ADC

Type definitions:

Name:	**strGroupInfo**
Type:	structure
Range:	Implementation Specific
Description:	This type of the external data structure shall contain the initialization data for the ADC group.

For the type strGroupInfo, the definition for each Channel shall contain:

- Group id , it shall be unique.
- Conversion mode , one shot or continuous.
- Access mode, single access or streaming.
- Trigger , software or hardware.
- Hardware trigger source , in case of hardware trigger , it can be PWM or timer or comparator or GPIO pins.
- Number of streaming samples , started from 1.
- Array of the pins in the group

Name:	**strModuleInfo**
Type:	structure
Range:	Implementation Specific
Description:	This type of the external data structure shall contain the initialization data for the ADC module.

For the type strModuleInfo, the definition for each Channel shall contain:

- Module ID, it shall be unique , started from 0.

- Sequencer number , it can be 0,1,2,3.
- the sampling rate , it can be 125k , 250k, 500k, 1M.
- Number of input channels of the group which will parsed to this module.
- The group ID which will be parsed to this module with those specification.

Name:	**ADC_enumerr**		
Type:	Enumeration		
Range:	ADC_OKAY	0x00	The function is done without any error.
	ADC_NOTOKAY	0x01	The function contains an error.
Description:	This type is an indication for most of the functions if there's an error through the run time or not.		

Name:	**ADC_enumconv**		
Type:	Enumeration		
Range:	CONV_DONE	0x00	The conversion is done
	CONV_NOT_YET	0x01	Conversion in process
	error	0x02	There is an error dunring the conversion
Description:	This type is an indication for the ADC conversion.		

Function definitions:

Service name:	**ADC_enuminit**	
Syntax:	ADC_enumerr ADC_enuminit(void);	
Sync/Async:	Synchronous	
Arguments:	None	
Return value:	ADC_enumerr	ADC_OKAY:No error. ADC_NOTOKAY:Error.
Description:	It initiate the ADC with the configuration chosen at config.h file and config.c	

Service name:	**ADC_enumSetBuffer**	
Syntax:	ADC_enumerr ADC_enumSetBuffer(u8 GP_ID, u32 * BufferPtr)	
Sync/Async:	Asynchronous	
Arguments:	GP_ID	Numeric ID of requested ADC channel group.
	BufferPtr	pointer to result data buffer
Return value:	ADC_enumerr	ADC_OKAY:No error. ADC_NOTOKAY:Error.
Description:	it gives for a certain group its buffer pointer to hold the converted values. * Note: it has to be called before the conversion	

Service name:	**ADC_enumInterrupt_Enable**	
Syntax:	ADC_enumerr ADC_enumInterrupt_Enable(u8 module_id);	
Sync/Async:	Asynchronous	
Arguments:	module_id	Numeric ID of requested ADC sequencer.
Return value:	ADC_enumerr	ADC_OKAY:No error. ADC_NOTOKAY:Error.

Description:	it enables the interrupt for a specific module "sequencer"

Service name:	**ADC_vidStartConv**	
Syntax:	void ADC_vidStartConv(u8 module_id);	
Sync/**Async**:	synchronous	
Arguments:	module_id	Numeric ID of requested ADC sequencer.
Return value:	None.	
Description:	it starts the module conversion with the trigger chosen at config.c file. note: ADC_Interrupt_Enable has to be called first	

Service name:	**ADC_enumGetData**		
Syntax:	ADC_enumconv ADC_enumGetData(u8 group_ID);		
Sync/Async:	Asynchronous		
Arguments:	group_ID		Numeric ID of requested ADC channel group.
Return value:	ADC_enumconv	CONV_DONE	Conversion is done , so it's ready to get the converted data.
		CONV_NOT_YET	Conversion is not done , so it's not to get the converted data.
		error	There is an error , maybe the group id is not exist.
Description:	It checks whether the conversion is done or not yet, and if it's done it moves the data from the FIFO to the group buffer.		