FS65 driver

Generated by Doxygen 1.8.14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 Driver API

Functions

- fs65_status_t FS65_Init (fs65_user_config_t ∗userConfig)

  This function runs full initialization of the SBC device.
- fs65_status_t FS65_WD_ChangeSeed (uint8_t wdSeed)

  Changes seed of LFSR used for watchdog.
- fs65_status_t FS65_SwitchAMUXchannel (fs65_amux_selection_t channelSelection)

  Switches a desired channel to the AMUX pin.
- fs65_status_t FS65_SetRegulatorState (fs65_reg_mode_t vreg, bool enable)

  Sets state (enable/disable) of the selected voltage regulator.
- fs65_status_t FS65_GetFaultErrorCounterValue (uint8_t ∗faultErrorCounterValue)

  Reads actual Fault Error Counter value.
- fs65_status_t FS65_GetMode (fs65_current_mode_t ∗currentMode, fs65_prev_mode_t ∗prevMode)

  This function gets current and previous mode of the SBC.
- fs65_status_t FS65_CheckVAUX (void)

  This function checks if VAUX is safety critical and optionally runs related diagnostics.
- fs65_status_t FS65_CheckFS1B (void)

  This function checks if FS1B has expected low level during initialization and runs related diagnostics.
- fs65_status_t FS65_ReleaseFSx (fs65_fsxb_release_t fsOutput)

  This function releases selected fail-safe output as a part of error recovery procedure.
- fs65_status_t FS65_CheckLbistAbistOk (void)

  Checks if LBIST and ABIST1 diagnostics passed.
- fs65_status_t FS65_SetLowPowerMode (bool autoWU)

  This function switches mode of the SBC to the LPOFF, optionally with automatic wake-up 1ms after transition.
- fs65_status_t FS65_RequestInterrupt (void)

  This function requests an interrupt (pulse on the INT pin).
- fs65_status_t FS65_CAN_SetMode (fs65_can_mode_t mode, bool autoDis)

  This function changes CAN mode and the automatic transition of the CAN transceiver to the low-power mode on specific events.
- fs65_status_t FS65_LIN_SetMode (fs65_lin_mode_t mode, bool autoDis)

  This function changes LIN mode and the automatic transition of the LIN transceiver to the low-power mode on specific events.
- fs65_status_t FS65_LDT_SetTimerOperation (fs65_ldt_function_t op)

This function sets operating function of the LDT.

- fs65_status_t FS65_LDT_SetTimerMode (fs65_ldt_mode_t mode)

  This function sets mode of the LDT (normal/calibration).
- fs65_status_t FS65_LDT_SetWakeUpRegSrc (fs65_ldt_wu_scr_t source)

  This function sets counter to read real-time counter or programmed value into wake-up register.
- fs65_status_t FS65_LDT_RunCounter (bool run)

  This function starts or stops the LDT counter.
- fs65_status_t FS65_LDT_SetAfterRunValue (uint16_t value)

  This function sets new after-run value for the LDT.
- fs65_status_t FS65_LDT_SetWakeUpValue (uint32_t value)

  This function sets new wake-up value for the LDT.
- fs65_status_t FS65_WD_ChangeWindow (uint8_t windowDuration)

  This function changes duration of watchdog window.
- fs65_status_t FS65_WD_Refresh (void)

  This function refreshes watchdog of the SBC device.
- fs65_status_t FS65_RequestReset (void)

  This function requests a low pulse on the RSTB (MCU reset).
- fs65_status_t FS65_RequestFSxLow (fs65_fsx_req_type_t fsxSelection)

  This function requests a low level on the selected fail-safe output.
- fs65_status_t FS65_SetOUT4 (bool level)

  This function sets level of the IO_4 when configured as an output.
- fs65_status_t FS65_ReadRegister (uint8_t address, fs65_rx_data_t *rxData)

  Performs a read from a single FS65 register.
- fs65_status_t FS65_WriteRegister (uint8_t address, uint8_t writeData, fs65_rx_data_t *rxData)

  Sends write command to the FS65.
- fs65_status_t FS65_WriteRegisters (fs65_reg_config_value_t *registers, uint8_t numOfItems)

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 FS65_CAN_SetMode()

```
fs65_status_t FS65_CAN_SetMode (
            fs65_can_mode_t mode,
            bool autoDis )
```

This function changes CAN mode and the automatic transition of the CAN transceiver to the low-power mode on specific events.

Parameters

in	*mode*	CAN mode.
in	*autoDis*	Automatic transition to the LPOFF on (CAN OT/TXD dominant/RXD recessive).

Returns

fs65_status_t Status return code.

4.1.2.2 FS65_CheckFS1B()

```
fs65_status_t FS65_CheckFS1B (
            void  )
```

This function checks if FS1B has expected low level during initialization and runs related diagnostics.

Returns

fs65_status_t Status return code.

4.1.2.3 FS65_CheckLbistAbistOk()

```
fs65_status_t FS65_CheckLbistAbistOk (
            void  )
```

Checks if LBIST and ABIST1 diagnostics passed.

Returns

fs65_status_t Status return code.

4.1.2.4 FS65_CheckVAUX()

```
fs65_status_t FS65_CheckVAUX (
            void  )
```

This function checks if VAUX is safety critical and optionally runs related diagnostics.

Returns

fs65_status_t Status return code.

4.1.2.5 FS65_GetFaultErrorCounterValue()

```
fs65_status_t FS65_GetFaultErrorCounterValue (
            uint8_t * faultErrorCounterValue )
```

Reads actual Fault Error Counter value.

Parameters

out	*faultErrorCounterValue*	Fault Error counter value storage.

Returns

fs65_status_t Status return code.

4.1.2.6 FS65_GetMode()

```
fs65_status_t FS65_GetMode (
            fs65_current_mode_t * currentMode,
            fs65_prev_mode_t * prevMode )
```

This function gets current and previous mode of the SBC.

Parameters

out	*currentMode*	Current mode of the SBC device.
out	*prevMode*	Previous mode of the SBC device.

Returns

fs65_status_t Status return code.

4.1.2.7 FS65_Init()

```
fs65_status_t FS65_Init (
            fs65_user_config_t * userConfig )
```

This function runs full initialization of the SBC device.

This function writes configuration values to selected SBC registers (i.e. content of the userConfig structure), checks results of built-in self-test diagnostics and optionally runs additional procedures to ensure correct functionality of safety critical features. The first watchdog refresh is part of the initialization procedure, the SBC should be in NORMAL mode upon the function exit.

Parameters

userConfig	Configuration structure of the SW driver.

Returns

fs65_status_t Status return code.

Remarks

Note that main register set is initialized only after power on reset. It retains its values after transition to low-power mode and back to normal. It is recommended to read-out wake-up sources and diagnostic statuses after initialization to clear all flags.

4.1.2.8 FS65_LDT_RunCounter()

```
fs65_status_t FS65_LDT_RunCounter (
            bool run )
```

This function starts or stops the LDT counter.

Parameters

in	*run*	Use true for start or false for stop the LDT counter.

Returns

fs65_status_t Status return code.

4.1.2.9 FS65_LDT_SetAfterRunValue()

```
fs65_status_t FS65_LDT_SetAfterRunValue (
            uint16_t value )
```

This function sets new after-run value for the LDT.

Parameters

in	*value*	After-run value.

Returns

fs65_status_t Status return code.

4.1.2.10 FS65_LDT_SetTimerMode()

```
fs65_status_t FS65_LDT_SetTimerMode (
            fs65_ldt_mode_t mode )
```

This function sets mode of the LDT (normal/calibration).

Parameters

in	*mode*	Mode of the LDT.

Returns

fs65_status_t Status return code.

Remarks

Resolution is 1s for normal mode and 488us for calibration mode.

4.1.2.11 FS65_LDT_SetTimerOperation()

```
fs65_status_t FS65_LDT_SetTimerOperation (
            fs65_ldt_function_t op )
```

This function sets operating function of the LDT.

Parameters

in	*op*	Operating function of the LDT.

Returns

fs65_status_t Status return code.

4.1.2.12 FS65_LDT_SetWakeUpRegSrc()

```
fs65_status_t FS65_LDT_SetWakeUpRegSrc (
            fs65_ldt_wu_scr_t source )
```

This function sets counter to read real-time counter or programmed value into wake-up register.

Parameters

in	*source*	Source for the wake-up register.

Returns

fs65_status_t Status return code.

4.1.2.13 FS65_LDT_SetWakeUpValue()

```
fs65_status_t FS65_LDT_SetWakeUpValue (
            uint32_t value )
```

This function sets new wake-up value for the LDT.

Parameters

in	*value*	Wake-up value (1 - 16777215).

Returns

fs65_status_t Status return code.

4.1.2.14 FS65_LIN_SetMode()

```
fs65_status_t FS65_LIN_SetMode (
            fs65_lin_mode_t mode,
            bool autoDis )
```

This function changes LIN mode and the automatic transition of the LIN transceiver to the low-power mode on specific events.

Parameters

in	*mode*	LIN mode.
in	*autoDis*	Automatic transition to the LPOFF on (LIN OT/TXDL dominant/RXDL recessive).

Returns

fs65_status_t Status return code.

4.1.2.15 FS65_ReadRegister()

```
fs65_status_t FS65_ReadRegister (
            uint8_t address,
            fs65_rx_data_t * rxData )
```

Performs a read from a single FS65 register.

Performs a single read register based on provided address. The response is returned in fs65_rx_data_t structure.

Parameters

in	*address*	Register address.
out	*rxData*	Structure holding the response from SBC.

Returns

fs65_status_t Status return code.

4.1.2.16 FS65_ReleaseFSx()

```
fs65_status_t FS65_ReleaseFSx (
            fs65_fsxb_release_t fsOutput )
```

This function releases selected fail-safe output as a part of error recovery procedure.

Parameters

in	*fsOutput*	Selection of fail-safe output.

Returns

fs65_status_t Status return code.

Remarks

When a fault is removed and the fault error counter changes back to level '0', a right word must be filled in the RELEASE_FSxB register. The value depends on the current WD_LFSR. LSB, MSB must be swapped, and a negative operation per bit must be applied. The RELEASE_FSxB write command must be done after the WD_LFSR read command within the same WD period.

4.1.2.17 FS65_RequestFSxLow()

```
fs65_status_t FS65_RequestFSxLow (
            fs65_fsx_req_type_t fsxSelection )
```

This function requests a low level on the selected fail-safe output.

Parameters

in	*fsxSelection*	Selection of fail-safe output.

Returns

fs65_status_t Status return code.

4.1.2.18 FS65_RequestInterrupt()

```
fs65_status_t FS65_RequestInterrupt (
            void  )
```

This function requests an interrupt (pulse on the INT pin).

Returns

fs65_status_t Status return code.

4.1.2.19 FS65_RequestReset()

```
fs65_status_t FS65_RequestReset (
            void  )
```

This function requests a low pulse on the RSTB (MCU reset).

Returns

fs65_status_t Status return code.

4.1.2.20 FS65_SetLowPowerMode()

```
fs65_status_t FS65_SetLowPowerMode (
            bool autoWU )
```

This function switches mode of the SBC to the LPOFF, optionally with automatic wake-up 1ms after transition.

Parameters

autoWU	Automatic wake-up 1 ms after transition.

Returns

fs65_status_t Status return code.

4.1.2.21 FS65_SetOUT4()

```
fs65_status_t FS65_SetOUT4 (
            bool level )
```

This function sets level of the IO_4 when configured as an output.

Parameters

in	*level*	Level of IO_4 when configured as output. True for high level, false for low level.

Returns

fs65_status_t Status return code.

Remarks

Note that this function automatically enables IO_4 to be used as output.

4.1.2.22 FS65_SetRegulatorState()

```
fs65_status_t FS65_SetRegulatorState (
            fs65_reg_mode_t vreg,
            bool enable )
```

Sets state (enable/disable) of the selected voltage regulator.

Parameters

in	*vreg*	Voltage regulator enum (VCAN, VAUX, VCCA, VCORE).
in	*enable*	State (enable = true / disable = false).

Returns

fs65_status_t Status return code.

4.1.2.23 FS65_SwitchAMUXchannel()

```
fs65_status_t FS65_SwitchAMUXchannel (
            fs65_amux_selection_t channelSelection )
```

Switches a desired channel to the AMUX pin.

Parameters

in	*channelSelection*	Selected channel to be delivered to AMUX pin.

Returns

fs65_status_t Status return code.

4.1.2.24 FS65_WD_ChangeSeed()

```
fs65_status_t FS65_WD_ChangeSeed (
            uint8_t wdSeed )
```

Changes seed of LFSR used for watchdog.

The watchdog seed can be changed just during the INIT_FS phase (for challenger WD) or during the OPEN watchdog window (for simple WD). Timing is up to the application!

Parameters

in	*wdSeed*	Watchdog LFSR seed.

Returns

fs65_status_t Status return code.

4.1.2.25 FS65_WD_ChangeWindow()

```
fs65_status_t FS65_WD_ChangeWindow (
            uint8_t windowDuration )
```

This function changes duration of watchdog window.

Parameters

in	*windowDuration*	Watchdog window duration. Use any FS65_W_FS_WD_WINDOW_∗ macro from sbc_fs65_map.h.

Returns

fs65_status_t Status return code.

4.1.2.26 FS65_WD_Refresh()

```
fs65_status_t FS65_WD_Refresh (
            void  )
```

This function refreshes watchdog of the SBC device.

Returns

fs65_status_t Status return code.

Remarks

MCU internally computes monitoring result based on a received challenge token (LFSR state) and sends it back as an answer.

4.1.2.27 FS65_WriteRegister()

```
fs65_status_t FS65_WriteRegister (
            uint8_t address,
            uint8_t writeData,
            fs65_rx_data_t * rxData )
```

Sends write command to the FS65.

Parameters

in	*address*	Register address.
in	*writeData*	Register write value.
out	*rxData*	Diagnostic data returned by the SBC.

Returns

fs65_status_t Status return code.

4.1.2.28 FS65_WriteRegisters()

```
fs65_status_t FS65_WriteRegisters (
            fs65_reg_config_value_t * registers,
            uint8_t numOfItems )
```

Writes set of configuration registers values.

If registers is NULL or numOfItems is 0, function just returns OK status code.

Parameters

in	*registers*	Pointer to array of register configuration values.
in	*numOfItems*	Number of items.

Returns

fs65_status_t Status return code.

4.2 Defines for SBC features

Macros

- #define FS65_FEATURE_FS1B

  FS1B functionality.
- #define FS65_FEATURE_CAN

  CAN functionality.
- #define FS65_FEATURE_LIN

  LIN functionality.
- #define FS65_FEATURE_LDT

  LDT functionality.

4.2.1 Detailed Description

Note

If some feature is not supported by the SBC, it can be disabled by the particular define renaming/deletion.

4.3 Enums definition

Enumerations

- enum fs65_reg_mode_t {
  fs65VCan = FS65_R_M_VCAN_EN_SHIFT, fs65Aux = FS65_R_M_VAUX_EN_SHIFT, fs65Vcca = FS65_↩
  R_M_VCCA_EN_SHIFT, fs65Vcore = FS65_R_M_VCORE_EN_SHIFT,
  fs65Vkam = 0xFF }

  Voltage outputs. Can be used with function FS65_SetRegulatorState().
- enum fs65_amux_selection_t {
  fs65AmuxVref = FS65_RW_M_AMUX_VREF, fs65AmuxVsnsWide = FS65_RW_M_AMUX_VSNS_W,
  fs65AmuxIO_0Wide = FS65_RW_M_AMUX_IO_0_W, fs65AmuxIO_5Wide = FS65_RW_M_AMUX_IO↩
  _5_W,
  fs65AmuxVsnsTight = FS65_RW_M_AMUX_VSNS_T, fs65AmuxIO_0Tight = FS65_RW_M_AMUX_IO_0_T,
  fs65AmuxIO_5TightDivVkam = FS65_RW_M_AMUX_IO_5_T, fs65AmuxDieTempSensor = FS65_RW_M↩
  _AMUX_TEMP_SENSOR }

  AMUX channel selection. Can be used with function FS65_SwitchAMUXchannel().
- enum fs65_can_mode_t { fs65CanModeSleepNoWakeup = FS65_RW_M_CAN_MODE_SLN_WU,
  fs65CanModeListenOnly = FS65_RW_M_CAN_MODE_LISTEN_ONLY, fs65CanModeSleepWakeup = F↩
  S65_RW_M_CAN_MODE_SL_WU, fs65CanModeNormal = FS65_RW_M_CAN_MODE_NORMAL }

  CAN mode. Can be used with function FS65_CAN_SetMode().
- enum fs65_lin_mode_t { fs65LinModeSleepNoWakeup = FS65_RW_M_LIN_MODE_SLN_WU, fs65LinModeListenOnly = FS65_RW_M_LIN_MODE_LISTEN_ONLY, fs65LinModeSleepWakeup = FS65_RW_M_LIN_MODE_SL↩
  _WU, fs65LinModeNormal = FS65_RW_M_LIN_MODE_NORMAL }

  LIN mode. Can be used with function FS65_LIN_SetMode().
- enum fs65_ldt_function_t {
  fs65LdtFunc1 = FS65_RW_M_F2_F0_FUNCTION1, fs65LdtFunc2 = FS65_RW_M_F2_F0_FUNCTION2,
  fs65LdtFunc3 = FS65_RW_M_F2_F0_FUNCTION3, fs65LdtFunc4 = FS65_RW_M_F2_F0_FUNCTION4,
  fs65LdtFunc5 = FS65_RW_M_F2_F0_FUNCTION5 }

  LDT operating function. Can be used with function FS65_LDT_SetTimerOperation().
- enum fs65_ldt_mode_t { fs65LdtModeCalibration = FS65_RW_M_MODE_CALIBRATION, fs65LdtModeNormal = FS65_RW_M_MODE_NORMAL }

  LDT mode. Can be used with function FS65_LDT_SetTimerMode().
- enum fs65_ldt_wu_scr_t { fs65LdtWakeupProg = FS65_RW_M_REG_SE_PROGRAMMED_REG, fs65LdtWakeupRTC = FS65_RW_M_REG_SE_RTC_REG }

  Wake-up register source. Can be used with function FS65_LDT_SetWakeUpRegSrc().
- enum fs65_fsx_req_type_t { fs65ReqFS0B = FS65_W_FS_FS0B_REQ_FS0B_REQ, fs65ReqFS1BDelay = FS65_W_FS_FS1B_DLY_REQ_FS1B_REQ, fs65ReqFS1B = FS65_W_FS_FS1B_REQ_FS1B_REQ }

  Fail-safe output selection for its low level request. Can be used with function FS65_RequestFSxLow().
- enum fs65_fsxb_release_t { fs65ReleaseFs0b = 0x60, fs65ReleaseFs1b = 0xC0, fs65ReleaseFs0bFs1b = 0xA0 }

  FSxb pin release options.
- enum fs65_status_t { fs65StatusOk = 0U, fs65StatusError = 1U }

  Status return codes.
- enum fs65_command_t { fs65RegRead, fs65RegWrite }

  Command type.
- enum fs65_parity_t { fs65ParityOdd, fs65ParityEven }

  Parity type.

Enums related to functions used to handle SBC mode.

- enum fs65_prev_mode_t { fs65PrevModePOR, fs65PrevModeDFS, fs65PrevModeLPOFF }

  Previous SBC state.
- enum fs65_current_mode_t { fs65ModeUnknown, fs65ModeInit, fs65ModeNormal }

  Actual SBC state.

4.3.1 Detailed Description

4.3.2 Enumeration Type Documentation

4.3.2.1 fs65_amux_selection_t

`enum fs65_amux_selection_t`

AMUX channel selection. Can be used with function FS65_SwitchAMUXchannel().

Enumerator

fs65AmuxVref	V_REF.
fs65AmuxVsnsWide	V_SNS wide range.
fs65AmuxIO_0Wide	IO_0 wide range.
fs65AmuxIO_5Wide	IO_5 wide range.
fs65AmuxVsnsTight	V_SNS tight range.
fs65AmuxIO_0Tight	IO_0 tight range.
fs65AmuxIO_5TightDivVkam	IO_5 tight range/VKAM.
fs65AmuxDieTempSensor	Die Temperature Sensor.

4.3.2.2 fs65_can_mode_t

`enum fs65_can_mode_t`

CAN mode. Can be used with function FS65_CAN_SetMode().

Enumerator

fs65CanModeSleepNoWakeup	Sleep/no wake-up capability.
fs65CanModeListenOnly	Listen only.
fs65CanModeSleepWakeup	Sleep/wake-up capability.
fs65CanModeNormal	Normal operation mode.

4.3.2.3 fs65_command_t

`enum fs65_command_t`

Command type.

Enumerator

fs65RegRead	Register Read.
fs65RegWrite	Register Write.

4.3.2.4 fs65_current_mode_t

`enum fs65_current_mode_t`

Actual SBC state.

Enumerator

fs65ModeUnknown	Current SBC mode is unknown.
fs65ModeInit	Current SBC mode is INIT.
fs65ModeNormal	Current SBC mode is NORMAL.

4.3.2.5 fs65_fsx_req_type_t

`enum fs65_fsx_req_type_t`

Fail-safe output selection for its low level request. Can be used with function FS65_RequestFSxLow().

Enumerator

fs65ReqFS0B	Request FS0B assertion.
fs65ReqFS1BDelay	Request FS1B assertion with tDELAY controlled by the backup delay (open S1).
fs65ReqFS1B	Request FS1B assertion with immediate assertion, no delay.

4.3.2.6 fs65_fsxb_release_t

`enum fs65_fsxb_release_t`

FSxb pin release options.

Enumerator

fs65ReleaseFs0b	Release FS0b pin only.
fs65ReleaseFs1b	Release FS1b pin only.
fs65ReleaseFs0bFs1b	Release both FS0b and FS1b pins.

4.3.2.7 fs65_ldt_function_t

`enum fs65_ldt_function_t`

LDT operating function. Can be used with function FS65_LDT_SetTimerOperation().

Enumerator

fs65LdtFunc1	In normal mode count and generate flag or INT when counter reaches the after run value.
fs65LdtFunc2	In normal mode count until after run value is reached, then enters in LPOFF.
fs65LdtFunc3	In normal mode count until after run value is reached, then enters in LPOFF. Once in LPOFF, count until wake-up value is reached and wake-up.
fs65LdtFunc4	In LPOFF, count until wake-up value is reached and wake-up.
fs65LdtFunc5	In LPOFF, count and do not wake-up. Counter value is stored in wake-up register.

4.3.2.8 fs65_ldt_mode_t

`enum fs65_ldt_mode_t`

LDT mode. Can be used with function FS65_LDT_SetTimerMode().

Enumerator

fs65LdtModeCalibration	Calibration mode (488 us resolution).
fs65LdtModeNormal	Normal mode (1 s resolution).

4.3.2.9 fs65_ldt_wu_scr_t

`enum fs65_ldt_wu_scr_t`

Wake-up register source. Can be used with function FS65_LDT_SetWakeUpRegSrc().

Enumerator

fs65LdtWakeupProg	Read programmed wake-up register.
fs65LdtWakeupRTC	Read real time counter into wake-up register (after counter is stopped with LDT_ENABLE bit).

4.3.2.10 fs65_lin_mode_t

`enum fs65_lin_mode_t`

LIN mode. Can be used with function FS65_LIN_SetMode().

Enumerator

fs65LinModeSleepNoWakeup	Sleep/no wake-up capability.
fs65LinModeListenOnly	Listen only.
fs65LinModeSleepWakeup	Sleep/wake-up capability.
fs65LinModeNormal	Normal operation mode.

4.3.2.11 fs65_parity_t

`enum fs65_parity_t`

Parity type.

Enumerator

fs65ParityOdd	Number of 1s is odd.
fs65ParityEven	Number of 1s is even.

4.3.2.12 fs65_prev_mode_t

`enum fs65_prev_mode_t`

Previous SBC state.

Enumerator

fs65PrevModePOR	Previous SBC mode was POR (power on reset).
fs65PrevModeDFS	Resume from deep fail-safe mode.
fs65PrevModeLPOFF	Resume from LPOFF mode.

4.3.2.13 fs65_reg_mode_t

`enum fs65_reg_mode_t`

Voltage outputs. Can be used with function FS65_SetRegulatorState().

Enumerator

fs65VCan	VCAN control.
fs65Aux	VAUX control (switch off not recommended if VAUX is safety critical).
fs65Vcca	VCCA control (switch off not recommended if VCCA is safety critical).
fs65Vcore	VCORE control (switch off not recommended if VCORE is safety critical).
fs65Vkam	VKAM control.

4.3.2.14 fs65_status_t

```
enum fs65_status_t
```

Status return codes.

Enumerator

fs65StatusOk	No error.
fs65StatusError	Error.

4.4 Struct definitions

Data Structures

- struct fs65_tx_data_t

  Structure representing transmit data frame.
- struct fs65_rx_data_t

  Structure representing received data frame.
- struct fs65_reg_config_value_t

  Structure representing configuration value of one register.
- struct fs65_user_config_t

  Structure for FS65 user configuration.

4.4.1 Detailed Description

4.5 MCU specific functions

Functions

- fs65_status_t MCU_SPI_TransferData (uint8_t *txFrame, uint8_t *rxFrame)

  This function transfers single frame through blocking SPI communication in both directions. MCU specific.
- void MCU_WaitUs (uint16_t delay)

  This function waits for specified amount of microseconds.

4.5.1 Detailed Description

Attention

Functions in this group must be implemented by the user.

4.5.2 Function Documentation

4.5.2.1 MCU_SPI_TransferData()

```
fs65_status_t MCU_SPI_TransferData (
            uint8_t * txFrame,
            uint8_t * rxFrame )
```

This function transfers single frame through blocking SPI communication in both directions. MCU specific.

This function must be implemented if SPI communication is used.

Warning

This function must be implemented as blocking as there is not synchronization mechanism implemented in the driver.

Parameters

in	*txFrame*	Frame to be send.
out	*rxFrame*	Received frame.

Returns

fs65_status_t Status return code.

4.5.2.2 MCU_WaitUs()

```
void MCU_WaitUs (
            uint16_t delay )
```

This function waits for specified amount of microseconds.

Parameters

in	*delay*	Delay in microseconds.

Chapter 5

Data Structure Documentation

5.1 fs65_reg_config_value_t Struct Reference

Structure representing configuration value of one register.

`#include <sbc_fs65_common.h>`

Data Fields

- uint8_t address
  Register address.
- uint8_t value
  Value of the register.
- uint8_t readMask
  Mask used for register read value check.
- bool isSecured
  True if register uses secure bits.

5.1.1 Detailed Description

Structure representing configuration value of one register.

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

- Sources/FS65_driver/sbc_fs65_common.h

5.2 fs65_rx_data_t Struct Reference

Structure representing received data frame.

`#include <sbc_fs65_common.h>`

Data Fields

- uint8_t deviceStatus

  A device status is returned into this byte after a successful transfer.
- uint8_t deviceStatusEx

  Extended diagnostics. Sent by Main or Fail-safe if secured register is accessed.
- uint8_t readData

  Content of a read register.

5.2.1 Detailed Description

Structure representing received data frame.

5.2.2 Field Documentation

5.2.2.1 deviceStatusEx

```
uint8_t fs65_rx_data_t::deviceStatusEx
```

Extended diagnostics. Sent by Main or Fail-safe if secured register is accessed.

Diagnostics content is saved in 4 highest bits.

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

- Sources/FS65_driver/sbc_fs65_common.h

5.3 fs65_tx_data_t Struct Reference

Structure representing transmit data frame.

```
#include <sbc_fs65_common.h>
```

Data Fields

- bool isFailSafe

  Main/Fail Safe register selection.
- bool isSecured

  true if the SPI command is secured.
- uint8_t registerAddress

  Register address.
- fs65_command_t commandType

  Command type (R/W).
- uint8_t writeData

  Data to be written to the register.

5.3.1 Detailed Description

Structure representing transmit data frame.

5.3.2 Field Documentation

5.3.2.1 writeData

```
uint8_t fs65_tx_data_t::writeData
```

Data to be written to the register.

If commandType is "read", this value will be ignored.

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

- Sources/FS65_driver/sbc_fs65_common.h

5.4 fs65_user_config_t Struct Reference

Structure for FS65 user configuration.

```
#include <sbc_fs65_common.h>
```

Data Fields

- fs65_reg_config_value_t * initMainRegs
  INIT main registers.
- uint8_t initMainRegsCount
  Number of INIT main registers.
- fs65_reg_config_value_t * initFailSafeRegs
  INIT_FS registers.
- uint8_t initFailSafeRegsCount
  Number of INIT_FS registers.
- fs65_reg_config_value_t * nonInitRegs
  Non-init registers.
- uint8_t nonInitRegsCount
  Number of non-init registers.
- uint8_t * initIntReg
  Pointer to INIT_INT register value.

5.4.1 Detailed Description

Structure for FS65 user configuration.

This structure is used as a parameter for FS65_Init() function.

Note

If any of the struct member is set to NULL (pointer values), such a member will be ignored and no write operation will be performed for this group of registers during the INIT phase.

5.4.2 Field Documentation

5.4.2.1 initFailSafeRegs

```
fs65_reg_config_value_t* fs65_user_config_t::initFailSafeRegs
```

INIT_FS registers.

Array of register configuration values for INIT FS registers. Following registers can be configured:

- INIT_FS1B_TIMING
- INIT_SUPERVISOR
- INIT_FAULT
- INIT_FSSM
- INIT_SF_IMPACT
- INIT_WD_CNT
- INIT_VCORE_OVUV_IMPACT
- INIT_VCCA_OVUV_IMPACT
- INIT_VAUX_OVUV_IMPACT

5.4.2.2 initIntReg

```
uint8_t* fs65_user_config_t::initIntReg
```

Pointer to INIT_INT register value.

If NULL, actual register value is read and written back.

5.4.2.3 initMainRegs

```
fs65_reg_config_value_t* fs65_user_config_t::initMainRegs
```

INIT main registers.

Array of register configuration values for INIT main registers. Following registers can be configured:

- INIT_VREG
- INIT_WU1
- INIT_WU2
- INIT_INH_INT

Note

INIT_INT register should not be present in this array as its writing causes transition from INIT MAIN to NORMAL MODE. INIT_INT register value can be set by initIntReg value (part of this structure).

5.4.2.4 nonInitRegs

```
fs65_reg_config_value_t* fs65_user_config_t::nonInitRegs
```

Non-init registers.

Configuration of the rest of the registers.

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

- Sources/FS65_driver/sbc_fs65_common.h

Chapter 6

File Documentation

6.1 Sources/FS65_driver/sbc_fs65.c File Reference

Driver functions for the FS65/FS45 SBC.

```
#include "sbc_fs65.h"
#include "sbc_fs65_communication.h"
#include "sbc_fs65_assert.h"
```

Macros

- #define FS65_SECURE_WRITE_SHIFT 4U

  Shift of the register write value if secured bits are used.
- #define FS65_IS_IN_RANGE(val, min, max) (((val) >= (min)) && ((val) <= (max)))

  Returns true if value VAL is in the range defined by MIN and MAX values (range includes the border values).

Functions

- fs65_status_t FS65_Init (fs65_user_config_t *userConfig)

  This function runs full initialization of the SBC device.
- fs65_status_t FS65_WD_Refresh (void)

  This function refreshes watchdog of the SBC device.
- fs65_status_t FS65_SwitchAMUXchannel (fs65_amux_selection_t channelSelection)

  Switches a desired channel to the AMUX pin.
- fs65_status_t FS65_SetRegulatorState (fs65_reg_mode_t vreg, bool enable)

  Sets state (enable/disable) of the selected voltage regulator.
- fs65_status_t FS65_GetFaultErrorCounterValue (uint8_t *faultErrorCounterValue)

  Reads actual Fault Error Counter value.
- fs65_status_t FS65_GetMode (fs65_current_mode_t *currentMode, fs65_prev_mode_t *prevMode)

  This function gets current and previous mode of the SBC.
- fs65_status_t FS65_CheckVAUX (void)

  This function checks if VAUX is safety critical and optionally runs related diagnostics.
- fs65_status_t FS65_CheckFS1B (void)

  This function checks if FS1B has expected low level during initialization and runs related diagnostics.

- fs65_status_t FS65_ReleaseFSx (fs65_fsxb_release_t fsOutput)

  This function releases selected fail-safe output as a part of error recovery procedure.
- fs65_status_t FS65_CheckLbistAbistOk (void)

  Checks if LBIST and ABIST1 diagnostics passed.
- fs65_status_t FS65_SetLowPowerMode (bool autoWU)

  This function switches mode of the SBC to the LPOFF, optionally with automatic wake-up 1ms after transition.
- fs65_status_t FS65_RequestInterrupt (void)

  This function requests an interrupt (pulse on the INT pin).
- fs65_status_t FS65_CAN_SetMode (fs65_can_mode_t mode, bool autoDis)

  This function changes CAN mode and the automatic transition of the CAN transceiver to the low-power mode on specific events.
- fs65_status_t FS65_LIN_SetMode (fs65_lin_mode_t mode, bool autoDis)

  This function changes LIN mode and the automatic transition of the LIN transceiver to the low-power mode on specific events.
- fs65_status_t FS65_LDT_SetTimerOperation (fs65_ldt_function_t op)

  This function sets operating function of the LDT.
- fs65_status_t FS65_LDT_SetTimerMode (fs65_ldt_mode_t mode)

  This function sets mode of the LDT (normal/calibration).
- fs65_status_t FS65_LDT_SetWakeUpRegSrc (fs65_ldt_wu_scr_t source)

  This function sets counter to read real-time counter or programmed value into wake-up register.
- fs65_status_t FS65_LDT_RunCounter (bool run)

  This function starts or stops the LDT counter.
- fs65_status_t FS65_LDT_SetAfterRunValue (uint16_t value)

  This function sets new after-run value for the LDT.
- fs65_status_t FS65_LDT_SetWakeUpValue (uint32_t value)

  This function sets new wake-up value for the LDT.
- fs65_status_t FS65_WD_ChangeWindow (uint8_t windowDuration)

  This function changes duration of watchdog window.
- fs65_status_t FS65_WD_ChangeSeed (uint8_t wdSeed)

  Changes seed of LFSR used for watchdog.
- fs65_status_t FS65_RequestReset (void)

  This function requests a low pulse on the RSTB (MCU reset).
- fs65_status_t FS65_RequestFSxLow (fs65_fsx_req_type_t fsxSelection)

  This function requests a low level on the selected fail-safe output.
- fs65_status_t FS65_SetOUT4 (bool level)

  This function sets level of the IO_4 when configured as an output.

6.1.1 Detailed Description

Driver functions for the FS65/FS45 SBC.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.1.2 Macro Definition Documentation

6.1.2.1 FS65_IS_IN_RANGE

```
#define FS65_IS_IN_RANGE(
              val,
              min,
              max ) (((val) >= (min)) && ((val) <= (max)))
```

Returns true if value VAL is in the range defined by MIN and MAX values (range includes the border values).

Parameters

val	Comparison value.
min	Minimal value of the range.
max	Maximal value of the range.

Returns

True if value is the range. False otherwise.

6.2 Sources/FS65_driver/sbc_fs65.h File Reference

FS65/FS45 driver interface.

```
#include "sbc_fs65_common.h"
#include "sbc_fs65_map.h"
```

Macros

- #define FS65_WD_SEED_DEFAULT 0xB2U

  Watchdog seed default value.

Enumerations

- enum fs65_reg_mode_t {
  fs65VCan = FS65_R_M_VCAN_EN_SHIFT, fs65Aux = FS65_R_M_VAUX_EN_SHIFT, fs65Vcca = FS65_↩
  R_M_VCCA_EN_SHIFT, fs65Vcore = FS65_R_M_VCORE_EN_SHIFT,
  fs65Vkam = 0xFF }

  Voltage outputs. Can be used with function FS65_SetRegulatorState().
- enum fs65_amux_selection_t {
  fs65AmuxVref = FS65_RW_M_AMUX_VREF, fs65AmuxVsnsWide = FS65_RW_M_AMUX_VSNS_W, fs65AmuxIO_0Wide = FS65_RW_M_AMUX_IO_0_W, fs65AmuxIO_5Wide = FS65_RW_M_AMUX_IO↩
  _5_W,
  fs65AmuxVsnsTight = FS65_RW_M_AMUX_VSNS_T, fs65AmuxIO_0Tight = FS65_RW_M_AMUX_IO_0_T, fs65AmuxIO_5TightDivVkam = FS65_RW_M_AMUX_IO_5_T, fs65AmuxDieTempSensor = FS65_RW_M↩
  _AMUX_TEMP_SENSOR }

AMUX channel selection. Can be used with function FS65_SwitchAMUXchannel().

- enum fs65_can_mode_t { fs65CanModeSleepNoWakeup = FS65_RW_M_CAN_MODE_SLN_WU, fs65CanModeListenOnly = FS65_RW_M_CAN_MODE_LISTEN_ONLY, fs65CanModeSleepWakeup = FS65_RW_M_CAN_MODE_SL_WU, fs65CanModeNormal = FS65_RW_M_CAN_MODE_NORMAL }

  CAN mode. Can be used with function FS65_CAN_SetMode().
- enum fs65_lin_mode_t { fs65LinModeSleepNoWakeup = FS65_RW_M_LIN_MODE_SLN_WU, fs65LinModeListenOnly = FS65_RW_M_LIN_MODE_LISTEN_ONLY, fs65LinModeSleepWakeup = FS65_RW_M_LIN_MODE_SL_WU, fs65LinModeNormal = FS65_RW_M_LIN_MODE_NORMAL }

  LIN mode. Can be used with function FS65_LIN_SetMode().
- enum fs65_ldt_function_t { fs65LdtFunc1 = FS65_RW_M_F2_F0_FUNCTION1, fs65LdtFunc2 = FS65_RW_M_F2_F0_FUNCTION2, fs65LdtFunc3 = FS65_RW_M_F2_F0_FUNCTION3, fs65LdtFunc4 = FS65_RW_M_F2_F0_FUNCTION4, fs65LdtFunc5 = FS65_RW_M_F2_F0_FUNCTION5 }

  LDT operating function. Can be used with function FS65_LDT_SetTimerOperation().
- enum fs65_ldt_mode_t { fs65LdtModeCalibration = FS65_RW_M_MODE_CALIBRATION, fs65LdtModeNormal = FS65_RW_M_MODE_NORMAL }

  LDT mode. Can be used with function FS65_LDT_SetTimerMode().
- enum fs65_ldt_wu_scr_t { fs65LdtWakeupProg = FS65_RW_M_REG_SE_PROGRAMMED_REG, fs65LdtWakeupRTC = FS65_RW_M_REG_SE_RTC_REG }

  Wake-up register source. Can be used with function FS65_LDT_SetWakeUpRegSrc().
- enum fs65_fsx_req_type_t { fs65ReqFS0B = FS65_W_FS_FS0B_REQ_FS0B_REQ, fs65ReqFS1BDelay = FS65_W_FS_FS1B_DLY_REQ_FS1B_REQ, fs65ReqFS1B = FS65_W_FS_FS1B_REQ_FS1B_REQ }

  Fail-safe output selection for its low level request. Can be used with function FS65_RequestFSxLow().

Functions

- fs65_status_t FS65_Init (fs65_user_config_t *userConfig)

  This function runs full initialization of the SBC device.
- fs65_status_t FS65_WD_ChangeSeed (uint8_t wdSeed)

  Changes seed of LFSR used for watchdog.
- fs65_status_t FS65_SwitchAMUXchannel (fs65_amux_selection_t channelSelection)

  Switches a desired channel to the AMUX pin.
- fs65_status_t FS65_SetRegulatorState (fs65_reg_mode_t vreg, bool enable)

  Sets state (enable/disable) of the selected voltage regulator.
- fs65_status_t FS65_GetFaultErrorCounterValue (uint8_t *faultErrorCounterValue)

  Reads actual Fault Error Counter value.
- fs65_status_t FS65_GetMode (fs65_current_mode_t *currentMode, fs65_prev_mode_t *prevMode)

  This function gets current and previous mode of the SBC.
- fs65_status_t FS65_CheckVAUX (void)

  This function checks if VAUX is safety critical and optionally runs related diagnostics.
- fs65_status_t FS65_CheckFS1B (void)

  This function checks if FS1B has expected low level during initialization and runs related diagnostics.
- fs65_status_t FS65_ReleaseFSx (fs65_fsxb_release_t fsOutput)

  This function releases selected fail-safe output as a part of error recovery procedure.
- fs65_status_t FS65_CheckLbistAbistOk (void)

  Checks if LBIST and ABIST1 diagnostics passed.
- fs65_status_t FS65_SetLowPowerMode (bool autoWU)

  This function switches mode of the SBC to the LPOFF, optionally with automatic wake-up 1ms after transition.
- fs65_status_t FS65_RequestInterrupt (void)

  This function requests an interrupt (pulse on the INT pin).
- fs65_status_t FS65_CAN_SetMode (fs65_can_mode_t mode, bool autoDis)

This function changes CAN mode and the automatic transition of the CAN transceiver to the low-power mode on specific events.

- fs65_status_t FS65_LIN_SetMode (fs65_lin_mode_t mode, bool autoDis)

  This function changes LIN mode and the automatic transition of the LIN transceiver to the low-power mode on specific events.
- fs65_status_t FS65_LDT_SetTimerOperation (fs65_ldt_function_t op)

  This function sets operating function of the LDT.
- fs65_status_t FS65_LDT_SetTimerMode (fs65_ldt_mode_t mode)

  This function sets mode of the LDT (normal/calibration).
- fs65_status_t FS65_LDT_SetWakeUpRegSrc (fs65_ldt_wu_scr_t source)

  This function sets counter to read real-time counter or programmed value into wake-up register.
- fs65_status_t FS65_LDT_RunCounter (bool run)

  This function starts or stops the LDT counter.
- fs65_status_t FS65_LDT_SetAfterRunValue (uint16_t value)

  This function sets new after-run value for the LDT.
- fs65_status_t FS65_LDT_SetWakeUpValue (uint32_t value)

  This function sets new wake-up value for the LDT.
- fs65_status_t FS65_WD_ChangeWindow (uint8_t windowDuration)

  This function changes duration of watchdog window.
- fs65_status_t FS65_WD_Refresh (void)

  This function refreshes watchdog of the SBC device.
- fs65_status_t FS65_RequestReset (void)

  This function requests a low pulse on the RSTB (MCU reset).
- fs65_status_t FS65_RequestFSxLow (fs65_fsx_req_type_t fsxSelection)

  This function requests a low level on the selected fail-safe output.
- fs65_status_t FS65_SetOUT4 (bool level)

  This function sets level of the IO_4 when configured as an output.

6.2.1 Detailed Description

FS65/FS45 driver interface.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.3 Sources/FS65_driver/sbc_fs65_assert.h File Reference

Assertion macro definition, for debugging purposes.

```
#include <stdbool.h>
```

Macros

- #define FS_ASSERT(x) ((void)0)

6.3.1 Detailed Description

Assertion macro definition, for debugging purposes.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.3.2 Macro Definition Documentation

6.3.2.1 FS_ASSERT

```
#define FS_ASSERT(
              x ) ((void)0)
```

Note

MISRA-C:2012 violations

Violates MISRA 2012 Advisory Rule 2.5, global macro not referenced. The macro is defined to be used by drivers to validate input parameters and can be disabled.

Violates MISRA 2012 Advisory Directive 4.9, Function-like macro defined. The macros are used to validate input parameters to driver functions.

Note

Error detection and reporting

FS65 driver can use a mechanism to validate data coming from upper software layers (application code) by performing a number of checks on input parameters' range or other invariants that can be statically checked (not dependent on runtime conditions). A failed validation is indicative of a software bug in application code, therefore it is important to use this mechanism during development.

The validation is performed by using FS_ASSERT macro. A default implementation of this macro is provided in this file. However, application developers can provide their own implementation in a custom file. This requires defining the CUSTOM_DEVASSERT symbol with the specific file name in the project configuration (for example: -DCUSTOM_DEVASSERT="custom_devassert.h")

The default implementation accommodates two behaviors, based on DEV_ERROR_DETECT symbol:

- When DEV_ERROR_DETECT symbol is defined in the project configuration (for example: -DDEV_ERRO↩ R_DETECT), the validation performed by the FS_ASSERT macro is enabled, and a failed validation triggers an infinite loop. This configuration is recommended for development environments, as it prevents further execution and allows investigating potential problems from the point of error detection.
- When DEV_ERROR_DETECT symbol is not defined, the FS_ASSERT macro is implemented as no-op, therefore disabling all validations. This configuration can be used to eliminate the overhead of development-time checks.

It is the application developer's responsibility to decide the error detection strategy for production code: one can opt to disable development-time checking altogether (by not defining DEV_ERROR_DETECT symbol), or one can opt to keep the checks in place and implement a recovery mechanism in case of a failed validation, by defining CUSTOM_DEVASSERT to point to the file containing the custom implementation.

6.4 Sources/FS65_driver/sbc_fs65_common.h File Reference

Driver common structures, enums, macros and configuration values.

```
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
```

Data Structures

- struct fs65_tx_data_t

  Structure representing transmit data frame.
- struct fs65_rx_data_t

  Structure representing received data frame.
- struct fs65_reg_config_value_t

  Structure representing configuration value of one register.
- struct fs65_user_config_t

  Structure for FS65 user configuration.

Macros

- #define FS65_FEATURE_FS1B

  FS1B functionality.
- #define FS65_FEATURE_CAN

  CAN functionality.
- #define FS65_FEATURE_LIN

  LIN functionality.
- #define FS65_FEATURE_LDT

  LDT functionality.
- #define FS65_IS_REG_FAILSAFE(address) ((address) & 0x20)

  Returns true if register is fail-safe.

Bitwise operations used by this driver.

- #define FS65_BO_SETVAL(data, val, mask) (((data) & ∼(mask)) | ((val) & (mask)))

  This macro updates value of bits specified by the mask. It is assumed that value is already shifted. Write value is masked also.
- #define FS65_BO_SETVAL_EXT(data, value, mask, shift) (((data) & ∼(mask << shift)) | (((value) & (mask)) << (shift)))

  This macro updates value of bits specified by the mask. Additionally range check on the value is performed. It is assumed that value is not shifted.
- #define FS65_BO_GETVAL(data, mask, shift) ((data) & (mask) << (shift))

  This macro returns value specified by the mask.
- #define FS65_BO_GET_REG_VALUE(value, mask, shift) (((value) & (mask)) >> (shift))

  Macro for getting value from register.

Enumerations

- enum fs65_fsxb_release_t { fs65ReleaseFs0b = 0x60, fs65ReleaseFs1b = 0xC0, fs65ReleaseFs0bFs1b = 0xA0 }

  FSxb pin release options.
- enum fs65_status_t { fs65StatusOk = 0U, fs65StatusError = 1U }

  Status return codes.
- enum fs65_command_t { fs65RegRead, fs65RegWrite }

  Command type.
- enum fs65_parity_t { fs65ParityOdd, fs65ParityEven }

  Parity type.

Enums related to functions used to handle SBC mode.

- enum fs65_prev_mode_t { fs65PrevModePOR, fs65PrevModeDFS, fs65PrevModeLPOFF }

  Previous SBC state.
- enum fs65_current_mode_t { fs65ModeUnknown, fs65ModeInit, fs65ModeNormal }

  Actual SBC state.

Functions

- fs65_status_t MCU_SPI_TransferData (uint8_t *txFrame, uint8_t *rxFrame)

  This function transfers single frame through blocking SPI communication in both directions. MCU specific.
- void MCU_WaitUs (uint16_t delay)

  This function waits for specified amount of microseconds.

6.4.1 Detailed Description

Driver common structures, enums, macros and configuration values.

This header file also contains prototypes of functions that have to be implemented by the user application (MCU_ prefix).

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.4.2 Macro Definition Documentation

6.4.2.1 FS65_BO_GET_REG_VALUE

```
#define FS65_BO_GET_REG_VALUE(
              value,
              mask,
              shift ) (((value) & (mask)) >> (shift))
```

Macro for getting value from register.

Parameters

value	Value read from register.
mask	Bit selection.
shift	Bit shift.

Returns

Masked and r-shifted value.

6.4.2.2 FS65_IS_REG_FAILSAFE

```
#define FS65_IS_REG_FAILSAFE(
              address ) ((address) & 0x20)
```

Returns true if register is fail-safe.

Parameters

address	Register address.

Returns

true if register is fail-safe.

6.5 Sources/FS65_driver/sbc_fs65_communication.c File Reference

Implementation of communication logic for NXP SBC FS65/FS45.

```
#include "sbc_fs65_communication.h"
#include "sbc_fs65_map.h"
#include "sbc_fs65_assert.h"
```

Functions

- fs65_status_t FS65_ReadRegister (uint8_t address, fs65_rx_data_t *rxData)

  Performs a read from a single FS65 register.
- fs65_status_t FS65_WriteRegister (uint8_t address, uint8_t writeData, fs65_rx_data_t *rxData)

  Sends write command to the FS65.
- fs65_status_t FS65_WriteRegisters (fs65_reg_config_value_t *registers, uint8_t numOfItems)

6.5.1 Detailed Description

Implementation of communication logic for NXP SBC FS65/FS45.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.6 Sources/FS65_driver/sbc_fs65_communication.h File Reference

This file contains functions for SPI communication.

```
#include "sbc_fs65_common.h"
```

Macros

- #define FS65_COMM_FRAME_SIZE_BYTES 0x02U

Functions

- fs65_status_t FS65_ReadRegister (uint8_t address, fs65_rx_data_t *rxData)
  Performs a read from a single FS65 register.
- fs65_status_t FS65_WriteRegister (uint8_t address, uint8_t writeData, fs65_rx_data_t *rxData)
  Sends write command to the FS65.
- fs65_status_t FS65_WriteRegisters (fs65_reg_config_value_t *registers, uint8_t numOfItems)

6.6.1 Detailed Description

This file contains functions for SPI communication.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.6.2 Macro Definition Documentation

6.6.2.1 FS65_COMM_FRAME_SIZE_BYTES

```
#define FS65_COMM_FRAME_SIZE_BYTES 0x02U
```

Length of the communication frame (bytes)

6.7 Sources/FS65_driver/sbc_fs65_map.h File Reference

Register map of the FS65/FS45 SBC series.

Macros

- #define FS65_R_M_BAT_FAIL_MASK 0x01U
- #define FS65_RW_M_VAUX_TRK_EN_MASK 0x02U
- #define FS65_RW_M_TAUX_LIM_OFF_MASK 0x04U
- #define FS65_RW_M_VCAN_OV_MON_MASK 0x10U
- #define FS65_RW_M_IPFF_DIS_MASK 0x20U
- #define FS65_RW_M_TCCA_LIM_OFF_MASK 0x40U
- #define FS65_RW_M_ICCA_LIM_MASK 0x80U
- #define FS65_R_M_BAT_FAIL_SHIFT 0x00U
- #define FS65_RW_M_VAUX_TRK_EN_SHIFT 0x01U
- #define FS65_RW_M_TAUX_LIM_OFF_SHIFT 0x02U
- #define FS65_RW_M_VCAN_OV_MON_SHIFT 0x04U
- #define FS65_RW_M_IPFF_DIS_SHIFT 0x05U
- #define FS65_RW_M_TCCA_LIM_OFF_SHIFT 0x06U
- #define FS65_RW_M_ICCA_LIM_SHIFT 0x07U
- #define FS65_R_M_BAT_FAIL_NO_POR (0x00U << FS65_R_M_BAT_FAIL_SHIFT)
- #define FS65_R_M_BAT_FAIL_POR (0x01U << FS65_R_M_BAT_FAIL_SHIFT)
- #define FS65_RW_M_VAUX_TRK_EN_NO_TRACKING (0x00U << FS65_RW_M_VAUX_TRK_EN_SH↩ IFT)
- #define FS65_RW_M_VAUX_TRK_EN_TRACKING (0x01U << FS65_RW_M_VAUX_TRK_EN_SHIFT)
- #define FS65_RW_M_TAUX_LIM_OFF_10_MS (0x00U << FS65_RW_M_TAUX_LIM_OFF_SHIFT)
- #define FS65_RW_M_TAUX_LIM_OFF_50_MS (0x01U << FS65_RW_M_TAUX_LIM_OFF_SHIFT)
- #define FS65_RW_M_VCAN_OV_MON_OFF (0x00U << FS65_RW_M_VCAN_OV_MON_SHIFT)
- #define FS65_RW_M_VCAN_OV_MON_ON (0x01U << FS65_RW_M_VCAN_OV_MON_SHIFT)
- #define FS65_RW_M_IPFF_DIS_ENABLED (0x00U << FS65_RW_M_IPFF_DIS_SHIFT)
- #define FS65_RW_M_IPFF_DIS_DISABLED (0x01U << FS65_RW_M_IPFF_DIS_SHIFT)
- #define FS65_RW_M_TCCA_LIM_OFF_10_MS (0x00U << FS65_RW_M_TCCA_LIM_OFF_SHIFT)
- #define FS65_RW_M_TCCA_LIM_OFF_50_MS (0x01U << FS65_RW_M_TCCA_LIM_OFF_SHIFT)
- #define FS65_RW_M_ICCA_LIM_ICCA_LIM_OUT (0x00U << FS65_RW_M_ICCA_LIM_SHIFT)
- #define FS65_RW_M_ICCA_LIM_ICCA_LIM_INT (0x01U << FS65_RW_M_ICCA_LIM_SHIFT)
- #define FS65_R_M_LDT_INT_MASK 0x01U
- #define FS65_RW_M_LDT_ENABLE_MASK 0x02U
- #define FS65_RW_M_MODE_MASK 0x04U
- #define FS65_R_M_LDT_RUNNING_MASK 0x08U
- #define FS65_RW_M_REG_SE_MASK 0x10U

- #define FS65_RW_M_F2_F0_MASK 0xE0U
- #define FS65_R_M_LDT_INT_SHIFT 0x00U
- #define FS65_RW_M_LDT_ENABLE_SHIFT 0x01U
- #define FS65_RW_M_MODE_SHIFT 0x02U
- #define FS65_R_M_LDT_RUNNING_SHIFT 0x03U
- #define FS65_RW_M_REG_SE_SHIFT 0x04U
- #define FS65_RW_M_F2_F0_SHIFT 0x05U
- #define FS65_R_M_LDT_INT_NOT_RUNNING (0x00U << FS65_R_M_LDT_INT_SHIFT)
- #define FS65_R_M_LDT_INT_RUNNING (0x01U << FS65_R_M_LDT_INT_SHIFT)
- #define FS65_RW_M_LDT_ENABLE_STOP (0x00U << FS65_RW_M_LDT_ENABLE_SHIFT)
- #define FS65_RW_M_LDT_ENABLE_START (0x01U << FS65_RW_M_LDT_ENABLE_SHIFT)
- #define FS65_RW_M_MODE_CALIBRATION (0x00U << FS65_RW_M_MODE_SHIFT)
- #define FS65_RW_M_MODE_NORMAL (0x01U << FS65_RW_M_MODE_SHIFT)
- #define FS65_R_M_LDT_RUNNING_NOT_RUNNING (0x00U << FS65_R_M_LDT_RUNNING_SHIFT)
- #define FS65_R_M_LDT_RUNNING_RUNNING (0x01U << FS65_R_M_LDT_RUNNING_SHIFT)
- #define FS65_RW_M_REG_SE_PROGRAMMED_REG (0x00U << FS65_RW_M_REG_SE_SHIFT)
- #define FS65_RW_M_REG_SE_RTC_REG (0x01U << FS65_RW_M_REG_SE_SHIFT)
- #define FS65_RW_M_F2_F0_FUNCTION1 (0x00U << FS65_RW_M_F2_F0_SHIFT)
- #define FS65_RW_M_F2_F0_FUNCTION2 (0x01U << FS65_RW_M_F2_F0_SHIFT)
- #define FS65_RW_M_F2_F0_FUNCTION3 (0x02U << FS65_RW_M_F2_F0_SHIFT)
- #define FS65_RW_M_F2_F0_FUNCTION4 (0x03U << FS65_RW_M_F2_F0_SHIFT)
- #define FS65_RW_M_F2_F0_FUNCTION5 (0x04U << FS65_RW_M_F2_F0_SHIFT)
- #define FS65_R_M_DBG_HW_MASK 0x01U
- #define FS65_R_M_DFS_HW1_MASK 0x02U
- #define FS65_R_M_VAUX_HW_MASK 0x08U
- #define FS65_R_M_VCCA_HW_MASK 0x10U
- #define FS65_R_M_VCCA_PNP_DET_MASK 0x20U
- #define FS65_R_M_LS_DETECT_MASK 0x80U
- #define FS65_R_M_DBG_HW_SHIFT 0x00U
- #define FS65_R_M_DFS_HW1_SHIFT 0x01U
- #define FS65_R_M_VAUX_HW_SHIFT 0x03U
- #define FS65_R_M_VCCA_HW_SHIFT 0x04U
- #define FS65_R_M_VCCA_PNP_DET_SHIFT 0x05U
- #define FS65_R_M_LS_DETECT_SHIFT 0x07U
- #define FS65_R_M_DBG_HW_NORMAL (0x00U << FS65_R_M_DBG_HW_SHIFT)
- #define FS65_R_M_DBG_HW_DEBUG (0x01U << FS65_R_M_DBG_HW_SHIFT)
- #define FS65_R_M_DFS_HW1_DISABLE (0x00U << FS65_R_M_DFS_HW1_SHIFT)
- #define FS65_R_M_DFS_HW1_ENABLE (0x01U << FS65_R_M_DFS_HW1_SHIFT)
- #define FS65_R_M_VAUX_HW_5_0V (0x00U << FS65_R_M_VAUX_HW_SHIFT)
- #define FS65_R_M_VAUX_HW_3_3V (0x01U << FS65_R_M_VAUX_HW_SHIFT)
- #define FS65_R_M_VCCA_HW_3_3V (0x00U << FS65_R_M_VCCA_HW_SHIFT)
- #define FS65_R_M_VCCA_HW_5_0V (0x01U << FS65_R_M_VCCA_HW_SHIFT)
- #define FS65_R_M_VCCA_PNP_DET_PNP_CONNECTED (0x00U << FS65_R_M_VCCA_PNP_DET_↩ SHIFT)
- #define FS65_R_M_VCCA_PNP_DET_INT_MOSFET (0x01U << FS65_R_M_VCCA_PNP_DET_SHIFT)
- #define FS65_R_M_LS_DETECT_BUCK_BOOST (0x00U << FS65_R_M_LS_DETECT_SHIFT)
- #define FS65_R_M_LS_DETECT_BUCK_ONLY (0x01U << FS65_R_M_LS_DETECT_SHIFT)
- #define FS65_R_M_PHY_WU_MASK 0x01U
- #define FS65_R_M_LDT_WU_MASK 0x02U
- #define FS65_R_M_AUTO_WU_MASK 0x04U
- #define FS65_R_M_IO_0_WU_MASK 0x08U
- #define FS65_R_M_IO_2_WU_MASK 0x10U
- #define FS65_R_M_IO_3_WU_MASK 0x20U
- #define FS65_R_M_IO_4_WU_MASK 0x40U
- #define FS65_R_M_IO_5_WU_MASK 0x80U

- #define FS65_R_M_PHY_WU_SHIFT 0x00U
- #define FS65_R_M_LDT_WU_SHIFT 0x01U
- #define FS65_R_M_AUTO_WU_SHIFT 0x02U
- #define FS65_R_M_IO_0_WU_SHIFT 0x03U
- #define FS65_R_M_IO_2_WU_SHIFT 0x04U
- #define FS65_R_M_IO_3_WU_SHIFT 0x05U
- #define FS65_R_M_IO_4_WU_SHIFT 0x06U
- #define FS65_R_M_IO_5_WU_SHIFT 0x07U
- #define FS65_R_M_PHY_WU_NO_EVENT (0x00U << FS65_R_M_PHY_WU_SHIFT)
- #define FS65_R_M_PHY_WU_EVENT (0x01U << FS65_R_M_PHY_WU_SHIFT)
- #define FS65_R_M_LDT_WU_NO_EVENT (0x00U << FS65_R_M_LDT_WU_SHIFT)
- #define FS65_R_M_LDT_WU_EVENT (0x01U << FS65_R_M_LDT_WU_SHIFT)
- #define FS65_R_M_AUTO_WU_NO_EVENT (0x00U << FS65_R_M_AUTO_WU_SHIFT)
- #define FS65_R_M_AUTO_WU_EVENT (0x01U << FS65_R_M_AUTO_WU_SHIFT)
- #define FS65_R_M_IO_0_WU_NO_EVENT (0x00U << FS65_R_M_IO_0_WU_SHIFT)
- #define FS65_R_M_IO_0_WU_EVENT (0x01U << FS65_R_M_IO_0_WU_SHIFT)
- #define FS65_R_M_IO_2_WU_NO_EVENT (0x00U << FS65_R_M_IO_2_WU_SHIFT)
- #define FS65_R_M_IO_2_WU_EVENT (0x01U << FS65_R_M_IO_2_WU_SHIFT)
- #define FS65_R_M_IO_3_WU_NO_EVENT (0x00U << FS65_R_M_IO_3_WU_SHIFT)
- #define FS65_R_M_IO_3_WU_EVENT (0x01U << FS65_R_M_IO_3_WU_SHIFT)
- #define FS65_R_M_IO_4_WU_NO_EVENT (0x00U << FS65_R_M_IO_4_WU_SHIFT)
- #define FS65_R_M_IO_4_WU_EVENT (0x01U << FS65_R_M_IO_4_WU_SHIFT)
- #define FS65_R_M_IO_5_WU_NO_EVENT (0x00U << FS65_R_M_IO_5_WU_SHIFT)
- #define FS65_R_M_IO_5_WU_EVENT (0x01U << FS65_R_M_IO_5_WU_SHIFT)
- #define FS65_R_M_DEV_REV_MASK 0x07U
- #define FS65_R_M_VKAM_MASK 0x08U
- #define FS65_R_M_PHY_MASK 0x30U
- #define FS65_R_M_VCORE_MASK 0xC0U
- #define FS65_R_M_DEV_REV_SHIFT 0x00U
- #define FS65_R_M_VKAM_SHIFT 0x03U
- #define FS65_R_M_PHY_SHIFT 0x04U
- #define FS65_R_M_VCORE_SHIFT 0x06U
- #define FS65_R_M_DEV_REV_REV_000 (0x00U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_001 (0x01U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_010 (0x02U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_011 (0x03U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_100 (0x04U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_101 (0x05U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_110 (0x06U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_DEV_REV_REV_111 (0x07U << FS65_R_M_DEV_REV_SHIFT)
- #define FS65_R_M_VKAM_OFF (0x00U << FS65_R_M_VKAM_SHIFT)
- #define FS65_R_M_VKAM_ON (0x01U << FS65_R_M_VKAM_SHIFT)
- #define FS65_R_M_PHY_NOCAN_NOLIN (0x00U << FS65_R_M_PHY_SHIFT)
- #define FS65_R_M_PHY_CAN (0x01U << FS65_R_M_PHY_SHIFT)
- #define FS65_R_M_PHY_LIN (0x02U << FS65_R_M_PHY_SHIFT)
- #define FS65_R_M_PHY_CAN_LIN (0x03U << FS65_R_M_PHY_SHIFT)
- #define FS65_R_M_VCORE_1_5A (0x00U << FS65_R_M_VCORE_SHIFT)
- #define FS65_R_M_VCORE_0_8A (0x01U << FS65_R_M_VCORE_SHIFT)
- #define FS65_R_M_VCORE_0_5A (0x02U << FS65_R_M_VCORE_SHIFT)
- #define FS65_R_M_VCORE_2_2A (0x03U << FS65_R_M_VCORE_SHIFT)
- #define FS65_R_M_IO_0_MASK 0x01U
- #define FS65_R_M_IO_2_MASK 0x08U
- #define FS65_R_M_IO_3_MASK 0x10U
- #define FS65_R_M_IO_4_MASK 0x40U
- #define FS65_R_M_IO_5_MASK 0x80U

- #define FS65_R_M_IO_0_SHIFT 0x00U
- #define FS65_R_M_IO_2_SHIFT 0x03U
- #define FS65_R_M_IO_3_SHIFT 0x04U
- #define FS65_R_M_IO_4_SHIFT 0x06U
- #define FS65_R_M_IO_5_SHIFT 0x07U
- #define FS65_R_M_IO_0_LOW (0x00U << FS65_R_M_IO_0_SHIFT)
- #define FS65_R_M_IO_0_HIGH (0x01U << FS65_R_M_IO_0_SHIFT)
- #define FS65_R_M_IO_2_LOW (0x00U << FS65_R_M_IO_2_SHIFT)
- #define FS65_R_M_IO_2_HIGH (0x01U << FS65_R_M_IO_2_SHIFT)
- #define FS65_R_M_IO_3_LOW (0x00U << FS65_R_M_IO_3_SHIFT)
- #define FS65_R_M_IO_3_HIGH (0x01U << FS65_R_M_IO_3_SHIFT)
- #define FS65_R_M_IO_4_LOW (0x00U << FS65_R_M_IO_4_SHIFT)
- #define FS65_R_M_IO_4_HIGH (0x01U << FS65_R_M_IO_4_SHIFT)
- #define FS65_R_M_IO_5_LOW (0x00U << FS65_R_M_IO_5_SHIFT)
- #define FS65_R_M_IO_5_HIGH (0x01U << FS65_R_M_IO_5_SHIFT)
- #define FS65_R_M_ILIM_PRE_MASK 0x02U
- #define FS65_R_M_VPRE_UV_MASK 0x04U
- #define FS65_R_M_VPRE_OV_MASK 0x08U
- #define FS65_R_M_TSD_PRE_MASK 0x10U
- #define FS65_R_M_TWARN_PRE_MASK 0x20U
- #define FS65_R_M_VPRE_STATE_MASK 0x40U
- #define FS65_R_M_BOB_MASK 0x80U
- #define FS65_R_M_ILIM_PRE_SHIFT 0x01U
- #define FS65_R_M_VPRE_UV_SHIFT 0x02U
- #define FS65_R_M_VPRE_OV_SHIFT 0x03U
- #define FS65_R_M_TSD_PRE_SHIFT 0x04U
- #define FS65_R_M_TWARN_PRE_SHIFT 0x05U
- #define FS65_R_M_VPRE_STATE_SHIFT 0x06U
- #define FS65_R_M_BOB_SHIFT 0x07U
- #define FS65_R_M_ILIM_PRE_NO_LIMITATION (0x00U << FS65_R_M_ILIM_PRE_SHIFT)
- #define FS65_R_M_ILIM_PRE_LIMITATION (0x01U << FS65_R_M_ILIM_PRE_SHIFT)
- #define FS65_R_M_VPRE_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VPRE_UV_SHIFT)
- #define FS65_R_M_VPRE_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VPRE_UV_SHIFT)
- #define FS65_R_M_VPRE_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VPRE_OV_SHIFT)
- #define FS65_R_M_VPRE_OV_OVERVOLTAGE (0x01U << FS65_R_M_VPRE_OV_SHIFT)
- #define FS65_R_M_TSD_PRE_NO_TSD (0x00U << FS65_R_M_TSD_PRE_SHIFT)
- #define FS65_R_M_TSD_PRE_TSD_OCCURRED (0x01U << FS65_R_M_TSD_PRE_SHIFT)
- #define FS65_R_M_TWARN_PRE_NO_WARNING (0x00U << FS65_R_M_TWARN_PRE_SHIFT)
- #define FS65_R_M_TWARN_PRE_WARNING (0x01U << FS65_R_M_TWARN_PRE_SHIFT)
- #define FS65_R_M_VPRE_STATE_OFF (0x00U << FS65_R_M_VPRE_STATE_SHIFT)
- #define FS65_R_M_VPRE_STATE_ON (0x01U << FS65_R_M_VPRE_STATE_SHIFT)
- #define FS65_R_M_BOB_BUCK (0x00U << FS65_R_M_BOB_SHIFT)
- #define FS65_R_M_BOB_BOOST (0x01U << FS65_R_M_BOB_SHIFT)
- #define FS65_R_M_VCORE_FB_UV_MASK 0x04U
- #define FS65_R_M_VCORE_FB_OV_MASK 0x08U
- #define FS65_R_M_TSD_CORE_MASK 0x10U
- #define FS65_R_M_TWARN_CORE_MASK 0x20U
- #define FS65_R_M_VCORE_STATE_MASK 0x40U
- #define FS65_R_M_VCORE_FB_UV_SHIFT 0x02U
- #define FS65_R_M_VCORE_FB_OV_SHIFT 0x03U
- #define FS65_R_M_TSD_CORE_SHIFT 0x04U
- #define FS65_R_M_TWARN_CORE_SHIFT 0x05U
- #define FS65_R_M_VCORE_STATE_SHIFT 0x06U
- #define FS65_R_M_VCORE_FB_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCORE_FB_UV_↩ SHIFT)

- #define FS65_R_M_VCORE_FB_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCORE_FB_UV_SHIFT)
- #define FS65_R_M_VCORE_FB_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCORE_FB_OV_S↩ HIFT)
- #define FS65_R_M_VCORE_FB_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCORE_FB_OV_SHIFT)
- #define FS65_R_M_TSD_CORE_NO_TSD (0x00U << FS65_R_M_TSD_CORE_SHIFT)
- #define FS65_R_M_TSD_CORE_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CORE_SHIFT)
- #define FS65_R_M_TWARN_CORE_NO_WARNING (0x00U << FS65_R_M_TWARN_CORE_SHIFT)
- #define FS65_R_M_TWARN_CORE_WARNING (0x01U << FS65_R_M_TWARN_CORE_SHIFT)
- #define FS65_R_M_VCORE_STATE_OFF (0x00U << FS65_R_M_VCORE_STATE_SHIFT)
- #define FS65_R_M_VCORE_STATE_ON (0x01U << FS65_R_M_VCORE_STATE_SHIFT)
- #define FS65_R_M_ILIM_CCA_OFF_MASK 0x01U
- #define FS65_R_M_ILIM_CCA_MASK 0x02U
- #define FS65_R_M_VCCA_UV_MASK 0x04U
- #define FS65_R_M_VCCA_OV_MASK 0x08U
- #define FS65_R_M_TSD_CCA_MASK 0x10U
- #define FS65_R_M_TWARN_CCA_MASK 0x20U
- #define FS65_R_M_ILIM_CCA_OFF_SHIFT 0x00U
- #define FS65_R_M_ILIM_CCA_SHIFT 0x01U
- #define FS65_R_M_VCCA_UV_SHIFT 0x02U
- #define FS65_R_M_VCCA_OV_SHIFT 0x03U
- #define FS65_R_M_TSD_CCA_SHIFT 0x04U
- #define FS65_R_M_TWARN_CCA_SHIFT 0x05U
- #define FS65_R_M_ILIM_CCA_OFF_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CCA_OFF_SHIFT)
- #define FS65_R_M_ILIM_CCA_OFF_LIMITATION (0x01U << FS65_R_M_ILIM_CCA_OFF_SHIFT)
- #define FS65_R_M_ILIM_CCA_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CCA_SHIFT)
- #define FS65_R_M_ILIM_CCA_LIMITATION (0x01U << FS65_R_M_ILIM_CCA_SHIFT)
- #define FS65_R_M_VCCA_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCCA_UV_SHIFT)
- #define FS65_R_M_VCCA_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCCA_UV_SHIFT)
- #define FS65_R_M_VCCA_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCCA_OV_SHIFT)
- #define FS65_R_M_VCCA_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCCA_OV_SHIFT)
- #define FS65_R_M_TSD_CCA_NO_TSD (0x00U << FS65_R_M_TSD_CCA_SHIFT)
- #define FS65_R_M_TSD_CCA_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CCA_SHIFT)
- #define FS65_R_M_TWARN_CCA_NO_WARNING (0x00U << FS65_R_M_TWARN_CCA_SHIFT)
- #define FS65_R_M_TWARN_CCA_WARNING (0x01U << FS65_R_M_TWARN_CCA_SHIFT)
- #define FS65_R_M_ILIM_AUX_OFF_MASK 0x01U
- #define FS65_R_M_ILIM_AUX_MASK 0x02U
- #define FS65_R_M_VAUX_UV_MASK 0x04U
- #define FS65_R_M_VAUX_OV_MASK 0x08U
- #define FS65_R_M_TSD_AUX_MASK 0x10U
- #define FS65_R_M_ILIM_AUX_OFF_SHIFT 0x00U
- #define FS65_R_M_ILIM_AUX_SHIFT 0x01U
- #define FS65_R_M_VAUX_UV_SHIFT 0x02U
- #define FS65_R_M_VAUX_OV_SHIFT 0x03U
- #define FS65_R_M_TSD_AUX_SHIFT 0x04U
- #define FS65_R_M_ILIM_AUX_OFF_NO_LIMITATION (0x00U << FS65_R_M_ILIM_AUX_OFF_SHIFT)
- #define FS65_R_M_ILIM_AUX_OFF_LIMITATION (0x01U << FS65_R_M_ILIM_AUX_OFF_SHIFT)
- #define FS65_R_M_ILIM_AUX_NO_LIMITATION (0x00U << FS65_R_M_ILIM_AUX_SHIFT)
- #define FS65_R_M_ILIM_AUX_LIMITATION (0x01U << FS65_R_M_ILIM_AUX_SHIFT)
- #define FS65_R_M_VAUX_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VAUX_UV_SHIFT)
- #define FS65_R_M_VAUX_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VAUX_UV_SHIFT)
- #define FS65_R_M_VAUX_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VAUX_OV_SHIFT)
- #define FS65_R_M_VAUX_OV_OVERVOLTAGE (0x01U << FS65_R_M_VAUX_OV_SHIFT)
- #define FS65_R_M_TSD_AUX_NO_TSD (0x00U << FS65_R_M_TSD_AUX_SHIFT)
- #define FS65_R_M_TSD_AUX_TSD_OCCURRED (0x01U << FS65_R_M_TSD_AUX_SHIFT)
- #define FS65_R_M_ILIM_CAN_MASK 0x02U

- #define FS65_R_M_VCAN_UV_MASK 0x04U
- #define FS65_R_M_VCAN_OV_MASK 0x08U
- #define FS65_R_M_TSD_CAN_MASK 0x10U
- #define FS65_R_M_IPFF_MASK 0x20U
- #define FS65_R_M_VSUP_UV_7_MASK 0x40U
- #define FS65_R_M_VSNS_UV_MASK 0x80U
- #define FS65_R_M_ILIM_CAN_SHIFT 0x01U
- #define FS65_R_M_VCAN_UV_SHIFT 0x02U
- #define FS65_R_M_VCAN_OV_SHIFT 0x03U
- #define FS65_R_M_TSD_CAN_SHIFT 0x04U
- #define FS65_R_M_IPFF_SHIFT 0x05U
- #define FS65_R_M_VSUP_UV_7_SHIFT 0x06U
- #define FS65_R_M_VSNS_UV_SHIFT 0x07U
- #define FS65_R_M_ILIM_CAN_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CAN_SHIFT)
- #define FS65_R_M_ILIM_CAN_LIMITATION (0x01U << FS65_R_M_ILIM_CAN_SHIFT)
- #define FS65_R_M_VCAN_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCAN_UV_SHIFT)
- #define FS65_R_M_VCAN_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCAN_UV_SHIFT)
- #define FS65_R_M_VCAN_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCAN_OV_SHIFT)
- #define FS65_R_M_VCAN_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCAN_OV_SHIFT)
- #define FS65_R_M_TSD_CAN_NO_TSD (0x00U << FS65_R_M_TSD_CAN_SHIFT)
- #define FS65_R_M_TSD_CAN_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CAN_SHIFT)
- #define FS65_R_M_IPFF_NORMAL (0x00U << FS65_R_M_IPFF_SHIFT)
- #define FS65_R_M_IPFF_IPFF (0x01U << FS65_R_M_IPFF_SHIFT)
- #define FS65_R_M_VSUP_UV_7_VSUP_G (0x00U << FS65_R_M_VSUP_UV_7_SHIFT)
- #define FS65_R_M_VSUP_UV_7_VSUP_L (0x01U << FS65_R_M_VSUP_UV_7_SHIFT)
- #define FS65_R_M_VSNS_UV_VBAT_G (0x00U << FS65_R_M_VSNS_UV_SHIFT)
- #define FS65_R_M_VSNS_UV_VBAT_L (0x01U << FS65_R_M_VSNS_UV_SHIFT)
- #define FS65_R_M_TXD_DOM_MASK 0x01U
- #define FS65_R_M_RXD_REC_MASK 0x02U
- #define FS65_R_M_CAN_DOM_MASK 0x08U
- #define FS65_R_M_CANL_GND_MASK 0x10U
- #define FS65_R_M_CANL_BATT_MASK 0x20U
- #define FS65_R_M_CANH_GND_MASK 0x40U
- #define FS65_R_M_CANH_BATT_MASK 0x80U
- #define FS65_R_M_TXD_DOM_SHIFT 0x00U
- #define FS65_R_M_RXD_REC_SHIFT 0x01U
- #define FS65_R_M_CAN_DOM_SHIFT 0x03U
- #define FS65_R_M_CANL_GND_SHIFT 0x04U
- #define FS65_R_M_CANL_BATT_SHIFT 0x05U
- #define FS65_R_M_CANH_GND_SHIFT 0x06U
- #define FS65_R_M_CANH_BATT_SHIFT 0x07U
- #define FS65_R_M_TXD_DOM_NO_FAILURE (0x00U << FS65_R_M_TXD_DOM_SHIFT)
- #define FS65_R_M_TXD_DOM_FAILURE (0x01U << FS65_R_M_TXD_DOM_SHIFT)
- #define FS65_R_M_RXD_REC_NO_FAILURE (0x00U << FS65_R_M_RXD_REC_SHIFT)
- #define FS65_R_M_RXD_REC_FAILURE (0x01U << FS65_R_M_RXD_REC_SHIFT)
- #define FS65_R_M_CAN_DOM_NO_FAILURE (0x00U << FS65_R_M_CAN_DOM_SHIFT)
- #define FS65_R_M_CAN_DOM_FAILURE (0x01U << FS65_R_M_CAN_DOM_SHIFT)
- #define FS65_R_M_CANL_GND_NO_FAILURE (0x00U << FS65_R_M_CANL_GND_SHIFT)
- #define FS65_R_M_CANL_GND_FAILURE (0x01U << FS65_R_M_CANL_GND_SHIFT)
- #define FS65_R_M_CANL_BATT_NO_FAILURE (0x00U << FS65_R_M_CANL_BATT_SHIFT)
- #define FS65_R_M_CANL_BATT_FAILURE (0x01U << FS65_R_M_CANL_BATT_SHIFT)
- #define FS65_R_M_CANH_GND_NO_FAILURE (0x00U << FS65_R_M_CANH_GND_SHIFT)
- #define FS65_R_M_CANH_GND_FAILURE (0x01U << FS65_R_M_CANH_GND_SHIFT)
- #define FS65_R_M_CANH_BATT_NO_FAILURE (0x00U << FS65_R_M_CANH_BATT_SHIFT)
- #define FS65_R_M_CANH_BATT_FAILURE (0x01U << FS65_R_M_CANH_BATT_SHIFT)

- #define FS65_R_M_CAN_OC_MASK 0x01U
- #define FS65_R_M_CAN_OT_MASK 0x02U
- #define FS65_R_M_LIN_OT_MASK 0x08U
- #define FS65_R_M_RXDL_REC_MASK 0x10U
- #define FS65_R_M_TDXL_DOM_MASK 0x40U
- #define FS65_R_M_LIN_DOM_MASK 0x80U
- #define FS65_R_M_CAN_OC_SHIFT 0x00U
- #define FS65_R_M_CAN_OT_SHIFT 0x01U
- #define FS65_R_M_LIN_OT_SHIFT 0x03U
- #define FS65_R_M_RXDL_REC_SHIFT 0x04U
- #define FS65_R_M_TDXL_DOM_SHIFT 0x06U
- #define FS65_R_M_LIN_DOM_SHIFT 0x07U
- #define FS65_R_M_CAN_OC_NO_FAILURE (0x00U << FS65_R_M_CAN_OC_SHIFT)
- #define FS65_R_M_CAN_OC_FAILURE (0x01U << FS65_R_M_CAN_OC_SHIFT)
- #define FS65_R_M_CAN_OT_NO_FAILURE (0x00U << FS65_R_M_CAN_OT_SHIFT)
- #define FS65_R_M_CAN_OT_FAILURE (0x01U << FS65_R_M_CAN_OT_SHIFT)
- #define FS65_R_M_LIN_OT_NO_FAILURE (0x00U << FS65_R_M_LIN_OT_SHIFT)
- #define FS65_R_M_LIN_OT_FAILURE (0x01U << FS65_R_M_LIN_OT_SHIFT)
- #define FS65_R_M_RXDL_REC_NO_FAILURE (0x00U << FS65_R_M_RXDL_REC_SHIFT)
- #define FS65_R_M_RXDL_REC_FAILURE (0x01U << FS65_R_M_RXDL_REC_SHIFT)
- #define FS65_R_M_TDXL_DOM_NO_FAILURE (0x00U << FS65_R_M_TDXL_DOM_SHIFT)
- #define FS65_R_M_TDXL_DOM_FAILURE (0x01U << FS65_R_M_TDXL_DOM_SHIFT)
- #define FS65_R_M_LIN_DOM_NO_FAILURE (0x00U << FS65_R_M_LIN_DOM_SHIFT)
- #define FS65_R_M_LIN_DOM_FAILURE (0x01U << FS65_R_M_LIN_DOM_SHIFT)
- #define FS65_R_M_SPI_PARITY_MASK 0x02U
- #define FS65_R_M_SPI_REQ_MASK 0x08U
- #define FS65_R_M_SPI_CLK_MASK 0x20U
- #define FS65_R_M_SPI_ERR_MASK 0x80U
- #define FS65_R_M_SPI_PARITY_SHIFT 0x01U
- #define FS65_R_M_SPI_REQ_SHIFT 0x03U
- #define FS65_R_M_SPI_CLK_SHIFT 0x05U
- #define FS65_R_M_SPI_ERR_SHIFT 0x07U
- #define FS65_R_M_SPI_PARITY_OK (0x00U << FS65_R_M_SPI_PARITY_SHIFT)
- #define FS65_R_M_SPI_PARITY_ERROR (0x01U << FS65_R_M_SPI_PARITY_SHIFT)
- #define FS65_R_M_SPI_REQ_NO_ERROR (0x00U << FS65_R_M_SPI_REQ_SHIFT)
- #define FS65_R_M_SPI_REQ_SPI_VIOLATION (0x01U << FS65_R_M_SPI_REQ_SHIFT)
- #define FS65_R_M_SPI_CLK_16_CLK_CYCLES (0x00U << FS65_R_M_SPI_CLK_SHIFT)
- #define FS65_R_M_SPI_CLK_WRONG_NUMBER (0x01U << FS65_R_M_SPI_CLK_SHIFT)
- #define FS65_R_M_SPI_ERR_NO_ERROR (0x00U << FS65_R_M_SPI_ERR_SHIFT)
- #define FS65_R_M_SPI_ERR_ERROR (0x01U << FS65_R_M_SPI_ERR_SHIFT)
- #define FS65_R_M_LPOFF_MASK 0x01U
- #define FS65_R_M_DFS_MASK 0x02U
- #define FS65_R_M_NORMAL_MASK 0x04U
- #define FS65_R_M_INIT_MASK 0x08U
- #define FS65_W_M_INT_REQ_MASK 0x10U
- #define FS65_W_M_GO_LPOFF_MASK 0x20U
- #define FS65_W_M_LPOFF_AUTO_WU_MASK 0x40U
- #define FS65_RW_M_VKAM_EN_MASK 0x80U
- #define FS65_R_M_LPOFF_SHIFT 0x00U
- #define FS65_R_M_DFS_SHIFT 0x01U
- #define FS65_R_M_NORMAL_SHIFT 0x02U
- #define FS65_R_M_INIT_SHIFT 0x03U
- #define FS65_W_M_INT_REQ_SHIFT 0x04U
- #define FS65_W_M_GO_LPOFF_SHIFT 0x05U
- #define FS65_W_M_LPOFF_AUTO_WU_SHIFT 0x06U

- #define FS65_RW_M_VKAM_EN_SHIFT 0x07U
- #define FS65_R_M_LPOFF_NOT_LPOFF (0x00U << FS65_R_M_LPOFF_SHIFT)
- #define FS65_R_M_LPOFF_RESUME_LPOFF (0x01U << FS65_R_M_LPOFF_SHIFT)
- #define FS65_R_M_DFS_NOT_DFS (0x00U << FS65_R_M_DFS_SHIFT)
- #define FS65_R_M_DFS_RESUME_DFS (0x01U << FS65_R_M_DFS_SHIFT)
- #define FS65_R_M_NORMAL_NOT_NORMAL (0x00U << FS65_R_M_NORMAL_SHIFT)
- #define FS65_R_M_NORMAL_NORMAL (0x01U << FS65_R_M_NORMAL_SHIFT)
- #define FS65_R_M_INIT_NOT_INIT (0x00U << FS65_R_M_INIT_SHIFT)
- #define FS65_R_M_INIT_INIT (0x01U << FS65_R_M_INIT_SHIFT)
- #define FS65_W_M_INT_REQ_NO (0x00U << FS65_W_M_INT_REQ_SHIFT)
- #define FS65_W_M_INT_REQ_INT_REQ (0x01U << FS65_W_M_INT_REQ_SHIFT)
- #define FS65_W_M_GO_LPOFF_NO_ACTION (0x00U << FS65_W_M_GO_LPOFF_SHIFT)
- #define FS65_W_M_GO_LPOFF_LPOFF (0x01U << FS65_W_M_GO_LPOFF_SHIFT)
- #define FS65_W_M_LPOFF_AUTO_WU_NO_ACTION (0x00U << FS65_W_M_LPOFF_AUTO_WU_SH↩ IFT)
- #define FS65_W_M_LPOFF_AUTO_WU_LPOFF (0x01U << FS65_W_M_LPOFF_AUTO_WU_SHIFT)
- #define FS65_RW_M_VKAM_EN_DISABLED (0x00U << FS65_RW_M_VKAM_EN_SHIFT)
- #define FS65_RW_M_VKAM_EN_ENABLED (0x01U << FS65_RW_M_VKAM_EN_SHIFT)
- #define FS65_R_M_VCAN_EN_MASK 0x01U
- #define FS65_R_M_VAUX_EN_MASK 0x02U
- #define FS65_R_M_VCCA_EN_MASK 0x04U
- #define FS65_R_M_VCORE_EN_MASK 0x08U
- #define FS65_W_M_VCAN_EN_MASK 0x10U
- #define FS65_W_M_VAUX_EN_MASK 0x20U
- #define FS65_W_M_VCCA_EN_MASK 0x40U
- #define FS65_W_M_VCORE_EN_MASK 0x80U
- #define FS65_R_M_VCAN_EN_SHIFT 0x00U
- #define FS65_R_M_VAUX_EN_SHIFT 0x01U
- #define FS65_R_M_VCCA_EN_SHIFT 0x02U
- #define FS65_R_M_VCORE_EN_SHIFT 0x03U
- #define FS65_W_M_VCAN_EN_SHIFT 0x04U
- #define FS65_W_M_VAUX_EN_SHIFT 0x05U
- #define FS65_W_M_VCCA_EN_SHIFT 0x06U
- #define FS65_W_M_VCORE_EN_SHIFT 0x07U
- #define FS65_R_M_VCAN_EN_DISABLED (0x00U << FS65_R_M_VCAN_EN_SHIFT)
- #define FS65_R_M_VCAN_EN_ENABLED (0x01U << FS65_R_M_VCAN_EN_SHIFT)
- #define FS65_R_M_VAUX_EN_DISABLED (0x00U << FS65_R_M_VAUX_EN_SHIFT)
- #define FS65_R_M_VAUX_EN_ENABLED (0x01U << FS65_R_M_VAUX_EN_SHIFT)
- #define FS65_R_M_VCCA_EN_DISABLED (0x00U << FS65_R_M_VCCA_EN_SHIFT)
- #define FS65_R_M_VCCA_EN_ENABLED (0x01U << FS65_R_M_VCCA_EN_SHIFT)
- #define FS65_R_M_VCORE_EN_DISABLED (0x00U << FS65_R_M_VCORE_EN_SHIFT)
- #define FS65_R_M_VCORE_EN_ENABLED (0x01U << FS65_R_M_VCORE_EN_SHIFT)
- #define FS65_W_M_VCAN_EN_DISABLED (0x00U << FS65_W_M_VCAN_EN_SHIFT)
- #define FS65_W_M_VCAN_EN_ENABLED (0x01U << FS65_W_M_VCAN_EN_SHIFT)
- #define FS65_W_M_VAUX_EN_DISABLED (0x00U << FS65_W_M_VAUX_EN_SHIFT)
- #define FS65_W_M_VAUX_EN_ENABLED (0x01U << FS65_W_M_VAUX_EN_SHIFT)
- #define FS65_W_M_VCCA_EN_DISABLED (0x00U << FS65_W_M_VCCA_EN_SHIFT)
- #define FS65_W_M_VCCA_EN_ENABLED (0x01U << FS65_W_M_VCCA_EN_SHIFT)
- #define FS65_W_M_VCORE_EN_DISABLED (0x00U << FS65_W_M_VCORE_EN_SHIFT)
- #define FS65_W_M_VCORE_EN_ENABLED (0x01U << FS65_W_M_VCORE_EN_SHIFT)
- #define FS65_RW_M_AMUX_MASK 0x07U
- #define FS65_RW_M_IO_OUT_4_MASK 0x40U
- #define FS65_RW_M_IO_OUT_4_EN_MASK 0x80U
- #define FS65_RW_M_AMUX_SHIFT 0x00U
- #define FS65_RW_M_IO_OUT_4_SHIFT 0x06U

- #define FS65_RW_M_IO_OUT_4_EN_SHIFT 0x07U
- #define FS65_RW_M_AMUX_VREF (0x00U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_VSNS_W (0x01U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_IO_0_W (0x02U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_IO_5_W (0x03U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_VSNS_T (0x04U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_IO_0_T (0x05U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_IO_5_T (0x06U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_AMUX_TEMP_SENSOR (0x07U << FS65_RW_M_AMUX_SHIFT)
- #define FS65_RW_M_IO_OUT_4_LOW (0x00U << FS65_RW_M_IO_OUT_4_SHIFT)
- #define FS65_RW_M_IO_OUT_4_HIGH (0x01U << FS65_RW_M_IO_OUT_4_SHIFT)
- #define FS65_RW_M_IO_OUT_4_EN_Z (0x00U << FS65_RW_M_IO_OUT_4_EN_SHIFT)
- #define FS65_RW_M_IO_OUT_4_EN_ENABLED (0x01U << FS65_RW_M_IO_OUT_4_EN_SHIFT)
- #define FS65_R_M_LIN_WU_MASK 0x01U
- #define FS65_R_M_CAN_WU_MASK 0x02U
- #define FS65_RW_M_LIN_AUTO_DIS_MASK 0x04U
- #define FS65_RW_M_LIN_MODE_MASK 0x18U
- #define FS65_RW_M_CAN_AUTO_DIS_MASK 0x20U
- #define FS65_RW_M_CAN_MODE_MASK 0xC0U
- #define FS65_R_M_LIN_WU_SHIFT 0x00U
- #define FS65_R_M_CAN_WU_SHIFT 0x01U
- #define FS65_RW_M_LIN_AUTO_DIS_SHIFT 0x02U
- #define FS65_RW_M_LIN_MODE_SHIFT 0x03U
- #define FS65_RW_M_CAN_AUTO_DIS_SHIFT 0x05U
- #define FS65_RW_M_CAN_MODE_SHIFT 0x06U
- #define FS65_R_M_LIN_WU_NO_WU (0x00U << FS65_R_M_LIN_WU_SHIFT)
- #define FS65_R_M_LIN_WU_WU (0x01U << FS65_R_M_LIN_WU_SHIFT)
- #define FS65_R_M_CAN_WU_NO_WU (0x00U << FS65_R_M_CAN_WU_SHIFT)
- #define FS65_R_M_CAN_WU_WU (0x01U << FS65_R_M_CAN_WU_SHIFT)
- #define FS65_RW_M_LIN_AUTO_DIS_NO (0x00U << FS65_RW_M_LIN_AUTO_DIS_SHIFT)
- #define FS65_RW_M_LIN_AUTO_DIS_RESET (0x01U << FS65_RW_M_LIN_AUTO_DIS_SHIFT)
- #define FS65_RW_M_LIN_MODE_SLN_WU (0x00U << FS65_RW_M_LIN_MODE_SHIFT)
- #define FS65_RW_M_LIN_MODE_LISTEN_ONLY (0x01U << FS65_RW_M_LIN_MODE_SHIFT)
- #define FS65_RW_M_LIN_MODE_SL_WU (0x02U << FS65_RW_M_LIN_MODE_SHIFT)
- #define FS65_RW_M_LIN_MODE_NORMAL (0x03U << FS65_RW_M_LIN_MODE_SHIFT)
- #define FS65_RW_M_CAN_AUTO_DIS_NO (0x00U << FS65_RW_M_CAN_AUTO_DIS_SHIFT)
- #define FS65_RW_M_CAN_AUTO_DIS_RESET (0x01U << FS65_RW_M_CAN_AUTO_DIS_SHIFT)
- #define FS65_RW_M_CAN_MODE_SLN_WU (0x00U << FS65_RW_M_CAN_MODE_SHIFT)
- #define FS65_RW_M_CAN_MODE_LISTEN_ONLY (0x01U << FS65_RW_M_CAN_MODE_SHIFT)
- #define FS65_RW_M_CAN_MODE_SL_WU (0x02U << FS65_RW_M_CAN_MODE_SHIFT)
- #define FS65_RW_M_CAN_MODE_NORMAL (0x03U << FS65_RW_M_CAN_MODE_SHIFT)
- #define FS65_R_M_ERR_INT_SW_MASK 0x01U
- #define FS65_R_M_ERR_INT_HW_MASK 0x02U
- #define FS65_R_M_WD_BAD_TIMING_MASK 0x04U
- #define FS65_R_M_IO_FS_G_MASK 0x08U
- #define FS65_R_M_FSO_G_MASK 0x10U
- #define FS65_R_M_WD_BAD_DATA_MASK 0x20U
- #define FS65_R_M_FSXB_MASK 0x40U
- #define FS65_R_M_RSTB_MASK 0x80U
- #define FS65_W_M_WD_ANSWER_MASK 0xFFU
- #define FS65_R_M_ERR_INT_SW_SHIFT 0x00U
- #define FS65_W_M_WD_ANSWER_SHIFT 0x00U
- #define FS65_R_M_ERR_INT_HW_SHIFT 0x01U
- #define FS65_R_M_WD_BAD_TIMING_SHIFT 0x02U
- #define FS65_R_M_IO_FS_G_SHIFT 0x03U

- #define FS65_R_M_FSO_G_SHIFT 0x04U
- #define FS65_R_M_WD_BAD_DATA_SHIFT 0x05U
- #define FS65_R_M_FSXB_SHIFT 0x06U
- #define FS65_R_M_RSTB_SHIFT 0x07U
- #define FS65_R_M_ERR_INT_SW_NO_ERROR (0x00U << FS65_R_M_ERR_INT_SW_SHIFT)
- #define FS65_R_M_ERR_INT_SW_ERROR (0x01U << FS65_R_M_ERR_INT_SW_SHIFT)
- #define FS65_R_M_ERR_INT_HW_NO_ERROR (0x00U << FS65_R_M_ERR_INT_HW_SHIFT)
- #define FS65_R_M_ERR_INT_HW_ERROR (0x01U << FS65_R_M_ERR_INT_HW_SHIFT)
- #define FS65_R_M_WD_BAD_TIMING_TIMING_OK (0x00U << FS65_R_M_WD_BAD_TIMING_SHIFT)
- #define FS65_R_M_WD_BAD_TIMING_WRONG_TIMING (0x01U << FS65_R_M_WD_BAD_TIMING_↩ SHIFT)
- #define FS65_R_M_IO_FS_G_NO_ERROR (0x00U << FS65_R_M_IO_FS_G_SHIFT)
- #define FS65_R_M_IO_FS_G_ERROR (0x01U << FS65_R_M_IO_FS_G_SHIFT)
- #define FS65_R_M_FSO_G_NO_FAILURE (0x00U << FS65_R_M_FSO_G_SHIFT)
- #define FS65_R_M_FSO_G_FAILURE (0x01U << FS65_R_M_FSO_G_SHIFT)
- #define FS65_R_M_WD_BAD_DATA_DATA_OK (0x00U << FS65_R_M_WD_BAD_DATA_SHIFT)
- #define FS65_R_M_WD_BAD_DATA_WRONG_DATA (0x01U << FS65_R_M_WD_BAD_DATA_SHIFT)
- #define FS65_R_M_FSXB_NO_FS (0x00U << FS65_R_M_FSXB_SHIFT)
- #define FS65_R_M_FSXB_FSE_OCCURRED (0x01U << FS65_R_M_FSXB_SHIFT)
- #define FS65_R_M_RSTB_NO_RESET (0x00U << FS65_R_M_RSTB_SHIFT)
- #define FS65_R_M_RSTB_RESET_OCCURRED (0x01U << FS65_R_M_RSTB_SHIFT)
- #define FS65_R_M_IO_45_FAIL_MASK 0x01U
- #define FS65_R_M_IO_23_FAIL_MASK 0x02U
- #define FS65_R_M_FS1B_DIAG_MASK 0x0CU
- #define FS65_R_M_FS0B_DIAG_MASK 0x30U
- #define FS65_R_M_RSTB_DIAG_MASK 0x40U
- #define FS65_R_M_RSTB_EXT_MASK 0x80U
- #define FS65_R_M_IO_45_FAIL_SHIFT 0x00U
- #define FS65_R_M_IO_23_FAIL_SHIFT 0x01U
- #define FS65_R_M_FS1B_DIAG_SHIFT 0x02U
- #define FS65_R_M_FS0B_DIAG_SHIFT 0x04U
- #define FS65_R_M_RSTB_DIAG_SHIFT 0x06U
- #define FS65_R_M_RSTB_EXT_SHIFT 0x07U
- #define FS65_R_M_IO_45_FAIL_NO_ERROR (0x00U << FS65_R_M_IO_45_FAIL_SHIFT)
- #define FS65_R_M_IO_45_FAIL_ERROR (0x01U << FS65_R_M_IO_45_FAIL_SHIFT)
- #define FS65_R_M_IO_23_FAIL_NO_ERROR (0x00U << FS65_R_M_IO_23_FAIL_SHIFT)
- #define FS65_R_M_IO_23_FAIL_ERROR (0x01U << FS65_R_M_IO_23_FAIL_SHIFT)
- #define FS65_R_M_FS1B_DIAG_NO_FAILURE (0x01U << FS65_R_M_FS1B_DIAG_SHIFT)
- #define FS65_R_M_FS1B_DIAG_SC_LOW (0x02U << FS65_R_M_FS1B_DIAG_SHIFT)
- #define FS65_R_M_FS1B_DIAG_SC_HIGH (0x03U << FS65_R_M_FS1B_DIAG_SHIFT)
- #define FS65_R_M_FS0B_DIAG_NO_FAILURE (0x01U << FS65_R_M_FS0B_DIAG_SHIFT)
- #define FS65_R_M_FS0B_DIAG_SC_LOW (0x02U << FS65_R_M_FS0B_DIAG_SHIFT)
- #define FS65_R_M_FS0B_DIAG_SC_HIGH (0x03U << FS65_R_M_FS0B_DIAG_SHIFT)
- #define FS65_R_M_RSTB_DIAG_NO_FAILURE (0x00U << FS65_R_M_RSTB_DIAG_SHIFT)
- #define FS65_R_M_RSTB_DIAG_SC_HIGH (0x01U << FS65_R_M_RSTB_DIAG_SHIFT)
- #define FS65_R_M_RSTB_EXT_NO (0x00U << FS65_R_M_RSTB_EXT_SHIFT)
- #define FS65_R_M_RSTB_EXT_EXTERNAL (0x01U << FS65_R_M_RSTB_EXT_SHIFT)
- #define FS65_R_M_WD_RFR_MASK 0x0EU
- #define FS65_R_M_WD_ERR_MASK 0xE0U
- #define FS65_R_M_WD_RFR_SHIFT 0x01U
- #define FS65_R_M_WD_ERR_SHIFT 0x05U
- #define FS65_R_M_FCRBM_UV_MASK 0x01U
- #define FS65_R_M_FCRBM_OV_MASK 0x02U
- #define FS65_R_M_V2P5_M_D_OV_MASK 0x04U
- #define FS65_R_M_V2P5_M_A_OV_MASK 0x08U

- #define FS65_R_M_FLT_ERR_MASK 0xE0U
- #define FS65_R_M_FCRBM_UV_SHIFT 0x00U
- #define FS65_R_M_FCRBM_OV_SHIFT 0x01U
- #define FS65_R_M_V2P5_M_D_OV_SHIFT 0x02U
- #define FS65_R_M_V2P5_M_A_OV_SHIFT 0x03U
- #define FS65_R_M_FLT_ERR_SHIFT 0x05U
- #define FS65_R_M_FCRBM_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_FCRBM_UV_SHIFT)
- #define FS65_R_M_FCRBM_UV_UNDERVOLTAGE (0x01U << FS65_R_M_FCRBM_UV_SHIFT)
- #define FS65_R_M_FCRBM_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_FCRBM_OV_SHIFT)
- #define FS65_R_M_FCRBM_OV_OVERVOLTAGE (0x01U << FS65_R_M_FCRBM_OV_SHIFT)
- #define FS65_R_M_V2P5_M_D_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_V2P5_M_D_OV_SH↩ IFT)
- #define FS65_R_M_V2P5_M_D_OV_OVERVOLTAGE (0x01U << FS65_R_M_V2P5_M_D_OV_SHIFT)
- #define FS65_R_M_V2P5_M_A_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_V2P5_M_A_OV_SH↩ IFT)
- #define FS65_R_M_V2P5_M_A_OV_OVERVOLTAGE (0x01U << FS65_R_M_V2P5_M_A_OV_SHIFT)
- #define FS65_R_M_FS1_MASK 0x01U
- #define FS65_R_M_DFS_HW2_MASK 0x02U
- #define FS65_R_M_FS1_SHIFT 0x00U
- #define FS65_R_M_DFS_HW2_SHIFT 0x01U
- #define FS65_R_M_FS1_DISABLED (0x00U << FS65_R_M_FS1_SHIFT)
- #define FS65_R_M_FS1_ENABLE (0x01U << FS65_R_M_FS1_SHIFT)
- #define FS65_R_M_DFS_HW2_DISABLE (0x00U << FS65_R_M_DFS_HW2_SHIFT)
- #define FS65_R_M_DFS_HW2_ENABLE (0x01U << FS65_R_M_DFS_HW2_SHIFT)
- #define FS65_R_FS_ABIST1_OK_MASK 0x01U
- #define FS65_R_FS_ABIST2_VAUX_OK_MASK 0x02U
- #define FS65_R_FS_ABIST2_FS1B_OK_MASK 0x04U
- #define FS65_R_FS_LBIST_OK_MASK 0x08U
- #define FS65_W_FS_ABIST2_VAUX_MASK 0x20U
- #define FS65_W_FS_ABIST2_FS1B_MASK 0x40U
- #define FS65_R_FS_ABIST1_OK_SHIFT 0x00U
- #define FS65_R_FS_ABIST2_VAUX_OK_SHIFT 0x01U
- #define FS65_R_FS_ABIST2_FS1B_OK_SHIFT 0x02U
- #define FS65_R_FS_LBIST_OK_SHIFT 0x03U
- #define FS65_W_FS_ABIST2_VAUX_SHIFT 0x05U
- #define FS65_W_FS_ABIST2_FS1B_SHIFT 0x06U
- #define FS65_R_FS_ABIST1_OK_FAIL (0x00U << FS65_R_FS_ABIST1_OK_SHIFT)
- #define FS65_R_FS_ABIST1_OK_PASS (0x01U << FS65_R_FS_ABIST1_OK_SHIFT)
- #define FS65_R_FS_ABIST2_VAUX_OK_FAIL (0x00U << FS65_R_FS_ABIST2_VAUX_OK_SHIFT)
- #define FS65_R_FS_ABIST2_VAUX_OK_PASS (0x01U << FS65_R_FS_ABIST2_VAUX_OK_SHIFT)
- #define FS65_R_FS_ABIST2_FS1B_OK_FAIL (0x00U << FS65_R_FS_ABIST2_FS1B_OK_SHIFT)
- #define FS65_R_FS_ABIST2_FS1B_OK_PASS (0x01U << FS65_R_FS_ABIST2_FS1B_OK_SHIFT)
- #define FS65_R_FS_LBIST_OK_FAIL (0x00U << FS65_R_FS_LBIST_OK_SHIFT)
- #define FS65_R_FS_LBIST_OK_PASS (0x01U << FS65_R_FS_LBIST_OK_SHIFT)
- #define FS65_W_FS_ABIST2_VAUX_NO_ACTION (0x00U << FS65_W_FS_ABIST2_VAUX_SHIFT)
- #define FS65_W_FS_ABIST2_VAUX_ABIST_VAUX (0x01U << FS65_W_FS_ABIST2_VAUX_SHIFT)
- #define FS65_W_FS_ABIST2_FS1B_NO_ACTION (0x00U << FS65_W_FS_ABIST2_FS1B_SHIFT)
- #define FS65_W_FS_ABIST2_FS1B_ABIST_FS1B (0x01U << FS65_W_FS_ABIST2_FS1B_SHIFT)
- #define FS65_R_FS_RSTB_SNS_MASK 0x01U
- #define FS65_R_FS_FS0B_SNS_MASK 0x02U
- #define FS65_R_FS_FS1B_SNS_MASK 0x04U
- #define FS65_W_FS_RELEASE_FSXB_MASK 0xFFU
- #define FS65_R_FS_RSTB_SNS_SHIFT 0x00U
- #define FS65_W_FS_RELEASE_FSXB_SHIFT 0x00U
- #define FS65_R_FS_FS0B_SNS_SHIFT 0x01U

- #define FS65_R_FS_FS1B_SNS_SHIFT 0x02U
- #define FS65_R_FS_RSTB_SNS_LOW (0x00U << FS65_R_FS_RSTB_SNS_SHIFT)
- #define FS65_R_FS_RSTB_SNS_HIGH (0x01U << FS65_R_FS_RSTB_SNS_SHIFT)
- #define FS65_R_FS_FS0B_SNS_LOW (0x00U << FS65_R_FS_FS0B_SNS_SHIFT)
- #define FS65_R_FS_FS0B_SNS_HIGH (0x01U << FS65_R_FS_FS0B_SNS_SHIFT)
- #define FS65_R_FS_FS1B_SNS_LOW (0x00U << FS65_R_FS_FS1B_SNS_SHIFT)
- #define FS65_R_FS_FS1B_SNS_HIGH (0x01U << FS65_R_FS_FS1B_SNS_SHIFT)
- #define FS65_R_FS_RSTB_DRV_MASK 0x01U
- #define FS65_R_FS_FS0B_DRV_MASK 0x02U
- #define FS65_R_FS_FS1B_DLY_DRV_MASK 0x04U
- #define FS65_R_FS_FS1B_DRV_MASK 0x08U
- #define FS65_W_FS_RSTB_REQ_MASK 0x10U
- #define FS65_W_FS_FS0B_REQ_MASK 0x20U
- #define FS65_W_FS_FS1B_DLY_REQ_MASK 0x40U
- #define FS65_W_FS_FS1B_REQ_MASK 0x80U
- #define FS65_R_FS_RSTB_DRV_SHIFT 0x00U
- #define FS65_R_FS_FS0B_DRV_SHIFT 0x01U
- #define FS65_R_FS_FS1B_DLY_DRV_SHIFT 0x02U
- #define FS65_R_FS_FS1B_DRV_SHIFT 0x03U
- #define FS65_W_FS_RSTB_REQ_SHIFT 0x04U
- #define FS65_W_FS_FS0B_REQ_SHIFT 0x05U
- #define FS65_W_FS_FS1B_DLY_REQ_SHIFT 0x06U
- #define FS65_W_FS_FS1B_REQ_SHIFT 0x07U
- #define FS65_R_FS_RSTB_DRV_LOW (0x00U << FS65_R_FS_RSTB_DRV_SHIFT)
- #define FS65_R_FS_RSTB_DRV_HIGH (0x01U << FS65_R_FS_RSTB_DRV_SHIFT)
- #define FS65_R_FS_FS0B_DRV_LOW (0x00U << FS65_R_FS_FS0B_DRV_SHIFT)
- #define FS65_R_FS_FS0B_DRV_HIGH (0x01U << FS65_R_FS_FS0B_DRV_SHIFT)
- #define FS65_R_FS_FS1B_DLY_DRV_FS1B_LOW (0x00U << FS65_R_FS_FS1B_DLY_DRV_SHIFT)
- #define FS65_R_FS_FS1B_DLY_DRV_FS1B_HIGH (0x01U << FS65_R_FS_FS1B_DLY_DRV_SHIFT)
- #define FS65_R_FS_FS1B_DRV_FS1B_LOW (0x00U << FS65_R_FS_FS1B_DRV_SHIFT)
- #define FS65_R_FS_FS1B_DRV_FS1B_HIGH (0x01U << FS65_R_FS_FS1B_DRV_SHIFT)
- #define FS65_W_FS_RSTB_REQ_NO_REQUEST (0x00U << FS65_W_FS_RSTB_REQ_SHIFT)
- #define FS65_W_FS_RSTB_REQ_RSTB_REQ (0x01U << FS65_W_FS_RSTB_REQ_SHIFT)
- #define FS65_W_FS_FS0B_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS0B_REQ_SHIFT)
- #define FS65_W_FS_FS0B_REQ_FS0B_REQ (0x01U << FS65_W_FS_FS0B_REQ_SHIFT)
- #define FS65_W_FS_FS1B_DLY_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS1B_DLY_REQ_SH↩ IFT)
- #define FS65_W_FS_FS1B_DLY_REQ_FS1B_REQ (0x01U << FS65_W_FS_FS1B_DLY_REQ_SHIFT)
- #define FS65_W_FS_FS1B_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS1B_REQ_SHIFT)
- #define FS65_W_FS_FS1B_REQ_FS1B_REQ (0x01U << FS65_W_FS_FS1B_REQ_SHIFT)
- #define FS65_RW_M_WU_IO4_MASK 0x03U
- #define FS65_RW_M_WU_IO3_MASK 0x0CU
- #define FS65_RW_M_WU_IO2_MASK 0x30U
- #define FS65_RW_M_WU_IO0_MASK 0xC0U
- #define FS65_RW_M_WU_IO4_SHIFT 0x00U
- #define FS65_RW_M_WU_IO3_SHIFT 0x02U
- #define FS65_RW_M_WU_IO2_SHIFT 0x04U
- #define FS65_RW_M_WU_IO0_SHIFT 0x06U
- #define FS65_RW_M_WU_IO4_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO4_SHIFT)
- #define FS65_RW_M_WU_IO4_RISING_EDGE (0x01U << FS65_RW_M_WU_IO4_SHIFT)
- #define FS65_RW_M_WU_IO4_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO4_SHIFT)
- #define FS65_RW_M_WU_IO4_ANY_EDGE (0x03U << FS65_RW_M_WU_IO4_SHIFT)
- #define FS65_RW_M_WU_IO3_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO3_SHIFT)
- #define FS65_RW_M_WU_IO3_RISING_EDGE (0x01U << FS65_RW_M_WU_IO3_SHIFT)
- #define FS65_RW_M_WU_IO3_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO3_SHIFT)

- #define FS65_RW_M_WU_IO3_ANY_EDGE (0x03U << FS65_RW_M_WU_IO3_SHIFT)
- #define FS65_RW_M_WU_IO2_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO2_SHIFT)
- #define FS65_RW_M_WU_IO2_RISING_EDGE (0x01U << FS65_RW_M_WU_IO2_SHIFT)
- #define FS65_RW_M_WU_IO2_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO2_SHIFT)
- #define FS65_RW_M_WU_IO2_ANY_EDGE (0x03U << FS65_RW_M_WU_IO2_SHIFT)
- #define FS65_RW_M_WU_IO0_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO0_SHIFT)
- #define FS65_RW_M_WU_IO0_RISING_EDGE (0x01U << FS65_RW_M_WU_IO0_SHIFT)
- #define FS65_RW_M_WU_IO0_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO0_SHIFT)
- #define FS65_RW_M_WU_IO0_ANY_EDGE (0x03U << FS65_RW_M_WU_IO0_SHIFT)
- #define FS65_RW_M_LIN_SR_MASK 0x03U
- #define FS65_RW_M_LIN_J2602_DIS_MASK 0x04U
- #define FS65_RW_M_CAN_WU_TO_MASK 0x10U
- #define FS65_RW_M_CAN_DIS_CFG_MASK 0x20U
- #define FS65_RW_M_WU_IO5_MASK 0xC0U
- #define FS65_RW_M_LIN_SR_SHIFT 0x00U
- #define FS65_RW_M_LIN_J2602_DIS_SHIFT 0x02U
- #define FS65_RW_M_CAN_WU_TO_SHIFT 0x04U
- #define FS65_RW_M_CAN_DIS_CFG_SHIFT 0x05U
- #define FS65_RW_M_WU_IO5_SHIFT 0x06U
- #define FS65_RW_M_LIN_SR_20KBITS (0x00U << FS65_RW_M_LIN_SR_SHIFT)
- #define FS65_RW_M_LIN_SR_10KBITS (0x01U << FS65_RW_M_LIN_SR_SHIFT)
- #define FS65_RW_M_LIN_SR_FAST_RATE (0x02U << FS65_RW_M_LIN_SR_SHIFT)
- #define FS65_RW_M_LIN_J2602_DIS_COMPLIANT (0x00U << FS65_RW_M_LIN_J2602_DIS_SHIFT)
- #define FS65_RW_M_LIN_J2602_DIS_NOT_COMPLIANT (0x01U << FS65_RW_M_LIN_J2602_DIS_S↩ HIFT)
- #define FS65_RW_M_CAN_WU_TO_120US (0x00U << FS65_RW_M_CAN_WU_TO_SHIFT)
- #define FS65_RW_M_CAN_WU_TO_2_8MS (0x01U << FS65_RW_M_CAN_WU_TO_SHIFT)
- #define FS65_RW_M_CAN_DIS_CFG_RX_ONLY (0x00U << FS65_RW_M_CAN_DIS_CFG_SHIFT)
- #define FS65_RW_M_CAN_DIS_CFG_SLEEP (0x01U << FS65_RW_M_CAN_DIS_CFG_SHIFT)
- #define FS65_RW_M_WU_IO5_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO5_SHIFT)
- #define FS65_RW_M_WU_IO5_RISING_EDGE (0x01U << FS65_RW_M_WU_IO5_SHIFT)
- #define FS65_RW_M_WU_IO5_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO5_SHIFT)
- #define FS65_RW_M_WU_IO5_ANY_EDGE (0x03U << FS65_RW_M_WU_IO5_SHIFT)
- #define FS65_RW_M_INT_INH_0_MASK 0x01U
- #define FS65_RW_M_INT_INH_2_MASK 0x02U
- #define FS65_RW_M_INT_INH_3_MASK 0x04U
- #define FS65_RW_M_INT_INH_4_MASK 0x08U
- #define FS65_RW_M_INT_INH_5_MASK 0x10U
- #define FS65_RW_M_INT_INH_0_SHIFT 0x00U
- #define FS65_RW_M_INT_INH_2_SHIFT 0x01U
- #define FS65_RW_M_INT_INH_3_SHIFT 0x02U
- #define FS65_RW_M_INT_INH_4_SHIFT 0x03U
- #define FS65_RW_M_INT_INH_5_SHIFT 0x04U
- #define FS65_RW_M_INT_INH_0_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_0_SHIFT)
- #define FS65_RW_M_INT_INH_0_MASKED (0x01U << FS65_RW_M_INT_INH_0_SHIFT)
- #define FS65_RW_M_INT_INH_2_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_2_SHIFT)
- #define FS65_RW_M_INT_INH_2_MASKED (0x01U << FS65_RW_M_INT_INH_2_SHIFT)
- #define FS65_RW_M_INT_INH_3_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_3_SHIFT)
- #define FS65_RW_M_INT_INH_3_MASKED (0x01U << FS65_RW_M_INT_INH_3_SHIFT)
- #define FS65_RW_M_INT_INH_4_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_4_SHIFT)
- #define FS65_RW_M_INT_INH_4_MASKED (0x01U << FS65_RW_M_INT_INH_4_SHIFT)
- #define FS65_RW_M_INT_INH_5_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_5_SHIFT)
- #define FS65_RW_M_INT_INH_5_MASKED (0x01U << FS65_RW_M_INT_INH_5_SHIFT)
- #define FS65_R_FS_FS1B_TIME_MASK 0x0FU
- #define FS65_W_FS_FS1B_TIME_MASK 0xF0U

- #define FS65_R_FS_FS1B_TIME_SHIFT 0x00U
- #define FS65_W_FS_FS1B_TIME_SHIFT 0x04U
- #define FS65_R_FS_FS1B_TIME_0_0 (0x00U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_10MS_80MS (0x01U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_13_104MS (0x02U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_17_135MS (0x03U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_22_176MS (0x04U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_29_228MS (0x05U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_37_297MS (0x06U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_48_386MS (0x07U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_63_502MS (0x08U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_82_653MS (0x09U << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_106_848MS (0x0AU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_138_1103MS (0x0BU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_179_1434MS (0x0CU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_233_1864MS (0x0DU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_303_2423MS (0x0EU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_394_3150MS (0x0FU << FS65_R_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_0_0 (0x00U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_10MS_80MS (0x01U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_13_104MS (0x02U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_17_135MS (0x03U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_22_176MS (0x04U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_29_228MS (0x05U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_37_297MS (0x06U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_48_386MS (0x07U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_63_502MS (0x08U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_82_653MS (0x09U << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_106_848MS (0x0AU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_138_1103MS (0x0BU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_179_1434MS (0x0CU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_233_1864MS (0x0DU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_303_2423MS (0x0EU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_W_FS_FS1B_TIME_394_3150MS (0x0FU << FS65_W_FS_FS1B_TIME_SHIFT)
- #define FS65_R_FS_FS1B_TIME_RANGE_MASK 0x01U
- #define FS65_R_FS_VAUX_5D_MASK 0x02U
- #define FS65_R_FS_VCCA_5D_MASK 0x04U
- #define FS65_R_FS_VCORE_5D_MASK 0x08U
- #define FS65_W_FS_FS1B_TIME_RANGE_MASK 0x10U
- #define FS65_W_FS_VAUX_5D_MASK 0x20U
- #define FS65_W_FS_VCCA_5D_MASK 0x40U
- #define FS65_W_FS_VCORE_5D_MASK 0x80U
- #define FS65_R_FS_FS1B_TIME_RANGE_SHIFT 0x00U
- #define FS65_R_FS_VAUX_5D_SHIFT 0x01U
- #define FS65_R_FS_VCCA_5D_SHIFT 0x02U
- #define FS65_R_FS_VCORE_5D_SHIFT 0x03U
- #define FS65_W_FS_FS1B_TIME_RANGE_SHIFT 0x04U
- #define FS65_W_FS_VAUX_5D_SHIFT 0x05U
- #define FS65_W_FS_VCCA_5D_SHIFT 0x06U
- #define FS65_W_FS_VCORE_5D_SHIFT 0x07U
- #define FS65_R_FS_FS1B_TIME_RANGE_X1 (0x00U << FS65_R_FS_FS1B_TIME_RANGE_SHIFT)
- #define FS65_R_FS_FS1B_TIME_RANGE_X8 (0x01U << FS65_R_FS_FS1B_TIME_RANGE_SHIFT)
- #define FS65_R_FS_VAUX_5D_NORMAL (0x00U << FS65_R_FS_VAUX_5D_SHIFT)
- #define FS65_R_FS_VAUX_5D_DEGRADED (0x01U << FS65_R_FS_VAUX_5D_SHIFT)
- #define FS65_R_FS_VCCA_5D_NORMAL (0x00U << FS65_R_FS_VCCA_5D_SHIFT)

- #define FS65_R_FS_VCCA_5D_DEGRADED (0x01U << FS65_R_FS_VCCA_5D_SHIFT)
- #define FS65_R_FS_VCORE_5D_NORMAL (0x00U << FS65_R_FS_VCORE_5D_SHIFT)
- #define FS65_R_FS_VCORE_5D_DEGRADED (0x01U << FS65_R_FS_VCORE_5D_SHIFT)
- #define FS65_W_FS_FS1B_TIME_RANGE_X1 (0x00U << FS65_W_FS_FS1B_TIME_RANGE_SHIFT)
- #define FS65_W_FS_FS1B_TIME_RANGE_X8 (0x01U << FS65_W_FS_FS1B_TIME_RANGE_SHIFT)
- #define FS65_W_FS_VAUX_5D_NORMAL (0x00U << FS65_W_FS_VAUX_5D_SHIFT)
- #define FS65_W_FS_VAUX_5D_DEGRADED (0x01U << FS65_W_FS_VAUX_5D_SHIFT)
- #define FS65_W_FS_VCCA_5D_NORMAL (0x00U << FS65_W_FS_VCCA_5D_SHIFT)
- #define FS65_W_FS_VCCA_5D_DEGRADED (0x01U << FS65_W_FS_VCCA_5D_SHIFT)
- #define FS65_W_FS_VCORE_5D_NORMAL (0x00U << FS65_W_FS_VCORE_5D_SHIFT)
- #define FS65_W_FS_VCORE_5D_DEGRADED (0x01U << FS65_W_FS_VCORE_5D_SHIFT)
- #define FS65_R_FS_FLT_ERR_IMP_MASK 0x03U
- #define FS65_R_FS_FS1B_CAN_IMPACT_MASK 0x04U
- #define FS65_R_FS_FLT_ERR_FS_MASK 0x08U
- #define FS65_W_FS_FLT_ERR_IMP_MASK 0x30U
- #define FS65_W_FS_FS1B_CAN_IMPACT_MASK 0x40U
- #define FS65_W_FS_FLT_ERR_FS_MASK 0x80U
- #define FS65_R_FS_FLT_ERR_IMP_SHIFT 0x00U
- #define FS65_R_FS_FS1B_CAN_IMPACT_SHIFT 0x02U
- #define FS65_R_FS_FLT_ERR_FS_SHIFT 0x03U
- #define FS65_W_FS_FLT_ERR_IMP_SHIFT 0x04U
- #define FS65_W_FS_FS1B_CAN_IMPACT_SHIFT 0x06U
- #define FS65_W_FS_FLT_ERR_FS_SHIFT 0x07U
- #define FS65_R_FS_FLT_ERR_IMP_NO_EFFECT (0x00U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_R_FS_FLT_ERR_IMP_FS0B (0x01U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_R_FS_FLT_ERR_IMP_RSTB (0x02U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_R_FS_FLT_ERR_IMP_FS0B_RSTB (0x03U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_R_FS_FS1B_CAN_IMPACT_NO_EFFECT (0x00U << FS65_R_FS_FS1B_CAN_IMPACT↩ _SHIFT)
- #define FS65_R_FS_FS1B_CAN_IMPACT_RX_ONLY (0x01U << FS65_R_FS_FS1B_CAN_IMPACT_S↩ HIFT)
- #define FS65_R_FS_FLT_ERR_FS_INT3_FIN6 (0x00U << FS65_R_FS_FLT_ERR_FS_SHIFT)
- #define FS65_R_FS_FLT_ERR_FS_INT1_FIN2 (0x01U << FS65_R_FS_FLT_ERR_FS_SHIFT)
- #define FS65_W_FS_FLT_ERR_IMP_NO_EFFECT (0x00U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_W_FS_FLT_ERR_IMP_FS0B (0x01U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_W_FS_FLT_ERR_IMP_RSTB (0x02U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_W_FS_FLT_ERR_IMP_FS0B_RSTB (0x03U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
- #define FS65_W_FS_FS1B_CAN_IMPACT_NO_EFFECT (0x00U << FS65_W_FS_FS1B_CAN_IMPAC↩ T_SHIFT)
- #define FS65_W_FS_FS1B_CAN_IMPACT_RX_ONLY (0x01U << FS65_W_FS_FS1B_CAN_IMPACT_↩ SHIFT)
- #define FS65_W_FS_FLT_ERR_FS_INT3_FIN6 (0x00U << FS65_W_FS_FLT_ERR_FS_SHIFT)
- #define FS65_W_FS_FLT_ERR_FS_INT1_FIN2 (0x01U << FS65_W_FS_FLT_ERR_FS_SHIFT)
- #define FS65_R_FS_RSTB_DURATION_MASK 0x01U
- #define FS65_R_FS_PS_MASK 0x02U
- #define FS65_R_FS_IO_23_FS_MASK 0x04U
- #define FS65_R_FS_IO_45_FS_MASK 0x08U
- #define FS65_W_FS_RSTB_DURATION_MASK 0x10U
- #define FS65_W_FS_PS_MASK 0x20U
- #define FS65_W_FS_IO_23_FS_MASK 0x40U
- #define FS65_W_FS_IO_45_FS_MASK 0x80U
- #define FS65_R_FS_RSTB_DURATION_SHIFT 0x00U
- #define FS65_R_FS_PS_SHIFT 0x01U
- #define FS65_R_FS_IO_23_FS_SHIFT 0x02U
- #define FS65_R_FS_IO_45_FS_SHIFT 0x03U

- #define FS65_W_FS_RSTB_DURATION_SHIFT 0x04U
- #define FS65_W_FS_PS_SHIFT 0x05U
- #define FS65_W_FS_IO_23_FS_SHIFT 0x06U
- #define FS65_W_FS_IO_45_FS_SHIFT 0x07U
- #define FS65_R_FS_RSTB_DURATION_10MS (0x00U << FS65_R_FS_RSTB_DURATION_SHIFT)
- #define FS65_R_FS_RSTB_DURATION_1MS (0x01U << FS65_R_FS_RSTB_DURATION_SHIFT)
- #define FS65_R_FS_PS_HIGH (0x00U << FS65_R_FS_PS_SHIFT)
- #define FS65_R_FS_PS_LOW (0x01U << FS65_R_FS_PS_SHIFT)
- #define FS65_R_FS_IO_23_FS_NOT_SAFETY (0x00U << FS65_R_FS_IO_23_FS_SHIFT)
- #define FS65_R_FS_IO_23_FS_SAFETY_CRITICAL (0x01U << FS65_R_FS_IO_23_FS_SHIFT)
- #define FS65_R_FS_IO_45_FS_NOT_SAFETY (0x00U << FS65_R_FS_IO_45_FS_SHIFT)
- #define FS65_R_FS_IO_45_FS_SAFETY_CRITICAL (0x01U << FS65_R_FS_IO_45_FS_SHIFT)
- #define FS65_W_FS_RSTB_DURATION_10MS (0x00U << FS65_W_FS_RSTB_DURATION_SHIFT)
- #define FS65_W_FS_RSTB_DURATION_1MS (0x01U << FS65_W_FS_RSTB_DURATION_SHIFT)
- #define FS65_W_FS_PS_HIGH (0x00U << FS65_W_FS_PS_SHIFT)
- #define FS65_W_FS_PS_LOW (0x01U << FS65_W_FS_PS_SHIFT)
- #define FS65_W_FS_IO_23_FS_NOT_SAFETY (0x00U << FS65_W_FS_IO_23_FS_SHIFT)
- #define FS65_W_FS_IO_23_FS_SAFETY_CRITICAL (0x01U << FS65_W_FS_IO_23_FS_SHIFT)
- #define FS65_W_FS_IO_45_FS_NOT_SAFETY (0x00U << FS65_W_FS_IO_45_FS_SHIFT)
- #define FS65_W_FS_IO_45_FS_SAFETY_CRITICAL (0x01U << FS65_W_FS_IO_45_FS_SHIFT)
- #define FS65_R_FS_WD_IMPACT_MASK 0x03U
- #define FS65_R_FS_DIS_8S_MASK 0x04U
- #define FS65_R_FS_TDLY_TDUR_MASK 0x08U
- #define FS65_W_FS_WD_IMPACT_MASK 0x30U
- #define FS65_W_FS_DIS_8S_MASK 0x40U
- #define FS65_W_FS_TDLY_TDUR_MASK 0x80U
- #define FS65_R_FS_WD_IMPACT_SHIFT 0x00U
- #define FS65_R_FS_DIS_8S_SHIFT 0x02U
- #define FS65_R_FS_TDLY_TDUR_SHIFT 0x03U
- #define FS65_W_FS_WD_IMPACT_SHIFT 0x04U
- #define FS65_W_FS_DIS_8S_SHIFT 0x06U
- #define FS65_W_FS_TDLY_TDUR_SHIFT 0x07U
- #define FS65_R_FS_WD_IMPACT_NO_EFFECT (0x00U << FS65_R_FS_WD_IMPACT_SHIFT)
- #define FS65_R_FS_WD_IMPACT_RSTB (0x01U << FS65_R_FS_WD_IMPACT_SHIFT)
- #define FS65_R_FS_WD_IMPACT_FS0B (0x02U << FS65_R_FS_WD_IMPACT_SHIFT)
- #define FS65_R_FS_WD_IMPACT_RSTB_FS0B (0x03U << FS65_R_FS_WD_IMPACT_SHIFT)
- #define FS65_R_FS_DIS_8S_ENABLED (0x00U << FS65_R_FS_DIS_8S_SHIFT)
- #define FS65_R_FS_DIS_8S_DISABLED (0x01U << FS65_R_FS_DIS_8S_SHIFT)
- #define FS65_R_FS_TDLY_TDUR_DELAY (0x00U << FS65_R_FS_TDLY_TDUR_SHIFT)
- #define FS65_R_FS_TDLY_TDUR_DURATION (0x01U << FS65_R_FS_TDLY_TDUR_SHIFT)
- #define FS65_W_FS_WD_IMPACT_NO_EFFECT (0x00U << FS65_W_FS_WD_IMPACT_SHIFT)
- #define FS65_W_FS_WD_IMPACT_RSTB (0x01U << FS65_W_FS_WD_IMPACT_SHIFT)
- #define FS65_W_FS_WD_IMPACT_FS0B (0x02U << FS65_W_FS_WD_IMPACT_SHIFT)
- #define FS65_W_FS_WD_IMPACT_RSTB_FS0B (0x03U << FS65_W_FS_WD_IMPACT_SHIFT)
- #define FS65_W_FS_DIS_8S_ENABLED (0x00U << FS65_W_FS_DIS_8S_SHIFT)
- #define FS65_W_FS_DIS_8S_DISABLED (0x01U << FS65_W_FS_DIS_8S_SHIFT)
- #define FS65_W_FS_TDLY_TDUR_DELAY (0x00U << FS65_W_FS_TDLY_TDUR_SHIFT)
- #define FS65_W_FS_TDLY_TDUR_DURATION (0x01U << FS65_W_FS_TDLY_TDUR_SHIFT)
- #define FS65_R_FS_WD_WINDOW_MASK 0x0FU
- #define FS65_W_FS_WD_WINDOW_MASK 0xF0U
- #define FS65_R_FS_WD_WINDOW_SHIFT 0x00U
- #define FS65_W_FS_WD_WINDOW_SHIFT 0x04U
- #define FS65_R_FS_WD_WINDOW_DISABLE (0x00U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_1MS (0x01U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_2MS (0x02U << FS65_R_FS_WD_WINDOW_SHIFT)

- #define FS65_R_FS_WD_WINDOW_3MS (0x03U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_4MS (0x04U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_6MS (0x05U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_8MS (0x06U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_12MS (0x07U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_16MS (0x08U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_24MS (0x09U << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_32MS (0x0AU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_64MS (0x0BU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_128MS (0x0CU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_256MS (0x0DU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_512MS (0x0EU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_R_FS_WD_WINDOW_1024MS (0x0FU << FS65_R_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_DISABLE (0x00U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_1MS (0x01U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_2MS (0x02U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_3MS (0x03U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_4MS (0x04U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_6MS (0x05U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_8MS (0x06U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_12MS (0x07U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_16MS (0x08U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_24MS (0x09U << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_32MS (0x0AU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_64MS (0x0BU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_128MS (0x0CU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_256MS (0x0DU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_512MS (0x0EU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_W_FS_WD_WINDOW_1024MS (0x0FU << FS65_W_FS_WD_WINDOW_SHIFT)
- #define FS65_RW_FS_WD_LFSR_MASK 0xFFU
- #define FS65_RW_FS_WD_LFSR_SHIFT 0x00U
- #define FS65_R_FS_WD_CNT_RFR_MASK 0x03U
- #define FS65_R_FS_WD_CNT_ERR_MASK 0x0CU
- #define FS65_W_FS_WD_CNT_RFR_MASK 0x30U
- #define FS65_W_FS_WD_CNT_ERR_MASK 0xC0U
- #define FS65_R_FS_WD_CNT_RFR_SHIFT 0x00U
- #define FS65_R_FS_WD_CNT_ERR_SHIFT 0x02U
- #define FS65_W_FS_WD_CNT_RFR_SHIFT 0x04U
- #define FS65_W_FS_WD_CNT_ERR_SHIFT 0x06U
- #define FS65_R_FS_WD_CNT_RFR_6 (0x00U << FS65_R_FS_WD_CNT_RFR_SHIFT)
- #define FS65_R_FS_WD_CNT_RFR_4 (0x01U << FS65_R_FS_WD_CNT_RFR_SHIFT)
- #define FS65_R_FS_WD_CNT_RFR_2 (0x02U << FS65_R_FS_WD_CNT_RFR_SHIFT)
- #define FS65_R_FS_WD_CNT_RFR_1 (0x03U << FS65_R_FS_WD_CNT_RFR_SHIFT)
- #define FS65_R_FS_WD_CNT_ERR_6 (0x00U << FS65_R_FS_WD_CNT_ERR_SHIFT)
- #define FS65_R_FS_WD_CNT_ERR_4 (0x02U << FS65_R_FS_WD_CNT_ERR_SHIFT)
- #define FS65_R_FS_WD_CNT_ERR_2 (0x03U << FS65_R_FS_WD_CNT_ERR_SHIFT)
- #define FS65_W_FS_WD_CNT_RFR_6 (0x00U << FS65_W_FS_WD_CNT_RFR_SHIFT)
- #define FS65_W_FS_WD_CNT_RFR_4 (0x01U << FS65_W_FS_WD_CNT_RFR_SHIFT)
- #define FS65_W_FS_WD_CNT_RFR_2 (0x02U << FS65_W_FS_WD_CNT_RFR_SHIFT)
- #define FS65_W_FS_WD_CNT_RFR_1 (0x03U << FS65_W_FS_WD_CNT_RFR_SHIFT)
- #define FS65_W_FS_WD_CNT_ERR_6 (0x00U << FS65_W_FS_WD_CNT_ERR_SHIFT)
- #define FS65_W_FS_WD_CNT_ERR_4 (0x02U << FS65_W_FS_WD_CNT_ERR_SHIFT)
- #define FS65_W_FS_WD_CNT_ERR_2 (0x03U << FS65_W_FS_WD_CNT_ERR_SHIFT)
- #define FS65_R_FS_VCORE_FS_UV_MASK 0x03U
- #define FS65_R_FS_VCORE_FS_OV_MASK 0x0CU

- #define FS65_W_FS_VCORE_FS_UV_MASK 0x30U
- #define FS65_W_FS_VCORE_FS_OV_MASK 0xC0U
- #define FS65_R_FS_VCORE_FS_UV_SHIFT 0x00U
- #define FS65_R_FS_VCORE_FS_OV_SHIFT 0x02U
- #define FS65_W_FS_VCORE_FS_UV_SHIFT 0x04U
- #define FS65_W_FS_VCORE_FS_OV_SHIFT 0x06U
- #define FS65_R_FS_VCORE_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VCORE_FS_UV_SHIFT)
- #define FS65_R_FS_VCORE_FS_UV_RSTB (0x01U << FS65_R_FS_VCORE_FS_UV_SHIFT)
- #define FS65_R_FS_VCORE_FS_UV_FS0B (0x02U << FS65_R_FS_VCORE_FS_UV_SHIFT)
- #define FS65_R_FS_VCORE_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VCORE_FS_UV_SHIFT)
- #define FS65_R_FS_VCORE_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VCORE_FS_OV_SHIFT)
- #define FS65_R_FS_VCORE_FS_OV_RSTB (0x01U << FS65_R_FS_VCORE_FS_OV_SHIFT)
- #define FS65_R_FS_VCORE_FS_OV_FS0B (0x02U << FS65_R_FS_VCORE_FS_OV_SHIFT)
- #define FS65_R_FS_VCORE_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VCORE_FS_OV_SHIFT)
- #define FS65_W_FS_VCORE_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VCORE_FS_UV_SHIFT)
- #define FS65_W_FS_VCORE_FS_UV_RSTB (0x01U << FS65_W_FS_VCORE_FS_UV_SHIFT)
- #define FS65_W_FS_VCORE_FS_UV_FS0B (0x02U << FS65_W_FS_VCORE_FS_UV_SHIFT)
- #define FS65_W_FS_VCORE_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VCORE_FS_UV_SHIFT)
- #define FS65_W_FS_VCORE_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VCORE_FS_OV_SHIFT)
- #define FS65_W_FS_VCORE_FS_OV_RSTB (0x01U << FS65_W_FS_VCORE_FS_OV_SHIFT)
- #define FS65_W_FS_VCORE_FS_OV_FS0B (0x02U << FS65_W_FS_VCORE_FS_OV_SHIFT)
- #define FS65_W_FS_VCORE_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VCORE_FS_OV_SHIFT)
- #define FS65_R_FS_VCCA_FS_UV_MASK 0x03U
- #define FS65_R_FS_VCCA_FS_OV_MASK 0x0CU
- #define FS65_W_FS_VCCA_FS_UV_MASK 0x30U
- #define FS65_W_FS_VCCA_FS_OV_MASK 0xC0U
- #define FS65_R_FS_VCCA_FS_UV_SHIFT 0x00U
- #define FS65_R_FS_VCCA_FS_OV_SHIFT 0x02U
- #define FS65_W_FS_VCCA_FS_UV_SHIFT 0x04U
- #define FS65_W_FS_VCCA_FS_OV_SHIFT 0x06U
- #define FS65_R_FS_VCCA_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VCCA_FS_UV_SHIFT)
- #define FS65_R_FS_VCCA_FS_UV_RSTB (0x01U << FS65_R_FS_VCCA_FS_UV_SHIFT)
- #define FS65_R_FS_VCCA_FS_UV_FS0B (0x02U << FS65_R_FS_VCCA_FS_UV_SHIFT)
- #define FS65_R_FS_VCCA_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VCCA_FS_UV_SHIFT)
- #define FS65_R_FS_VCCA_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VCCA_FS_OV_SHIFT)
- #define FS65_R_FS_VCCA_FS_OV_RSTB (0x01U << FS65_R_FS_VCCA_FS_OV_SHIFT)
- #define FS65_R_FS_VCCA_FS_OV_FS0B (0x02U << FS65_R_FS_VCCA_FS_OV_SHIFT)
- #define FS65_R_FS_VCCA_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VCCA_FS_OV_SHIFT)
- #define FS65_W_FS_VCCA_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VCCA_FS_UV_SHIFT)
- #define FS65_W_FS_VCCA_FS_UV_RSTB (0x01U << FS65_W_FS_VCCA_FS_UV_SHIFT)
- #define FS65_W_FS_VCCA_FS_UV_FS0B (0x02U << FS65_W_FS_VCCA_FS_UV_SHIFT)
- #define FS65_W_FS_VCCA_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VCCA_FS_UV_SHIFT)
- #define FS65_W_FS_VCCA_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VCCA_FS_OV_SHIFT)
- #define FS65_W_FS_VCCA_FS_OV_RSTB (0x01U << FS65_W_FS_VCCA_FS_OV_SHIFT)
- #define FS65_W_FS_VCCA_FS_OV_FS0B (0x02U << FS65_W_FS_VCCA_FS_OV_SHIFT)
- #define FS65_W_FS_VCCA_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VCCA_FS_OV_SHIFT)
- #define FS65_R_FS_VAUX_FS_UV_MASK 0x03U
- #define FS65_R_FS_VAUX_FS_OV_MASK 0x0CU
- #define FS65_W_FS_VAUX_FS_UV_MASK 0x30U
- #define FS65_W_FS_VAUX_FS_OV_MASK 0xC0U
- #define FS65_R_FS_VAUX_FS_UV_SHIFT 0x00U
- #define FS65_R_FS_VAUX_FS_OV_SHIFT 0x02U
- #define FS65_W_FS_VAUX_FS_UV_SHIFT 0x04U
- #define FS65_W_FS_VAUX_FS_OV_SHIFT 0x06U
- #define FS65_R_FS_VAUX_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VAUX_FS_UV_SHIFT)

- #define FS65_R_FS_VAUX_FS_UV_RSTB (0x01U << FS65_R_FS_VAUX_FS_UV_SHIFT)
- #define FS65_R_FS_VAUX_FS_UV_FS0B (0x02U << FS65_R_FS_VAUX_FS_UV_SHIFT)
- #define FS65_R_FS_VAUX_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VAUX_FS_UV_SHIFT)
- #define FS65_R_FS_VAUX_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VAUX_FS_OV_SHIFT)
- #define FS65_R_FS_VAUX_FS_OV_RSTB (0x01U << FS65_R_FS_VAUX_FS_OV_SHIFT)
- #define FS65_R_FS_VAUX_FS_OV_FS0B (0x02U << FS65_R_FS_VAUX_FS_OV_SHIFT)
- #define FS65_R_FS_VAUX_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VAUX_FS_OV_SHIFT)
- #define FS65_W_FS_VAUX_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VAUX_FS_UV_SHIFT)
- #define FS65_W_FS_VAUX_FS_UV_RSTB (0x01U << FS65_W_FS_VAUX_FS_UV_SHIFT)
- #define FS65_W_FS_VAUX_FS_UV_FS0B (0x02U << FS65_W_FS_VAUX_FS_UV_SHIFT)
- #define FS65_W_FS_VAUX_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VAUX_FS_UV_SHIFT)
- #define FS65_W_FS_VAUX_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VAUX_FS_OV_SHIFT)
- #define FS65_W_FS_VAUX_FS_OV_RSTB (0x01U << FS65_W_FS_VAUX_FS_OV_SHIFT)
- #define FS65_W_FS_VAUX_FS_OV_FS0B (0x02U << FS65_W_FS_VAUX_FS_OV_SHIFT)
- #define FS65_W_FS_VAUX_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VAUX_FS_OV_SHIFT)
- #define FS65_RW_M_INT_INH_CAN_MASK 0x01U
- #define FS65_RW_M_INT_INH_VOTHER_MASK 0x02U
- #define FS65_RW_M_INT_INH_VCORE_MASK 0x04U
- #define FS65_RW_M_INT_INH_VPRE_MASK 0x08U
- #define FS65_RW_M_INT_INH_VSNS_MASK 0x10U
- #define FS65_RW_M_INT_INH_ALL_MASK 0x20U
- #define FS65_RW_M_INT_INH_LIN_MASK 0x40U
- #define FS65_RW_M_NT_DURATION_MASK 0x80U
- #define FS65_RW_M_INT_INH_CAN_SHIFT 0x00U
- #define FS65_RW_M_INT_INH_VOTHER_SHIFT 0x01U
- #define FS65_RW_M_INT_INH_VCORE_SHIFT 0x02U
- #define FS65_RW_M_INT_INH_VPRE_SHIFT 0x03U
- #define FS65_RW_M_INT_INH_VSNS_SHIFT 0x04U
- #define FS65_RW_M_INT_INH_ALL_SHIFT 0x05U
- #define FS65_RW_M_INT_INH_LIN_SHIFT 0x06U
- #define FS65_RW_M_NT_DURATION_SHIFT 0x07U
- #define FS65_RW_M_INT_INH_CAN_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_CAN_SHIFT)
- #define FS65_RW_M_INT_INH_CAN_CAN_INHIBITED (0x01U << FS65_RW_M_INT_INH_CAN_SHIFT)
- #define FS65_RW_M_INT_INH_VOTHER_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VOTHE↩ R_SHIFT)
- #define FS65_RW_M_INT_INH_VOTHER_VOTHER_INHIBITED (0x01U << FS65_RW_M_INT_INH_V↩ OTHER_SHIFT)
- #define FS65_RW_M_INT_INH_VCORE_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VCORE_↩ SHIFT)
- #define FS65_RW_M_INT_INH_VCORE_VCORE_INHIBITED (0x01U << FS65_RW_M_INT_INH_VCO↩ RE_SHIFT)
- #define FS65_RW_M_INT_INH_VPRE_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VPRE_SHI↩ FT)
- #define FS65_RW_M_INT_INH_VPRE_VPRE_INHIBITED (0x01U << FS65_RW_M_INT_INH_VPRE_S↩ HIFT)
- #define FS65_RW_M_INT_INH_VSNS_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VSNS_SHI↩ FT)
- #define FS65_RW_M_INT_INH_VSNS_VSNS_UV_INHIBITED (0x01U << FS65_RW_M_INT_INH_VSN↩ S_SHIFT)
- #define FS65_RW_M_INT_INH_ALL_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_ALL_SHIFT)
- #define FS65_RW_M_INT_INH_ALL_ALL_INHIBITED (0x01U << FS65_RW_M_INT_INH_ALL_SHIFT)
- #define FS65_RW_M_INT_INH_LIN_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_LIN_SHIFT)
- #define FS65_RW_M_INT_INH_LIN_LIN_INHIBITED (0x01U << FS65_RW_M_INT_INH_LIN_SHIFT)
- #define FS65_RW_M_NT_DURATION_100US (0x00U << FS65_RW_M_NT_DURATION_SHIFT)
- #define FS65_RW_M_NT_DURATION_25US (0x01U << FS65_RW_M_NT_DURATION_SHIFT)

6.7.1 Detailed Description

Register map of the FS65/FS45 SBC series.

Author

nxf44615

Version

1.0

Date

13-Nov-2018

Copyright

Copyright 2016 - 2018 NXP

6.7.2 Macro Definition Documentation

6.7.2.1 FS65_R_FS_ABIST1_OK_FAIL

```
#define FS65_R_FS_ABIST1_OK_FAIL (0x00U << FS65_R_FS_ABIST1_OK_SHIFT)
```

ABIST1 fail

6.7.2.2 FS65_R_FS_ABIST1_OK_MASK

```
#define FS65_R_FS_ABIST1_OK_MASK 0x01U
```

Diagnostic of analog BIST1 (automatically executed)

6.7.2.3 FS65_R_FS_ABIST1_OK_PASS

```
#define FS65_R_FS_ABIST1_OK_PASS (0x01U << FS65_R_FS_ABIST1_OK_SHIFT)
```

ABIST1 pass

6.7.2.4 FS65_R_FS_ABIST1_OK_SHIFT

```
#define FS65_R_FS_ABIST1_OK_SHIFT 0x00U
```

Diagnostic of analog BIST1 (automatically executed)

6.7.2.5 FS65_R_FS_ABIST2_FS1B_OK_FAIL

```
#define FS65_R_FS_ABIST2_FS1B_OK_FAIL (0x00U << FS65_R_FS_ABIST2_FS1B_OK_SHIFT)
```

FS1B ABIST fail or not executed

6.7.2.6 FS65_R_FS_ABIST2_FS1B_OK_MASK

```
#define FS65_R_FS_ABIST2_FS1B_OK_MASK 0x04U
```

Diagnostic of FS1B Analog BIST2 (executed on demand)

6.7.2.7 FS65_R_FS_ABIST2_FS1B_OK_PASS

```
#define FS65_R_FS_ABIST2_FS1B_OK_PASS (0x01U << FS65_R_FS_ABIST2_FS1B_OK_SHIFT)
```

FS1B ABIST pass

6.7.2.8 FS65_R_FS_ABIST2_FS1B_OK_SHIFT

```
#define FS65_R_FS_ABIST2_FS1B_OK_SHIFT 0x02U
```

Diagnostic of FS1B Analog BIST2 (executed on demand)

6.7.2.9 FS65_R_FS_ABIST2_VAUX_OK_FAIL

```
#define FS65_R_FS_ABIST2_VAUX_OK_FAIL (0x00U << FS65_R_FS_ABIST2_VAUX_OK_SHIFT)
```

VAUX ABIST fail or not executed

6.7.2.10 FS65_R_FS_ABIST2_VAUX_OK_MASK

```
#define FS65_R_FS_ABIST2_VAUX_OK_MASK 0x02U
```

Diagnostic of VAUX Analog BIST2 (executed on demand)

6.7.2.11 FS65_R_FS_ABIST2_VAUX_OK_PASS

```
#define FS65_R_FS_ABIST2_VAUX_OK_PASS (0x01U << FS65_R_FS_ABIST2_VAUX_OK_SHIFT)
```

VAUX ABIST pass

6.7.2.12 FS65_R_FS_ABIST2_VAUX_OK_SHIFT

```
#define FS65_R_FS_ABIST2_VAUX_OK_SHIFT 0x01U
```

Diagnostic of VAUX Analog BIST2 (executed on demand)

6.7.2.13 FS65_R_FS_DIS_8S_DISABLED

```
#define FS65_R_FS_DIS_8S_DISABLED (0x01U << FS65_R_FS_DIS_8S_SHIFT)
```

Disabled

6.7.2.14 FS65_R_FS_DIS_8S_ENABLED

```
#define FS65_R_FS_DIS_8S_ENABLED (0x00U << FS65_R_FS_DIS_8S_SHIFT)
```

Enabled

6.7.2.15 FS65_R_FS_DIS_8S_MASK

```
#define FS65_R_FS_DIS_8S_MASK 0x04U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.16 FS65_R_FS_DIS_8S_SHIFT

```
#define FS65_R_FS_DIS_8S_SHIFT 0x02U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.17 FS65_R_FS_FLT_ERR_FS_INT1_FIN2

```
#define FS65_R_FS_FLT_ERR_FS_INT1_FIN2 (0x01U << FS65_R_FS_FLT_ERR_FS_SHIFT)
```

intermediate = 1; final = 2

6.7.2.18 FS65_R_FS_FLT_ERR_FS_INT3_FIN6

```
#define FS65_R_FS_FLT_ERR_FS_INT3_FIN6 (0x00U << FS65_R_FS_FLT_ERR_FS_SHIFT)
```

intermediate = 3; final = 6

6.7.2.19 FS65_R_FS_FLT_ERR_FS_MASK

```
#define FS65_R_FS_FLT_ERR_FS_MASK 0x08U
```

Configure the values of the fault error counter

6.7.2.20 FS65_R_FS_FLT_ERR_FS_SHIFT

```
#define FS65_R_FS_FLT_ERR_FS_SHIFT 0x03U
```

Configure the values of the fault error counter

6.7.2.21 FS65_R_FS_FLT_ERR_IMP_FS0B

```
#define FS65_R_FS_FLT_ERR_IMP_FS0B (0x01U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
```

FS0B is asserted low if FLT_ERR_CNT >= intermediate value

6.7.2.22 FS65_R_FS_FLT_ERR_IMP_FS0B_RSTB

```
#define FS65_R_FS_FLT_ERR_IMP_FS0B_RSTB (0x03U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
```

FS0B is asserted low if FLT_ERR_CNT >= intermediate value RSTB is asserted low if FLT_ERR_CNT >= intermediate value and WD error counter = WD_CNT_ERR[1:0]

6.7.2.23 FS65_R_FS_FLT_ERR_IMP_MASK

```
#define FS65_R_FS_FLT_ERR_IMP_MASK 0x03U
```

Configure RSTB and FS0B behavior when fault error counter >= intermediate value

6.7.2.24 FS65_R_FS_FLT_ERR_IMP_NO_EFFECT

```
#define FS65_R_FS_FLT_ERR_IMP_NO_EFFECT (0x00U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
```

No effect on RSTB and FS0B

6.7.2.25 FS65_R_FS_FLT_ERR_IMP_RSTB

```
#define FS65_R_FS_FLT_ERR_IMP_RSTB (0x02U << FS65_R_FS_FLT_ERR_IMP_SHIFT)
```

RSTB is asserted low if FLT_ERR_CNT >= intermediate value and WD error counter = WD_CNT_ERR[1:0]

6.7.2.26 FS65_R_FS_FLT_ERR_IMP_SHIFT

```
#define FS65_R_FS_FLT_ERR_IMP_SHIFT 0x00U
```

Configure RSTB and FS0B behavior when fault error counter >= intermediate value

6.7.2.27 FS65_R_FS_FS0B_DRV_HIGH

```
#define FS65_R_FS_FS0B_DRV_HIGH (0x01U << FS65_R_FS_FS0B_DRV_SHIFT)
```

FS0B driver sense high

6.7.2.28 FS65_R_FS_FS0B_DRV_LOW

```
#define FS65_R_FS_FS0B_DRV_LOW (0x00U << FS65_R_FS_FS0B_DRV_SHIFT)
```

FS0B driver sense low

6.7.2.29 FS65_R_FS_FS0B_DRV_MASK

```
#define FS65_R_FS_FS0B_DRV_MASK 0x02U
```

Sense of FS0B driver command from fail-safe logic

6.7.2.30 FS65_R_FS_FS0B_DRV_SHIFT

```
#define FS65_R_FS_FS0B_DRV_SHIFT 0x01U
```

Sense of FS0B driver command from fail-safe logic

6.7.2.31 FS65_R_FS_FS0B_SNS_HIGH

```
#define FS65_R_FS_FS0B_SNS_HIGH (0x01U << FS65_R_FS_FS0B_SNS_SHIFT)
```

FS0B pad sense high

6.7.2.32 FS65_R_FS_FS0B_SNS_LOW

```
#define FS65_R_FS_FS0B_SNS_LOW (0x00U << FS65_R_FS_FS0B_SNS_SHIFT)
```

FS0B pad sense low

6.7.2.33 FS65_R_FS_FS0B_SNS_MASK

```
#define FS65_R_FS_FS0B_SNS_MASK 0x02U
```

Sense of FS0B pad

6.7.2.34 FS65_R_FS_FS0B_SNS_SHIFT

```
#define FS65_R_FS_FS0B_SNS_SHIFT 0x01U
```

Sense of FS0B pad

6.7.2.35 FS65_R_FS_FS1B_CAN_IMPACT_MASK

```
#define FS65_R_FS_FS1B_CAN_IMPACT_MASK 0x04U
```

Configure CAN behavior when FS1B is asserted low

6.7.2.36 FS65_R_FS_FS1B_CAN_IMPACT_NO_EFFECT

```
#define FS65_R_FS_FS1B_CAN_IMPACT_NO_EFFECT (0x00U << FS65_R_FS_FS1B_CAN_IMPACT_SHIFT)
```

No effect

6.7.2.37 FS65_R_FS_FS1B_CAN_IMPACT_RX_ONLY

```
#define FS65_R_FS_FS1B_CAN_IMPACT_RX_ONLY (0x01U << FS65_R_FS_FS1B_CAN_IMPACT_SHIFT)
```

CAN in Rx only or sleep mode when FS1B is asserted (depends on CAN_DIS_CFG bit in INIT_WU2 register)

6.7.2.38 FS65_R_FS_FS1B_CAN_IMPACT_SHIFT

```
#define FS65_R_FS_FS1B_CAN_IMPACT_SHIFT 0x02U
```

Configure CAN behavior when FS1B is asserted low

6.7.2.39 FS65_R_FS_FS1B_DLY_DRV_FS1B_HIGH

```
#define FS65_R_FS_FS1B_DLY_DRV_FS1B_HIGH (0x01U << FS65_R_FS_FS1B_DLY_DRV_SHIFT)
```

FS1B analog driver sense high

6.7.2.40 FS65_R_FS_FS1B_DLY_DRV_FS1B_LOW

```
#define FS65_R_FS_FS1B_DLY_DRV_FS1B_LOW (0x00U << FS65_R_FS_FS1B_DLY_DRV_SHIFT)
```

FS1B analog driver sense low

6.7.2.41 FS65_R_FS_FS1B_DLY_DRV_MASK

```
#define FS65_R_FS_FS1B_DLY_DRV_MASK 0x04U
```

Sense of FS1B driver command from backup delay (analog)

6.7.2.42 FS65_R_FS_FS1B_DLY_DRV_SHIFT

```
#define FS65_R_FS_FS1B_DLY_DRV_SHIFT 0x02U
```

Sense of FS1B driver command from backup delay (analog)

6.7.2.43 FS65_R_FS_FS1B_DRV_FS1B_HIGH

```
#define FS65_R_FS_FS1B_DRV_FS1B_HIGH (0x01U << FS65_R_FS_FS1B_DRV_SHIFT)
```

FS1B digital driver sense high

6.7.2.44 FS65_R_FS_FS1B_DRV_FS1B_LOW

```
#define FS65_R_FS_FS1B_DRV_FS1B_LOW (0x00U << FS65_R_FS_FS1B_DRV_SHIFT)
```

FS1B digital driver sense low

6.7.2.45 FS65_R_FS_FS1B_DRV_MASK

```
#define FS65_R_FS_FS1B_DRV_MASK 0x08U
```

Sense of FS1B driver command from fail-safe logic (digital)

6.7.2.46 FS65_R_FS_FS1B_DRV_SHIFT

```
#define FS65_R_FS_FS1B_DRV_SHIFT 0x03U
```

Sense of FS1B driver command from fail-safe logic (digital)

6.7.2.47 FS65_R_FS_FS1B_SNS_HIGH

```
#define FS65_R_FS_FS1B_SNS_HIGH (0x01U << FS65_R_FS_FS1B_SNS_SHIFT)
```

FS1B pad sense high

6.7.2.48 FS65_R_FS_FS1B_SNS_LOW

```
#define FS65_R_FS_FS1B_SNS_LOW (0x00U << FS65_R_FS_FS1B_SNS_SHIFT)
```

FS1B pad sense low

6.7.2.49 FS65_R_FS_FS1B_SNS_MASK

```
#define FS65_R_FS_FS1B_SNS_MASK 0x04U
```

Sense of FS1B pad

6.7.2.50 FS65_R_FS_FS1B_SNS_SHIFT

```
#define FS65_R_FS_FS1B_SNS_SHIFT 0x02U
```

Sense of FS1B pad

6.7.2.51 FS65_R_FS_FS1B_TIME_0_0

```
#define FS65_R_FS_FS1B_TIME_0_0 (0x00U << FS65_R_FS_FS1B_TIME_SHIFT)
```

0

6.7.2.52 FS65_R_FS_FS1B_TIME_106_848MS

```
#define FS65_R_FS_FS1B_TIME_106_848MS (0x0AU << FS65_R_FS_FS1B_TIME_SHIFT)
```

106 ms (FS1B_TIME_RANGE bit = 0) | 848 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.53 FS65_R_FS_FS1B_TIME_10MS_80MS

```
#define FS65_R_FS_FS1B_TIME_10MS_80MS (0x01U << FS65_R_FS_FS1B_TIME_SHIFT)
```

10 ms (FS1B_TIME_RANGE bit = 0) | 80 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.54 FS65_R_FS_FS1B_TIME_138_1103MS

```
#define FS65_R_FS_FS1B_TIME_138_1103MS (0x0BU << FS65_R_FS_FS1B_TIME_SHIFT)
```

138 ms (FS1B_TIME_RANGE bit = 0) | 1103 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.55 FS65_R_FS_FS1B_TIME_13_104MS

```
#define FS65_R_FS_FS1B_TIME_13_104MS (0x02U << FS65_R_FS_FS1B_TIME_SHIFT)
```

13 ms (FS1B_TIME_RANGE bit = 0) | 104 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.56 FS65_R_FS_FS1B_TIME_179_1434MS

```
#define FS65_R_FS_FS1B_TIME_179_1434MS (0x0CU << FS65_R_FS_FS1B_TIME_SHIFT)
```

179 ms (FS1B_TIME_RANGE bit = 0) | 1434 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.57 FS65_R_FS_FS1B_TIME_17_135MS

```
#define FS65_R_FS_FS1B_TIME_17_135MS (0x03U << FS65_R_FS_FS1B_TIME_SHIFT)
```

17 ms (FS1B_TIME_RANGE bit = 0) | 135 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.58 FS65_R_FS_FS1B_TIME_22_176MS

```
#define FS65_R_FS_FS1B_TIME_22_176MS (0x04U << FS65_R_FS_FS1B_TIME_SHIFT)
```

22 ms (FS1B_TIME_RANGE bit = 0) | 176 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.59 FS65_R_FS_FS1B_TIME_233_1864MS

```
#define FS65_R_FS_FS1B_TIME_233_1864MS (0x0DU << FS65_R_FS_FS1B_TIME_SHIFT)
```

233 ms (FS1B_TIME_RANGE bit = 0) | 1864 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.60 FS65_R_FS_FS1B_TIME_29_228MS

```
#define FS65_R_FS_FS1B_TIME_29_228MS (0x05U << FS65_R_FS_FS1B_TIME_SHIFT)
```

29 ms (FS1B_TIME_RANGE bit = 0) | 228 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.61 FS65_R_FS_FS1B_TIME_303_2423MS

```
#define FS65_R_FS_FS1B_TIME_303_2423MS (0x0EU << FS65_R_FS_FS1B_TIME_SHIFT)
```

303 ms (FS1B_TIME_RANGE bit = 0) | 2423 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.62 FS65_R_FS_FS1B_TIME_37_297MS

```
#define FS65_R_FS_FS1B_TIME_37_297MS (0x06U << FS65_R_FS_FS1B_TIME_SHIFT)
```

37 ms (FS1B_TIME_RANGE bit = 0) | 297 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.63 FS65_R_FS_FS1B_TIME_394_3150MS

```
#define FS65_R_FS_FS1B_TIME_394_3150MS (0x0FU << FS65_R_FS_FS1B_TIME_SHIFT)
```

394 ms (FS1B_TIME_RANGE bit = 0) | 3150 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.64 FS65_R_FS_FS1B_TIME_48_386MS

```
#define FS65_R_FS_FS1B_TIME_48_386MS (0x07U << FS65_R_FS_FS1B_TIME_SHIFT)
```

48 ms (FS1B_TIME_RANGE bit = 0) | 386 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.65 FS65_R_FS_FS1B_TIME_63_502MS

```
#define FS65_R_FS_FS1B_TIME_63_502MS (0x08U << FS65_R_FS_FS1B_TIME_SHIFT)
```

63 ms (FS1B_TIME_RANGE bit = 0) | 502 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.66 FS65_R_FS_FS1B_TIME_82_653MS

```
#define FS65_R_FS_FS1B_TIME_82_653MS (0x09U << FS65_R_FS_FS1B_TIME_SHIFT)
```

82 ms (FS1B_TIME_RANGE bit = 0) | 653 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.67 FS65_R_FS_FS1B_TIME_MASK

```
#define FS65_R_FS_FS1B_TIME_MASK 0x0FU
```

FS1B timing range factor x1(FS1B_TIME_RANGE bit = 0), FS1B timing range factor x8(FS1B_TIME_RANGE bit = 1)

6.7.2.68 FS65_R_FS_FS1B_TIME_RANGE_MASK

```
#define FS65_R_FS_FS1B_TIME_RANGE_MASK 0x01U
```

Configure the FS1B timing range factor x1 or x8

6.7.2.69 FS65_R_FS_FS1B_TIME_RANGE_SHIFT

```
#define FS65_R_FS_FS1B_TIME_RANGE_SHIFT 0x00U
```

Configure the FS1B timing range factor x1 or x8

6.7.2.70 FS65_R_FS_FS1B_TIME_RANGE_X1

```
#define FS65_R_FS_FS1B_TIME_RANGE_X1 (0x00U << FS65_R_FS_FS1B_TIME_RANGE_SHIFT)
```

x1 timing range factor

6.7.2.71 FS65_R_FS_FS1B_TIME_RANGE_X8

```
#define FS65_R_FS_FS1B_TIME_RANGE_X8 (0x01U << FS65_R_FS_FS1B_TIME_RANGE_SHIFT)
```

x8 timing range factor

6.7.2.72 FS65_R_FS_FS1B_TIME_SHIFT

```
#define FS65_R_FS_FS1B_TIME_SHIFT 0x00U
```

FS1B timing range factor x1(FS1B_TIME_RANGE bit = 0), FS1B timing range factor x8(FS1B_TIME_RANGE bit = 1)

6.7.2.73 FS65_R_FS_IO_23_FS_MASK

```
#define FS65_R_FS_IO_23_FS_MASK 0x04U
```

Configure the couple of IO_3:2 as safety inputs for FCCU monitoring

6.7.2.74 FS65_R_FS_IO_23_FS_NOT_SAFETY

```
#define FS65_R_FS_IO_23_FS_NOT_SAFETY (0x00U << FS65_R_FS_IO_23_FS_SHIFT)
```

Not_safety

6.7.2.75 FS65_R_FS_IO_23_FS_SAFETY_CRITICAL

```
#define FS65_R_FS_IO_23_FS_SAFETY_CRITICAL (0x01U << FS65_R_FS_IO_23_FS_SHIFT)
```

Safety_critical

6.7.2.76 FS65_R_FS_IO_23_FS_SHIFT

```
#define FS65_R_FS_IO_23_FS_SHIFT 0x02U
```

Configure the couple of IO_3:2 as safety inputs for FCCU monitoring

6.7.2.77 FS65_R_FS_IO_45_FS_MASK

```
#define FS65_R_FS_IO_45_FS_MASK 0x08U
```

Configure the couple of IO_4:5 as safety inputs for external IC error monitoring

6.7.2.78 FS65_R_FS_IO_45_FS_NOT_SAFETY

```
#define FS65_R_FS_IO_45_FS_NOT_SAFETY (0x00U << FS65_R_FS_IO_45_FS_SHIFT)
```

Not safety

6.7.2.79 FS65_R_FS_IO_45_FS_SAFETY_CRITICAL

```
#define FS65_R_FS_IO_45_FS_SAFETY_CRITICAL (0x01U << FS65_R_FS_IO_45_FS_SHIFT)
```

Safety critical

6.7.2.80 FS65_R_FS_IO_45_FS_SHIFT

```
#define FS65_R_FS_IO_45_FS_SHIFT 0x03U
```

Configure the couple of IO_4:5 as safety inputs for external IC error monitoring

6.7.2.81 FS65_R_FS_LBIST_OK_FAIL

```
#define FS65_R_FS_LBIST_OK_FAIL (0x00U << FS65_R_FS_LBIST_OK_SHIFT)
```

LBIST fail

6.7.2.82 FS65_R_FS_LBIST_OK_MASK

```
#define FS65_R_FS_LBIST_OK_MASK 0x08U
```

Diagnostic of fail-safe logic BIST (automatically executed)

6.7.2.83 FS65_R_FS_LBIST_OK_PASS

```
#define FS65_R_FS_LBIST_OK_PASS (0x01U << FS65_R_FS_LBIST_OK_SHIFT)
```

LBIST pass

6.7.2.84 FS65_R_FS_LBIST_OK_SHIFT

```
#define FS65_R_FS_LBIST_OK_SHIFT 0x03U
```

Diagnostic of fail-safe logic BIST (automatically executed)

6.7.2.85 FS65_R_FS_PS_HIGH

```
#define FS65_R_FS_PS_HIGH (0x00U << FS65_R_FS_PS_SHIFT)
```

Fccu_eaout_1:0 active high

6.7.2.86 FS65_R_FS_PS_LOW

```
#define FS65_R_FS_PS_LOW (0x01U << FS65_R_FS_PS_SHIFT)
```

Fccu_eaout_1:0 active low

6.7.2.87 FS65_R_FS_PS_MASK

```
#define FS65_R_FS_PS_MASK 0x02U
```

Configure the FCCU polarity

6.7.2.88 FS65_R_FS_PS_SHIFT

```
#define FS65_R_FS_PS_SHIFT 0x01U
```

Configure the FCCU polarity

6.7.2.89 FS65_R_FS_RSTB_DRV_HIGH

```
#define FS65_R_FS_RSTB_DRV_HIGH (0x01U << FS65_R_FS_RSTB_DRV_SHIFT)
```

RSTB driver sense high

6.7.2.90 FS65_R_FS_RSTB_DRV_LOW

```
#define FS65_R_FS_RSTB_DRV_LOW (0x00U << FS65_R_FS_RSTB_DRV_SHIFT)
```

RSTB driver sense low

6.7.2.91 FS65_R_FS_RSTB_DRV_MASK

```
#define FS65_R_FS_RSTB_DRV_MASK 0x01U
```

Sense of RSTB driver command from fail-safe logic

6.7.2.92 FS65_R_FS_RSTB_DRV_SHIFT

```
#define FS65_R_FS_RSTB_DRV_SHIFT 0x00U
```

Sense of RSTB driver command from fail-safe logic

6.7.2.93 FS65_R_FS_RSTB_DURATION_10MS

```
#define FS65_R_FS_RSTB_DURATION_10MS (0x00U << FS65_R_FS_RSTB_DURATION_SHIFT)
```

10 ms

6.7.2.94 FS65_R_FS_RSTB_DURATION_1MS

```
#define FS65_R_FS_RSTB_DURATION_1MS (0x01U << FS65_R_FS_RSTB_DURATION_SHIFT)
```

1.0 ms

6.7.2.95 FS65_R_FS_RSTB_DURATION_MASK

```
#define FS65_R_FS_RSTB_DURATION_MASK 0x01U
```

Configure the RSTB low duration time

6.7.2.96 FS65_R_FS_RSTB_DURATION_SHIFT

```
#define FS65_R_FS_RSTB_DURATION_SHIFT 0x00U
```

Configure the RSTB low duration time

6.7.2.97 FS65_R_FS_RSTB_SNS_HIGH

```
#define FS65_R_FS_RSTB_SNS_HIGH (0x01U << FS65_R_FS_RSTB_SNS_SHIFT)
```

RSTB pad sense high

6.7.2.98 FS65_R_FS_RSTB_SNS_LOW

```
#define FS65_R_FS_RSTB_SNS_LOW (0x00U << FS65_R_FS_RSTB_SNS_SHIFT)
```

RSTB pad sense low

6.7.2.99 FS65_R_FS_RSTB_SNS_MASK

```
#define FS65_R_FS_RSTB_SNS_MASK 0x01U
```

Sense of RSTB pad

6.7.2.100 FS65_R_FS_RSTB_SNS_SHIFT

```
#define FS65_R_FS_RSTB_SNS_SHIFT 0x00U
```

Sense of RSTB pad

6.7.2.101 FS65_R_FS_TDLY_TDUR_DELAY

```
#define FS65_R_FS_TDLY_TDUR_DELAY (0x00U << FS65_R_FS_TDLY_TDUR_SHIFT)
```

FS1B tDELAY mode

6.7.2.102 FS65_R_FS_TDLY_TDUR_DURATION

```
#define FS65_R_FS_TDLY_TDUR_DURATION (0x01U << FS65_R_FS_TDLY_TDUR_SHIFT)
```

FS1B tDURATION mode

6.7.2.103 FS65_R_FS_TDLY_TDUR_MASK

```
#define FS65_R_FS_TDLY_TDUR_MASK 0x08U
```

FS1B delay or FS1B duration mode selection

6.7.2.104 FS65_R_FS_TDLY_TDUR_SHIFT

```
#define FS65_R_FS_TDLY_TDUR_SHIFT 0x03U
```

FS1B delay or FS1B duration mode selection

6.7.2.105 FS65_R_FS_VAUX_5D_DEGRADED

```
#define FS65_R_FS_VAUX_5D_DEGRADED (0x01U << FS65_R_FS_VAUX_5D_SHIFT)
```

Degraded mode; lower undervoltage detection threshold applied (VAUX_UV_5D)

6.7.2.106 FS65_R_FS_VAUX_5D_MASK

```
#define FS65_R_FS_VAUX_5D_MASK 0x02U
```

Configure the VAUX undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.107 FS65_R_FS_VAUX_5D_NORMAL

```
#define FS65_R_FS_VAUX_5D_NORMAL (0x00U << FS65_R_FS_VAUX_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VAUX_UV_5)

6.7.2.108 FS65_R_FS_VAUX_5D_SHIFT

```
#define FS65_R_FS_VAUX_5D_SHIFT 0x01U
```

Configure the VAUX undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.109 FS65_R_FS_VAUX_FS_OV_FS0B

```
#define FS65_R_FS_VAUX_FS_OV_FS0B (0x02U << FS65_R_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on FS0B only

6.7.2.110 FS65_R_FS_VAUX_FS_OV_MASK

```
#define FS65_R_FS_VAUX_FS_OV_MASK 0x0CU
```

VAUX overvoltage safety impact

6.7.2.111 FS65_R_FS_VAUX_FS_OV_NO_EFFECT

```
#define FS65_R_FS_VAUX_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VAUX_FS_OV_SHIFT)
```

No effect of VAUX_OV on RSTB and FS0B

6.7.2.112 FS65_R_FS_VAUX_FS_OV_RSTB

```
#define FS65_R_FS_VAUX_FS_OV_RSTB (0x01U << FS65_R_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on RSTB only

6.7.2.113 FS65_R_FS_VAUX_FS_OV_RSTB_FS0B

```
#define FS65_R_FS_VAUX_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on RSTB and FS0B

6.7.2.114 FS65_R_FS_VAUX_FS_OV_SHIFT

```
#define FS65_R_FS_VAUX_FS_OV_SHIFT 0x02U
```

VAUX overvoltage safety impact

6.7.2.115 FS65_R_FS_VAUX_FS_UV_FS0B

```
#define FS65_R_FS_VAUX_FS_UV_FS0B (0x02U << FS65_R_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on FS0B only

6.7.2.116 FS65_R_FS_VAUX_FS_UV_MASK

```
#define FS65_R_FS_VAUX_FS_UV_MASK 0x03U
```

VAUX undervoltage safety impact

6.7.2.117 FS65_R_FS_VAUX_FS_UV_NO_EFFECT

```
#define FS65_R_FS_VAUX_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VAUX_FS_UV_SHIFT)
```

No effect of VAUX_UV on RSTB and FS0B

6.7.2.118 FS65_R_FS_VAUX_FS_UV_RSTB

```
#define FS65_R_FS_VAUX_FS_UV_RSTB (0x01U << FS65_R_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on RSTB only

6.7.2.119 FS65_R_FS_VAUX_FS_UV_RSTB_FS0B

```
#define FS65_R_FS_VAUX_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on RSTB and FS0B

6.7.2.120 FS65_R_FS_VAUX_FS_UV_SHIFT

```
#define FS65_R_FS_VAUX_FS_UV_SHIFT 0x00U
```

VAUX undervoltage safety impact

6.7.2.121 FS65_R_FS_VCCA_5D_DEGRADED

```
#define FS65_R_FS_VCCA_5D_DEGRADED (0x01U << FS65_R_FS_VCCA_5D_SHIFT)
```

Degraded mode, lower undervoltage detection threshold applied (VCCA_UV_D)

6.7.2.122 FS65_R_FS_VCCA_5D_MASK

```
#define FS65_R_FS_VCCA_5D_MASK 0x04U
```

Configure the VCCA undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.123 FS65_R_FS_VCCA_5D_NORMAL

```
#define FS65_R_FS_VCCA_5D_NORMAL (0x00U << FS65_R_FS_VCCA_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VCCA_UV_5)

6.7.2.124 FS65_R_FS_VCCA_5D_SHIFT

```
#define FS65_R_FS_VCCA_5D_SHIFT 0x02U
```

Configure the VCCA undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.125 FS65_R_FS_VCCA_FS_OV_FS0B

```
#define FS65_R_FS_VCCA_FS_OV_FS0B (0x02U << FS65_R_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on FS0B only

6.7.2.126 FS65_R_FS_VCCA_FS_OV_MASK

```
#define FS65_R_FS_VCCA_FS_OV_MASK 0x0CU
```

VCCA overvoltage safety impact

6.7.2.127 FS65_R_FS_VCCA_FS_OV_NO_EFFECT

```
#define FS65_R_FS_VCCA_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VCCA_FS_OV_SHIFT)
```

No effect of VCCA_OV on RSTB and FS0B

6.7.2.128 FS65_R_FS_VCCA_FS_OV_RSTB

```
#define FS65_R_FS_VCCA_FS_OV_RSTB (0x01U << FS65_R_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on RSTB only

6.7.2.129 FS65_R_FS_VCCA_FS_OV_RSTB_FS0B

```
#define FS65_R_FS_VCCA_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on RSTB and FS0B

6.7.2.130 FS65_R_FS_VCCA_FS_OV_SHIFT

```
#define FS65_R_FS_VCCA_FS_OV_SHIFT 0x02U
```

VCCA overvoltage safety impact

6.7.2.131 FS65_R_FS_VCCA_FS_UV_FS0B

```
#define FS65_R_FS_VCCA_FS_UV_FS0B (0x02U << FS65_R_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on FS0B only

6.7.2.132 FS65_R_FS_VCCA_FS_UV_MASK

```
#define FS65_R_FS_VCCA_FS_UV_MASK 0x03U
```

VCCA undervoltage safety impact

6.7.2.133 FS65_R_FS_VCCA_FS_UV_NO_EFFECT

```
#define FS65_R_FS_VCCA_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VCCA_FS_UV_SHIFT)
```

No effect of VCCA_UV on RSTB and FS0B

6.7.2.134 FS65_R_FS_VCCA_FS_UV_RSTB

```
#define FS65_R_FS_VCCA_FS_UV_RSTB (0x01U << FS65_R_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on RSTB only

6.7.2.135 FS65_R_FS_VCCA_FS_UV_RSTB_FS0B

```
#define FS65_R_FS_VCCA_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on RSTB and FS0B

6.7.2.136 FS65_R_FS_VCCA_FS_UV_SHIFT

```
#define FS65_R_FS_VCCA_FS_UV_SHIFT 0x00U
```

VCCA undervoltage safety impact

6.7.2.137 FS65_R_FS_VCORE_5D_DEGRADED

```
#define FS65_R_FS_VCORE_5D_DEGRADED (0x01U << FS65_R_FS_VCORE_5D_SHIFT)
```

Degraded mode, lower undervoltage detection threshold applied (VCORE_FB_UV_D)

6.7.2.138 FS65_R_FS_VCORE_5D_MASK

```
#define FS65_R_FS_VCORE_5D_MASK 0x08U
```

Configure the VCORE undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.139 FS65_R_FS_VCORE_5D_NORMAL

```
#define FS65_R_FS_VCORE_5D_NORMAL (0x00U << FS65_R_FS_VCORE_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VCORE_FB_UV)

6.7.2.140 FS65_R_FS_VCORE_5D_SHIFT

```
#define FS65_R_FS_VCORE_5D_SHIFT 0x03U
```

Configure the VCORE undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.141 FS65_R_FS_VCORE_FS_OV_FS0B

```
#define FS65_R_FS_VCORE_FS_OV_FS0B (0x02U << FS65_R_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on FS0B only

6.7.2.142 FS65_R_FS_VCORE_FS_OV_MASK

```
#define FS65_R_FS_VCORE_FS_OV_MASK 0x0CU
```

VCORE_FB overvoltage safety impact

6.7.2.143 FS65_R_FS_VCORE_FS_OV_NO_EFFECT

```
#define FS65_R_FS_VCORE_FS_OV_NO_EFFECT (0x00U << FS65_R_FS_VCORE_FS_OV_SHIFT)
```

No effect of VCORE_FB_OV on RSTB and FS0B

6.7.2.144 FS65_R_FS_VCORE_FS_OV_RSTB

```
#define FS65_R_FS_VCORE_FS_OV_RSTB (0x01U << FS65_R_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on RSTB only

6.7.2.145 FS65_R_FS_VCORE_FS_OV_RSTB_FS0B

```
#define FS65_R_FS_VCORE_FS_OV_RSTB_FS0B (0x03U << FS65_R_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on RSTB and FS0B

6.7.2.146 FS65_R_FS_VCORE_FS_OV_SHIFT

```
#define FS65_R_FS_VCORE_FS_OV_SHIFT 0x02U
```

VCORE_FB overvoltage safety impact

6.7.2.147 FS65_R_FS_VCORE_FS_UV_FS0B

```
#define FS65_R_FS_VCORE_FS_UV_FS0B (0x02U << FS65_R_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on FS0B only

6.7.2.148 FS65_R_FS_VCORE_FS_UV_MASK

```
#define FS65_R_FS_VCORE_FS_UV_MASK 0x03U
```

VCORE_FB undervoltage safety impact

6.7.2.149 FS65_R_FS_VCORE_FS_UV_NO_EFFECT

```
#define FS65_R_FS_VCORE_FS_UV_NO_EFFECT (0x00U << FS65_R_FS_VCORE_FS_UV_SHIFT)
```

No effect of VCORE_FB_UV on RSTB and FS0B

6.7.2.150 FS65_R_FS_VCORE_FS_UV_RSTB

```
#define FS65_R_FS_VCORE_FS_UV_RSTB (0x01U << FS65_R_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on RSTB only

6.7.2.151 FS65_R_FS_VCORE_FS_UV_RSTB_FS0B

```
#define FS65_R_FS_VCORE_FS_UV_RSTB_FS0B (0x03U << FS65_R_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on RSTB and FS0B

6.7.2.152 FS65_R_FS_VCORE_FS_UV_SHIFT

```
#define FS65_R_FS_VCORE_FS_UV_SHIFT 0x00U
```

VCORE_FB undervoltage safety impact

6.7.2.153 FS65_R_FS_WD_CNT_ERR_2

```
#define FS65_R_FS_WD_CNT_ERR_2 (0x03U << FS65_R_FS_WD_CNT_ERR_SHIFT)
```

2

6.7.2.154 FS65_R_FS_WD_CNT_ERR_4

```
#define FS65_R_FS_WD_CNT_ERR_4 (0x02U << FS65_R_FS_WD_CNT_ERR_SHIFT)
```

4

6.7.2.155 FS65_R_FS_WD_CNT_ERR_6

```
#define FS65_R_FS_WD_CNT_ERR_6 (0x00U << FS65_R_FS_WD_CNT_ERR_SHIFT)
```

6

6.7.2.156 FS65_R_FS_WD_CNT_ERR_MASK

```
#define FS65_R_FS_WD_CNT_ERR_MASK 0x0CU
```

Configure the maximum value of the WD error counter

6.7.2.157 FS65_R_FS_WD_CNT_ERR_SHIFT

```
#define FS65_R_FS_WD_CNT_ERR_SHIFT 0x02U
```

Configure the maximum value of the WD error counter

6.7.2.158 FS65_R_FS_WD_CNT_RFR_1

```
#define FS65_R_FS_WD_CNT_RFR_1 (0x03U << FS65_R_FS_WD_CNT_RFR_SHIFT)
```

1

6.7.2.159 FS65_R_FS_WD_CNT_RFR_2

```
#define FS65_R_FS_WD_CNT_RFR_2 (0x02U << FS65_R_FS_WD_CNT_RFR_SHIFT)
```

2

6.7.2.160 FS65_R_FS_WD_CNT_RFR_4

```
#define FS65_R_FS_WD_CNT_RFR_4 (0x01U << FS65_R_FS_WD_CNT_RFR_SHIFT)
```

4

6.7.2.161 FS65_R_FS_WD_CNT_RFR_6

```
#define FS65_R_FS_WD_CNT_RFR_6 (0x00U << FS65_R_FS_WD_CNT_RFR_SHIFT)
```

6

6.7.2.162 FS65_R_FS_WD_CNT_RFR_MASK

```
#define FS65_R_FS_WD_CNT_RFR_MASK 0x03U
```

Configure the maximum value of the WD refresh counter

6.7.2.163 FS65_R_FS_WD_CNT_RFR_SHIFT

```
#define FS65_R_FS_WD_CNT_RFR_SHIFT 0x00U
```

Configure the maximum value of the WD refresh counter

6.7.2.164 FS65_R_FS_WD_IMPACT_FS0B

```
#define FS65_R_FS_WD_IMPACT_FS0B (0x02U << FS65_R_FS_WD_IMPACT_SHIFT)
```

FS0B only is asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.165 FS65_R_FS_WD_IMPACT_MASK

```
#define FS65_R_FS_WD_IMPACT_MASK 0x03U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.166 FS65_R_FS_WD_IMPACT_NO_EFFECT

```
#define FS65_R_FS_WD_IMPACT_NO_EFFECT (0x00U << FS65_R_FS_WD_IMPACT_SHIFT)
```

No effect on RSTB and FS0B if WD error counter = WD_CNT_ERR[1:0]

6.7.2.167 FS65_R_FS_WD_IMPACT_RSTB

```
#define FS65_R_FS_WD_IMPACT_RSTB (0x01U << FS65_R_FS_WD_IMPACT_SHIFT)
```

RSTB only is asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.168 FS65_R_FS_WD_IMPACT_RSTB_FS0B

```
#define FS65_R_FS_WD_IMPACT_RSTB_FS0B (0x03U << FS65_R_FS_WD_IMPACT_SHIFT)
```

RSTB and FS0B are asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.169 FS65_R_FS_WD_IMPACT_SHIFT

```
#define FS65_R_FS_WD_IMPACT_SHIFT 0x00U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.170 FS65_R_FS_WD_WINDOW_1024MS

```
#define FS65_R_FS_WD_WINDOW_1024MS (0x0FU << FS65_R_FS_WD_WINDOW_SHIFT)
```

1024 ms

6.7.2.171 FS65_R_FS_WD_WINDOW_128MS

```
#define FS65_R_FS_WD_WINDOW_128MS (0x0CU << FS65_R_FS_WD_WINDOW_SHIFT)
```

128 ms

6.7.2.172 FS65_R_FS_WD_WINDOW_12MS

```
#define FS65_R_FS_WD_WINDOW_12MS (0x07U << FS65_R_FS_WD_WINDOW_SHIFT)
```

12.0 ms

6.7.2.173 FS65_R_FS_WD_WINDOW_16MS

```
#define FS65_R_FS_WD_WINDOW_16MS (0x08U << FS65_R_FS_WD_WINDOW_SHIFT)
```

16 ms

6.7.2.174 FS65_R_FS_WD_WINDOW_1MS

```
#define FS65_R_FS_WD_WINDOW_1MS (0x01U << FS65_R_FS_WD_WINDOW_SHIFT)
```

1.0 ms

6.7.2.175 FS65_R_FS_WD_WINDOW_24MS

```
#define FS65_R_FS_WD_WINDOW_24MS (0x09U << FS65_R_FS_WD_WINDOW_SHIFT)
```

24 ms

6.7.2.176 FS65_R_FS_WD_WINDOW_256MS

```
#define FS65_R_FS_WD_WINDOW_256MS (0x0DU << FS65_R_FS_WD_WINDOW_SHIFT)
```

256 ms

6.7.2.177 FS65_R_FS_WD_WINDOW_2MS

```
#define FS65_R_FS_WD_WINDOW_2MS (0x02U << FS65_R_FS_WD_WINDOW_SHIFT)
```

2.0 ms

6.7.2.178 FS65_R_FS_WD_WINDOW_32MS

```
#define FS65_R_FS_WD_WINDOW_32MS (0x0AU << FS65_R_FS_WD_WINDOW_SHIFT)
```

32 ms

6.7.2.179 FS65_R_FS_WD_WINDOW_3MS

```
#define FS65_R_FS_WD_WINDOW_3MS (0x03U << FS65_R_FS_WD_WINDOW_SHIFT)
```

3.0 ms

6.7.2.180 FS65_R_FS_WD_WINDOW_4MS

```
#define FS65_R_FS_WD_WINDOW_4MS (0x04U << FS65_R_FS_WD_WINDOW_SHIFT)
```

4.0 ms

6.7.2.181 FS65_R_FS_WD_WINDOW_512MS

```
#define FS65_R_FS_WD_WINDOW_512MS (0x0EU << FS65_R_FS_WD_WINDOW_SHIFT)
```

512 ms

6.7.2.182 FS65_R_FS_WD_WINDOW_64MS

```
#define FS65_R_FS_WD_WINDOW_64MS (0x0BU << FS65_R_FS_WD_WINDOW_SHIFT)
```

64 ms

6.7.2.183 FS65_R_FS_WD_WINDOW_6MS

```
#define FS65_R_FS_WD_WINDOW_6MS (0x05U << FS65_R_FS_WD_WINDOW_SHIFT)
```

6.0 ms

6.7.2.184 FS65_R_FS_WD_WINDOW_8MS

```
#define FS65_R_FS_WD_WINDOW_8MS (0x06U << FS65_R_FS_WD_WINDOW_SHIFT)
```

8.0 ms

6.7.2.185 FS65_R_FS_WD_WINDOW_DISABLE

```
#define FS65_R_FS_WD_WINDOW_DISABLE (0x00U << FS65_R_FS_WD_WINDOW_SHIFT)
```

Disable (in INIT phase only)

6.7.2.186 FS65_R_FS_WD_WINDOW_MASK

```
#define FS65_R_FS_WD_WINDOW_MASK 0x0FU
```

Configure the watchdog window duration. Duty cycle if set to 50 %

6.7.2.187 FS65_R_FS_WD_WINDOW_SHIFT

```
#define FS65_R_FS_WD_WINDOW_SHIFT 0x00U
```

Configure the watchdog window duration. Duty cycle if set to 50 %

6.7.2.188 FS65_R_M_AUTO_WU_EVENT

```
#define FS65_R_M_AUTO_WU_EVENT (0x01U << FS65_R_M_AUTO_WU_SHIFT)
```

Wake-up event detected

6.7.2.189 FS65_R_M_AUTO_WU_MASK

```
#define FS65_R_M_AUTO_WU_MASK 0x04U
```

Report an automatic wake-up event

6.7.2.190 FS65_R_M_AUTO_WU_NO_EVENT

```
#define FS65_R_M_AUTO_WU_NO_EVENT (0x00U << FS65_R_M_AUTO_WU_SHIFT)
```

No wake-up

6.7.2.191 FS65_R_M_AUTO_WU_SHIFT

```
#define FS65_R_M_AUTO_WU_SHIFT 0x02U
```

Report an automatic wake-up event

6.7.2.192 FS65_R_M_BAT_FAIL_MASK

```
#define FS65_R_M_BAT_FAIL_MASK 0x01U
```

Report a battery disconnection (POR of the main logic)

6.7.2.193 FS65_R_M_BAT_FAIL_NO_POR

```
#define FS65_R_M_BAT_FAIL_NO_POR (0x00U << FS65_R_M_BAT_FAIL_SHIFT)
```

NO POR

6.7.2.194 FS65_R_M_BAT_FAIL_POR

```
#define FS65_R_M_BAT_FAIL_POR (0x01U << FS65_R_M_BAT_FAIL_SHIFT)
```

POR occurred

6.7.2.195 FS65_R_M_BAT_FAIL_SHIFT

```
#define FS65_R_M_BAT_FAIL_SHIFT 0x00U
```

Report a battery disconnection (POR of the main logic)

6.7.2.196 FS65_R_M_BOB_BOOST

```
#define FS65_R_M_BOB_BOOST (0x01U << FS65_R_M_BOB_SHIFT)
```

Boost

6.7.2.197 FS65_R_M_BOB_BUCK

```
#define FS65_R_M_BOB_BUCK (0x00U << FS65_R_M_BOB_SHIFT)
```

Buck

6.7.2.198 FS65_R_M_BOB_MASK

```
#define FS65_R_M_BOB_MASK 0x80U
```

Report a running mode of VPRE

6.7.2.199 FS65_R_M_BOB_SHIFT

```
#define FS65_R_M_BOB_SHIFT 0x07U
```

Report a running mode of VPRE

6.7.2.200 FS65_R_M_CAN_DOM_FAILURE

```
#define FS65_R_M_CAN_DOM_FAILURE (0x01U << FS65_R_M_CAN_DOM_SHIFT)
```

Failure detected

6.7.2.201 FS65_R_M_CAN_DOM_MASK

```
#define FS65_R_M_CAN_DOM_MASK 0x08U
```

CAN Bus dominant clamping detection

6.7.2.202 FS65_R_M_CAN_DOM_NO_FAILURE

```
#define FS65_R_M_CAN_DOM_NO_FAILURE (0x00U << FS65_R_M_CAN_DOM_SHIFT)
```

No failure

6.7.2.203 FS65_R_M_CAN_DOM_SHIFT

```
#define FS65_R_M_CAN_DOM_SHIFT 0x03U
```

CAN Bus dominant clamping detection

6.7.2.204 FS65_R_M_CAN_OC_FAILURE

```
#define FS65_R_M_CAN_OC_FAILURE (0x01U << FS65_R_M_CAN_OC_SHIFT)
```

Failure detected

6.7.2.205 FS65_R_M_CAN_OC_MASK

```
#define FS65_R_M_CAN_OC_MASK 0x01U
```

CAN overcurrent detection

6.7.2.206 FS65_R_M_CAN_OC_NO_FAILURE

```
#define FS65_R_M_CAN_OC_NO_FAILURE (0x00U << FS65_R_M_CAN_OC_SHIFT)
```

No failure

6.7.2.207 FS65_R_M_CAN_OC_SHIFT

```
#define FS65_R_M_CAN_OC_SHIFT 0x00U
```

CAN overcurrent detection

6.7.2.208 FS65_R_M_CAN_OT_FAILURE

```
#define FS65_R_M_CAN_OT_FAILURE (0x01U << FS65_R_M_CAN_OT_SHIFT)
```

Failure detected

6.7.2.209 FS65_R_M_CAN_OT_MASK

```
#define FS65_R_M_CAN_OT_MASK 0x02U
```

CAN overtemperature detection

6.7.2.210 FS65_R_M_CAN_OT_NO_FAILURE

```
#define FS65_R_M_CAN_OT_NO_FAILURE (0x00U << FS65_R_M_CAN_OT_SHIFT)
```

No failure

6.7.2.211 FS65_R_M_CAN_OT_SHIFT

```
#define FS65_R_M_CAN_OT_SHIFT 0x01U
```

CAN overtemperature detection

6.7.2.212 FS65_R_M_CAN_WU_MASK

```
#define FS65_R_M_CAN_WU_MASK 0x02U
```

Report a wake-up event from the CAN

6.7.2.213 FS65_R_M_CAN_WU_NO_WU

```
#define FS65_R_M_CAN_WU_NO_WU (0x00U << FS65_R_M_CAN_WU_SHIFT)
```

No wake-up

6.7.2.214 FS65_R_M_CAN_WU_SHIFT

```
#define FS65_R_M_CAN_WU_SHIFT 0x01U
```

Report a wake-up event from the CAN

6.7.2.215 FS65_R_M_CAN_WU_WU

```
#define FS65_R_M_CAN_WU_WU (0x01U << FS65_R_M_CAN_WU_SHIFT)
```

Wake-up detected

6.7.2.216 FS65_R_M_CANH_BATT_FAILURE

```
#define FS65_R_M_CANH_BATT_FAILURE (0x01U << FS65_R_M_CANH_BATT_SHIFT)
```

Failure detected

6.7.2.217 FS65_R_M_CANH_BATT_MASK

```
#define FS65_R_M_CANH_BATT_MASK 0x80U
```

CANH short-circuit to battery detection

6.7.2.218 FS65_R_M_CANH_BATT_NO_FAILURE

```
#define FS65_R_M_CANH_BATT_NO_FAILURE (0x00U << FS65_R_M_CANH_BATT_SHIFT)
```

No failure

6.7.2.219 FS65_R_M_CANH_BATT_SHIFT

```
#define FS65_R_M_CANH_BATT_SHIFT 0x07U
```

CANH short-circuit to battery detection

6.7.2.220 FS65_R_M_CANH_GND_FAILURE

```
#define FS65_R_M_CANH_GND_FAILURE (0x01U << FS65_R_M_CANH_GND_SHIFT)
```

Failure detected

6.7.2.221 FS65_R_M_CANH_GND_MASK

```
#define FS65_R_M_CANH_GND_MASK 0x40U
```

CANH short-circuit to GND detection

6.7.2.222 FS65_R_M_CANH_GND_NO_FAILURE

```
#define FS65_R_M_CANH_GND_NO_FAILURE (0x00U << FS65_R_M_CANH_GND_SHIFT)
```

No failure

6.7.2.223 FS65_R_M_CANH_GND_SHIFT

```
#define FS65_R_M_CANH_GND_SHIFT 0x06U
```

CANH short-circuit to GND detection

6.7.2.224 FS65_R_M_CANL_BATT_FAILURE

```
#define FS65_R_M_CANL_BATT_FAILURE (0x01U << FS65_R_M_CANL_BATT_SHIFT)
```

Failure detected

6.7.2.225 FS65_R_M_CANL_BATT_MASK

```
#define FS65_R_M_CANL_BATT_MASK 0x20U
```

CANL short-circuit to battery detection

6.7.2.226 FS65_R_M_CANL_BATT_NO_FAILURE

```
#define FS65_R_M_CANL_BATT_NO_FAILURE (0x00U << FS65_R_M_CANL_BATT_SHIFT)
```

No failure

6.7.2.227 FS65_R_M_CANL_BATT_SHIFT

```
#define FS65_R_M_CANL_BATT_SHIFT 0x05U
```

CANL short-circuit to battery detection

6.7.2.228 FS65_R_M_CANL_GND_FAILURE

```
#define FS65_R_M_CANL_GND_FAILURE (0x01U << FS65_R_M_CANL_GND_SHIFT)
```

Failure detected

6.7.2.229 FS65_R_M_CANL_GND_MASK

```
#define FS65_R_M_CANL_GND_MASK 0x10U
```

CANL short-circuit to GND detection

6.7.2.230 FS65_R_M_CANL_GND_NO_FAILURE

```
#define FS65_R_M_CANL_GND_NO_FAILURE (0x00U << FS65_R_M_CANL_GND_SHIFT)
```

No failure

6.7.2.231 FS65_R_M_CANL_GND_SHIFT

```
#define FS65_R_M_CANL_GND_SHIFT 0x04U
```

CANL short-circuit to GND detection

6.7.2.232 FS65_R_M_DBG_HW_DEBUG

```
#define FS65_R_M_DBG_HW_DEBUG (0x01U << FS65_R_M_DBG_HW_SHIFT)
```

Debug mode selected

6.7.2.233 FS65_R_M_DBG_HW_MASK

```
#define FS65_R_M_DBG_HW_MASK 0x01U
```

Report the configuration of the DEBUG mode

6.7.2.234 FS65_R_M_DBG_HW_NORMAL

```
#define FS65_R_M_DBG_HW_NORMAL (0x00U << FS65_R_M_DBG_HW_SHIFT)
```

Normal operation

6.7.2.235 FS65_R_M_DBG_HW_SHIFT

```
#define FS65_R_M_DBG_HW_SHIFT 0x00U
```

Report the configuration of the DEBUG mode

6.7.2.236 FS65_R_M_DEV_REV_MASK

```
#define FS65_R_M_DEV_REV_MASK 0x07U
```

Device silicon revision

6.7.2.237 FS65_R_M_DEV_REV_REV_000

```
#define FS65_R_M_DEV_REV_REV_000 (0x00U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 000

6.7.2.238 FS65_R_M_DEV_REV_REV_001

```
#define FS65_R_M_DEV_REV_REV_001 (0x01U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 001

6.7.2.239 FS65_R_M_DEV_REV_REV_010

```
#define FS65_R_M_DEV_REV_REV_010 (0x02U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 010

6.7.2.240 FS65_R_M_DEV_REV_REV_011

```
#define FS65_R_M_DEV_REV_REV_011 (0x03U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 011

6.7.2.241 FS65_R_M_DEV_REV_REV_100

```
#define FS65_R_M_DEV_REV_REV_100 (0x04U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 100

6.7.2.242 FS65_R_M_DEV_REV_REV_101

```
#define FS65_R_M_DEV_REV_REV_101 (0x05U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 101

6.7.2.243 FS65_R_M_DEV_REV_REV_110

```
#define FS65_R_M_DEV_REV_REV_110 (0x06U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 110

6.7.2.244 FS65_R_M_DEV_REV_REV_111

```
#define FS65_R_M_DEV_REV_REV_111 (0x07U << FS65_R_M_DEV_REV_SHIFT)
```

Silicon Rev. 111

6.7.2.245 FS65_R_M_DEV_REV_SHIFT

```
#define FS65_R_M_DEV_REV_SHIFT 0x00U
```

Device silicon revision

6.7.2.246 FS65_R_M_DFS_HW1_DISABLE

```
#define FS65_R_M_DFS_HW1_DISABLE (0x00U << FS65_R_M_DFS_HW1_SHIFT)
```

Deep fail-safe disable

6.7.2.247 FS65_R_M_DFS_HW1_ENABLE

```
#define FS65_R_M_DFS_HW1_ENABLE (0x01U << FS65_R_M_DFS_HW1_SHIFT)
```

Deep fail-safe enable

6.7.2.248 FS65_R_M_DFS_HW1_MASK

```
#define FS65_R_M_DFS_HW1_MASK 0x02U
```

Report the deep fail-safe hardware configuration (main logic)

6.7.2.249 FS65_R_M_DFS_HW1_SHIFT

```
#define FS65_R_M_DFS_HW1_SHIFT 0x01U
```

Report the deep fail-safe hardware configuration (main logic)

6.7.2.250 FS65_R_M_DFS_HW2_DISABLE

```
#define FS65_R_M_DFS_HW2_DISABLE (0x00U << FS65_R_M_DFS_HW2_SHIFT)
```

Deep fail-safe disable

6.7.2.251 FS65_R_M_DFS_HW2_ENABLE

```
#define FS65_R_M_DFS_HW2_ENABLE (0x01U << FS65_R_M_DFS_HW2_SHIFT)
```

Deep fail-safe enable

6.7.2.252 FS65_R_M_DFS_HW2_MASK

```
#define FS65_R_M_DFS_HW2_MASK 0x02U
```

Report the deep fail-safe hardware configuration (fail-safe logic)

6.7.2.253 FS65_R_M_DFS_HW2_SHIFT

```
#define FS65_R_M_DFS_HW2_SHIFT 0x01U
```

Report the deep fail-safe hardware configuration (fail-safe logic)

6.7.2.254 FS65_R_M_DFS_MASK

```
#define FS65_R_M_DFS_MASK 0x02U
```

Report if the device resume from deep fail-safe mode

6.7.2.255 FS65_R_M_DFS_NOT_DFS

```
#define FS65_R_M_DFS_NOT_DFS (0x00U << FS65_R_M_DFS_SHIFT)
```

Not in deep fail-safe

6.7.2.256 FS65_R_M_DFS_RESUME_DFS

```
#define FS65_R_M_DFS_RESUME_DFS (0x01U << FS65_R_M_DFS_SHIFT)
```

Resume from deep fail-safe

6.7.2.257 FS65_R_M_DFS_SHIFT

```
#define FS65_R_M_DFS_SHIFT 0x01U
```

Report if the device resume from deep fail-safe mode

6.7.2.258 FS65_R_M_ERR_INT_HW_ERROR

```
#define FS65_R_M_ERR_INT_HW_ERROR (0x01U << FS65_R_M_ERR_INT_HW_SHIFT)
```

Error detected

6.7.2.259 FS65_R_M_ERR_INT_HW_MASK

```
#define FS65_R_M_ERR_INT_HW_MASK 0x02U
```

Report an error from an internal redundant structure of the fail-safe state machine

6.7.2.260 FS65_R_M_ERR_INT_HW_NO_ERROR

```
#define FS65_R_M_ERR_INT_HW_NO_ERROR (0x00U << FS65_R_M_ERR_INT_HW_SHIFT)
```

No error

6.7.2.261 FS65_R_M_ERR_INT_HW_SHIFT

```
#define FS65_R_M_ERR_INT_HW_SHIFT 0x01U
```

Report an error from an internal redundant structure of the fail-safe state machine

6.7.2.262 FS65_R_M_ERR_INT_SW_ERROR

```
#define FS65_R_M_ERR_INT_SW_ERROR (0x01U << FS65_R_M_ERR_INT_SW_SHIFT)
```

Error detected

6.7.2.263 FS65_R_M_ERR_INT_SW_MASK

```
#define FS65_R_M_ERR_INT_SW_MASK 0x01U
```

Report an error from the EDC of the fail-safe state machine (error detection correction)

6.7.2.264 FS65_R_M_ERR_INT_SW_NO_ERROR

```
#define FS65_R_M_ERR_INT_SW_NO_ERROR (0x00U << FS65_R_M_ERR_INT_SW_SHIFT)
```

No error

6.7.2.265 FS65_R_M_ERR_INT_SW_SHIFT

```
#define FS65_R_M_ERR_INT_SW_SHIFT 0x00U
```

Report an error from the EDC of the fail-safe state machine (error detection correction)

6.7.2.266 FS65_R_M_FCRBM_OV_MASK

```
#define FS65_R_M_FCRBM_OV_MASK 0x02U
```

Report an overvoltage on FCRBM

6.7.2.267 FS65_R_M_FCRBM_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_FCRBM_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_FCRBM_OV_SHIFT)
```

No overvoltage (FB_Core - FCRBM $<$ 150 mV)

6.7.2.268 FS65_R_M_FCRBM_OV_OVERVOLTAGE

```
#define FS65_R_M_FCRBM_OV_OVERVOLTAGE (0x01U << FS65_R_M_FCRBM_OV_SHIFT)
```

Overvoltage detected (FB_Core - FCRBM $>$ 150 mV)

6.7.2.269 FS65_R_M_FCRBM_OV_SHIFT

```
#define FS65_R_M_FCRBM_OV_SHIFT 0x01U
```

Report an overvoltage on FCRBM

6.7.2.270 FS65_R_M_FCRBM_UV_MASK

```
#define FS65_R_M_FCRBM_UV_MASK 0x01U
```

Report an undervoltage on FCRBM

6.7.2.271 FS65_R_M_FCRBM_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_FCRBM_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_FCRBM_UV_SHIFT)
```

No undervoltage (FB_Core - FCRBM $>$ -150 mV)

6.7.2.272 FS65_R_M_FCRBM_UV_SHIFT

```
#define FS65_R_M_FCRBM_UV_SHIFT 0x00U
```

Report an undervoltage on FCRBM

6.7.2.273 FS65_R_M_FCRBM_UV_UNDERVOLTAGE

```
#define FS65_R_M_FCRBM_UV_UNDERVOLTAGE (0x01U << FS65_R_M_FCRBM_UV_SHIFT)
```

Undervoltage detected (FB_Core - FCRBM $<$ -150 mV)

6.7.2.274 FS65_R_M_FLT_ERR_MASK

```
#define FS65_R_M_FLT_ERR_MASK 0xE0U
```

Report the value of the fault error counter

6.7.2.275 FS65_R_M_FLT_ERR_SHIFT

```
#define FS65_R_M_FLT_ERR_SHIFT 0x05U
```

Report the value of the fault error counter

6.7.2.276 FS65_R_M_FS0B_DIAG_MASK

```
#define FS65_R_M_FS0B_DIAG_MASK 0x30U
```

Report a failure on FS0B

6.7.2.277 FS65_R_M_FS0B_DIAG_NO_FAILURE

```
#define FS65_R_M_FS0B_DIAG_NO_FAILURE (0x01U << FS65_R_M_FS0B_DIAG_SHIFT)
```

No Failure

6.7.2.278 FS65_R_M_FS0B_DIAG_SC_HIGH

```
#define FS65_R_M_FS0B_DIAG_SC_HIGH (0x03U << FS65_R_M_FS0B_DIAG_SHIFT)
```

Short-circuit high

6.7.2.279 FS65_R_M_FS0B_DIAG_SC_LOW

```
#define FS65_R_M_FS0B_DIAG_SC_LOW (0x02U << FS65_R_M_FS0B_DIAG_SHIFT)
```

Short-circuit low/open load

6.7.2.280 FS65_R_M_FS0B_DIAG_SHIFT

```
#define FS65_R_M_FS0B_DIAG_SHIFT 0x04U
```

Report a failure on FS0B

6.7.2.281 FS65_R_M_FS1_DISABLED

```
#define FS65_R_M_FS1_DISABLED (0x00U << FS65_R_M_FS1_SHIFT)
```

Disabled

6.7.2.282 FS65_R_M_FS1_ENABLE

```
#define FS65_R_M_FS1_ENABLE (0x01U << FS65_R_M_FS1_SHIFT)
```

Enabled

6.7.2.283 FS65_R_M_FS1_MASK

```
#define FS65_R_M_FS1_MASK 0x01U
```

Report the FS1B function availability (depends on part number)

6.7.2.284 FS65_R_M_FS1_SHIFT

```
#define FS65_R_M_FS1_SHIFT 0x00U
```

Report the FS1B function availability (depends on part number)

6.7.2.285 FS65_R_M_FS1B_DIAG_MASK

```
#define FS65_R_M_FS1B_DIAG_MASK 0x0CU
```

Report a failure on FS1B

6.7.2.286 FS65_R_M_FS1B_DIAG_NO_FAILURE

```
#define FS65_R_M_FS1B_DIAG_NO_FAILURE (0x01U << FS65_R_M_FS1B_DIAG_SHIFT)
```

No Failure

6.7.2.287 FS65_R_M_FS1B_DIAG_SC_HIGH

```
#define FS65_R_M_FS1B_DIAG_SC_HIGH (0x03U << FS65_R_M_FS1B_DIAG_SHIFT)
```

Short-circuit high

6.7.2.288 FS65_R_M_FS1B_DIAG_SC_LOW

```
#define FS65_R_M_FS1B_DIAG_SC_LOW (0x02U << FS65_R_M_FS1B_DIAG_SHIFT)
```

Short-circuit low/open load

6.7.2.289 FS65_R_M_FS1B_DIAG_SHIFT

```
#define FS65_R_M_FS1B_DIAG_SHIFT 0x02U
```

Report a failure on FS1B

6.7.2.290 FS65_R_M_FSO_G_FAILURE

```
#define FS65_R_M_FSO_G_FAILURE (0x01U << FS65_R_M_FSO_G_SHIFT)
```

Failure

6.7.2.291 FS65_R_M_FSO_G_MASK

```
#define FS65_R_M_FSO_G_MASK 0x10U
```

Report a fail-safe output failure

6.7.2.292 FS65_R_M_FSO_G_NO_FAILURE

```
#define FS65_R_M_FSO_G_NO_FAILURE (0x00U << FS65_R_M_FSO_G_SHIFT)
```

No failure

6.7.2.293 FS65_R_M_FSO_G_SHIFT

```
#define FS65_R_M_FSO_G_SHIFT 0x04U
```

Report a fail-safe output failure

6.7.2.294 FS65_R_M_FSXB_FSE_OCCURRED

```
#define FS65_R_M_FSXB_FSE_OCCURRED (0x01U << FS65_R_M_FSXB_SHIFT)
```

Fail-safe event occurred (default state at power-up and after LPOFF as FS0B/FS1B are asserted low)

6.7.2.295 FS65_R_M_FSXB_MASK

```
#define FS65_R_M_FSXB_MASK 0x40U
```

Report a fail safe event

6.7.2.296 FS65_R_M_FSXB_NO_FS

```
#define FS65_R_M_FSXB_NO_FS (0x00U << FS65_R_M_FSXB_SHIFT)
```

No fail-safe

6.7.2.297 FS65_R_M_FSXB_SHIFT

```
#define FS65_R_M_FSXB_SHIFT 0x06U
```

Report a fail safe event

6.7.2.298 FS65_R_M_ILIM_AUX_LIMITATION

```
#define FS65_R_M_ILIM_AUX_LIMITATION (0x01U << FS65_R_M_ILIM_AUX_SHIFT)
```

Current limitation (IAUX $>$ IAUX_LIM)

6.7.2.299 FS65_R_M_ILIM_AUX_MASK

```
#define FS65_R_M_ILIM_AUX_MASK 0x02U
```

Report a current limitation condition on VAUX

6.7.2.300 FS65_R_M_ILIM_AUX_NO_LIMITATION

```
#define FS65_R_M_ILIM_AUX_NO_LIMITATION (0x00U << FS65_R_M_ILIM_AUX_SHIFT)
```

No current limitation (IAUX < IAUX_LIM)

6.7.2.301 FS65_R_M_ILIM_AUX_OFF_LIMITATION

```
#define FS65_R_M_ILIM_AUX_OFF_LIMITATION (0x01U << FS65_R_M_ILIM_AUX_OFF_SHIFT)
```

T_LIMITATION > TAUX_LIM_OFF

6.7.2.302 FS65_R_M_ILIM_AUX_OFF_MASK

```
#define FS65_R_M_ILIM_AUX_OFF_MASK 0x01U
```

Maximum current limitation duration

6.7.2.303 FS65_R_M_ILIM_AUX_OFF_NO_LIMITATION

```
#define FS65_R_M_ILIM_AUX_OFF_NO_LIMITATION (0x00U << FS65_R_M_ILIM_AUX_OFF_SHIFT)
```

T_LIMITATION < TAUX_LIM_OFF

6.7.2.304 FS65_R_M_ILIM_AUX_OFF_SHIFT

```
#define FS65_R_M_ILIM_AUX_OFF_SHIFT 0x00U
```

Maximum current limitation duration

6.7.2.305 FS65_R_M_ILIM_AUX_SHIFT

```
#define FS65_R_M_ILIM_AUX_SHIFT 0x01U
```

Report a current limitation condition on VAUX

6.7.2.306 FS65_R_M_ILIM_CAN_LIMITATION

```
#define FS65_R_M_ILIM_CAN_LIMITATION (0x01U << FS65_R_M_ILIM_CAN_SHIFT)
```

Current limitation (ICAN > ICAN _LIM)

6.7.2.307 FS65_R_M_ILIM_CAN_MASK

```
#define FS65_R_M_ILIM_CAN_MASK 0x02U
```

Report a current limitation condition on VCAN

6.7.2.308 FS65_R_M_ILIM_CAN_NO_LIMITATION

```
#define FS65_R_M_ILIM_CAN_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CAN_SHIFT)
```

No current limitation (ICAN $<$ ICAN_LIM)

6.7.2.309 FS65_R_M_ILIM_CAN_SHIFT

```
#define FS65_R_M_ILIM_CAN_SHIFT 0x01U
```

Report a current limitation condition on VCAN

6.7.2.310 FS65_R_M_ILIM_CCA_LIMITATION

```
#define FS65_R_M_ILIM_CCA_LIMITATION (0x01U << FS65_R_M_ILIM_CCA_SHIFT)
```

Current limitation (ICCA $>$ ICCA_LIM)

6.7.2.311 FS65_R_M_ILIM_CCA_MASK

```
#define FS65_R_M_ILIM_CCA_MASK 0x02U
```

Report a current limitation condition on VCCA

6.7.2.312 FS65_R_M_ILIM_CCA_NO_LIMITATION

```
#define FS65_R_M_ILIM_CCA_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CCA_SHIFT)
```

No current limitation (ICCA $<$ ICCA_LIM)

6.7.2.313 FS65_R_M_ILIM_CCA_OFF_LIMITATION

```
#define FS65_R_M_ILIM_CCA_OFF_LIMITATION (0x01U << FS65_R_M_ILIM_CCA_OFF_SHIFT)
```

T_LIMITATION $>$ TCCA_LIM_OFF

6.7.2.314 FS65_R_M_ILIM_CCA_OFF_MASK

```
#define FS65_R_M_ILIM_CCA_OFF_MASK 0x01U
```

Maximum current limitation duration. Available only when an external PNP is connected

6.7.2.315 FS65_R_M_ILIM_CCA_OFF_NO_LIMITATION

```
#define FS65_R_M_ILIM_CCA_OFF_NO_LIMITATION (0x00U << FS65_R_M_ILIM_CCA_OFF_SHIFT)
```

T_LIMITATION < TCCA_LIM_OFF

6.7.2.316 FS65_R_M_ILIM_CCA_OFF_SHIFT

```
#define FS65_R_M_ILIM_CCA_OFF_SHIFT 0x00U
```

Maximum current limitation duration. Available only when an external PNP is connected

6.7.2.317 FS65_R_M_ILIM_CCA_SHIFT

```
#define FS65_R_M_ILIM_CCA_SHIFT 0x01U
```

Report a current limitation condition on VCCA

6.7.2.318 FS65_R_M_ILIM_PRE_LIMITATION

```
#define FS65_R_M_ILIM_PRE_LIMITATION (0x01U << FS65_R_M_ILIM_PRE_SHIFT)
```

Current limitation (IPRE_PK > IPRE_LIM)

6.7.2.319 FS65_R_M_ILIM_PRE_MASK

```
#define FS65_R_M_ILIM_PRE_MASK 0x02U
```

Report a current limitation condition on VPRE

6.7.2.320 FS65_R_M_ILIM_PRE_NO_LIMITATION

```
#define FS65_R_M_ILIM_PRE_NO_LIMITATION (0x00U << FS65_R_M_ILIM_PRE_SHIFT)
```

No current limitation (IPRE_PK < IPRE_LIM)

6.7.2.321 FS65_R_M_ILIM_PRE_SHIFT

```
#define FS65_R_M_ILIM_PRE_SHIFT 0x01U
```

Report a current limitation condition on VPRE

6.7.2.322 FS65_R_M_INIT_INIT

```
#define FS65_R_M_INIT_INIT (0x01U << FS65_R_M_INIT_SHIFT)
```

INIT mode

6.7.2.323 FS65_R_M_INIT_MASK

```
#define FS65_R_M_INIT_MASK 0x08U
```

Report if INIT mode of the main logic state machine is entered

6.7.2.324 FS65_R_M_INIT_NOT_INIT

```
#define FS65_R_M_INIT_NOT_INIT (0x00U << FS65_R_M_INIT_SHIFT)
```

Not in INIT mode

6.7.2.325 FS65_R_M_INIT_SHIFT

```
#define FS65_R_M_INIT_SHIFT 0x03U
```

Report if INIT mode of the main logic state machine is entered

6.7.2.326 FS65_R_M_IO_0_HIGH

```
#define FS65_R_M_IO_0_HIGH (0x01U << FS65_R_M_IO_0_SHIFT)
```

High

6.7.2.327 FS65_R_M_IO_0_LOW

```
#define FS65_R_M_IO_0_LOW (0x00U << FS65_R_M_IO_0_SHIFT)
```

Low

6.7.2.328 FS65_R_M_IO_0_MASK

```
#define FS65_R_M_IO_0_MASK 0x01U
```

Report IO_0 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.329 FS65_R_M_IO_0_SHIFT

```
#define FS65_R_M_IO_0_SHIFT 0x00U
```

Report IO_0 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.330 FS65_R_M_IO_0_WU_EVENT

```
#define FS65_R_M_IO_0_WU_EVENT (0x01U << FS65_R_M_IO_0_WU_SHIFT)
```

Wake-up event detected

6.7.2.331 FS65_R_M_IO_0_WU_MASK

```
#define FS65_R_M_IO_0_WU_MASK 0x08U
```

Report a wake-up event from IO_0

6.7.2.332 FS65_R_M_IO_0_WU_NO_EVENT

```
#define FS65_R_M_IO_0_WU_NO_EVENT (0x00U << FS65_R_M_IO_0_WU_SHIFT)
```

No wake-up

6.7.2.333 FS65_R_M_IO_0_WU_SHIFT

```
#define FS65_R_M_IO_0_WU_SHIFT 0x03U
```

Report a wake-up event from IO_0

6.7.2.334 FS65_R_M_IO_23_FAIL_ERROR

```
#define FS65_R_M_IO_23_FAIL_ERROR (0x01U << FS65_R_M_IO_23_FAIL_SHIFT)
```

Error detected

6.7.2.335 FS65_R_M_IO_23_FAIL_MASK

```
#define FS65_R_M_IO_23_FAIL_MASK 0x02U
```

Report an error in the FCCU protocol

6.7.2.336 FS65_R_M_IO_23_FAIL_NO_ERROR

```
#define FS65_R_M_IO_23_FAIL_NO_ERROR (0x00U << FS65_R_M_IO_23_FAIL_SHIFT)
```

No error

6.7.2.337 FS65_R_M_IO_23_FAIL_SHIFT

```
#define FS65_R_M_IO_23_FAIL_SHIFT 0x01U
```

Report an error in the FCCU protocol

6.7.2.338 FS65_R_M_IO_2_HIGH

```
#define FS65_R_M_IO_2_HIGH (0x01U << FS65_R_M_IO_2_SHIFT)
```

High

6.7.2.339 FS65_R_M_IO_2_LOW

```
#define FS65_R_M_IO_2_LOW (0x00U << FS65_R_M_IO_2_SHIFT)
```

Low

6.7.2.340 FS65_R_M_IO_2_MASK

```
#define FS65_R_M_IO_2_MASK 0x08U
```

Report IO_2 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.341 FS65_R_M_IO_2_SHIFT

```
#define FS65_R_M_IO_2_SHIFT 0x03U
```

Report IO_2 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.342 FS65_R_M_IO_2_WU_EVENT

```
#define FS65_R_M_IO_2_WU_EVENT (0x01U << FS65_R_M_IO_2_WU_SHIFT)
```

Wake-up event detected

6.7.2.343 FS65_R_M_IO_2_WU_MASK

```
#define FS65_R_M_IO_2_WU_MASK 0x10U
```

Report a wake-up event from IO_2

6.7.2.344 FS65_R_M_IO_2_WU_NO_EVENT

```
#define FS65_R_M_IO_2_WU_NO_EVENT (0x00U << FS65_R_M_IO_2_WU_SHIFT)
```

No wake-up

6.7.2.345 FS65_R_M_IO_2_WU_SHIFT

```
#define FS65_R_M_IO_2_WU_SHIFT 0x04U
```

Report a wake-up event from IO_2

6.7.2.346 FS65_R_M_IO_3_HIGH

```
#define FS65_R_M_IO_3_HIGH (0x01U << FS65_R_M_IO_3_SHIFT)
```

High

6.7.2.347 FS65_R_M_IO_3_LOW

```
#define FS65_R_M_IO_3_LOW (0x00U << FS65_R_M_IO_3_SHIFT)
```

Low

6.7.2.348 FS65_R_M_IO_3_MASK

```
#define FS65_R_M_IO_3_MASK 0x10U
```

Report IO_3 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.349 FS65_R_M_IO_3_SHIFT

```
#define FS65_R_M_IO_3_SHIFT 0x04U
```

Report IO_3 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.350 FS65_R_M_IO_3_WU_EVENT

```
#define FS65_R_M_IO_3_WU_EVENT (0x01U << FS65_R_M_IO_3_WU_SHIFT)
```

Wake-up event detected

6.7.2.351 FS65_R_M_IO_3_WU_MASK

```
#define FS65_R_M_IO_3_WU_MASK 0x20U
```

Report a wake-up event from IO_3

6.7.2.352 FS65_R_M_IO_3_WU_NO_EVENT

```
#define FS65_R_M_IO_3_WU_NO_EVENT (0x00U << FS65_R_M_IO_3_WU_SHIFT)
```

No wake-up

6.7.2.353 FS65_R_M_IO_3_WU_SHIFT

```
#define FS65_R_M_IO_3_WU_SHIFT 0x05U
```

Report a wake-up event from IO_3

6.7.2.354 FS65_R_M_IO_45_FAIL_ERROR

```
#define FS65_R_M_IO_45_FAIL_ERROR (0x01U << FS65_R_M_IO_45_FAIL_SHIFT)
```

Error detected

6.7.2.355 FS65_R_M_IO_45_FAIL_MASK

```
#define FS65_R_M_IO_45_FAIL_MASK 0x01U
```

Report an error in the IO_45 protocol

6.7.2.356 FS65_R_M_IO_45_FAIL_NO_ERROR

```
#define FS65_R_M_IO_45_FAIL_NO_ERROR (0x00U << FS65_R_M_IO_45_FAIL_SHIFT)
```

No error

6.7.2.357 FS65_R_M_IO_45_FAIL_SHIFT

```
#define FS65_R_M_IO_45_FAIL_SHIFT 0x00U
```

Report an error in the IO_45 protocol

6.7.2.358 FS65_R_M_IO_4_HIGH

```
#define FS65_R_M_IO_4_HIGH (0x01U << FS65_R_M_IO_4_SHIFT)
```

High

6.7.2.359 FS65_R_M_IO_4_LOW

```
#define FS65_R_M_IO_4_LOW (0x00U << FS65_R_M_IO_4_SHIFT)
```

Low

6.7.2.360 FS65_R_M_IO_4_MASK

```
#define FS65_R_M_IO_4_MASK 0x40U
```

Report IO_4 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.361 FS65_R_M_IO_4_SHIFT

```
#define FS65_R_M_IO_4_SHIFT 0x06U
```

Report IO_4 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.362 FS65_R_M_IO_4_WU_EVENT

```
#define FS65_R_M_IO_4_WU_EVENT (0x01U << FS65_R_M_IO_4_WU_SHIFT)
```

Wake-up event detected

6.7.2.363 FS65_R_M_IO_4_WU_MASK

```
#define FS65_R_M_IO_4_WU_MASK 0x40U
```

Report a wake-up event from IO_4

6.7.2.364 FS65_R_M_IO_4_WU_NO_EVENT

```
#define FS65_R_M_IO_4_WU_NO_EVENT (0x00U << FS65_R_M_IO_4_WU_SHIFT)
```

No wake-up

6.7.2.365 FS65_R_M_IO_4_WU_SHIFT

```
#define FS65_R_M_IO_4_WU_SHIFT 0x06U
```

Report a wake-up event from IO_4

6.7.2.366 FS65_R_M_IO_5_HIGH

```
#define FS65_R_M_IO_5_HIGH (0x01U << FS65_R_M_IO_5_SHIFT)
```

High

6.7.2.367 FS65_R_M_IO_5_LOW

```
#define FS65_R_M_IO_5_LOW (0x00U << FS65_R_M_IO_5_SHIFT)
```

Low

6.7.2.368 FS65_R_M_IO_5_MASK

```
#define FS65_R_M_IO_5_MASK 0x80U
```

Report IO_5 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.369 FS65_R_M_IO_5_SHIFT

```
#define FS65_R_M_IO_5_SHIFT 0x07U
```

Report IO_5 digital state in normal mode. No update in LPOFF mode since wake-up features available

6.7.2.370 FS65_R_M_IO_5_WU_EVENT

```
#define FS65_R_M_IO_5_WU_EVENT (0x01U << FS65_R_M_IO_5_WU_SHIFT)
```

Wake-up event detected

6.7.2.371 FS65_R_M_IO_5_WU_MASK

```
#define FS65_R_M_IO_5_WU_MASK 0x80U
```

Report a wake-up event from IO_5

6.7.2.372 FS65_R_M_IO_5_WU_NO_EVENT

```
#define FS65_R_M_IO_5_WU_NO_EVENT (0x00U << FS65_R_M_IO_5_WU_SHIFT)
```

No wake-up

6.7.2.373 FS65_R_M_IO_5_WU_SHIFT

```
#define FS65_R_M_IO_5_WU_SHIFT 0x07U
```

Report a wake-up event from IO_5

6.7.2.374 FS65_R_M_IO_FS_G_ERROR

```
#define FS65_R_M_IO_FS_G_ERROR (0x01U << FS65_R_M_IO_FS_G_SHIFT)
```

Error detected

6.7.2.375 FS65_R_M_IO_FS_G_MASK

```
#define FS65_R_M_IO_FS_G_MASK 0x08U
```

Report an IO monitoring error

6.7.2.376 FS65_R_M_IO_FS_G_NO_ERROR

```
#define FS65_R_M_IO_FS_G_NO_ERROR (0x00U << FS65_R_M_IO_FS_G_SHIFT)
```

No error

6.7.2.377 FS65_R_M_IO_FS_G_SHIFT

```
#define FS65_R_M_IO_FS_G_SHIFT 0x03U
```

Report an IO monitoring error

6.7.2.378 FS65_R_M_IPFF_IPFF

```
#define FS65_R_M_IPFF_IPFF (0x01U << FS65_R_M_IPFF_SHIFT)
```

IPFF mode activated

6.7.2.379 FS65_R_M_IPFF_MASK

```
#define FS65_R_M_IPFF_MASK 0x20U
```

Input power feed forward (IPFF)

6.7.2.380 FS65_R_M_IPFF_NORMAL

```
#define FS65_R_M_IPFF_NORMAL (0x00U << FS65_R_M_IPFF_SHIFT)
```

Normal operation

6.7.2.381 FS65_R_M_IPFF_SHIFT

```
#define FS65_R_M_IPFF_SHIFT 0x05U
```

Input power feed forward (IPFF)

6.7.2.382 FS65_R_M_LDT_INT_MASK

```
#define FS65_R_M_LDT_INT_MASK 0x01U
```

Counter status

6.7.2.383 FS65_R_M_LDT_INT_NOT_RUNNING

```
#define FS65_R_M_LDT_INT_NOT_RUNNING (0x00U << FS65_R_M_LDT_INT_SHIFT)
```

Counter not running

6.7.2.384 FS65_R_M_LDT_INT_RUNNING

```
#define FS65_R_M_LDT_INT_RUNNING (0x01U << FS65_R_M_LDT_INT_SHIFT)
```

Counter running

6.7.2.385 FS65_R_M_LDT_INT_SHIFT

```
#define FS65_R_M_LDT_INT_SHIFT 0x00U
```

Counter status

6.7.2.386 FS65_R_M_LDT_RUNNING_MASK

```
#define FS65_R_M_LDT_RUNNING_MASK 0x08U
```

Counter status

6.7.2.387 FS65_R_M_LDT_RUNNING_NOT_RUNNING

```
#define FS65_R_M_LDT_RUNNING_NOT_RUNNING (0x00U << FS65_R_M_LDT_RUNNING_SHIFT)
```

Counter not running

6.7.2.388 FS65_R_M_LDT_RUNNING_RUNNING

```
#define FS65_R_M_LDT_RUNNING_RUNNING (0x01U << FS65_R_M_LDT_RUNNING_SHIFT)
```

Counter running

6.7.2.389 FS65_R_M_LDT_RUNNING_SHIFT

```
#define FS65_R_M_LDT_RUNNING_SHIFT 0x03U
```

Counter status

6.7.2.390 FS65_R_M_LDT_WU_EVENT

```
#define FS65_R_M_LDT_WU_EVENT (0x01U << FS65_R_M_LDT_WU_SHIFT)
```

Wake-up event detected

6.7.2.391 FS65_R_M_LDT_WU_MASK

```
#define FS65_R_M_LDT_WU_MASK 0x02U
```

Report a wake-up event from long duration timer

6.7.2.392 FS65_R_M_LDT_WU_NO_EVENT

```
#define FS65_R_M_LDT_WU_NO_EVENT (0x00U << FS65_R_M_LDT_WU_SHIFT)
```

No wake-up

6.7.2.393 FS65_R_M_LDT_WU_SHIFT

```
#define FS65_R_M_LDT_WU_SHIFT 0x01U
```

Report a wake-up event from long duration timer

6.7.2.394 FS65_R_M_LIN_DOM_FAILURE

```
#define FS65_R_M_LIN_DOM_FAILURE (0x01U << FS65_R_M_LIN_DOM_SHIFT)
```

Failure detected

6.7.2.395 FS65_R_M_LIN_DOM_MASK

```
#define FS65_R_M_LIN_DOM_MASK 0x80U
```

LIN bus dominant clamping detection

6.7.2.396 FS65_R_M_LIN_DOM_NO_FAILURE

```
#define FS65_R_M_LIN_DOM_NO_FAILURE (0x00U << FS65_R_M_LIN_DOM_SHIFT)
```

No failure

6.7.2.397 FS65_R_M_LIN_DOM_SHIFT

```
#define FS65_R_M_LIN_DOM_SHIFT 0x07U
```

LIN bus dominant clamping detection

6.7.2.398 FS65_R_M_LIN_OT_FAILURE

```
#define FS65_R_M_LIN_OT_FAILURE (0x01U << FS65_R_M_LIN_OT_SHIFT)
```

Failure detected

6.7.2.399 FS65_R_M_LIN_OT_MASK

```
#define FS65_R_M_LIN_OT_MASK 0x08U
```

LIN overtemperature detection

6.7.2.400 FS65_R_M_LIN_OT_NO_FAILURE

```
#define FS65_R_M_LIN_OT_NO_FAILURE (0x00U << FS65_R_M_LIN_OT_SHIFT)
```

No failure

6.7.2.401 FS65_R_M_LIN_OT_SHIFT

```
#define FS65_R_M_LIN_OT_SHIFT 0x03U
```

LIN overtemperature detection

6.7.2.402 FS65_R_M_LIN_WU_MASK

```
#define FS65_R_M_LIN_WU_MASK 0x01U
```

Report a wake-up event from the LIN

6.7.2.403 FS65_R_M_LIN_WU_NO_WU

```
#define FS65_R_M_LIN_WU_NO_WU (0x00U << FS65_R_M_LIN_WU_SHIFT)
```

No wake-up

6.7.2.404 FS65_R_M_LIN_WU_SHIFT

```
#define FS65_R_M_LIN_WU_SHIFT 0x00U
```

Report a wake-up event from the LIN

6.7.2.405 FS65_R_M_LIN_WU_WU

```
#define FS65_R_M_LIN_WU_WU (0x01U << FS65_R_M_LIN_WU_SHIFT)
```

Wake-up detected

6.7.2.406 FS65_R_M_LPOFF_MASK

```
#define FS65_R_M_LPOFF_MASK 0x01U
```

Report if the device resume from LPOFF-sleep or LPOFF_AUTO_WU mode

6.7.2.407 FS65_R_M_LPOFF_NOT_LPOFF

```
#define FS65_R_M_LPOFF_NOT_LPOFF (0x00U << FS65_R_M_LPOFF_SHIFT)
```

Not in LPOFF

6.7.2.408 FS65_R_M_LPOFF_RESUME_LPOFF

```
#define FS65_R_M_LPOFF_RESUME_LPOFF (0x01U << FS65_R_M_LPOFF_SHIFT)
```

Resume from LPOFF

6.7.2.409 FS65_R_M_LPOFF_SHIFT

```
#define FS65_R_M_LPOFF_SHIFT 0x00U
```

Report if the device resume from LPOFF-sleep or LPOFF_AUTO_WU mode

6.7.2.410 FS65_R_M_LS_DETECT_BUCK_BOOST

```
#define FS65_R_M_LS_DETECT_BUCK_BOOST (0x00U << FS65_R_M_LS_DETECT_SHIFT)
```

Buck-boost

6.7.2.411 FS65_R_M_LS_DETECT_BUCK_ONLY

```
#define FS65_R_M_LS_DETECT_BUCK_ONLY (0x01U << FS65_R_M_LS_DETECT_SHIFT)
```

Buck only

6.7.2.412 FS65_R_M_LS_DETECT_MASK

```
#define FS65_R_M_LS_DETECT_MASK 0x80U
```

Report the hardware configuration of VPRE

6.7.2.413 FS65_R_M_LS_DETECT_SHIFT

```
#define FS65_R_M_LS_DETECT_SHIFT 0x07U
```

Report the hardware configuration of VPRE

6.7.2.414 FS65_R_M_NORMAL_MASK

```
#define FS65_R_M_NORMAL_MASK 0x04U
```

Report if normal mode of the main logic state machine is entered

6.7.2.415 FS65_R_M_NORMAL_NORMAL

```
#define FS65_R_M_NORMAL_NORMAL (0x01U << FS65_R_M_NORMAL_SHIFT)
```

Normal mode

6.7.2.416 FS65_R_M_NORMAL_NOT_NORMAL

```
#define FS65_R_M_NORMAL_NOT_NORMAL (0x00U << FS65_R_M_NORMAL_SHIFT)
```

Not in normal mode

6.7.2.417 FS65_R_M_NORMAL_SHIFT

```
#define FS65_R_M_NORMAL_SHIFT 0x02U
```

Report if normal mode of the main logic state machine is entered

6.7.2.418 FS65_R_M_PHY_CAN

```
#define FS65_R_M_PHY_CAN (0x01U << FS65_R_M_PHY_SHIFT)
```

CAN only

6.7.2.419 FS65_R_M_PHY_CAN_LIN

```
#define FS65_R_M_PHY_CAN_LIN (0x03U << FS65_R_M_PHY_SHIFT)
```

CAN and LIN

6.7.2.420 FS65_R_M_PHY_LIN

```
#define FS65_R_M_PHY_LIN (0x02U << FS65_R_M_PHY_SHIFT)
```

LIN only

6.7.2.421 FS65_R_M_PHY_MASK

```
#define FS65_R_M_PHY_MASK 0x30U
```

CAN or LIN physical layer

6.7.2.422 FS65_R_M_PHY_NOCAN_NOLIN

```
#define FS65_R_M_PHY_NOCAN_NOLIN (0x00U << FS65_R_M_PHY_SHIFT)
```

No CAN/no LIN

6.7.2.423 FS65_R_M_PHY_SHIFT

```
#define FS65_R_M_PHY_SHIFT 0x04U
```

CAN or LIN physical layer

6.7.2.424 FS65_R_M_PHY_WU_EVENT

```
#define FS65_R_M_PHY_WU_EVENT (0x01U << FS65_R_M_PHY_WU_SHIFT)
```

Wake-up event detected

6.7.2.425 FS65_R_M_PHY_WU_MASK

```
#define FS65_R_M_PHY_WU_MASK 0x01U
```

Report a wake-up event from CAN or LIN

6.7.2.426 FS65_R_M_PHY_WU_NO_EVENT

```
#define FS65_R_M_PHY_WU_NO_EVENT (0x00U << FS65_R_M_PHY_WU_SHIFT)
```

No wake-up

6.7.2.427 FS65_R_M_PHY_WU_SHIFT

```
#define FS65_R_M_PHY_WU_SHIFT 0x00U
```

Report a wake-up event from CAN or LIN

6.7.2.428 FS65_R_M_RSTB_DIAG_MASK

```
#define FS65_R_M_RSTB_DIAG_MASK 0x40U
```

Report a RSTB short-circuit to high

6.7.2.429 FS65_R_M_RSTB_DIAG_NO_FAILURE

```
#define FS65_R_M_RSTB_DIAG_NO_FAILURE (0x00U << FS65_R_M_RSTB_DIAG_SHIFT)
```

No Failure

6.7.2.430 FS65_R_M_RSTB_DIAG_SC_HIGH

```
#define FS65_R_M_RSTB_DIAG_SC_HIGH (0x01U << FS65_R_M_RSTB_DIAG_SHIFT)
```

Short-circuit high

6.7.2.431 FS65_R_M_RSTB_DIAG_SHIFT

```
#define FS65_R_M_RSTB_DIAG_SHIFT 0x06U
```

Report a RSTB short-circuit to high

6.7.2.432 FS65_R_M_RSTB_EXT_EXTERNAL

```
#define FS65_R_M_RSTB_EXT_EXTERNAL (0x01U << FS65_R_M_RSTB_EXT_SHIFT)
```

External RSTB

6.7.2.433 FS65_R_M_RSTB_EXT_MASK

```
#define FS65_R_M_RSTB_EXT_MASK 0x80U
```

Report an external RSTB

6.7.2.434 FS65_R_M_RSTB_EXT_NO

```
#define FS65_R_M_RSTB_EXT_NO (0x00U << FS65_R_M_RSTB_EXT_SHIFT)
```

No external RSTB

6.7.2.435 FS65_R_M_RSTB_EXT_SHIFT

```
#define FS65_R_M_RSTB_EXT_SHIFT 0x07U
```

Report an external RSTB

6.7.2.436 FS65_R_M_RSTB_MASK

```
#define FS65_R_M_RSTB_MASK 0x80U
```

Report a reset event

6.7.2.437 FS65_R_M_RSTB_NO_RESET

```
#define FS65_R_M_RSTB_NO_RESET (0x00U << FS65_R_M_RSTB_SHIFT)
```

No reset

6.7.2.438 FS65_R_M_RSTB_RESET_OCCURRED

```
#define FS65_R_M_RSTB_RESET_OCCURRED (0x01U << FS65_R_M_RSTB_SHIFT)
```

Reset occurred

6.7.2.439 FS65_R_M_RSTB_SHIFT

```
#define FS65_R_M_RSTB_SHIFT 0x07U
```

Report a reset event

6.7.2.440 FS65_R_M_RXD_REC_FAILURE

```
#define FS65_R_M_RXD_REC_FAILURE (0x01U << FS65_R_M_RXD_REC_SHIFT)
```

Failure detected

6.7.2.441 FS65_R_M_RXD_REC_MASK

```
#define FS65_R_M_RXD_REC_MASK 0x02U
```

RXD recessive clamping detection (short-circuit to 5.0 V)

6.7.2.442 FS65_R_M_RXD_REC_NO_FAILURE

```
#define FS65_R_M_RXD_REC_NO_FAILURE (0x00U << FS65_R_M_RXD_REC_SHIFT)
```

No failure

6.7.2.443 FS65_R_M_RXD_REC_SHIFT

```
#define FS65_R_M_RXD_REC_SHIFT 0x01U
```

RXD recessive clamping detection (short-circuit to 5.0 V)

6.7.2.444 FS65_R_M_RXDL_REC_FAILURE

```
#define FS65_R_M_RXDL_REC_FAILURE (0x01U << FS65_R_M_RXDL_REC_SHIFT)
```

Failure detected

6.7.2.445 FS65_R_M_RXDL_REC_MASK

```
#define FS65_R_M_RXDL_REC_MASK 0x10U
```

LIN RXD recessive clamping detection (short-circuit to 5.0 V)

6.7.2.446 FS65_R_M_RXDL_REC_NO_FAILURE

```
#define FS65_R_M_RXDL_REC_NO_FAILURE (0x00U << FS65_R_M_RXDL_REC_SHIFT)
```

No failure

6.7.2.447 FS65_R_M_RXDL_REC_SHIFT

```
#define FS65_R_M_RXDL_REC_SHIFT 0x04U
```

LIN RXD recessive clamping detection (short-circuit to 5.0 V)

6.7.2.448 FS65_R_M_SPI_CLK_16_CLK_CYCLES

```
#define FS65_R_M_SPI_CLK_16_CLK_CYCLES (0x00U << FS65_R_M_SPI_CLK_SHIFT)
```

16 clock cycles during NCS low

6.7.2.449 FS65_R_M_SPI_CLK_MASK

```
#define FS65_R_M_SPI_CLK_MASK 0x20U
```

SCLK error detection

6.7.2.450 FS65_R_M_SPI_CLK_SHIFT

```
#define FS65_R_M_SPI_CLK_SHIFT 0x05U
```

SCLK error detection

6.7.2.451 FS65_R_M_SPI_CLK_WRONG_NUMBER

```
#define FS65_R_M_SPI_CLK_WRONG_NUMBER (0x01U << FS65_R_M_SPI_CLK_SHIFT)
```

Wrong number of clock cycles (<16 or > 16)

6.7.2.452 FS65_R_M_SPI_ERR_ERROR

```
#define FS65_R_M_SPI_ERR_ERROR (0x01U << FS65_R_M_SPI_ERR_SHIFT)
```

Error detected in the secured bits

6.7.2.453 FS65_R_M_SPI_ERR_MASK

```
#define FS65_R_M_SPI_ERR_MASK 0x80U
```

Secured SPI communication check

6.7.2.454 FS65_R_M_SPI_ERR_NO_ERROR

```
#define FS65_R_M_SPI_ERR_NO_ERROR (0x00U << FS65_R_M_SPI_ERR_SHIFT)
```

No error

6.7.2.455 FS65_R_M_SPI_ERR_SHIFT

```
#define FS65_R_M_SPI_ERR_SHIFT 0x07U
```

Secured SPI communication check

6.7.2.456 FS65_R_M_SPI_PARITY_ERROR

```
#define FS65_R_M_SPI_PARITY_ERROR (0x01U << FS65_R_M_SPI_PARITY_SHIFT)
```

Parity bit error

6.7.2.457 FS65_R_M_SPI_PARITY_MASK

```
#define FS65_R_M_SPI_PARITY_MASK 0x02U
```

SPI parity bit error detection

6.7.2.458 FS65_R_M_SPI_PARITY_OK

```
#define FS65_R_M_SPI_PARITY_OK (0x00U << FS65_R_M_SPI_PARITY_SHIFT)
```

Parity bit ok

6.7.2.459 FS65_R_M_SPI_PARITY_SHIFT

```
#define FS65_R_M_SPI_PARITY_SHIFT 0x01U
```

SPI parity bit error detection

6.7.2.460 FS65_R_M_SPI_REQ_MASK

```
#define FS65_R_M_SPI_REQ_MASK 0x08U
```

Invalid SPI access (wrong write or read, write to INIT registers in normal mode, wrong address)

6.7.2.461 FS65_R_M_SPI_REQ_NO_ERROR

```
#define FS65_R_M_SPI_REQ_NO_ERROR (0x00U << FS65_R_M_SPI_REQ_SHIFT)
```

No error

6.7.2.462 FS65_R_M_SPI_REQ_SHIFT

```
#define FS65_R_M_SPI_REQ_SHIFT 0x03U
```

Invalid SPI access (wrong write or read, write to INIT registers in normal mode, wrong address)

6.7.2.463 FS65_R_M_SPI_REQ_SPI_VIOLATION

```
#define FS65_R_M_SPI_REQ_SPI_VIOLATION (0x01U << FS65_R_M_SPI_REQ_SHIFT)
```

SPI violation

6.7.2.464 FS65_R_M_TDXL_DOM_FAILURE

```
#define FS65_R_M_TDXL_DOM_FAILURE (0x01U << FS65_R_M_TDXL_DOM_SHIFT)
```

Failure detected

6.7.2.465 FS65_R_M_TDXL_DOM_MASK

```
#define FS65_R_M_TDXL_DOM_MASK 0x40U
```

LIN TXD dominant clamping detection (short-circuit to GND)

6.7.2.466 FS65_R_M_TDXL_DOM_NO_FAILURE

```
#define FS65_R_M_TDXL_DOM_NO_FAILURE (0x00U << FS65_R_M_TDXL_DOM_SHIFT)
```

No failure

6.7.2.467 FS65_R_M_TDXL_DOM_SHIFT

```
#define FS65_R_M_TDXL_DOM_SHIFT 0x06U
```

LIN TXD dominant clamping detection (short-circuit to GND)

6.7.2.468 FS65_R_M_TSD_AUX_MASK

```
#define FS65_R_M_TSD_AUX_MASK 0x10U
```

Thermal shutdown of VAUX

6.7.2.469 FS65_R_M_TSD_AUX_NO_TSD

```
#define FS65_R_M_TSD_AUX_NO_TSD (0x00U << FS65_R_M_TSD_AUX_SHIFT)
```

No TSD (TJ < TSD_AUX)

6.7.2.470 FS65_R_M_TSD_AUX_SHIFT

```
#define FS65_R_M_TSD_AUX_SHIFT 0x04U
```

Thermal shutdown of VAUX

6.7.2.471 FS65_R_M_TSD_AUX_TSD_OCCURRED

```
#define FS65_R_M_TSD_AUX_TSD_OCCURRED (0x01U << FS65_R_M_TSD_AUX_SHIFT)
```

TSD occurred (TJ > TSD_AUX)

6.7.2.472 FS65_R_M_TSD_CAN_MASK

```
#define FS65_R_M_TSD_CAN_MASK 0x10U
```

Thermal shutdown of VCAN

6.7.2.473 FS65_R_M_TSD_CAN_NO_TSD

```
#define FS65_R_M_TSD_CAN_NO_TSD (0x00U << FS65_R_M_TSD_CAN_SHIFT)
```

NO TSD (TJ < TSD_CAN)

6.7.2.474 FS65_R_M_TSD_CAN_SHIFT

```
#define FS65_R_M_TSD_CAN_SHIFT 0x04U
```

Thermal shutdown of VCAN

6.7.2.475 FS65_R_M_TSD_CAN_TSD_OCCURRED

```
#define FS65_R_M_TSD_CAN_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CAN_SHIFT)
```

TSD occurred (TJ > TSD_CAN)

6.7.2.476 FS65_R_M_TSD_CCA_MASK

```
#define FS65_R_M_TSD_CCA_MASK 0x10U
```

Thermal shutdown of VCCA

6.7.2.477 FS65_R_M_TSD_CCA_NO_TSD

```
#define FS65_R_M_TSD_CCA_NO_TSD (0x00U << FS65_R_M_TSD_CCA_SHIFT)
```

NO TSD (TJ < TSD_CCA)

6.7.2.478 FS65_R_M_TSD_CCA_SHIFT

```
#define FS65_R_M_TSD_CCA_SHIFT 0x04U
```

Thermal shutdown of VCCA

6.7.2.479 FS65_R_M_TSD_CCA_TSD_OCCURRED

```
#define FS65_R_M_TSD_CCA_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CCA_SHIFT)
```

TSD occurred (TJ > TSD_CCA)

6.7.2.480 FS65_R_M_TSD_CORE_MASK

```
#define FS65_R_M_TSD_CORE_MASK 0x10U
```

Thermal shutdown of VCORE

6.7.2.481 FS65_R_M_TSD_CORE_NO_TSD

```
#define FS65_R_M_TSD_CORE_NO_TSD (0x00U << FS65_R_M_TSD_CORE_SHIFT)
```

No TSD (TJ < TSD_PRE)

6.7.2.482 FS65_R_M_TSD_CORE_SHIFT

```
#define FS65_R_M_TSD_CORE_SHIFT 0x04U
```

Thermal shutdown of VCORE

6.7.2.483 FS65_R_M_TSD_CORE_TSD_OCCURRED

```
#define FS65_R_M_TSD_CORE_TSD_OCCURRED (0x01U << FS65_R_M_TSD_CORE_SHIFT)
```

TSD occurred (TJ > TSD_CORE)

6.7.2.484 FS65_R_M_TSD_PRE_MASK

```
#define FS65_R_M_TSD_PRE_MASK 0x10U
```

Thermal shutdown of VPRE

6.7.2.485 FS65_R_M_TSD_PRE_NO_TSD

```
#define FS65_R_M_TSD_PRE_NO_TSD (0x00U << FS65_R_M_TSD_PRE_SHIFT)
```

No TSD (TJ < TSD_PRE)

6.7.2.486 FS65_R_M_TSD_PRE_SHIFT

```
#define FS65_R_M_TSD_PRE_SHIFT 0x04U
```

Thermal shutdown of VPRE

6.7.2.487 FS65_R_M_TSD_PRE_TSD_OCCURRED

```
#define FS65_R_M_TSD_PRE_TSD_OCCURRED (0x01U << FS65_R_M_TSD_PRE_SHIFT)
```

TSD occurred (TJ > TSD_PRE)

6.7.2.488 FS65_R_M_TWARN_CCA_MASK

```
#define FS65_R_M_TWARN_CCA_MASK 0x20U
```

Report a thermal warning from VCCA. Available only for internal pass MOSFET

6.7.2.489 FS65_R_M_TWARN_CCA_NO_WARNING

```
#define FS65_R_M_TWARN_CCA_NO_WARNING (0x00U << FS65_R_M_TWARN_CCA_SHIFT)
```

No thermal warning (TJ < TWARN_CCA)

6.7.2.490 FS65_R_M_TWARN_CCA_SHIFT

```
#define FS65_R_M_TWARN_CCA_SHIFT 0x05U
```

Report a thermal warning from VCCA. Available only for internal pass MOSFET

6.7.2.491 FS65_R_M_TWARN_CCA_WARNING

```
#define FS65_R_M_TWARN_CCA_WARNING (0x01U << FS65_R_M_TWARN_CCA_SHIFT)
```

Thermal warning (TJ > TWARN_CCA)

6.7.2.492 FS65_R_M_TWARN_CORE_MASK

```
#define FS65_R_M_TWARN_CORE_MASK 0x20U
```

Report a thermal warning from VCORE

6.7.2.493 FS65_R_M_TWARN_CORE_NO_WARNING

```
#define FS65_R_M_TWARN_CORE_NO_WARNING (0x00U << FS65_R_M_TWARN_CORE_SHIFT)
```

No thermal warning (TJ < TWARN_CORE)

6.7.2.494 FS65_R_M_TWARN_CORE_SHIFT

```
#define FS65_R_M_TWARN_CORE_SHIFT 0x05U
```

Report a thermal warning from VCORE

6.7.2.495 FS65_R_M_TWARN_CORE_WARNING

```
#define FS65_R_M_TWARN_CORE_WARNING (0x01U << FS65_R_M_TWARN_CORE_SHIFT)
```

Thermal warning (TJ > TWARN_CORE)

6.7.2.496 FS65_R_M_TWARN_PRE_MASK

```
#define FS65_R_M_TWARN_PRE_MASK 0x20U
```

Report a thermal warning from VPRE

6.7.2.497 FS65_R_M_TWARN_PRE_NO_WARNING

```
#define FS65_R_M_TWARN_PRE_NO_WARNING (0x00U << FS65_R_M_TWARN_PRE_SHIFT)
```

No thermal warning (TJ < TWARN_PRE)

6.7.2.498 FS65_R_M_TWARN_PRE_SHIFT

```
#define FS65_R_M_TWARN_PRE_SHIFT 0x05U
```

Report a thermal warning from VPRE

6.7.2.499 FS65_R_M_TWARN_PRE_WARNING

```
#define FS65_R_M_TWARN_PRE_WARNING (0x01U << FS65_R_M_TWARN_PRE_SHIFT)
```

Thermal warning (TJ > TWARN_PRE)

6.7.2.500 FS65_R_M_TXD_DOM_FAILURE

```
#define FS65_R_M_TXD_DOM_FAILURE (0x01U << FS65_R_M_TXD_DOM_SHIFT)
```

Failure detected

6.7.2.501 FS65_R_M_TXD_DOM_MASK

```
#define FS65_R_M_TXD_DOM_MASK 0x01U
```

TXD dominant clamping detection (short-circuit to GND)

6.7.2.502 FS65_R_M_TXD_DOM_NO_FAILURE

```
#define FS65_R_M_TXD_DOM_NO_FAILURE (0x00U << FS65_R_M_TXD_DOM_SHIFT)
```

No failure

6.7.2.503 FS65_R_M_TXD_DOM_SHIFT

```
#define FS65_R_M_TXD_DOM_SHIFT 0x00U
```

TXD dominant clamping detection (short-circuit to GND)

6.7.2.504 FS65_R_M_V2P5_M_A_OV_MASK

```
#define FS65_R_M_V2P5_M_A_OV_MASK 0x08U
```

Report an overvoltage on V2P5 main analog regulator

6.7.2.505 FS65_R_M_V2P5_M_A_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_V2P5_M_A_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_V2P5_M_A_OV_SHIFT)
```

No overvoltage (V2P5_M_A < V2P5_M_A_OV)

6.7.2.506 FS65_R_M_V2P5_M_A_OV_OVERVOLTAGE

```
#define FS65_R_M_V2P5_M_A_OV_OVERVOLTAGE (0x01U << FS65_R_M_V2P5_M_A_OV_SHIFT)
```

Overvoltage detected (V2P5_M_A > V2P5_M_A_OV)

6.7.2.507 FS65_R_M_V2P5_M_A_OV_SHIFT

```
#define FS65_R_M_V2P5_M_A_OV_SHIFT 0x03U
```

Report an overvoltage on V2P5 main analog regulator

6.7.2.508 FS65_R_M_V2P5_M_D_OV_MASK

```
#define FS65_R_M_V2P5_M_D_OV_MASK 0x04U
```

Report an overvoltage on V2P5 main digital regulator

6.7.2.509 FS65_R_M_V2P5_M_D_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_V2P5_M_D_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_V2P5_M_D_OV_SHIFT)
```

No overvoltage (V2P5_M_D $<$ V2P5_M_D_OV)

6.7.2.510 FS65_R_M_V2P5_M_D_OV_OVERVOLTAGE

```
#define FS65_R_M_V2P5_M_D_OV_OVERVOLTAGE (0x01U << FS65_R_M_V2P5_M_D_OV_SHIFT)
```

Overvoltage detected (V2P5_M_D $>$ V2P5_M_D_OV)

6.7.2.511 FS65_R_M_V2P5_M_D_OV_SHIFT

```
#define FS65_R_M_V2P5_M_D_OV_SHIFT 0x02U
```

Report an overvoltage on V2P5 main digital regulator

6.7.2.512 FS65_R_M_VAUX_EN_DISABLED

```
#define FS65_R_M_VAUX_EN_DISABLED (0x00U << FS65_R_M_VAUX_EN_SHIFT)
```

Disabled

6.7.2.513 FS65_R_M_VAUX_EN_ENABLED

```
#define FS65_R_M_VAUX_EN_ENABLED (0x01U << FS65_R_M_VAUX_EN_SHIFT)
```

Enabled

6.7.2.514 FS65_R_M_VAUX_EN_MASK

```
#define FS65_R_M_VAUX_EN_MASK 0x02U
```

VAUX control (switch off not recommended if VAUX is safety critical)

6.7.2.515 FS65_R_M_VAUX_EN_SHIFT

```
#define FS65_R_M_VAUX_EN_SHIFT 0x01U
```

VAUX control (switch off not recommended if VAUX is safety critical)

6.7.2.516 FS65_R_M_VAUX_HW_3_3V

```
#define FS65_R_M_VAUX_HW_3_3V (0x01U << FS65_R_M_VAUX_HW_SHIFT)
```

3.3 V

6.7.2.517 FS65_R_M_VAUX_HW_5_0V

```
#define FS65_R_M_VAUX_HW_5_0V (0x00U << FS65_R_M_VAUX_HW_SHIFT)
```

5.0 V

6.7.2.518 FS65_R_M_VAUX_HW_MASK

```
#define FS65_R_M_VAUX_HW_MASK 0x08U
```

Report the hardware configuration for VAUX

6.7.2.519 FS65_R_M_VAUX_HW_SHIFT

```
#define FS65_R_M_VAUX_HW_SHIFT 0x03U
```

Report the hardware configuration for VAUX

6.7.2.520 FS65_R_M_VAUX_OV_MASK

```
#define FS65_R_M_VAUX_OV_MASK 0x08U
```

VAUX overvoltage detection

6.7.2.521 FS65_R_M_VAUX_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_VAUX_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VAUX_OV_SHIFT)
```

No overvoltage (VAUX $<$ VAUX_OV)

6.7.2.522 FS65_R_M_VAUX_OV_OVERVOLTAGE

```
#define FS65_R_M_VAUX_OV_OVERVOLTAGE (0x01U << FS65_R_M_VAUX_OV_SHIFT)
```

Overvoltage detected (VAUX $>$ VAUX_OV)

6.7.2.523 FS65_R_M_VAUX_OV_SHIFT

```
#define FS65_R_M_VAUX_OV_SHIFT 0x03U
```

VAUX overvoltage detection

6.7.2.524 FS65_R_M_VAUX_UV_MASK

```
#define FS65_R_M_VAUX_UV_MASK 0x04U
```

VAUX undervoltage detection

6.7.2.525 FS65_R_M_VAUX_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_VAUX_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VAUX_UV_SHIFT)
```

No undervoltage (VAUX $>$ VAUX_UV)

6.7.2.526 FS65_R_M_VAUX_UV_SHIFT

```
#define FS65_R_M_VAUX_UV_SHIFT 0x02U
```

VAUX undervoltage detection

6.7.2.527 FS65_R_M_VAUX_UV_UNDERVOLTAGE

```
#define FS65_R_M_VAUX_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VAUX_UV_SHIFT)
```

Undervoltage detected (VAUX $<$ VAUX_UV)

6.7.2.528 FS65_R_M_VCAN_EN_DISABLED

```
#define FS65_R_M_VCAN_EN_DISABLED (0x00U << FS65_R_M_VCAN_EN_SHIFT)
```

Disabled

6.7.2.529 FS65_R_M_VCAN_EN_ENABLED

```
#define FS65_R_M_VCAN_EN_ENABLED (0x01U << FS65_R_M_VCAN_EN_SHIFT)
```

Enabled

6.7.2.530 FS65_R_M_VCAN_EN_MASK

```
#define FS65_R_M_VCAN_EN_MASK 0x01U
```

VCAN control

6.7.2.531 FS65_R_M_VCAN_EN_SHIFT

```
#define FS65_R_M_VCAN_EN_SHIFT 0x00U
```

VCAN control

6.7.2.532 FS65_R_M_VCAN_OV_MASK

```
#define FS65_R_M_VCAN_OV_MASK 0x08U
```

VCAN overvoltage detection

6.7.2.533 FS65_R_M_VCAN_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_VCAN_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCAN_OV_SHIFT)
```

No overvoltage (VCAN $<$ VCAN_OV)

6.7.2.534 FS65_R_M_VCAN_OV_OVERVOLTAGE

```
#define FS65_R_M_VCAN_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCAN_OV_SHIFT)
```

Overvoltage detected (VCAN $>$ VCAN_OV)

6.7.2.535 FS65_R_M_VCAN_OV_SHIFT

```
#define FS65_R_M_VCAN_OV_SHIFT 0x03U
```

VCAN overvoltage detection

6.7.2.536 FS65_R_M_VCAN_UV_MASK

```
#define FS65_R_M_VCAN_UV_MASK 0x04U
```

VCAN undervoltage detection

6.7.2.537 FS65_R_M_VCAN_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_VCAN_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCAN_UV_SHIFT)
```

No undervoltage (VCAN $>$ VCAN_UV)

6.7.2.538 FS65_R_M_VCAN_UV_SHIFT

```
#define FS65_R_M_VCAN_UV_SHIFT 0x02U
```

VCAN undervoltage detection

6.7.2.539 FS65_R_M_VCAN_UV_UNDERVOLTAGE

```
#define FS65_R_M_VCAN_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCAN_UV_SHIFT)
```

Undervoltage detected (VCAN < VCAN_UV)

6.7.2.540 FS65_R_M_VCCA_EN_DISABLED

```
#define FS65_R_M_VCCA_EN_DISABLED (0x00U << FS65_R_M_VCCA_EN_SHIFT)
```

Disabled

6.7.2.541 FS65_R_M_VCCA_EN_ENABLED

```
#define FS65_R_M_VCCA_EN_ENABLED (0x01U << FS65_R_M_VCCA_EN_SHIFT)
```

Enabled

6.7.2.542 FS65_R_M_VCCA_EN_MASK

```
#define FS65_R_M_VCCA_EN_MASK 0x04U
```

VCCA control (switch off not recommended if VCCA is safety critical)

6.7.2.543 FS65_R_M_VCCA_EN_SHIFT

```
#define FS65_R_M_VCCA_EN_SHIFT 0x02U
```

VCCA control (switch off not recommended if VCCA is safety critical)

6.7.2.544 FS65_R_M_VCCA_HW_3_3V

```
#define FS65_R_M_VCCA_HW_3_3V (0x00U << FS65_R_M_VCCA_HW_SHIFT)
```

3.3 V

6.7.2.545 FS65_R_M_VCCA_HW_5_0V

```
#define FS65_R_M_VCCA_HW_5_0V (0x01U << FS65_R_M_VCCA_HW_SHIFT)
```

5.0 V

6.7.2.546 FS65_R_M_VCCA_HW_MASK

```
#define FS65_R_M_VCCA_HW_MASK 0x10U
```

Report the hardware configuration for VCCA

6.7.2.547 FS65_R_M_VCCA_HW_SHIFT

```
#define FS65_R_M_VCCA_HW_SHIFT 0x04U
```

Report the hardware configuration for VCCA

6.7.2.548 FS65_R_M_VCCA_OV_MASK

```
#define FS65_R_M_VCCA_OV_MASK 0x08U
```

VCCA overvoltage detection

6.7.2.549 FS65_R_M_VCCA_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_VCCA_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCCA_OV_SHIFT)
```

No overvoltage (VCCA $<$ VCCA_OV)

6.7.2.550 FS65_R_M_VCCA_OV_OVERVOLTAGE

```
#define FS65_R_M_VCCA_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCCA_OV_SHIFT)
```

Overvoltage detected (VCCA $>$ VCCA_OV)

6.7.2.551 FS65_R_M_VCCA_OV_SHIFT

```
#define FS65_R_M_VCCA_OV_SHIFT 0x03U
```

VCCA overvoltage detection

6.7.2.552 FS65_R_M_VCCA_PNP_DET_INT_MOSFET

```
#define FS65_R_M_VCCA_PNP_DET_INT_MOSFET (0x01U << FS65_R_M_VCCA_PNP_DET_SHIFT)
```

Internal MOSFET

6.7.2.553 FS65_R_M_VCCA_PNP_DET_MASK

```
#define FS65_R_M_VCCA_PNP_DET_MASK 0x20U
```

Report the connection of an external PNP on VCCA

6.7.2.554 FS65_R_M_VCCA_PNP_DET_PNP_CONNECTED

```
#define FS65_R_M_VCCA_PNP_DET_PNP_CONNECTED (0x00U << FS65_R_M_VCCA_PNP_DET_SHIFT)
```

External PNP connected

6.7.2.555 FS65_R_M_VCCA_PNP_DET_SHIFT

```
#define FS65_R_M_VCCA_PNP_DET_SHIFT 0x05U
```

Report the connection of an external PNP on VCCA

6.7.2.556 FS65_R_M_VCCA_UV_MASK

```
#define FS65_R_M_VCCA_UV_MASK 0x04U
```

VCCA undervoltage detection

6.7.2.557 FS65_R_M_VCCA_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_VCCA_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCCA_UV_SHIFT)
```

No undervoltage (VCCA $>$ VCCA_UV)

6.7.2.558 FS65_R_M_VCCA_UV_SHIFT

```
#define FS65_R_M_VCCA_UV_SHIFT 0x02U
```

VCCA undervoltage detection

6.7.2.559 FS65_R_M_VCCA_UV_UNDERVOLTAGE

```
#define FS65_R_M_VCCA_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCCA_UV_SHIFT)
```

Undervoltage detected (VCCA $<$ VCCA_UV)

6.7.2.560 FS65_R_M_VCORE_0_5A

```
#define FS65_R_M_VCORE_0_5A (0x02U << FS65_R_M_VCORE_SHIFT)
```

0.5 A

6.7.2.561 FS65_R_M_VCORE_0_8A

```
#define FS65_R_M_VCORE_0_8A (0x01U << FS65_R_M_VCORE_SHIFT)
```

0.8 A

6.7.2.562 FS65_R_M_VCORE_1_5A

```
#define FS65_R_M_VCORE_1_5A (0x00U << FS65_R_M_VCORE_SHIFT)
```

1.5 A

6.7.2.563 FS65_R_M_VCORE_2_2A

```
#define FS65_R_M_VCORE_2_2A (0x03U << FS65_R_M_VCORE_SHIFT)
```

2.2 A

6.7.2.564 FS65_R_M_VCORE_EN_DISABLED

```
#define FS65_R_M_VCORE_EN_DISABLED (0x00U << FS65_R_M_VCORE_EN_SHIFT)
```

Disabled

6.7.2.565 FS65_R_M_VCORE_EN_ENABLED

```
#define FS65_R_M_VCORE_EN_ENABLED (0x01U << FS65_R_M_VCORE_EN_SHIFT)
```

Enabled

6.7.2.566 FS65_R_M_VCORE_EN_MASK

```
#define FS65_R_M_VCORE_EN_MASK 0x08U
```

VCORE control (switch off not recommended if VCORE is safety critical)

6.7.2.567 FS65_R_M_VCORE_EN_SHIFT

```
#define FS65_R_M_VCORE_EN_SHIFT 0x03U
```

VCORE control (switch off not recommended if VCORE is safety critical)

6.7.2.568 FS65_R_M_VCORE_FB_OV_MASK

```
#define FS65_R_M_VCORE_FB_OV_MASK 0x08U
```

VCORE overvoltage detection

6.7.2.569 FS65_R_M_VCORE_FB_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_VCORE_FB_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VCORE_FB_OV_SHIFT)
```

No overvoltage (VCORE_FB $<$ VCORE_FB_OV)

6.7.2.570 FS65_R_M_VCORE_FB_OV_OVERVOLTAGE

```
#define FS65_R_M_VCORE_FB_OV_OVERVOLTAGE (0x01U << FS65_R_M_VCORE_FB_OV_SHIFT)
```

Overvoltage detected (VPRE $>$ VPRE_OV)

6.7.2.571 FS65_R_M_VCORE_FB_OV_SHIFT

```
#define FS65_R_M_VCORE_FB_OV_SHIFT 0x03U
```

VCORE overvoltage detection

6.7.2.572 FS65_R_M_VCORE_FB_UV_MASK

```
#define FS65_R_M_VCORE_FB_UV_MASK 0x04U
```

VCORE undervoltage detection

6.7.2.573 FS65_R_M_VCORE_FB_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_VCORE_FB_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VCORE_FB_UV_SHIFT)
```

No undervoltage (VCORE_FB $>$ VCORE_FB_UV)

6.7.2.574 FS65_R_M_VCORE_FB_UV_SHIFT

```
#define FS65_R_M_VCORE_FB_UV_SHIFT 0x02U
```

VCORE undervoltage detection

6.7.2.575 FS65_R_M_VCORE_FB_UV_UNDERVOLTAGE

```
#define FS65_R_M_VCORE_FB_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VCORE_FB_UV_SHIFT)
```

Undervoltage (VCORE_FB $<$ VCORE_FB_UV)

6.7.2.576 FS65_R_M_VCORE_MASK

```
#define FS65_R_M_VCORE_MASK 0xC0U
```

VCORE current capability

6.7.2.577 FS65_R_M_VCORE_SHIFT

```
#define FS65_R_M_VCORE_SHIFT 0x06U
```

VCORE current capability

6.7.2.578 FS65_R_M_VCORE_STATE_MASK

```
#define FS65_R_M_VCORE_STATE_MASK 0x40U
```

Report the activation state of VCORE SMPS

6.7.2.579 FS65_R_M_VCORE_STATE_OFF

```
#define FS65_R_M_VCORE_STATE_OFF (0x00U << FS65_R_M_VCORE_STATE_SHIFT)
```

SMPS off

6.7.2.580 FS65_R_M_VCORE_STATE_ON

```
#define FS65_R_M_VCORE_STATE_ON (0x01U << FS65_R_M_VCORE_STATE_SHIFT)
```

SMPS on

6.7.2.581 FS65_R_M_VCORE_STATE_SHIFT

```
#define FS65_R_M_VCORE_STATE_SHIFT 0x06U
```

Report the activation state of VCORE SMPS

6.7.2.582 FS65_R_M_VKAM_MASK

```
#define FS65_R_M_VKAM_MASK 0x08U
```

VKAM supply

6.7.2.583 FS65_R_M_VKAM_OFF

```
#define FS65_R_M_VKAM_OFF (0x00U << FS65_R_M_VKAM_SHIFT)
```

VKAM off by default

6.7.2.584 FS65_R_M_VKAM_ON

```
#define FS65_R_M_VKAM_ON (0x01U << FS65_R_M_VKAM_SHIFT)
```

VKAM on by default

6.7.2.585 FS65_R_M_VKAM_SHIFT

```
#define FS65_R_M_VKAM_SHIFT 0x03U
```

VKAM supply

6.7.2.586 FS65_R_M_VPRE_OV_MASK

```
#define FS65_R_M_VPRE_OV_MASK 0x08U
```

VPRE overvoltage detection

6.7.2.587 FS65_R_M_VPRE_OV_NO_OVERVOLTAGE

```
#define FS65_R_M_VPRE_OV_NO_OVERVOLTAGE (0x00U << FS65_R_M_VPRE_OV_SHIFT)
```

No overvoltage (VPRE $<$ VPRE_OV)

6.7.2.588 FS65_R_M_VPRE_OV_OVERVOLTAGE

```
#define FS65_R_M_VPRE_OV_OVERVOLTAGE (0x01U << FS65_R_M_VPRE_OV_SHIFT)
```

Overvoltage detected (VPRE $>$ VPRE_OV)

6.7.2.589 FS65_R_M_VPRE_OV_SHIFT

```
#define FS65_R_M_VPRE_OV_SHIFT 0x03U
```

VPRE overvoltage detection

6.7.2.590 FS65_R_M_VPRE_STATE_MASK

```
#define FS65_R_M_VPRE_STATE_MASK 0x40U
```

Report the activation state of VPRE SMPS

6.7.2.591 FS65_R_M_VPRE_STATE_OFF

```
#define FS65_R_M_VPRE_STATE_OFF (0x00U << FS65_R_M_VPRE_STATE_SHIFT)
```

SMPS off

6.7.2.592 FS65_R_M_VPRE_STATE_ON

```
#define FS65_R_M_VPRE_STATE_ON (0x01U << FS65_R_M_VPRE_STATE_SHIFT)
```

SMPS on

6.7.2.593 FS65_R_M_VPRE_STATE_SHIFT

```
#define FS65_R_M_VPRE_STATE_SHIFT 0x06U
```

Report the activation state of VPRE SMPS

6.7.2.594 FS65_R_M_VPRE_UV_MASK

```
#define FS65_R_M_VPRE_UV_MASK 0x04U
```

VPRE undervoltage detection

6.7.2.595 FS65_R_M_VPRE_UV_NO_UNDERVOLTAGE

```
#define FS65_R_M_VPRE_UV_NO_UNDERVOLTAGE (0x00U << FS65_R_M_VPRE_UV_SHIFT)
```

No undervoltage (VPRE > VPRE_UV)

6.7.2.596 FS65_R_M_VPRE_UV_SHIFT

```
#define FS65_R_M_VPRE_UV_SHIFT 0x02U
```

VPRE undervoltage detection

6.7.2.597 FS65_R_M_VPRE_UV_UNDERVOLTAGE

```
#define FS65_R_M_VPRE_UV_UNDERVOLTAGE (0x01U << FS65_R_M_VPRE_UV_SHIFT)
```

Undervoltage detected (VPRE < VPRE_UV)

6.7.2.598 FS65_R_M_VSNS_UV_MASK

```
#define FS65_R_M_VSNS_UV_MASK 0x80U
```

Detection of battery voltage below VSNS_UV

6.7.2.599 FS65_R_M_VSNS_UV_SHIFT

```
#define FS65_R_M_VSNS_UV_SHIFT 0x07U
```

Detection of battery voltage below VSNS_UV

6.7.2.600 FS65_R_M_VSNS_UV_VBAT_G

```
#define FS65_R_M_VSNS_UV_VBAT_G (0x00U << FS65_R_M_VSNS_UV_SHIFT)
```

VBAT > VSNS_UV

6.7.2.601 FS65_R_M_VSNS_UV_VBAT_L

```
#define FS65_R_M_VSNS_UV_VBAT_L (0x01U << FS65_R_M_VSNS_UV_SHIFT)
```

VBAT < VSNS_UV

6.7.2.602 FS65_R_M_VSUP_UV_7_MASK

```
#define FS65_R_M_VSUP_UV_7_MASK 0x40U
```

Detection of VSUP below VSUP_UV_7

6.7.2.603 FS65_R_M_VSUP_UV_7_SHIFT

```
#define FS65_R_M_VSUP_UV_7_SHIFT 0x06U
```

Detection of VSUP below VSUP_UV_7

6.7.2.604 FS65_R_M_VSUP_UV_7_VSUP_G

```
#define FS65_R_M_VSUP_UV_7_VSUP_G (0x00U << FS65_R_M_VSUP_UV_7_SHIFT)
```

VSUP $>$ VSUP_UV_7

6.7.2.605 FS65_R_M_VSUP_UV_7_VSUP_L

```
#define FS65_R_M_VSUP_UV_7_VSUP_L (0x01U << FS65_R_M_VSUP_UV_7_SHIFT)
```

VSUP $<$ VSUP_UV_7

6.7.2.606 FS65_R_M_WD_BAD_DATA_DATA_OK

```
#define FS65_R_M_WD_BAD_DATA_DATA_OK (0x00U << FS65_R_M_WD_BAD_DATA_SHIFT)
```

WD data refresh ok

6.7.2.607 FS65_R_M_WD_BAD_DATA_MASK

```
#define FS65_R_M_WD_BAD_DATA_MASK 0x20U
```

Report a watchdog data refresh error

6.7.2.608 FS65_R_M_WD_BAD_DATA_SHIFT

```
#define FS65_R_M_WD_BAD_DATA_SHIFT 0x05U
```

Report a watchdog data refresh error

6.7.2.609 FS65_R_M_WD_BAD_DATA_WRONG_DATA

```
#define FS65_R_M_WD_BAD_DATA_WRONG_DATA (0x01U << FS65_R_M_WD_BAD_DATA_SHIFT)
```

Wrong WD data refresh

6.7.2.610 FS65_R_M_WD_BAD_TIMING_MASK

```
#define FS65_R_M_WD_BAD_TIMING_MASK 0x04U
```

Report a watchdog timing refresh error

6.7.2.611 FS65_R_M_WD_BAD_TIMING_SHIFT

```
#define FS65_R_M_WD_BAD_TIMING_SHIFT 0x02U
```

Report a watchdog timing refresh error

6.7.2.612 FS65_R_M_WD_BAD_TIMING_TIMING_OK

```
#define FS65_R_M_WD_BAD_TIMING_TIMING_OK (0x00U << FS65_R_M_WD_BAD_TIMING_SHIFT)
```

WD timing refresh OK

6.7.2.613 FS65_R_M_WD_BAD_TIMING_WRONG_TIMING

```
#define FS65_R_M_WD_BAD_TIMING_WRONG_TIMING (0x01U << FS65_R_M_WD_BAD_TIMING_SHIFT)
```

Wrong WD timing refresh

6.7.2.614 FS65_R_M_WD_ERR_MASK

```
#define FS65_R_M_WD_ERR_MASK 0xE0U
```

Report the value of the watchdog error counter from 0 to 5 (6 generate an increase of the FLT_ERR_CNT and this counter is reset to 0)

6.7.2.615 FS65_R_M_WD_ERR_SHIFT

```
#define FS65_R_M_WD_ERR_SHIFT 0x05U
```

Report the value of the watchdog error counter from 0 to 5 (6 generate an increase of the FLT_ERR_CNT and this counter is reset to 0)

6.7.2.616 FS65_R_M_WD_RFR_MASK

```
#define FS65_R_M_WD_RFR_MASK 0x0EU
```

Report the value of the watchdog refresh counter from 0 to 6 (7 generate a decrease of the FLT_ERR_CNT and this counter is reset to 0)

6.7.2.617 FS65_R_M_WD_RFR_SHIFT

```
#define FS65_R_M_WD_RFR_SHIFT 0x01U
```

Report the value of the watchdog refresh counter from 0 to 6 (7 generate a decrease of the FLT_ERR_CNT and this counter is reset to 0)

6.7.2.618 FS65_RW_FS_WD_LFSR_MASK

```
#define FS65_RW_FS_WD_LFSR_MASK 0xFFU
```

WD 8 bits LFSR value. Used to write the seed at any time. Default value at start-up or after a power on reset: 0xB2 bit7:bit0: 10110010. Value Bit7:Bit0: 1111 1111 is prohibited. During a write command, MISO reports the previous register content.

6.7.2.619 FS65_RW_FS_WD_LFSR_SHIFT

```
#define FS65_RW_FS_WD_LFSR_SHIFT 0x00U
```

WD 8 bits LFSR value. Used to write the seed at any time. Default value at start-up or after a power on reset: 0xB2 bit7:bit0: 10110010. Value Bit7:Bit0: 1111 1111 is prohibited. During a write command, MISO reports the previous register content.

6.7.2.620 FS65_RW_M_AMUX_IO_0_T

```
#define FS65_RW_M_AMUX_IO_0_T (0x05U << FS65_RW_M_AMUX_SHIFT)
```

IO_0 tight range

6.7.2.621 FS65_RW_M_AMUX_IO_0_W

```
#define FS65_RW_M_AMUX_IO_0_W (0x02U << FS65_RW_M_AMUX_SHIFT)
```

IO_0 wide range

6.7.2.622 FS65_RW_M_AMUX_IO_5_T

```
#define FS65_RW_M_AMUX_IO_5_T (0x06U << FS65_RW_M_AMUX_SHIFT)
```

IO_5 tight range/VKAM

6.7.2.623 FS65_RW_M_AMUX_IO_5_W

```
#define FS65_RW_M_AMUX_IO_5_W (0x03U << FS65_RW_M_AMUX_SHIFT)
```

IO_5 wide range

6.7.2.624 FS65_RW_M_AMUX_MASK

```
#define FS65_RW_M_AMUX_MASK 0x07U
```

Select AMUX output

6.7.2.625 FS65_RW_M_AMUX_SHIFT

```
#define FS65_RW_M_AMUX_SHIFT 0x00U
```

Select AMUX output

6.7.2.626 FS65_RW_M_AMUX_TEMP_SENSOR

```
#define FS65_RW_M_AMUX_TEMP_SENSOR (0x07U << FS65_RW_M_AMUX_SHIFT)
```

Die Temperature Sensor

6.7.2.627 FS65_RW_M_AMUX_VREF

```
#define FS65_RW_M_AMUX_VREF (0x00U << FS65_RW_M_AMUX_SHIFT)
```

VREF

6.7.2.628 FS65_RW_M_AMUX_VSNS_T

```
#define FS65_RW_M_AMUX_VSNS_T (0x04U << FS65_RW_M_AMUX_SHIFT)
```

VSNS tight range

6.7.2.629 FS65_RW_M_AMUX_VSNS_W

```
#define FS65_RW_M_AMUX_VSNS_W (0x01U << FS65_RW_M_AMUX_SHIFT)
```

VSNS wide range

6.7.2.630 FS65_RW_M_CAN_AUTO_DIS_MASK

```
#define FS65_RW_M_CAN_AUTO_DIS_MASK 0x20U
```

Automatic CAN Tx disable

6.7.2.631 FS65_RW_M_CAN_AUTO_DIS_NO

```
#define FS65_RW_M_CAN_AUTO_DIS_NO (0x00U << FS65_RW_M_CAN_AUTO_DIS_SHIFT)
```

NO auto disable

6.7.2.632 FS65_RW_M_CAN_AUTO_DIS_RESET

```
#define FS65_RW_M_CAN_AUTO_DIS_RESET (0x01U << FS65_RW_M_CAN_AUTO_DIS_SHIFT)
```

Reset CAN_mode from 11 to 01 on CAN_OT or TXD_DOM or RXD_REC event

6.7.2.633 FS65_RW_M_CAN_AUTO_DIS_SHIFT

```
#define FS65_RW_M_CAN_AUTO_DIS_SHIFT 0x05U
```

Automatic CAN Tx disable

6.7.2.634 FS65_RW_M_CAN_DIS_CFG_MASK

```
#define FS65_RW_M_CAN_DIS_CFG_MASK 0x20U
```

Define CAN behavior when FS1B is asserted low

6.7.2.635 FS65_RW_M_CAN_DIS_CFG_RX_ONLY

```
#define FS65_RW_M_CAN_DIS_CFG_RX_ONLY (0x00U << FS65_RW_M_CAN_DIS_CFG_SHIFT)
```

CAN in Rx only mode (when FS1B_CAN_ IMPACT = 1 in INIT_FAULT register)

6.7.2.636 FS65_RW_M_CAN_DIS_CFG_SHIFT

```
#define FS65_RW_M_CAN_DIS_CFG_SHIFT 0x05U
```

Define CAN behavior when FS1B is asserted low

6.7.2.637 FS65_RW_M_CAN_DIS_CFG_SLEEP

```
#define FS65_RW_M_CAN_DIS_CFG_SLEEP (0x01U << FS65_RW_M_CAN_DIS_CFG_SHIFT)
```

CAN in sleep mode (when FS1B_CAN_ IMPACT = 1 in INIT_FAULT register)

6.7.2.638 FS65_RW_M_CAN_MODE_LISTEN_ONLY

```
#define FS65_RW_M_CAN_MODE_LISTEN_ONLY (0x01U << FS65_RW_M_CAN_MODE_SHIFT)
```

Listen only

6.7.2.639 FS65_RW_M_CAN_MODE_MASK

```
#define FS65_RW_M_CAN_MODE_MASK 0xC0U
```

Configure the CAN mode

6.7.2.640 FS65_RW_M_CAN_MODE_NORMAL

```
#define FS65_RW_M_CAN_MODE_NORMAL (0x03U << FS65_RW_M_CAN_MODE_SHIFT)
```

Normal operation mode

6.7.2.641 FS65_RW_M_CAN_MODE_SHIFT

```
#define FS65_RW_M_CAN_MODE_SHIFT 0x06U
```

Configure the CAN mode

6.7.2.642 FS65_RW_M_CAN_MODE_SL_WU

```
#define FS65_RW_M_CAN_MODE_SL_WU (0x02U << FS65_RW_M_CAN_MODE_SHIFT)
```

Sleep/wake-up capability

6.7.2.643 FS65_RW_M_CAN_MODE_SLN_WU

```
#define FS65_RW_M_CAN_MODE_SLN_WU (0x00U << FS65_RW_M_CAN_MODE_SHIFT)
```

Sleep/no wake-up capability

6.7.2.644 FS65_RW_M_CAN_WU_TO_120US

```
#define FS65_RW_M_CAN_WU_TO_120US (0x00U << FS65_RW_M_CAN_WU_TO_SHIFT)
```

120 us

6.7.2.645 FS65_RW_M_CAN_WU_TO_2_8MS

```
#define FS65_RW_M_CAN_WU_TO_2_8MS (0x01U << FS65_RW_M_CAN_WU_TO_SHIFT)
```

2.8 ms

6.7.2.646 FS65_RW_M_CAN_WU_TO_MASK

```
#define FS65_RW_M_CAN_WU_TO_MASK 0x10U
```

Define the CAN wake-up timeout (when CAN_WU_CONF = 0)

6.7.2.647 FS65_RW_M_CAN_WU_TO_SHIFT

```
#define FS65_RW_M_CAN_WU_TO_SHIFT 0x04U
```

Define the CAN wake-up timeout (when CAN_WU_CONF = 0)

6.7.2.648 FS65_RW_M_F2_F0_FUNCTION1

```
#define FS65_RW_M_F2_F0_FUNCTION1 (0x00U << FS65_RW_M_F2_F0_SHIFT)
```

In normal mode count and generate flag or INT when counter reaches the after run value

6.7.2.649 FS65_RW_M_F2_F0_FUNCTION2

```
#define FS65_RW_M_F2_F0_FUNCTION2 (0x01U << FS65_RW_M_F2_F0_SHIFT)
```

In normal mode count until after run value is reached, then enters in LPOFF

6.7.2.650 FS65_RW_M_F2_F0_FUNCTION3

```
#define FS65_RW_M_F2_F0_FUNCTION3 (0x02U << FS65_RW_M_F2_F0_SHIFT)
```

In normal mode count until after run value is reached, then enters in LPOFF. Once in LPOFF, count until wake-up value is reached and wake-up

6.7.2.651 FS65_RW_M_F2_F0_FUNCTION4

```
#define FS65_RW_M_F2_F0_FUNCTION4 (0x03U << FS65_RW_M_F2_F0_SHIFT)
```

In LPOFF, count until wake-up value is reached and wake-up

6.7.2.652 FS65_RW_M_F2_F0_FUNCTION5

```
#define FS65_RW_M_F2_F0_FUNCTION5 (0x04U << FS65_RW_M_F2_F0_SHIFT)
```

In LPOFF, count and do not wake-up. Counter value is stored in wake-up register

6.7.2.653 FS65_RW_M_F2_F0_MASK

```
#define FS65_RW_M_F2_F0_MASK 0xE0U
```

Select timer operating function

6.7.2.654 FS65_RW_M_F2_F0_SHIFT

```
#define FS65_RW_M_F2_F0_SHIFT 0x05U
```

Select timer operating function

6.7.2.655 FS65_RW_M_ICCA_LIM_ICCA_LIM_INT

```
#define FS65_RW_M_ICCA_LIM_ICCA_LIM_INT (0x01U << FS65_RW_M_ICCA_LIM_SHIFT)
```

ICCA_LIM_INT

6.7.2.656 FS65_RW_M_ICCA_LIM_ICCA_LIM_OUT

```
#define FS65_RW_M_ICCA_LIM_ICCA_LIM_OUT (0x00U << FS65_RW_M_ICCA_LIM_SHIFT)
```

ICCA_LIM_OUT

6.7.2.657 FS65_RW_M_ICCA_LIM_MASK

```
#define FS65_RW_M_ICCA_LIM_MASK 0x80U
```

Configure the current limitation threshold for VCCA. Only available for external PNP.

6.7.2.658 FS65_RW_M_ICCA_LIM_SHIFT

```
#define FS65_RW_M_ICCA_LIM_SHIFT 0x07U
```

Configure the current limitation threshold for VCCA. Only available for external PNP.

6.7.2.659 FS65_RW_M_INT_INH_0_MASK

```
#define FS65_RW_M_INT_INH_0_MASK 0x01U
```

Inhibit the interrupt pulse for IO_0 (masked in IO_G)

6.7.2.660 FS65_RW_M_INT_INH_0_MASKED

```
#define FS65_RW_M_INT_INH_0_MASKED (0x01U << FS65_RW_M_INT_INH_0_SHIFT)
```

INT masked

6.7.2.661 FS65_RW_M_INT_INH_0_NOT_MASKED

```
#define FS65_RW_M_INT_INH_0_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_0_SHIFT)
```

INT not masked

6.7.2.662 FS65_RW_M_INT_INH_0_SHIFT

```
#define FS65_RW_M_INT_INH_0_SHIFT 0x00U
```

Inhibit the interrupt pulse for IO_0 (masked in IO_G)

6.7.2.663 FS65_RW_M_INT_INH_2_MASK

```
#define FS65_RW_M_INT_INH_2_MASK 0x02U
```

Inhibit the interrupt pulse for IO_2 (masked in IO_G)

6.7.2.664 FS65_RW_M_INT_INH_2_MASKED

```
#define FS65_RW_M_INT_INH_2_MASKED (0x01U << FS65_RW_M_INT_INH_2_SHIFT)
```

INT masked

6.7.2.665 FS65_RW_M_INT_INH_2_NOT_MASKED

```
#define FS65_RW_M_INT_INH_2_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_2_SHIFT)
```

INT not masked

6.7.2.666 FS65_RW_M_INT_INH_2_SHIFT

```
#define FS65_RW_M_INT_INH_2_SHIFT 0x01U
```

Inhibit the interrupt pulse for IO_2 (masked in IO_G)

6.7.2.667 FS65_RW_M_INT_INH_3_MASK

```
#define FS65_RW_M_INT_INH_3_MASK 0x04U
```

Inhibit the interrupt pulse for IO_3 (masked in IO_G)

6.7.2.668 FS65_RW_M_INT_INH_3_MASKED

```
#define FS65_RW_M_INT_INH_3_MASKED (0x01U << FS65_RW_M_INT_INH_3_SHIFT)
```

INT masked

6.7.2.669 FS65_RW_M_INT_INH_3_NOT_MASKED

```
#define FS65_RW_M_INT_INH_3_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_3_SHIFT)
```

INT not masked

6.7.2.670 FS65_RW_M_INT_INH_3_SHIFT

```
#define FS65_RW_M_INT_INH_3_SHIFT 0x02U
```

Inhibit the interrupt pulse for IO_3 (masked in IO_G)

6.7.2.671 FS65_RW_M_INT_INH_4_MASK

```
#define FS65_RW_M_INT_INH_4_MASK 0x08U
```

Inhibit the interrupt pulse for IO_4 (masked in IO_G)

6.7.2.672 FS65_RW_M_INT_INH_4_MASKED

```
#define FS65_RW_M_INT_INH_4_MASKED (0x01U << FS65_RW_M_INT_INH_4_SHIFT)
```

INT masked

6.7.2.673 FS65_RW_M_INT_INH_4_NOT_MASKED

```
#define FS65_RW_M_INT_INH_4_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_4_SHIFT)
```

INT not masked

6.7.2.674 FS65_RW_M_INT_INH_4_SHIFT

```
#define FS65_RW_M_INT_INH_4_SHIFT 0x03U
```

Inhibit the interrupt pulse for IO_4 (masked in IO_G)

6.7.2.675 FS65_RW_M_INT_INH_5_MASK

```
#define FS65_RW_M_INT_INH_5_MASK 0x10U
```

Inhibit the interrupt pulse for IO_5 (masked in IO_G)

6.7.2.676 FS65_RW_M_INT_INH_5_MASKED

```
#define FS65_RW_M_INT_INH_5_MASKED (0x01U << FS65_RW_M_INT_INH_5_SHIFT)
```

INT masked

6.7.2.677 FS65_RW_M_INT_INH_5_NOT_MASKED

```
#define FS65_RW_M_INT_INH_5_NOT_MASKED (0x00U << FS65_RW_M_INT_INH_5_SHIFT)
```

INT not masked

6.7.2.678 FS65_RW_M_INT_INH_5_SHIFT

```
#define FS65_RW_M_INT_INH_5_SHIFT 0x04U
```

Inhibit the interrupt pulse for IO_5 (masked in IO_G)

6.7.2.679 FS65_RW_M_INT_INH_ALL_ALL_INHIBITED

```
#define FS65_RW_M_INT_INH_ALL_ALL_INHIBITED (0x01U << FS65_RW_M_INT_INH_ALL_SHIFT)
```

All INT inhibited

6.7.2.680 FS65_RW_M_INT_INH_ALL_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_ALL_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_ALL_SHIFT)
```

All INT sources

6.7.2.681 FS65_RW_M_INT_INH_ALL_MASK

```
#define FS65_RW_M_INT_INH_ALL_MASK 0x20U
```

Inhibit ALL the interrupt

6.7.2.682 FS65_RW_M_INT_INH_ALL_SHIFT

```
#define FS65_RW_M_INT_INH_ALL_SHIFT 0x05U
```

Inhibit ALL the interrupt

6.7.2.683 FS65_RW_M_INT_INH_CAN_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_CAN_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_CAN_SHIFT)
```

All INT sources

6.7.2.684 FS65_RW_M_INT_INH_CAN_CAN_INHIBITED

```
#define FS65_RW_M_INT_INH_CAN_CAN_INHIBITED (0x01U << FS65_RW_M_INT_INH_CAN_SHIFT)
```

CAN error bits change inhibited

6.7.2.685 FS65_RW_M_INT_INH_CAN_MASK

```
#define FS65_RW_M_INT_INH_CAN_MASK 0x01U
```

Inhibit the interrupt for CAN error bits

6.7.2.686 FS65_RW_M_INT_INH_CAN_SHIFT

```
#define FS65_RW_M_INT_INH_CAN_SHIFT 0x00U
```

Inhibit the interrupt for CAN error bits

6.7.2.687 FS65_RW_M_INT_INH_LIN_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_LIN_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_LIN_SHIFT)
```

All INT sources

6.7.2.688 FS65_RW_M_INT_INH_LIN_LIN_INHIBITED

```
#define FS65_RW_M_INT_INH_LIN_LIN_INHIBITED (0x01U << FS65_RW_M_INT_INH_LIN_SHIFT)
```

LIN error bits change INHIBITED

6.7.2.689 FS65_RW_M_INT_INH_LIN_MASK

```
#define FS65_RW_M_INT_INH_LIN_MASK 0x40U
```

Inhibit the interrupt for LIN error bits

6.7.2.690 FS65_RW_M_INT_INH_LIN_SHIFT

```
#define FS65_RW_M_INT_INH_LIN_SHIFT 0x06U
```

Inhibit the interrupt for LIN error bits

6.7.2.691 FS65_RW_M_INT_INH_VCORE_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_VCORE_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VCORE_SHIFT)
```

All INT sources

6.7.2.692 FS65_RW_M_INT_INH_VCORE_MASK

```
#define FS65_RW_M_INT_INH_VCORE_MASK 0x04U
```

Inhibit the interrupt for VCORE status event

6.7.2.693 FS65_RW_M_INT_INH_VCORE_SHIFT

```
#define FS65_RW_M_INT_INH_VCORE_SHIFT 0x02U
```

Inhibit the interrupt for VCORE status event

6.7.2.694 FS65_RW_M_INT_INH_VCORE_VCORE_INHIBITED

```
#define FS65_RW_M_INT_INH_VCORE_VCORE_INHIBITED (0x01U << FS65_RW_M_INT_INH_VCORE_SHIFT)
```

VCORE status change inhibited

6.7.2.695 FS65_RW_M_INT_INH_VOTHER_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_VOTHER_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VOTHER_SHIFT)
```

All INT sources

6.7.2.696 FS65_RW_M_INT_INH_VOTHER_MASK

```
#define FS65_RW_M_INT_INH_VOTHER_MASK 0x02U
```

Inhibit the interrupt for VCCA/VAUX and VCAN status event

6.7.2.697 FS65_RW_M_INT_INH_VOTHER_SHIFT

```
#define FS65_RW_M_INT_INH_VOTHER_SHIFT 0x01U
```

Inhibit the interrupt for VCCA/VAUX and VCAN status event

6.7.2.698 FS65_RW_M_INT_INH_VOTHER_VOTHER_INHIBITED

```
#define FS65_RW_M_INT_INH_VOTHER_VOTHER_INHIBITED (0x01U << FS65_RW_M_INT_INH_VOTHER_SHIFT)
```

VCCA/VAUX/VCAN status change inhibited

6.7.2.699 FS65_RW_M_INT_INH_VPRE_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_VPRE_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VPRE_SHIFT)
```

All INT sources

6.7.2.700 FS65_RW_M_INT_INH_VPRE_MASK

```
#define FS65_RW_M_INT_INH_VPRE_MASK 0x08U
```

Inhibit the interrupt for VPRE status event

6.7.2.701 FS65_RW_M_INT_INH_VPRE_SHIFT

```
#define FS65_RW_M_INT_INH_VPRE_SHIFT 0x03U
```

Inhibit the interrupt for VPRE status event

6.7.2.702 FS65_RW_M_INT_INH_VPRE_VPRE_INHIBITED

```
#define FS65_RW_M_INT_INH_VPRE_VPRE_INHIBITED (0x01U << FS65_RW_M_INT_INH_VPRE_SHIFT)
```

VPRE status change inhibited

6.7.2.703 FS65_RW_M_INT_INH_VSNS_ALL_SOURCES

```
#define FS65_RW_M_INT_INH_VSNS_ALL_SOURCES (0x00U << FS65_RW_M_INT_INH_VSNS_SHIFT)
```

All INT sources

6.7.2.704 FS65_RW_M_INT_INH_VSNS_MASK

```
#define FS65_RW_M_INT_INH_VSNS_MASK 0x10U
```

Inhibit the interrupt for VSNS_UV

6.7.2.705 FS65_RW_M_INT_INH_VSNS_SHIFT

```
#define FS65_RW_M_INT_INH_VSNS_SHIFT 0x04U
```

Inhibit the interrupt for VSNS_UV

6.7.2.706 FS65_RW_M_INT_INH_VSNS_VSNS_UV_INHIBITED

```
#define FS65_RW_M_INT_INH_VSNS_VSNS_UV_INHIBITED (0x01U << FS65_RW_M_INT_INH_VSNS_SHIFT)
```

VSNS_UV INT inhibited

6.7.2.707 FS65_RW_M_IO_OUT_4_EN_ENABLED

```
#define FS65_RW_M_IO_OUT_4_EN_ENABLED (0x01U << FS65_RW_M_IO_OUT_4_EN_SHIFT)
```

Enabled (IO_4 configured as output gate driver)

6.7.2.708 FS65_RW_M_IO_OUT_4_EN_MASK

```
#define FS65_RW_M_IO_OUT_4_EN_MASK 0x80U
```

Enable the output gate driver capability for IO_4

6.7.2.709 FS65_RW_M_IO_OUT_4_EN_SHIFT

```
#define FS65_RW_M_IO_OUT_4_EN_SHIFT 0x07U
```

Enable the output gate driver capability for IO_4

6.7.2.710 FS65_RW_M_IO_OUT_4_EN_Z

```
#define FS65_RW_M_IO_OUT_4_EN_Z (0x00U << FS65_RW_M_IO_OUT_4_EN_SHIFT)
```

High-impedance (IO_4 configured as input)

6.7.2.711 FS65_RW_M_IO_OUT_4_HIGH

```
#define FS65_RW_M_IO_OUT_4_HIGH (0x01U << FS65_RW_M_IO_OUT_4_SHIFT)
```

High

6.7.2.712 FS65_RW_M_IO_OUT_4_LOW

```
#define FS65_RW_M_IO_OUT_4_LOW (0x00U << FS65_RW_M_IO_OUT_4_SHIFT)
```

Low

6.7.2.713 FS65_RW_M_IO_OUT_4_MASK

```
#define FS65_RW_M_IO_OUT_4_MASK 0x40U
```

Configure IO_4 output gate driver state

6.7.2.714 FS65_RW_M_IO_OUT_4_SHIFT

```
#define FS65_RW_M_IO_OUT_4_SHIFT 0x06U
```

Configure IO_4 output gate driver state

6.7.2.715 FS65_RW_M_IPFF_DIS_DISABLED

```
#define FS65_RW_M_IPFF_DIS_DISABLED (0x01U << FS65_RW_M_IPFF_DIS_SHIFT)
```

Disabled

6.7.2.716 FS65_RW_M_IPFF_DIS_ENABLED

```
#define FS65_RW_M_IPFF_DIS_ENABLED (0x00U << FS65_RW_M_IPFF_DIS_SHIFT)
```

Enabled

6.7.2.717 FS65_RW_M_IPFF_DIS_MASK

```
#define FS65_RW_M_IPFF_DIS_MASK 0x20U
```

DISABLE the input power feed forward (IPFF) function of VPRE

6.7.2.718 FS65_RW_M_IPFF_DIS_SHIFT

```
#define FS65_RW_M_IPFF_DIS_SHIFT 0x05U
```

DISABLE the input power feed forward (IPFF) function of VPRE

6.7.2.719 FS65_RW_M_LDT_ENABLE_MASK

```
#define FS65_RW_M_LDT_ENABLE_MASK 0x02U
```

LDT counter control

6.7.2.720 FS65_RW_M_LDT_ENABLE_SHIFT

```
#define FS65_RW_M_LDT_ENABLE_SHIFT 0x01U
```

LDT counter control

6.7.2.721 FS65_RW_M_LDT_ENABLE_START

```
#define FS65_RW_M_LDT_ENABLE_START (0x01U << FS65_RW_M_LDT_ENABLE_SHIFT)
```

LDT counter start

6.7.2.722 FS65_RW_M_LDT_ENABLE_STOP

```
#define FS65_RW_M_LDT_ENABLE_STOP (0x00U << FS65_RW_M_LDT_ENABLE_SHIFT)
```

LDT counter stop

6.7.2.723 FS65_RW_M_LIN_AUTO_DIS_MASK

```
#define FS65_RW_M_LIN_AUTO_DIS_MASK 0x04U
```

Automatic LIN mode disable

6.7.2.724 FS65_RW_M_LIN_AUTO_DIS_NO

```
#define FS65_RW_M_LIN_AUTO_DIS_NO (0x00U << FS65_RW_M_LIN_AUTO_DIS_SHIFT)
```

No auto disable

6.7.2.725 FS65_RW_M_LIN_AUTO_DIS_RESET

```
#define FS65_RW_M_LIN_AUTO_DIS_RESET (0x01U << FS65_RW_M_LIN_AUTO_DIS_SHIFT)
```

Reset LIN_mode from 11 to 01 on LIN_OT or TXDL_DOM or RXDL_REC event

6.7.2.726 FS65_RW_M_LIN_AUTO_DIS_SHIFT

```
#define FS65_RW_M_LIN_AUTO_DIS_SHIFT 0x02U
```

Automatic LIN mode disable

6.7.2.727 FS65_RW_M_LIN_J2602_DIS_COMPLIANT

```
#define FS65_RW_M_LIN_J2602_DIS_COMPLIANT (0x00U << FS65_RW_M_LIN_J2602_DIS_SHIFT)
```

Compliant with J2602 standard

6.7.2.728 FS65_RW_M_LIN_J2602_DIS_MASK

```
#define FS65_RW_M_LIN_J2602_DIS_MASK 0x04U
```

To comply with J2602 standard. Recessive mode when VSUP $<$ 7.0 V

6.7.2.729 FS65_RW_M_LIN_J2602_DIS_NOT_COMPLIANT

```
#define FS65_RW_M_LIN_J2602_DIS_NOT_COMPLIANT (0x01U << FS65_RW_M_LIN_J2602_DIS_SHIFT)
```

Not compliant with J2602 standard

6.7.2.730 FS65_RW_M_LIN_J2602_DIS_SHIFT

```
#define FS65_RW_M_LIN_J2602_DIS_SHIFT 0x02U
```

To comply with J2602 standard. Recessive mode when VSUP $<$ 7.0 V

6.7.2.731 FS65_RW_M_LIN_MODE_LISTEN_ONLY

```
#define FS65_RW_M_LIN_MODE_LISTEN_ONLY (0x01U << FS65_RW_M_LIN_MODE_SHIFT)
```

Listen only

6.7.2.732 FS65_RW_M_LIN_MODE_MASK

```
#define FS65_RW_M_LIN_MODE_MASK 0x18U
```

Configure the LIN mode

6.7.2.733 FS65_RW_M_LIN_MODE_NORMAL

```
#define FS65_RW_M_LIN_MODE_NORMAL (0x03U << FS65_RW_M_LIN_MODE_SHIFT)
```

Normal operation mode

6.7.2.734 FS65_RW_M_LIN_MODE_SHIFT

```
#define FS65_RW_M_LIN_MODE_SHIFT 0x03U
```

Configure the LIN mode

6.7.2.735 FS65_RW_M_LIN_MODE_SL_WU

```
#define FS65_RW_M_LIN_MODE_SL_WU (0x02U << FS65_RW_M_LIN_MODE_SHIFT)
```

Sleep/wake-up capability

6.7.2.736 FS65_RW_M_LIN_MODE_SLN_WU

```
#define FS65_RW_M_LIN_MODE_SLN_WU (0x00U << FS65_RW_M_LIN_MODE_SHIFT)
```

Sleep/no wake-up capability

6.7.2.737 FS65_RW_M_LIN_SR_10KBITS

```
#define FS65_RW_M_LIN_SR_10KBITS (0x01U << FS65_RW_M_LIN_SR_SHIFT)
```

10 kbits/s

6.7.2.738 FS65_RW_M_LIN_SR_20KBITS

```
#define FS65_RW_M_LIN_SR_20KBITS (0x00U << FS65_RW_M_LIN_SR_SHIFT)
```

20 kbits/s

6.7.2.739 FS65_RW_M_LIN_SR_FAST_RATE

```
#define FS65_RW_M_LIN_SR_FAST_RATE (0x02U << FS65_RW_M_LIN_SR_SHIFT)
```

Fast baud rate (Max: 100 kbits/s)

6.7.2.740 FS65_RW_M_LIN_SR_MASK

```
#define FS65_RW_M_LIN_SR_MASK 0x03U
```

Configure the LIN slew rate

6.7.2.741 FS65_RW_M_LIN_SR_SHIFT

```
#define FS65_RW_M_LIN_SR_SHIFT 0x00U
```

Configure the LIN slew rate

6.7.2.742 FS65_RW_M_MODE_CALIBRATION

```
#define FS65_RW_M_MODE_CALIBRATION (0x00U << FS65_RW_M_MODE_SHIFT)
```

Calibration mode (488 us resolution)

6.7.2.743 FS65_RW_M_MODE_MASK

```
#define FS65_RW_M_MODE_MASK 0x04U
```

Operating mode selection

6.7.2.744 FS65_RW_M_MODE_NORMAL

```
#define FS65_RW_M_MODE_NORMAL (0x01U << FS65_RW_M_MODE_SHIFT)
```

Normal mode (1 s resolution)

6.7.2.745 FS65_RW_M_MODE_SHIFT

```
#define FS65_RW_M_MODE_SHIFT 0x02U
```

Operating mode selection

6.7.2.746 FS65_RW_M_NT_DURATION_100US

```
#define FS65_RW_M_NT_DURATION_100US (0x00U << FS65_RW_M_NT_DURATION_SHIFT)
```

100 us

6.7.2.747 FS65_RW_M_NT_DURATION_25US

```
#define FS65_RW_M_NT_DURATION_25US (0x01U << FS65_RW_M_NT_DURATION_SHIFT)
```

25 us

6.7.2.748 FS65_RW_M_NT_DURATION_MASK

```
#define FS65_RW_M_NT_DURATION_MASK 0x80U
```

Define the duration of the interrupt pulse

6.7.2.749 FS65_RW_M_NT_DURATION_SHIFT

```
#define FS65_RW_M_NT_DURATION_SHIFT 0x07U
```

Define the duration of the interrupt pulse

6.7.2.750 FS65_RW_M_REG_SE_MASK

```
#define FS65_RW_M_REG_SE_MASK 0x10U
```

Counter register selection

6.7.2.751 FS65_RW_M_REG_SE_PROGRAMMED_REG

```
#define FS65_RW_M_REG_SE_PROGRAMMED_REG (0x00U << FS65_RW_M_REG_SE_SHIFT)
```

Read programmed wake-up register

6.7.2.752 FS65_RW_M_REG_SE_RTC_REG

```
#define FS65_RW_M_REG_SE_RTC_REG (0x01U << FS65_RW_M_REG_SE_SHIFT)
```

Read real time counter into wake-up register (after counter is stopped with LDT_ENABLE bit)

6.7.2.753 FS65_RW_M_REG_SE_SHIFT

```
#define FS65_RW_M_REG_SE_SHIFT 0x04U
```

Counter register selection

6.7.2.754 FS65_RW_M_TAUX_LIM_OFF_10_MS

```
#define FS65_RW_M_TAUX_LIM_OFF_10_MS (0x00U << FS65_RW_M_TAUX_LIM_OFF_SHIFT)
```

10 ms

6.7.2.755 FS65_RW_M_TAUX_LIM_OFF_50_MS

```
#define FS65_RW_M_TAUX_LIM_OFF_50_MS (0x01U << FS65_RW_M_TAUX_LIM_OFF_SHIFT)
```

50 ms

6.7.2.756 FS65_RW_M_TAUX_LIM_OFF_MASK

```
#define FS65_RW_M_TAUX_LIM_OFF_MASK 0x04U
```

Configure the current limitation duration before VAUX is switched off.

6.7.2.757 FS65_RW_M_TAUX_LIM_OFF_SHIFT

```
#define FS65_RW_M_TAUX_LIM_OFF_SHIFT 0x02U
```

Configure the current limitation duration before VAUX is switched off.

6.7.2.758 FS65_RW_M_TCCA_LIM_OFF_10_MS

```
#define FS65_RW_M_TCCA_LIM_OFF_10_MS (0x00U << FS65_RW_M_TCCA_LIM_OFF_SHIFT)
```

10 ms

6.7.2.759 FS65_RW_M_TCCA_LIM_OFF_50_MS

```
#define FS65_RW_M_TCCA_LIM_OFF_50_MS (0x01U << FS65_RW_M_TCCA_LIM_OFF_SHIFT)
```

50 ms

6.7.2.760 FS65_RW_M_TCCA_LIM_OFF_MASK

```
#define FS65_RW_M_TCCA_LIM_OFF_MASK 0x40U
```

Configure the current limitation duration before VCCA is switched off. Only available for external PNP.

6.7.2.761 FS65_RW_M_TCCA_LIM_OFF_SHIFT

```
#define FS65_RW_M_TCCA_LIM_OFF_SHIFT 0x06U
```

Configure the current limitation duration before VCCA is switched off. Only available for external PNP.

6.7.2.762 FS65_RW_M_VAUX_TRK_EN_MASK

```
#define FS65_RW_M_VAUX_TRK_EN_MASK 0x02U
```

Configure VAUX regulator as a tracker of VCCA

6.7.2.763 FS65_RW_M_VAUX_TRK_EN_NO_TRACKING

```
#define FS65_RW_M_VAUX_TRK_EN_NO_TRACKING (0x00U << FS65_RW_M_VAUX_TRK_EN_SHIFT)
```

NO tracking

6.7.2.764 FS65_RW_M_VAUX_TRK_EN_SHIFT

```
#define FS65_RW_M_VAUX_TRK_EN_SHIFT 0x01U
```

Configure VAUX regulator as a tracker of VCCA

6.7.2.765 FS65_RW_M_VAUX_TRK_EN_TRACKING

```
#define FS65_RW_M_VAUX_TRK_EN_TRACKING (0x01U << FS65_RW_M_VAUX_TRK_EN_SHIFT)
```

Tracking mode enabled and latched

6.7.2.766 FS65_RW_M_VCAN_OV_MON_MASK

```
#define FS65_RW_M_VCAN_OV_MON_MASK 0x10U
```

CAN_5V overvoltage monitoring

6.7.2.767 FS65_RW_M_VCAN_OV_MON_OFF

```
#define FS65_RW_M_VCAN_OV_MON_OFF (0x00U << FS65_RW_M_VCAN_OV_MON_SHIFT)
```

Off. VCAN OV is not monitored. Flag is ignored.

6.7.2.768 FS65_RW_M_VCAN_OV_MON_ON

```
#define FS65_RW_M_VCAN_OV_MON_ON (0x01U << FS65_RW_M_VCAN_OV_MON_SHIFT)
```

On. VCAN OV is monitored. If OV the CAN_5V regulator is switched off.

6.7.2.769 FS65_RW_M_VCAN_OV_MON_SHIFT

```
#define FS65_RW_M_VCAN_OV_MON_SHIFT 0x04U
```

CAN_5V overvoltage monitoring

6.7.2.770 FS65_RW_M_VKAM_EN_DISABLED

```
#define FS65_RW_M_VKAM_EN_DISABLED (0x00U << FS65_RW_M_VKAM_EN_SHIFT)
```

DISABLED

6.7.2.771 FS65_RW_M_VKAM_EN_ENABLED

```
#define FS65_RW_M_VKAM_EN_ENABLED (0x01U << FS65_RW_M_VKAM_EN_SHIFT)
```

ENABLED

6.7.2.772 FS65_RW_M_VKAM_EN_MASK

```
#define FS65_RW_M_VKAM_EN_MASK 0x80U
```

VKAM control (default state depends on part number)

6.7.2.773 FS65_RW_M_VKAM_EN_SHIFT

```
#define FS65_RW_M_VKAM_EN_SHIFT 0x07U
```

VKAM control (default state depends on part number)

6.7.2.774 FS65_RW_M_WU_IO0_ANY_EDGE

```
#define FS65_RW_M_WU_IO0_ANY_EDGE (0x03U << FS65_RW_M_WU_IO0_SHIFT)
```

Wake-up on any edge

6.7.2.775 FS65_RW_M_WU_IO0_FALLING_EDGE

```
#define FS65_RW_M_WU_IO0_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO0_SHIFT)
```

Wake-up on falling edge - or low level

6.7.2.776 FS65_RW_M_WU_IO0_MASK

```
#define FS65_RW_M_WU_IO0_MASK 0xC0U
```

IO_0 wake-up configuration

6.7.2.777 FS65_RW_M_WU_IO0_NO_WAKEUP

```
#define FS65_RW_M_WU_IO0_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO0_SHIFT)
```

NO wake-up capability

6.7.2.778 FS65_RW_M_WU_IO0_RISING_EDGE

```
#define FS65_RW_M_WU_IO0_RISING_EDGE (0x01U << FS65_RW_M_WU_IO0_SHIFT)
```

Wake-up on rising edge - or high level

6.7.2.779 FS65_RW_M_WU_IO0_SHIFT

```
#define FS65_RW_M_WU_IO0_SHIFT 0x06U
```

IO_0 wake-up configuration

6.7.2.780 FS65_RW_M_WU_IO2_ANY_EDGE

```
#define FS65_RW_M_WU_IO2_ANY_EDGE (0x03U << FS65_RW_M_WU_IO2_SHIFT)
```

Wake-up on any edge

6.7.2.781 FS65_RW_M_WU_IO2_FALLING_EDGE

```
#define FS65_RW_M_WU_IO2_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO2_SHIFT)
```

Wake-up on falling edge - or low level

6.7.2.782 FS65_RW_M_WU_IO2_MASK

```
#define FS65_RW_M_WU_IO2_MASK 0x30U
```

IO_2 wake-up configuration

6.7.2.783 FS65_RW_M_WU_IO2_NO_WAKEUP

```
#define FS65_RW_M_WU_IO2_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO2_SHIFT)
```

NO wake-up capability

6.7.2.784 FS65_RW_M_WU_IO2_RISING_EDGE

```
#define FS65_RW_M_WU_IO2_RISING_EDGE (0x01U << FS65_RW_M_WU_IO2_SHIFT)
```

Wake-up on rising edge - or high level

6.7.2.785 FS65_RW_M_WU_IO2_SHIFT

```
#define FS65_RW_M_WU_IO2_SHIFT 0x04U
```

IO_2 wake-up configuration

6.7.2.786 FS65_RW_M_WU_IO3_ANY_EDGE

```
#define FS65_RW_M_WU_IO3_ANY_EDGE (0x03U << FS65_RW_M_WU_IO3_SHIFT)
```

Wake-up on any edge

6.7.2.787 FS65_RW_M_WU_IO3_FALLING_EDGE

```
#define FS65_RW_M_WU_IO3_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO3_SHIFT)
```

Wake-up on falling edge - or low level

6.7.2.788 FS65_RW_M_WU_IO3_MASK

```
#define FS65_RW_M_WU_IO3_MASK 0x0CU
```

IO_3 wake-up configuration

6.7.2.789 FS65_RW_M_WU_IO3_NO_WAKEUP

```
#define FS65_RW_M_WU_IO3_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO3_SHIFT)
```

NO wake-up capability

6.7.2.790 FS65_RW_M_WU_IO3_RISING_EDGE

```
#define FS65_RW_M_WU_IO3_RISING_EDGE (0x01U << FS65_RW_M_WU_IO3_SHIFT)
```

Wake-up on rising edge - or high level

6.7.2.791 FS65_RW_M_WU_IO3_SHIFT

```
#define FS65_RW_M_WU_IO3_SHIFT 0x02U
```

IO_3 wake-up configuration

6.7.2.792 FS65_RW_M_WU_IO4_ANY_EDGE

```
#define FS65_RW_M_WU_IO4_ANY_EDGE (0x03U << FS65_RW_M_WU_IO4_SHIFT)
```

Wake-up on any edge

6.7.2.793 FS65_RW_M_WU_IO4_FALLING_EDGE

```
#define FS65_RW_M_WU_IO4_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO4_SHIFT)
```

Wake-up on falling edge - or low level

6.7.2.794 FS65_RW_M_WU_IO4_MASK

```
#define FS65_RW_M_WU_IO4_MASK 0x03U
```

IO_4 wake-up configuration

6.7.2.795 FS65_RW_M_WU_IO4_NO_WAKEUP

```
#define FS65_RW_M_WU_IO4_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO4_SHIFT)
```

NO wake-up capability

6.7.2.796 FS65_RW_M_WU_IO4_RISING_EDGE

```
#define FS65_RW_M_WU_IO4_RISING_EDGE (0x01U << FS65_RW_M_WU_IO4_SHIFT)
```

Wake-up on rising edge - or high level

6.7.2.797 FS65_RW_M_WU_IO4_SHIFT

```
#define FS65_RW_M_WU_IO4_SHIFT 0x00U
```

IO_4 wake-up configuration

6.7.2.798 FS65_RW_M_WU_IO5_ANY_EDGE

```
#define FS65_RW_M_WU_IO5_ANY_EDGE (0x03U << FS65_RW_M_WU_IO5_SHIFT)
```

Wake-up on any edge

6.7.2.799 FS65_RW_M_WU_IO5_FALLING_EDGE

```
#define FS65_RW_M_WU_IO5_FALLING_EDGE (0x02U << FS65_RW_M_WU_IO5_SHIFT)
```

Wake-up on falling edge - or low level

6.7.2.800 FS65_RW_M_WU_IO5_MASK

```
#define FS65_RW_M_WU_IO5_MASK 0xC0U
```

IO_5 wake-up configuration

6.7.2.801 FS65_RW_M_WU_IO5_NO_WAKEUP

```
#define FS65_RW_M_WU_IO5_NO_WAKEUP (0x00U << FS65_RW_M_WU_IO5_SHIFT)
```

NO wake-up capability

6.7.2.802 FS65_RW_M_WU_IO5_RISING_EDGE

```
#define FS65_RW_M_WU_IO5_RISING_EDGE (0x01U << FS65_RW_M_WU_IO5_SHIFT)
```

Wake-up on rising edge - or high level

6.7.2.803 FS65_RW_M_WU_IO5_SHIFT

```
#define FS65_RW_M_WU_IO5_SHIFT 0x06U
```

IO_5 wake-up configuration

6.7.2.804 FS65_W_FS_ABIST2_FS1B_ABIST_FS1B

```
#define FS65_W_FS_ABIST2_FS1B_ABIST_FS1B (0x01U << FS65_W_FS_ABIST2_FS1B_SHIFT)
```

Launch ABIST on FS1B

6.7.2.805 FS65_W_FS_ABIST2_FS1B_MASK

```
#define FS65_W_FS_ABIST2_FS1B_MASK 0x40U
```

Request ABIST execution on FS1B

6.7.2.806 FS65_W_FS_ABIST2_FS1B_NO_ACTION

```
#define FS65_W_FS_ABIST2_FS1B_NO_ACTION (0x00U << FS65_W_FS_ABIST2_FS1B_SHIFT)
```

No action

6.7.2.807 FS65_W_FS_ABIST2_FS1B_SHIFT

```
#define FS65_W_FS_ABIST2_FS1B_SHIFT 0x06U
```

Request ABIST execution on FS1B

6.7.2.808 FS65_W_FS_ABIST2_VAUX_ABIST_VAUX

```
#define FS65_W_FS_ABIST2_VAUX_ABIST_VAUX (0x01U << FS65_W_FS_ABIST2_VAUX_SHIFT)
```

Launch ABIST on VAUX

6.7.2.809 FS65_W_FS_ABIST2_VAUX_MASK

```
#define FS65_W_FS_ABIST2_VAUX_MASK 0x20U
```

Request ABIST execution on VAUX

6.7.2.810 FS65_W_FS_ABIST2_VAUX_NO_ACTION

```
#define FS65_W_FS_ABIST2_VAUX_NO_ACTION (0x00U << FS65_W_FS_ABIST2_VAUX_SHIFT)
```

No action

6.7.2.811 FS65_W_FS_ABIST2_VAUX_SHIFT

```
#define FS65_W_FS_ABIST2_VAUX_SHIFT 0x05U
```

Request ABIST execution on VAUX

6.7.2.812 FS65_W_FS_DIS_8S_DISABLED

```
#define FS65_W_FS_DIS_8S_DISABLED (0x01U << FS65_W_FS_DIS_8S_SHIFT)
```

Disabled

6.7.2.813 FS65_W_FS_DIS_8S_ENABLED

```
#define FS65_W_FS_DIS_8S_ENABLED (0x00U << FS65_W_FS_DIS_8S_SHIFT)
```

Enabled

6.7.2.814 FS65_W_FS_DIS_8S_MASK

```
#define FS65_W_FS_DIS_8S_MASK 0x40U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.815 FS65_W_FS_DIS_8S_SHIFT

```
#define FS65_W_FS_DIS_8S_SHIFT 0x06U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.816 FS65_W_FS_FLT_ERR_FS_INT1_FIN2

```
#define FS65_W_FS_FLT_ERR_FS_INT1_FIN2 (0x01U << FS65_W_FS_FLT_ERR_FS_SHIFT)
```

intermediate = 1; final = 2

6.7.2.817 FS65_W_FS_FLT_ERR_FS_INT3_FIN6

```
#define FS65_W_FS_FLT_ERR_FS_INT3_FIN6 (0x00U << FS65_W_FS_FLT_ERR_FS_SHIFT)
```

intermediate = 3; final = 6

6.7.2.818 FS65_W_FS_FLT_ERR_FS_MASK

```
#define FS65_W_FS_FLT_ERR_FS_MASK 0x80U
```

Configure the values of the fault error counter

6.7.2.819 FS65_W_FS_FLT_ERR_FS_SHIFT

```
#define FS65_W_FS_FLT_ERR_FS_SHIFT 0x07U
```

Configure the values of the fault error counter

6.7.2.820 FS65_W_FS_FLT_ERR_IMP_FS0B

```
#define FS65_W_FS_FLT_ERR_IMP_FS0B (0x01U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
```

FS0B is asserted low if FLT_ERR_CNT >= intermediate value

6.7.2.821 FS65_W_FS_FLT_ERR_IMP_FS0B_RSTB

```
#define FS65_W_FS_FLT_ERR_IMP_FS0B_RSTB (0x03U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
```

FS0B is asserted low if FLT_ERR_CNT >= intermediate value RSTB is asserted low if FLT_ERR_CNT >= intermediate value and WD error counter = WD_CNT_ERR[1:0]

6.7.2.822 FS65_W_FS_FLT_ERR_IMP_MASK

```
#define FS65_W_FS_FLT_ERR_IMP_MASK 0x30U
```

Configure RSTB and FS0B behavior when fault error counter >= intermediate value

6.7.2.823 FS65_W_FS_FLT_ERR_IMP_NO_EFFECT

```
#define FS65_W_FS_FLT_ERR_IMP_NO_EFFECT (0x00U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
```

No effect on RSTB and FS0B

6.7.2.824 FS65_W_FS_FLT_ERR_IMP_RSTB

```
#define FS65_W_FS_FLT_ERR_IMP_RSTB (0x02U << FS65_W_FS_FLT_ERR_IMP_SHIFT)
```

RSTB is asserted low if FLT_ERR_CNT >= intermediate value and WD error counter = WD_CNT_ERR[1:0]

6.7.2.825 FS65_W_FS_FLT_ERR_IMP_SHIFT

```
#define FS65_W_FS_FLT_ERR_IMP_SHIFT 0x04U
```

Configure RSTB and FS0B behavior when fault error counter >= intermediate value

6.7.2.826 FS65_W_FS_FS0B_REQ_FS0B_REQ

```
#define FS65_W_FS_FS0B_REQ_FS0B_REQ (0x01U << FS65_W_FS_FS0B_REQ_SHIFT)
```

Request FS0B assertion

6.7.2.827 FS65_W_FS_FS0B_REQ_MASK

```
#define FS65_W_FS_FS0B_REQ_MASK 0x20U
```

Request FS0B to be asserted low

6.7.2.828 FS65_W_FS_FS0B_REQ_NO_REQUEST

```
#define FS65_W_FS_FS0B_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS0B_REQ_SHIFT)
```

No request

6.7.2.829 FS65_W_FS_FS0B_REQ_SHIFT

```
#define FS65_W_FS_FS0B_REQ_SHIFT 0x05U
```

Request FS0B to be asserted low

6.7.2.830 FS65_W_FS_FS1B_CAN_IMPACT_MASK

```
#define FS65_W_FS_FS1B_CAN_IMPACT_MASK 0x40U
```

Configure CAN behavior when FS1B is asserted low

6.7.2.831 FS65_W_FS_FS1B_CAN_IMPACT_NO_EFFECT

```
#define FS65_W_FS_FS1B_CAN_IMPACT_NO_EFFECT (0x00U << FS65_W_FS_FS1B_CAN_IMPACT_SHIFT)
```

No effect

6.7.2.832 FS65_W_FS_FS1B_CAN_IMPACT_RX_ONLY

```
#define FS65_W_FS_FS1B_CAN_IMPACT_RX_ONLY (0x01U << FS65_W_FS_FS1B_CAN_IMPACT_SHIFT)
```

CAN in Rx only or sleep mode when FS1B is asserted (depends on CAN_DIS_CFG bit in INIT_WU2 register)

6.7.2.833 FS65_W_FS_FS1B_CAN_IMPACT_SHIFT

```
#define FS65_W_FS_FS1B_CAN_IMPACT_SHIFT 0x06U
```

Configure CAN behavior when FS1B is asserted low

6.7.2.834 FS65_W_FS_FS1B_DLY_REQ_FS1B_REQ

```
#define FS65_W_FS_FS1B_DLY_REQ_FS1B_REQ (0x01U << FS65_W_FS_FS1B_DLY_REQ_SHIFT)
```

Request FS1B assertion with tDELAY controlled by the backup delay (open S1)

6.7.2.835 FS65_W_FS_FS1B_DLY_REQ_MASK

```
#define FS65_W_FS_FS1B_DLY_REQ_MASK 0x40U
```

Request activation of FS1B backup delay (open/close switch S1)

6.7.2.836 FS65_W_FS_FS1B_DLY_REQ_NO_REQUEST

```
#define FS65_W_FS_FS1B_DLY_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS1B_DLY_REQ_SHIFT)
```

No request (close S1)

6.7.2.837 FS65_W_FS_FS1B_DLY_REQ_SHIFT

```
#define FS65_W_FS_FS1B_DLY_REQ_SHIFT 0x06U
```

Request activation of FS1B backup delay (open/close switch S1)

6.7.2.838 FS65_W_FS_FS1B_REQ_FS1B_REQ

```
#define FS65_W_FS_FS1B_REQ_FS1B_REQ (0x01U << FS65_W_FS_FS1B_REQ_SHIFT)
```

Request FS1B assertion with immediate assertion, no delay

6.7.2.839 FS65_W_FS_FS1B_REQ_MASK

```
#define FS65_W_FS_FS1B_REQ_MASK 0x80U
```

Request FS1B to be asserted low

6.7.2.840 FS65_W_FS_FS1B_REQ_NO_REQUEST

```
#define FS65_W_FS_FS1B_REQ_NO_REQUEST (0x00U << FS65_W_FS_FS1B_REQ_SHIFT)
```

No request

6.7.2.841 FS65_W_FS_FS1B_REQ_SHIFT

```
#define FS65_W_FS_FS1B_REQ_SHIFT 0x07U
```

Request FS1B to be asserted low

6.7.2.842 FS65_W_FS_FS1B_TIME_0_0

```
#define FS65_W_FS_FS1B_TIME_0_0 (0x00U << FS65_W_FS_FS1B_TIME_SHIFT)
```

0

6.7.2.843 FS65_W_FS_FS1B_TIME_106_848MS

```
#define FS65_W_FS_FS1B_TIME_106_848MS (0x0AU << FS65_W_FS_FS1B_TIME_SHIFT)
```

106 ms (FS1B_TIME_RANGE bit = 0) | 848 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.844 FS65_W_FS_FS1B_TIME_10MS_80MS

```
#define FS65_W_FS_FS1B_TIME_10MS_80MS (0x01U << FS65_W_FS_FS1B_TIME_SHIFT)
```

10 ms (FS1B_TIME_RANGE bit = 0) | 80 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.845 FS65_W_FS_FS1B_TIME_138_1103MS

```
#define FS65_W_FS_FS1B_TIME_138_1103MS (0x0BU << FS65_W_FS_FS1B_TIME_SHIFT)
```

138 ms (FS1B_TIME_RANGE bit = 0) | 1103 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.846 FS65_W_FS_FS1B_TIME_13_104MS

```
#define FS65_W_FS_FS1B_TIME_13_104MS (0x02U << FS65_W_FS_FS1B_TIME_SHIFT)
```

13 ms (FS1B_TIME_RANGE bit = 0) | 104 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.847 FS65_W_FS_FS1B_TIME_179_1434MS

```
#define FS65_W_FS_FS1B_TIME_179_1434MS (0x0CU << FS65_W_FS_FS1B_TIME_SHIFT)
```

179 ms (FS1B_TIME_RANGE bit = 0) | 1434 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.848 FS65_W_FS_FS1B_TIME_17_135MS

```
#define FS65_W_FS_FS1B_TIME_17_135MS (0x03U << FS65_W_FS_FS1B_TIME_SHIFT)
```

17 ms (FS1B_TIME_RANGE bit = 0) | 135 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.849 FS65_W_FS_FS1B_TIME_22_176MS

```
#define FS65_W_FS_FS1B_TIME_22_176MS (0x04U << FS65_W_FS_FS1B_TIME_SHIFT)
```

22 ms (FS1B_TIME_RANGE bit = 0) | 176 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.850 FS65_W_FS_FS1B_TIME_233_1864MS

```
#define FS65_W_FS_FS1B_TIME_233_1864MS (0x0DU << FS65_W_FS_FS1B_TIME_SHIFT)
```

233 ms (FS1B_TIME_RANGE bit = 0) | 1864 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.851 FS65_W_FS_FS1B_TIME_29_228MS

```
#define FS65_W_FS_FS1B_TIME_29_228MS (0x05U << FS65_W_FS_FS1B_TIME_SHIFT)
```

29 ms (FS1B_TIME_RANGE bit = 0) | 228 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.852 FS65_W_FS_FS1B_TIME_303_2423MS

```
#define FS65_W_FS_FS1B_TIME_303_2423MS (0x0EU << FS65_W_FS_FS1B_TIME_SHIFT)
```

303 ms (FS1B_TIME_RANGE bit = 0) | 2423 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.853 FS65_W_FS_FS1B_TIME_37_297MS

```
#define FS65_W_FS_FS1B_TIME_37_297MS (0x06U << FS65_W_FS_FS1B_TIME_SHIFT)
```

37 ms (FS1B_TIME_RANGE bit = 0) | 297 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.854 FS65_W_FS_FS1B_TIME_394_3150MS

```
#define FS65_W_FS_FS1B_TIME_394_3150MS (0x0FU << FS65_W_FS_FS1B_TIME_SHIFT)
```

394 ms (FS1B_TIME_RANGE bit = 0) | 3150 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.855 FS65_W_FS_FS1B_TIME_48_386MS

```
#define FS65_W_FS_FS1B_TIME_48_386MS (0x07U << FS65_W_FS_FS1B_TIME_SHIFT)
```

48 ms (FS1B_TIME_RANGE bit = 0) | 386 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.856 FS65_W_FS_FS1B_TIME_63_502MS

```
#define FS65_W_FS_FS1B_TIME_63_502MS (0x08U << FS65_W_FS_FS1B_TIME_SHIFT)
```

63 ms (FS1B_TIME_RANGE bit = 0) | 502 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.857 FS65_W_FS_FS1B_TIME_82_653MS

```
#define FS65_W_FS_FS1B_TIME_82_653MS (0x09U << FS65_W_FS_FS1B_TIME_SHIFT)
```

82 ms (FS1B_TIME_RANGE bit = 0) | 653 ms (FS1B_TIME_RANGE bit = 1)

6.7.2.858 FS65_W_FS_FS1B_TIME_MASK

```
#define FS65_W_FS_FS1B_TIME_MASK 0xF0U
```

FS1B timing range factor x1(FS1B_TIME_RANGE bit = 0), FS1B timing range factor x8(FS1B_TIME_RANGE bit = 1)

6.7.2.859 FS65_W_FS_FS1B_TIME_RANGE_MASK

```
#define FS65_W_FS_FS1B_TIME_RANGE_MASK 0x10U
```

Configure the FS1B timing range factor x1 or x8

6.7.2.860 FS65_W_FS_FS1B_TIME_RANGE_SHIFT

```
#define FS65_W_FS_FS1B_TIME_RANGE_SHIFT 0x04U
```

Configure the FS1B timing range factor x1 or x8

6.7.2.861 FS65_W_FS_FS1B_TIME_RANGE_X1

```
#define FS65_W_FS_FS1B_TIME_RANGE_X1 (0x00U << FS65_W_FS_FS1B_TIME_RANGE_SHIFT)
```

x1 timing range factor

6.7.2.862 FS65_W_FS_FS1B_TIME_RANGE_X8

```
#define FS65_W_FS_FS1B_TIME_RANGE_X8 (0x01U << FS65_W_FS_FS1B_TIME_RANGE_SHIFT)
```

x8 timing range factor

6.7.2.863 FS65_W_FS_FS1B_TIME_SHIFT

```
#define FS65_W_FS_FS1B_TIME_SHIFT 0x04U
```

FS1B timing range factor x1(FS1B_TIME_RANGE bit = 0), FS1B timing range factor x8(FS1B_TIME_RANGE bit = 1)

6.7.2.864 FS65_W_FS_IO_23_FS_MASK

```
#define FS65_W_FS_IO_23_FS_MASK 0x40U
```

Configure the couple of IO_3:2 as safety inputs for FCCU monitoring

6.7.2.865 FS65_W_FS_IO_23_FS_NOT_SAFETY

```
#define FS65_W_FS_IO_23_FS_NOT_SAFETY (0x00U << FS65_W_FS_IO_23_FS_SHIFT)
```

Not_safety

6.7.2.866 FS65_W_FS_IO_23_FS_SAFETY_CRITICAL

```
#define FS65_W_FS_IO_23_FS_SAFETY_CRITICAL (0x01U << FS65_W_FS_IO_23_FS_SHIFT)
```

Safety_critical

6.7.2.867 FS65_W_FS_IO_23_FS_SHIFT

```
#define FS65_W_FS_IO_23_FS_SHIFT 0x06U
```

Configure the couple of IO_3:2 as safety inputs for FCCU monitoring

6.7.2.868 FS65_W_FS_IO_45_FS_MASK

```
#define FS65_W_FS_IO_45_FS_MASK 0x80U
```

Configure the couple of IO_4:5 as safety inputs for external IC error monitoring

6.7.2.869 FS65_W_FS_IO_45_FS_NOT_SAFETY

```
#define FS65_W_FS_IO_45_FS_NOT_SAFETY (0x00U << FS65_W_FS_IO_45_FS_SHIFT)
```

Not safety

6.7.2.870 FS65_W_FS_IO_45_FS_SAFETY_CRITICAL

```
#define FS65_W_FS_IO_45_FS_SAFETY_CRITICAL (0x01U << FS65_W_FS_IO_45_FS_SHIFT)
```

Safety critical

6.7.2.871 FS65_W_FS_IO_45_FS_SHIFT

```
#define FS65_W_FS_IO_45_FS_SHIFT 0x07U
```

Configure the couple of IO_4:5 as safety inputs for external IC error monitoring

6.7.2.872 FS65_W_FS_PS_HIGH

```
#define FS65_W_FS_PS_HIGH (0x00U << FS65_W_FS_PS_SHIFT)
```

Fccu_eaout_1:0 active high

6.7.2.873 FS65_W_FS_PS_LOW

```
#define FS65_W_FS_PS_LOW (0x01U << FS65_W_FS_PS_SHIFT)
```

Fccu_eaout_1:0 active low

6.7.2.874 FS65_W_FS_PS_MASK

```
#define FS65_W_FS_PS_MASK 0x20U
```

Configure the FCCU polarity

6.7.2.875 FS65_W_FS_PS_SHIFT

```
#define FS65_W_FS_PS_SHIFT 0x05U
```

Configure the FCCU polarity

6.7.2.876 FS65_W_FS_RELEASE_FSXB_MASK

```
#define FS65_W_FS_RELEASE_FSXB_MASK 0xFFU
```

Secured 8 bits word to release the FS0B and FS1B pins, depend on LFSR_out value and calculation

6.7.2.877 FS65_W_FS_RELEASE_FSXB_SHIFT

```
#define FS65_W_FS_RELEASE_FSXB_SHIFT 0x00U
```

Secured 8 bits word to release the FS0B and FS1B pins, depend on LFSR_out value and calculation

6.7.2.878 FS65_W_FS_RSTB_DURATION_10MS

```
#define FS65_W_FS_RSTB_DURATION_10MS (0x00U << FS65_W_FS_RSTB_DURATION_SHIFT)
```

10 ms

6.7.2.879 FS65_W_FS_RSTB_DURATION_1MS

```
#define FS65_W_FS_RSTB_DURATION_1MS (0x01U << FS65_W_FS_RSTB_DURATION_SHIFT)
```

1.0 ms

6.7.2.880 FS65_W_FS_RSTB_DURATION_MASK

```
#define FS65_W_FS_RSTB_DURATION_MASK 0x10U
```

Configure the RSTB low duration time

6.7.2.881 FS65_W_FS_RSTB_DURATION_SHIFT

```
#define FS65_W_FS_RSTB_DURATION_SHIFT 0x04U
```

Configure the RSTB low duration time

6.7.2.882 FS65_W_FS_RSTB_REQ_MASK

```
#define FS65_W_FS_RSTB_REQ_MASK 0x10U
```

Request a RSTB low pulse

6.7.2.883 FS65_W_FS_RSTB_REQ_NO_REQUEST

```
#define FS65_W_FS_RSTB_REQ_NO_REQUEST (0x00U << FS65_W_FS_RSTB_REQ_SHIFT)
```

No request

6.7.2.884 FS65_W_FS_RSTB_REQ_RSTB_REQ

```
#define FS65_W_FS_RSTB_REQ_RSTB_REQ (0x01U << FS65_W_FS_RSTB_REQ_SHIFT)
```

Request a RSTB low pulse

6.7.2.885 FS65_W_FS_RSTB_REQ_SHIFT

```
#define FS65_W_FS_RSTB_REQ_SHIFT 0x04U
```

Request a RSTB low pulse

6.7.2.886 FS65_W_FS_TDLY_TDUR_DELAY

```
#define FS65_W_FS_TDLY_TDUR_DELAY (0x00U << FS65_W_FS_TDLY_TDUR_SHIFT)
```

FS1B tDELAY mode

6.7.2.887 FS65_W_FS_TDLY_TDUR_DURATION

```
#define FS65_W_FS_TDLY_TDUR_DURATION (0x01U << FS65_W_FS_TDLY_TDUR_SHIFT)
```

FS1B tDURATION mode

6.7.2.888 FS65_W_FS_TDLY_TDUR_MASK

```
#define FS65_W_FS_TDLY_TDUR_MASK 0x80U
```

FS1B delay or FS1B duration mode selection

6.7.2.889 FS65_W_FS_TDLY_TDUR_SHIFT

```
#define FS65_W_FS_TDLY_TDUR_SHIFT 0x07U
```

FS1B delay or FS1B duration mode selection

6.7.2.890 FS65_W_FS_VAUX_5D_DEGRADED

```
#define FS65_W_FS_VAUX_5D_DEGRADED (0x01U << FS65_W_FS_VAUX_5D_SHIFT)
```

Degraded mode; lower undervoltage detection threshold applied (VAUX_UV_5D)

6.7.2.891 FS65_W_FS_VAUX_5D_MASK

```
#define FS65_W_FS_VAUX_5D_MASK 0x20U
```

Configure the VAUX undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.892 FS65_W_FS_VAUX_5D_NORMAL

```
#define FS65_W_FS_VAUX_5D_NORMAL (0x00U << FS65_W_FS_VAUX_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VAUX_UV_5)

6.7.2.893 FS65_W_FS_VAUX_5D_SHIFT

```
#define FS65_W_FS_VAUX_5D_SHIFT 0x05U
```

Configure the VAUX undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.894 FS65_W_FS_VAUX_FS_OV_FS0B

```
#define FS65_W_FS_VAUX_FS_OV_FS0B (0x02U << FS65_W_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on FS0B only

6.7.2.895 FS65_W_FS_VAUX_FS_OV_MASK

```
#define FS65_W_FS_VAUX_FS_OV_MASK 0xC0U
```

VAUX overvoltage safety impact

6.7.2.896 FS65_W_FS_VAUX_FS_OV_NO_EFFECT

```
#define FS65_W_FS_VAUX_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VAUX_FS_OV_SHIFT)
```

No effect of VAUX_OV on RSTB and FS0B

6.7.2.897 FS65_W_FS_VAUX_FS_OV_RSTB

```
#define FS65_W_FS_VAUX_FS_OV_RSTB (0x01U << FS65_W_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on RSTB only

6.7.2.898 FS65_W_FS_VAUX_FS_OV_RSTB_FS0B

```
#define FS65_W_FS_VAUX_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VAUX_FS_OV_SHIFT)
```

VAUX_OV does have an impact on RSTB and FS0B

6.7.2.899 FS65_W_FS_VAUX_FS_OV_SHIFT

```
#define FS65_W_FS_VAUX_FS_OV_SHIFT 0x06U
```

VAUX overvoltage safety impact

6.7.2.900 FS65_W_FS_VAUX_FS_UV_FS0B

```
#define FS65_W_FS_VAUX_FS_UV_FS0B (0x02U << FS65_W_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on FS0B only

6.7.2.901 FS65_W_FS_VAUX_FS_UV_MASK

```
#define FS65_W_FS_VAUX_FS_UV_MASK 0x30U
```

VAUX undervoltage safety impact

6.7.2.902 FS65_W_FS_VAUX_FS_UV_NO_EFFECT

```
#define FS65_W_FS_VAUX_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VAUX_FS_UV_SHIFT)
```

No effect of VAUX_UV on RSTB and FS0B

6.7.2.903 FS65_W_FS_VAUX_FS_UV_RSTB

```
#define FS65_W_FS_VAUX_FS_UV_RSTB (0x01U << FS65_W_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on RSTB only

6.7.2.904 FS65_W_FS_VAUX_FS_UV_RSTB_FS0B

```
#define FS65_W_FS_VAUX_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VAUX_FS_UV_SHIFT)
```

VAUX_UV does have an impact on RSTB and FS0B

6.7.2.905 FS65_W_FS_VAUX_FS_UV_SHIFT

```
#define FS65_W_FS_VAUX_FS_UV_SHIFT 0x04U
```

VAUX undervoltage safety impact

6.7.2.906 FS65_W_FS_VCCA_5D_DEGRADED

```
#define FS65_W_FS_VCCA_5D_DEGRADED (0x01U << FS65_W_FS_VCCA_5D_SHIFT)
```

Degraded mode, lower undervoltage detection threshold applied (VCCA_UV_D)

6.7.2.907 FS65_W_FS_VCCA_5D_MASK

```
#define FS65_W_FS_VCCA_5D_MASK 0x40U
```

Configure the VCCA undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.908 FS65_W_FS_VCCA_5D_NORMAL

```
#define FS65_W_FS_VCCA_5D_NORMAL (0x00U << FS65_W_FS_VCCA_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VCCA_UV_5)

6.7.2.909 FS65_W_FS_VCCA_5D_SHIFT

```
#define FS65_W_FS_VCCA_5D_SHIFT 0x06U
```

Configure the VCCA undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.910 FS65_W_FS_VCCA_FS_OV_FS0B

```
#define FS65_W_FS_VCCA_FS_OV_FS0B (0x02U << FS65_W_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on FS0B only

6.7.2.911 FS65_W_FS_VCCA_FS_OV_MASK

```
#define FS65_W_FS_VCCA_FS_OV_MASK 0xC0U
```

VCCA overvoltage safety impact

6.7.2.912 FS65_W_FS_VCCA_FS_OV_NO_EFFECT

```
#define FS65_W_FS_VCCA_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VCCA_FS_OV_SHIFT)
```

No effect of VCCA_OV on RSTB and FS0B

6.7.2.913 FS65_W_FS_VCCA_FS_OV_RSTB

```
#define FS65_W_FS_VCCA_FS_OV_RSTB (0x01U << FS65_W_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on RSTB only

6.7.2.914 FS65_W_FS_VCCA_FS_OV_RSTB_FS0B

```
#define FS65_W_FS_VCCA_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VCCA_FS_OV_SHIFT)
```

VCCA_OV does have an impact on RSTB and FS0B

6.7.2.915 FS65_W_FS_VCCA_FS_OV_SHIFT

```
#define FS65_W_FS_VCCA_FS_OV_SHIFT 0x06U
```

VCCA overvoltage safety impact

6.7.2.916 FS65_W_FS_VCCA_FS_UV_FS0B

```
#define FS65_W_FS_VCCA_FS_UV_FS0B (0x02U << FS65_W_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on FS0B only

6.7.2.917 FS65_W_FS_VCCA_FS_UV_MASK

```
#define FS65_W_FS_VCCA_FS_UV_MASK 0x30U
```

VCCA undervoltage safety impact

6.7.2.918 FS65_W_FS_VCCA_FS_UV_NO_EFFECT

```
#define FS65_W_FS_VCCA_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VCCA_FS_UV_SHIFT)
```

No effect of VCCA_UV on RSTB and FS0B

6.7.2.919 FS65_W_FS_VCCA_FS_UV_RSTB

```
#define FS65_W_FS_VCCA_FS_UV_RSTB (0x01U << FS65_W_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on RSTB only

6.7.2.920 FS65_W_FS_VCCA_FS_UV_RSTB_FS0B

```
#define FS65_W_FS_VCCA_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VCCA_FS_UV_SHIFT)
```

VCCA_UV does have an impact on RSTB and FS0B

6.7.2.921 FS65_W_FS_VCCA_FS_UV_SHIFT

```
#define FS65_W_FS_VCCA_FS_UV_SHIFT 0x04U
```

VCCA undervoltage safety impact

6.7.2.922 FS65_W_FS_VCORE_5D_DEGRADED

```
#define FS65_W_FS_VCORE_5D_DEGRADED (0x01U << FS65_W_FS_VCORE_5D_SHIFT)
```

Degraded mode, lower undervoltage detection threshold applied (VCORE_FB_UV_D)

6.7.2.923 FS65_W_FS_VCORE_5D_MASK

```
#define FS65_W_FS_VCORE_5D_MASK 0x80U
```

Configure the VCORE undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.924 FS65_W_FS_VCORE_5D_NORMAL

```
#define FS65_W_FS_VCORE_5D_NORMAL (0x00U << FS65_W_FS_VCORE_5D_SHIFT)
```

Normal 5.0 V undervoltage detection threshold (VCORE_FB_UV)

6.7.2.925 FS65_W_FS_VCORE_5D_SHIFT

```
#define FS65_W_FS_VCORE_5D_SHIFT 0x07U
```

Configure the VCORE undervoltage in degraded mode. Only valid for 5.0 V

6.7.2.926 FS65_W_FS_VCORE_FS_OV_FS0B

```
#define FS65_W_FS_VCORE_FS_OV_FS0B (0x02U << FS65_W_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on FS0B only

6.7.2.927 FS65_W_FS_VCORE_FS_OV_MASK

```
#define FS65_W_FS_VCORE_FS_OV_MASK 0xC0U
```

VCORE_FB overvoltage safety impact

6.7.2.928 FS65_W_FS_VCORE_FS_OV_NO_EFFECT

```
#define FS65_W_FS_VCORE_FS_OV_NO_EFFECT (0x00U << FS65_W_FS_VCORE_FS_OV_SHIFT)
```

No effect of VCORE_FB_OV on RSTB and FS0B

6.7.2.929 FS65_W_FS_VCORE_FS_OV_RSTB

```
#define FS65_W_FS_VCORE_FS_OV_RSTB (0x01U << FS65_W_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on RSTB only

6.7.2.930 FS65_W_FS_VCORE_FS_OV_RSTB_FS0B

```
#define FS65_W_FS_VCORE_FS_OV_RSTB_FS0B (0x03U << FS65_W_FS_VCORE_FS_OV_SHIFT)
```

VCORE_FB_OV does have an impact on RSTB and FS0B

6.7.2.931 FS65_W_FS_VCORE_FS_OV_SHIFT

```
#define FS65_W_FS_VCORE_FS_OV_SHIFT 0x06U
```

VCORE_FB overvoltage safety impact

6.7.2.932 FS65_W_FS_VCORE_FS_UV_FS0B

```
#define FS65_W_FS_VCORE_FS_UV_FS0B (0x02U << FS65_W_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on FS0B only

6.7.2.933 FS65_W_FS_VCORE_FS_UV_MASK

```
#define FS65_W_FS_VCORE_FS_UV_MASK 0x30U
```

VCORE_FB undervoltage safety impact

6.7.2.934 FS65_W_FS_VCORE_FS_UV_NO_EFFECT

```
#define FS65_W_FS_VCORE_FS_UV_NO_EFFECT (0x00U << FS65_W_FS_VCORE_FS_UV_SHIFT)
```

No effect of VCORE_FB_UV on RSTB and FS0B

6.7.2.935 FS65_W_FS_VCORE_FS_UV_RSTB

```
#define FS65_W_FS_VCORE_FS_UV_RSTB (0x01U << FS65_W_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on RSTB only

6.7.2.936 FS65_W_FS_VCORE_FS_UV_RSTB_FS0B

```
#define FS65_W_FS_VCORE_FS_UV_RSTB_FS0B (0x03U << FS65_W_FS_VCORE_FS_UV_SHIFT)
```

VCORE_FB_UV does have an impact on RSTB and FS0B

6.7.2.937 FS65_W_FS_VCORE_FS_UV_SHIFT

```
#define FS65_W_FS_VCORE_FS_UV_SHIFT 0x04U
```

VCORE_FB undervoltage safety impact

6.7.2.938 FS65_W_FS_WD_CNT_ERR_2

```
#define FS65_W_FS_WD_CNT_ERR_2 (0x03U << FS65_W_FS_WD_CNT_ERR_SHIFT)
```

2

6.7.2.939 FS65_W_FS_WD_CNT_ERR_4

```
#define FS65_W_FS_WD_CNT_ERR_4 (0x02U << FS65_W_FS_WD_CNT_ERR_SHIFT)
```

4

6.7.2.940 FS65_W_FS_WD_CNT_ERR_6

```
#define FS65_W_FS_WD_CNT_ERR_6 (0x00U << FS65_W_FS_WD_CNT_ERR_SHIFT)
```

6

6.7.2.941 FS65_W_FS_WD_CNT_ERR_MASK

```
#define FS65_W_FS_WD_CNT_ERR_MASK 0xC0U
```

Configure the maximum value of the WD error counter

6.7.2.942 FS65_W_FS_WD_CNT_ERR_SHIFT

```
#define FS65_W_FS_WD_CNT_ERR_SHIFT 0x06U
```

Configure the maximum value of the WD error counter

6.7.2.943 FS65_W_FS_WD_CNT_RFR_1

```
#define FS65_W_FS_WD_CNT_RFR_1 (0x03U << FS65_W_FS_WD_CNT_RFR_SHIFT)
```

1

6.7.2.944 FS65_W_FS_WD_CNT_RFR_2

```
#define FS65_W_FS_WD_CNT_RFR_2 (0x02U << FS65_W_FS_WD_CNT_RFR_SHIFT)
```

2

6.7.2.945 FS65_W_FS_WD_CNT_RFR_4

```
#define FS65_W_FS_WD_CNT_RFR_4 (0x01U << FS65_W_FS_WD_CNT_RFR_SHIFT)
```

4

6.7.2.946 FS65_W_FS_WD_CNT_RFR_6

```
#define FS65_W_FS_WD_CNT_RFR_6 (0x00U << FS65_W_FS_WD_CNT_RFR_SHIFT)
```

6

6.7.2.947 FS65_W_FS_WD_CNT_RFR_MASK

```
#define FS65_W_FS_WD_CNT_RFR_MASK 0x30U
```

Configure the maximum value of the WD refresh counter

6.7.2.948 FS65_W_FS_WD_CNT_RFR_SHIFT

```
#define FS65_W_FS_WD_CNT_RFR_SHIFT 0x04U
```

Configure the maximum value of the WD refresh counter

6.7.2.949 FS65_W_FS_WD_IMPACT_FS0B

```
#define FS65_W_FS_WD_IMPACT_FS0B (0x02U << FS65_W_FS_WD_IMPACT_SHIFT)
```

FS0B only is asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.950 FS65_W_FS_WD_IMPACT_MASK

```
#define FS65_W_FS_WD_IMPACT_MASK 0x30U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.951 FS65_W_FS_WD_IMPACT_NO_EFFECT

```
#define FS65_W_FS_WD_IMPACT_NO_EFFECT (0x00U << FS65_W_FS_WD_IMPACT_SHIFT)
```

No effect on RSTB and FS0B if WD error counter = WD_CNT_ERR[1:0]

6.7.2.952 FS65_W_FS_WD_IMPACT_RSTB

```
#define FS65_W_FS_WD_IMPACT_RSTB (0x01U << FS65_W_FS_WD_IMPACT_SHIFT)
```

RSTB only is asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.953 FS65_W_FS_WD_IMPACT_RSTB_FS0B

```
#define FS65_W_FS_WD_IMPACT_RSTB_FS0B (0x03U << FS65_W_FS_WD_IMPACT_SHIFT)
```

RSTB and FS0B are asserted low if WD error counter = WD_CNT_ERR[1:0]

6.7.2.954 FS65_W_FS_WD_IMPACT_SHIFT

```
#define FS65_W_FS_WD_IMPACT_SHIFT 0x04U
```

Watchdog impact on RSTB and/or FS0B assertion

6.7.2.955 FS65_W_FS_WD_WINDOW_1024MS

```
#define FS65_W_FS_WD_WINDOW_1024MS (0x0FU << FS65_W_FS_WD_WINDOW_SHIFT)
```

1024 ms

6.7.2.956 FS65_W_FS_WD_WINDOW_128MS

```
#define FS65_W_FS_WD_WINDOW_128MS (0x0CU << FS65_W_FS_WD_WINDOW_SHIFT)
```

128 ms

6.7.2.957 FS65_W_FS_WD_WINDOW_12MS

```
#define FS65_W_FS_WD_WINDOW_12MS (0x07U << FS65_W_FS_WD_WINDOW_SHIFT)
```

12.0 ms

6.7.2.958 FS65_W_FS_WD_WINDOW_16MS

```
#define FS65_W_FS_WD_WINDOW_16MS (0x08U << FS65_W_FS_WD_WINDOW_SHIFT)
```

16 ms

6.7.2.959 FS65_W_FS_WD_WINDOW_1MS

```
#define FS65_W_FS_WD_WINDOW_1MS (0x01U << FS65_W_FS_WD_WINDOW_SHIFT)
```

1.0 ms

6.7.2.960 FS65_W_FS_WD_WINDOW_24MS

```
#define FS65_W_FS_WD_WINDOW_24MS (0x09U << FS65_W_FS_WD_WINDOW_SHIFT)
```

24 ms

6.7.2.961 FS65_W_FS_WD_WINDOW_256MS

```
#define FS65_W_FS_WD_WINDOW_256MS (0x0DU << FS65_W_FS_WD_WINDOW_SHIFT)
```

256 ms

6.7.2.962 FS65_W_FS_WD_WINDOW_2MS

```
#define FS65_W_FS_WD_WINDOW_2MS (0x02U << FS65_W_FS_WD_WINDOW_SHIFT)
```

2.0 ms

6.7.2.963 FS65_W_FS_WD_WINDOW_32MS

```
#define FS65_W_FS_WD_WINDOW_32MS (0x0AU << FS65_W_FS_WD_WINDOW_SHIFT)
```

32 ms

6.7.2.964 FS65_W_FS_WD_WINDOW_3MS

```
#define FS65_W_FS_WD_WINDOW_3MS (0x03U << FS65_W_FS_WD_WINDOW_SHIFT)
```

3.0 ms

6.7.2.965 FS65_W_FS_WD_WINDOW_4MS

```
#define FS65_W_FS_WD_WINDOW_4MS (0x04U << FS65_W_FS_WD_WINDOW_SHIFT)
```

4.0 ms

6.7.2.966 FS65_W_FS_WD_WINDOW_512MS

```
#define FS65_W_FS_WD_WINDOW_512MS (0x0EU << FS65_W_FS_WD_WINDOW_SHIFT)
```

512 ms

6.7.2.967 FS65_W_FS_WD_WINDOW_64MS

```
#define FS65_W_FS_WD_WINDOW_64MS (0x0BU << FS65_W_FS_WD_WINDOW_SHIFT)
```

64 ms

6.7.2.968 FS65_W_FS_WD_WINDOW_6MS

```
#define FS65_W_FS_WD_WINDOW_6MS (0x05U << FS65_W_FS_WD_WINDOW_SHIFT)
```

6.0 ms

6.7.2.969 FS65_W_FS_WD_WINDOW_8MS

```
#define FS65_W_FS_WD_WINDOW_8MS (0x06U << FS65_W_FS_WD_WINDOW_SHIFT)
```

8.0 ms

6.7.2.970 FS65_W_FS_WD_WINDOW_DISABLE

```
#define FS65_W_FS_WD_WINDOW_DISABLE (0x00U << FS65_W_FS_WD_WINDOW_SHIFT)
```

Disable (in INIT phase only)

6.7.2.971 FS65_W_FS_WD_WINDOW_MASK

```
#define FS65_W_FS_WD_WINDOW_MASK 0xF0U
```

Configure the watchdog window duration. Duty cycle if set to 50 %

6.7.2.972 FS65_W_FS_WD_WINDOW_SHIFT

```
#define FS65_W_FS_WD_WINDOW_SHIFT 0x04U
```

Configure the watchdog window duration. Duty cycle if set to 50 %

6.7.2.973 FS65_W_M_GO_LPOFF_LPOFF

```
#define FS65_W_M_GO_LPOFF_LPOFF (0x01U << FS65_W_M_GO_LPOFF_SHIFT)
```

Go to LPOFF mode and wait for wake-up event

6.7.2.974 FS65_W_M_GO_LPOFF_MASK

```
#define FS65_W_M_GO_LPOFF_MASK 0x20U
```

Configure the device in LPOFF-SLEEP

6.7.2.975 FS65_W_M_GO_LPOFF_NO_ACTION

```
#define FS65_W_M_GO_LPOFF_NO_ACTION (0x00U << FS65_W_M_GO_LPOFF_SHIFT)
```

No action

6.7.2.976 FS65_W_M_GO_LPOFF_SHIFT

```
#define FS65_W_M_GO_LPOFF_SHIFT 0x05U
```

Configure the device in LPOFF-SLEEP

6.7.2.977 FS65_W_M_INT_REQ_INT_REQ

```
#define FS65_W_M_INT_REQ_INT_REQ (0x01U << FS65_W_M_INT_REQ_SHIFT)
```

Request for an INT pulse

6.7.2.978 FS65_W_M_INT_REQ_MASK

```
#define FS65_W_M_INT_REQ_MASK 0x10U
```

Request for an INT pulse

6.7.2.979 FS65_W_M_INT_REQ_NO

```
#define FS65_W_M_INT_REQ_NO (0x00U << FS65_W_M_INT_REQ_SHIFT)
```

No Request

6.7.2.980 FS65_W_M_INT_REQ_SHIFT

```
#define FS65_W_M_INT_REQ_SHIFT 0x04U
```

Request for an INT pulse

6.7.2.981 FS65_W_M_LPOFF_AUTO_WU_LPOFF

```
#define FS65_W_M_LPOFF_AUTO_WU_LPOFF (0x01U << FS65_W_M_LPOFF_AUTO_WU_SHIFT)
```

Go to LPOFF mode and wake-up automatically after 1.0 ms

6.7.2.982 FS65_W_M_LPOFF_AUTO_WU_MASK

```
#define FS65_W_M_LPOFF_AUTO_WU_MASK 0x40U
```

Configure the device in LPOFF_AUTO_WU

6.7.2.983 FS65_W_M_LPOFF_AUTO_WU_NO_ACTION

```
#define FS65_W_M_LPOFF_AUTO_WU_NO_ACTION (0x00U << FS65_W_M_LPOFF_AUTO_WU_SHIFT)
```

No action

6.7.2.984 FS65_W_M_LPOFF_AUTO_WU_SHIFT

```
#define FS65_W_M_LPOFF_AUTO_WU_SHIFT 0x06U
```

Configure the device in LPOFF_AUTO_WU

6.7.2.985 FS65_W_M_VAUX_EN_DISABLED

```
#define FS65_W_M_VAUX_EN_DISABLED (0x00U << FS65_W_M_VAUX_EN_SHIFT)
```

Disabled

6.7.2.986 FS65_W_M_VAUX_EN_ENABLED

```
#define FS65_W_M_VAUX_EN_ENABLED (0x01U << FS65_W_M_VAUX_EN_SHIFT)
```

Enabled

6.7.2.987 FS65_W_M_VAUX_EN_MASK

```
#define FS65_W_M_VAUX_EN_MASK 0x20U
```

VAUX control (switch off not recommended if VAUX is safety critical)

6.7.2.988 FS65_W_M_VAUX_EN_SHIFT

```
#define FS65_W_M_VAUX_EN_SHIFT 0x05U
```

VAUX control (switch off not recommended if VAUX is safety critical)

6.7.2.989 FS65_W_M_VCAN_EN_DISABLED

```
#define FS65_W_M_VCAN_EN_DISABLED (0x00U << FS65_W_M_VCAN_EN_SHIFT)
```

Disabled

6.7.2.990 FS65_W_M_VCAN_EN_ENABLED

```
#define FS65_W_M_VCAN_EN_ENABLED (0x01U << FS65_W_M_VCAN_EN_SHIFT)
```

Enabled

6.7.2.991 FS65_W_M_VCAN_EN_MASK

```
#define FS65_W_M_VCAN_EN_MASK 0x10U
```

VCAN control

6.7.2.992 FS65_W_M_VCAN_EN_SHIFT

```
#define FS65_W_M_VCAN_EN_SHIFT 0x04U
```

VCAN control

6.7.2.993 FS65_W_M_VCCA_EN_DISABLED

```
#define FS65_W_M_VCCA_EN_DISABLED (0x00U << FS65_W_M_VCCA_EN_SHIFT)
```

Disabled

6.7.2.994 FS65_W_M_VCCA_EN_ENABLED

```
#define FS65_W_M_VCCA_EN_ENABLED (0x01U << FS65_W_M_VCCA_EN_SHIFT)
```

Enabled

6.7.2.995 FS65_W_M_VCCA_EN_MASK

```
#define FS65_W_M_VCCA_EN_MASK 0x40U
```

VCCA control (switch off not recommended if VCCA is safety critical)

6.7.2.996 FS65_W_M_VCCA_EN_SHIFT

```
#define FS65_W_M_VCCA_EN_SHIFT 0x06U
```

VCCA control (switch off not recommended if VCCA is safety critical)

6.7.2.997 FS65_W_M_VCORE_EN_DISABLED

```
#define FS65_W_M_VCORE_EN_DISABLED (0x00U << FS65_W_M_VCORE_EN_SHIFT)
```

Disabled

6.7.2.998 FS65_W_M_VCORE_EN_ENABLED

```
#define FS65_W_M_VCORE_EN_ENABLED (0x01U << FS65_W_M_VCORE_EN_SHIFT)
```

Enabled

6.7.2.999 FS65_W_M_VCORE_EN_MASK

```
#define FS65_W_M_VCORE_EN_MASK 0x80U
```

VCORE control (switch off not recommended if VCORE is safety critical)

6.7.2.1000 FS65_W_M_VCORE_EN_SHIFT

```
#define FS65_W_M_VCORE_EN_SHIFT 0x07U
```

VCORE control (switch off not recommended if VCORE is safety critical)

6.7.2.1001 FS65_W_M_WD_ANSWER_MASK

```
#define FS65_W_M_WD_ANSWER_MASK 0xFFU
```

WD answer from the MCU, Answer = (NOT(((LFSR x 4)+6)-4))/4

6.7.2.1002 FS65_W_M_WD_ANSWER_SHIFT

```
#define FS65_W_M_WD_ANSWER_SHIFT 0x00U
```

WD answer from the MCU, Answer = (NOT(((LFSR x 4)+6)-4))/4

Index