when the Firewall is opened.

__HAL_FIREWALL_GET_PREARM

Description:

- Check whether or not the Firewall pre arm bit is set.

Return value:

- FPA: bit setting status (TRUE or FALSE).

Notes:

- This macro can be executed inside a code area protected by the Firewall. This macro can be executed whatever the Firewall state (opened or closed) when NVDSL register is equal to 0. Otherwise (when NVDSL register is different from 0, that is, when the non volatile data segment is defined), the macro can be executed only when the Firewall is opened.

FIREWALL pre arm status

FIREWALL_PRE_ARM_RESET

FIREWALL_PRE_ARM_SET

FIREWALL volatile data segment execution status

FIREWALL_VOLATILEDATA_NOT_EXECUTABLE

FIREWALL_VOLATILEDATA_EXECUTABLE

FIREWALL volatile data segment share status

FIREWALL_VOLATILEDATA_NOT_SHARED

FIREWALL_VOLATILEDATA_SHARED

24 HAL FLASH Generic Driver

24.1 FLASH Firmware driver registers structures

24.1.1 FLASH_EraseInitTypeDef

Data Fields

- ***uint32_t TypeErase***
- ***uint32_t Banks***
- ***uint32_t Page***
- ***uint32_t NbPages***

Field Documentation

- ***uint32_t FLASH_EraseInitTypeDef::TypeErase***
  Mass erase or page erase. This parameter can be a value of ***FLASH_Type_Erase***
- ***uint32_t FLASH_EraseInitTypeDef::Banks***
  Select bank to erase. This parameter must be a value of ***FLASH_Banks***
  (FLASH_BANK_BOTH should be used only for mass erase)
- ***uint32_t FLASH_EraseInitTypeDef::Page***
  Initial Flash page to erase when page erase is disabled This parameter must be a value between 0 and (max number of pages in the bank - 1) (eg: 255 for 1MB dual bank)
- ***uint32_t FLASH_EraseInitTypeDef::NbPages***
  Number of pages to be erased. This parameter must be a value between 1 and (max number of pages in the bank - value of initial page)

24.1.2 FLASH_OBProgramInitTypeDef

Data Fields

- ***uint32_t OptionType***
- ***uint32_t WRPArea***
- ***uint32_t WRPStartOffset***
- ***uint32_t WRPEndOffset***
- ***uint32_t RDPLevel***
- ***uint32_t USERType***
- ***uint32_t USERConfig***
- ***uint32_t PCROPConfig***
- ***uint32_t PCROPStartAddr***
- ***uint32_t PCROPEndAddr***

Field Documentation

- ***uint32_t FLASH_OBProgramInitTypeDef::OptionType***
  Option byte to be configured. This parameter can be a combination of the values of ***FLASH_OB_Type***
- ***uint32_t FLASH_OBProgramInitTypeDef::WRPArea***
  Write protection area to be programmed (used for OPTIONBYTE_WRP). Only one WRP area could be programmed at the same time. This parameter can be value of ***FLASH_OB_WRP_Area***
- ***uint32_t FLASH_OBProgramInitTypeDef::WRPStartOffset***
  Write protection start offset (used for OPTIONBYTE_WRP). This parameter must be a

value between 0 and (max number of pages in the bank - 1) (eg: 25 for 1MB dual bank)

- ***uint32_t FLASH_OBProgramInitTypeDef::WRPEndOffset***
  Write protection end offset (used for OPTIONBYTE_WRP). This parameter must be a value between WRPStartOffset and (max number of pages in the bank - 1)
- ***uint32_t FLASH_OBProgramInitTypeDef::RDPLevel***
  Set the read protection level.. (used for OPTIONBYTE_RDP). This parameter can be a value of ***FLASH_OB_Read_Protection***
- ***uint32_t FLASH_OBProgramInitTypeDef::USERType***
  User option byte(s) to be configured (used for OPTIONBYTE_USER). This parameter can be a combination of ***FLASH_OB_USER_Type***
- ***uint32_t FLASH_OBProgramInitTypeDef::USERConfig***
  Value of the user option byte (used for OPTIONBYTE_USER). This parameter can be a combination of ***FLASH_OB_USER_BOR_LEVEL***, ***FLASH_OB_USER_nRST_STOP***, ***FLASH_OB_USER_nRST_STANDBY***, ***FLASH_OB_USER_nRST_SHUTDOWN***, ***FLASH_OB_USER_IWDG_SW***, ***FLASH_OB_USER_IWDG_STOP***, ***FLASH_OB_USER_IWDG_STANDBY***, ***FLASH_OB_USER_WWDG_SW***, ***FLASH_OB_USER_BFB2***, ***FLASH_OB_USER_DUALBANK***, ***FLASH_OB_USER_nBOOT1***, ***FLASH_OB_USER_SRAM2_PE*** and ***FLASH_OB_USER_SRAM2_RST***
- ***uint32_t FLASH_OBProgramInitTypeDef::PCROPConfig***
  Configuration of the PCROP (used for OPTIONBYTE_PCROP). This parameter must be a combination of ***FLASH_Banks*** (except FLASH_BANK_BOTH) and ***FLASH_OB_PCROP_RDP***
- ***uint32_t FLASH_OBProgramInitTypeDef::PCROPStartAddr***
  PCROP Start address (used for OPTIONBYTE_PCROP). This parameter must be a value between begin and end of bank => Be careful of the bank swapping for the address
- ***uint32_t FLASH_OBProgramInitTypeDef::PCROPEndAddr***
  PCROP End address (used for OPTIONBYTE_PCROP). This parameter must be a value between PCROP Start address and end of bank

24.1.3 FLASH_ProcessTypeDef

Data Fields

- ***HAL_LockTypeDef Lock***
- ***__IO uint32_t ErrorCode***
- ***__IO FLASH_ProcedureTypeDef ProcedureOnGoing***
- ***__IO uint32_t Address***
- ***__IO uint32_t Bank***
- ***__IO uint32_t Page***
- ***__IO uint32_t NbPagesToErase***
- ***__IO FLASH_CacheTypeDef CacheToReactivate***

Field Documentation

- ***HAL_LockTypeDef FLASH_ProcessTypeDef::Lock***
- ***__IO uint32_t FLASH_ProcessTypeDef::ErrorCode***
- ***__IO FLASH_ProcedureTypeDef FLASH_ProcessTypeDef::ProcedureOnGoing***
- ***__IO uint32_t FLASH_ProcessTypeDef::Address***
- ***__IO uint32_t FLASH_ProcessTypeDef::Bank***
- ***__IO uint32_t FLASH_ProcessTypeDef::Page***
- ***__IO uint32_t FLASH_ProcessTypeDef::NbPagesToErase***
- ***__IO FLASH_CacheTypeDef FLASH_ProcessTypeDef::CacheToReactivate***

24.2 FLASH Firmware driver API description

24.2.1 FLASH peripheral features

The Flash memory interface manages CPU AHB I-Code and D-Code accesses to the Flash memory. It implements the erase and program Flash memory operations and the read and write protection mechanisms.

The Flash memory interface accelerates code execution with a system of instruction prefetch and cache lines.

The FLASH main features are:

- Flash memory read operations
- Flash memory program/erase operations
- Read / write protections
- Option bytes programming
- Prefetch on I-Code
- 32 cache lines of 4*64 bits on I-Code
- 8 cache lines of 4*64 bits on D-Code
- Error code correction (ECC): Data in flash are 72-bits word (8 bits added per double word)

24.2.2 How to use this driver

This driver provides functions and macros to configure and program the FLASH memory of all STM32L4xx devices.

1. Flash Memory IO Programming functions:
   - Lock and Unlock the FLASH interface using HAL_FLASH_Unlock() and HAL_FLASH_Lock() functions
   - Program functions: double word and fast program (full row programming)
   - There Two modes of programming:
     - Polling mode using HAL_FLASH_Program() function
     - Interrupt mode using HAL_FLASH_Program_IT() function
2. Interrupts and flags management functions:
   - Handle FLASH interrupts by calling HAL_FLASH_IRQHandler()
   - Callback functions are called when the flash operations are finished: HAL_FLASH_EndOfOperationCallback() when everything is ok, otherwise HAL_FLASH_OperationErrorCallback()
   - Get error flag status by calling HAL_GetError()
3. Option bytes management functions:
   - Lock and Unlock the option bytes using HAL_FLASH_OB_Unlock() and HAL_FLASH_OB_Lock() functions
   - Launch the reload of the option bytes using HAL_FLASH_Launch() function. In this case, a reset is generated

In addition to these functions, this driver includes a set of macros allowing to handle the following operations:

- Set the latency
- Enable/Disable the prefetch buffer
- Enable/Disable the Instruction cache and the Data cache
- Reset the Instruction cache and the Data cache
- Enable/Disable the Flash power-down during low-power run and sleep modes
- Enable/Disable the Flash interrupts
- Monitor the Flash flags status

24.2.3 Programming operation functions

This subsection provides a set of functions allowing to manage the FLASH program operations.

This section contains the following APIs:

- ***HAL_FLASH_Program()***
- ***HAL_FLASH_Program_IT()***
- ***HAL_FLASH_IRQHandler()***
- ***HAL_FLASH_EndOfOperationCallback()***
- ***HAL_FLASH_OperationErrorCallback()***

24.2.4 Peripheral Control functions

This subsection provides a set of functions allowing to control the FLASH memory operations.

This section contains the following APIs:

- ***HAL_FLASH_Unlock()***
- ***HAL_FLASH_Lock()***
- ***HAL_FLASH_OB_Unlock()***
- ***HAL_FLASH_OB_Lock()***
- ***HAL_FLASH_OB_Launch()***

24.2.5 Peripheral Errors functions

This subsection permits to get in run-time Errors of the FLASH peripheral.

This section contains the following APIs:

- ***HAL_FLASH_GetError()***

24.2.6 Detailed description of functions

HAL_FLASH_Program

Function name	**HAL_StatusTypeDef HAL_FLASH_Program (uint32_t TypeProgram, uint32_t Address, uint64_t Data)**
Function description	Program double word or fast program of a row at a specified address.
Parameters	• **TypeProgram:** Indicate the way to program at a specified address. This parameter can be a value of FLASH Program Type • **Address:** specifies the address to be programmed. • **Data:** specifies the data to be programmed This parameter is the data for the double word program and the address where are stored the data for the row fast program
Return values	• **HAL_StatusTypeDef:** HAL Status

HAL_FLASH_Program_IT

Function name	**HAL_StatusTypeDef HAL_FLASH_Program_IT (uint32_t TypeProgram, uint32_t Address, uint64_t Data)**
Function description	Program double word or fast program of a row at a specified

address with interrupt enabled.

Parameters
- **TypeProgram:** Indicate the way to program at a specified address. This parameter can be a value of FLASH Program Type
- **Address:** specifies the address to be programmed.
- **Data:** specifies the data to be programmed This parameter is the data for the double word program and the address where are stored the data for the row fast program

Return values
- **HAL:** Status

HAL_FLASH_IRQHandler

Function name **void HAL_FLASH_IRQHandler (void)**

Function description Handle FLASH interrupt request.

Return values
- **None:**

HAL_FLASH_EndOfOperationCallback

Function name **void HAL_FLASH_EndOfOperationCallback (uint32_t ReturnValue)**

Function description FLASH end of operation interrupt callback.

Parameters
- **ReturnValue:** The value saved in this parameter depends on the ongoing procedure Mass Erase: Bank number which has been requested to erase Page Erase: Page which has been erased (if 0xFFFFFFFF, it means that all the selected pages have been erased) Program: Address which was selected for data program

Return values
- **None:**

HAL_FLASH_OperationErrorCallback

Function name **void HAL_FLASH_OperationErrorCallback (uint32_t ReturnValue)**

Function description FLASH operation error interrupt callback.

Parameters
- **ReturnValue:** The value saved in this parameter depends on the ongoing procedure Mass Erase: Bank number which has been requested to erase Page Erase: Page number which returned an error Program: Address which was selected for data program

Return values
- **None:**

HAL_FLASH_Unlock

Function name **HAL_StatusTypeDef HAL_FLASH_Unlock (void)**

Function description Unlock the FLASH control register access.

Return values
- **HAL:** Status

HAL_FLASH_Lock

Function name **HAL_StatusTypeDef HAL_FLASH_Lock (void)**

Function description Lock the FLASH control register access.

Return values
- **HAL:** Status

HAL_FLASH_OB_Unlock

Function name **HAL_StatusTypeDef HAL_FLASH_OB_Unlock (void)**

Function description Unlock the FLASH Option Bytes Registers access.

Return values
- **HAL:** Status

HAL_FLASH_OB_Lock

Function name **HAL_StatusTypeDef HAL_FLASH_OB_Lock (void)**

Function description Lock the FLASH Option Bytes Registers access.

Return values
- **HAL:** Status

HAL_FLASH_OB_Launch

Function name **HAL_StatusTypeDef HAL_FLASH_OB_Launch (void)**

Function description Launch the option byte loading.

Return values
- **HAL:** Status

HAL_FLASH_GetError

Function name **uint32_t HAL_FLASH_GetError (void)**

Function description Get the specific FLASH error flag.

Return values
- **FLASH_ErrorCode:** The returned value can be:
  - HAL_FLASH_ERROR_RD: FLASH Read Protection error flag (PCROP)
  - HAL_FLASH_ERROR_PGS: FLASH Programming Sequence error flag
  - HAL_FLASH_ERROR_PGP: FLASH Programming Parallelism error flag
  - HAL_FLASH_ERROR_PGA: FLASH Programming Alignment error flag
  - HAL_FLASH_ERROR_WRP: FLASH Write protected error flag
  - HAL_FLASH_ERROR_OPERATION: FLASH operation Error flag
  - HAL_FLASH_ERROR_NONE: No error set
  - HAL_FLASH_ERROR_OP: FLASH Operation error
  - HAL_FLASH_ERROR_PROG: FLASH Programming error
  - HAL_FLASH_ERROR_WRP: FLASH Write protection error
  - HAL_FLASH_ERROR_PGA: FLASH Programming alignment error

- HAL_FLASH_ERROR_SIZ: FLASH Size error
- HAL_FLASH_ERROR_PGS: FLASH Programming sequence error
- HAL_FLASH_ERROR_MIS: FLASH Fast programming data miss error
- HAL_FLASH_ERROR_FAST: FLASH Fast programming error
- HAL_FLASH_ERROR_RD: FLASH PCROP read error
- HAL_FLASH_ERROR_OPTV: FLASH Option validity error
- FLASH_FLAG_PEMPTY: FLASH Boot from not programmed flash (apply only for STM32L43x/STM32L44x devices)
- HAL_FLASH_ERROR_ECCD: FLASH two ECC errors have been detected

24.3 FLASH Firmware driver defines

24.3.1 FLASH

FLASH Banks

FLASH_BANK_1	Bank 1
FLASH_BANK_2	Bank 2
FLASH_BANK_BOTH	Bank1 and Bank2

FLASH Error

HAL_FLASH_ERROR_NONE

HAL_FLASH_ERROR_OP

HAL_FLASH_ERROR_PROG

HAL_FLASH_ERROR_WRP

HAL_FLASH_ERROR_PGA

HAL_FLASH_ERROR_SIZ

HAL_FLASH_ERROR_PGS

HAL_FLASH_ERROR_MIS

HAL_FLASH_ERROR_FAST

HAL_FLASH_ERROR_RD

HAL_FLASH_ERROR_OPTV

HAL_FLASH_ERROR_ECCD

HAL_FLASH_ERROR_PEMPTY

FLASH Exported Macros

__HAL_FLASH_SET_LATENCY

Description:

- Set the FLASH Latency.

Parameters:

- __LATENCY__: FLASH

Latency This parameter can be one of the following values:
- FLASH_LATENCY_0: FLASH Zero wait state
- FLASH_LATENCY_1: FLASH One wait state
- FLASH_LATENCY_2: FLASH Two wait states
- FLASH_LATENCY_3: FLASH Three wait states
- FLASH_LATENCY_4: FLASH Four wait states

Return value:

- None

__HAL_FLASH_GET_LATENCY

Description:

- Get the FLASH Latency.

Return value:

- FLASH: Latency This parameter can be one of the following values:
  - FLASH_LATENCY_0: FLASH Zero wait state
  - FLASH_LATENCY_1: FLASH One wait state
  - FLASH_LATENCY_2: FLASH Two wait states
  - FLASH_LATENCY_3: FLASH Three wait states
  - FLASH_LATENCY_4: FLASH Four wait states

__HAL_FLASH_PREFETCH_BUFFER_ENABLE

Description:

- Enable the FLASH prefetch buffer.

Return value:

- None

__HAL_FLASH_PREFETCH_BUFFER_DISABLE

Description:

- Disable the FLASH prefetch buffer.

Return value:

- None

__HAL_FLASH_INSTRUCTION_CACHE_ENABLE

Description:

- Enable the FLASH instruction cache.

Return value:

- none

__HAL_FLASH_INSTRUCTION_CACHE_DISABLE

Description:

- Disable the FLASH instruction cache.

Return value:

- none

__HAL_FLASH_DATA_CACHE_ENABLE

Description:

- Enable the FLASH data cache.

Return value:

- none

__HAL_FLASH_DATA_CACHE_DISABLE

Description:

- Disable the FLASH data cache.

Return value:

- none

__HAL_FLASH_INSTRUCTION_CACHE_RESET

Description:

- Reset the FLASH instruction Cache.

Return value:

- None

Notes:

- This function must be used only when the Instruction Cache is disabled.

__HAL_FLASH_DATA_CACHE_RESET

Description:

- Reset the FLASH data Cache.

Return value:

- None

Notes:

- This function must be used only when the data Cache is disabled.

__HAL_FLASH_POWER_DOWN_ENABLE

Notes:

- Writing this bit to 0 this bit, automatically the keys are loss and a new unlock sequence is necessary to re-write it to 1.

__HAL_FLASH_POWER_DOWN_DISABLE

Notes:

- Writing this bit to 0 this bit,

automatically the keys are loss and a new unlock sequence is necessary to re-write it to 1.

__HAL_FLASH_SLEEP_POWERDOWN_ENABLE

Description:

- Enable the FLASH power down during Low-Power sleep mode.

Return value:

- none

__HAL_FLASH_SLEEP_POWERDOWN_DISABLE

Description:

- Disable the FLASH power down during Low-Power sleep mode.

Return value:

- none

FLASH Flags Definition

FLASH_FLAG_EOP	FLASH End of operation flag
FLASH_FLAG_OPERR	FLASH Operation error flag
FLASH_FLAG_PROGERR	FLASH Programming error flag
FLASH_FLAG_WRPERR	FLASH Write protection error flag
FLASH_FLAG_PGAERR	FLASH Programming alignment error flag
FLASH_FLAG_SIZERR	FLASH Size error flag
FLASH_FLAG_PGSERR	FLASH Programming sequence error flag
FLASH_FLAG_MISERR	FLASH Fast programming data miss error flag
FLASH_FLAG_FASTERR	FLASH Fast programming error flag
FLASH_FLAG_RDERR	FLASH PCROP read error flag
FLASH_FLAG_OPTVERR	FLASH Option validity error flag
FLASH_FLAG_BSY	FLASH Busy flag
FLASH_FLAG_PEMPTY	FLASH Program empty
FLASH_FLAG_ECCC	FLASH ECC correction
FLASH_FLAG_ECCD	FLASH ECC detection
FLASH_FLAG_ALL_ERRORS	

FLASH Interrupts Macros

__HAL_FLASH_ENABLE_IT

Description:

- Enable the specified FLASH interrupt.

Parameters:

- __INTERRUPT__: FLASH interrupt This parameter can be any combination of the following values:
  – FLASH_IT_EOP: End of FLASH Operation

Interrupt
- FLASH_IT_OPERR: Error Interrupt
- FLASH_IT_RDERR: PCROP Read Error Interrupt
- FLASH_IT_ECCC: ECC Correction Interrupt

Return value:

- none

__HAL_FLASH_DISABLE_IT

Description:

- Disable the specified FLASH interrupt.

Parameters:

- __INTERRUPT__: FLASH interrupt This parameter can be any combination of the following values:
  - FLASH_IT_EOP: End of FLASH Operation Interrupt
  - FLASH_IT_OPERR: Error Interrupt
  - FLASH_IT_RDERR: PCROP Read Error Interrupt
  - FLASH_IT_ECCC: ECC Correction Interrupt

Return value:

- none

__HAL_FLASH_GET_FLAG

Description:

- Check whether the specified FLASH flag is set or not.

Parameters:

- __FLAG__: specifies the FLASH flag to check. This parameter can be one of the following values:
  - FLASH_FLAG_EOP: FLASH End of Operation flag
  - FLASH_FLAG_OPERR: FLASH Operation error flag
  - FLASH_FLAG_PROGERR: FLASH Programming error flag
  - FLASH_FLAG_WRPERR: FLASH Write protection error flag
  - FLASH_FLAG_PGAERR: FLASH Programming alignment error flag
  - FLASH_FLAG_SIZERR: FLASH Size error flag
  - FLASH_FLAG_PGSERR: FLASH Programming sequence error flag
  - FLASH_FLAG_MISERR: FLASH Fast programming data miss error flag
  - FLASH_FLAG_FASTERR: FLASH Fast programming error flag
  - FLASH_FLAG_RDERR: FLASH PCROP read error flag
  - FLASH_FLAG_OPTVERR: FLASH Option validity error flag
  - FLASH_FLAG_BSY: FLASH write/erase

operations in progress flag
  - FLASH_FLAG_PEMPTY: FLASH Boot from not programmed flash (apply only for STM32L43x/STM32L44x devices)
  - FLASH_FLAG_ECCC: FLASH one ECC error has been detected and corrected
  - FLASH_FLAG_ECCD: FLASH two ECC errors have been detected

Return value:

- The: new state of FLASH_FLAG (SET or RESET).

__HAL_FLASH_CLEAR_FLAG **Description:**

- Clear the FLASH's pending flags.

Parameters:

- __FLAG__: specifies the FLASH flags to clear. This parameter can be any combination of the following values:
  - FLASH_FLAG_EOP: FLASH End of Operation flag
  - FLASH_FLAG_OPERR: FLASH Operation error flag
  - FLASH_FLAG_PROGERR: FLASH Programming error flag
  - FLASH_FLAG_WRPERR: FLASH Write protection error flag
  - FLASH_FLAG_PGAERR: FLASH Programming alignment error flag
  - FLASH_FLAG_SIZERR: FLASH Size error flag
  - FLASH_FLAG_PGSERR: FLASH Programming sequence error flag
  - FLASH_FLAG_MISERR: FLASH Fast programming data miss error flag
  - FLASH_FLAG_FASTERR: FLASH Fast programming error flag
  - FLASH_FLAG_RDERR: FLASH PCROP read error flag
  - FLASH_FLAG_OPTVERR: FLASH Option validity error flag
  - FLASH_FLAG_ECCC: FLASH one ECC error has been detected and corrected
  - FLASH_FLAG_ECCD: FLASH two ECC errors have been detected
  - FLASH_FLAG_ALL_ERRORS: FLASH All errors flags

Return value:

- None

FLASH Interrupts Definition

FLASH_IT_EOP	End of FLASH Operation Interrupt source
FLASH_IT_OPERR	Error Interrupt source

FLASH_IT_RDERR	PCROP Read Error Interrupt source
FLASH_IT_ECCC	ECC Correction Interrupt source

FLASH Keys

FLASH_KEY1	Flash key1
FLASH_KEY2	Flash key2: used with FLASH_KEY1 to unlock the FLASH registers access
FLASH_PDKEY1	Flash power down key1
FLASH_PDKEY2	Flash power down key2: used with FLASH_PDKEY1 to unlock the RUN_PD bit in FLASH_ACR
FLASH_OPTKEY1	Flash option byte key1
FLASH_OPTKEY2	Flash option byte key2: used with FLASH_OPTKEY1 to allow option bytes operations

FLASH Latency

FLASH_LATENCY_0	FLASH Zero wait state
FLASH_LATENCY_1	FLASH One wait state
FLASH_LATENCY_2	FLASH Two wait states
FLASH_LATENCY_3	FLASH Three wait states
FLASH_LATENCY_4	FLASH Four wait states
FLASH_LATENCY_5	FLASH Five wait state
FLASH_LATENCY_6	FLASH Six wait state
FLASH_LATENCY_7	FLASH Seven wait states
FLASH_LATENCY_8	FLASH Eight wait states
FLASH_LATENCY_9	FLASH Nine wait states
FLASH_LATENCY_10	FLASH Ten wait state
FLASH_LATENCY_11	FLASH Eleven wait state
FLASH_LATENCY_12	FLASH Twelve wait states
FLASH_LATENCY_13	FLASH Thirteen wait states
FLASH_LATENCY_14	FLASH Fourteen wait states
FLASH_LATENCY_15	FLASH Fifteen wait states

FLASH Option Bytes PCROP On RDP Level Type

OB_PCROP_RDP_NOT_ERASE	PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0
OB_PCROP_RDP_ERASE	PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase)

FLASH Option Bytes Read Protection

OB_RDP_LEVEL_0	
OB_RDP_LEVEL_1	
OB_RDP_LEVEL_2	Warning: When enabling read protection level 2 it's no more

possible to go back to level 1 or 0

FLASH Option Bytes Type

OPTIONBYTE_WRP WRP option byte configuration

OPTIONBYTE_RDP RDP option byte configuration

OPTIONBYTE_USER USER option byte configuration

OPTIONBYTE_PCROP PCROP option byte configuration

FLASH Option Bytes User BFB2 Mode

OB_BFB2_DISABLE Dual-bank boot disable

OB_BFB2_ENABLE Dual-bank boot enable

FLASH Option Bytes User BOR Level

OB_BOR_LEVEL_0 Reset level threshold is around 1.7V

OB_BOR_LEVEL_1 Reset level threshold is around 2.0V

OB_BOR_LEVEL_2 Reset level threshold is around 2.2V

OB_BOR_LEVEL_3 Reset level threshold is around 2.5V

OB_BOR_LEVEL_4 Reset level threshold is around 2.8V

FLASH Option Bytes User DBANK Type

OB_DBANK_128_BITS Single-bank with 128-bits data

OB_DBANK_64_BITS Dual-bank with 64-bits data

FLASH Option Bytes User Dual-bank Type

OB_DUALBANK_SINGLE 1 MB/512 kB Single-bank Flash

OB_DUALBANK_DUAL 1 MB/512 kB Dual-bank Flash

FLASH Option Bytes User IWDG Mode On Standby

OB_IWDG_STDBY_FREEZE Independent watchdog counter is frozen in Standby mode

OB_IWDG_STDBY_RUN Independent watchdog counter is running in Standby mode

FLASH Option Bytes User IWDG Mode On Stop

OB_IWDG_STOP_FREEZE Independent watchdog counter is frozen in Stop mode

OB_IWDG_STOP_RUN Independent watchdog counter is running in Stop mode

FLASH Option Bytes User IWDG Type

OB_IWDG_HW Hardware independent watchdog

OB_IWDG_SW Software independent watchdog

FLASH Option Bytes User BOOT1 Type

OB_BOOT1_SRAM Embedded SRAM1 is selected as boot space (if BOOT0=1)

OB_BOOT1_SYSTEM System memory is selected as boot space (if BOOT0=1)

FLASH Option Bytes User Reset On Shutdown

OB_SHUTDOWN_RST Reset generated when entering the shutdown mode

OB_SHUTDOWN_NORST No reset generated when entering the shutdown mode

FLASH Option Bytes User Reset On Standby

OB_STANDBY_RST	Reset generated when entering the standby mode
OB_STANDBY_NORST	No reset generated when entering the standby mode

FLASH Option Bytes User Reset On Stop

OB_STOP_RST	Reset generated when entering the stop mode
OB_STOP_NORST	No reset generated when entering the stop mode

FLASH Option Bytes User SRAM2 Parity Check Type

OB_SRAM2_PARITY_ENABLE	SRAM2 parity check enable
OB_SRAM2_PARITY_DISABLE	SRAM2 parity check disable

FLASH Option Bytes User SRAM2 Erase On Reset Type

OB_SRAM2_RST_ERASE	SRAM2 erased when a system reset occurs
OB_SRAM2_RST_NOT_ERASE	SRAM2 is not erased when a system reset occurs

FLASH Option Bytes User Type

OB_USER_BOR_LEV	BOR reset Level
OB_USER_nRST_STOP	Reset generated when entering the stop mode
OB_USER_nRST_STDBY	Reset generated when entering the standby mode
OB_USER_IWDG_SW	Independent watchdog selection
OB_USER_IWDG_STOP	Independent watchdog counter freeze in stop mode
OB_USER_IWDG_STDBY	Independent watchdog counter freeze in standby mode
OB_USER_WWDG_SW	Window watchdog selection
OB_USER_BFB2	Dual-bank boot
OB_USER_DUALBANK	Dual-Bank on 1MB or 512kB Flash memory devices
OB_USER_nBOOT1	Boot configuration
OB_USER_SRAM2_PE	SRAM2 parity check enable
OB_USER_SRAM2_RST	SRAM2 Erase when system reset
OB_USER_nRST_SHDW	Reset generated when entering the shutdown mode
OB_USER_nSWBOOT0	Software BOOT0
OB_USER_nBOOT0	nBOOT0 option bit
OB_USER_DBANK	Single bank with 128-bits data or two banks with 64-bits data

FLASH Option Bytes User WWDG Type

OB_WWDG_HW	Hardware window watchdog
OB_WWDG_SW	Software window watchdog

FLASH WRP Area

OB_WRPAREA_BANK1_AREAA	Flash Bank 1 Area A
OB_WRPAREA_BANK1_AREAB	Flash Bank 1 Area B
OB_WRPAREA_BANK2_AREAA	Flash Bank 2 Area A

OB_WRPAREA_BANK2_AREAB	Flash Bank 2 Area B

FLASH Erase Type

FLASH_TYPEERASE_PAGES	Pages erase only
FLASH_TYPEERASE_MASSERASE	Flash mass erase activation

FLASH Program Type

FLASH_TYPEPROGRAM_DOUBLEWORD	Program a double-word (64-bit) at a specified address.
FLASH_TYPEPROGRAM_FAST	Fast program a 32 row double-word (64-bit) at a specified address. And another 32 row double-word (64-bit) will be programmed
FLASH_TYPEPROGRAM_FAST_AND_LAST	Fast program a 32 row double-word (64-bit) at a specified address. And this is the last 32 row double-word (64-bit) programmed

25 HAL FLASH Extension Driver

25.1 FLASHEx Firmware driver API description

25.1.1 Flash Extended features

Comparing to other previous devices, the FLASH interface for STM32L4xx devices contains the following additional features

- Capacity up to 2 Mbyte with dual bank architecture supporting read-while-write capability (RWW)
- Dual bank memory organization
- PCROP protection for all banks

25.1.2 How to use this driver

This driver provides functions to configure and program the FLASH memory of all STM32L4xx devices. It includes

1. Flash Memory Erase functions:
   - Lock and Unlock the FLASH interface using HAL_FLASH_Unlock() and HAL_FLASH_Lock() functions
   - Erase function: Erase page, erase all sectors
   - There are two modes of erase:
     - Polling Mode using HAL_FLASHEx_Erase()
     - Interrupt Mode using HAL_FLASHEx_Erase_IT()
2. Option Bytes Programming function: Use HAL_FLASHEx_OBProgram() to:
   - Set/Reset the write protection
   - Set the Read protection Level
   - Program the user Option Bytes
   - Configure the PCROP protection
3. Get Option Bytes Configuration function: Use HAL_FLASHEx_OBGetConfig() to:
   - Get the value of a write protection area
   - Know if the read protection is activated
   - Get the value of the user Option Bytes
   - Get the value of a PCROP area

25.1.3 Extended programming operation functions

This subsection provides a set of functions allowing to manage the Extended FLASH programming operations Operations.

This section contains the following APIs:

- ***HAL_FLASHEx_Erase()***
- ***HAL_FLASHEx_Erase_IT()***
- ***HAL_FLASHEx_OBProgram()***
- ***HAL_FLASHEx_OBGetConfig()***

25.1.4 Extended specific configuration functions

This subsection provides a set of functions allowing to manage the Extended FLASH specific configurations.

This section contains the following APIs:

- *HAL_FLASHEx_ConfigLVEPin()*

25.1.5 Detailed description of functions

HAL_FLASHEx_Erase

Function name — **HAL_StatusTypeDef HAL_FLASHEx_Erase (FLASH_EraseInitTypeDef * pEraseInit, uint32_t * PageError)**

Function description — Perform a mass erase or erase the specified FLASH memory pages.

Parameters
- **pEraseInit:** pointer to an FLASH_EraseInitTypeDef structure that contains the configuration information for the erasing.
- **PageError:** : pointer to variable that contains the configuration information on faulty page in case of error (0xFFFFFFFF means that all the pages have been correctly erased)

Return values
- **HAL:** Status

HAL_FLASHEx_Erase_IT

Function name — **HAL_StatusTypeDef HAL_FLASHEx_Erase_IT (FLASH_EraseInitTypeDef * pEraseInit)**

Function description — Perform a mass erase or erase the specified FLASH memory pages with interrupt enabled.

Parameters
- **pEraseInit:** pointer to an FLASH_EraseInitTypeDef structure that contains the configuration information for the erasing.

Return values
- **HAL:** Status

HAL_FLASHEx_OBProgram

Function name — **HAL_StatusTypeDef HAL_FLASHEx_OBProgram (FLASH_OBProgramInitTypeDef * pOBInit)**

Function description — Program Option bytes.

Parameters
- **pOBInit:** pointer to an FLASH_OBInitStruct structure that contains the configuration information for the programming.

Return values
- **HAL:** Status

HAL_FLASHEx_OBGetConfig

Function name — **void HAL_FLASHEx_OBGetConfig (FLASH_OBProgramInitTypeDef * pOBInit)**

Function description — Get the Option bytes configuration.

Parameters
- **pOBInit:** pointer to an FLASH_OBInitStruct structure that contains the configuration information.

Return values
- **None:**

Notes
- The fields pOBInit->WRPArea and pOBInit->PCROPConfig should indicate which area is requested for the WRP and

PCROP, else no information will be returned

HAL_FLASHEx_ConfigLVEPin

Function name	**HAL_StatusTypeDef HAL_FLASHEx_ConfigLVEPin (uint32_t ConfigLVE)**
Function description	Configuration of the LVE pin of the Flash (managed by power controller or forced to low in order to use an external SMPS)
Parameters	• **ConfigLVE:** Configuration of the LVE pin, This parameter can be one of the following values: – FLASH_LVE_PIN_CTRL: LVE FLASH pin controlled by power controller – FLASH_LVE_PIN_FORCED: LVE FLASH pin enforced to low (external SMPS used)
Return values	• **HAL:** Status
Notes	• Before enforcing the LVE pin to low, the SOC should be in low voltage range 2 and the voltage VDD12 should be higher than 1.08V and SMPS is ON.

25.2 FLASHEx Firmware driver defines

25.2.1 FLASHEx

FLASHEx LVE pin configuration

FLASH_LVE_PIN_CTRL	LVE FLASH pin controlled by power controller
FLASH_LVE_PIN_FORCED	LVE FLASH pin enforced to low (external SMPS used)

26 HAL FLASH__RAMFUNC Generic Driver

26.1 FLASH__RAMFUNC Firmware driver API description

26.1.1 Flash RAM functions

Arm Compiler

RAM functions are defined using the toolchain options. Functions that are executed in RAM should reside in a separate source module. Using the 'Options for File' dialog you can simply change the 'Code / Const' area of a module to a memory space in physical RAM. Available memory areas are declared in the 'Target' tab of the Options for Target' dialog.

ICCARM Compiler

RAM functions are defined using a specific toolchain keyword "__ramfunc".

GNU Compiler

RAM functions are defined using a specific toolchain attribute "__attribute__((section(".RamFunc")))".

26.1.2 ramfunc functions

This subsection provides a set of functions that should be executed from RAM.

This section contains the following APIs:

- ***HAL_FLASHEx_EnableRunPowerDown()***
- ***HAL_FLASHEx_DisableRunPowerDown()***
- ***HAL_FLASHEx_OB_DBankConfig()***

26.1.3 Detailed description of functions

HAL_FLASHEx_EnableRunPowerDown

Function name	**__RAM_FUNC HAL_FLASHEx_EnableRunPowerDown (void)**
Function description	Enable the Power down in Run Mode.
Return values	• **None:**
Notes	• This function should be called and executed from SRAM memory

HAL_FLASHEx_DisableRunPowerDown

Function name	**__RAM_FUNC HAL_FLASHEx_DisableRunPowerDown (void)**
Function description	Disable the Power down in Run Mode.
Return values	• **None:**
Notes	• This function should be called and executed from SRAM memory