MPC5744P TRAINING

Automotive Microcontroller and Processor BL
Stefan Ruan(阮小飞) | Application Engineer

JUNE. 2016

NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD

Agenda

- Overview
- Development Tools and Support
- Peripheral
  - Clock, PMC, Timer, FCCU, SIUL, Interrupt, FlexCAN, FlexPWM, eTimer, CTU, ADC, SGEN, etc.
- Demonstration with Low-level driver
- Function Safety Activities at NXP

OVERVIEW

What's a Power Architecture™ MCU?

An industry benchmark architecture, very successful in embedded systems from automotive powertrain, navigation systems to netcom applications

A long term partnership between FSL and IBM

The 32bit architecture of choice for FSL in automotive applications

Single issue, multiple execution unit, 32 bit RISC CPU

Why 32bit?

- Evolving software development methodologies
  - More structured software architectures engineered for reuse creating overhead
    Example: AUTOSAR standardization adding about 10-15% performance overhead.
  - Linear paged architecture and 32bit data handling allows for easier implementation of model-based designs and autocoding.

- Scalability upwards allowing platform designs from low- to high-end systems.

- More affordable as technology shrinks the cost delta between 32bit and 16bit architectures reduces.

SafeAssure - *Simplification*

- SafeAssure products are conceived to **simplify** system level functional safety **design** and cut down time to **compliance**

- Component safety measures **augment** system level safety measures

- Key functional safety activities addressed
  - Safety analysis *(FMEA, FTA, FMEDA)*
  - Hardware integration *(Safety Manual)*
  - Software integration *(Safety Manual)*
  - Support interface *(Roles & Responsibilities)*

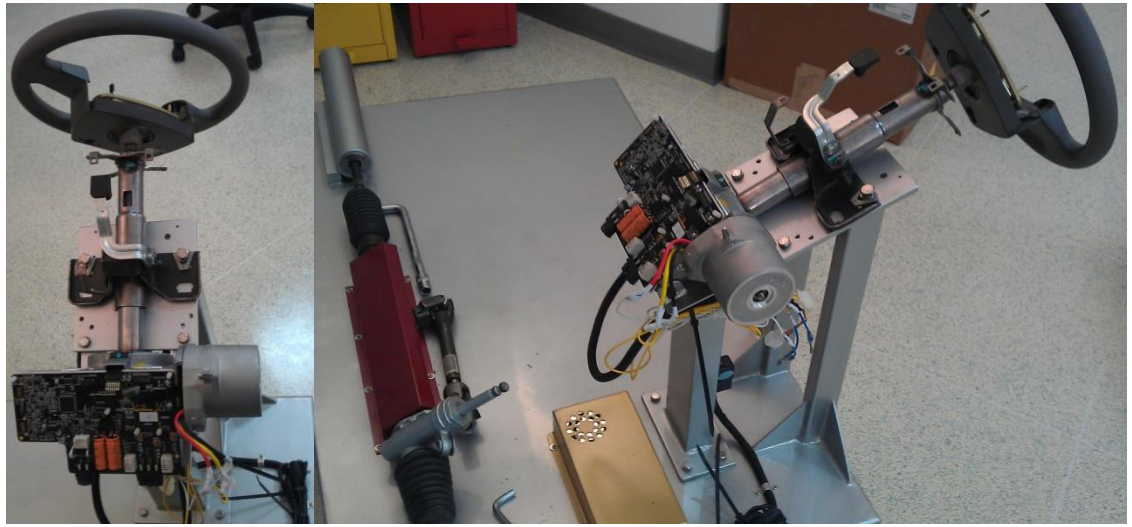
MPC5744P --ASIL D

- Single-point fault metric (SPFM)
- Latent-fault metric (LFM)
- Probabilistic Metric for random Hardware Failures (PMHF)

	ASIL B	ASIL C	ASIL D
SPFM	>90%	>97%	>99%
LFM	>60%	>80%	>90%
PMHF [1/h]	$<10^{-7}$	$<10^{-7}$	$<10^{-8}$

Target Applications

- Chassis applications
- Electronic power steering(EPS)
- Suspension
- Braking
- Airbag and sensor fusion applications
- HEV/EV: Inverter, VCU, BMS



Dual Core Architecture - Safety Elements

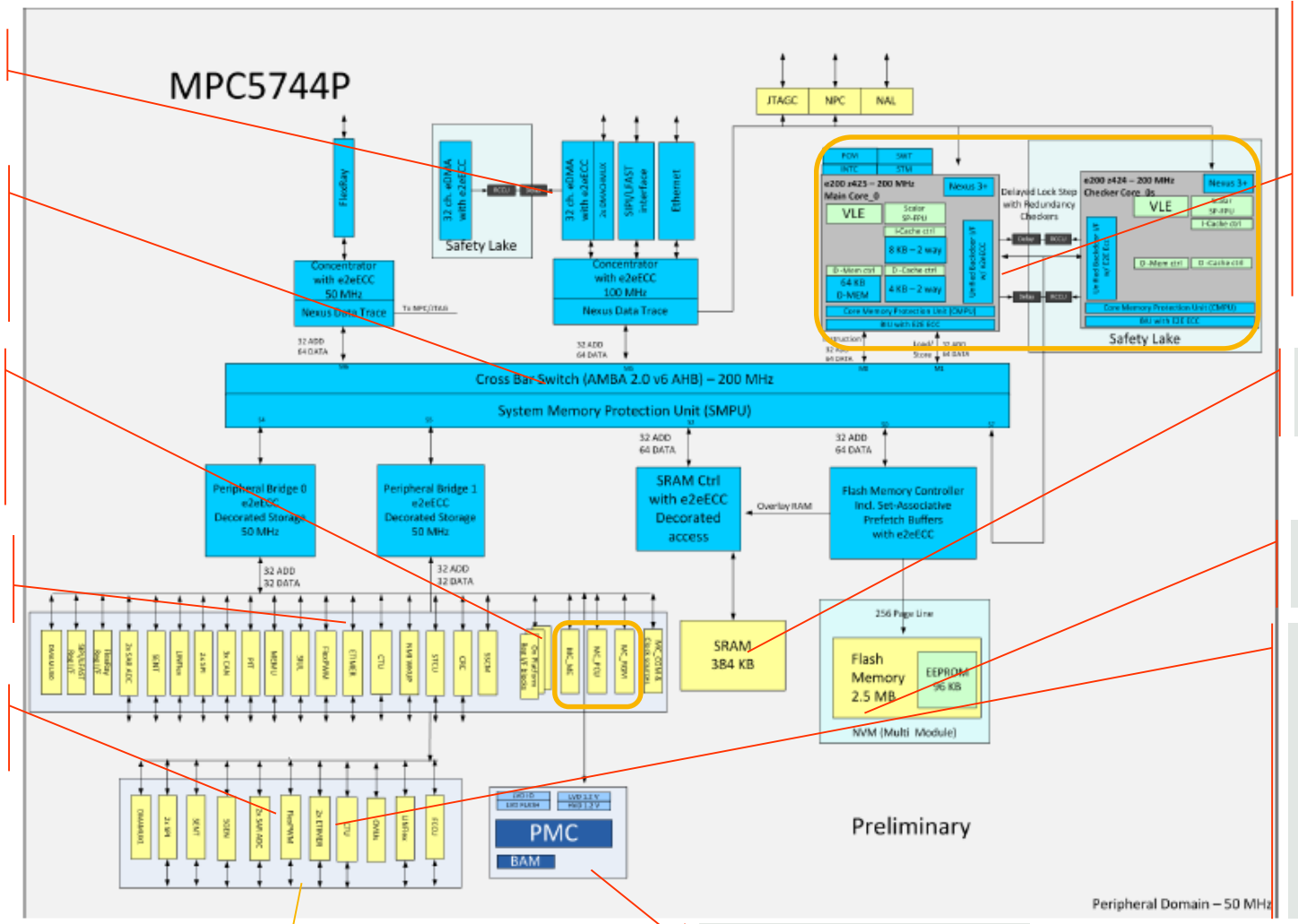
Fault Com. Tolerant

XBAR MPU:
- *redundant*
- *RC Units at Gates to non redundant sphere*

Clock Monitoring
- *Detects and mitigates clock disturbances*
- PLL

CRC Unit
- *Application Signature*

ADC
- *Hardware BIST*



Redundant use of IO & Application checks

Processing:
Dual Core Lockstep

Sphere of Replication:
- *Replicated e200Core*
- *replicated eDMA*
- *redundant INTC, SWT, etc*
- *redundant MMU*

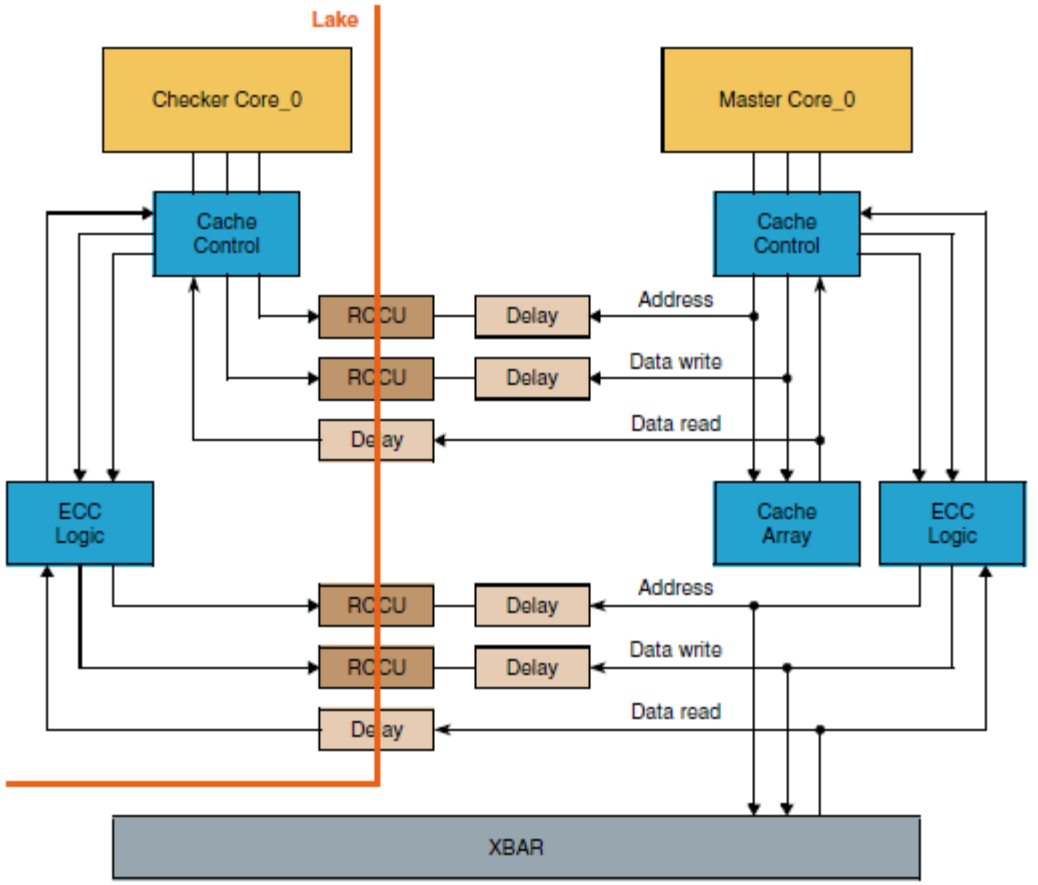
RAM
- *ECC(MEMU)*

Flash
- *ECC(MEMU)*

Fault Collection and Control Unit
- *detects when errors have occurred*
- *indicates error to external*
- *independent of software operation*

Power Monitoring
- *internal Vreg*
- *redundant Vmonitor*

Dual core concept



MPC5744P Feature Summary

Feature	Details
CPU	Power Architecture 2 x e200z4 in delayed lock step
Architecture	Harvard
Execution speed	0 MHz to 200 MHz (+2% FM)
System MPU	Yes (16 regions)
Memory	
Code/data flash memory	2.5 MB, ECC, RWW
Data flash memory	Supported with RWW
SRAM	384 KB, ECC
Modules	
Interrupt controller	32 interrupt priority levels, 16 SW programmable interrupts
PIT	1 module with 4 channels
System Timer Module (STM)	1 module with 4 channels
Software Watchdog Timer (SWT)	Yes
FlexRay	1 module with 64 message buffer, dual channel
FlexCAN	3 modules with 64 message buffer
LINFlexD	(UART and LIN with DMA support) 2 modules

MPC5744P Feature Summary

Feature	Details
Modules	
Fault Control and Collection Unit (FCCU)	Yes
Cross Triggering Unit (CTU)	2 modules
eTimer	3 modules with 6 channels
FlexPWM	2 modules with 4 x (2+1) channels
Analog-to-digital converter (ADC)	4 modules with 12-bit ADC, each with 16 channels(25 external channels including shared channels plus internal channels)
SPI	4 modules As many as 8 chip selects
Supply	
Device Power Supply	3.3 V with external ballast transistor 3.3 V with external 1.25 V low drop-out (LDO) regulator
ADC Analog Reference voltage	3.15 V to 3.6 V and 4.5 V to 5.5 V
Low power modes	
HALT and STOP	Yes
Package	LQFP 144 / MAPBGA 257

MPC574xP Family Members

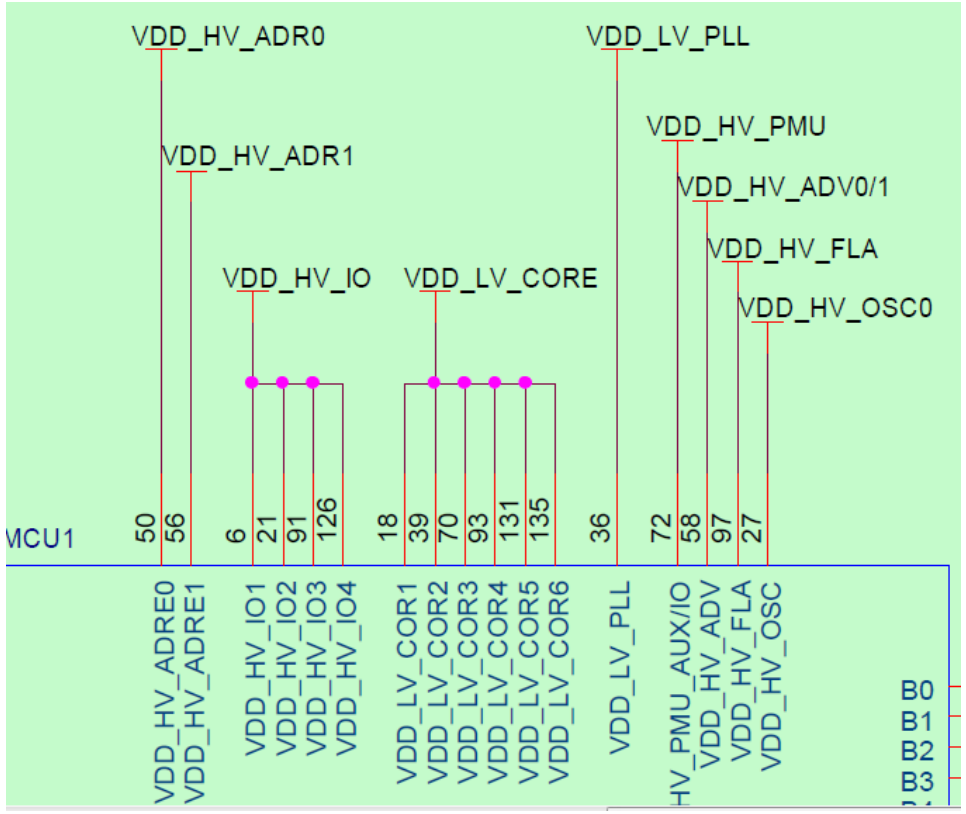
	Internal RAM	Internal Flash	Core: Number of cores	Ambient Operating Temperature	Package Type and Termination Count	Operating Frequency
MPC5741P	128 KB	1000 KB	2	-40 °C to 125 °C	144 LQFP/257 MAPBGA	200 MHz
MPC5742P	192 KB	1500 KB	2	-40 °C to 125 °C	144 LQFP/257 MAPBGA	200 MHz
MPC5743P	256 KB	2000 KB	2	-40 °C to 125 °C	144 LQFP/257 MAPBGA	200 MHz
MPC5744P	384 KB	2500 KB	2	-40 °C to 125 °C	144 LQFP/257 MAPBGA	200 MHz

MPC574xP feature differing by package

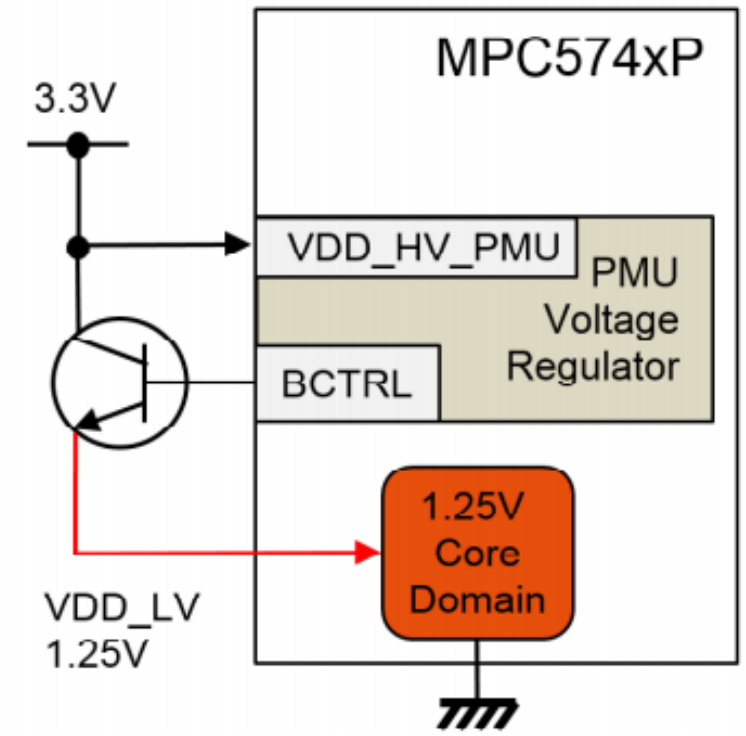
Feature	144 LQFP	257 MAPBGA
FlexPWM1	A[0-2]/B[0-2]	A[0-3]/B[0-3]/X[0-3]/Fault[0:3]
eTimer2	ETC2-5	ETC0-5
GPIO	79 GPIO, 26 GPI	112 GPIO, 29 GPI
DSPI3	No external signals	CS0-3
CTU external trigger(s)	CTU0	CTU0, CTU1
ADCs	22 analog pads assigned to ADC0, ADC1, ADC2, and ADC3 Shared channels between ADC0/ADC1, ADC0/ADC2, and ADC1/ADC3	25 analog pads assigned to ADC0, ADC1, ADC2, and ADC3 Shared channels between ADC0/ADC1, ADC0/ADC2, ADC1/ADC3, and ADC2/ADC3
SIPI/LFAST	No	Yes
Ethernet	No	Yes
Nexus	Yes (MDO interface)	Yes (MDO interface)
Nexus Aurora Port	No	Yes (TX0/TX0_P/TX1/TX1_P/CLK/CLK_P)

Power Supplies

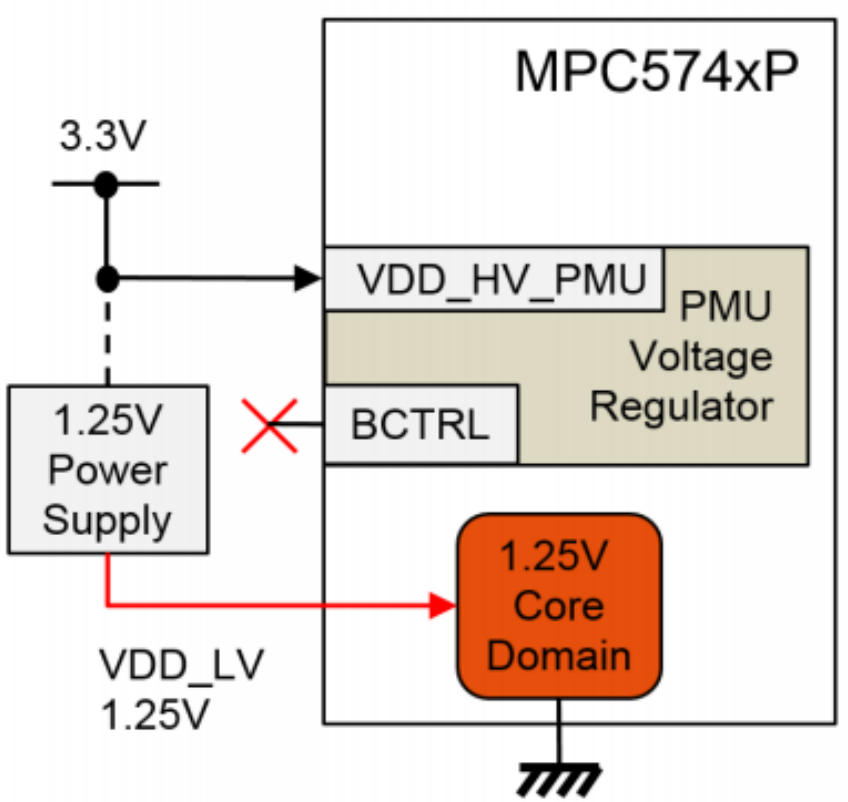
Name	Type	Description
VDD_HV_ADRE0[1]	Reference (3.3 V or 5 V)	ADC0 high reference voltage
VDD_HV_ADRE1[1]	Reference (3.3 V or 5 V)	ADC1 high reference voltage
VDD_HV_ADV	Supply (3.3 V)	High voltage supply for the ADC modules
VDD_HV_IO	Supply (3.3 V)	High voltage power supply for the I/Os
VDD_HV_PMU	Supply (3.3 V)	High voltage power supply for the internal Power Management Unit (PMU)
VDD_HV_OSC	Supply (3.3 V)	High voltage power supply for the internal crystal oscillator circuitry
VDD_HV_FLA	Supply (3.3 V)	High voltage supply for the internal Flash memory
VDD_LV_LFAST[2]	Supply (1.25 V)	Low voltage power supply for the LFAST module
VDD_LV_NEXUS[2]	Supply (1.25 V)	Low voltage power supply for the Nexus module
VDD_LV_PLL	Supply (1.25 V)	Low voltage power supply for the PLLs
VDD_LV_COR	Supply (1.25 V)	Low voltage power supply for the core digital logic
VSS_LV_LFAST[2]	Ground	Ground supply for the LFAST module
VSS_LV_NEXUS[2]	Ground	Ground supply for the Nexus module
VSS_LV_PLL	Ground	Ground supply for the PLLs
VSS_LV_COR	Ground	Ground supply for the core digital logic
VSS_HV_IO	Ground	Ground supply for the I/Os
VSS_HV_OSC	Ground	Ground supply for the oscillator
VSS_HV_ADRE0	Ground/Reference	ADC0 ground and low reference voltage
VSS_HV_ADRE1	Ground/Reference	ADC1 ground and low reference voltage
VSS_HV_ADV	Ground	Ground supply for the ADC modules



Power Supplies



Using an external ballast transistor to generate 1.25 V



Supplying 1.25 V with an external power supply

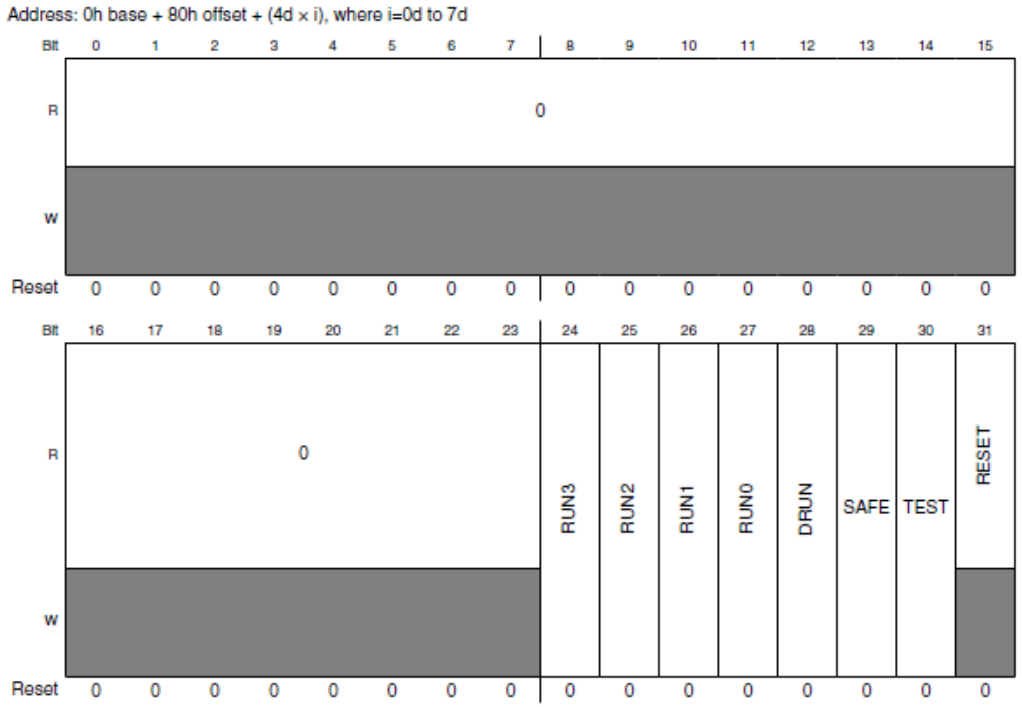
e200z4 core feature

- The e200z425 processor family is a set of CPU cores that implement low-cost versions of the *PowerISA 2.06* architecture.
- The e200z425 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general purpose registers (GPRs).
- The e200z425 core implements the VLE (variable-length encoding) ISA, providing improved code density.
- An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.
- A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The e200z425 also contains an 8 KB Instruction Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.
- A Memory Protection Unit is also included which supports protections of various instruction and data memory areas.

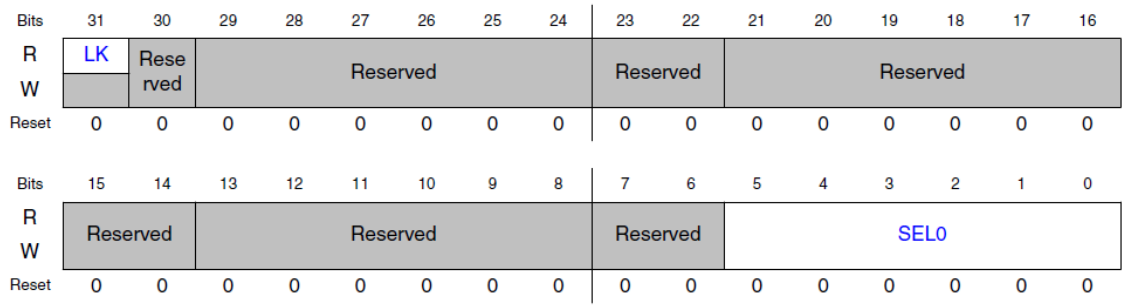
Register- Big endian

- Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Power architecture



ARM architecture



DEVELOPMENT TOOL AND SUPPORT

Evaluation Kit

Sample available

- SPC5744PFK1AMLQ9 (144LQFP package) - available now.
- SPC5744PGK1AMMM9 (257MAPBGA package) - available in MAY 2016.

Full Evaluation Kit

- Evaluation system (main module, mini module) allows full access to the CPU, all of the CPU's I/O signals, and the motherboard peripherals (such as CAN, SCI, LIN).

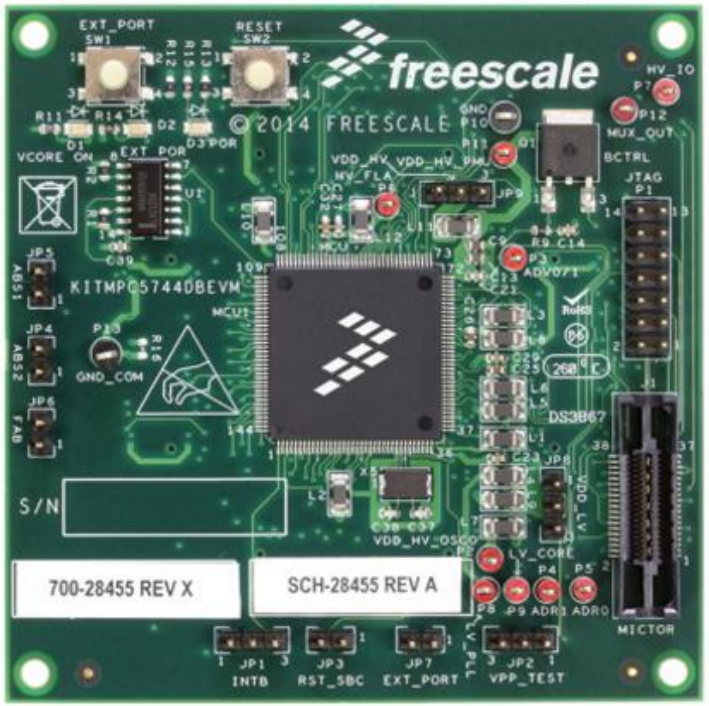
▶Daughter board
KITMPC5744DBEVM

▶Mother board
KIT33908MBEVBE

P&E Hardware Interface Cable

- P&E- Multilink Universal / FX

KITMPC5744DBEVM Evaluation Daughter Board



KIT33908MBEVBE Evaluation Mother Board

EVB

- The MPC574xPEVB is an evaluation system supporting NXP MPC574xP MCU family.

- Mother board

  -MPC57XXXMB

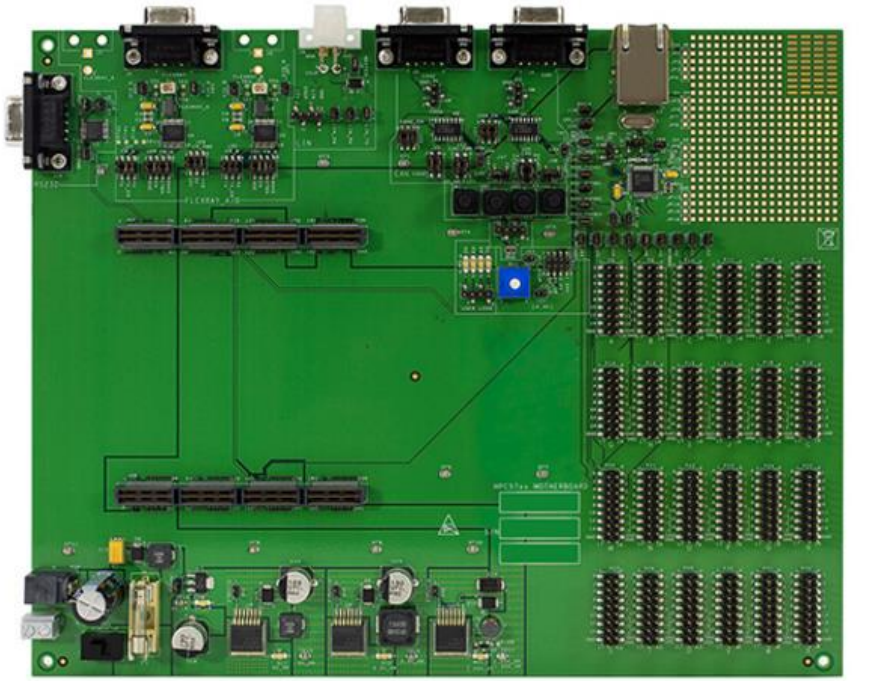
- Daughter board

  -MPC5744P-144DS

  -MPC5744P-257DS.

LINK

MPC57XXXMB MOTHERBOARD



MPC5744P-257DC Adapter

MPC5744P-144DC Adapter

User's Guide

Qorivva MPC5744P Evaluation Board 144LQFP Expansion Board User's Guide

by: Barbara Johnson
Applications Engineering

1 Introduction

This document describes the Qorivva MPC5744P evaluation board (EVB) expansion board for the 144LQFP (part number MPC5744P-144DC). The EVB is targeted at providing a platform for the evaluation and development of the MPC5744P automotive MCU, facilitating hardware and software development as well as debugging. Settings for switches, jumpers, LEDs, and push-buttons are shown for basic operation of the prototype version of the EVB.

Contents

Qorivva MPC5744P Evaluation Board 257BGA Expansion Board User's Guide

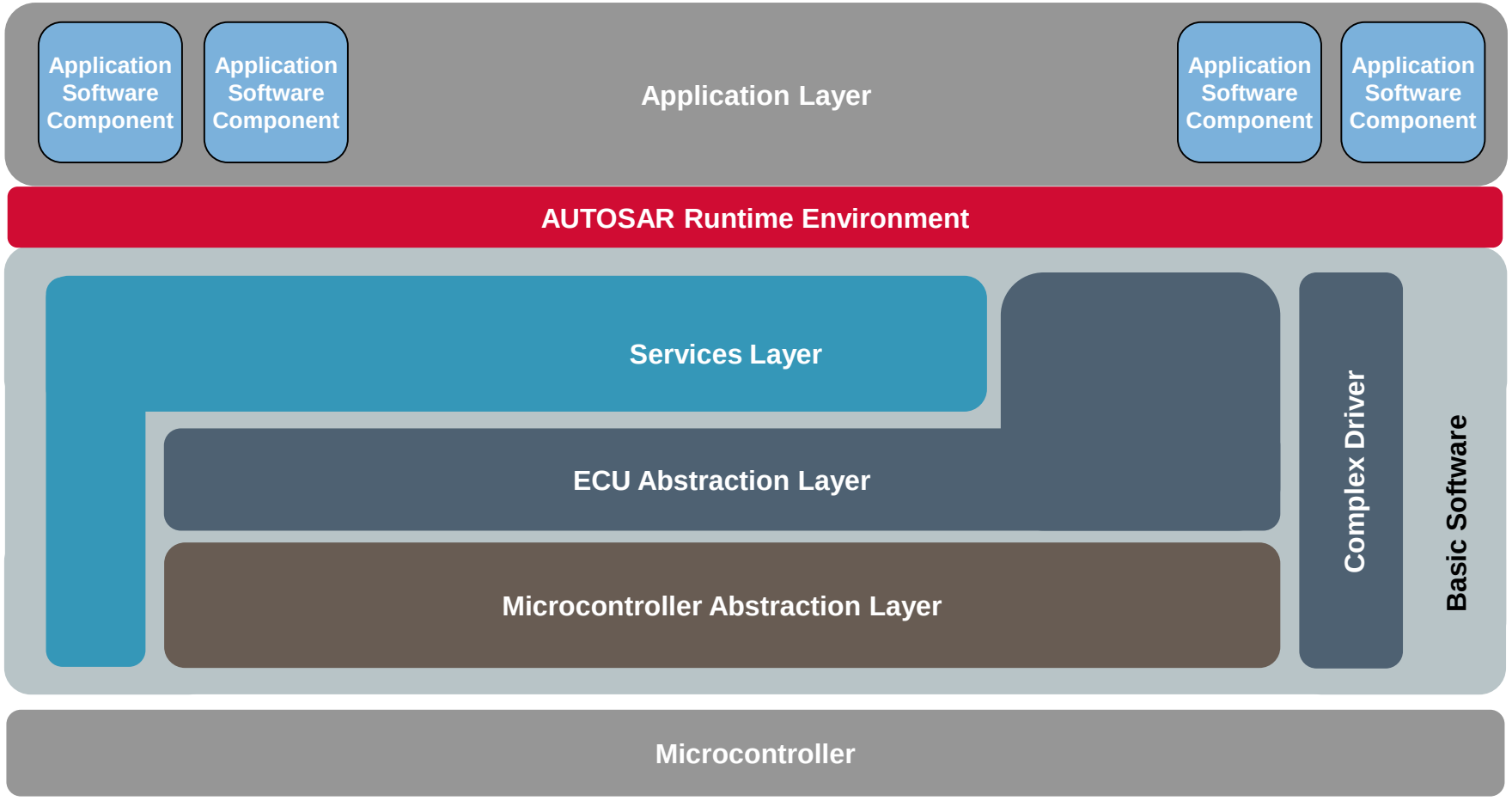
1 Introduction

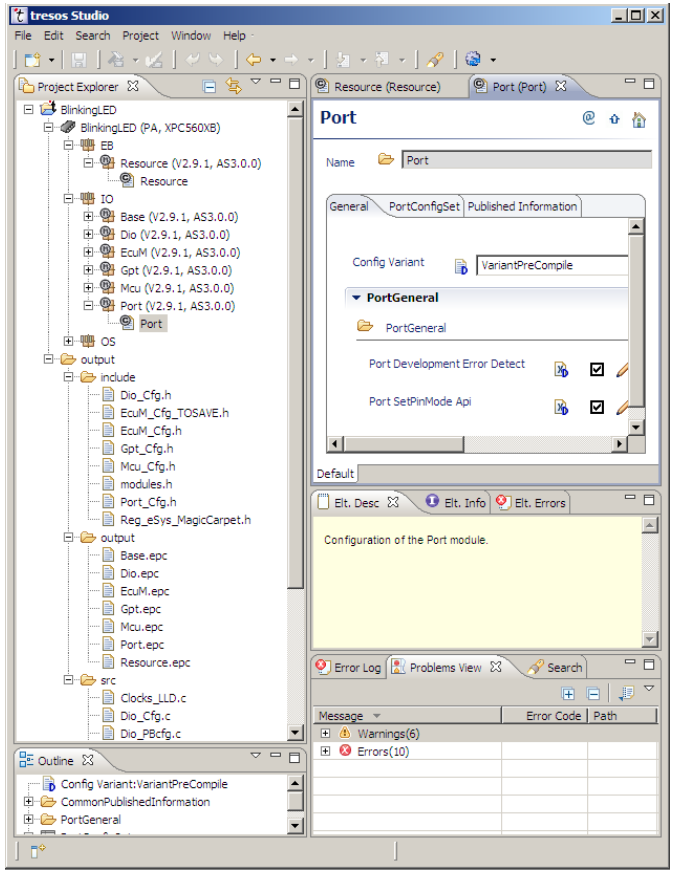
This document describes the Qorivva MPC5744P evaluation board (EVB) expansion board for the 257BGA (part number MPC5744PE257DC). The EVB is targeted at providing a platform for the evaluation and development of the MPC5744P automotive MCU, facilitating hardware and software development as well as debugging. Settings for switches, jumpers, LEDs, and push-buttons are shown for basic operation of the prototype version of the EVB.

Contents

MPC57xx Power Architecture Tools Support

AUTOSAR - MCAL4.0





AUTOSAR BSW Architecture – Basic Layers

What is S32DS: Basic Tool Frame Work

S32 = new unified automotive processors brand
e.g. Freescale S32K (Kinetis)

Existing Products

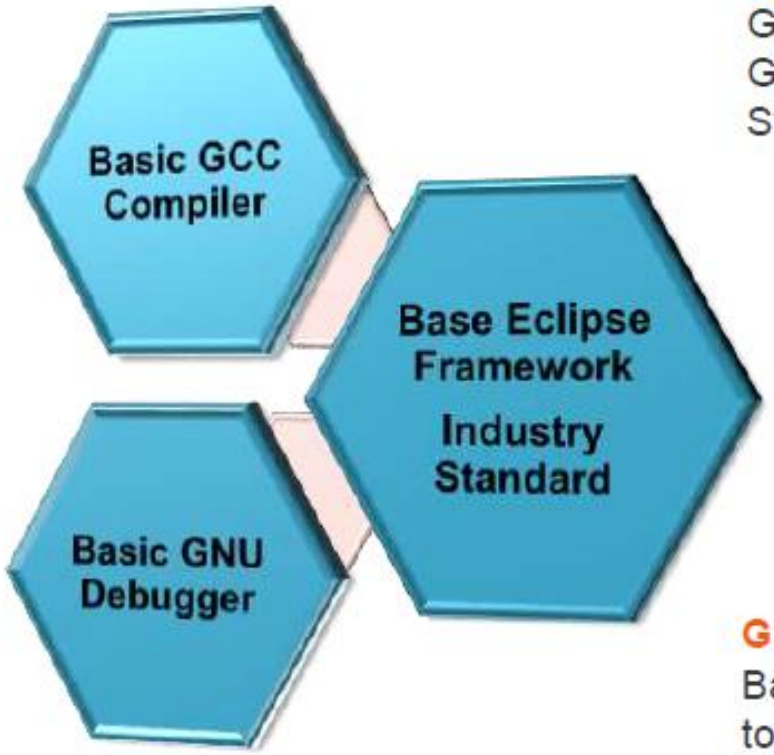
ARM®-based processors
MPC57xx, MPC56xx, MPC55xx MCUs
Power Architecture-based processors
S12(X) MCUs
S12 MagniV mixed-signal MCUs
Image Cognition Processors
Kinetis auto MCUs
S08 MCUs
ARM Cortex-based MCUs
MAC57Dxx MCUs
Others

New Products

S32 Automotive Processors

DS = Design Studio
for Automotive & Ultra-Reliable Industrial Processors

What is S32DS: Basic Tool Frame Work



GNU C/C++ Compiler
GCC from **ARM Cortex-M** and **A** cores
GCC for **Power Architecture**
Standard Solid Compilers

Basic Eclipse Backplane
C/C++ Development Tools
Integrated Editor with C/C++ tools
Managed Make Facility
Over 10,000 Eclipse Plug-in Available

GNU GDB Debugger
Basic Debugger interface
to low cost JTAG debugger
(**Pemicro** , **Segger**, and **OpenOCD**)

What is S32DS: Premium Compiler/Debugger Support

Third-Party Premium Compiler Support
ISO Certified compilers to support

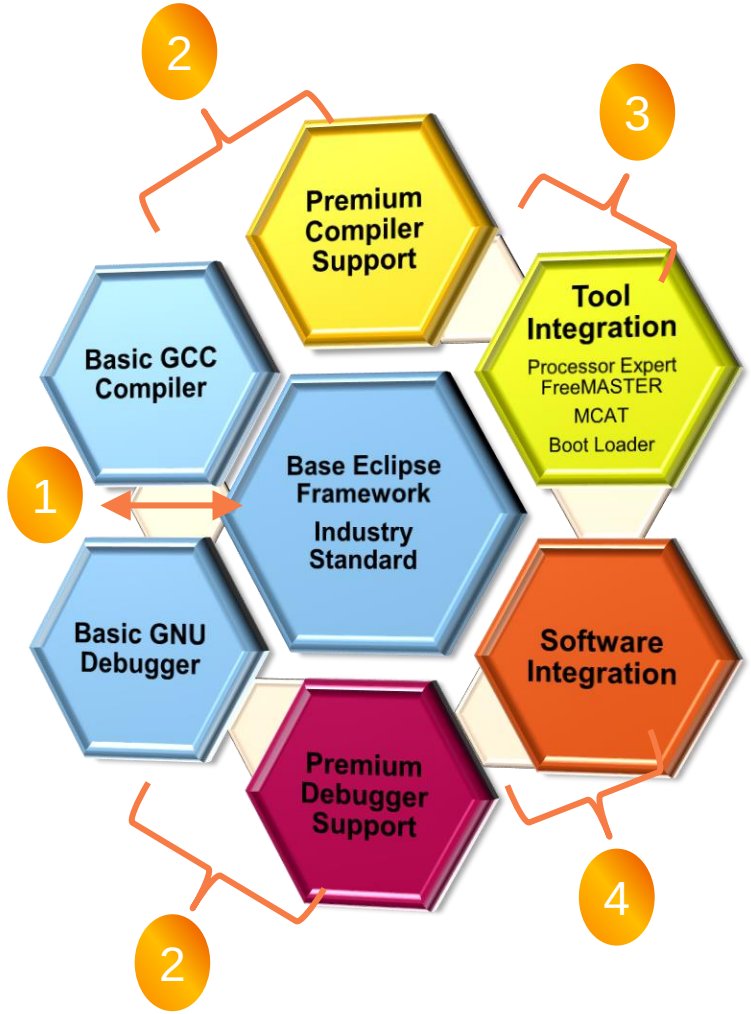
Best in Class Compilers for Code Density and Code Performance

Examples: **GHS** and **IAR** both have ISO 26262 certifications and Certification Kits

Third-Party Premium Debugger Support
When required for Trace of code execution and advanced debugging **Lauterbach** and **iSystems** are among Industry leaders.

These debuggers plug-in to the S32DS seamlessly integrated for use when the most difficult software problems need to be solved.

S32 Design Studio: 4 Step Software Overview



1
- **Create a new S32DS Project**
- Select MCU and target package

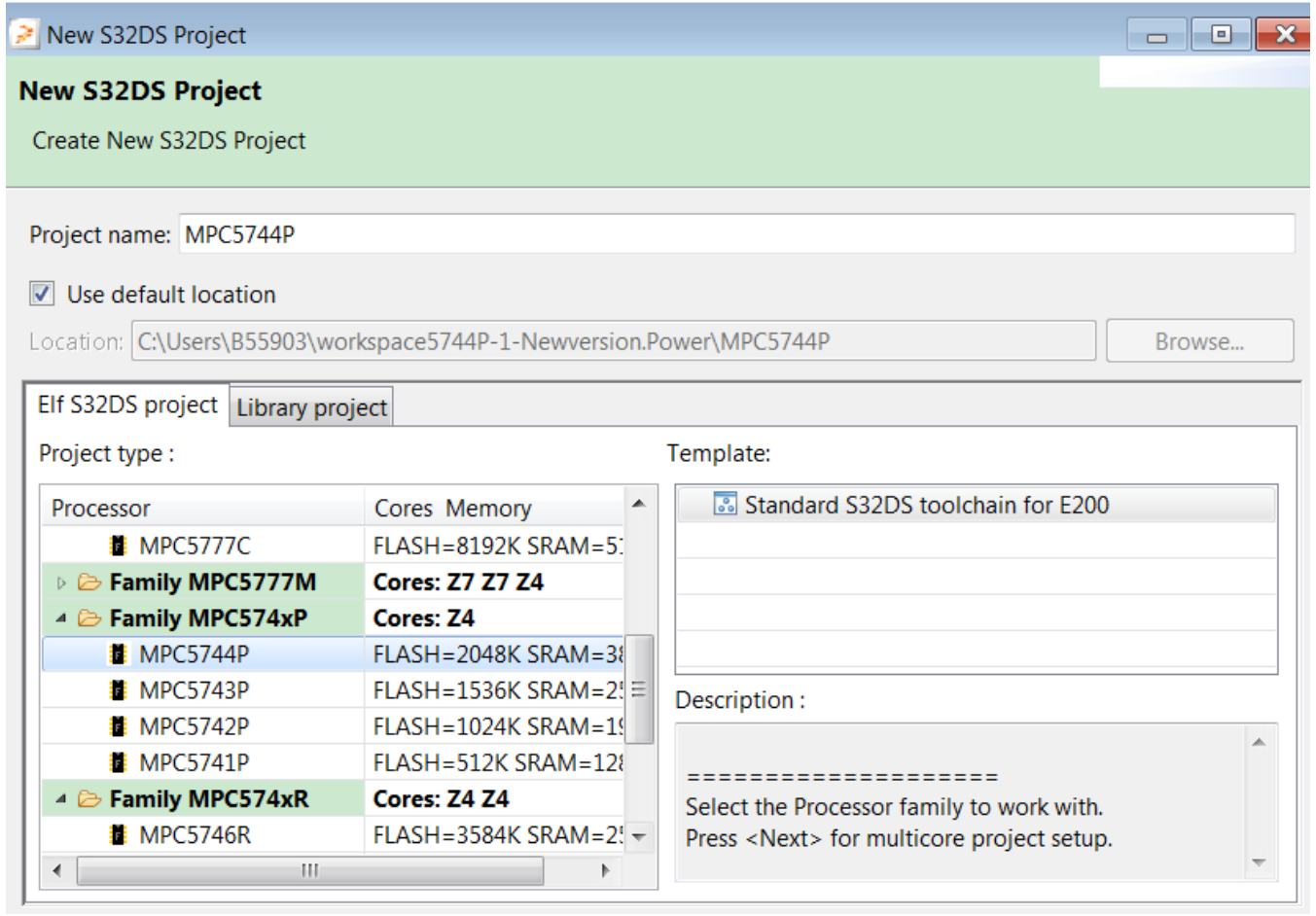
2
- Select **Compiler**
  - GCC or 3rd party Premium Compiler
- Select **Debugger**
  - Basic GNU or 3rd party Premium Debugger

3
- Select **Integration tools**
  - FreeMASTER GUI
  - Motor Control Tool (MCAT)

4
- Select **Software Integration**
  - Math & Motor Control Library (MMCL)
  - Core Self-Test functions

Get started with S32 Design Studio

- Create New Project
- Debugger Configuration
- Basic Debug Session



PERIPHERAL

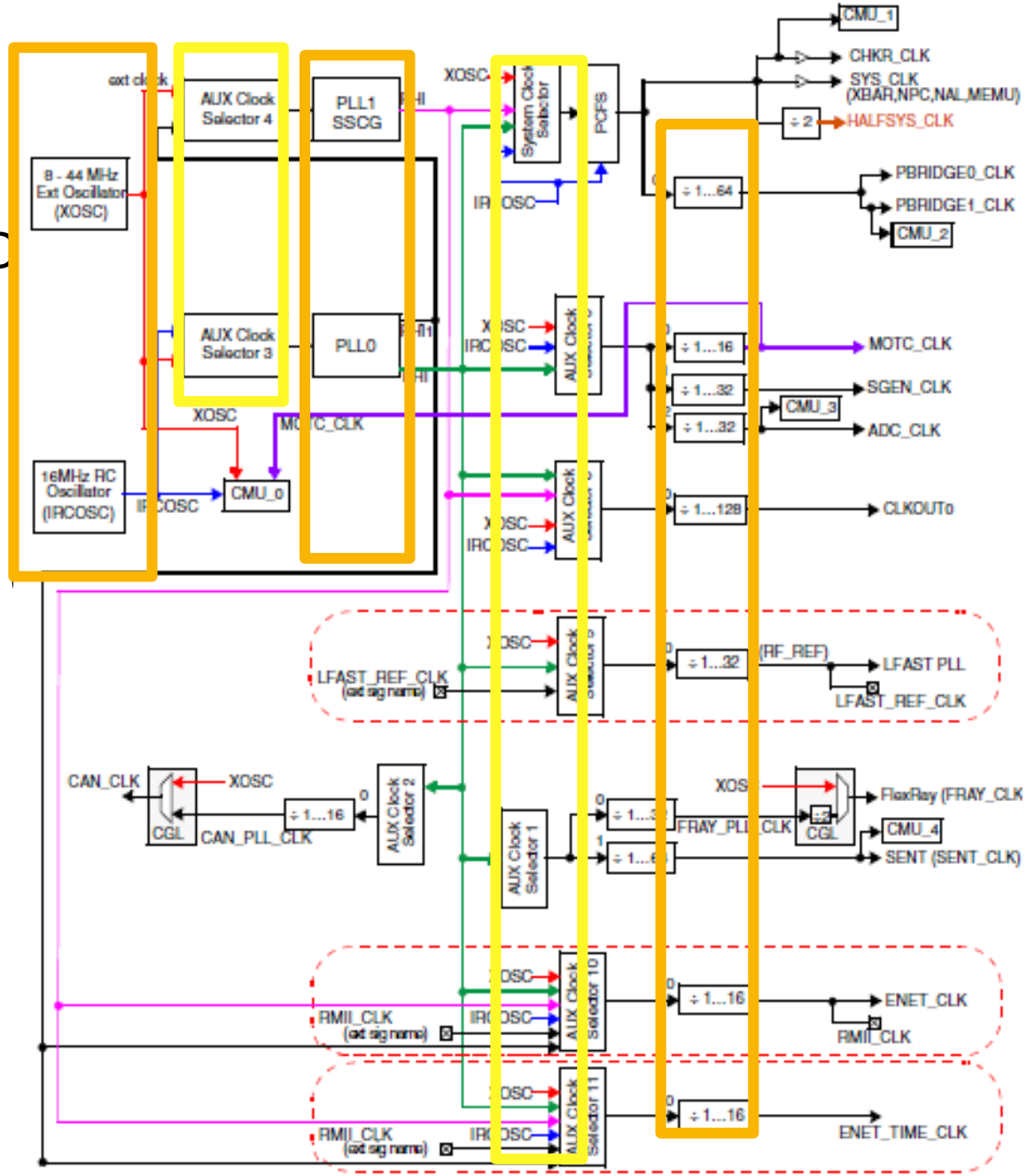
CLOCK

(CGM,PLL, PMU)

CGM Clock sources

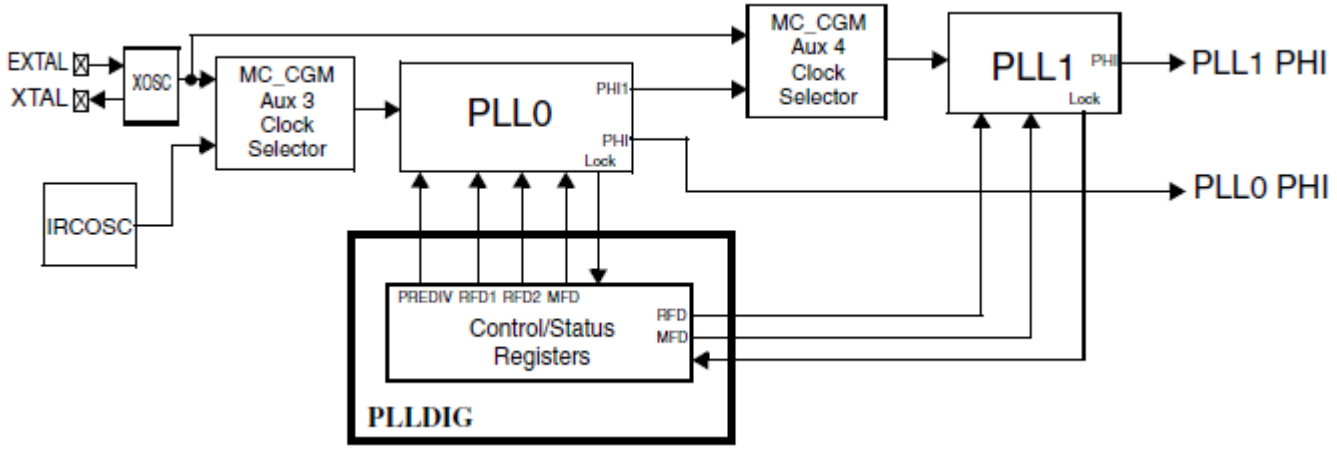
- Main Clock
  - 8-40 MHz External Crystal/Oscillator -> FXOSC
  - 16 MHz Internal RC Oscillator -> IRC

- 2 PLL
  - Optional output clock frequency modulation for
  - Optionallya monitored by CMU



PLL

- A Dual PLL that provides separate system and peripheral clocks. The PLLs are disabled after power on and must be enabled by software.

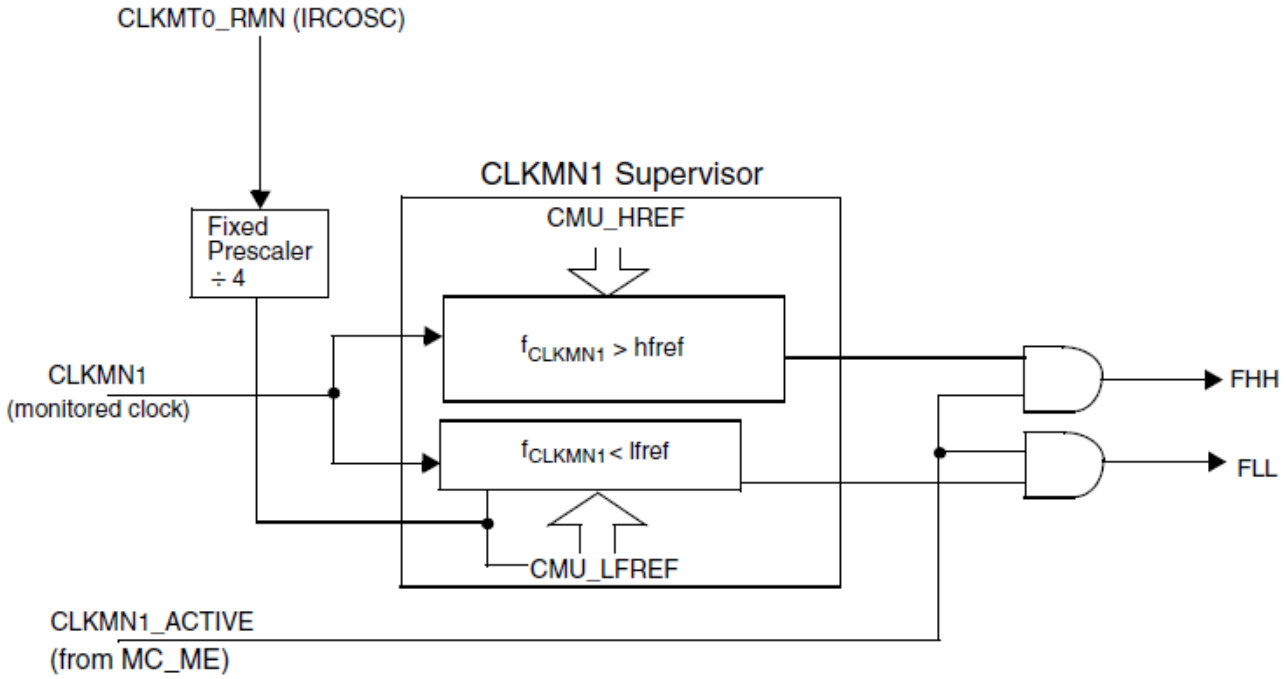


Example:

- fPLL0_PHI = fPLL0_ref x PLL0DV[MFD] / (PLL0DV[PREDIV] x PLL0DV[RFDPHI]
  = 40MHz x 8 / (1 x 2) = 160 MHz
- fPLL0_PHI1 = fPLL0_ref x PLL0DV[MFD] / (PLL0DV[PREDIV] x PLL0DV[RFDPHI1])
  = 40MHz x 8 / (1 x 8) = 40 MHz
- fPLL1_PHI = fPLL1_REF * ((PLL1DV[MFD] + PLL1FD[FRCDIV]/2^12) / (2 x PLL1DV[RFDPHI]))
  = 40MHz x (20 + 0) / (2 x 2) = 200 MHz

CMU (Clock Monitor Unit)

- IRCOSC is the reference clock for all clock monitors.
- CMU0 uses the IRCOSC clock to measure if the XOSC is too low.
- CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC



CMU[1:4] Block Diagram

Clock module	Monitored clock
CMU0	MOTC_CLK (CLKMN1 for CMU0), XOSC, IRCOSC
CMU1	CHKR_CLK
CMU2	PBRIDGE0_CLK, PBRIDGE1_CLK
CMU3	ADC_CLK
CMU4	SENT_CLK

PMC
(Pow Management Control)

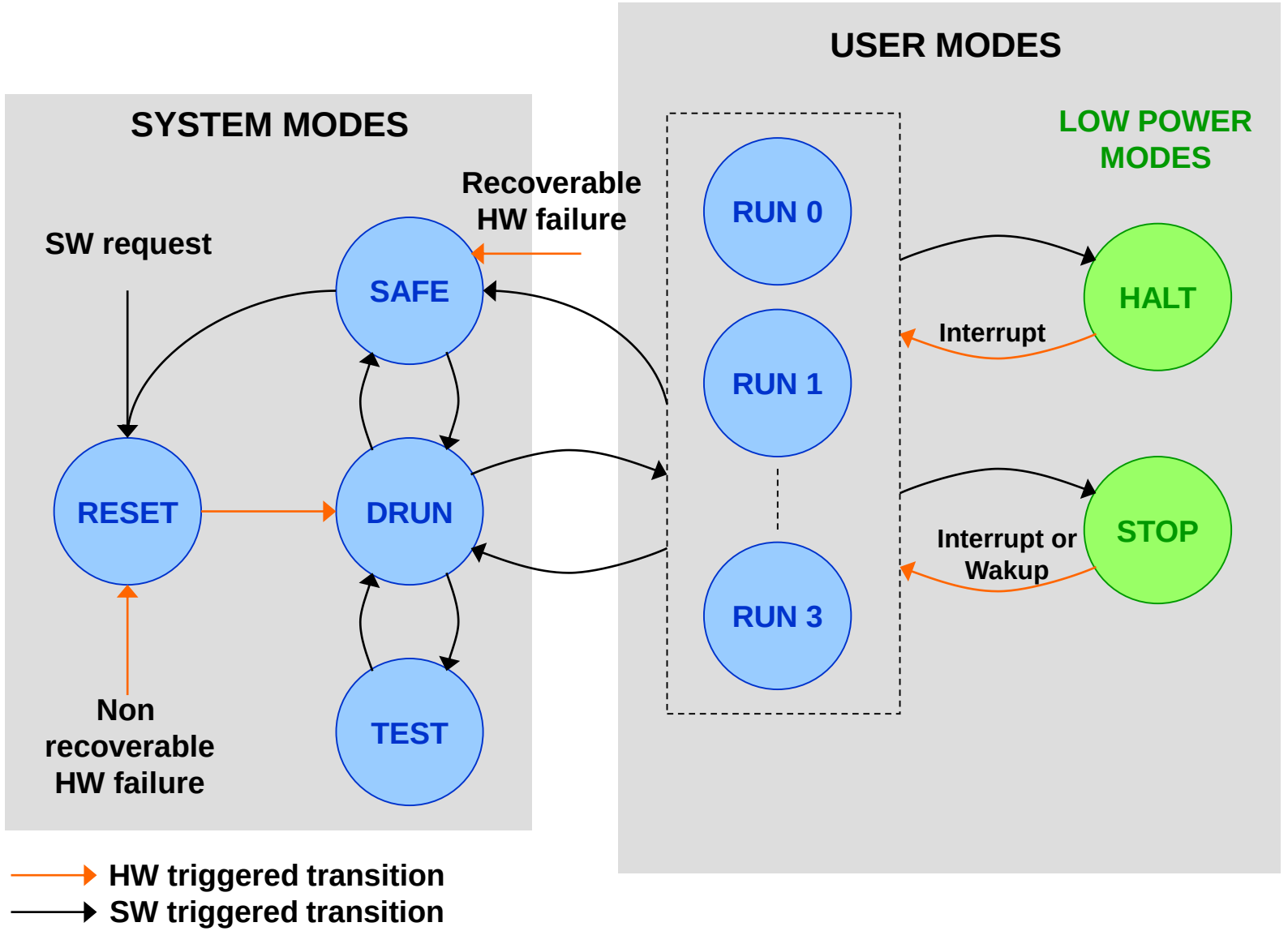
MC_ME(Mode Entry Module)

Purpose:

- The purpose of the Mode Entry (ME) is to centralize the control of all device modes and related modules / parameters within a unique module

- The ME simplify the implementation of mode management and so increase its robustness
  - Avoiding to manage the power modes on a module by module basis (e.g. peripherals, FLASH mode, voltage regulator)
  - Defining the available modes and their related configuration
  - Providing a SAFE mode to manage HW failure

MC_ME(Mode Entry Module)



Mode Overview

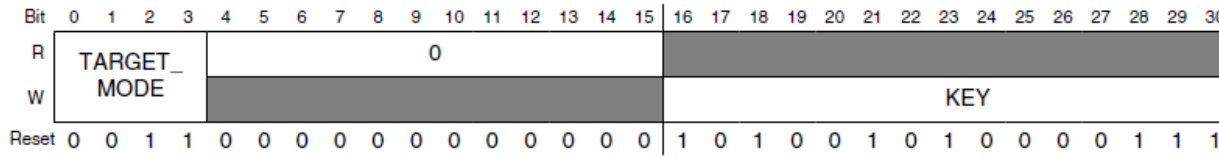
- Provide SYSTEM modes and USER modes
  - SYSTEM: RESET, DRUN (Default RUN), SAFE and TEST
  - USER: RUN(0..3), HALT, STOP
  - For each mode the following parameters are configured/controlled
  - System clock sources (ON/OFF)
  - System clock source selection
  - Memory (flash and RAM) power mode (ON, low power, power down)
  - Pad output driver state
  - Peripherals' clock (gated/clocked)
  - Power domains
- Control without CPU intervention the target mode's parameters and mode transition

RESET Mode Configuration Register (MC_ME_RESET_MC)
TEST Mode Configuration Register (MC_ME_TEST_MC)
SAFE Mode Configuration Register (MC_ME_SAFE_MC)
DRUN Mode Configuration Register (MC_ME_DRUN_MC)
RUN0 Mode Configuration Register (MC_ME_RUN0_MC)
RUN1 Mode Configuration Register (MC_ME_RUN1_MC)
RUN2 Mode Configuration Register (MC_ME_RUN2_MC)
RUN3 Mode Configuration Register (MC_ME_RUN3_MC)
HALT0 Mode Configuration Register (MC_ME_HALT0_MC)
STOP0 Mode Configuration Register (MC_ME_STOP0_MC)

MC_ME(Mode Entry Module)

Example:

```
/* Enable XOSC in DRUN mode and select as SYS_CLK */
MC_ME.DRUN_MC.R = 0x00130031;
```




```
 /* RE enter the DRUN mode, to update the configuration */
MC_ME.MCTL.R = 0x30005AF0;          /* Mode & Key */
MC_ME.MCTL.R = 0x3000A50F;          /* Mode & Key inverted */

while(MC_ME.GS.B.S_MTRANS == 1);    /* Wait for mode entry to complete */
while(MC_ME.GS.B.S_CURRENT_MODE != 0x3);  /* Check DRUN mode has
  been entered */
while(!MC_ME.GS.B.S_XOSC);          /* Wait for clock to stabilise */
```

0000	RESET (triggers a 'functional' reset event)
0001	TEST
0010	SAFE
0011	DRUN
0100	RUN0
0101	RUN1
0110	RUN2
0111	RUN3
1000	HALT0
1001	reserved
1010	STOP0
1011	reserved

Mode transition is controlled by writing twice ME_MCTL register

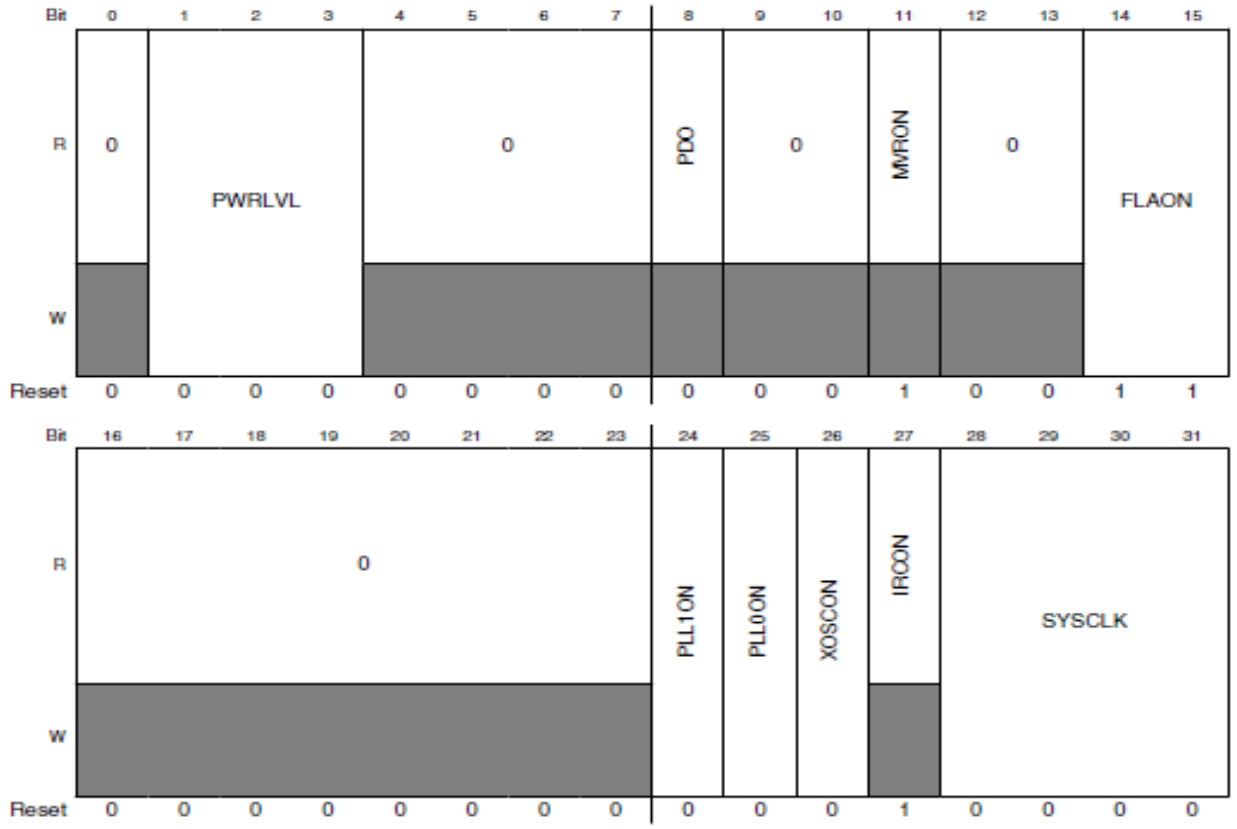
1st write: TARGET_MODE + KEY

2nd write: TARGET_MODE + INVERTED KEY

Mode Configuration Register

- Each mode has a Mode Configuration register ME_XXX(RUN0)_MC, where XXX is the mode, e.g. RUN0, HALT, STOP etc.

• PDO: output power-down control
•MVRON: control VREG on/off
• Flash module
  - Normal
  -Low Power
  - Power Down
•PLLON: control PLL on/off
• XOSCON: control XOSC on/off
• IRCON: control IRC16M on/off
• SYSCLK: select system clock
  -16 MHz IRCOSC
  -4-40 MHz XOSC
  -Primary PLL (PHI)
  -Second PLL



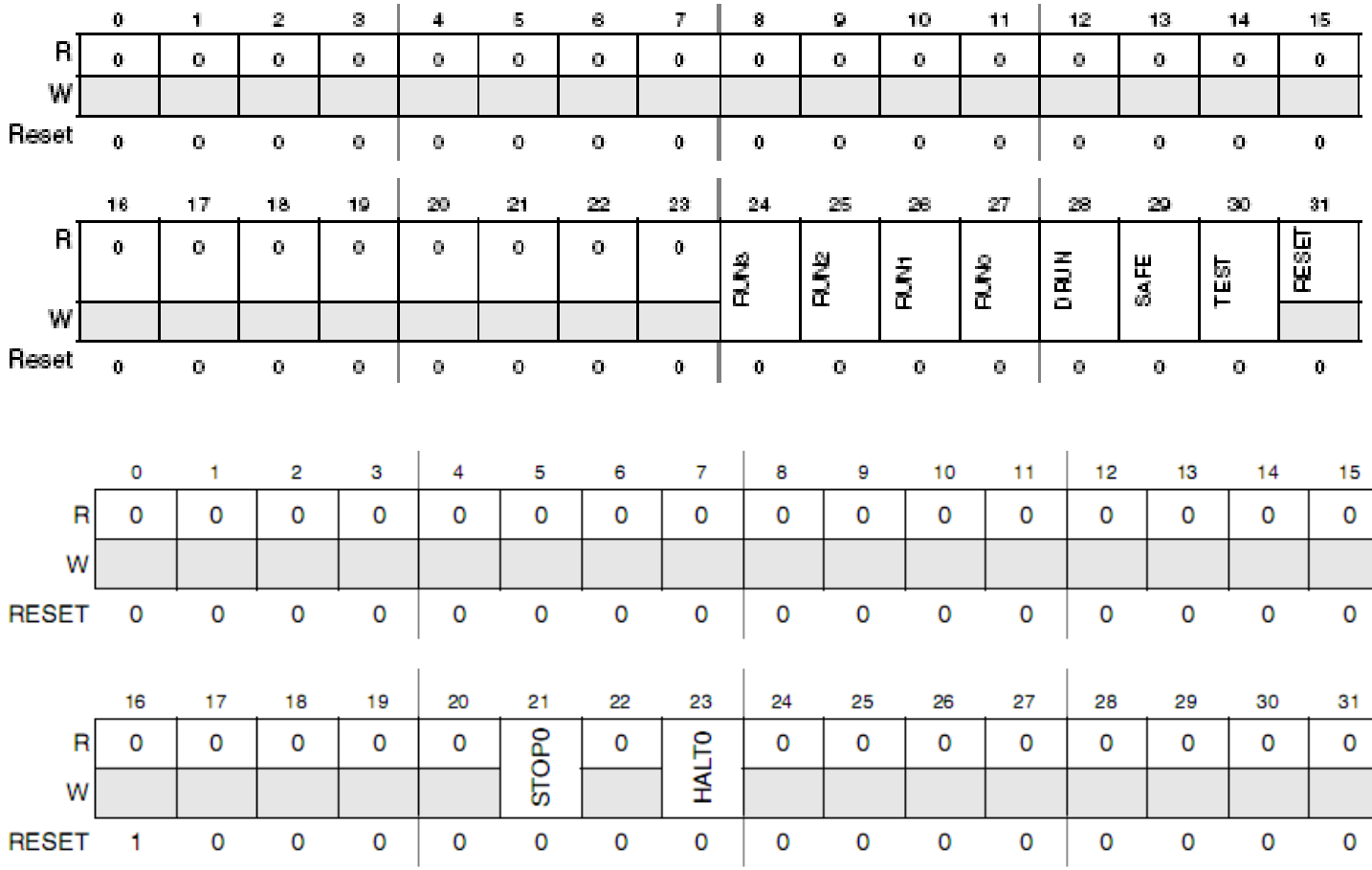
MC_ME Resource Control

MC_ME Resource Control Overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
IRC	on	√ on	on	on	on	on	on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	off	√ off	off	√ off	√ off	√ off	off
PLL1	off	√ off	off	√ off	√ off	√ off	√ off
FLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down
MVREG	on	on	on	on	on	on	on
PDO	off	√ off	√ on	off	off	off	√ off

RUN/LP Configuration Registers

- ME_RUN_PC[0…7] registers configure the behavior of peripherals during RUN Modes
- ME_LP_PC[0…7] registers configure the behavior of peripherals during Low Power Modes
- Configuration bits
  - 0: peripheral is frozen with clock gated
  - 1: peripheral is active



Peripheral Control Registers

- ME_PCTL[9…255] registers select the Running/Low Power Mode configurations for each peripheral

Peripheral Control Registers (ME_PCTL0…143)

- DBG_F bit controls the state of the peripheral in Debug Mode
  - 0: peripheral status depends on LP_CFG / RUN_CFG
  - 1: peripheral is frozen
- LP_CFG bits specify a Low Power Mode configuration as defined in ME_LP_PCs
  - 000: select the ME_LP_PC[0] configuration
  - …
  - 111: select the ME_LP_PC[7] configuration
- RUN_CFG bits specify a Running Mode configuration as defined in ME_RUN_PCs
  - 000: select the ME_RUN_PC[0] configuration
  - …
  - 111: select the ME_RUN_PC[7] configuration

Timer
(STM,PIT,SWT)

System Timer Module (STM)

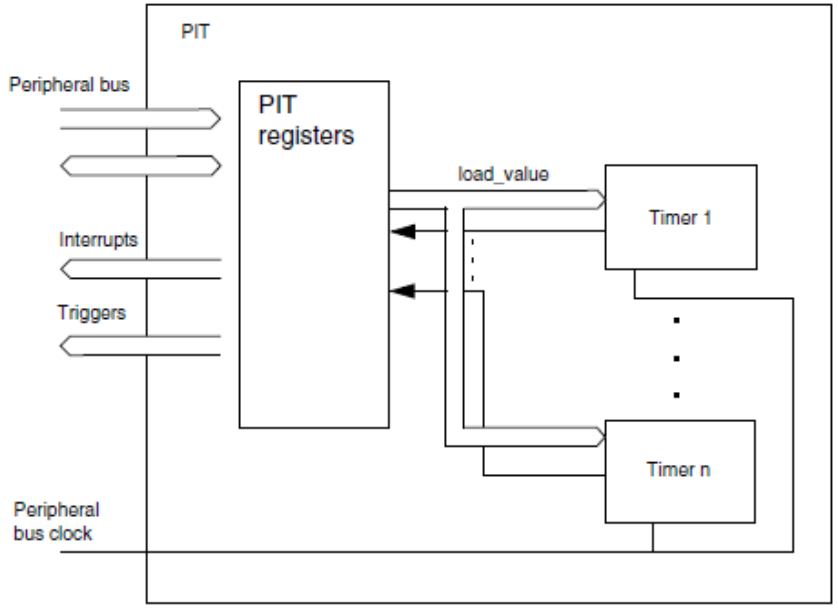
- The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value

- **Features:**
  - One 32-bit up counter with 8-bit prescaler
  - Four 32-bit compare channels
  - Independent interrupt source for each channel
  - Counter can be stopped in debug mode

PIT

Features:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer



SWT

- The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out

- **Features:**
  -32-bit time-out register to set the time-out period
  -Programmable selection of window mode or regular servicing
  -Programmable selection of reset or interrupt on an initial time-out
  -Programmable selection of the servicing mode

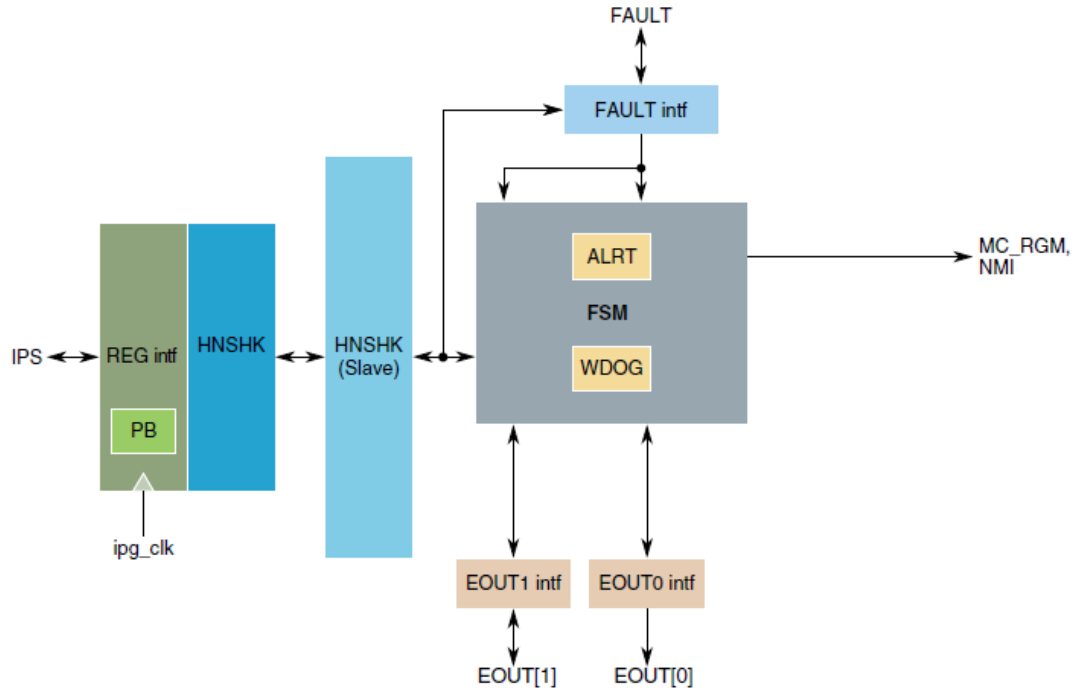
FCCU

FCCU (Fault Collection and Control Unit)

The 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. No CPU intervention is requested for collection and control operation.

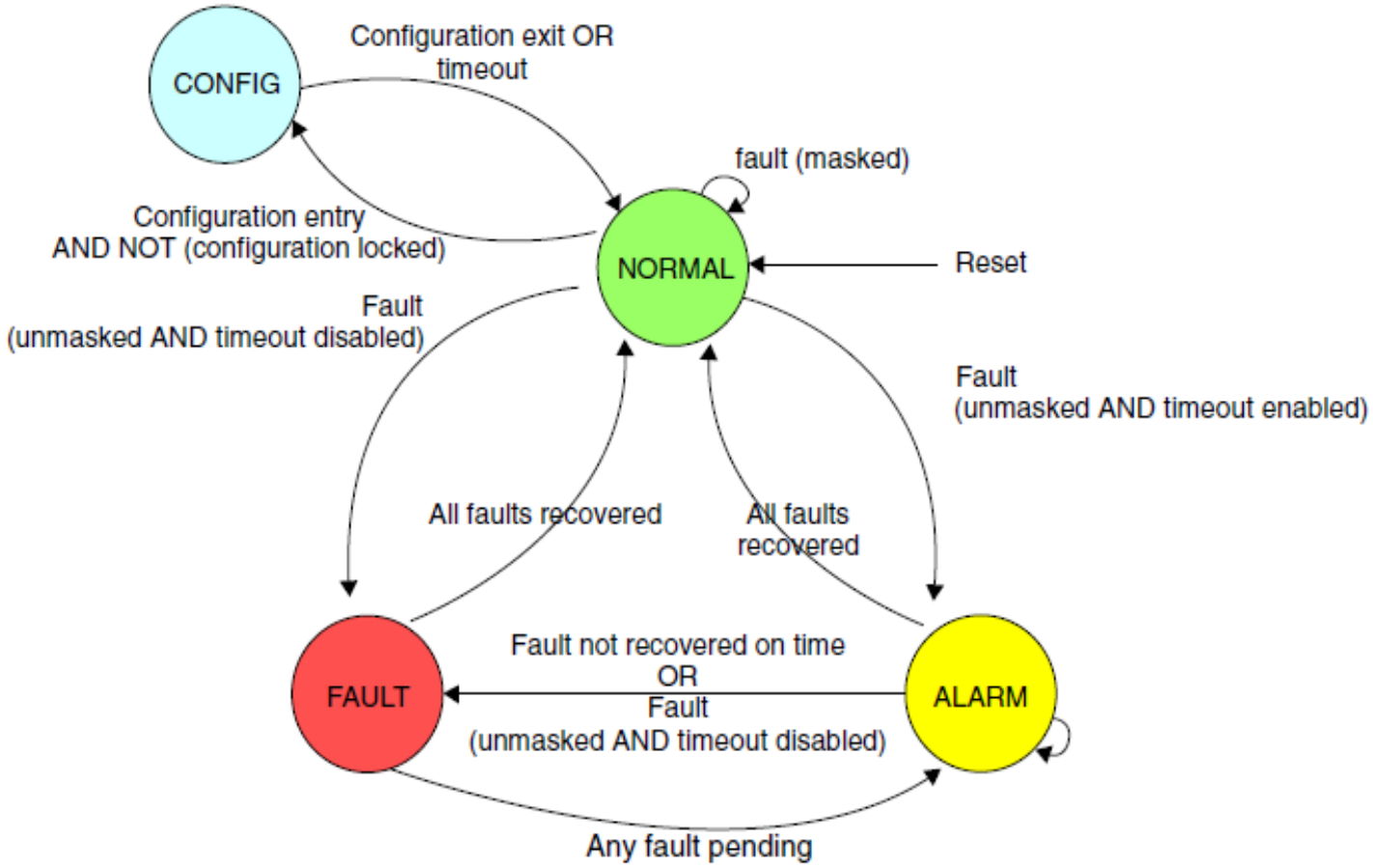
Main Features:

- Management of non-critical faults
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is
- controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
- Long/short functional reset request pulse
- NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase



FCCU block diagram

FCCU State Diagram



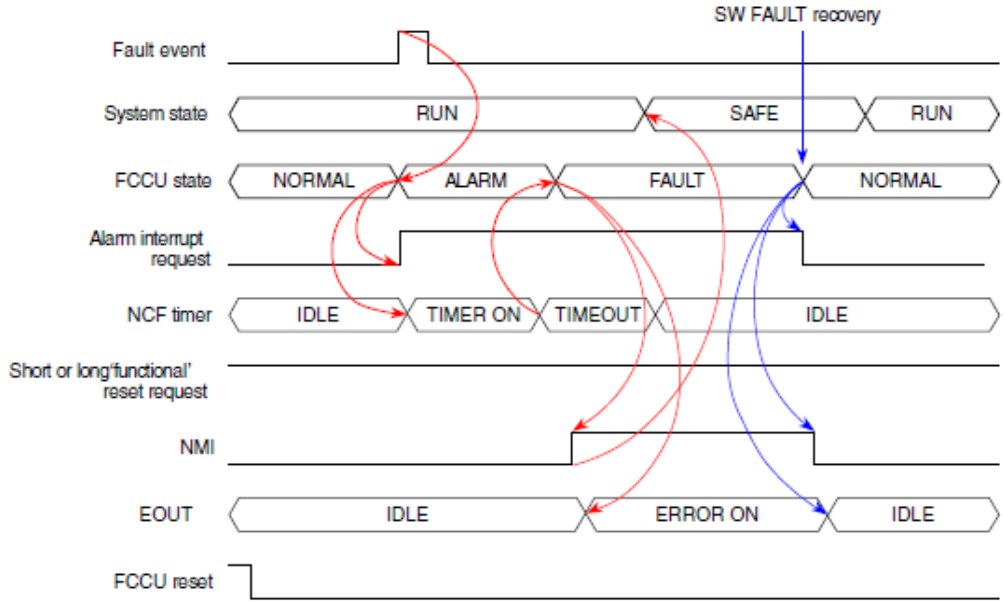
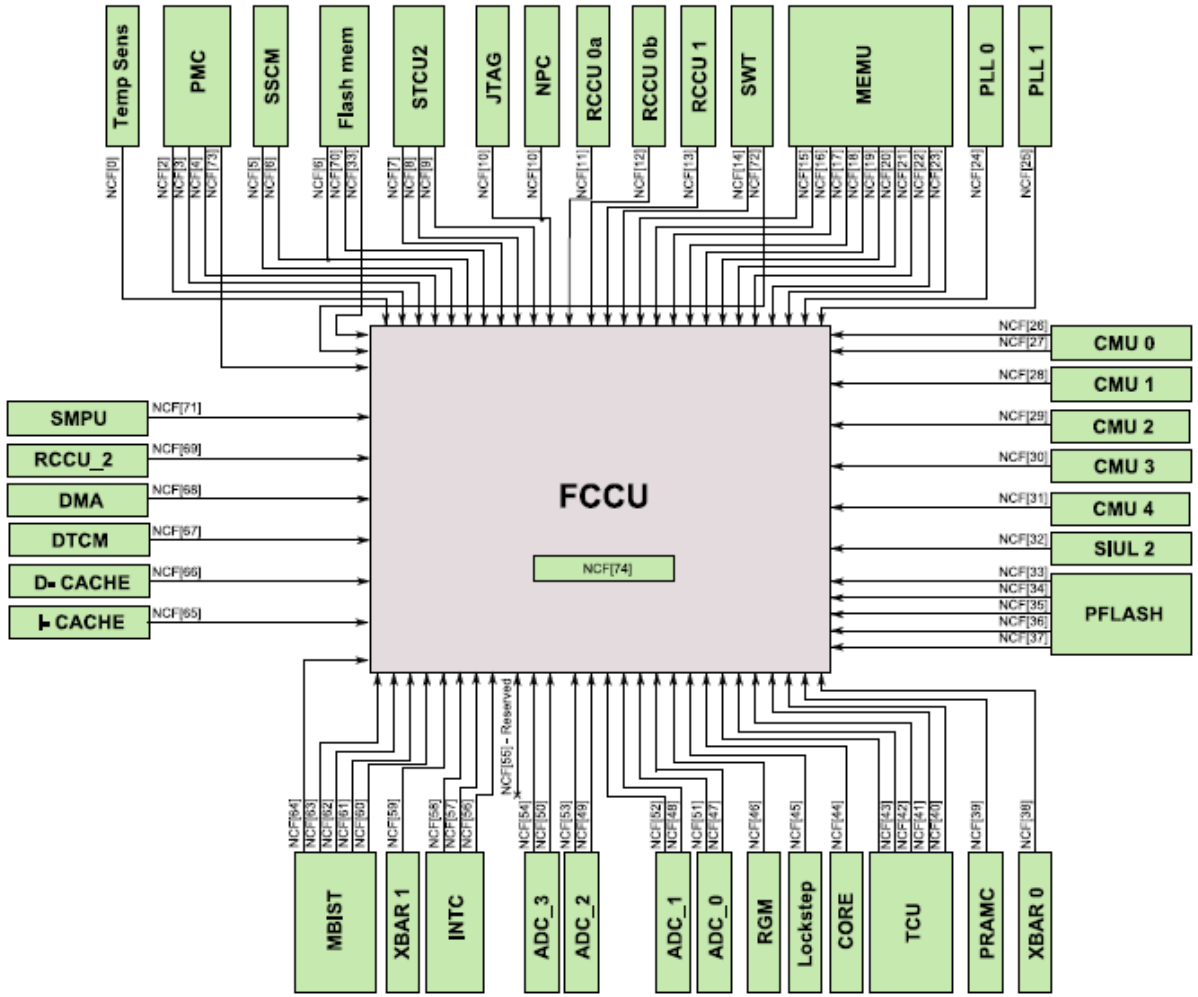


Figure 69-5. FAULT (ALARM → FAULT state) recovery

FCCU fault sources (**total 75 faults**)

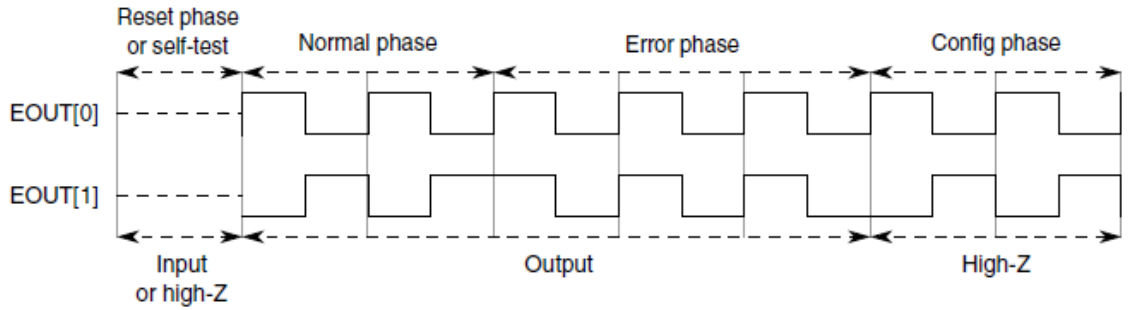
Module	Fault typ
Core	Core redundancy mismatch
CMU	Loss of crystal; Frequency out of range
PLL_0	Loss of lock
PLL_1	Loss of lock
XOSC/ IRCOSC	clock frequency out of range
SWT	SW Watchdog Timeout
JTAG	JTAG/NPC monitor
PMC	LVD,HVD
Flash	ECC error
SRAM	ECC error
…	…



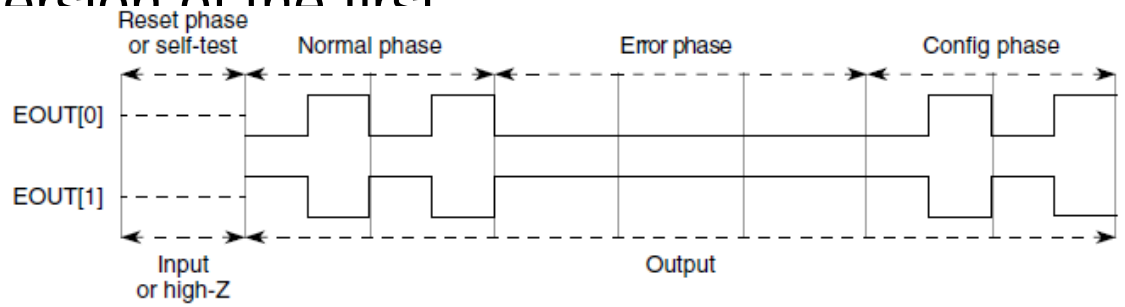
Documents: RM(FCCU chapter) and AN5259

Output Indication

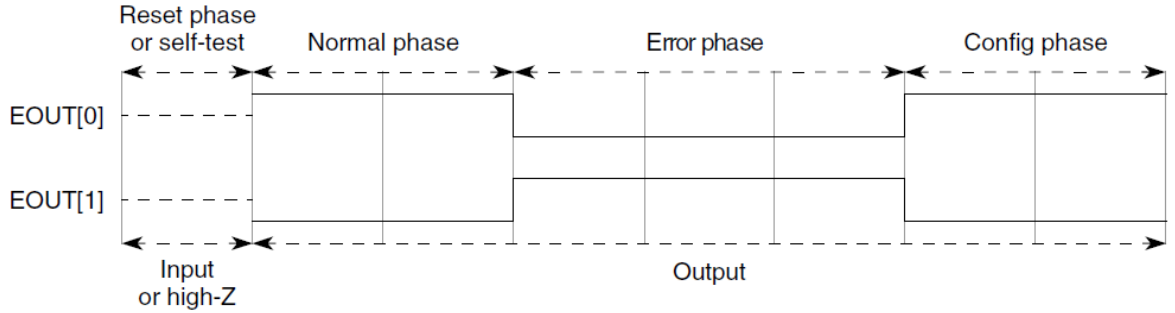
- Two output pins EOUT[0] and EOUT[1] can b
  indicate the fault condition
- There is a choice of fault indication protocols
  - Both external signals are used only in dual-rail p
    protocols, the second output is the inverted version of the first
    output
- Dual-Rail Protocol
  - EOUT[0] and EOUT[1] toggle such that they
    except when an error occurred
- Time switching protocol
  - EOUT[0] toggles unless there is a fault when
- Bi-Stable protocol
  - EOUT[0] is high until there is a fault then low



Dual-rail protocol (slow switching mode)

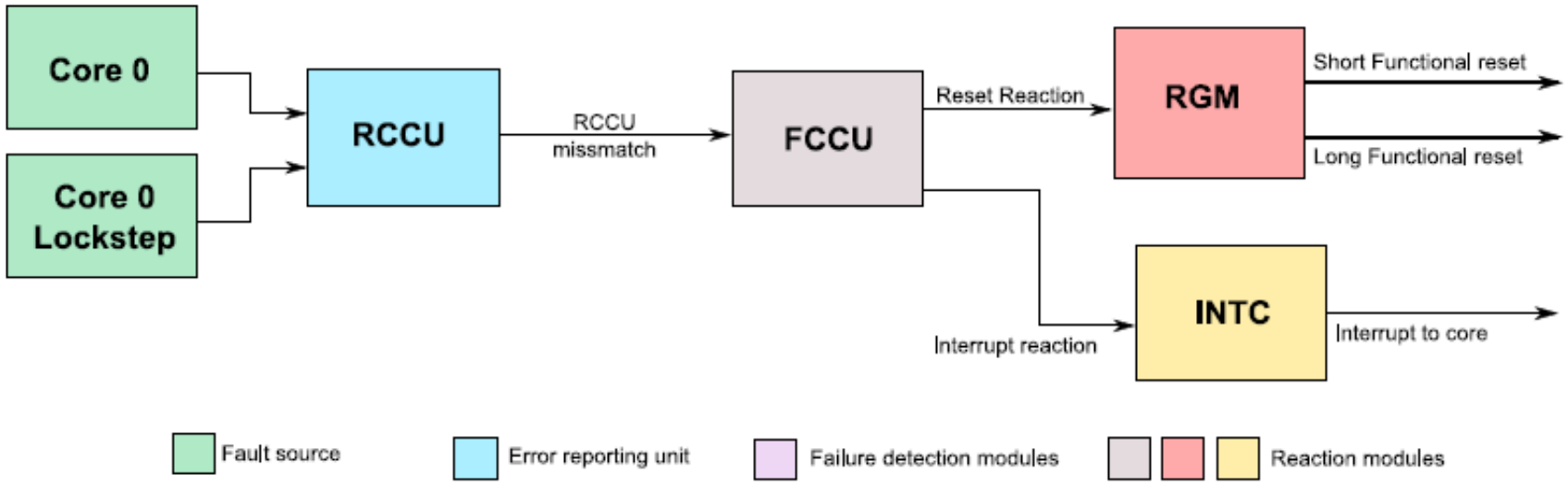


Time-switching protocol



Bi-stable protocol

Example: Core Redundancy Mismatch



Core redundancy mismatch: interface

Example: Software Watchdog Timer

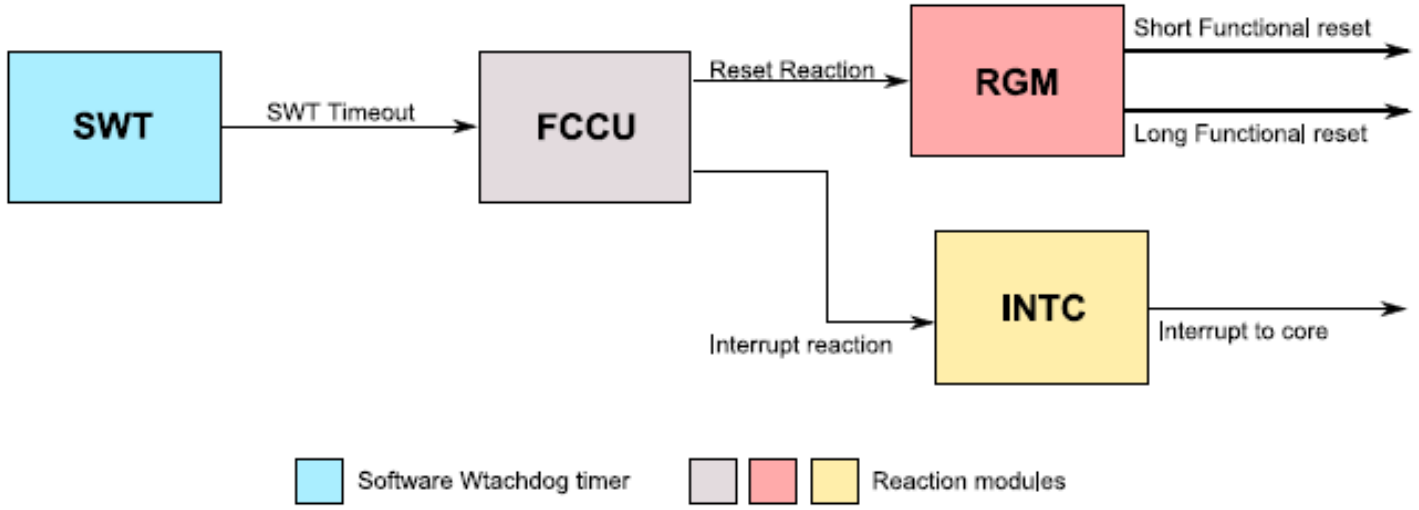


Figure 16. SWT timeout

Note: The SWT is not connected directly to the RGM, and the FCCU is by default configured not to react on faults. When the SWT expires sooner than the FCCU is configured to react on a fault, no reaction is taken on the SWT timeout event.

NCF Mapping

Table 7-36. FCCU Non-Critical Faults Mapping

Non-Critical Fault	Source	Signal Description	Default Functional Reset (Short, Long, None)	Set/Clear injection[1]	Fault enabled	Time-out enabled
NCF[0]	Temp Sens 0/1	Temperature out of range 0/1	—	X (by PMC itself)	X	X
NCF[1]	Reserved		—	—	—	—
NCF[2]	PMC	LVDs ORed	—	X (by PMC itself)	X	X
NCF[3]	PMC	HVDs ORed	—	X (by PMC itself)	X	X
NCF[4]	PMC	Safety error (BIST)	—	X (by PMC itself)	X	X
NCF[5]	DCF/SSCM	Memories DCF client safety error	—	—[2]	X	X
NCF[6]	SSCM / flash memory	Safety error: SSCM transfer error (during STCU2 configuration loading) ORed with flash memory reset error	—	—	X	X
NCF[7]	STCU2	STCU2 fault condition (run in application mode)	—	X	X	X
NCF[8]	STCU2	BIST results (critical faults)	Reset	X	X	X
NCF[9]	STCU2	BIST results (non-critical faults)	—	X	X	X
NCF[10]	JTAGC_NPC_MON	JTAG/NPC monitor	—	X	X	X
NCF[11]	RCCU_0a	Core redundancy mismatch: interface (other than D-MEM or DMA) out of lockstep[3]	—	X	X	X
NCF[12]	RCCU_0b	Core redundancy mismatch: D-MEM array interface out of lockstep[3]	—	X	X	X
NCF[13]	RCCU_1	Core redundancy mismatch: DMA array interface out of lockstep[3]	—	X	X	X
NCF[14]	SWT_0	Software watchdog timer	—	—	X	X
NCF[15]	MEMU	System RAMs correctable ECC error	—	—	X	X

Config state

```
void FCCU_CONFIG (void)
{
  /* Unlock configuration */
  FCCU.TRANS_LOCK.R = 0xBC;

  /* provide Config state key */
  FCCU.CTRLK.R = 0x913756AF;          //key for OP1
  /* enter config state - OP1 */
  FCCU.CTRL.R = 0x1;                    //set OP1 - set up FCCU into the CONFIG mode

  /* wait for successful state transition */
  while (FCCU.CTRL.B.OPS != 0x3);          //operation status successful

  /* Insert here the FCCU configuration */

  /* set up the NOMAL mode of FCCU */
  FCCU.CTRLK.R = 0x825A132B;           //key for OP2
  FCCU.CTRL.R = 0x2;                   //set the OP2 - set up FCCU into the NORMAL mode
  while (FCCU.CTRL.B.OPS != 0x3);      //operational status successful
}
```

SIUL
(System Integrity Unit Lite2)

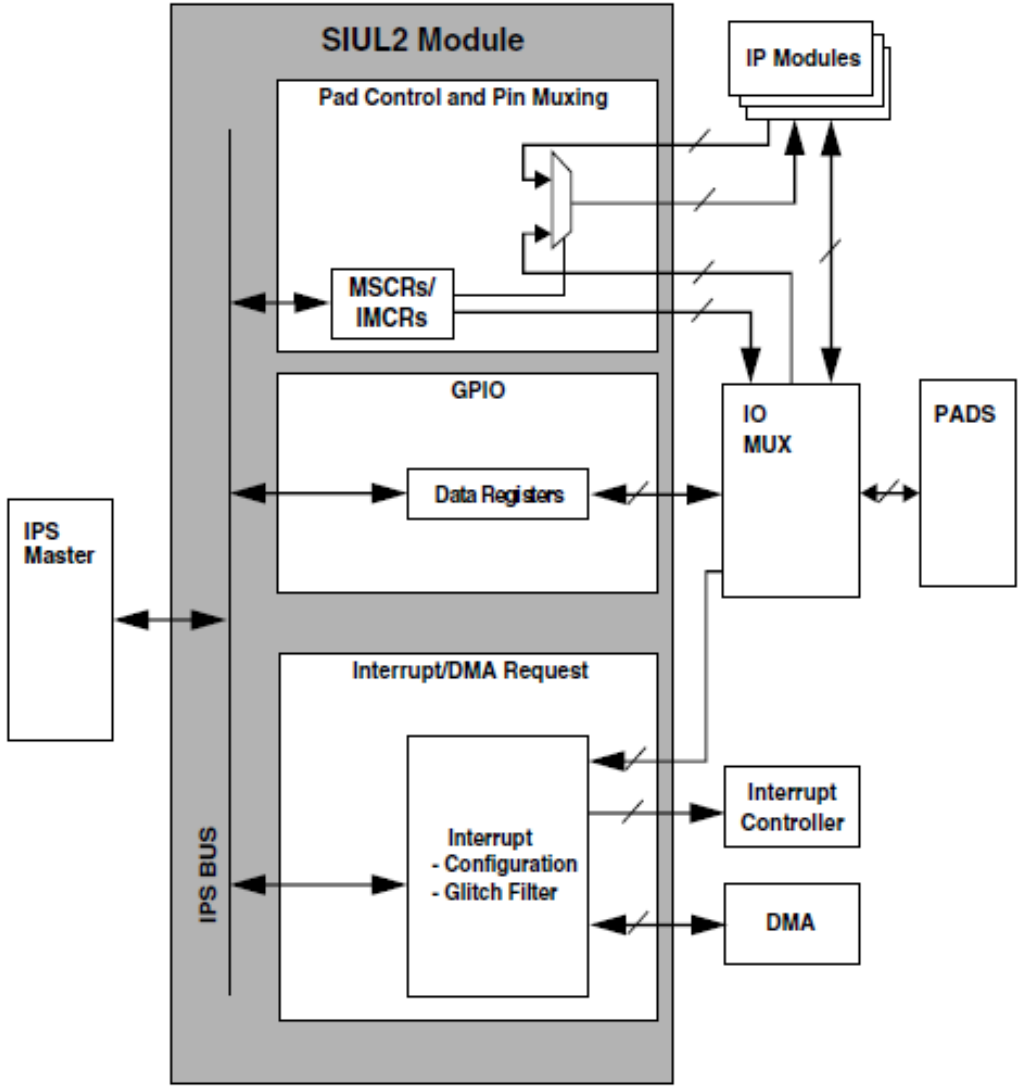
Features

- Two 16-bit registers can be read/written with one access for a 32-bit port, if needed.
- The SIUL2 supports 32 external interrupts on this device. The 32 external interrupt sources are grouped into 4 sets with 8 sources each. Each of the 4 groups is assigned one interrupt vector and mapped to the INTC
- External interrupt/DMA request support
- 1 to 32 programmable digital glitch filters, one for each REQ pin
- 1 to 4 system DMA request channels for 1 to 4 REQ pins
- Edge detection
-

SIUL Pad Control and IOMux configuration

- Pad Control is managed through **MSCR** registers

- IOMux configuration is managed through:
  - MSCR Registers (output functionalities)
  - IMCR Registers (input functionalities)



System Integration Unit Lite2 block diagram

MSCR and IMCR

MSCR

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		SRC[1:0]		0		OBE	ODE	SMC	APC	0		IBE	HYS	PUS	PUE
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

- **SRC (Slew Rate Control):** Controls the slew rate of the output signals
- **OBE (Output Buffer Enable):** Enables the output buffer of the pad when in GPIO mode.
- **IBE (Input Buffer Enable):** Enables the input buffer of the pad.
- **ODE (Open Drain Output Enable):** Selects either open drain. This feature applies to output pads only.
- **SMC (Safe Mode Control):** Override automatic deactivation of the output buffer upon entering SAFE mode of the SoC.
- **APC (Analog Pad Control):** Enables the usage of the pad as analog input.
- **PUEE (Pull Up/Down Enable):** Enables/Disables the pull up/down on the pad.
- **PUS (Pull Up/Down Select):** Selects weak pull up/down if enabled.
- **SSS**: Selects which source signal is connected to the associated destination

IMCR

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

SIUL Pad Control and IOMux configuration

- Example:

➢ Output pad:

If A[0] used as clock out of DSPI2, MSCR[0].SSS=0010;

➢ Input pad:

If A[0] used as input capture of eTimer_0
IMCR[59].SSS= 0010
MSCR[0].IBE=1;

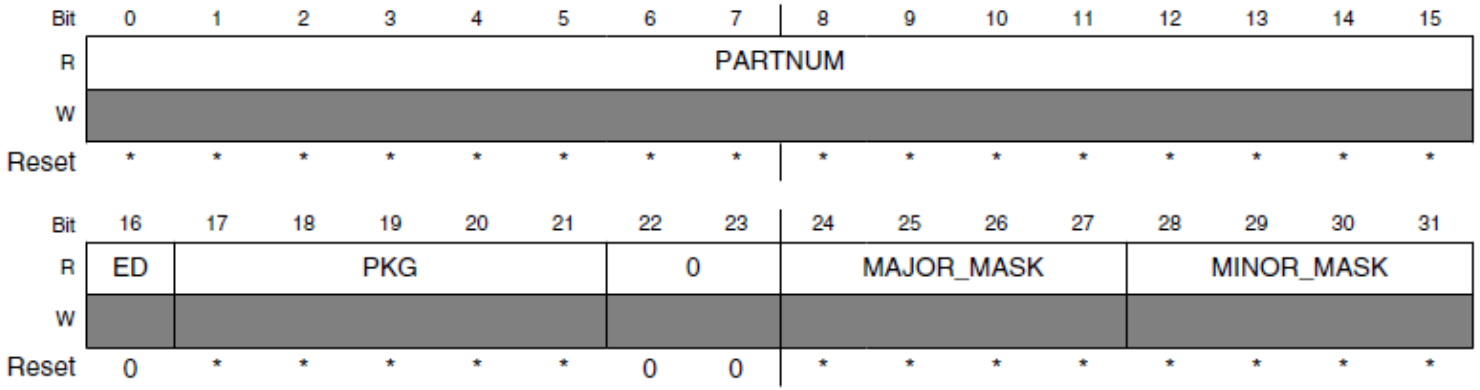
Refer to Table 4-7. Pin muxing

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
A[0]	MSCR[0]	0000 (Default)[2]	GPIO[0]	SIUL2-GPIO[0]	General Purpose IO A[0]	I/O	73	P12
		0001	ETC0	eTimer_0	eTimer_0 Input/Output Data Channel 0	I/O		
		0010	SCK	DSPI2	DSPI 2 Serial Clock (output)	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[48]	0001	SCK	DSPI2	DSPI 2 Serial Clock (input)	I/O		
	IMCR[59]	0010	ETC0	eTimer_0	eTimer_0 Input Data Channel 0	I/O		
	IMCR[173]	0001	REQ0	SIUL2	SIUL2 External Interrupt 0	I		
A[1]	MSCR[1]	0000 (Default)	GPIO[1]	SIUL2-GPIO[1]	General Purpose IO A[1]	I/O	74	T14
		0001	ETC1	eTimer_0	eTimer_0 Input/Output Data Channel 1	I/O		
		0010	SOUT	DSPI2	DSPI 2 Serial Data Out	O		
		0011-1111	—	Reserved	—	—		
	IMCR[60]	0010	ETC1	eTimer_0	eTimer_0 Input Data Channel 1	I/O		
	IMCR[174]	0001	REQ1	SIUL2	SIUL2 External Interrupt	I		

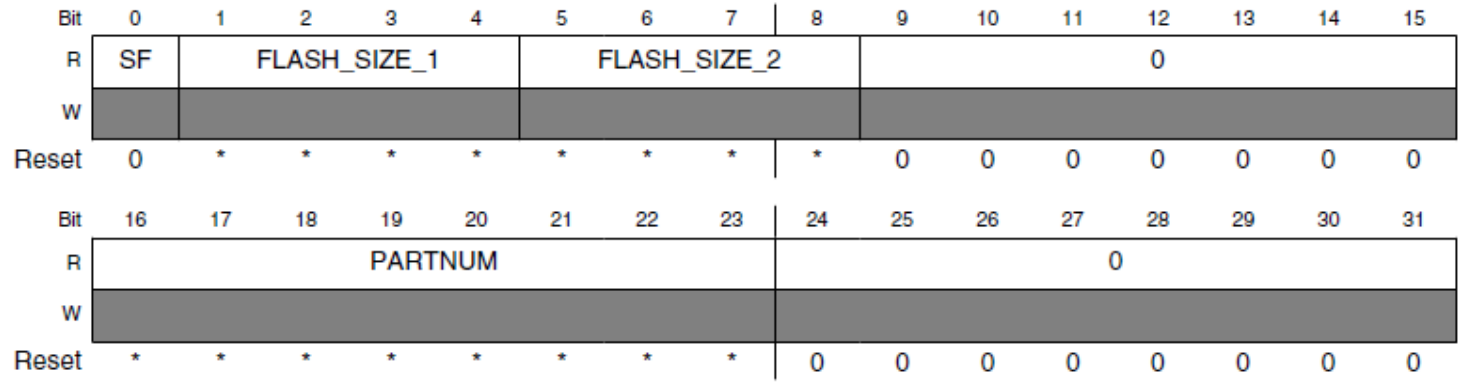
SIUL MCU Identification Registers (MDIR)

•SIUL includes two registers that can be read by users and tools manufactures to determine what device is present (which part number, pkg, flash size etc.) and to take the corresponding actions. These registers are called:

– MIDR1



– MIDR2



Demo

MEMORY

Memory Map On-Chip Memory

Start address	End address	Allocated size	Used size	PCTL number	Description
Flash memory—see Table 5-4 for details					
0x00000000	0x003FFFFF	-			Reserved
0x00400000	0x00403FFF	16 KB		-	UTest NVM block - no overlay See Table 5-6 for details
0x00404000	0x007FFFFF	-			Reserved
0x00800000	0x00817FFF	96 KB		-	Data flash memory blocks - no overlay
0x00818000	0x009FFFFF	-			Reserved
0x00A00000	0x00FFFFFF	6 MB	256 KB	-	Small and medium flash memory blocks - no overlay
0x01000000	0x01FFFFFF	16 MB	3584 KB	-	Large flash memory blocks - no overlay
0x00C68000	0x089FFFFF	8 MB			Reserved
0x08A00000	0x08FFFFFF	6 MB	256 KB	-	Mirror small and medium flash memory blocks
0x09000000	0x09FFFFFF	16 MB	3.5 MB	-	Mirror large flash memory blocks
0x00000000	0x3FFFFFFF	-			Reserved
System RAM—see Table 5-5 for details					
0x40000000	0x4005FFFF	384 KB		-	System RAM
0x40060000	0x4007FFFF	-			Reserved System RAM
0x40080000	0x4FFFFFFF	-		-	Reserved

RAM

- MPC5744P includes 384 KB general-purpose on-chip ECC SRAM. The SRAM can be configured for either 0- or 1-wait state read operation latency using the PRCR1 register in the SRAM controller.

- A portion of the system SRAM can be used as the overlay SRAM. *The overlay SRAM* feature included in MPC5744P is part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables calibration of constant data without requiring additional external RAMs and calibration memory interfaces.

Flash

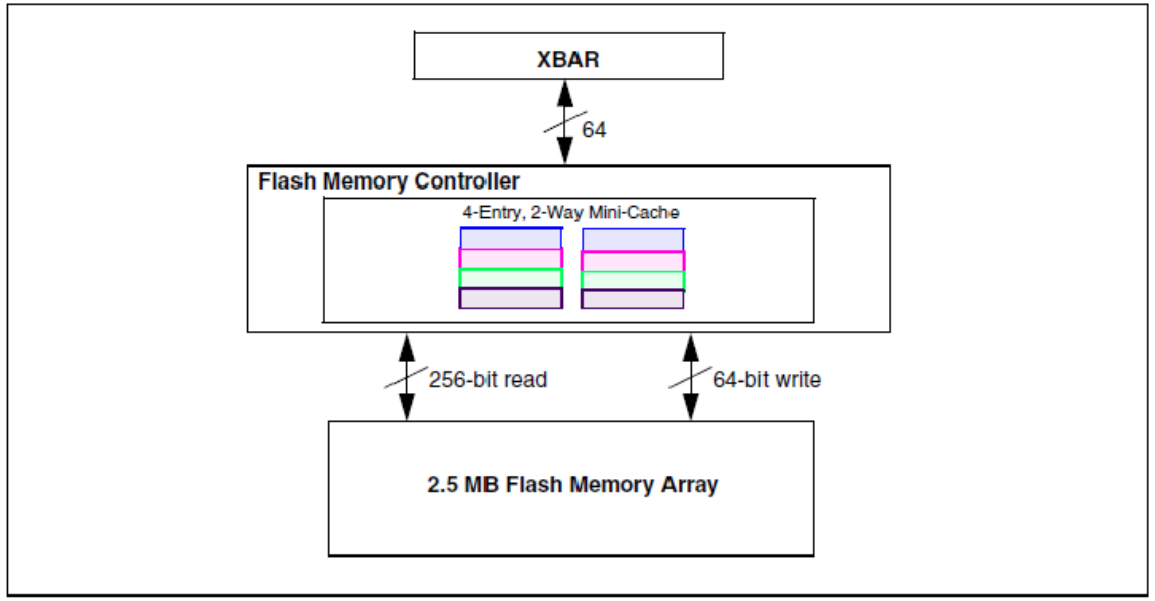
Feature:

- 2.5MB of Flash in unique multi partitioned hard macro
  Flash partitioning
  4x 16 KB in partition 0/1 (2x blocks EEPROM emulation enabled)
  2x 32 KB in partition 2/3 (EEPROM emulation enabled)
  6x 64 KB in partition 4/5
  8x 256 KB in partition 6/7

- Support for reading-while-writing when the accesses are to different partitions.

- Flash protection
  -Write protection and OTP available for dedicated blocks.
  -Test information stored in a non-volatile UTest block which will be OTP.
  -Erase suspend, program suspend and erase-suspended program all supported.

- Support e2eECC



Device flash memory block diagram

Interrupt

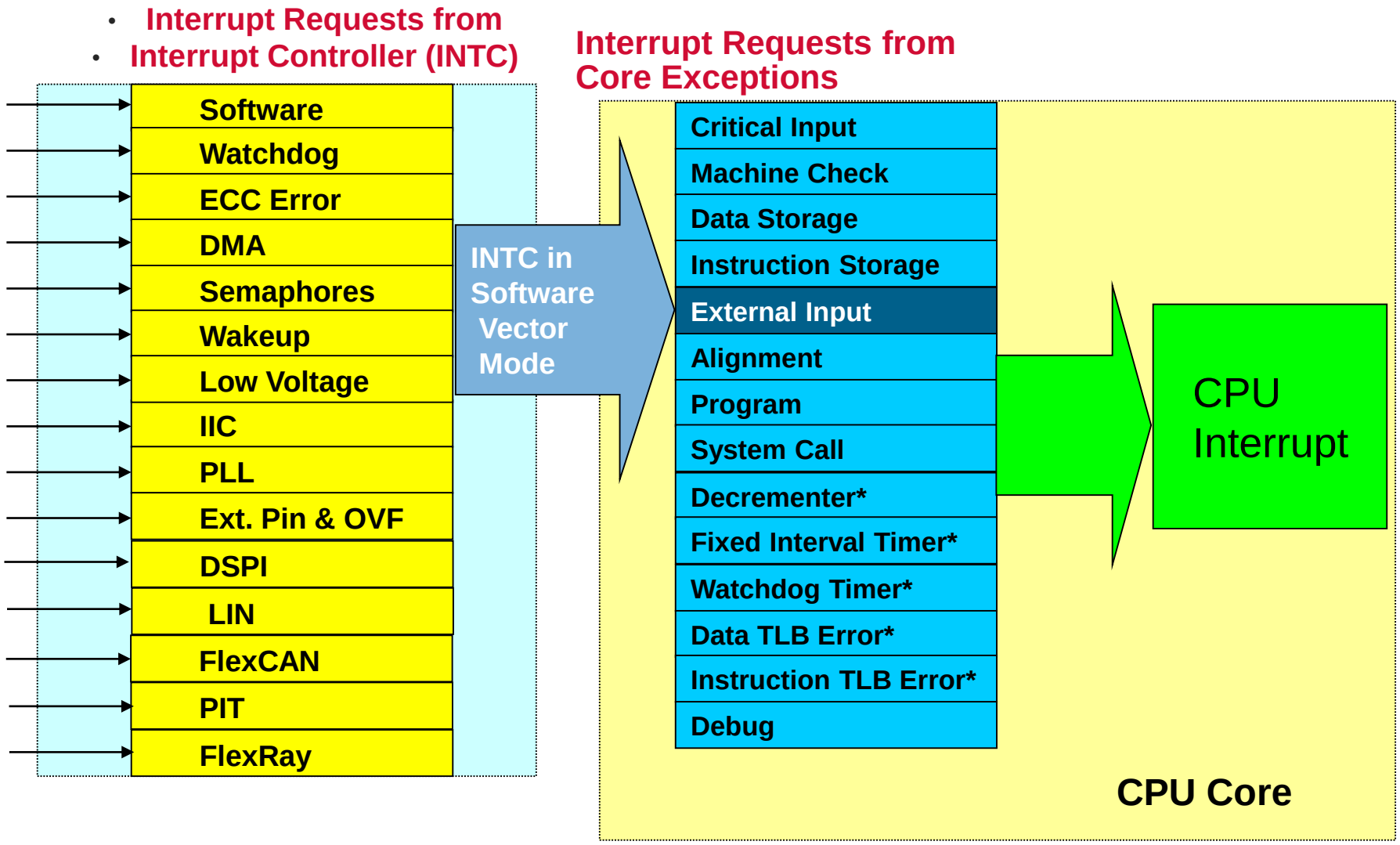
Interrupts: e200z4 Interrupt Vectors

IVOR #	Interrupt Type	Enables[1]	State Saved In	Examples
IVOR0	Critical Input	CE	CSRR0:1	Non-maskable interrupt
IVOR1	Machine Check	ME	CSRR0:1	ISI, ITLB error on 1st instr'n of exception handler
IVOR2	Data Storage	-	SRR0:1	Incorrect privilege mode for R/W access
IVOR3	Instruction Storage	-	SRR0:1	Incorrect privilege mode for instruction
IVOR4	External Input	EE, src	SRR0:1	Peripherals, IRQ pins, software
IVOR5	Alignment	-	SRR0:1	Load or store operand not word aligned
IVOR6	Program	-	SRR0:1	Illegal instruction, trap
IVOR7[2]	FP Unavailable	-	SRR0:1	FP instruction attempt with MSR[FP]=0
IVOR8	System Call	-	SRR0:1	System call, "sc", instruction
IVOR10[2]	Decrementer	EE, DIE	SRR0:1	Decrementer timeout
IVOR11[2]	Fixed-Interval Timer	EE, FIE	SRR0:1	Fixed-interval timer timeout
IVOR12[2]	Watchdog Timer	CE, WIE	CSRR0:1	Watchdog timeout when ENW=1, WIS=0
IVOR13[2]	Data TLB Error	-	SRR0:1	Data TLB miss in MMU
IVOR14	Instruct'n TLB Error	-	SRR0:1	Instruction TLB miss in MMU
IVOR15	Debug	DE, IDM	CSSR0:1	ROM Debugger when HID0[DAPUEN]=0
		DE, IDM	DSRR0:1	ROM Debugger when HID0[DAPUEN]=1

1 CE, ME, EE, DE are in MSR. DIE, FIE, WIE are in TCR. "src" is individual enable for each INTC source. Debug interrupt, IVOR15, also requires EDM = 0 (EDM and IDM are in DBCR0).

2 Unused on e200z4

Interrupts: Software Interrupt Vector Mode Structure



Interrupts: Hardware and Software Vector Mode

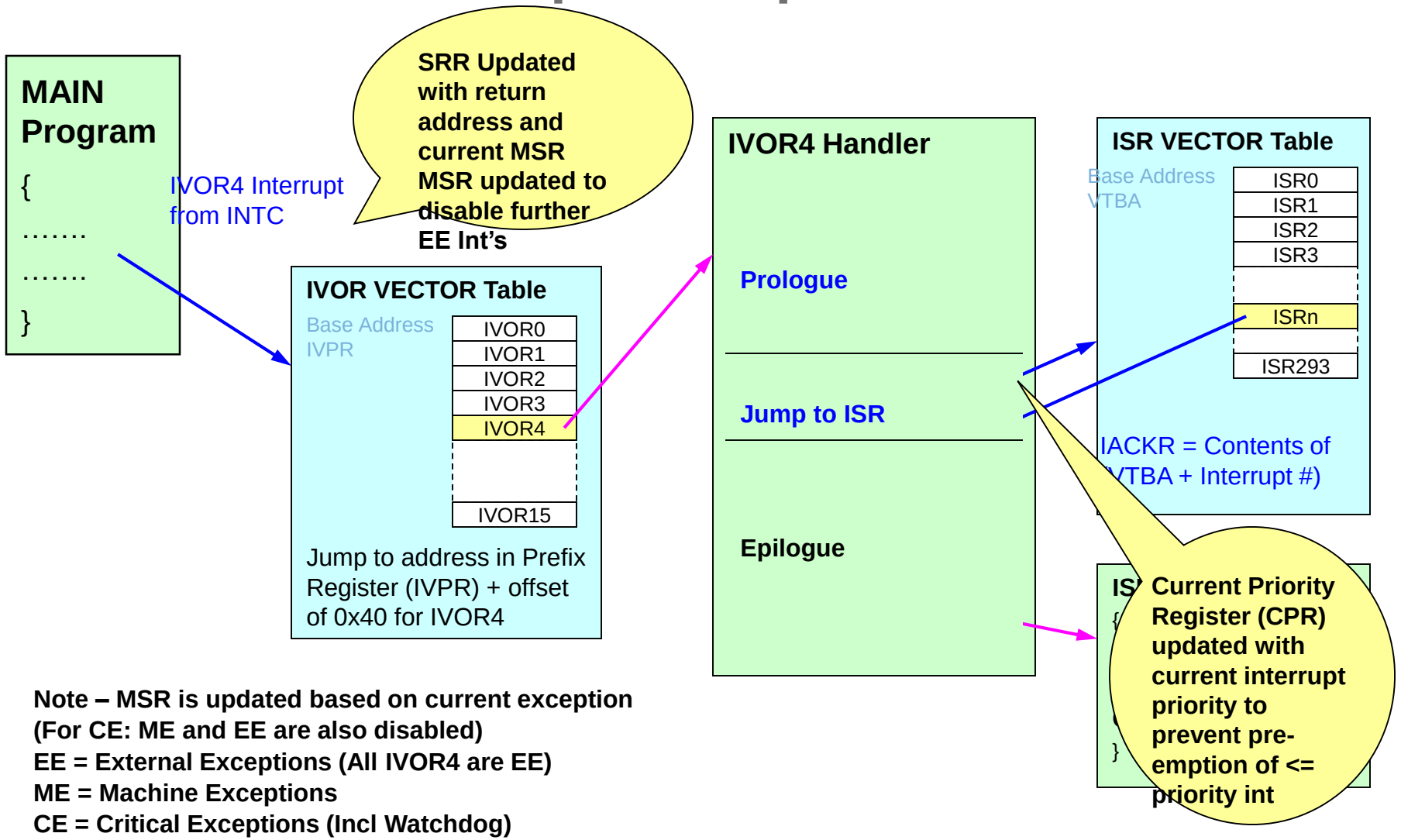
- Software Vector Mode
  - The CPU branches to one of the 16 core interrupt vectors (IVOR's) which contains a branch to an exception handler (eg IVOR4 handler)
  - The exception handler is common for the majority of exceptions
    - Prologue
    - Identification of specific interrupt by reading a register
    - Location of ISR is read from a jump table
    - Branch to ISR
    - Common epilogue

Note – Hardware and software vector mode are only relevant to IVOR4 (INTC) exceptions

- Hardware Vector Mode
  - The IVOR4 vector is not used, instead each interrupt has a unique vector entry containing the jump address of the ISR. The ISR contains:
    - Unique Prologue
    - ISR
    - Unique Epilogue

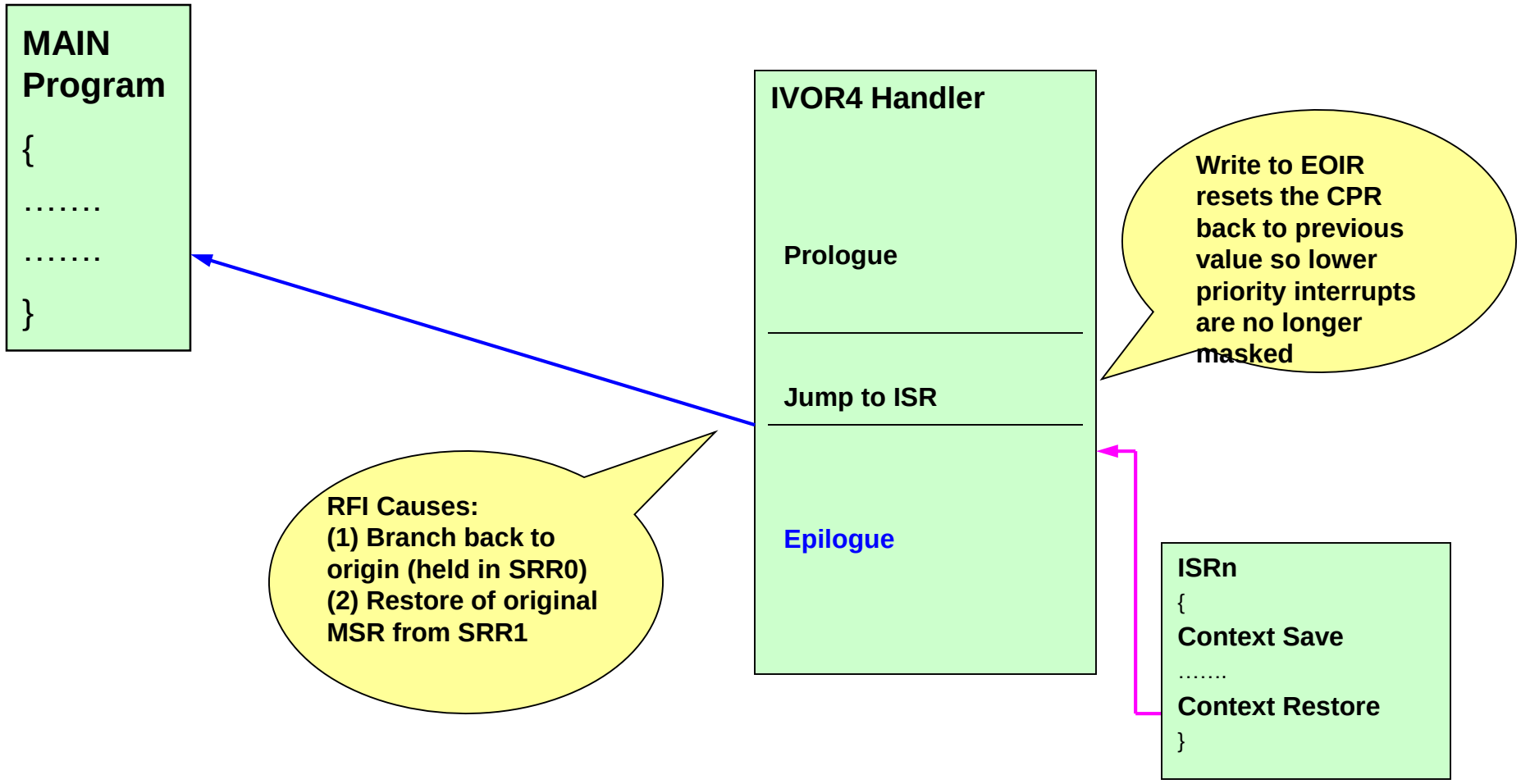
- HW and SW Vector Modes are configurable by core (in **INTC_BCR**).

INTC: Software INTC Interrupt Example



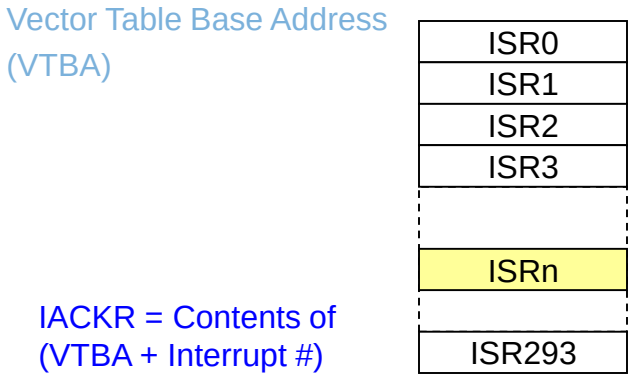
Note – MSR is updated based on current exception (For CE: ME and EE are also disabled)
EE = External Exceptions (All IVOR4 are EE)
ME = Machine Exceptions
CE = Critical Exceptions (Incl Watchdog)

INTC: Software INTC Interrupt Example



INTC: Software Vector Mode Interrupt Acknowledge

- INTC has an Interrupt Acknowledge Register (IACKR) valid for IVOR4 (INTC) exceptions in software vector mode.
  - **INCT_IACKR_PCR0**
  - Reading this register acknowledges the interrupt has taken place and prevents the same interrupt occurring again
  - Reading IACKR also calculates and returns the address of the relevant Interrupt Service Routine based on reading the 32-bit address at “VTBA + ISR Offset”



Interrupts: Software Vector Mode Interrupt Handler

- The IVOR4 Interrupt handler consists of 3 main parts:

- Prologue
  - Save SRR's to stack
  - Read **IACKR** to determine which INTC interrupt occurred
  - Re-enable interrupts in MSR
  - Save GPR's to stack
- Jump to ISR
  - Branch with link to IACKR
- Epilogue
  - Execute mbar to ensure all pending data operations are complete before restoring any registers!
  - Write to EOIR (Sets CPR back to previous value)
  - Restore GPR's
  - Disable Interrupts in MSR
  - Restore SRR's
  - Execute RFI (Return to address in SRR0 and restore MSR)

Address: 0h base + 20h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA																				INTVEC										0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IACKR*n* field descriptions

Interrupts: Hardware Vs Software

- What are the advantages / disadvantages of HW and SW interrupts?

- Software
  - Conforms to Power Architecture (minor point)
  - Common IVOR handler for ISR's saves code.
  - More convoluted than hardware

- Hardware
  - Faster than software to execute
  - Inherently simpler to code / understand than software
  - Less code efficient with prologue and epilogue in each handler!

Example: Configure PIT Interrupt In S32DS

Configure Interrupt:

1.Enable clock for PIT

2.Enable PIT mocule --PIT_0.MCR.B.MDIS = 0;

3.Configure priority for specified interrupt
-- INTC_0.PSR[229].R = 0x4002;

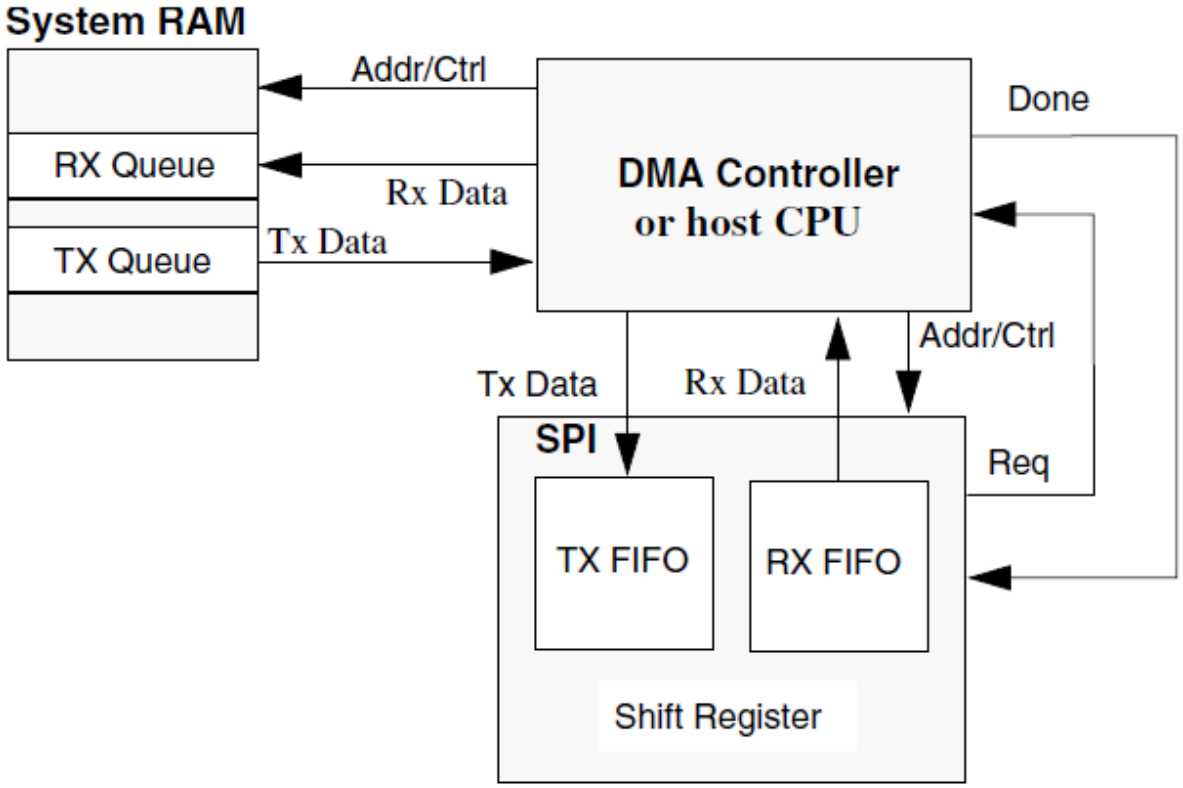
4. Write ISR in the vector_table
-- void IRQ_PIT0_CH3(void)

5. Enable PIT CH3 interrupt
-- PIT_0.CH[3].TCTRL.R = 0x00000003;

SPI

Feature

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 5 entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 5 entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues



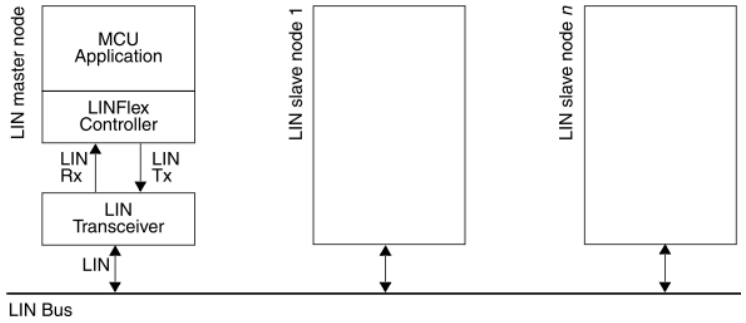
SPI with queues and DMA

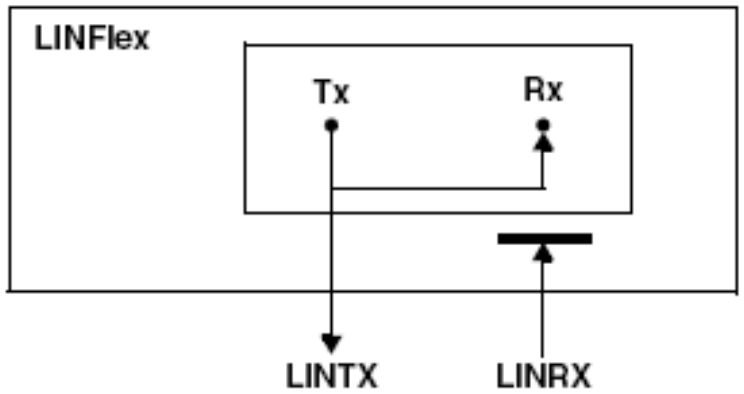
LINFlexD

LINFlexD

- **Features:**
  - Supports LIN protocol version 1.3, 2.0, 2.1 and J2602
  - UART mode: 15/167/8-bit data, parity/no-parity, 1/2/3 stop bit
- **LIN Master Mode**
  - Autonomous message handling
  - Once the software has triggered the header transmission, no further intervention needed:
    - until the next header transmission request in transmission mode
    - until the checksum reception in reception mode
- **LIN Slave Mode**
- Software intervention needed only to:
  - Trigger transmission, reception or discard depending on the identifier,
  - Fill the buffer (transmission) or get data from buffer (reception).
- **UART mode**
  - 15/16/7/8 bits data, parity, 1/2/3 stop bits
  - 4-byte buffer for reception, 4-byte buffer for transmission
  - 12-bit counter for timeout management
- **Self-test mode:**
- LINFlex treats transmitted messages as received messages to be independent of external events

UART: UARTCR.B.UART=1
LIN: UARTCR.B.UART=0





FlexCAN

Features

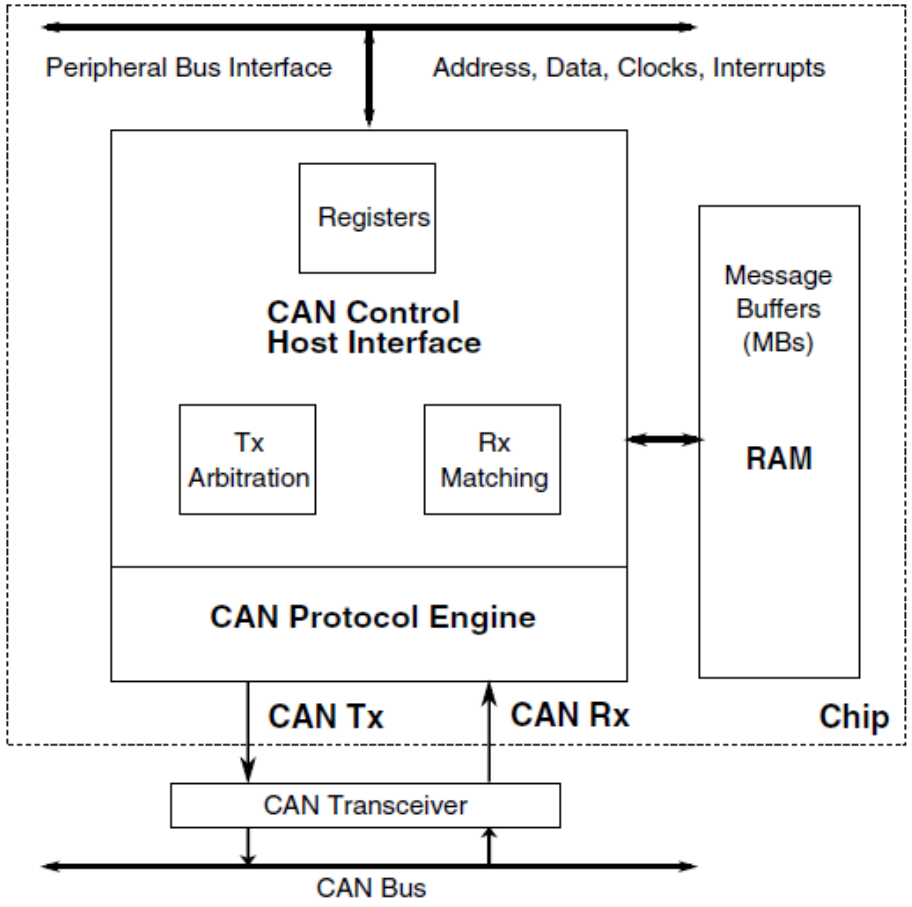
CAN module which supports both CAN A and CAN B specifications

64 message buffers per module

Filters for receive message buffers

Message buffers and errors can cause interrupts

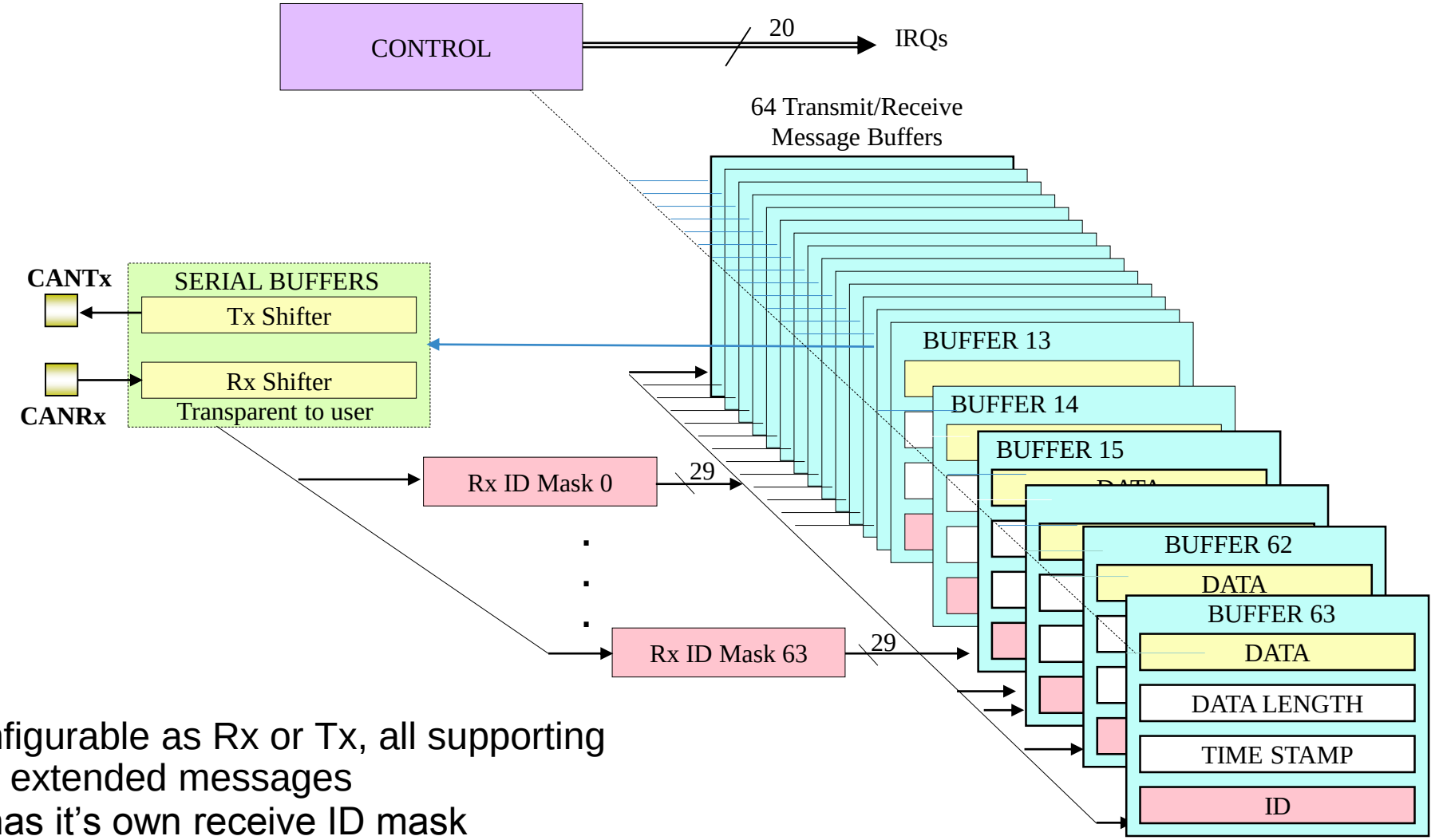
Programmable loop-back for self test operation



Features

- Full Implementation of the CAN Protocol
  - Standard ID and Remote Frames
  - Extended ID and Remote Frames
  - Zero to eight bytes data
  - Programmable bit rate up to 1 Mb/sec
- CAN clock from Either Internal (PLL) or External Source (OSC)
- Unused Message Buffer space can be used as general purpose RAM
- 16-bit time Stamp
- Independent of the transmission medium
- Maskable interrupts

FlexCAN: Architecture



- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Each buffer has it's own receive ID mask

FlexCAN: Message Buffer Structure

FlexCAN Message Buffer Structure

0x0	-	**CODE**	- SRR* IDE RTR	LENGTH	TIME STAMP
0x4	-	ID (Extended/Standard) |			ID (Extended) *
0x8	Data Byte 0		Data Byte 1	Data Byte 2	Data Byte 3
0xC	Data Byte 4		Data Byte 5	Data Byte 6	Data Byte 7

Receive Buffer Codes

Before	After	Description
0000	**-**	**Buffer not active**
0010	**0010 / 0110**	**Buffer is Full/Overrun**
0100	**0010**	**Buffer Active & Empty**
0110	**0110**	**Overrun (2nd frame rec'd before CPU read 1st frame)**
0101	**0010**	**An empty buffer was filled**
0011	**0110**	**A full/overrun buf. was filled**

Transmit Buffer Codes

Before	After	Description
1000	**-**	**Buffer not ready to transmit**
1100	**1000**	**Tx buffer ready to transmit once**
1100	**0100**	**Remote frame will transfer and msg. buf. becomes Rx buf.**
1010	**1010**	**Data frame will transfer only as a response to remote frame**
1110	**1010**	**Data frame to transfer once then only as a response to frame, always**

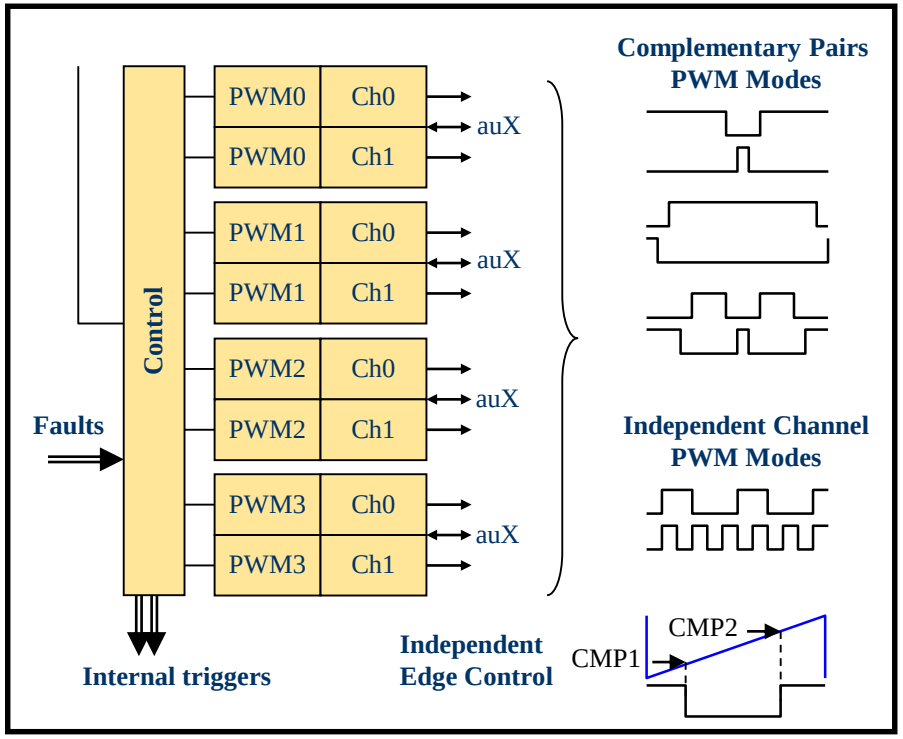
Before buffer codes: Host CPU writes commands **After** buffer codes: Reflects the status of that buffer.

* Fields ignored when using standard frames (IDE=0)

Demo

FlexPWM

FlexPWM



- Permanent Magnet Synchronous Motor (PMSM, PMAC),
- Brushless DC motor (BLDC),
- Brush DC motor (BDC),
- AC Induction Motor (ACIM),
- Switched Reluctance Motor (SRM),
- Variable Reluctance Motor (VRM),
- Stepper Motors
- DC/DC converters.

Main Features

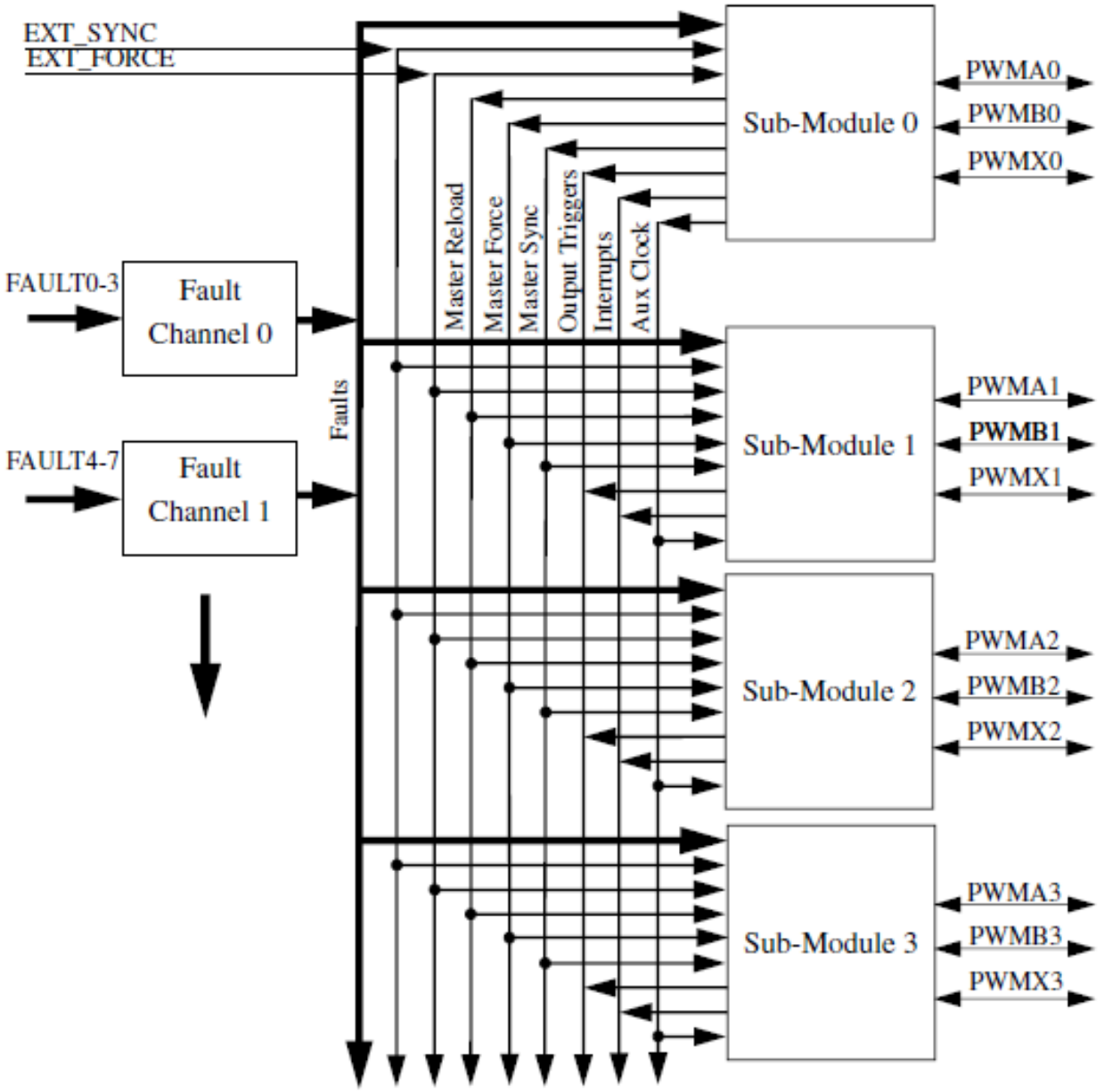
- 4 Submodules, each with complementary PWM generation, Isense IC/OC and Fault Input
- 16 bits of resolution for center, edge aligned, and asymmetrical PWMs
- PWM outputs can operate as complimentary pairs or independent channels
- Independent control of both edges of each PWM output
- Independently programmable PWM output polarity
- Separate dead time for rising and falling edges
- Each complementary pair can operate with its own PWM frequency and deadtime values
- All outputs can be programmed to change simultaneously via a "Force Out" event
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability

Safety

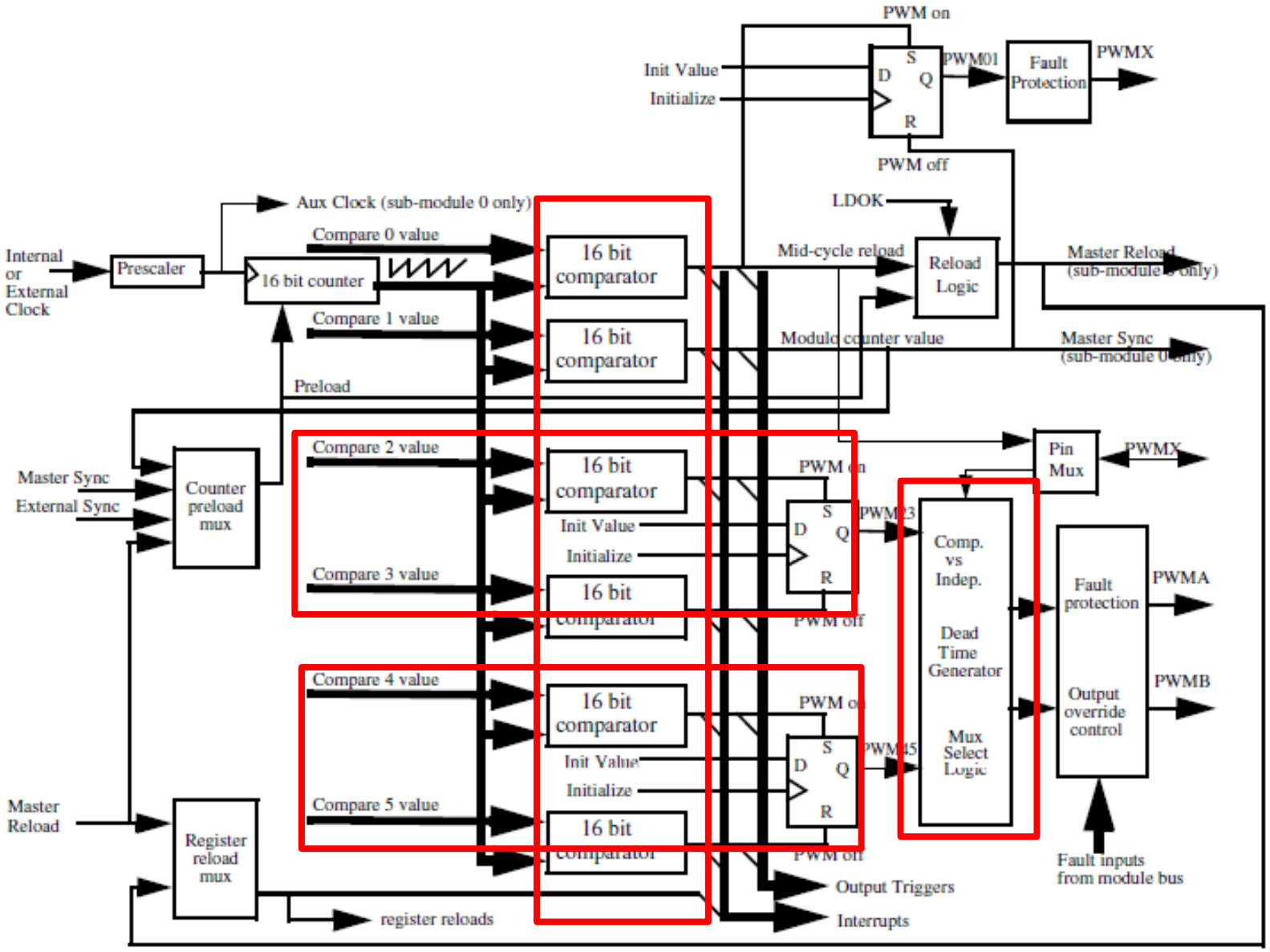
- Write protection for critical registers
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs

FlexPWM

- FLEXPWM Motor Controller.
- Contains 4 PWM sub-modules each set-up to control an H-bridge power stage and 4 fault inputs.
- Capable of controlling most motor types:
  - AC induction motors
  - Permanent Magnet AC Motors
  - Brushless & brushed DC Motors
  - Switched & Variable Reluctant Motors
  - And Stepper Motors



FlexPWM Sub-Module Block Diagram



FlexPWM External Signal Descriptions

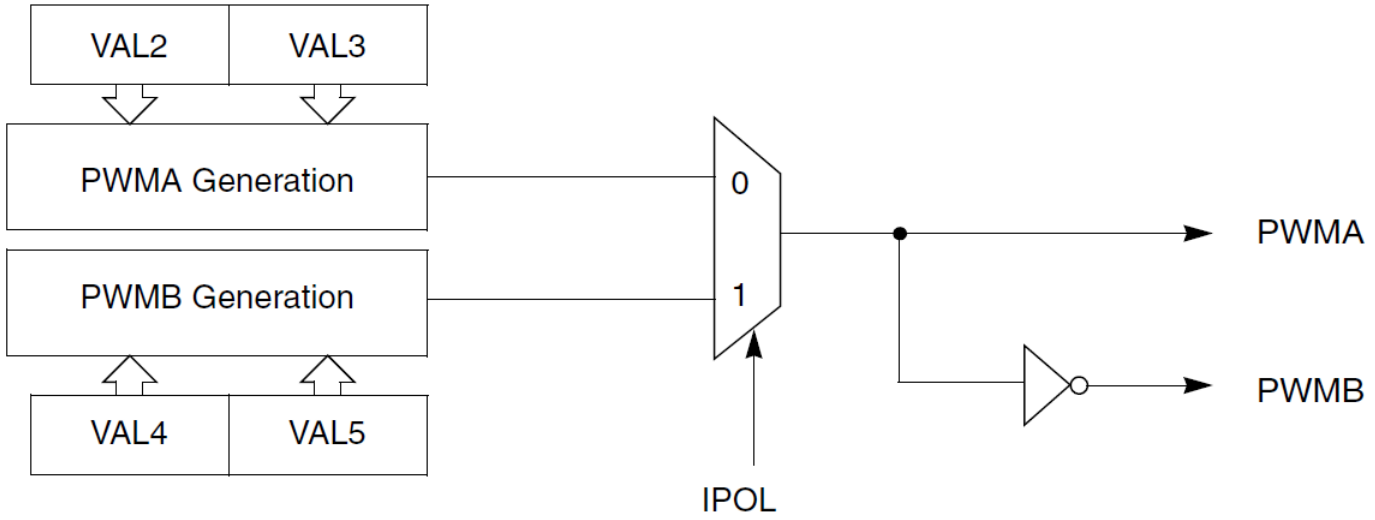
- PWMA[0-3] & PWMB[0-3] – The output pins of the PWM channels
- PWMX[0-3] – The aux output pins of the PWM channels
- Fault[0-3] – input pins for disabling selected PWM channels
- EXT_SYNC – allows a source external to the PWM to initialise the PWM counter
- EXT_FORCE – allows a source external to the PWM to force an update of PWM output
- OUT_TRIG – allows the PWM sub-module to control timing of the ADC conversions.
- EXT_CLK – allows an on-chip external source to control the PWM clocking, allowing the PWM to synchronised to the timer.

FlexPWM

- **Independent or Complementary Channel Operation**

-Writing a logic one to the INDEP bit of the CNFG register configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

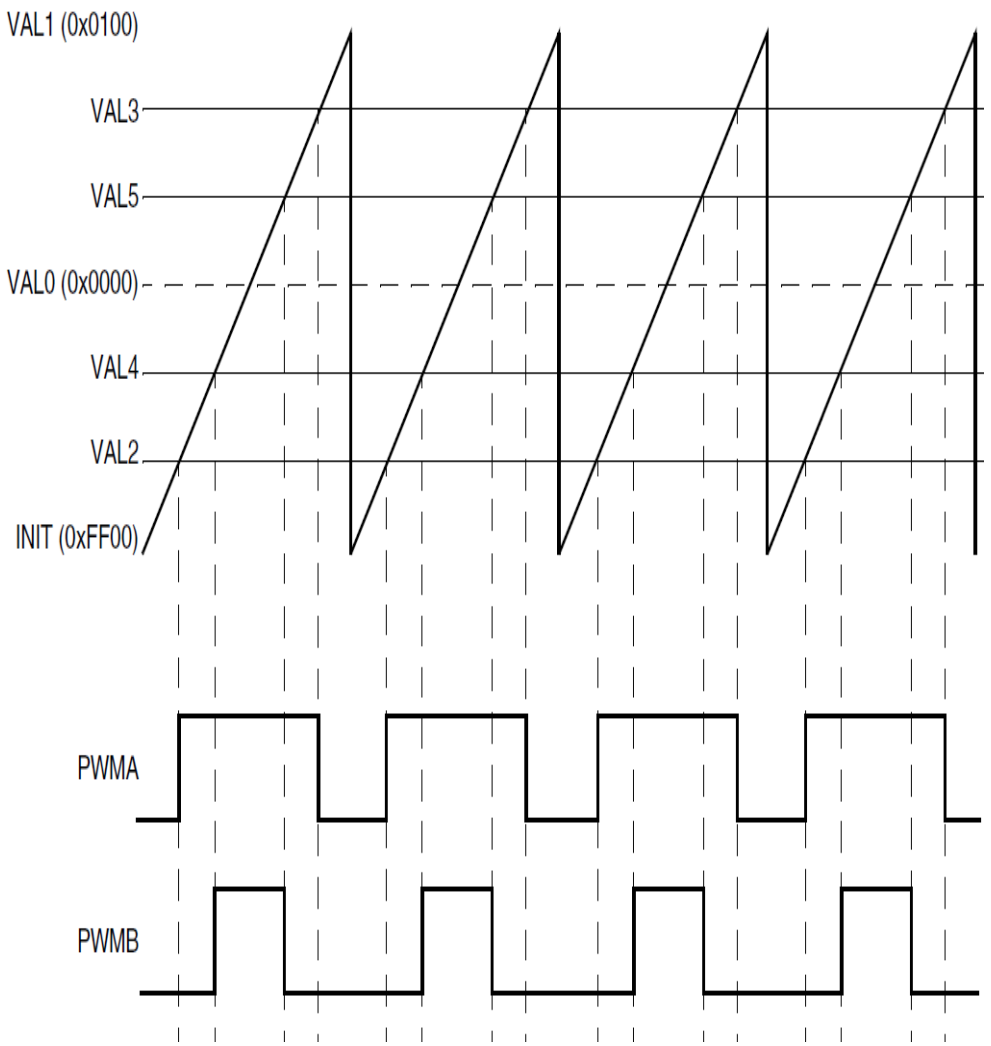
-Writing a logic zero to the INDEP bit configures the PWM output as a pair of complementary channels. The IPOL bit determines which signal is connected to the output pin (PWMA or PWMB).



FlexPWM

Centre Aligned PWMs

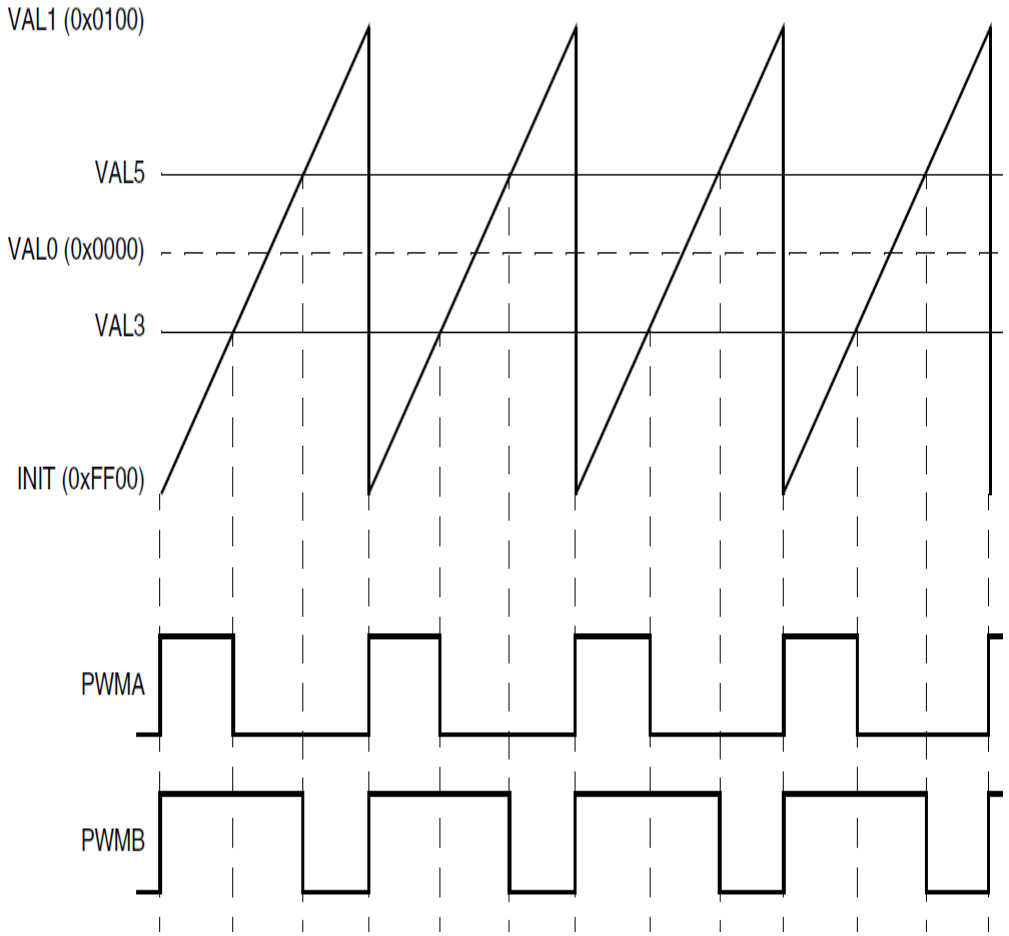
- The turn on edge and the turn off edge determine the pulse width.
  - The edge times can be signed and values that are 2's complements of each other give centre-aligned PWMs
- There is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.



FlexPWM

Edge Aligned PWMs

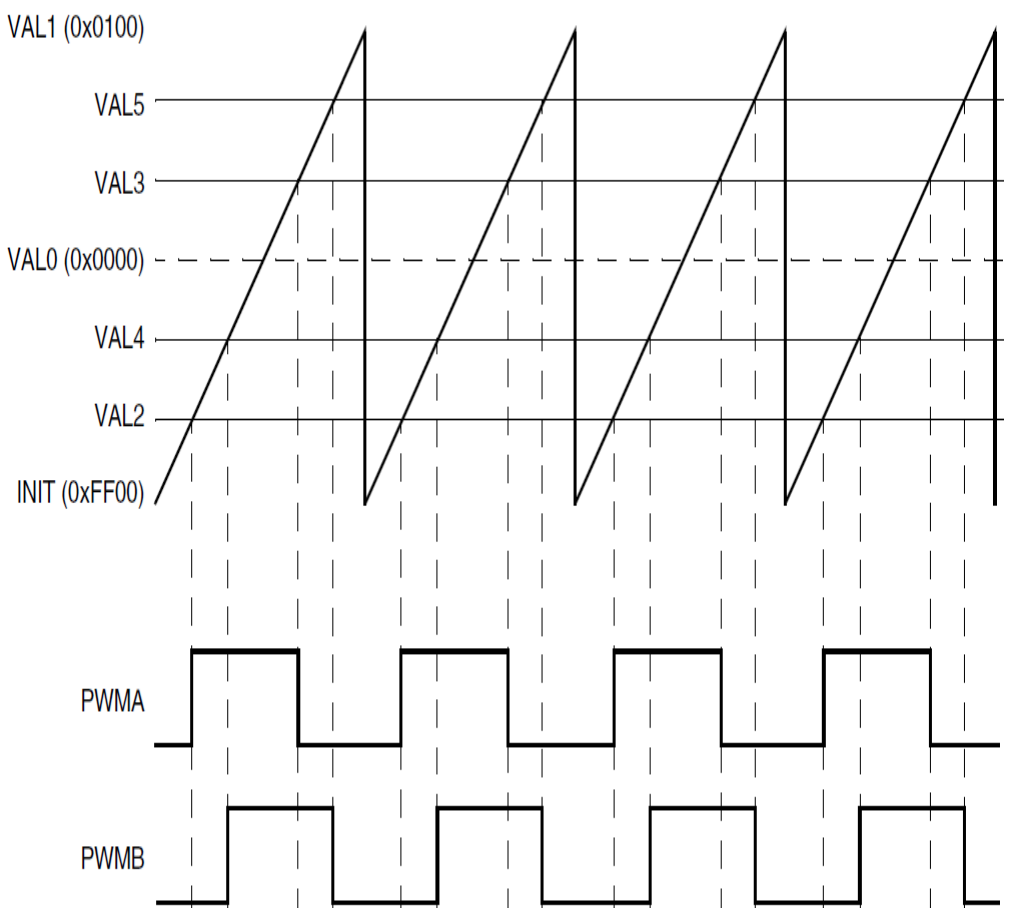
- The INIT value can be used to edge align the turn on edge.
- Setting the turn on edge to the same value as the INIT value results in left-aligned PWMs (INIT = VAL2 = VAL4)
  - Only the turn off edge changes the pulse width.



FlexPWM

Phase Shifted PWMs

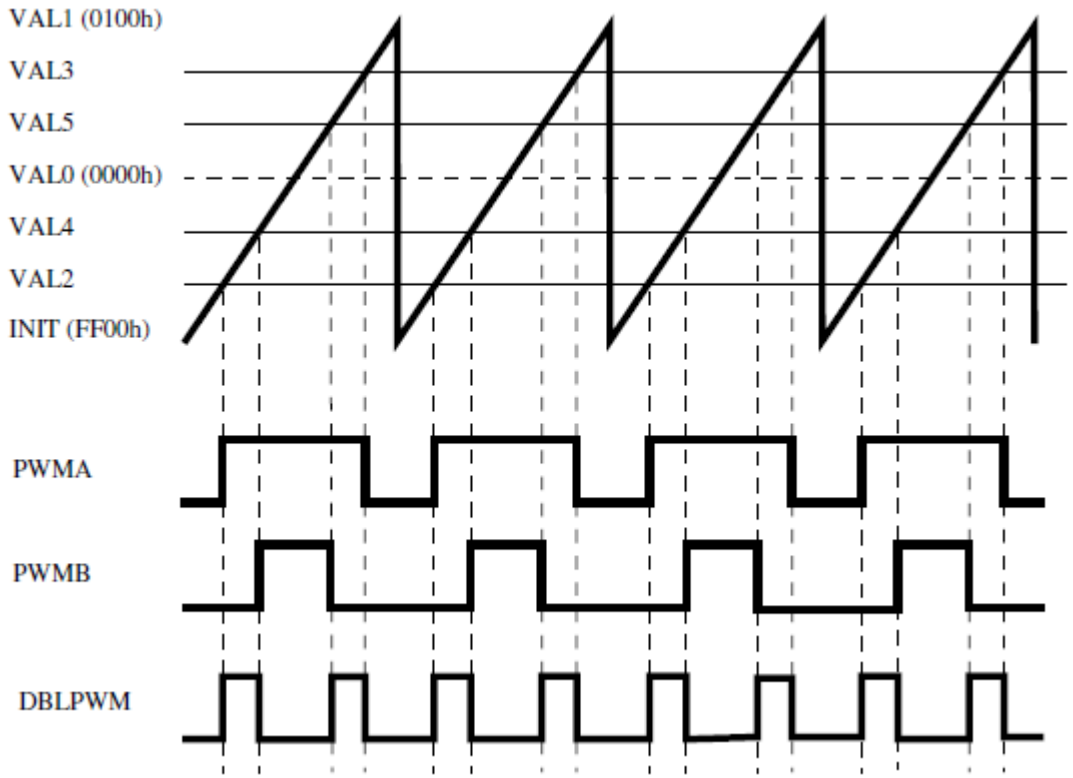
- It is possible to have phase-shifted PWMs if numerical biases are applied to different PWM channels
- Phase switching can open up timing windows between switching edges.
  - This reduces noise and allows a signal to be sampled by the ADC.
- Phase shifting is ideally suited for transformer loads.



Double Switching PWMs

- Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction.

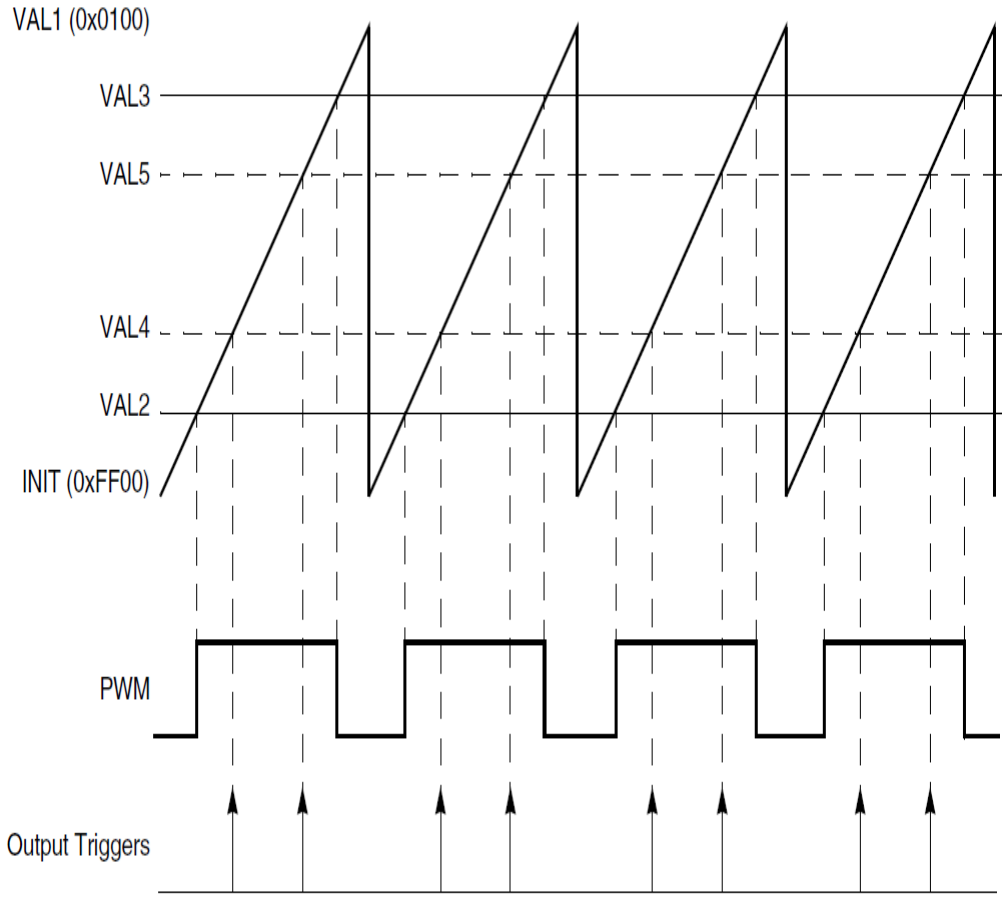
- This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel



FlexPWM

ADC Triggering

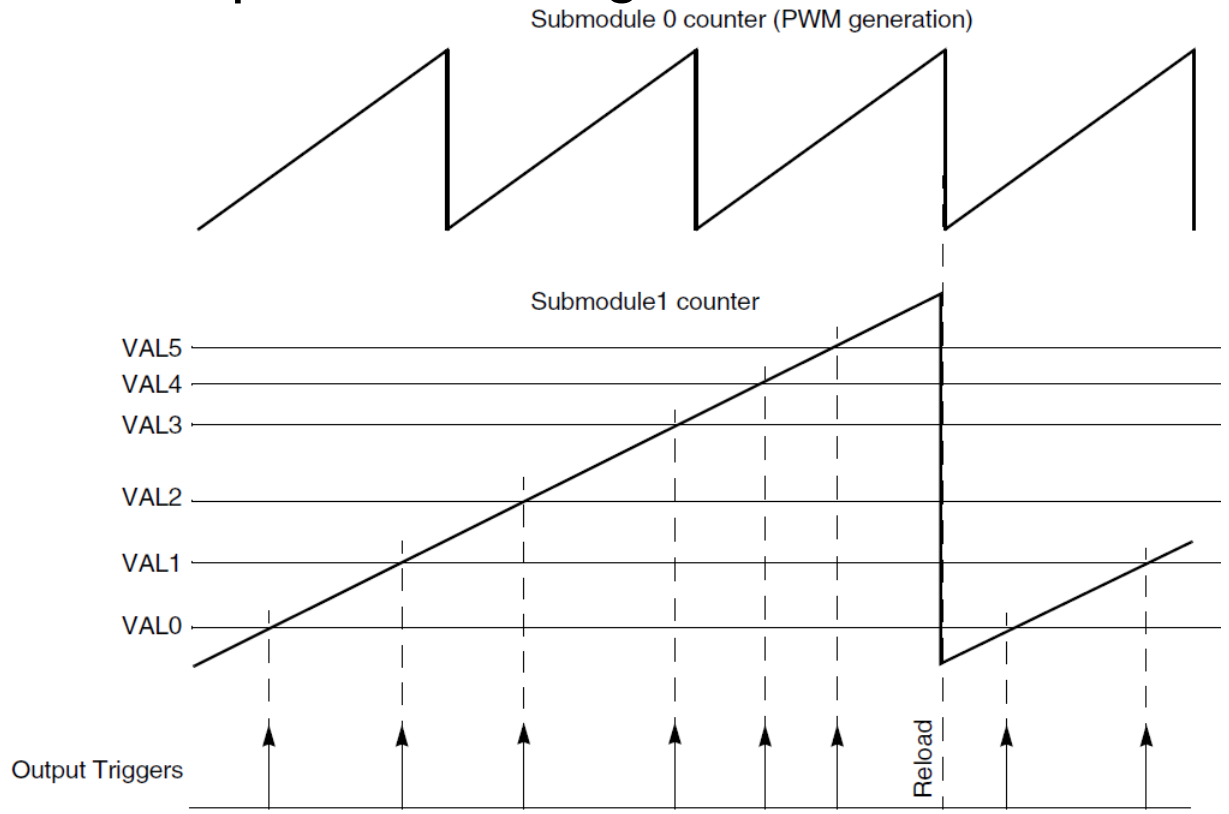
- When the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated.
  - Unused comparators can provide this function
- Multiple ADC triggers can be generated in hardware per PWM cycle.



FlexPWM

ADC Triggering

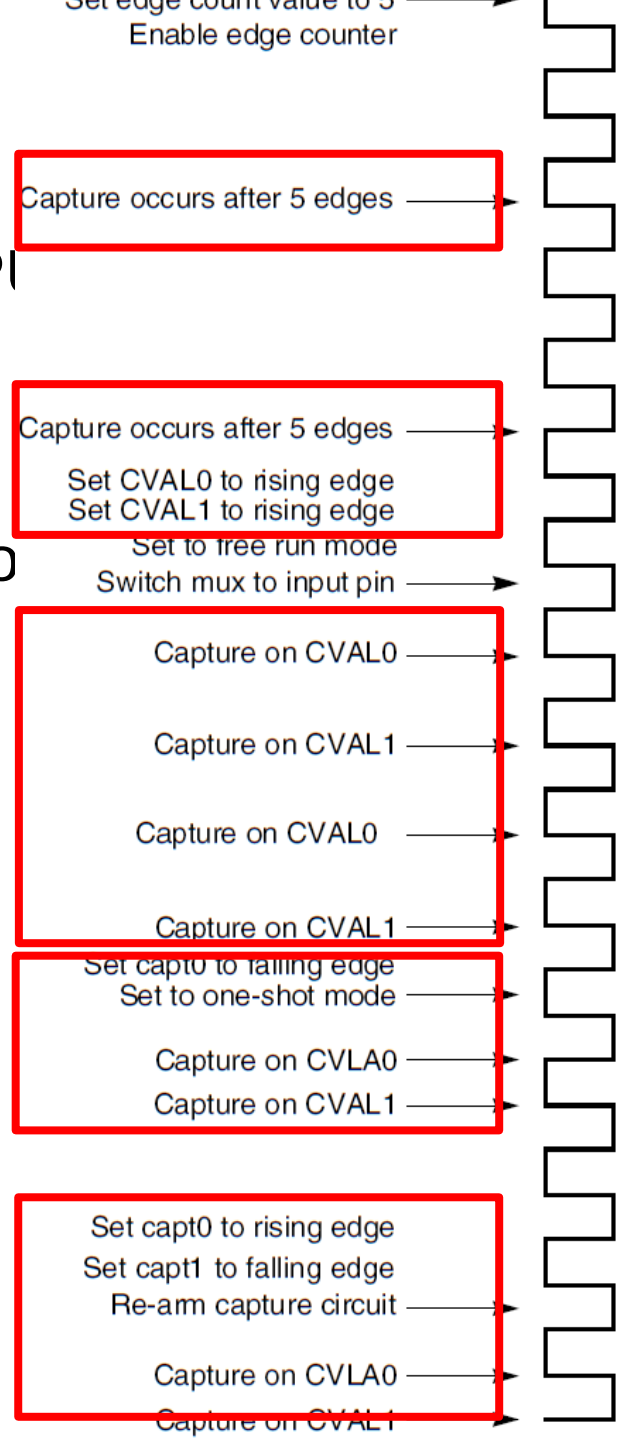
- It’s possible to synchronise sub-modules to sub-module 0 and run the secondary module at a slower rate
- The secondary module comparators can generate even more ADC triggers



FlexPWM

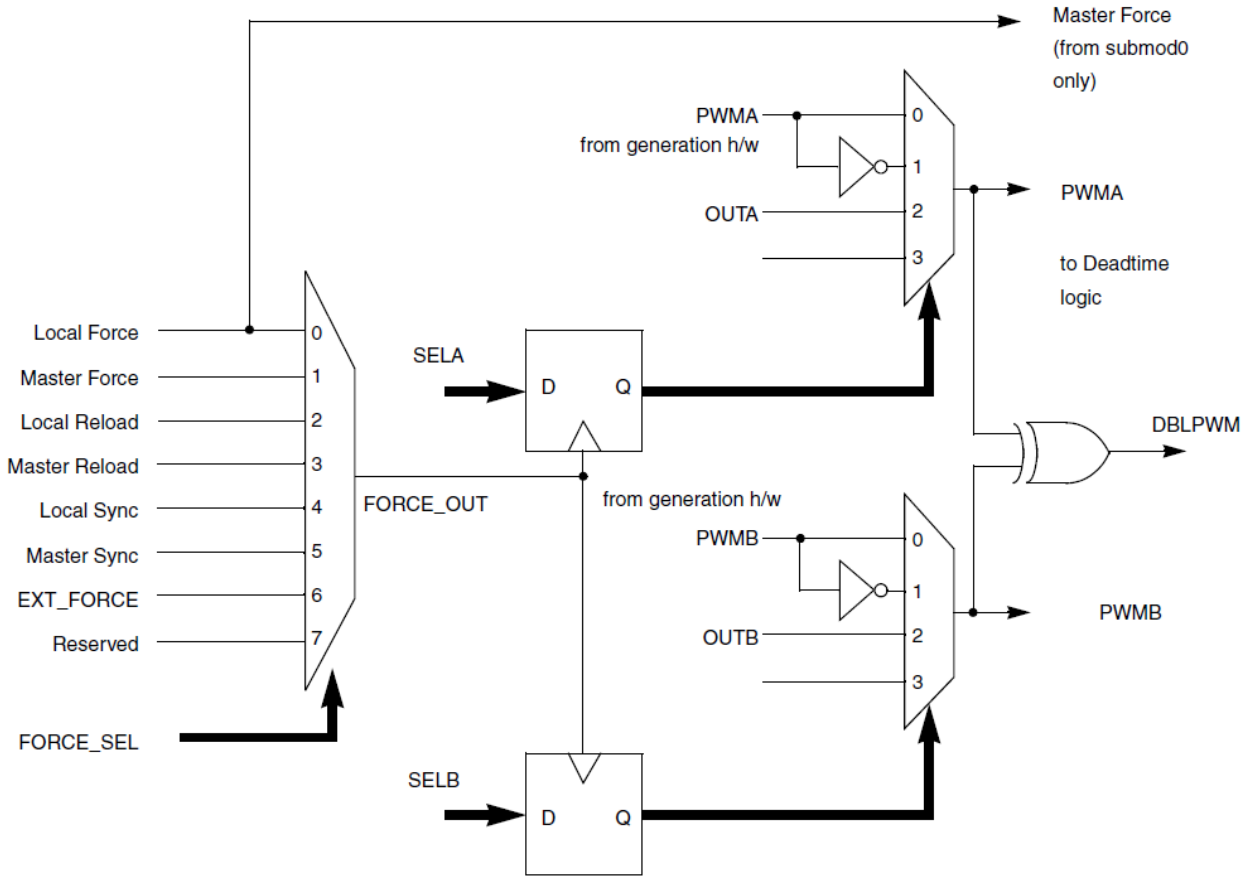
Enhanced Capture (Input-Capture)

- The PWM channel can be used as an INPU
- Free running or one shot mode
- Calculate pulse widths
- The module can count a specific number o performing a capture or interrupt.



FlexPWM Force output controls

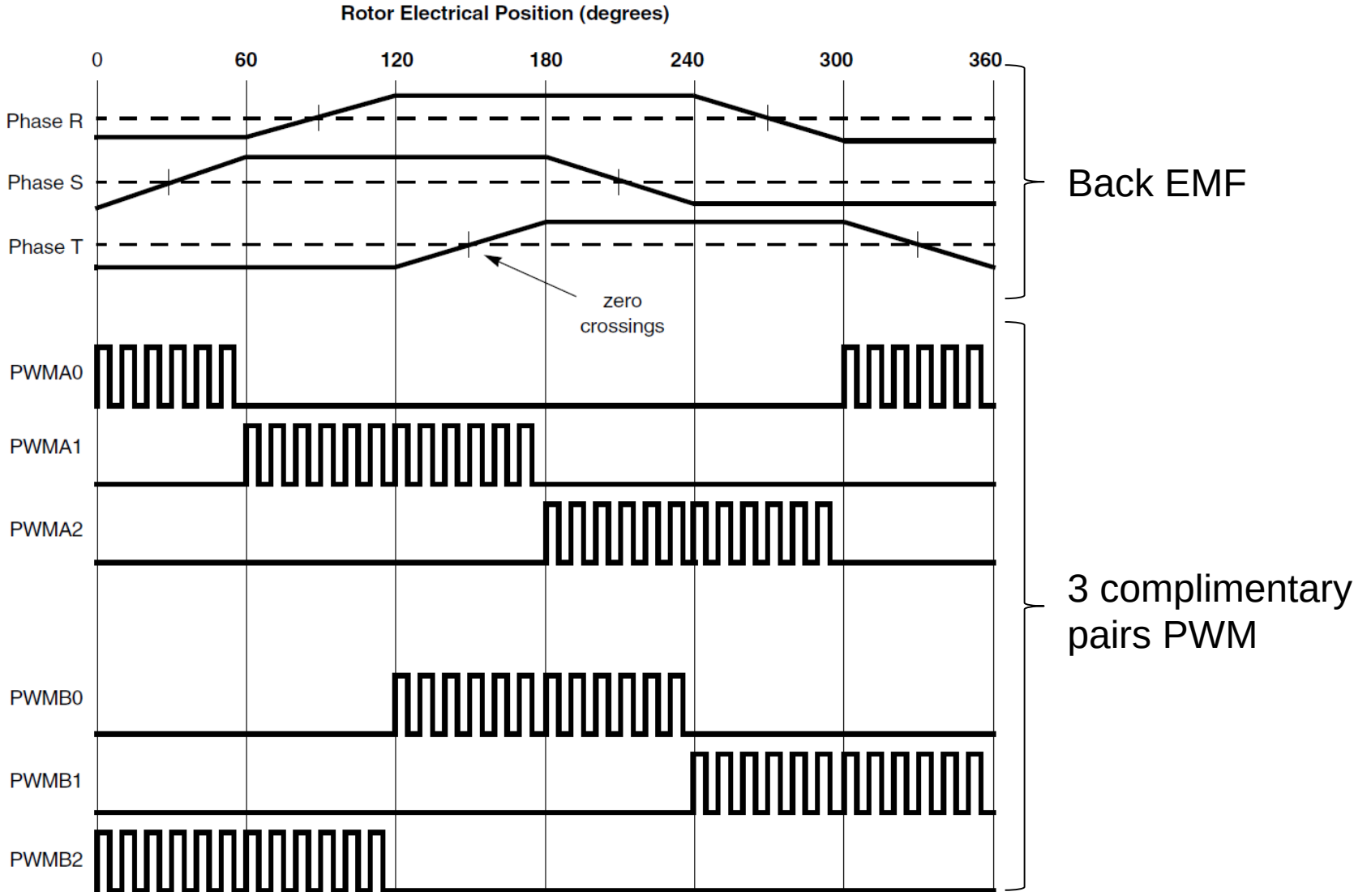
- Allows each sub-module to switch outputs between any internally sourced signal or off
- update all of the submodule outputs at the same time.
- The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter



Sensorless BLDC motor commutation

- The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event.

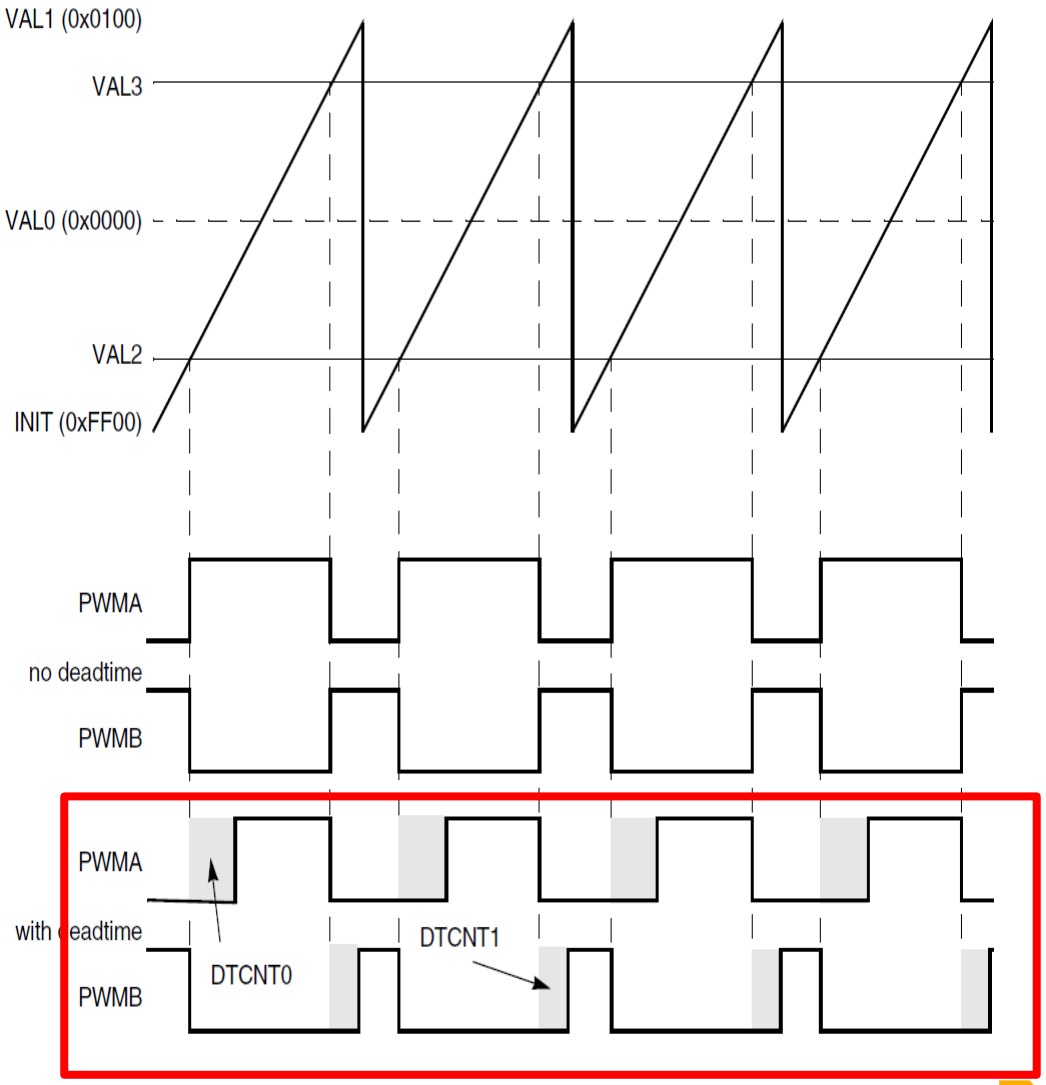
- immediately changes the state of the PWM pins to the next commutation state with no software latency.



FLEXPWM

Deadtime Insertion

- Deadtime is used to create non overlapping complementary signals.
- The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs.
- The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay.

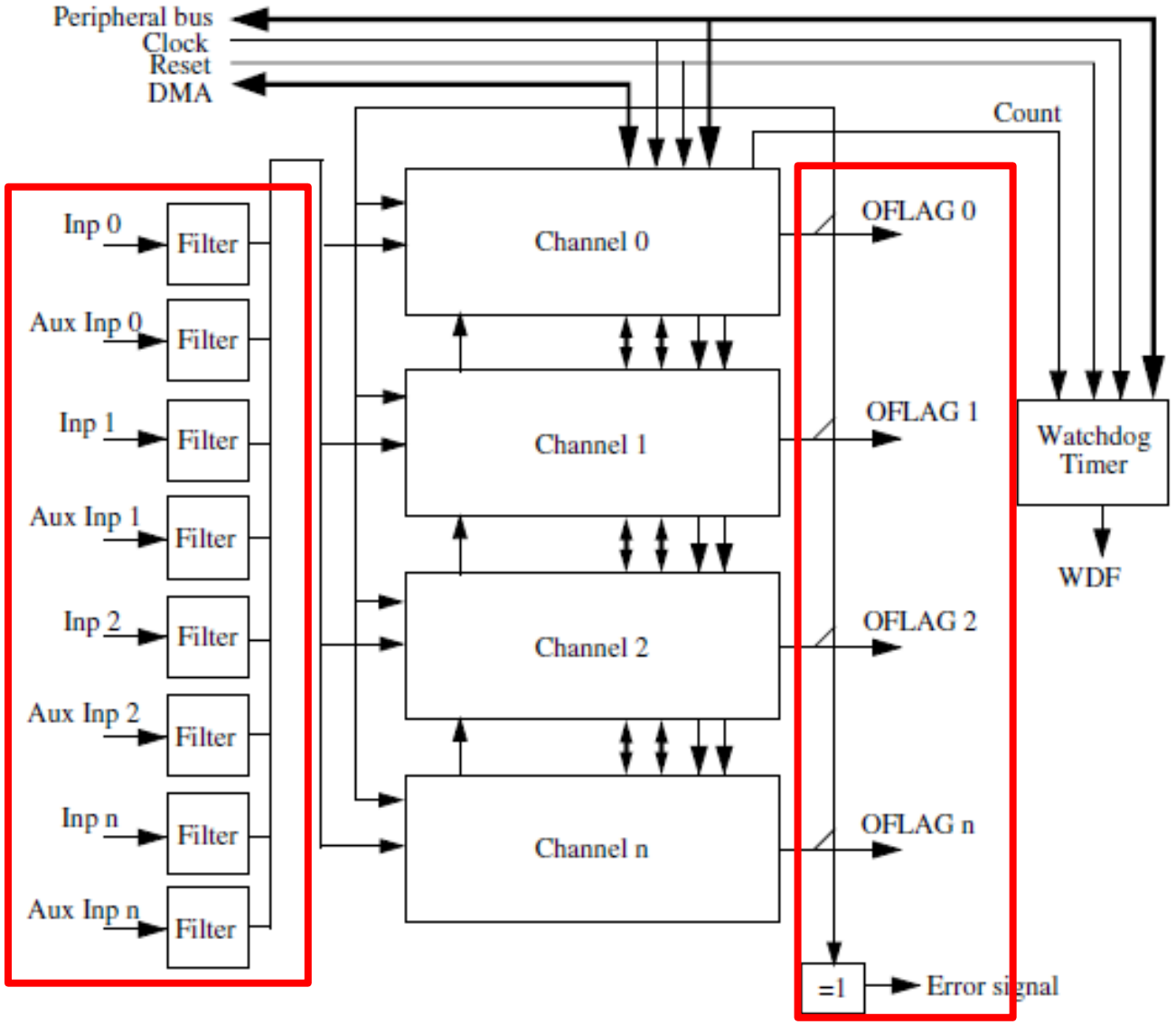


eTimer

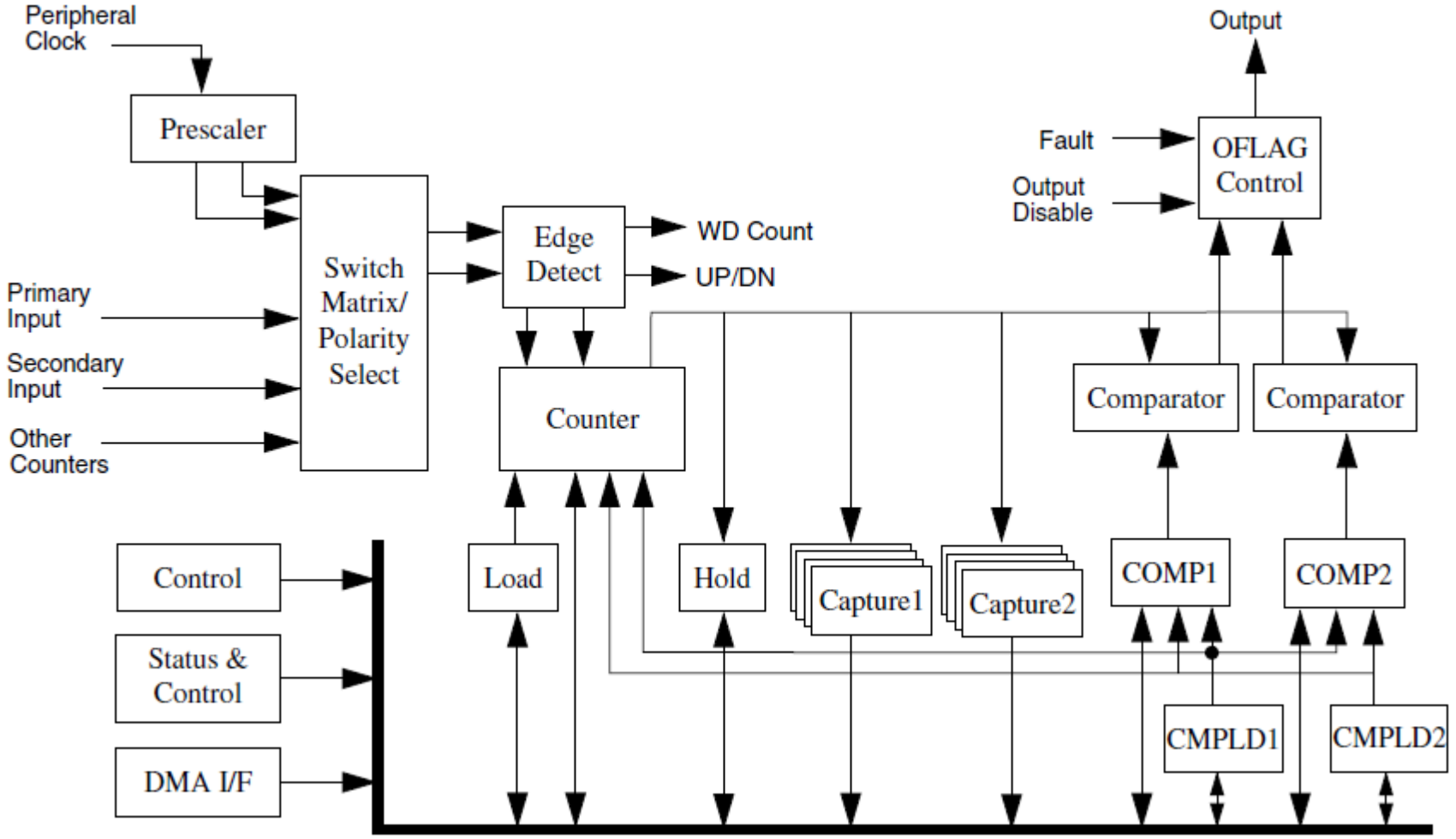
eTimer Overview

- Contains 3 general purpose eTimer units
- Each module features six highly-flexible, self-contained 16-bit channels each containing
  - Prescaler and Counter
  - Load register and hold register
  - Two queued capture FIFOs
  - Two compare registers and two compare pre-load registers
  - Input filter
  - Interface to TCU and DMA capability
- For each timer there is a choice of counting modes and primary and secondary count sources
  - Counting modes determine how the count behaves
  - Count sources determine what the counter counts and how those counts are enabled, gated or captured

eTimer Block Diagram



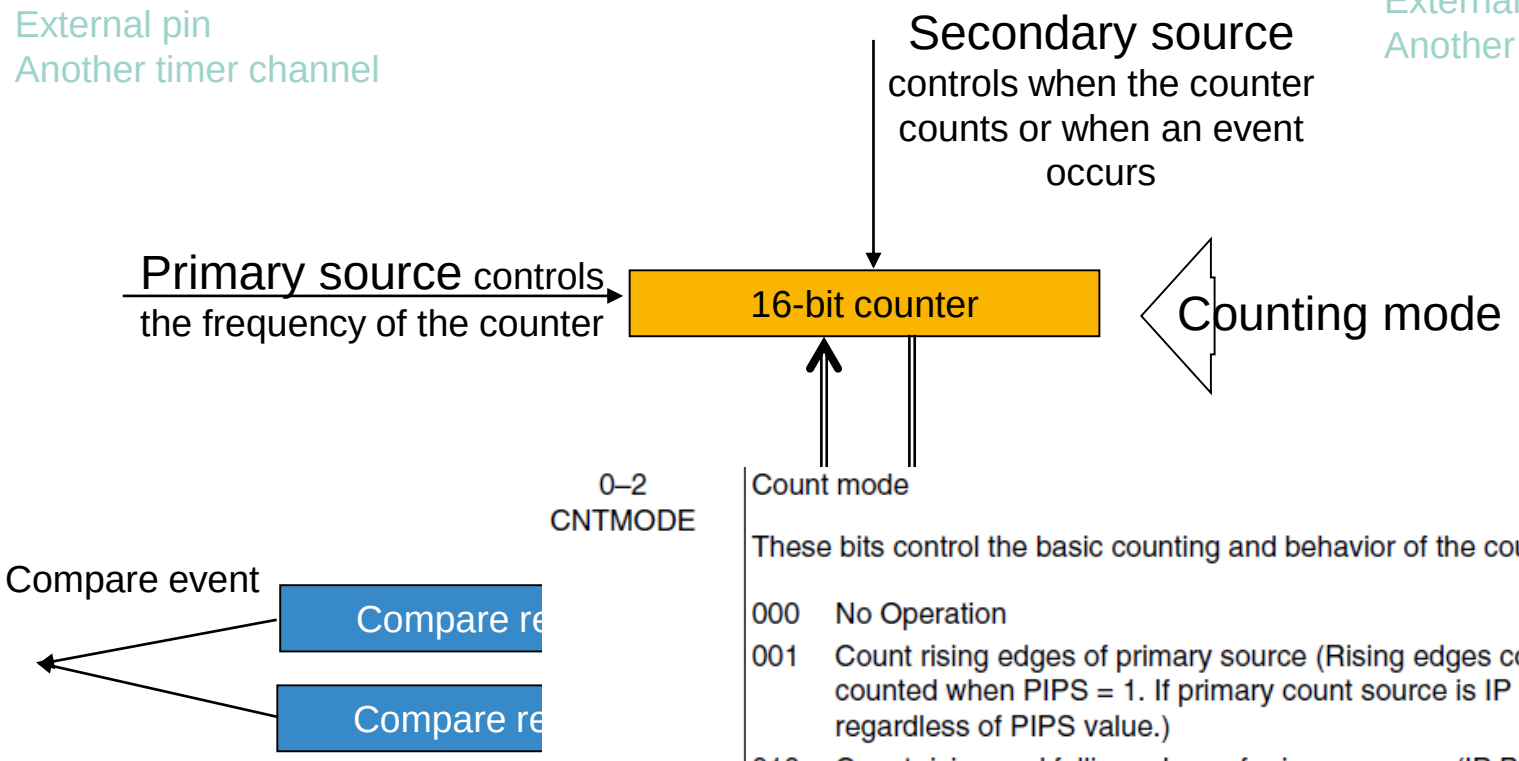
eTimer Channel Block Diagram



eTimer Conceptual Overview

Primary source can be:
Bus clock (prescaler choice)
External pin
Another timer channel

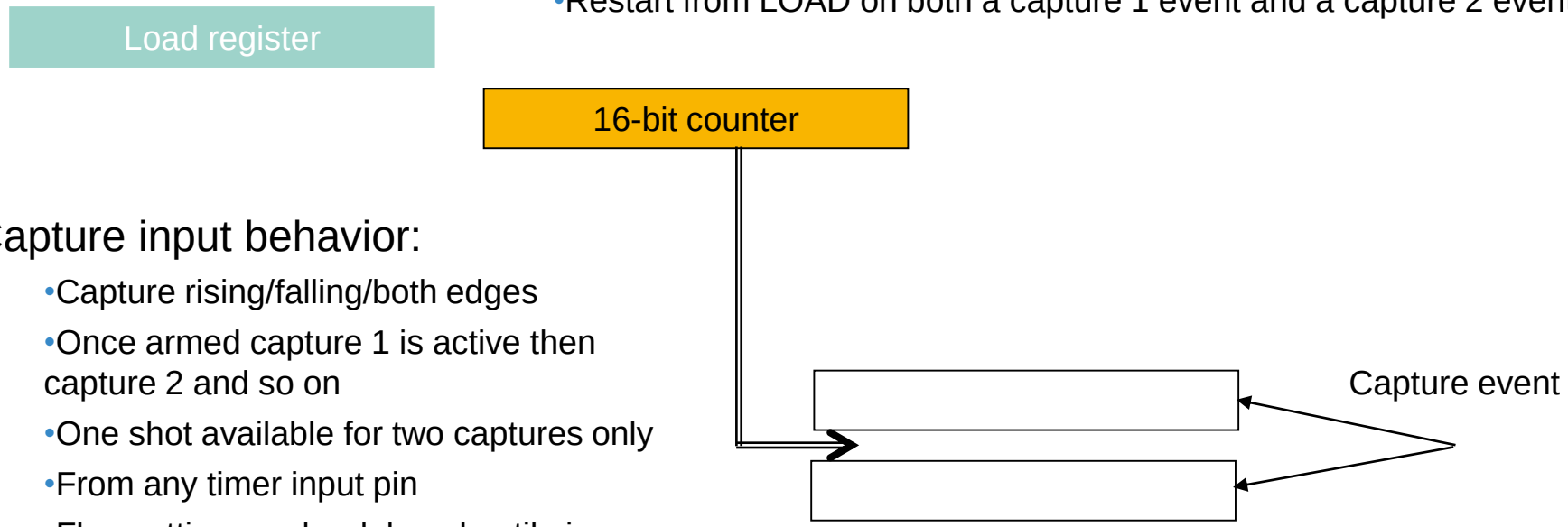
Secondary source can be:
External pin
Another timer channel



0–2 CNTMODE	Count mode These bits control the basic counting and behavior of the counter.	
	000	No Operation
	001	Count rising edges of primary source (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1. If primary count source is IP bus clock, only rising edges are counted regardless of PIPS value.)
	010	Count rising and falling edges of primary source (IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.)
	011	Count rising edges of primary source while secondary input high active
	100	Quadrature count mode, uses primary and secondary sources
	101	Count primary source rising edges, secondary source specifies direction (1 = minus) (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1.)

Input Capture

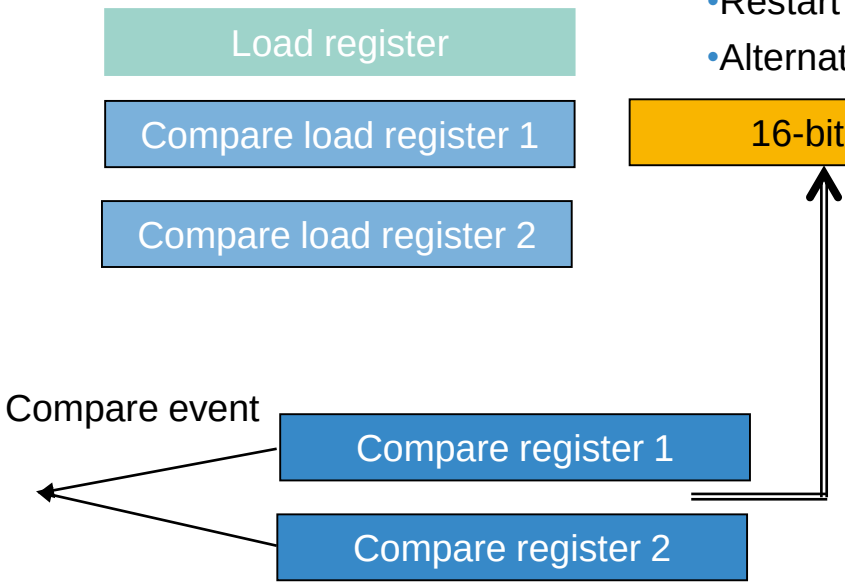
- On capture event timer can:
  - Continue counting
  - Restart from LOAD on a capture 1 event.
  - Restart from LOAD on a capture 2 event.
  - Restart from LOAD on both a capture 1 event and a capture 2 event.



- Capture input behavior:
  - Capture rising/falling/both edges
  - Once armed capture 1 is active then capture 2 and so on
  - One shot available for two captures only
  - From any timer input pin
  - Flag setting can be delayed until given number of FIFO entries are available
  - Filter available to reject glitches

Output Compare

- On compare eventtimer can:
  - Continue counting
  - Restart from LOAD
  - Restart from Compare load register 1
  - Restart from Compare load register 2
  - Alternate restart from Compare load register 1 & 2



- **Output on compare event: CTRL2.OUTMODE**
  - No output
  - Output low/high/toggle on CMP1 or CMP2
  - Toggle output using alternating compare registers
  - Output high on CMP1/CMP2, low on secondary input edge
  - Output high on compare, low on counter roll-over
  - Output high on CMP1, low on CMP2
  - Output high when counter is active, low when stopped.
  - Output high when counting up, low when counting down
  - Output to gated clock while counter is active

CTU
(Cross-Triggering Unit)

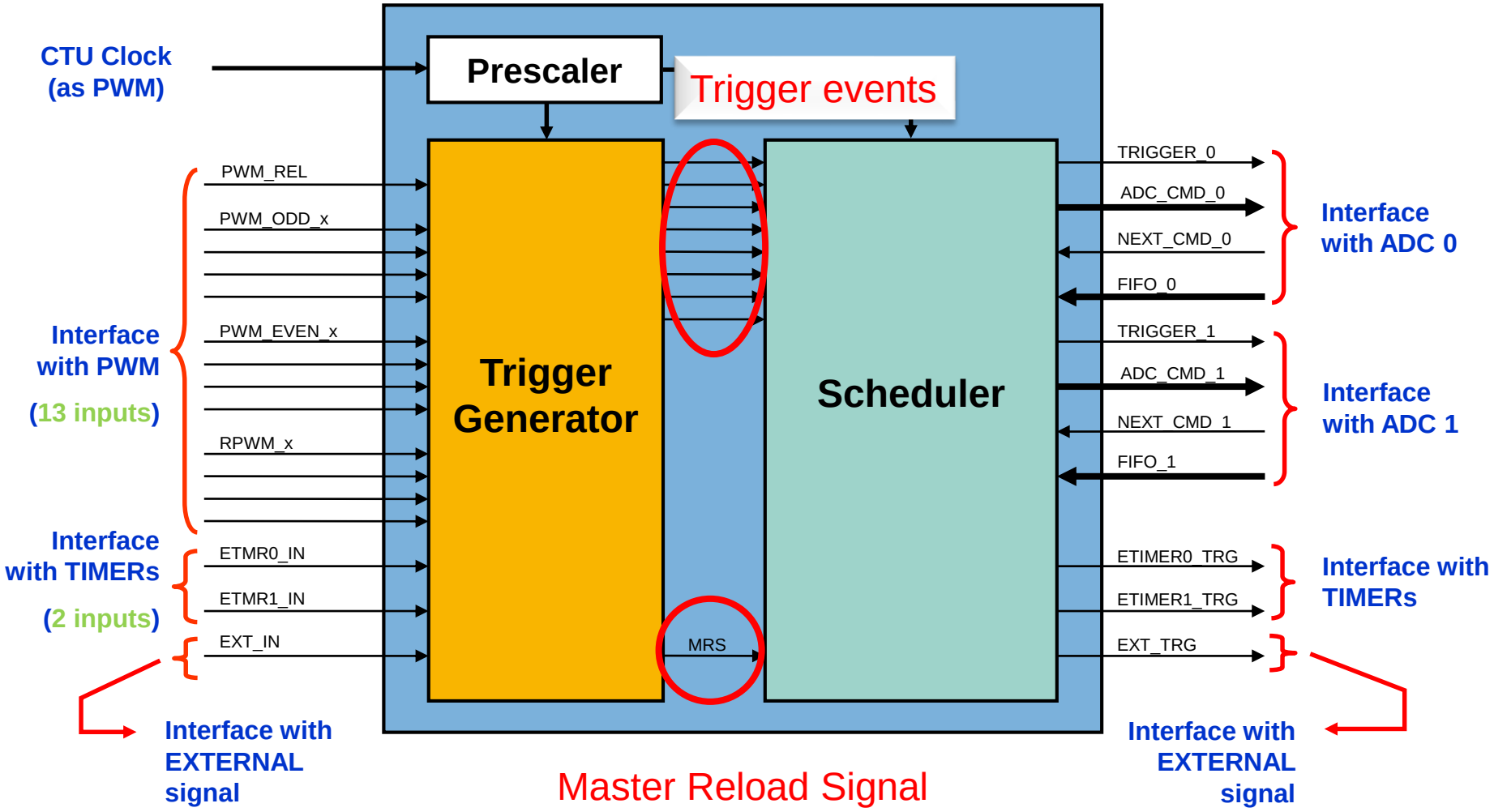
CTU (Cross-Triggering Unit)

- In the PWM driven systems it is important to precisely schedule acquisition of the state variables with respect to PWM cycle

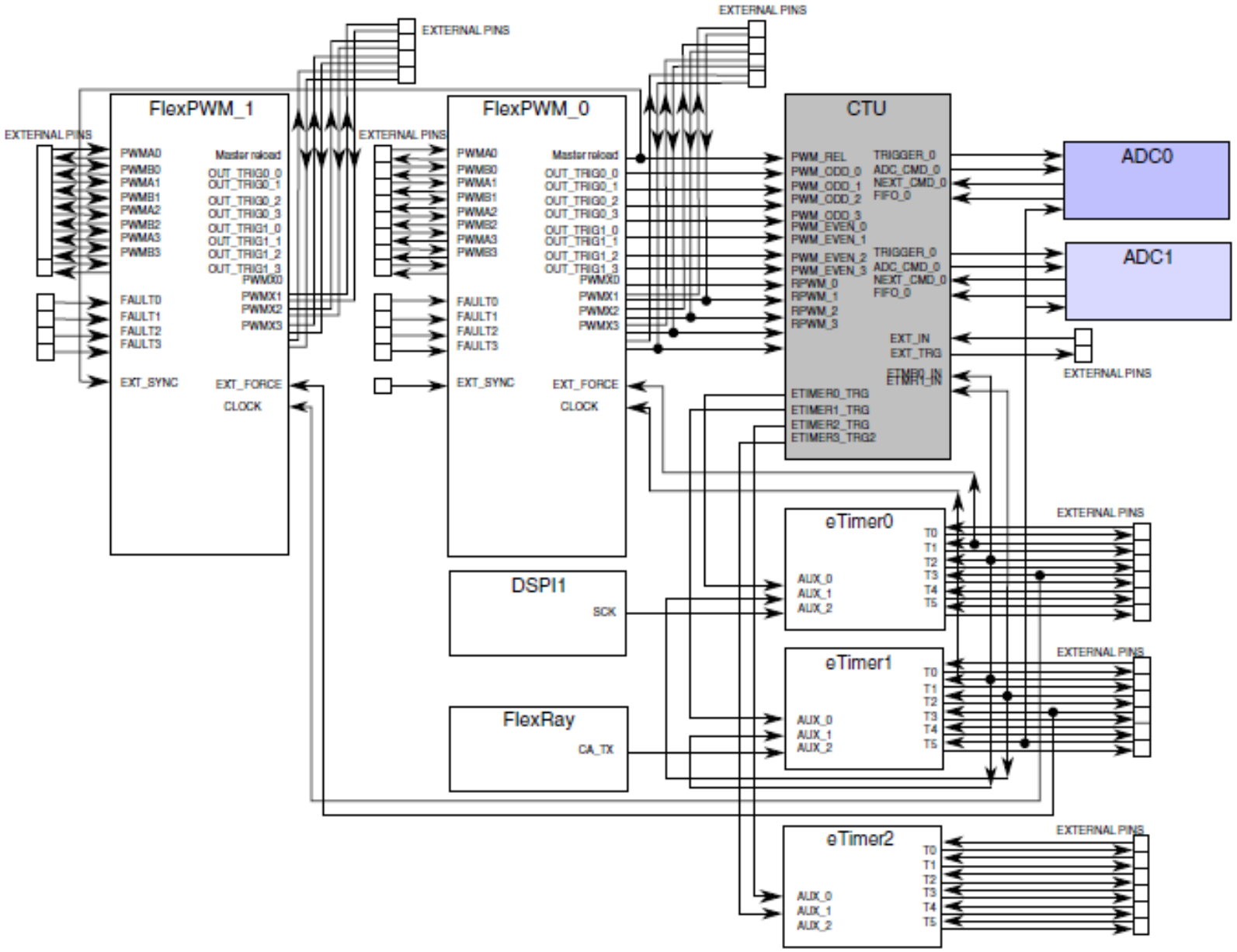
- The CTU receives several signals from different sources such as PWM, timers, position decoder and/or external pins. These signals are then processed to generate up to eight trigger events. An input event can be a rising edge, a falling edge or both edges of the incoming signal.

- The output can be a pulse, an ADC command, a stream of consecutive ADC commands for over-sampling support. The outputs are targeted to one or more peripherals such as ADCs and timers.

Block Diagram



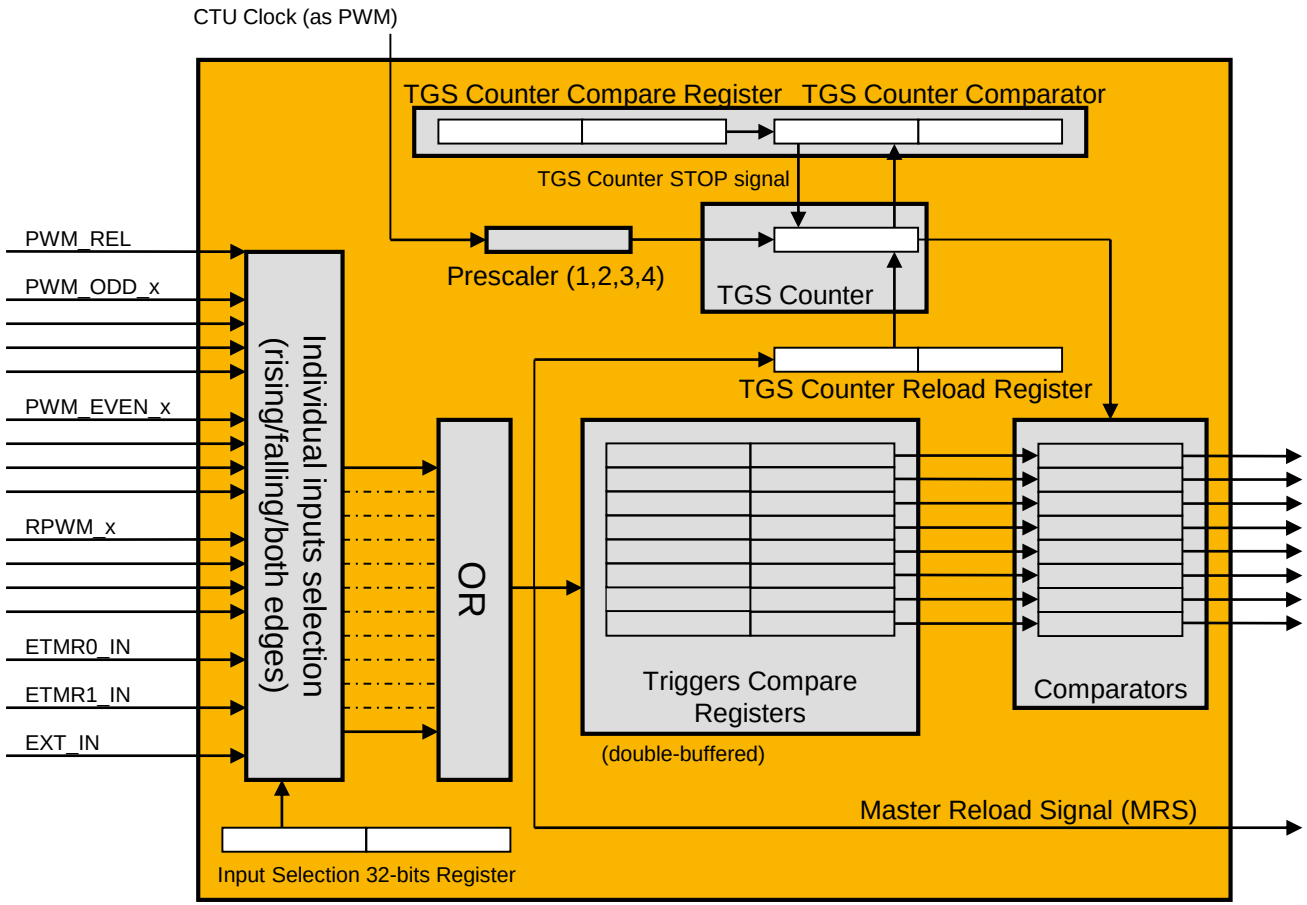
Motor Control Module Interconnections



Trigger Generator

- The Trigger Generator takes the 16 input events from the PWM and eTimer and generates
  - Eight trigger outputs that cause the scheduler to initiate some events
  - A Master Reload Signal
- Each trigger event has the following characteristics:
  - Generation of the trigger event is sequential in time
  - The triggers list uses eight 16-bit double-buffered registers
  - On each Master Reload Signal (MRS), the new triggers list is loaded
  - The triggers list is reloaded on a MRS occurrence, only if the reload enable bit is set

CTU Trigger Generator - Triggered Mode



2. Comparator matches raise triggers

1. MRS causes reload of trigger comparators and TGS (Trigger Generator Subunit) counter

Trigger Generator in Triggered Mode

- Each of the 16 inputs can be configured to be active or inactive
  - Rising edge, falling edge or both
- These active inputs signals are ORed and generate a Master Reload Signal (MRS)
  - This loads the Trigger Generator Counter register, using the preload value written into its double-buffered register
  - It also loads eight Trigger Compare registers
- The Trigger Generator counter begins to increment and when the Compare registers find a match a trigger output occurs

CTU modes

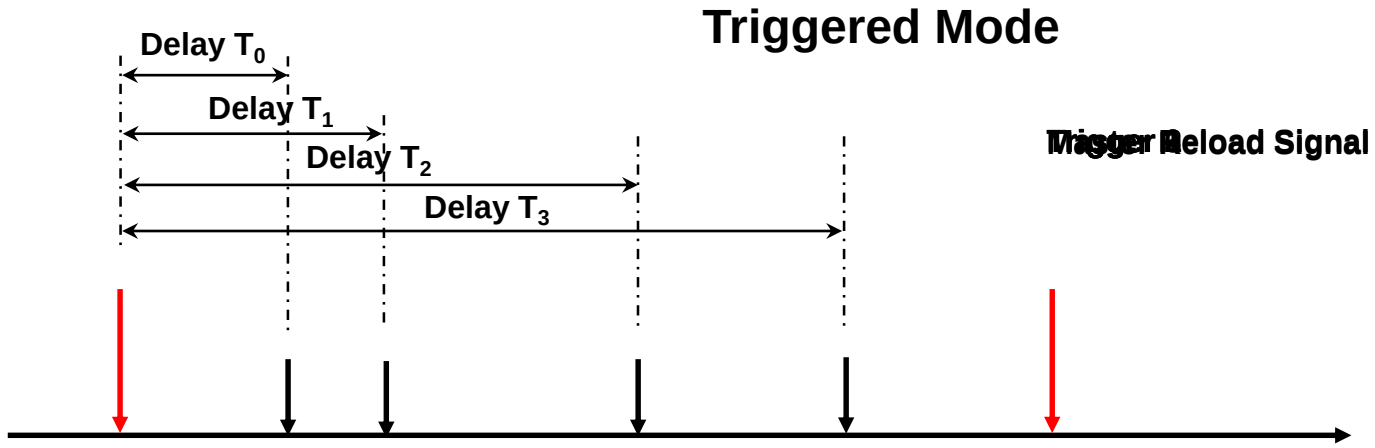
- The CTU has two main modes of operation:

➢ Triggered mode: the input event from the CTU interface is used to generate a sequence of up to 8 triggers to several outputs such as the ADCs, timers and External triggers. Internal sequencer logic is used to schedule the triggers based upon the input event occurrence.

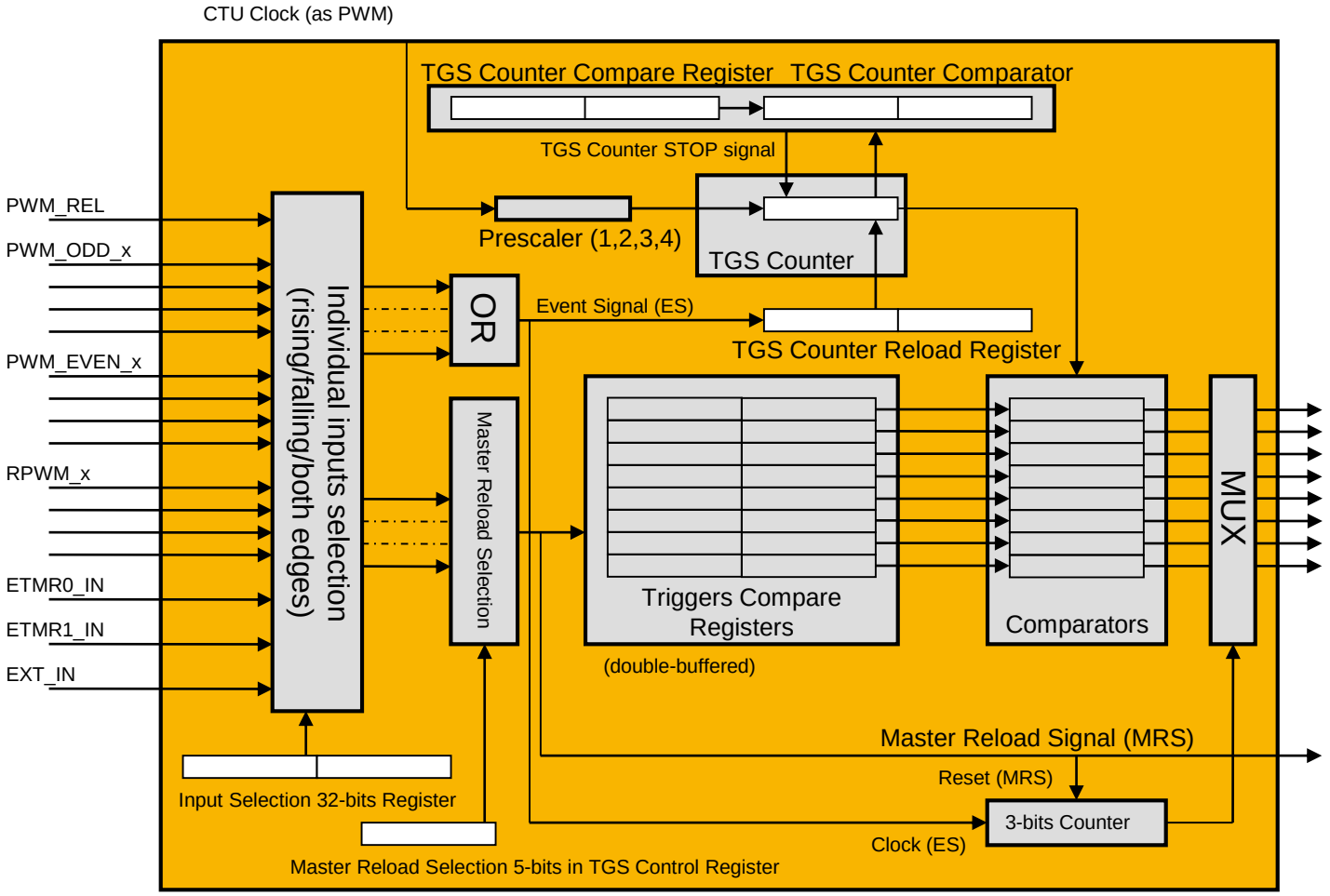
➢ Sequential mode: each input event generates only one trigger for the selected output, such as ADCs, timers and External triggers.

Trigger Generator CTU

Example Timing



CTU Trigger Generator - Sequential Mode

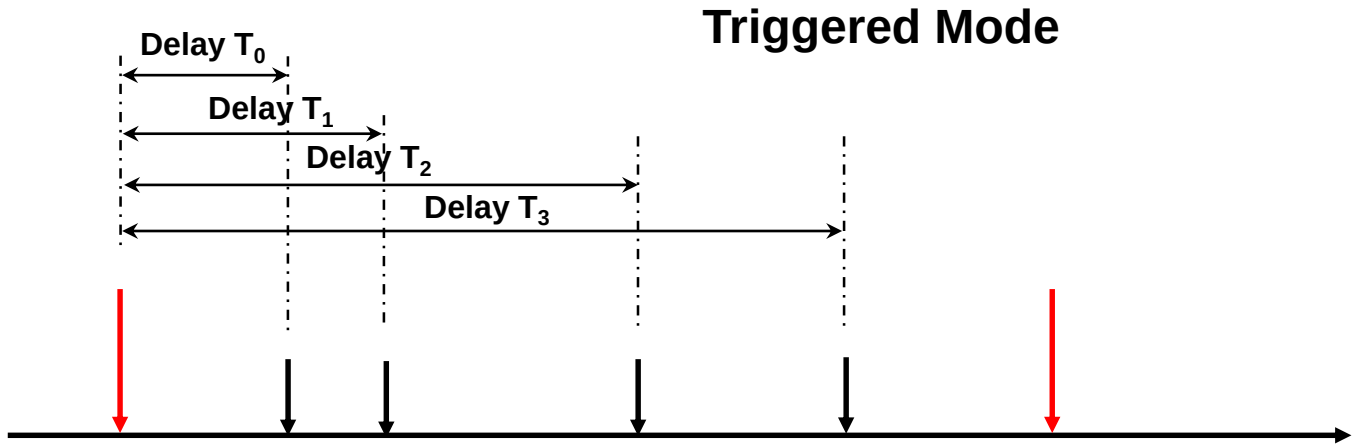


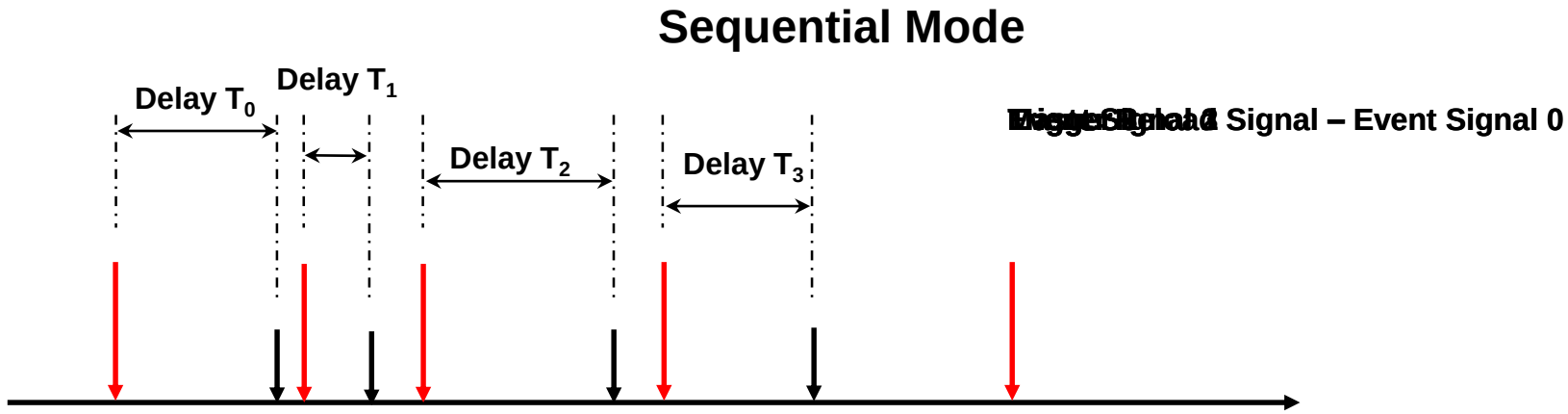
Trigger Generator in Sequential Mode

- Each of the 16 inputs can be configured to be active or inactive
  - Rising edge, falling edge or both
- These active inputs signals are ORed and generate an event signal (ES)
  - This loads the TGS counter register and comparators
  - It also increments a 3-bit counter which select s the next active trigger
- One of the 16 inputs can be selected to trigger the MRS
  - This reloads the triggers list and resets the 3-bit counter
- In this mode, each incoming event sequentially enables only one trigger event through the 3-bit counter
  - The value of the 3-bit counter selects one of the comparators as the next trigger output

Trigger Generator CTU

Example Timing



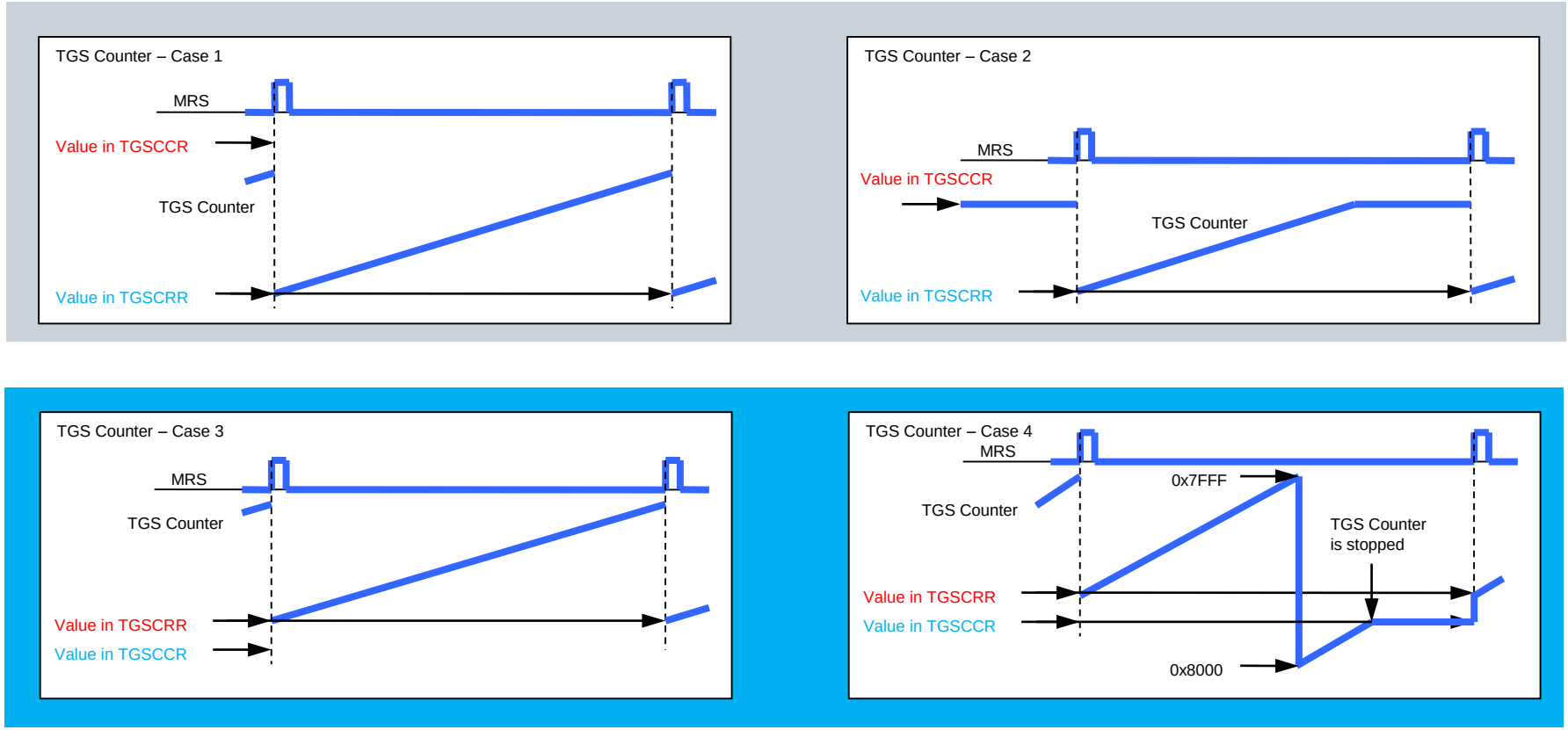


CTU Trigger Generator - Counter

- The TGS counter counts from negative to positive values.The maximum
- counter value is 0x7FFF, after that a counter wrap occurs and the counter value transition from 0x7FFF to 0x8000.

MRS occurs before compare count

MRS occurs after compare count



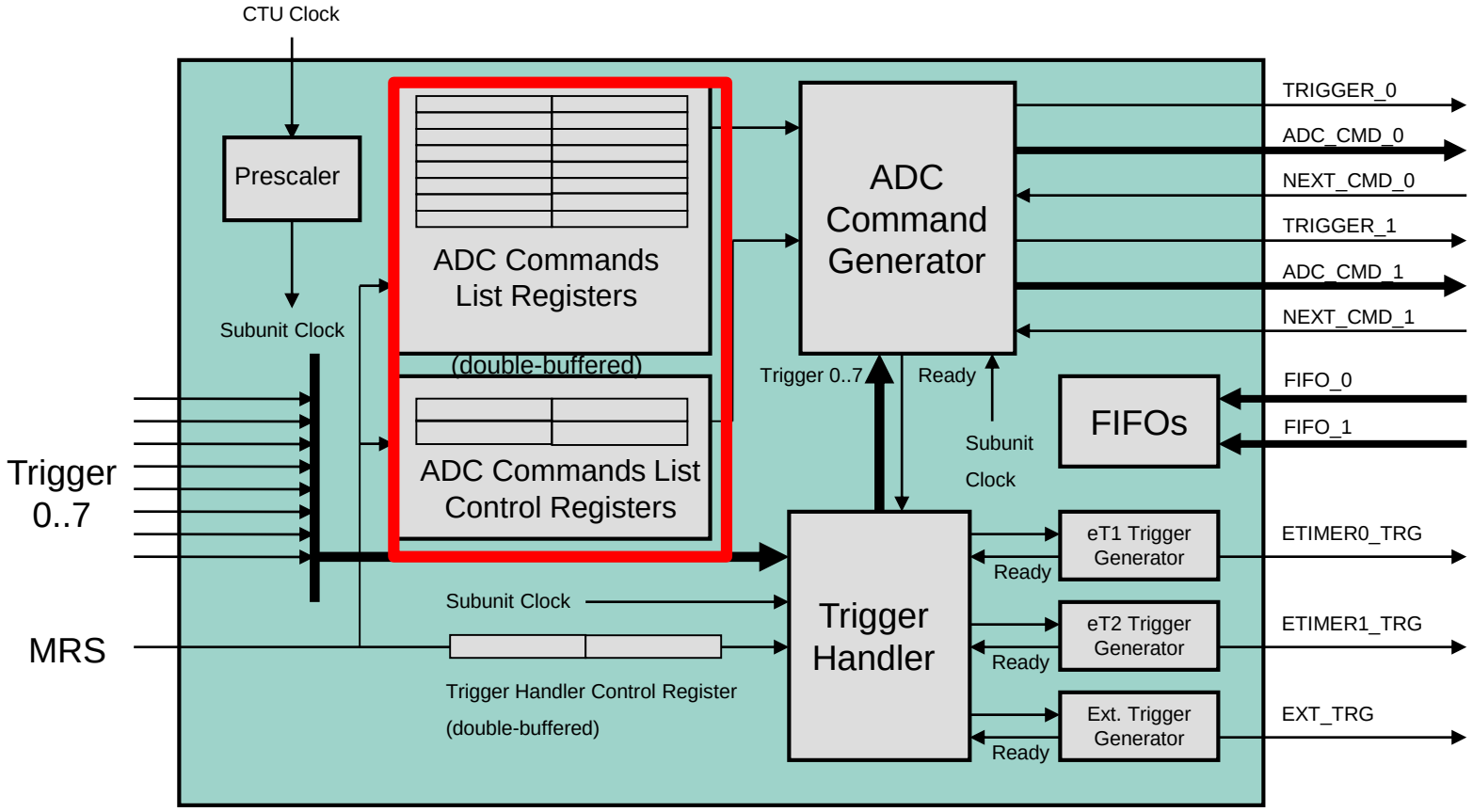
TGSCCR: TGS Counter Compare Register

TGSCRR: TGS Counter Reload Register

Scheduler

- The Scheduler takes the Trigger Generator output and activates
  - An ADC command or ADC stream of commands
  - An eTimer1 pulse (ETIMER0_TRG internally connected to eTimer_0 AUX0)
  - An eTimer2 pulse (ETIMER1_TRG internally connected to eTimer_1 AUX0
  - An external trigger pulse

CTU Scheduler



CTU ADC Command Structure

Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_*n*)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CIR	LC	CMS	FIFO			ST0	0			SU	0	CH			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLR_A_*n* field descriptions

ADC Command List Register Format

Bit 0 = CIR : Command Interrupt Request

Bit 1 = LC : Last Command

Bit 2 = CMS : Conversion Mode Select

Bit 4:5 = FIFO : FIFO Select 0-3

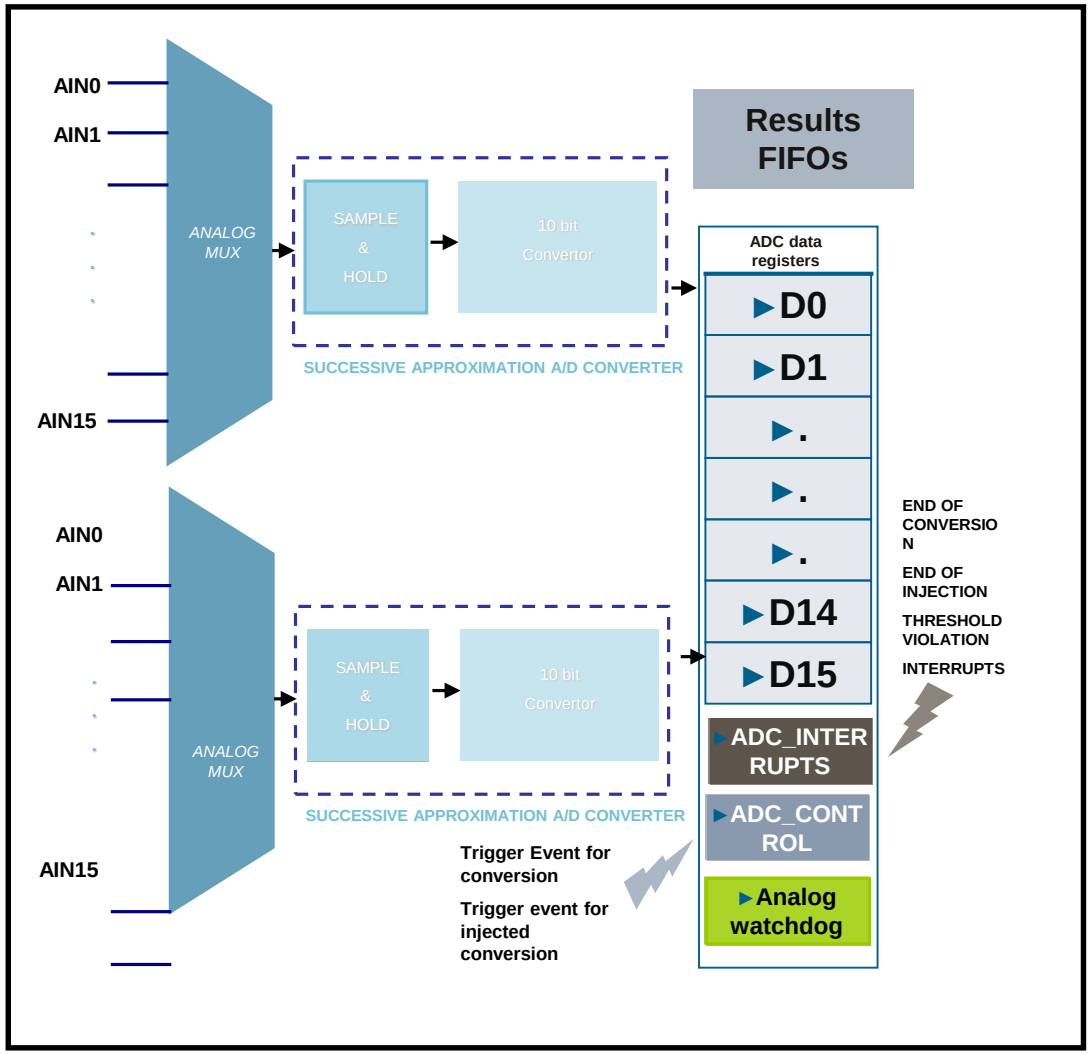
Bit 6 = ST : Self-Test Bit (0: Normal Mode; 1: Self-Test Mode)

Bit 10 = SU : Selection Unit - ADC_0 Or ADC_1

Bit 12:15 = CH : ADC Channel

ADC

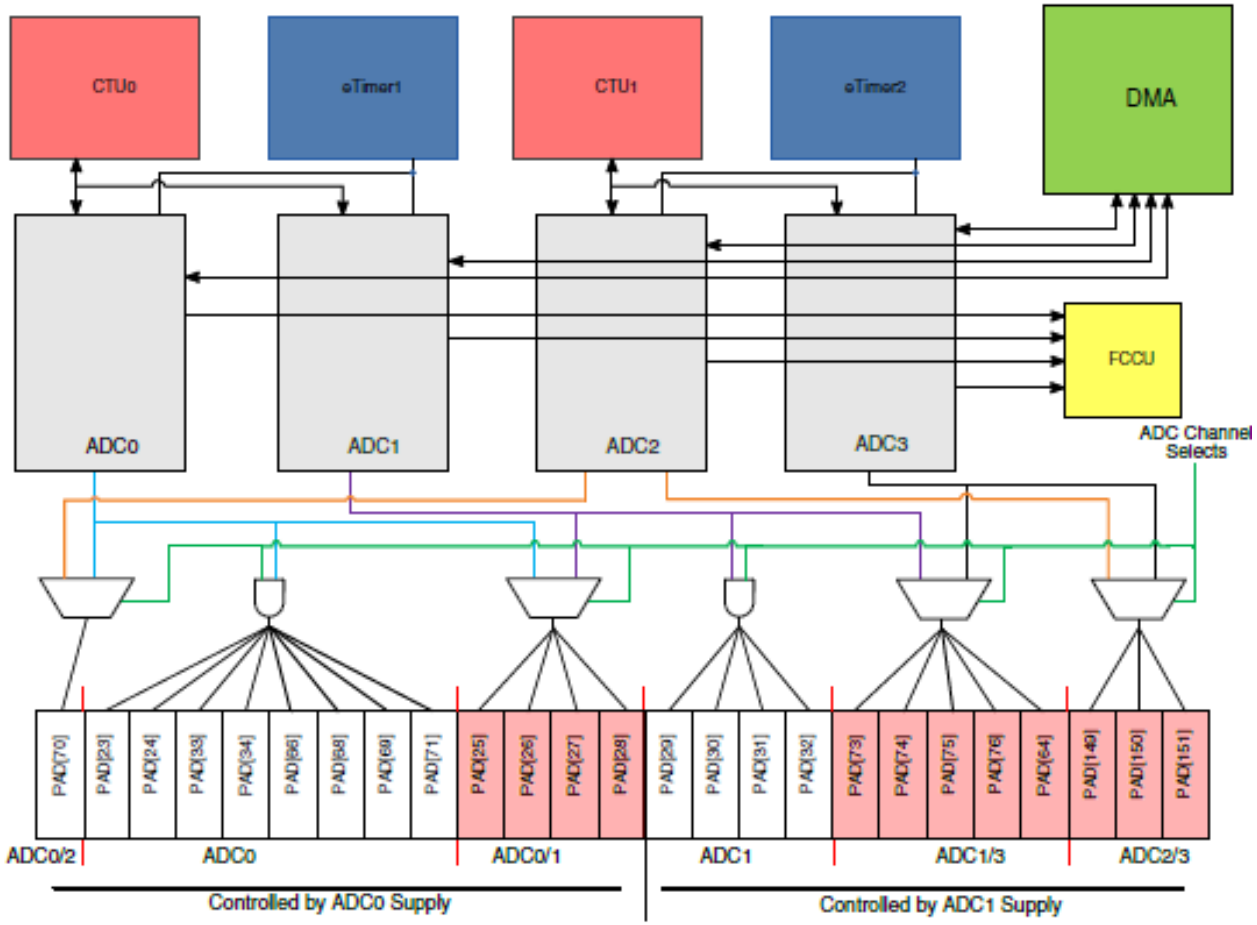
ADC



Main Features

- 4 Independent units
- <1us sampling + conversion time
- 12-bit resolution
- Analog watchdogs allow continuous hardware monitoring.
- Motor control mode
  - ADC can only be started using ADCcommand
  - Results are sent to result FIFOs to support signal oversampling
    - DMA support for each FIFO via thresholds
  - Results can be read in different modes
    - 16bit / 32bit with channel information
    - Left aligned or right aligned
- Normal mode
  - Regular conversion commands (single or multiple), software injected higher priority commands, hardware injected conversion commands
  - On going conversion is cancelled and resumed after high priority command is processed.

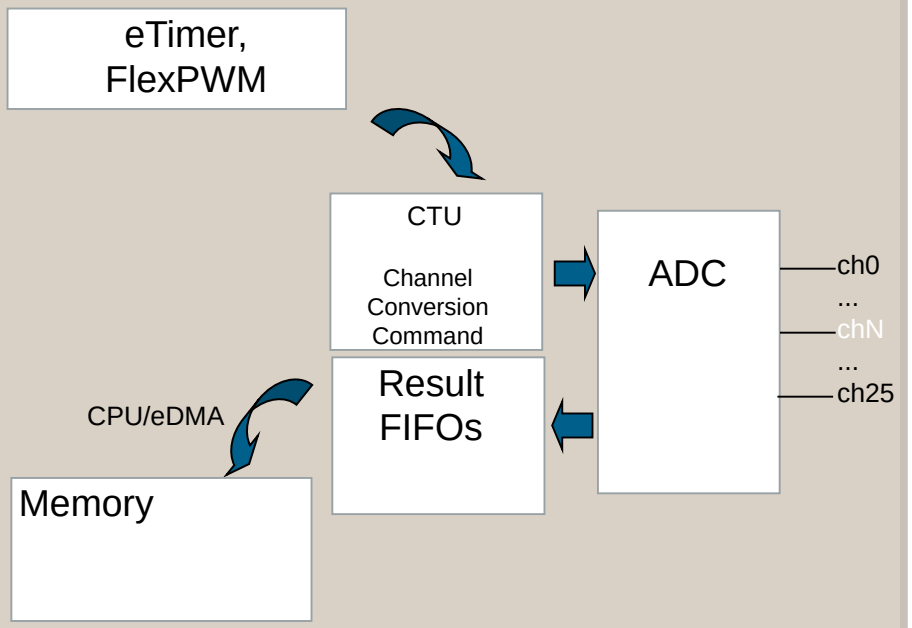
Block Diagram



ADC : conversion modes

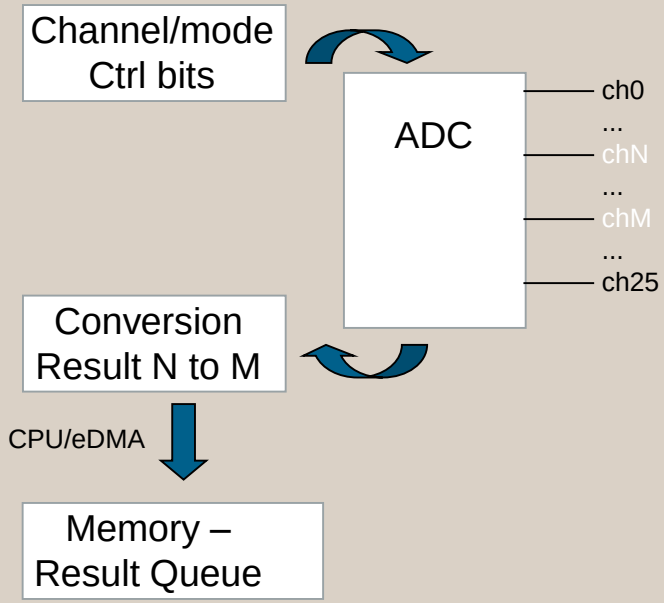
Motor control mode

Conversions only triggered by CTU



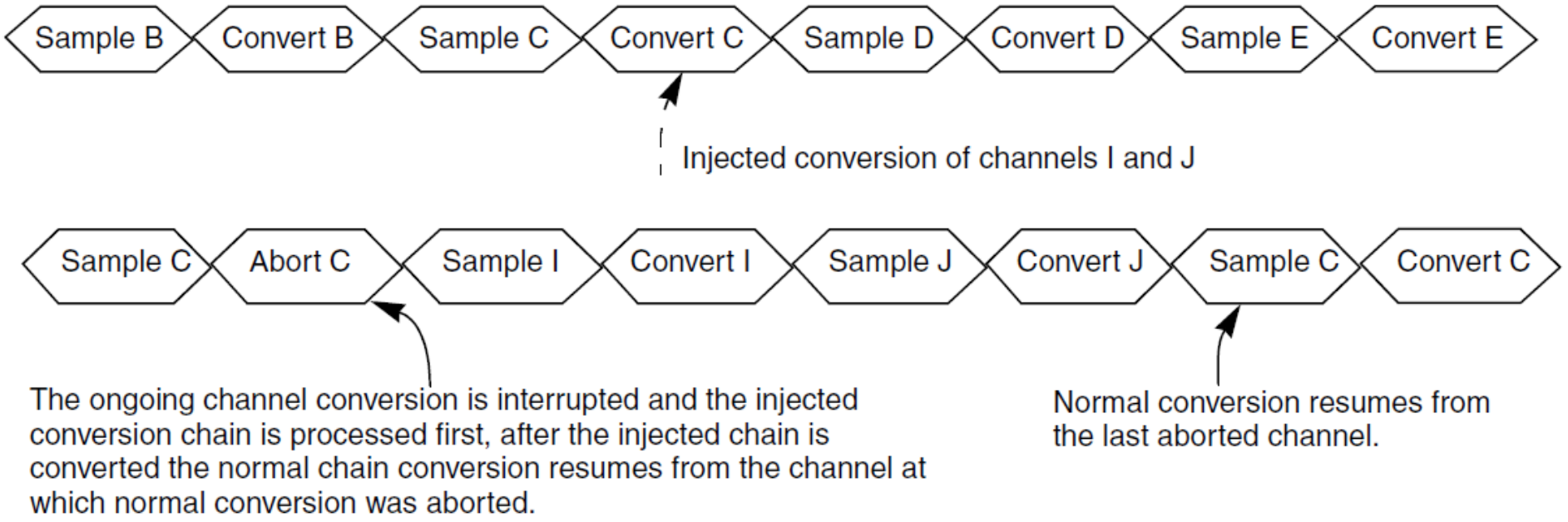
Regular mode

Conversions triggered by CPU or by hardware



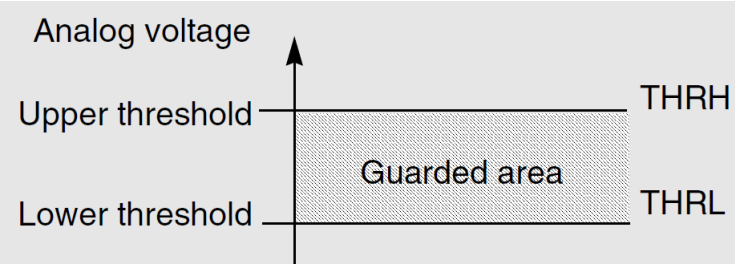
Injected channel conversion

- A conversion chain can be injected into the ongoing Normal conversion by configuring the Injected Conversion Mask Registers (JCMR).



Programmable Analog Watchdog

- The analog watchdogs are used for determining whether the result of a channel conversion lies within a given guarded area.
- After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values.

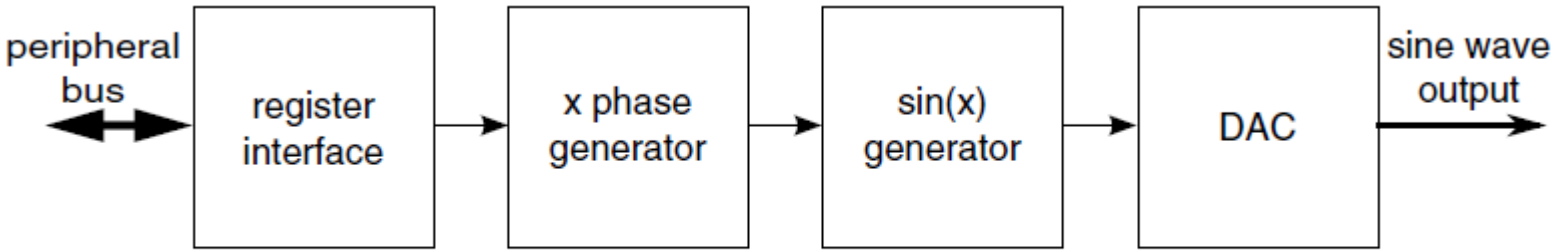


WDGxH	WDGxL	Converted data
1	0	Converted data > THRH
0	1	Converted data < THRL
0	0	THRL ≤ Converted data ≤ THRH

SGEN
(Sine Wave Generator)

Feature

- Input clock frequency range: 12 MHz–20 MHz
  - Output sinusoidal signal:
  - Frequency range: 1 kHz–50 kHz
  - Peak-to-peak amplitude: user adjustable

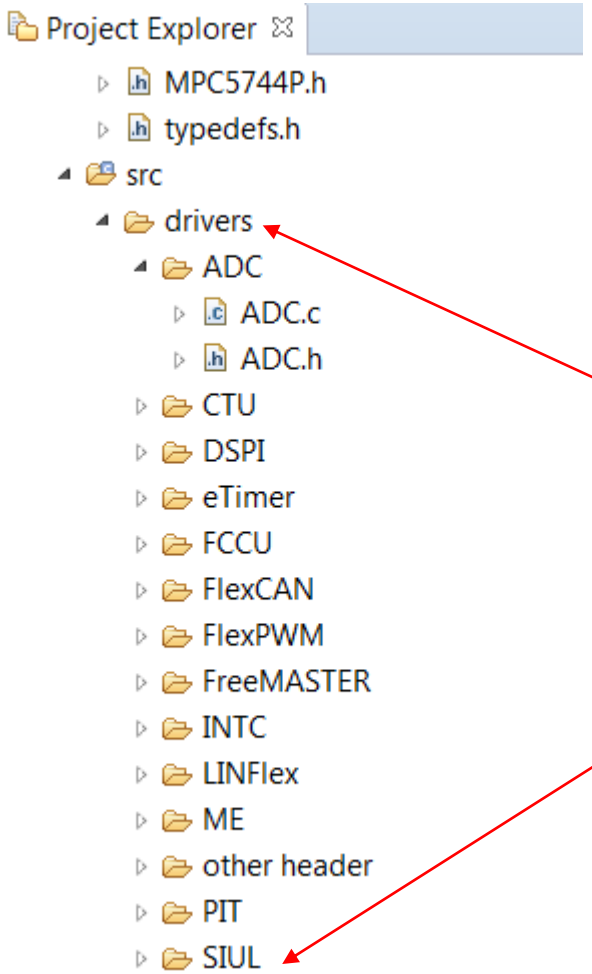


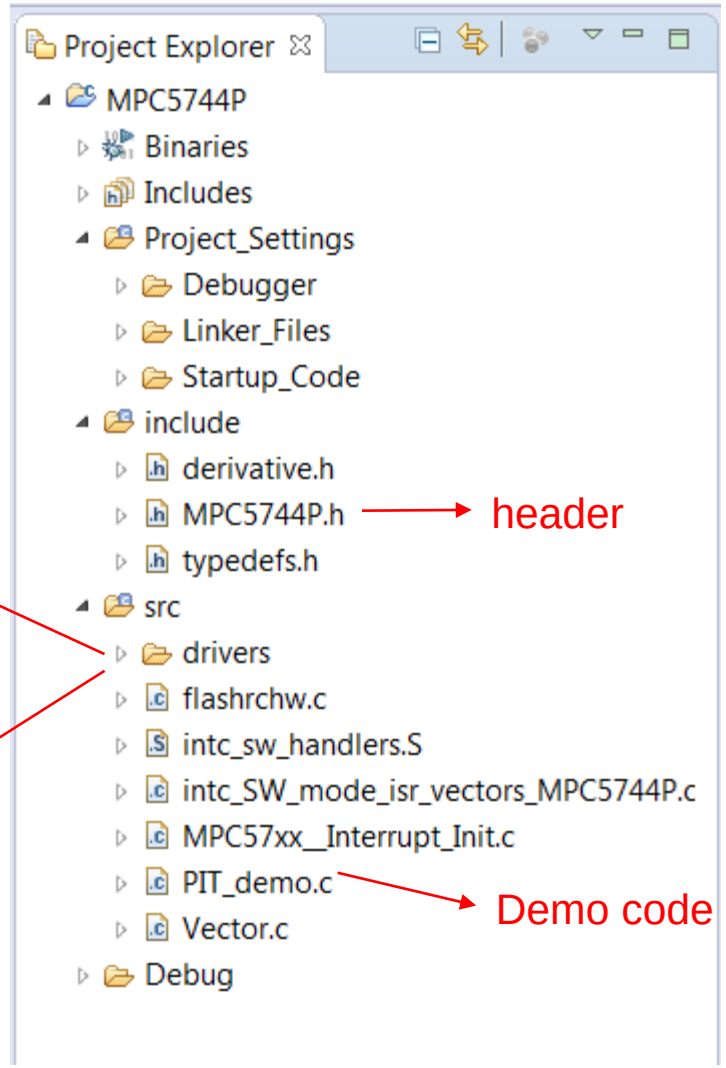
SGEN block diagram

LOW-LEVEL DRIVER

Software Architecture (based on S32DS For Power1.0)

- **Directory structure**





Software Architecture

- **startup code**
  - startup.S
- **Linker Files**
  - mem.ld
  - section.ld
  - libs.ld
- **Boot header and boot options**
  - flashrchw.c
- **Interrupt vector**
- - init_SW_mode_isr_vectors_MPC5744P.c

- **Driver**
  - (xx.c)
  - (xx.h)

  like (PIT.c, PIT.h)

- **Demo code**
  - xx_demo.c

  like PIT_demo.c

Startup Code and Initial Code

- **Startup Code (from reset to main)**

  Refer to the document “AN4670_Initializing_the_MPC5746R”

- **Initial code (after clock configuration in main)**

  1. Set IVPR
  2. Init IVOR (Interrupt vector table base address)
  3. Initialize INTC for SW vector mode
  4. Initialize priority if interrupt is used
  5. Enable interrupt if needed.

- **Header File**

  MPC5744P.h

Demo code and Driver Available

- **Demo code**
  - ✓ PIT
  - ✓ SIUL_GPIO
  - ✓ FlexCAN (interrupt and interrupt)
  - ✓ ADC (non-interrupt and interrupt)
  - ✓ FlexPWM
  - ✓ LINFlexD (support UART)
  - ✓ SPI
  - ✓ eTimer
  - ✓ SWG
  - ✓ FlexPWM-CTU-ADC
  - ✓ FreeMASTER (UART/CAN communication)
  - ✓ Mass Modules (include all above driver)

- **Driver**
  - ✓ PIT
  - ✓ ME
  - ✓ INTC
  - ✓ FCCU
  - ✓ SIUL
  - ✓ FlexCAN
  - ✓ ADC
  - ✓ FlexPWM
  - ✓ CTU
  - ✓ LINFlexD
  - ✓ UART
  - ✓ DSPI
  - ✓ eTimer
  - ✓ SWG
  - ✓ FreeMASTER

FUNCTION SAFETY ACTIVITIES AT NXP

Enablement to simplify design for function safety

- Safety manual 安全手册
- FMEA (Failure Modes Effects and Diagnostic Analysis)
  失效模式、影响及其诊断分析
- Application note for system integrity

Safety Manual

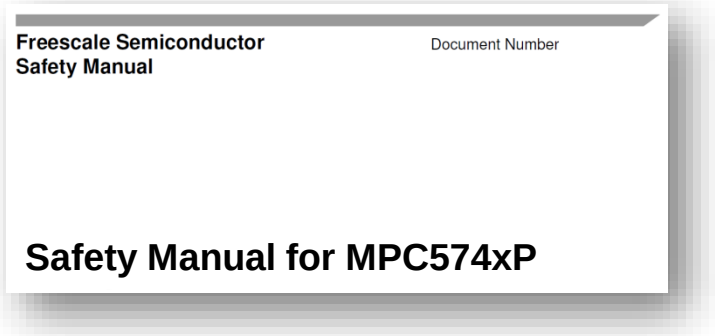
Objective

- Enables customers to build their safety system using the MCU safety mechanisms and defines system level HW & SW assumptions
- Simplify integration of NXP's safety products into applications
- A comprehensible description of all information relating to FS in a single entity to ensure integrity of information

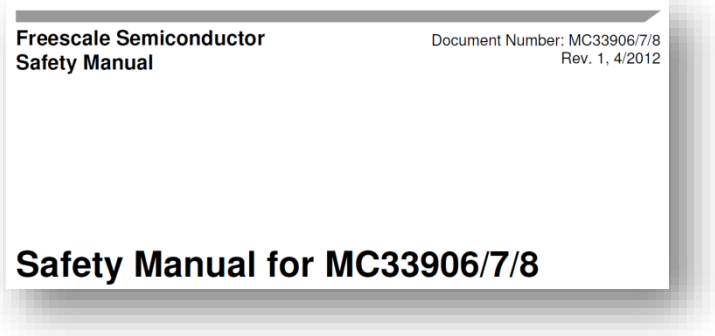
Content

- MCU Safety Context
- MCU Safety Concept
- System level hardware assumptions
- System level software assumptions
- FMEDA summary
- Dependent Failures Analysis summary

Safety Manual for MCU Solution

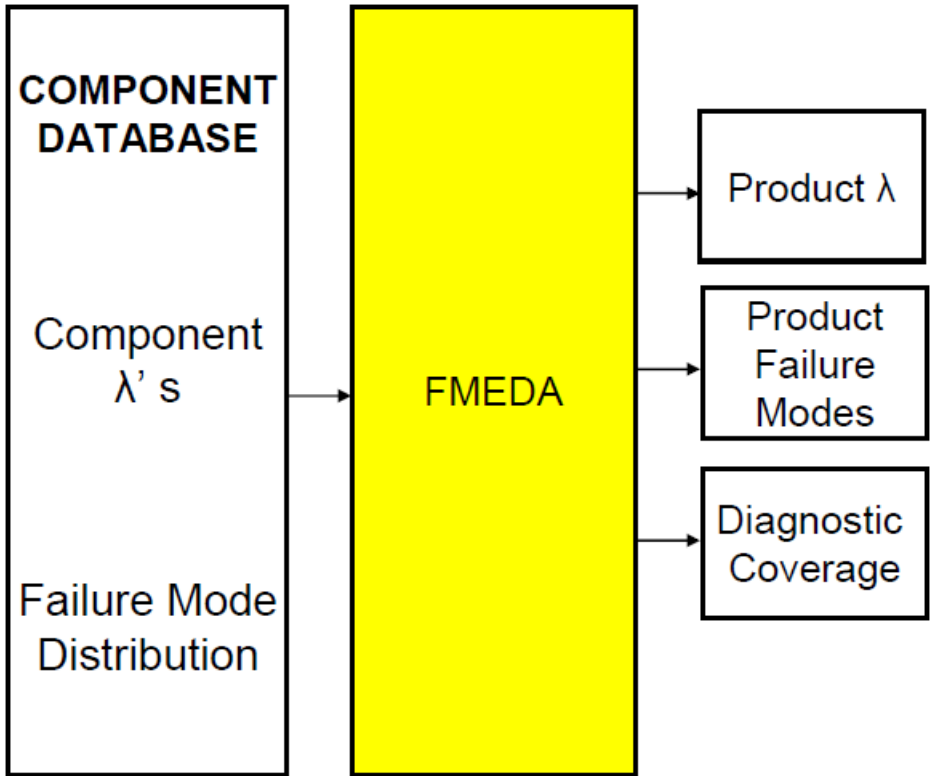


Safety Manual for Analog Solution

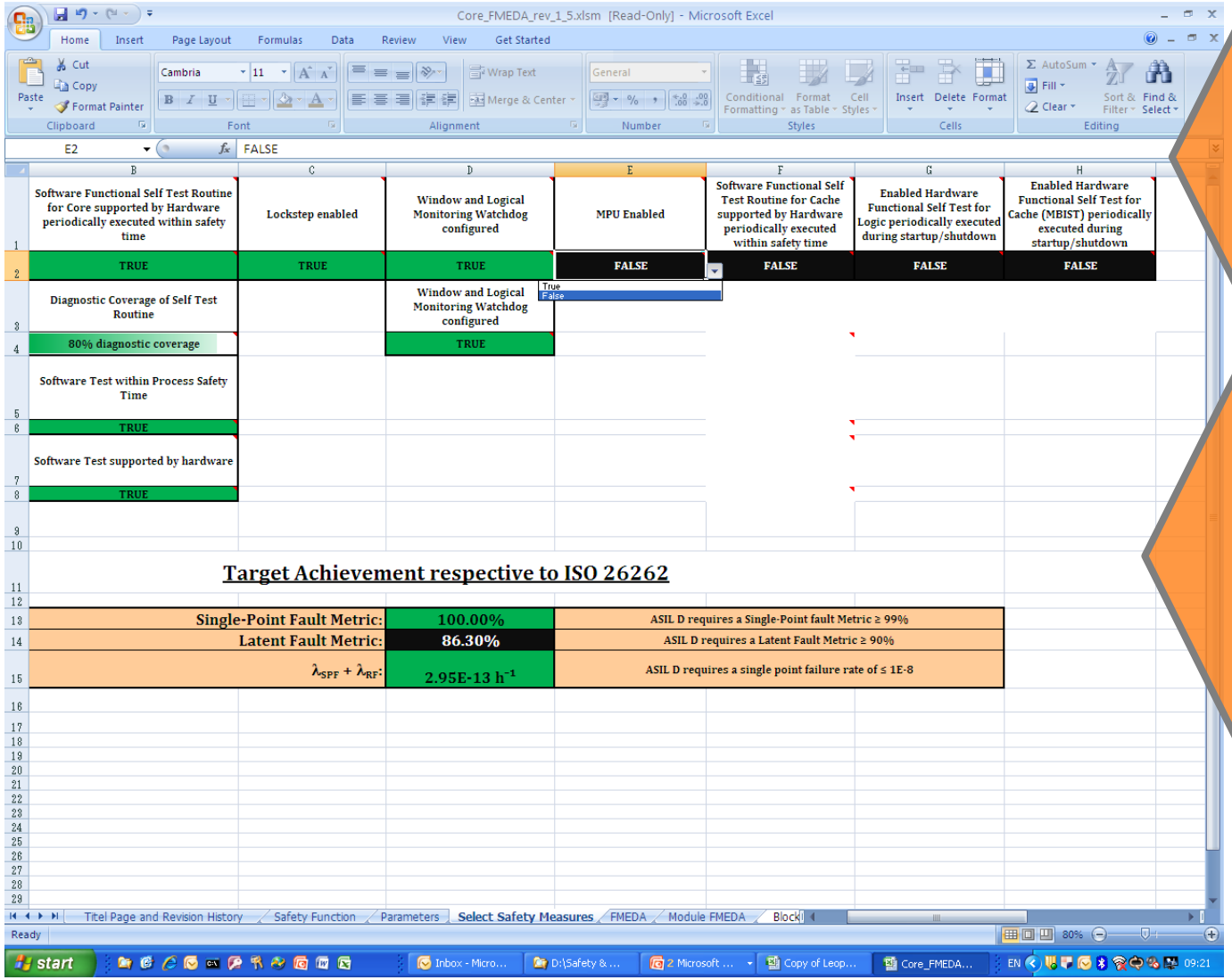


FMEDA Inputs and Outputs

- **The FMEDA technique considers:**
  - All components of a design
  - The functionality of each component
  - The failure modes of each component
  - The impact of each component failure mode on the product functionality,
  - The ability of any automatic diagnostics to detect the failure



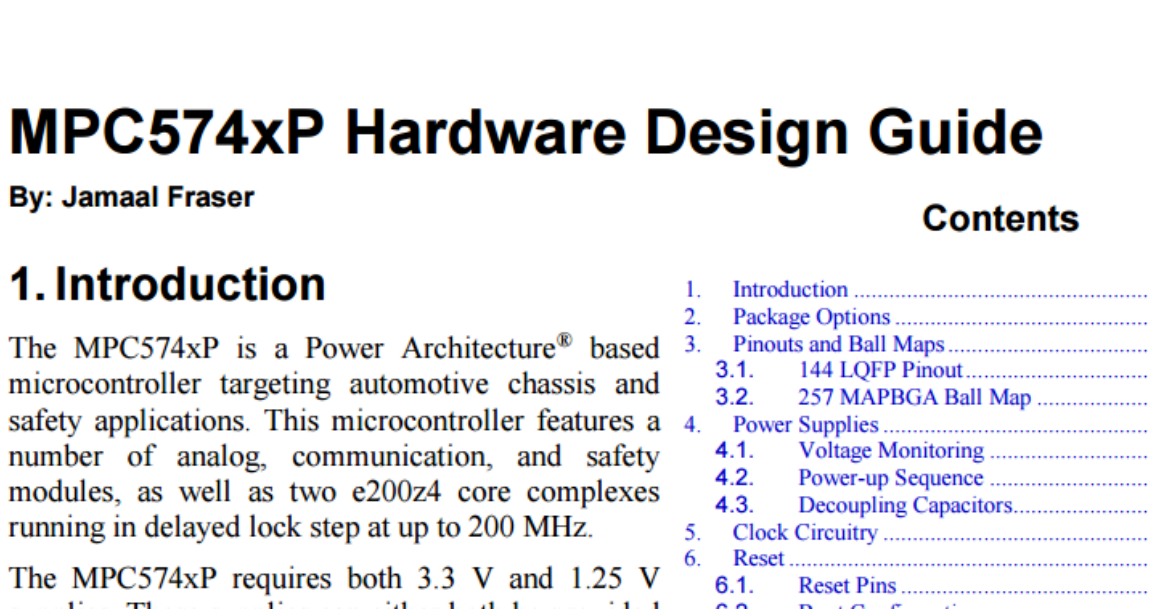
SafeAssure Deliverables - FMEDA



One of the key deliverables customers require is a FMEDA.

NXP has developed a new FMEDA tool that allows customers to tailor for their application cases. This is exactly what SafeAssure stands for – simplifying functional safety.

- # Application note for system integrity



MPC574xP Hardware Design Guide

By: Jamaal Fraser

Contents

1. Introduction
2. Package Options
3. Pinouts and Ball Maps
   - 3.1. 144 LQFP Pinout
   - 3.2. 257 MAPBGA Ball Map
4. Power Supplies
   - 4.1. Voltage Monitoring
   - 4.2. Power-up Sequence
   - 4.3. Decoupling Capacitors
5. Clock Circuitry
6. Reset
   - 6.1. Reset Pins

1. Introduction

The MPC574xP is a Power Architecture® based microcontroller targeting automotive chassis and safety applications. This microcontroller features a number of analog, communication, and safety modules, as well as two e200z4 core complexes running in delayed lock step at up to 200 MHz.

The MPC574xP requires both 3.3 V and 1.25 V

Freescale Semiconductor, Inc. Document Number: AN5099

应用笔记 Rev. 0, 04/2015

面向安全应用的MPC5744P和MC33907/08集成

作者：Tomas Kulig

内容

1. 简介
2. MPC5744P 概述
   - 2.1. 安全概念
   - 2.2. 电源要求
   - 2.3. 通信接口
   - 2.4. 故障收集和控制单元 (FCCU)
3. MC33907/08 特性
   - 3.1. 稳压器
   - 3.2. 内置 CAN 收发器
   - 3.3. 内置 LIN 收发器
   - 3.4. 看门狗功能

1. 简介

本应用笔记为Freescale MPC5744P微控制器单元(MCU)与Freescale MC33907/08系统基础芯片的集成提供设计指南，可用于汽车电气与电子系统中，从而满足ISO 26262功能安全标准。本应用笔记概述了MPC5744P和MC33907/08的功能集，并且涵盖了达到ASIL D级安全性的功能安全要求。在同一个系统

More documents and materials, please click this link: link to website for MPC5744P

Further Information

- For more in-depth training, visit A&M Tech’s MPC5744P Training Course
- Contact Munir Bannoura, at Munir@bannoura.com

SECURE CONNECTIONS
FOR A SMARTER WORLD