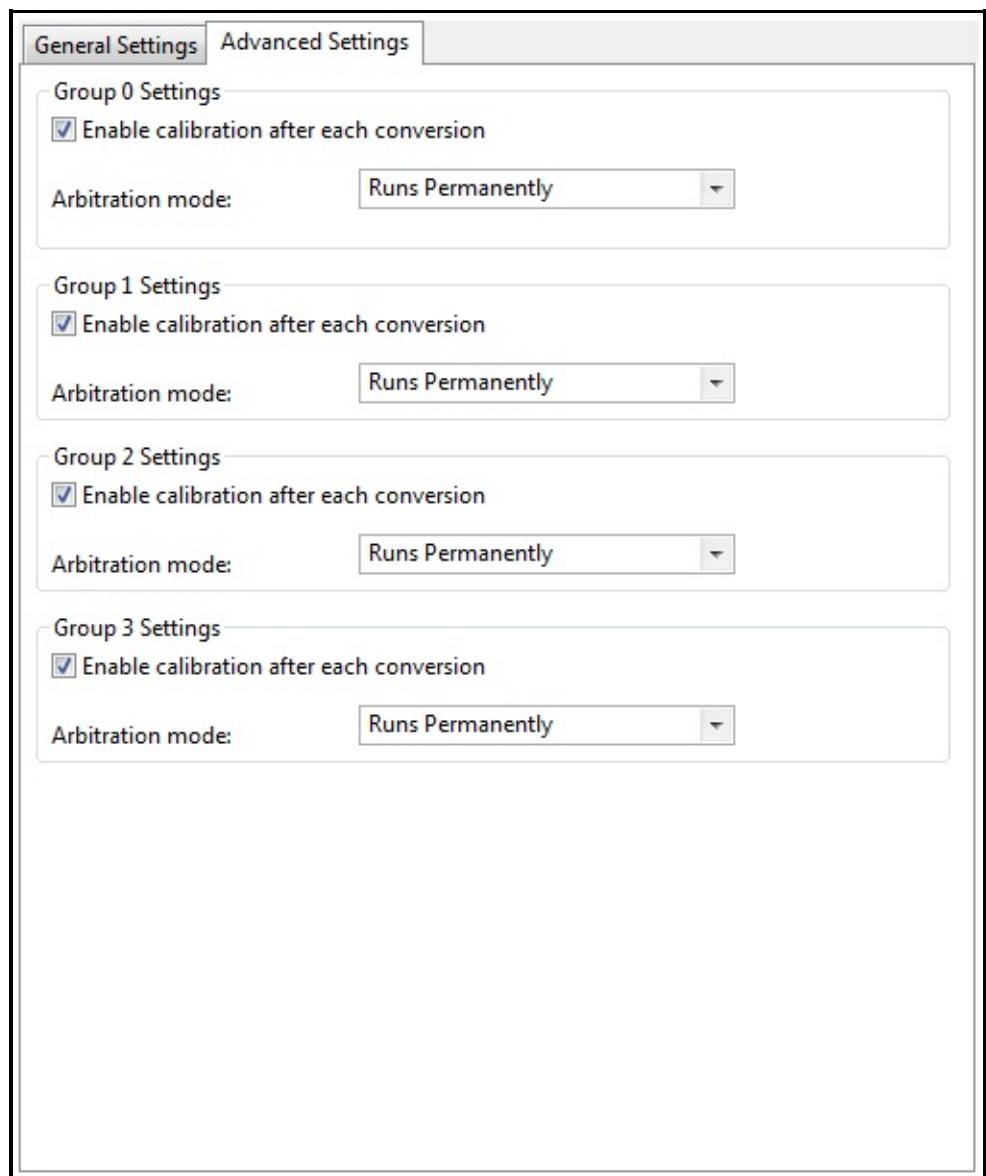


Figure 2: Advanced Settings



GLOBAL_ADC

Home

Enumerations

enum **GLOBAL_ADC_STATUS** {
 GLOBAL_ADC_SUCCESS = 0,
 GLOBAL_ADC_FAILURE,
 GLOBAL_ADC_UNINITIALIZED,
};
GLOBAL_ADC state
information. More...

typedef enum **GLOBAL_ADC_STATUS** **GLOBAL_ADC_STATUS_t**
GLOBAL_ADC state
information.

Enumeration Type Documentation

enum **GLOBAL_ADC_STATUS**

GLOBAL_ADC state information.

Enumerator:

GLOBAL_ADC_SUCCESS

APP is in INITIALIZED state after execution of the Init function

GLOBAL_ADC_FAILURE

Initialization failed returns this as status

GLOBAL_ADC_UNINITIALIZED

This is the default state after power on reset.

Definition at line **101** of file **GLOBAL_ADC.h**.

GLOBAL_ADC

Home

Data Structures

Data structures

Data Structures

struct GLOBAL_ADC_GROUP	Structure to hold the configuration information of a group. More...
typedef struct GLOBAL_ADC_GROUP	GLOBAL_ADC_GROUP_t Structure to hold the configuration information of a group.
typedef struct GLOBAL_ADC	GLOBAL_ADC_t Configuration Data structure of GLOBAL_ADC APP .

GLOBAL_ADC

Home

Methods

DAVE_APP_VERSION_t **GLOBAL_ADC_GetAppVersion** (void)
Get GLOBAL_ADC APP version.

GLOBAL_ADC_STATUS_t **GLOBAL_ADC_Init** (**GLOBAL_ADC_t**
*const handle_ptr)
Initializes the ADC global as per user
configured values.

Methods

Function Documentation

DAVE_APP_VERSION_t GLOBAL_ADC_GetAppVersion (void)

Get **GLOBAL_ADC** APP version.

Returns:

DAVE_APP_VERSION_t APP version information (major, minor and patch number)

Description:

The function can be used to check application software compatibility with a specific version of the APP.

Example Usage:

```
#include <DAVE.h>

int main(void) {
    DAVE_STATUS_t init_status;
    DAVE_APP_VERSION_t version;

    // Initialize GLOBAL_ADC APP:
    // GLOBAL_ADC_Init() is called from within DAVE_Init().
    init_status = DAVE_Init();

    version = GLOBAL_ADC_GetAppVersion();
    if (version.major != 1U) {
        // Probably, not the right version.
    }

    // More code here
    while(1) {

    }
```

```
    return (0);
}
```

Inclusion of header file

Definition at line **96** of file [GLOBAL_ADC.c](#).

[GLOBAL_ADC_STATUS_t GLOBAL_ADC_Init \(GLOBAL_ADC_t *c](#)

Initializes the ADC global as per user configured values.

Returns:

void

Description:

Initializes the VADC peripheral. Invokes various VADC LLD drivers to initialize the VADC peripheral. This would invoke The XMC_VADC_GLOBAL_Init(), XMC_VADC_GROUP_Init(). It also invokes XMC_VADC_GROUP_SetPowerMode() to power on available groups.

Example Usage:

```
#include <DAVE.h>
int main (void)
{
    DAVE_Init(); //GLOBAL_ADC_Init is called with
in DAVE_Init
    while(1);
    return 0;
}
```

This function initializes all instances of the ADC Global APP and low level app.

Definition at line **110** of file [GLOBAL_ADC.c](#).

References `GLOBAL_ADC::enable_startup_calibration`,
`GLOBAL_ADC_SUCCESS`, `GLOBAL_ADC_UNINITIALIZED`,
`GLOBAL_ADC::global_config_handle`,
`GLOBAL_ADC::global_shs_ptr`,
`GLOBAL_ADC_GROUP::group_config_handle`,
`GLOBAL_ADC_GROUP::group_handle`,
`GLOBAL_ADC::group_ptrs_array`, `GLOBAL_ADC::init_state`,
`GLOBAL_ADC::module_ptr`,
`GLOBAL_ADC_GROUP::post_calibration`, and
`GLOBAL_ADC_GROUP::state`.

GLOBAL_ADC

Home

Release History

Release History



GLOBAL_ADC

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

Data Structures

Here are the data structures with brief descriptions:

[GLOBAL_ADC](#)

Configuration Data structure of
[GLOBAL_ADC APP](#)

[GLOBAL_ADC_GROUP](#)

Structure to hold the configuration
information of a group

GLOBAL_ADC

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

GLOBAL_ADC Struct Reference

Detailed Description

Configuration Data structure of **GLOBAL_ADC APP**.

Definition at line **136** of file **GLOBAL_ADC.h**.

```
#include <GLOBAL_ADC.h>
```

Data Fields

GLOBAL_ADC_GROUP_t *const	group_ptrs_array
const XMC_VADC_GLOBAL_CONFIG_t *const	[XMC_VADC_MAXIMUM_NUM] global_config_handle
XMC_VADC_GLOBAL_t *const	module_ptr
XMC_VADC_GLOBAL_SHS_t *const	global_shs_ptr
GLOBAL_ADC_STATUS_t init_state	
const bool enable_startup_calibration	

Field Documentation

const bool GLOBAL_ADC::enable_startup_calibration

Enable startup calibration for all the converters

Definition at line [150](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

const XMC_VADC_GLOBAL_CONFIG_t* const GLOBAL_ADC::global_lld

This is the pointer to the Global LLD Handle.

Definition at line [141](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

XMC_VADC_GLOBAL_SHS_t* const GLOBAL_ADC::global_shs_pt

This is the sample and hold structure pointer

Definition at line [146](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

GLOBAL_ADC_GROUP_t* const GLOBAL_ADC::group_ptrs_array

This is an array of pointers to the ADC Groups

Definition at line [139](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

GLOBAL_ADC_STATUS_t GLOBAL_ADC::init_state

This hold the State of the **GLOBAL_ADC APP**

Definition at line **148** of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

XMC_VADC_GLOBAL_t* const GLOBAL_ADC::module_ptr

This is the register structure pointer to the VADC kernel.

Definition at line **143** of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

The documentation for this struct was generated from the following file:

- **GLOBAL_ADC.h**
-

GLOBAL_ADC

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

GLOBAL_ADC_GROUP Struct Reference

[Data structures](#)

Detailed Description

Structure to hold the configuration information of a group.

Definition at line [122](#) of file [GLOBAL_ADC.h](#).

```
#include <GLOBAL\_ADC.h>
```

Data Fields

```
XMC_VADC_GROUP_t *const group_handle  
const XMC_VADC_GROUP_CONFIG_t  
    *const group_config_handle  
    const bool post_calibration  
GLOBAL_ADC_STATUS_t state
```

Field Documentation

const XMC_VADC_GROUP_CONFIG_t* const GLOBAL_ADC_GRO

This is the pointer to the Handle of the Group APP.

Definition at line [126](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

XMC_VADC_GROUP_t* const GLOBAL_ADC_GROUP::group_hanc

This holds the VADC group Registers.

Definition at line [124](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

const bool GLOBAL_ADC_GROUP::post_calibration

This enables the post calibration for a specific group

Definition at line [128](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

GLOBAL_ADC_STATUS_t GLOBAL_ADC_GROUP::state

This enumerates the state of the APP.

Definition at line [130](#) of file **GLOBAL_ADC.h**.

Referenced by **GLOBAL_ADC_Init()**.

The documentation for this struct was generated from the following file:

- [**GLOBAL_ADC.h**](#)
-



GLOBAL_ADC

Home

Data Structures

Data Structure Index

Data Fields

Data Structure Index

G

G

GLOBAL_ADC_GROUP

GLOBAL_ADC

G



GLOBAL_ADC

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- enable_startup_calibration : [GLOBAL_ADC](#)
 - global_config_handle : [GLOBAL_ADC](#)
 - global_shs_ptr : [GLOBAL_ADC](#)
 - group_config_handle : [GLOBAL_ADC_GROUP](#)
 - group_handle : [GLOBAL_ADC_GROUP](#)
 - group_ptrs_array : [GLOBAL_ADC](#)
 - init_state : [GLOBAL_ADC](#)
 - module_ptr : [GLOBAL_ADC](#)
 - post_calibration : [GLOBAL_ADC_GROUP](#)
 - state : [GLOBAL_ADC_GROUP](#)
-

GLOBAL_ADC

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

- enable_startup_calibration : GLOBAL_ADC
 - global_config_handle : GLOBAL_ADC
 - global_shs_ptr : GLOBAL_ADC
 - group_config_handle : GLOBAL_ADC_GROUP
 - group_handle : GLOBAL_ADC_GROUP
 - group_ptrs_array : GLOBAL_ADC
 - init_state : GLOBAL_ADC
 - module_ptr : GLOBAL_ADC
 - post_calibration : GLOBAL_ADC_GROUP
 - state : GLOBAL_ADC_GROUP
-
-
-

GLOBAL_ADC

Home

File List

Globals

File List

Here is a list of all documented files with brief descriptions:

[GLOBAL_ADC.c](#) [code]

[GLOBAL_ADC.h](#) [code]

GLOBAL_ADC

[Home](#)

[File List](#)

[Globals](#)

[Functions](#)

GLOBAL_ADC.c File Reference

Detailed Description

Date:

2015-03-18

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file [**GLOBAL_ADC.c**](#).

Functions

DAVE_APP_VERSION_t	GLOBAL_ADC_GetAppVersion (void) Get GLOBAL_ADC APP version.
GLOBAL_ADC_STATUS_t	GLOBAL_ADC_Init (GLOBAL_ADC_t *const handle_ptr) Initializes the ADC global as per user configured values.

Function Documentation

GLOBAL_ADC_STATUS_t GLOBAL_ADC_Init (GLOBAL_ADC_t *config)

Initializes the ADC global as per user configured values.

This function initializes all instances of the ADC Global APP and low level app.

Definition at line [110](#) of file **GLOBAL_ADC.c**.

References [GLOBAL_ADC::enable_startup_calibration](#),
[GLOBAL_ADC_SUCCESS](#), [GLOBAL_ADC_UNINITIALIZED](#),
[GLOBAL_ADC::global_config_handle](#),
[GLOBAL_ADC::global_shs_ptr](#),
[GLOBAL_ADC_GROUP::group_config_handle](#),
[GLOBAL_ADC_GROUP::group_handle](#),
[GLOBAL_ADC::group_ptrs_array](#), [GLOBAL_ADC::init_state](#),
[GLOBAL_ADC::module_ptr](#),
[GLOBAL_ADC_GROUP::post_calibration](#), and
[GLOBAL_ADC_GROUP::state](#).

Go to the source code of this file.

GLOBAL_ADC

[Home](#)

[File List](#)

[Globals](#)

[Data Structures](#)

GLOBAL_ADC.h File Reference

Detailed Description

Date:

2015-03-18

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file [**GLOBAL_ADC.h**](#).

Data Structures

struct **GLOBAL_ADC_GROUP**

Structure to hold the configuration information of a group.

[More...](#)

struct **GLOBAL_ADC**

Configuration Data structure of **GLOBAL_ADC APP**. [More...](#)

TypeDefs

typedef struct **GLOBAL_ADC_GROUP** **GLOBAL_ADC_GROUP_t**
Structure to hold the
configuration information of
a group.

typedef struct **GLOBAL_ADC** **GLOBAL_ADC_t**
Configuration Data structure
of **GLOBAL_ADC APP**.

Functions

DAVE_APP_VERSION_t	GLOBAL_ADC_GetAppVersion (void) Get GLOBAL_ADC APP version.
GLOBAL_ADC_STATUS_t	GLOBAL_ADC_Init (GLOBAL_ADC_t *const handle_ptr) Initializes the ADC global as per user configured values.
enum	GLOBAL_ADC_STATUS { GLOBAL_ADC_SUCCESS = 0, GLOBAL_ADC_FAILURE , GLOBAL_ADC_UNINITIALIZED , } GLOBAL_ADC state information. More...
typedef enum GLOBAL_ADC_STATUS	GLOBAL_ADC_STATUS_t GLOBAL_ADC state information.

Go to the source code of this file.

GLOBAL_ADC

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- GLOBAL_ADC_FAILURE : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_GetAppVersion() : [GLOBAL_ADC.c](#) , [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_GROUP_t : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_Init() : [GLOBAL_ADC.c](#) , [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_STATUS : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_STATUS_t : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_SUCCESS : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_t : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_UNINITIALIZED : [GLOBAL_ADC.h](#)
-

GLOBAL_ADC

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- GLOBAL_ADC_GetAppVersion() : **GLOBAL_ADC.c** , **GLOBAL_ADC.h**
 - GLOBAL_ADC_Init() : **GLOBAL_ADC.c** , **GLOBAL_ADC.h**
-



GLOBAL_ADC

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- GLOBAL_ADC_GROUP_t : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_STATUS_t : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_t : [GLOBAL_ADC.h](#)
-



GLOBAL_ADC

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- GLOBAL_ADC_STATUS : [GLOBAL_ADC.h](#)
-



GLOBAL_ADC

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- GLOBAL_ADC_FAILURE : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_SUCCESS : [GLOBAL_ADC.h](#)
 - GLOBAL_ADC_UNINITIALIZED : [GLOBAL_ADC.h](#)
-



GLOBAL_ADC

Home	
File List	Globals

GLOBAL_ADC.h

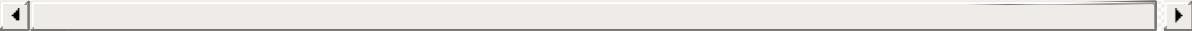
[Go to the documentation of this file.](#)

```
00001
00072 #ifndef GLOBAL_ADC_H
00073 #define GLOBAL_ADC_H
00074
00075 /*****
00076 * HEADER FILES
00077 *****/
00078 #include <xmc_vadc.h>
00079 #include "global_adc_conf.h"
00080 #include <DAVE_common.h>
00081
00082 *****/
00083 * MACROS
00084 *****/
00085 #if (((XMC_LIB_MAJOR_VERSION == 2U) && \
00086         (XMC_LIB_MINOR_VERSION >= 0U) && \
00087         (XMC_LIB_PATCH_VERSION >= 0U)))
00088 #error "GLOBAL_ADC requires XMC Peripheral Library v2.0.0 or higher"
```

```
00089 #endif
00090 /*****
00091 ****
00092 * ENUMS
00093 ****
00094 ****
00095 ****
00096 ****
00097 ****
00098 ****
00099 ****
00100 ****
00101 typedef enum GLOBAL_ADC_STATUS
00102 {
00103     GLOBAL_ADC_SUCCESS = 0,
00104     GLOBAL_ADC_FAILURE,
00105     GLOBAL_ADC_UNINITIALIZED
00106 } GLOBAL_ADC_STATUS_t;
00107
00108 /*****
00109 ****
00110 ****
00111 ****
00112 * DATA STRUCTURES
00113 ****
00114 ****
00115 ****
00116 ****
00117 #if XMC_VADC_GROUP_AVAILABLE == 1U
00118 typedef struct GLOBAL_ADC_GROUP
00119 {
00120     XMC_VADC_GROUP_t *const group_handle;
00121
00122     const XMC_VADC_GROUP_CONFIG_t* const group_config_handle;
00123     const bool post_calibration;
00124
00125     GLOBAL_ADC_STATUS_t state;
00126
00127 } GLOBAL_ADC_GROUP_t;
00128 #endif
00129
00130
00131
00132
00133
```

```
00136 typedef struct GLOBAL_ADC
00137 {
00138 #if XMC_VADC_GROUP_AVAILABLE == 1U
00139     GLOBAL_ADC_GROUP_t* const group_ptrs_array
00140     [XMC_VADC_MAXIMUM_NUM_GROUPS];
00141     const XMC_VADC_GLOBAL_CONFIG_t* const global_config_handle;
00143     XMC_VADC_GLOBAL_t* const module_ptr;
00145 #if(XMC_VADC_SHS_AVAILABLE == 1U)
00146     XMC_VADC_GLOBAL_SHS_t* const global_shs_ptr
00147 ;
00147 #endif
00148     GLOBAL_ADC_STATUS_t init_state;
00150     const bool enable_startup_calibration;
00151 } GLOBAL_ADC_t;
00152
00162 #ifdef __cplusplus
00163 extern "C" {
00164 #endif
00165 ****
00166 ****
00167 ****
00168 ****
00203 DAVE_APP_VERSION_t GLOBAL_ADC_GetAppVersion(
void);
00204
00227 GLOBAL_ADC_STATUS_t GLOBAL_ADC_Init(GLOBAL_A
DC_t *const handle_ptr);
00228
00229 #include "global_adc_extern.h"
```

```
00230
00235 #ifdef __cplusplus
00236 }
00237 #endif
00238
00239 #endif /* GLOBAL_ADC_H_ */
```



GLOBAL_ADC

Home	
File List	Globals

GLOBAL_ADC.c

[Go to the documentation of this file.](#)

```
00001
00072 /*****
***** 
***** 
00073 * HEADER FILES
00074 *****
***** 
***** 
***** /
00075
00077 #include "global_adc.h"
00078
00079 /*****
***** 
***** 
***** 
00080 * MACROS
00081 *****
***** 
***** 
***** /
00082
00083 /*****
***** 
***** 
***** 
00084 * LOCAL DATA
00085 *****
***** 
***** 
***** /
00086
```

```
00087 /*****  
*****  
*****  
00088 * LOCAL ROUTINES  
00089 *****  
*****  
*****  
*****  
*****  
00090  
00091 /*****  
*****  
*****  
*****  
*****  
00092 * API IMPLEMENTATION  
00093 *****  
*****  
*****  
*****  
*****  
00094  
00095 /*This function returns the version of the G  
LOBAL_ADC APP*/  
00096 DAVE_APP_VERSION_t GLOBAL_ADC_GetAppVersion(  
void)  
00097 {  
00098     DAVE_APP_VERSION_t version;  
00099  
00100     version.major = (uint8_t) GLOBAL_ADC_MAJOR  
_VERSION;  
00101     version.minor = (uint8_t) GLOBAL_ADC_MINOR  
_VERSION;  
00102     version.patch = (uint8_t) GLOBAL_ADC_PATCH  
_VERSION;  
00103  
00104     return version;  
00105 }  
00106 /*~~~~~  
~~~~~  
~~~~~*/  
00110 GLOBAL_ADC_STATUS_t GLOBAL_ADC_Init(GLOBAL_A  
DC_t *const handle_ptr)
```

```
00111 {
00112     XMC_ASSERT("GLOBAL_ADC_Init:Invalid handle
00113 _ptr", (handle_ptr != NULL))
00114     uint32_t group_index;
00115 #endif
00116
00117     if (GLOBAL_ADC_UNINITIALIZED == handle_ptr
00118 ->init_state)
00119     {
00120         /* Initialize an instance of Global hard
00121 ware */
00122         XMC_VADC_GLOBAL_Init(handle_ptr->module_
00123 ptr, handle_ptr->global_config_handle);
00124
00125         /* Initialize all the Groups */
00126 #if (XMC_VADC_GROUP_AVAILABLE == 1U)
00127         for(group_index = (uint32_t)0; group_index < XMC_VADC_MAXIMUM_NUM_GROUPS; group_index++)
00128         {
00129             /*Initialize Group*/
00130             XMC_VADC_GROUP_Init(handle_ptr->group_
00131 ptrs_array[group_index]->group_handle,
00132                                     handle_ptr->grou
00133 p_ptrs_array[group_index]->group_config_handle);
00134
00135             /* Switch on the converter of the Grou
00136 p[group_index]*/
00137             XMC_VADC_GROUP_SetPowerMode(handle_ptr
00138 ->group_ptrs_array[group_index]->group_handle,
00139                                     XMC_VADC_G
00140 ROUP_POWERMODE_NORMAL);
00141
00142             /* Disable the post calibration option
00143 for the respective group*/
00144             if ((bool)false == handle_ptr->group_p
00145 trs_array[group_index]->post_calibration)
```

```
00136      {
00137          XMC_VADC_GLOBAL_DisablePostCalibration(handle_ptr->module_ptr, group_index);
00138      }
00139
00140 #if(XMC_VADC_SHS_AVAILABLE == 1U)
00141     XMC_VADC_GLOBAL_SHS_EnableAcceleratedMode(handle_ptr->global_shs_ptr, (XMC_VADC_GROUP_INDEX_t)group_index);
00142 #endif
00143
00144     handle_ptr->group_ptrs_array[group_index]->state = GLOBAL_ADC_SUCCESS;
00145 }
00146 #endif /* _XMC_VADC_GROUP_AVAILABLE_ */
00147     if((bool)true == handle_ptr->enable_startup_calibration)
00148     {
00149         XMC_VADC_GLOBAL_StartupCalibration(handle_ptr->module_ptr);
00150     }
00151     handle_ptr->init_state = GLOBAL_ADC_SUCCESS;
00152 }
00153 return (handle_ptr->init_state);
00154 }
```