AN12866
Fault Collection and Control Unit (FCCU) Usage on S32V23x

Rev. 0 — November, 2020 Application Note

by: NXP Semiconductors

Contents

1 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. The FCCU offers a systematic approach to fault collection and control. No CPU intervention is required for fault collection and control operation, unless the FCCU is specifically configured to cause software intervention (by triggering IRQs or NMIs).

This application note describes how to use FCCU based on MCAL SW product for S32V23x. Please install EB21.0 and S32V234_MCAL4_2_RTM_HF1_1_0_1 on your computer before using the MCAL FCCU driver. EB and MCAL development environment setup are out of the scope of this document. It assumes that you are familiar with the MCAL software development environment and the document focuses on FCCU configurations in EB and usage in the application. FCCU also depends on other MCAL modules such as MCU, DET (see [1]for more details).

This document contains the following sections:

- FCCU functional description: This section introduces FCCU state machine and fault reaction mechanism.
- MCAL FCCU configuration: This section introduces how to config FCCU and different fault reactions in EB.
- Mcem APIs descriptions: This section describes functionalities of Mcem APIs.
- Use case examples: This section describes two use cases based on alarm interrupt and reset reaction of fault.
- Summary: This section describes the document scope and provides some implementation hints about FCCU usage for target application.

The following abbreviations are used in the application note.

Table 1. Acronyms and abbreviations

Abbreviations	Description
EOUT	Error Out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
NCF	Noncritical fault
NMI	Nonmaskable interrupt
WKPU	Wake-up unit
MCEM	Microcontroller Error Management

2 FCCU functional description

The following figure shows the architecture of FCCU module. At the heart of the FCCU is a Finite State Machine (FSM) around which, various signaling connections and logic modules are connected.

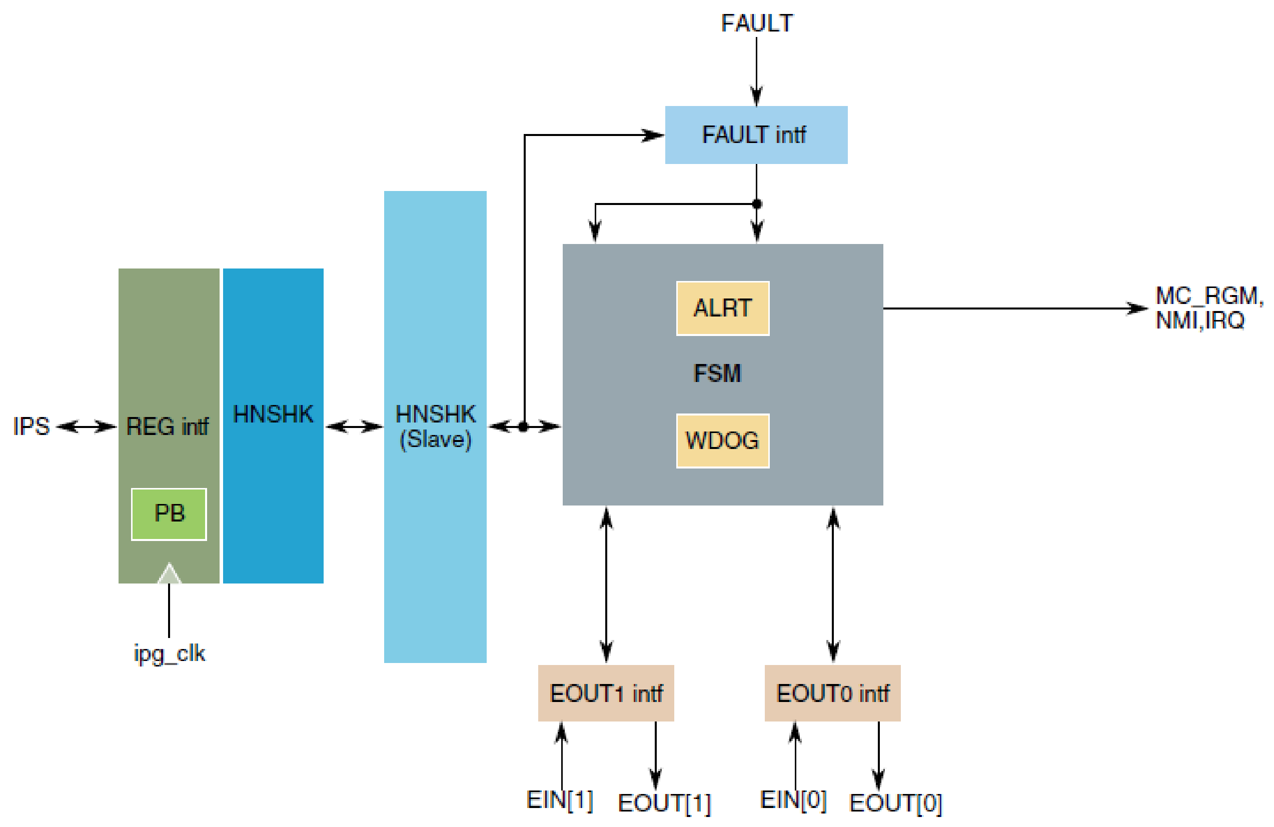


Figure 1. FCCU block diagram

Table 2. FCCU submodules

Submodule	Submodule Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of two asynchronous clocks(IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) : for reconfiguration phase, FCCU automatically leaves Configuration state and enters Normal state if WDG timeouts. • Alarm timer (ALRT): the timer starts when FCCU enters ALARM state if the faults are not removed in timer interval, FCCU leaves ALARM state and enters FAULT state.

Table continues on the next page...

Table 2. FCCU submodules (continued)

Submodule	Submodule Description
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

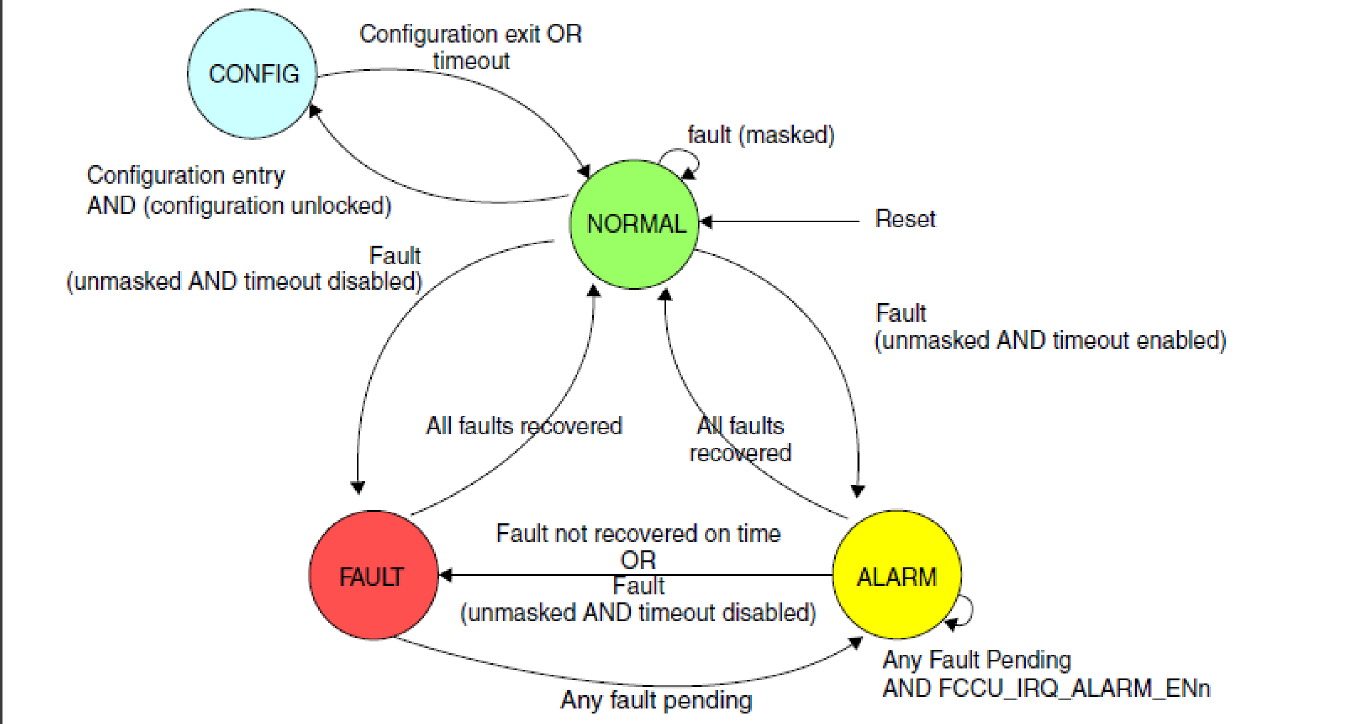

CONFIG
Configuration exit OR timeout
Configuration entry AND (configuration unlocked)
fault (masked)
NORMAL
Reset
Fault (unmasked AND timeout disabled)
Fault (unmasked AND timeout enabled)
All faults recovered
All faults recovered
FAULT
Fault not recovered on time OR Fault (unmasked AND timeout disabled)
ALARM
Any Fault Pending AND FCCU_IRQ_ALARM_ENn
Any fault pending

Figure 2. FCCU state diagram

Four stares are identified with following meaning:

- **CONFIG**: Put FCCU in configuration state to change its configuration. A subset of FCCU registers, dedicated to FCCU configuration can be accessed in write mode only in Configuration state.
- **NORMAL**: This is FCCU's operating state when no faults are occurring or all faults have been recovered. It is also the default state on the reset exit.
- **ALARM**: : FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state following the recovery from Fault state.
- **FAULT**: FCCU moves into Fault state:
  1. Timeout related to a noncritical fault when FCCU is in Alarm state.
  2. Unmasked noncritical faults with the timeout disabled.

The transition from Normal/Alarm to Fault state goes along with the generation of:

- **NMI interrupt (optional)**: Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction.

- **EOUT signaling (optional)**: EOUT[1:0] signals used to indicate FCCU's condition to off-chip logic. Different fault-output modes (protocols) for the fault-output (EOUT) pins are supported: Dual-Rail, Time-Switching, Bistable and Test (see section 73.8.8 of S32V234 RM for details).
- **SW option**: Soft reaction (Short functional reset).
- **SW option**: Hard reaction (Long functional reset).

MC_RGM will generate a short/long functional reset if a fault occurrs and configured with short/long reset reaction.

3 MCAL FCCU configurations

The FCCU and MEMU IPs of the S32V23x are used by MCEM driver. In this note, only the FCCU is discussed. FCCU driver is released as a part of the Mcem module in the MCAL package. FCCU configurations and the fault management could be configured in EB if the module is imported into a tresos project.

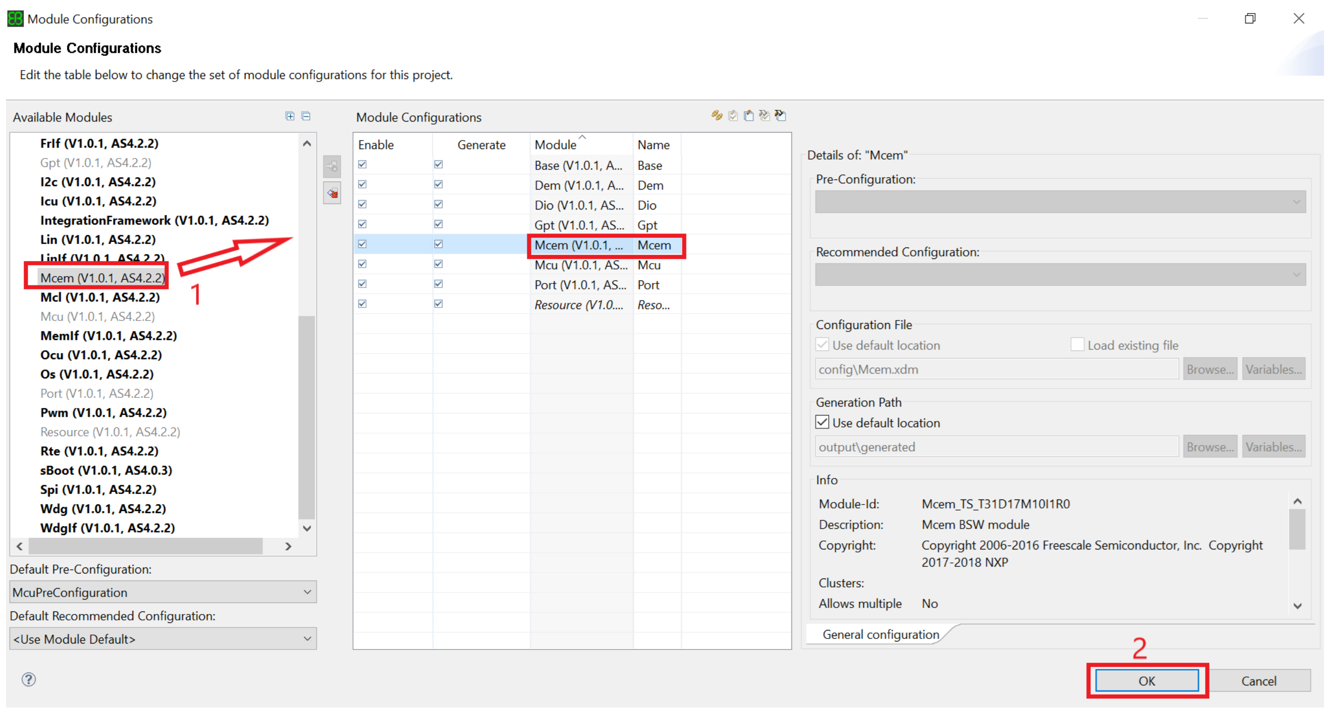
The MCEM plugin supports configuring the reaction to the faults in the system. The MCEM driver has the following major features:

1. Set the reaction time for faults.
2. Set the reaction type for each fault.
3. Set the recovery type for each fault.
4. Set the User Callback for ALARM state.
5. Set the type of external reporting of the error.

Mcem driver provides the functionality of initializing FCCU, clear FCCU fault, inject fake fault, FCCU alarm interrupt callback function definition and etc.

3.1 Configuring FCCU

You may enable Mcem in your existing EB project as shown in the following figure.



Module Configurations

Edit the table below to change the set of module configurations for this project.

Available Modules:
FrIf (V1.0.1, AS4.2.2)
Gpt (V1.0.1, AS4.2.2)
I2c (V1.0.1, AS4.2.2)
Icu (V1.0.1, AS4.2.2)
IntegrationFramework (V1.0.1, AS4.2.2)
Lin (V1.0.1, AS4.2.2)
LinIf (V1.0.1, AS4.2.2)
Mcem (V1.0.1, AS4.2.2)
Mcl (V1.0.1, AS4.2.2)
Mcu (V1.0.1, AS4.2.2)
MemIf (V1.0.1, AS4.2.2)
Ocu (V1.0.1, AS4.2.2)
Os (V1.0.1, AS4.2.2)
Port (V1.0.1, AS4.2.2)
Pwm (V1.0.1, AS4.2.2)
Resource (V1.0.1, AS4.2.2)
Rte (V1.0.1, AS4.2.2)
sBoot (V1.0.1, AS4.0.3)
Spi (V1.0.1, AS4.2.2)
Wdg (V1.0.1, AS4.2.2)
WdgIf (V1.0.1, AS4.2.2)

Default Pre-Configuration: McuPreConfiguration

Default Recommended Configuration: <Use Module Default>

Module Configurations:

Enable	Generate	Module	Name
✓	✓	Base (V1.0.1, A...	Base
✓	✓	Dem (V1.0.1, A...	Dem
✓	✓	Dio (V1.0.1, AS...	Dio
✓	✓	Gpt (V1.0.1, AS...	Gpt
✓	✓	Mcem (V1.0.1, ...	Mcem
✓	✓	Mcu (V1.0.1, AS...	Mcu
✓	✓	Port (V1.0.1, AS...	Port
✓	✓	*Resource (V1.0....*	*Reso...*

Details of: "Mcem"

Pre-Configuration:

Recommended Configuration:

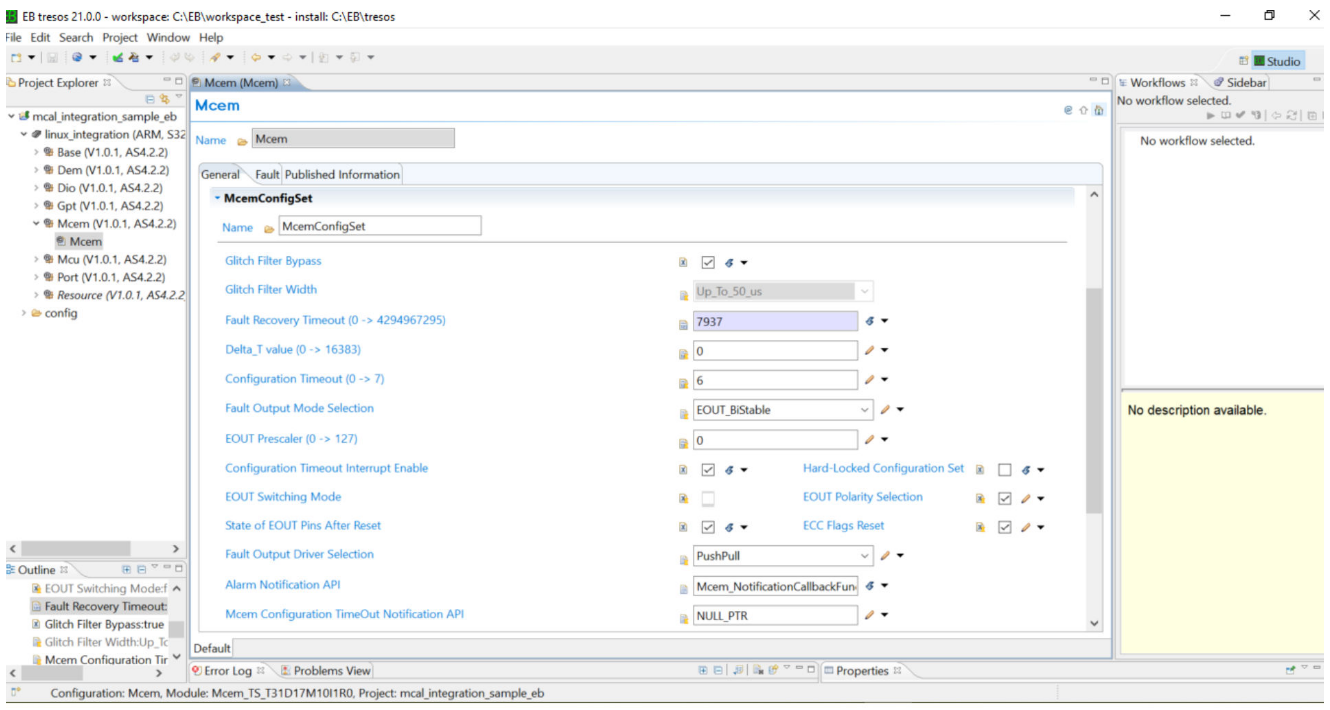
Configuration File: ✓ Use default location ☐ Load existing file
config\Mcem.xdm

Generation Path: ✓ Use default location
output\generated

Info

Module-Id:	Mcem_TS_T31D17M10I1R0
Description:	Mcem BSW module
Copyright:	Copyright 2006-2016 Freescale Semiconductor, Inc. Copyright 2017-2018 NXP
Clusters:	
Allows multiple	No

Figure 3. Enable mcem module in EB



EB tresos 21.0.0 - workspace: C:\EB\workspace_test - install: C:\EB\tresos

Mcem

Name: Mcem

McemConfigSet

Name	McemConfigSet
Glitch Filter Bypass	✓
Glitch Filter Width	Up_To_50_us
Fault Recovery Timeout (0 -> 4294967295)	7937
Delta_T value (0 -> 16383)	0
Configuration Timeout (0 -> 7)	6
Fault Output Mode Selection	EOUT_BiStable
EOUT Prescaler (0 -> 127)	0
Configuration Timeout Interrupt Enable	✓
Hard-Locked Configuration Set	☐
EOUT Switching Mode	☐
EOUT Polarity Selection	✓
State of EOUT Pins After Reset	✓
ECC Flags Reset	✓
Fault Output Driver Selection	PushPull
Alarm Notification API	Mcem_NotificationCallbackFun
Mcem Configuration TimeOut Notification API	NULL_PTR

Configuration: Mcem, Module: Mcem_TS_T31D17M10I1R0, Project: mcal_integration_sample_eb

Figure 4. Mcem config set

After enabling Mcem in the EB project, Mcem is now configurable. FCCU system parameters are configured in the McemConfigSet interface.

Fault Recovery Timeout – Non-critical fault timeout. If the fault with enabled recovery timeout is not recovered within the timeout inteval, the FCCU moves from the ALARM state to the FAULT state. NCF timeout can be computed using the following equations:

Timeout = TRC16 MHz × FaultTimeout

Timeout = 0.0625 × FaultTimeout [μs]

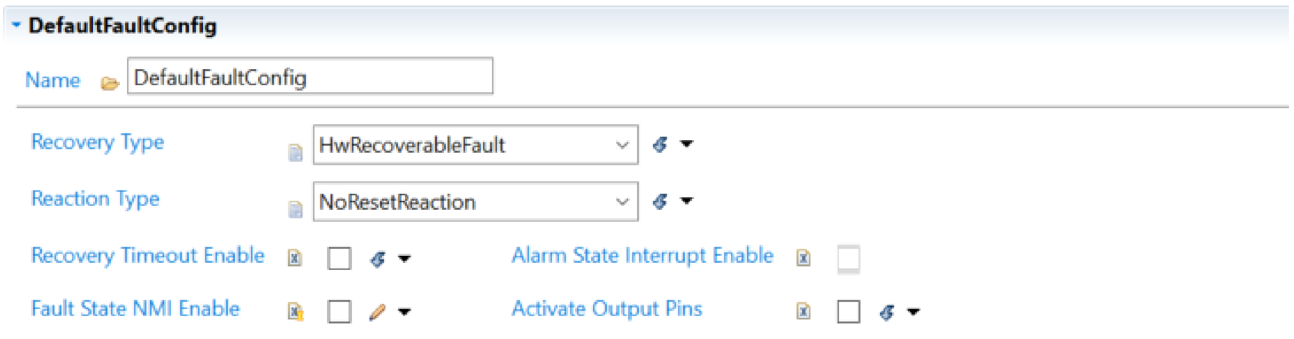
Please note that the timeout value shall always be less than FOCU count, which is fixed preprogrammed as 0x1F40. Otherwise, a destructive reset will be generated by a FOSU time-out (A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed). This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

Configuration Timeout – FCCU configuration timeout.

Fault Output Mode Selection – Select fault output mode as Daul-Rail, Time Switching or Bi-Stable.

Configuration Timeout Interrupt Enable – Enable timeout interrupt during mode configuration. If this item is checked, the timeout shall notify the application using this callback in **Mcem Configuration TimeOut Notification API**.

Alarm Notification API – This callback is used to notify the application in case of a failure event，whose reaction is configured as an alarm interrupt.



DefaultFaultConfig

Name: DefaultFaultConfig

Recovery Type: HwRecoverableFault

Reaction Type: NoResetReaction

Recovery Timeout Enable: ☐

Alarm State Interrupt Enable: ☐

Fault State NMI Enable: ☐

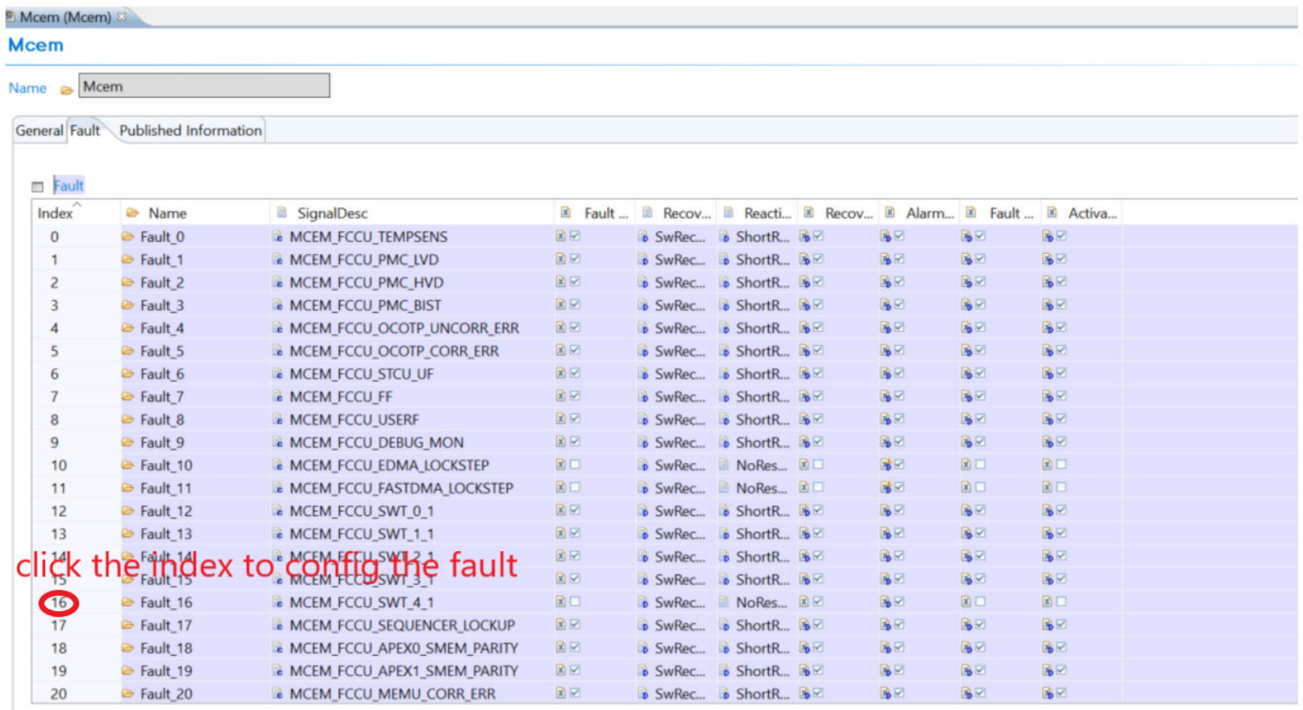
Activate Output Pins: ☐

Figure 5. Default fault config

If the faults are not configured in the Fault panel, they are configured as enabled in *DefaultFaultConfig* by default. It is recommended to disable the fault in Fault panel if you are not intended to enable this fault.

3.2 Configuring NCF reaction

All the faults are configurable in the Fault panel as shown in the following figure. Click the index to configure the fault.



Mcem

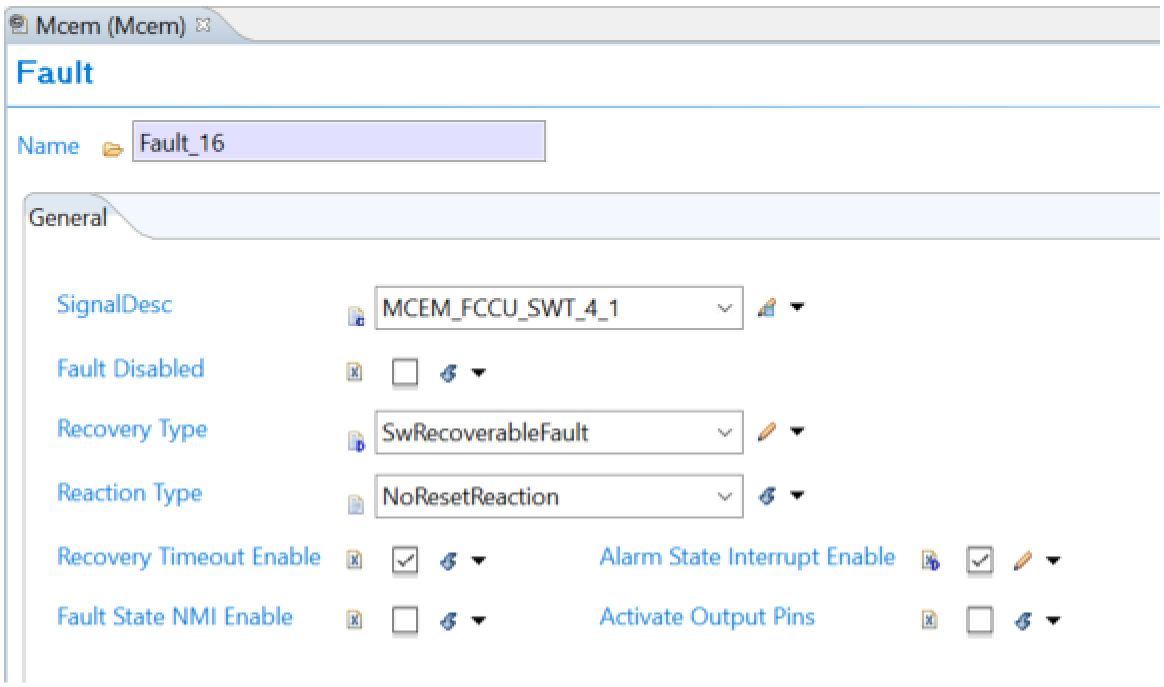
Name: Mcem

Fault

Index	Name	SignalDesc	Fault ...	Recov...	Reacti...	Recov...	Alarm...	Fault ...	Activa...
0	Fault_0	MCEM_FCCU_TEMPSENS	✓	SwRec...	ShortR...	✓	✓	✓	✓
1	Fault_1	MCEM_FCCU_PMC_LVD	✓	SwRec...	ShortR...	✓	✓	✓	✓
2	Fault_2	MCEM_FCCU_PMC_HVD	✓	SwRec...	ShortR...	✓	✓	✓	✓
3	Fault_3	MCEM_FCCU_PMC_BIST	✓	SwRec...	ShortR...	✓	✓	✓	✓
4	Fault_4	MCEM_FCCU_OCOTP_UNCORR_ERR	✓	SwRec...	ShortR...	✓	✓	✓	✓
5	Fault_5	MCEM_FCCU_OCOTP_CORR_ERR	✓	SwRec...	ShortR...	✓	✓	✓	✓
6	Fault_6	MCEM_FCCU_STCU_UF	✓	SwRec...	ShortR...	✓	✓	✓	✓
7	Fault_7	MCEM_FCCU_FF	✓	SwRec...	ShortR...	✓	✓	✓	✓
8	Fault_8	MCEM_FCCU_USERF	✓	SwRec...	ShortR...	✓	✓	✓	✓
9	Fault_9	MCEM_FCCU_DEBUG_MON	✓	SwRec...	ShortR...	✓	✓	✓	✓
10	Fault_10	MCEM_FCCU_EDMA_LOCKSTEP	☐	SwRec...	NoRes...	☐	✓	☐	☐
11	Fault_11	MCEM_FCCU_FASTDMA_LOCKSTEP	☐	SwRec...	NoRes...	☐	✓	☐	☐
12	Fault_12	MCEM_FCCU_SWT_0_1	✓	SwRec...	ShortR...	✓	✓	✓	✓
13	Fault_13	MCEM_FCCU_SWT_1_1	✓	SwRec...	ShortR...	✓	✓	✓	✓
14	Fault_14	MCEM_FCCU_SWT_2_1	✓	SwRec...	ShortR...	✓	✓	✓	✓
15	Fault_15	MCEM_FCCU_SWT_3_1	✓	SwRec...	ShortR...	✓	✓	✓	✓
16	Fault_16	MCEM_FCCU_SWT_4_1	☐	SwRec...	NoRes...	✓	✓	☐	☐
17	Fault_17	MCEM_FCCU_SEQUENCER_LOCKUP	✓	SwRec...	ShortR...	✓	✓	✓	✓
18	Fault_18	MCEM_FCCU_APEX0_SMEM_PARITY	✓	SwRec...	ShortR...	✓	✓	✓	✓
19	Fault_19	MCEM_FCCU_APEX1_SMEM_PARITY	✓	SwRec...	ShortR...	✓	✓	✓	✓
20	Fault_20	MCEM_FCCU_MEMU_CORR_ERR	✓	SwRec...	ShortR...	✓	✓	✓	✓

Figure 6. Fault config interface

Take an example of NCF[16], it is a fault caused by SWT4 first timeout.


Mcem (Mcem)

Fault

Name: Fault_16

General

SignalDesc	MCEM_FCCU_SWT_4_1
Fault Disabled	☐
Recovery Type	SwRecoverableFault
Reaction Type	NoResetReaction
Recovery Timeout Enable	✓
Alarm State Interrupt Enable	✓
Fault State NMI Enable	☐
Activate Output Pins	☐

Figure 7. NCF[16] config interface

- **SignalDesc** – Fault is selected here.
- **Fault Disabled** – Check to disable a fault, uncheck means enable the fault.
- **Recovery Type** – Defines fault recovery mode as hardware recoverable or software recoverable.

  HW recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault)has been removed.

  SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.
- **Reaction Type** – Set the fault state reaction type for fault. There are three reaction types for fault state no reaction, short reset reaction and long reset reaction.
- **Recovery Timeout Enable** – Enable/disable recovery timeout. FCCU transitions form ALARM to FAULT state if the respective fault is not recovered within the Recovery Timeout.
- **Alarm State Interrupt Enable** – Enable/disable alarm state interrupt for fault source.

  If you want to config fault reaction as IRQ, both Recovery Timeout Enable and ALARM State Interrupt Enable shall be enabled.
- **Fault State NMI Enable** – Enable/disable NMI for fault source.

NOTE

If FCCU faults are managed on M4 and configured as NMI reaction, please enable the bitfield FCCU_NMI_DIS of register SRC_GPR16 for M4.

- **Active Output Pins** – Enable/disable EOUT signal for fault source in fault state. The EOUT protocol is set in Fault Output Mode Selection.

4 Mcem APIs descriptions

Functionality of all APIs supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev002. This section introduces the basic APIs, for the full specification please see [3].

- Mcem_Init() – Initializes the MCEM driver according to the configuration parameters passed as an input.
- Mcem_CheckNMI() – Checks whether the initialized MCEM module is the source of the NMI request.

- Mcem_GetErrors() – Return logged error status. This function stores status of each NCF into a container provided by the application.
- Mcem_ClearFaults() – Clear software recoverable fault. When a software recoverable fault occurs, a software routine resolves the fault cause, then it clears the fault status using this function.
- Mcem_InjectFaults() – Inject a fake fault.
- Mcem_GetVersionInfo() – Get software version.

5 Use case examples

This section demonstrates the use of two fault reaction types, Fault reaction with ALARM IRQ and fault reaction with reset. The following figure shows the work flow of the use case examples.

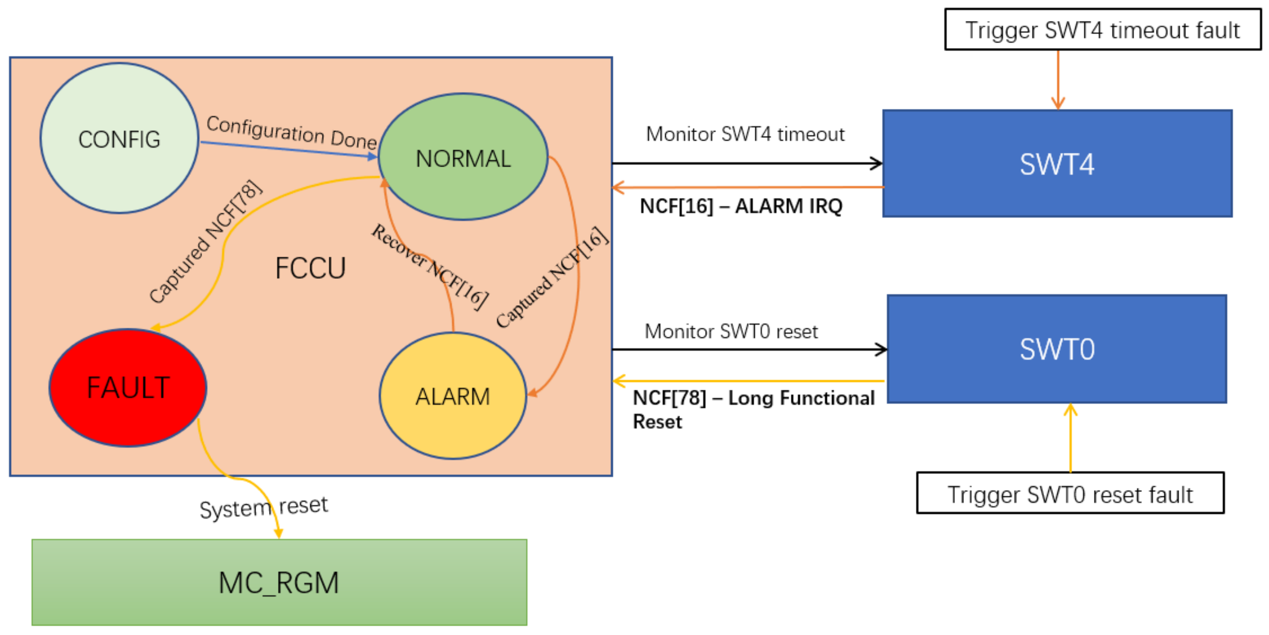


Figure 8. Use case workflow

5.1 ALARM IRQ reaction

This section describes the process to handle a Mcem IRQ reaction fault, take an example of SWT4 timeout fault (NCF[16]).

SWT4 is enabled via MCAL wdg driver, the timeout value is set as 500 ms and SWT4 is serviced every 200 ms.

```
/* Initialize SWT4 */
Wdg_43_Instance4_Init( &WdgSettingsConfig_4 );
Wdg_43_Instance4_SetMode( WDGIF_SLOW_MODE );
```

Firstly, Mcem shall be initailized. Mcem is configured as shown in Figure 4, Mcem is used to configure the SWT4 first timeout fault reaction as IRQ as per Figure 7.

```
/* Initialize Mcem */
Mcem_Init(&McemConfigSet);
```

If there is a fault configured with its reaction as IRQ, please enable FCCU ALARM interrupt, and Mcem interrupt service routine(ISR) *FCCU_ALARM_ISR* shall also be registered, e.g.

```
/* Register Handler of interrupt 74 */
NVIC_SetPriority( 74, 0xf0 );
RegisterISRHandler( 74, &FCCU_ALARM_ISR );
NVIC_EnableIRQ( 74 );
```

SWT4 timeout fault can be triggered manually, by modifying the SWT Time-out (SWT_TO) register to make the periodic cycle smaller than 200ms. If such a fault was injected, FCCU will capture such a fault, and trigger the ALARM IRQ.

As Alarm Notification API is configured as *Mcem_NotificationCallbackFunction,* and the prototype for the function is as follow.

```
void Mcem_NotificationCallbackFunction( Mcem_FaultType FaultID );
```

FaultID is the fault source caused FCCU entered ALARM state, the ID is consistent with S32V234_NCF_List.xslx. If removed the reference link in reference section, just add the description for its location as (is present as an attachment in the [2]) when it occurs firstly in this document. In this case, FaultID is 16.

Please clear the fault as the column “Fault Clearing Mechanism”described in S32V234_NCF_List.xlsx. For MCEM_FCCU_SWT_4_1, the fault shall be recovered first, then clear the corresponding fault status FCCU_NCF_S0[16]. The pseudocode is as follows.

```
/* disable swt4 */
(&INTERNAL_SWT4)->CR.B.WEN = false;
/* re-set swt4 timeout value */
(&INTERNAL_SWT4)->TO.B.WTO = SWT_TO_VALUE;
/* clear swt4 timeout interrupt flag */
(&INTERNAL_SWT4)->IR.B.TIF = 1;
/* re-enable swt4 */
(&INTERNAL_SWT4)->CR.B.WEN = true;
/*clear fault status*/
Mcem_ClearFaults( Fault_Id )
```

After recovering the fault, FCCU transitions to Normal state, please see following figure for more details.

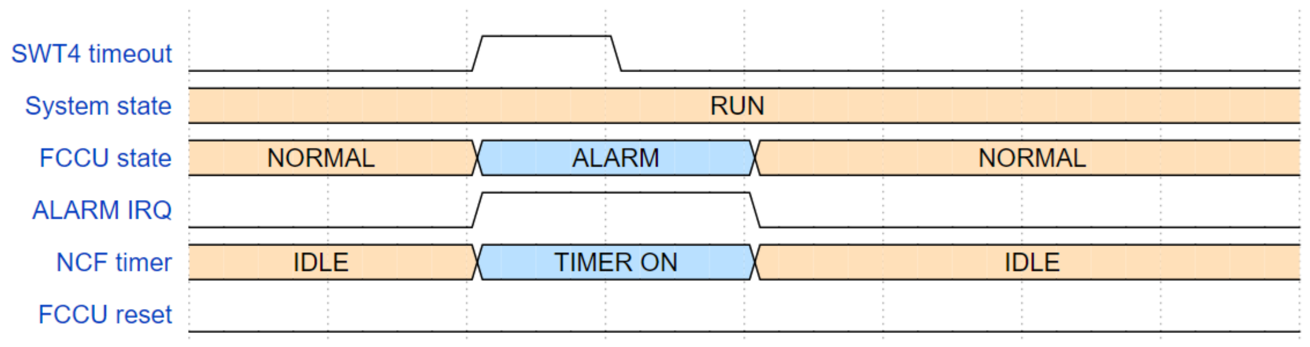
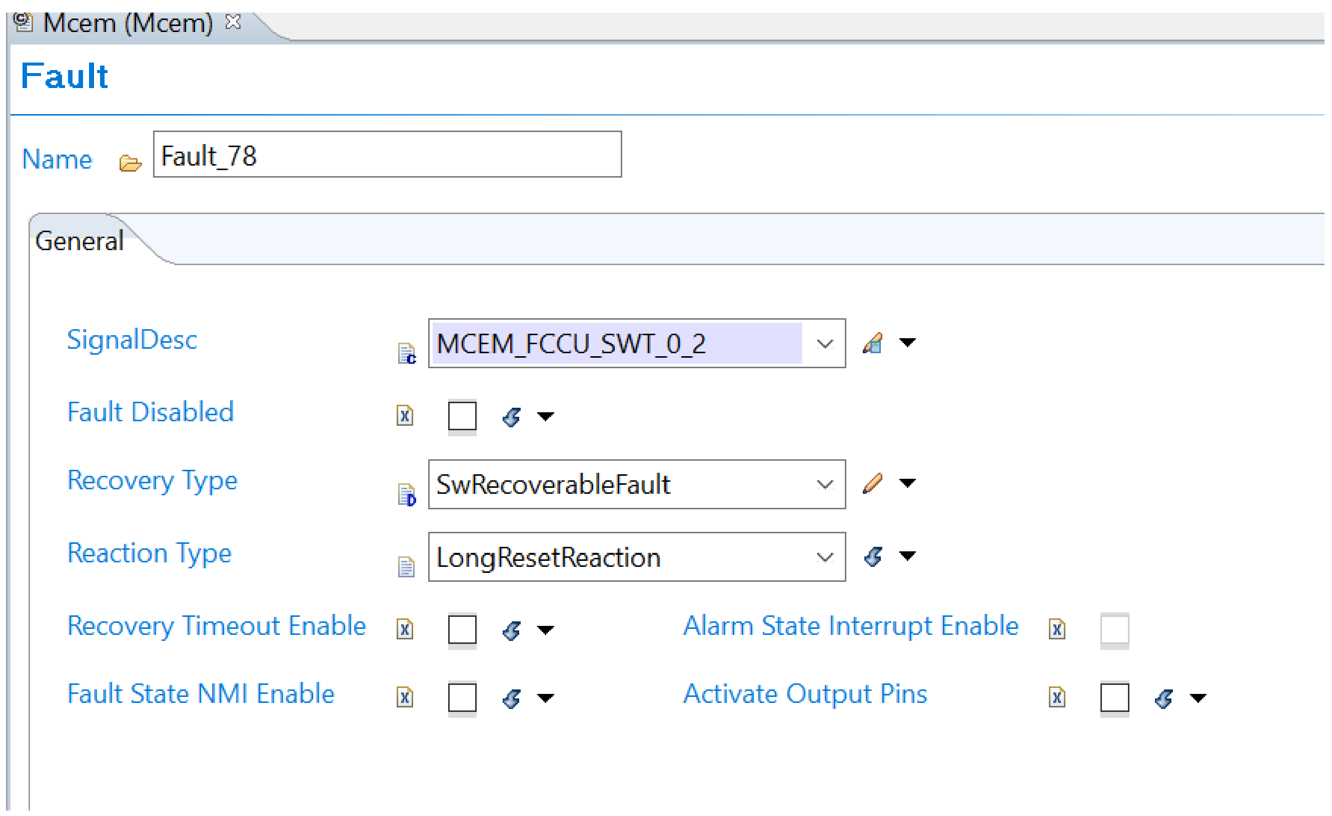


Figure 9. SWT4 timeout fault(Alarm State) recovery

5.2 Reset reaction

This section describes the use case of a Mcem reset reaction fault, take an example of SWT0 reset fault(SWT0 System Reset Request-NCF[78]).

NCF[78] shall be configured as the following figure. Here the reaction type is configured as Long Reset Reaction.


Mcem (Mcem)

Fault

Name	Fault_78

General

SignalDesc	MCEM_FCCU_SWT_0_2		
Fault Disabled	☐		
Recovery Type	SwRecoverableFault		
Reaction Type	LongResetReaction		
Recovery Timeout Enable	☐	Alarm State Interrupt Enable	☐
Fault State NMI Enable	☐	Activate Output Pins	☐

Figure 10. NCF[78] config interface

SWT0 shall be enabled before Mcem initialization.

```
/* Initialize SWT0 */
Wdg_43_Instance0_Init( &WdgSettingsConfig_0 );
Wdg_43_Instance0_SetMode( WDGIF_SLOW_MODE );
/* Initialize Mcem */
Mcem_Init(&McemConfigSet);
```

Assuming that the fault is triggered by swt0 timeout, after the FCCU captures this fault, the system will reset, as shown in the following figure. To validate the correct reset reaction, the MC_RGM_FESS register can be checked after system restart, SS_FCCU_HARD bit will be set.When the SS_FCCU_HARD bit is set, the FCCU has successfully triggered a long functional reset.

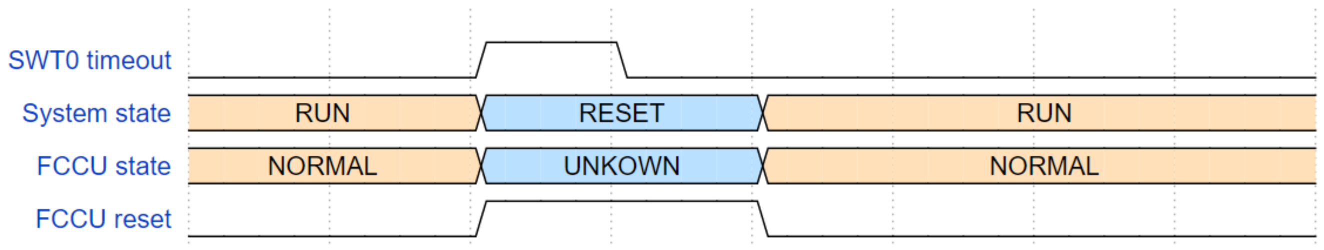


Figure 11. SWT0 timeout fault recovery

6 Summary

FCCU features can be enabled and integrated in various software environments. NXP provides a FCCU driver in the S32V234 ISO26262 MCAL product, which is developed according to the SEooC requirements of ISO26262. This document describes the usage of FCCU based on MCAL. If you are intended to implement FCCU features in other software environments, please refer to section 73.11 in [2] for more guidelines and recommendations.

All faults detected by hardware measures are reported to the FCCU. There are total of 128 NCFs which could be monitored by FCCU. Depending on the type of fault, the FCCU places the device into an appropriate safe state. To achieve this, application software is required to configure FCCU appropriately.

The details of 128 NCFs are described in S32V234_NCF_List.xlsx, including the fault clearing mechanism and recommended reactions for each fault.

In this document, the implementations of alarm reaction and reset reaction for two faults are introduced. About the EOUT reaction, failure of the S32V23x is signaled to one or two pins (FCCU_F0 and FCCU_F1) for external failure indication. If the S32V23x signals an internal failure on its error out signals, the system may no longer rely on the integrity of the other S32V23x outputs for safety functions. The overall system needs to include measures to monitor FCCU_Fn of the S32V23x and move the system to a safe state when an error is indicated.

In summary, FCCU provides a flexible mechanism for users to manage faults. It is the responsibility of the application software to manage the fault reaction according to the specific safety requirements.

7 Reference

1. Integration Manual for S32V234 Mcem Driver, IM17MCEMASR4.2 Rev002R1.0.1,Rev. 1.0 (included in AUTOSAR 4.2 MCAL (ISO 26262) for S32V234)
2. S32V234 Reference Manual,S32V234RM, Rev. 5, 11/2019
3. User Manual for S32V234 Mcem Driver,UM17MCEMASR4.2 Rev002R1.0.1, Rev. 1.0(included in AUTOSAR 4.2 MCAL (ISO 26262) for S32V234)

Date of release: November, 2020

Document identifier: AN12866