Low-Power Modes for MPC5xxx and Beyond

Steve Mihalik

Senior Field Application Engineer

October 2018 | AMF-AUT-T3381

SECURE CONNECTIONS FOR A SMARTER WORLD

Agenda

- Mode Entry Module Introduction
  - Use Case
  - Low Power Modes
- HW Implications:
  - Terminations
  - Pad Keeping
- SW Implications:
  - Configuration
  - Entering low power
  - Exiting low power
  - MCAL
- Debugging and When Things Go Wrong

Abstract

Achieving the lowest MCU power during sleep presents challenges to HW and SW designers from switching off clocks and power to modules.

Low power use cases and concepts for NXP’s Mode Entry module are presented along with implications to hardware design, software design and debugging.

Mode Entry Module Introduction

Use Case: Conserving Power While Software Runs (1 of 2)

- Software ***runs*** one of the three following tasks:
  - **Analog Monitor**
    - Uses ADC to look for particular voltages on inputs
    - Only requires 16 MHz FIRC, which is also sysclk
    - If analog input measurements meet a criteria, software transitions to the communication task
  - **Communication**
    - Uses FlexCAN_0, FlexCAN_1 to transmit analog data and receive response
    - Only requires FXOSC, which is also sysclk
    - If response is positive, software transitions to the whole chip task
  - **Whole Chip**
    - Requires all peripherals active
    - Sysclk = maximum frequency FMPLL, which also requires FXOSC

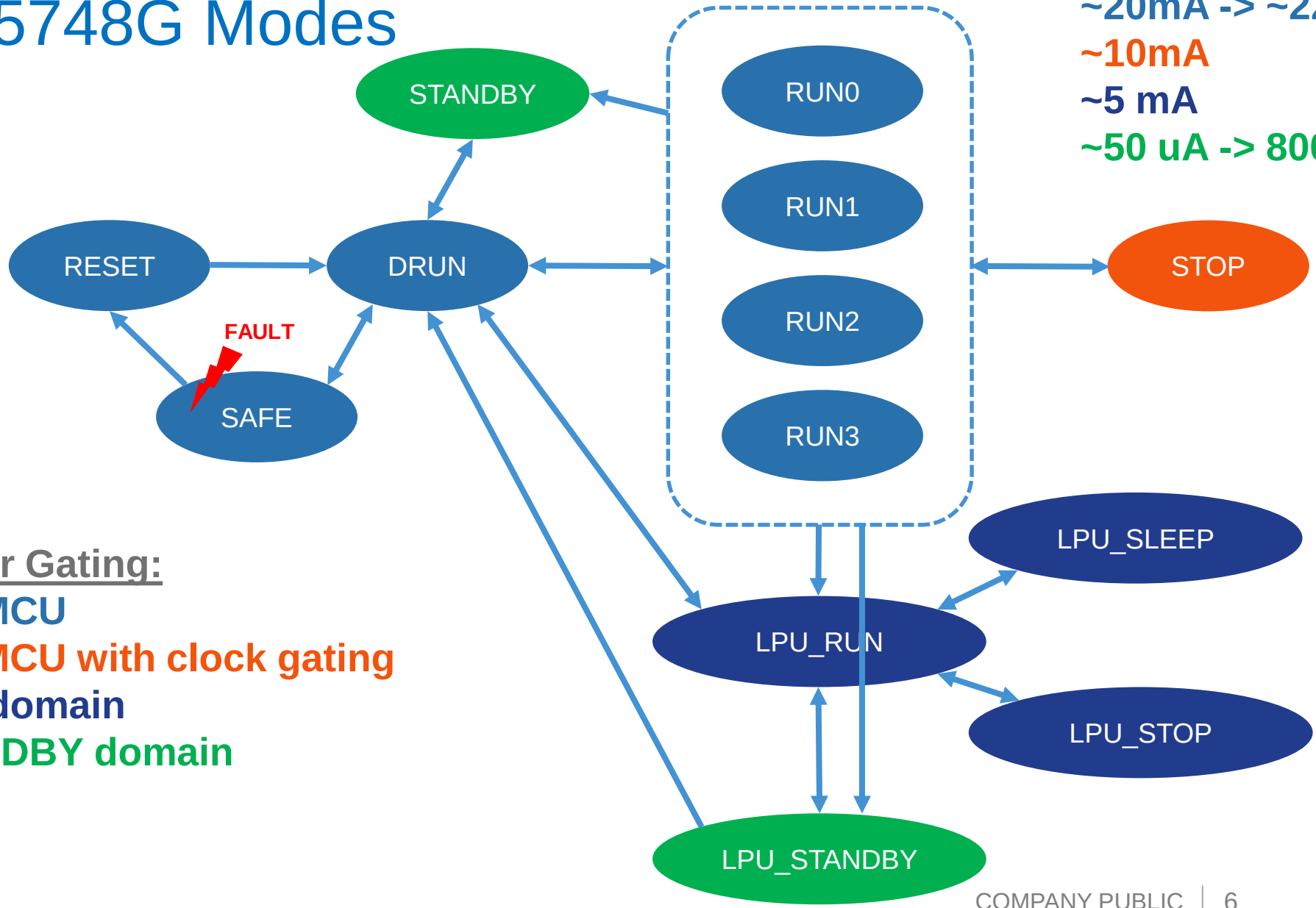
Use Case: Conserving Power While Software Runs (2 of 2)

Mode	sysclk	Clock Sources Req'd			Peripherals Requiring Enabled Clock
		FIRC	XOSC	FMPLL	
	← *Mode Configuration* →				← *Peripheral Configuration* →
RUN0: AnalogMon	FIRC	X			ADC, SIU
RUN1: Comm	XOSC		X		FlexCAN_0, FlexCAN_1, SIU
RUN2: WholeChip	FMPLL		X	X	All

Peripherals	Modes with Enabled Clock				Peripheral Configuration for RUN modes	Peripheral's Control Register (Specifies Peri. Config. to be used)
	RUN0	RUN1	RUN2	RUN3		
ADC	X		X		Run PC0	PCTL[32]
FlexCAN_0, 1		X	X		Run PC1	PCTL[16], PCTL[17]
SIU	X	X	X		Run PC2	PCTL[68]
All others			X		Run PC3	Other PCTL's

MPC5748G Modes

Idd typical@25C
~20mA -> ~220mA
~10mA
~5 mA
~50 uA -> 800uA



Power Gating:
Full MCU
Full MCU with clock gating
LPU domain
STANDBY domain

Low Power Modes Summary (MPC5748G)

- **STOP** (11mA)
  - In STOP mode the MCU remains completely powered
  - Exit STOP mode on interrupt or wake-up
  - Exiting STOP mode the MCU context is retained so no need to reconfigure core or peripherals

- **STANDBY & LPU_STANDBY** (71uA)
  - In STANDBY mode 99% of the MCU is powered-down
  - Upon wake-up the MCU context is lost (except STANDBY peripherals)

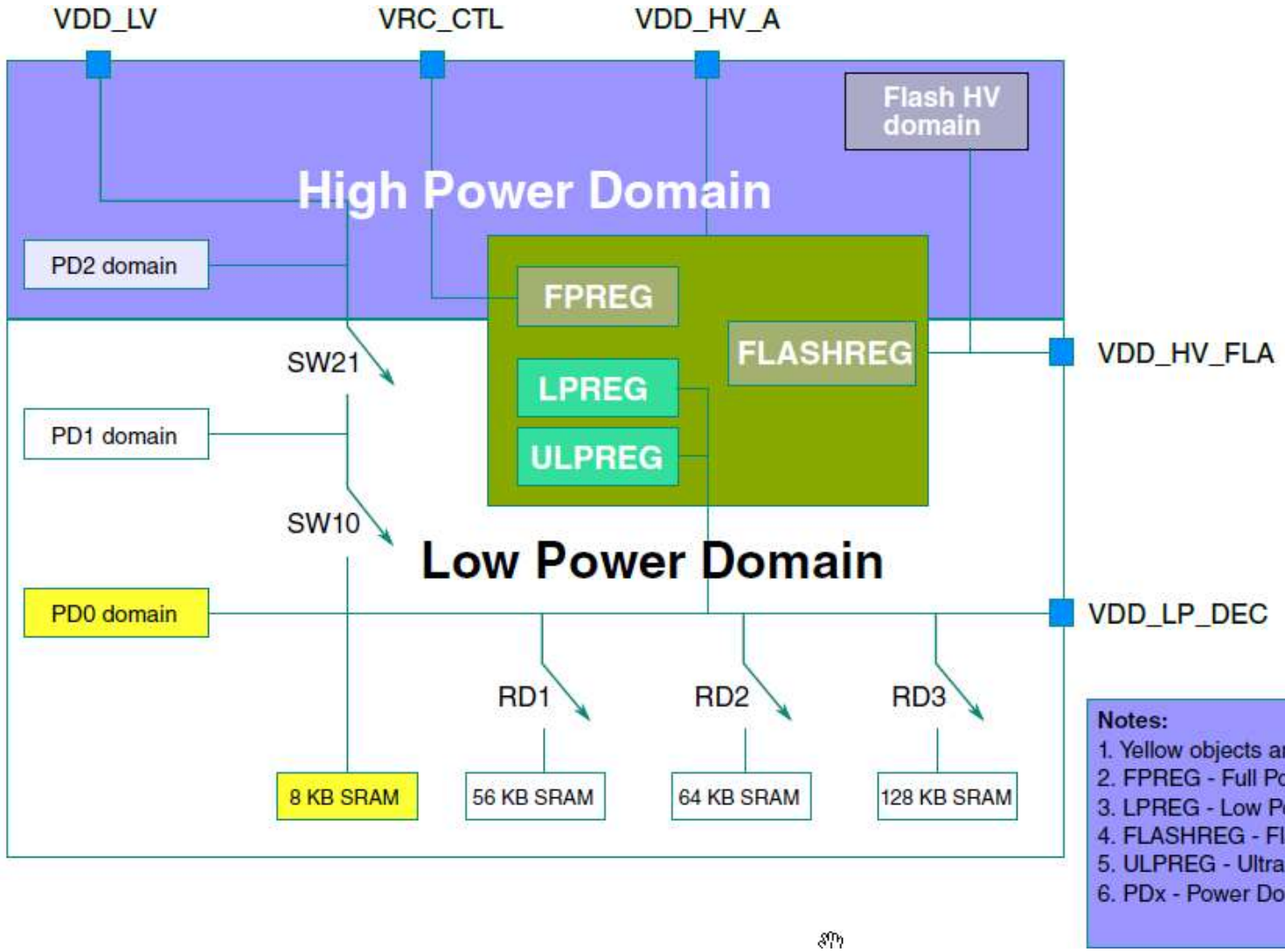
- **Wake-up source options for STOP and STANDBY modes**
  - 30x input pins
  - Analog comparator
  - Timers
  - Non-Maskable Interrupt
  - Interrupts (STOP mode only!)

Power block diagram

High performance blocks

LPU domain

Always on



MPC574XG 6MB and 3MB STANDBY Current

MPC5746C Data Sheet, Rev 5.1, 05/2017

Table 12. STANDBY Current consumption characteristics

Symbol	Parameter	Conditions[1]	Min	Typ	Max	Unit	Min	Typ	Max	Unit
STANDBY0	STANDBY with 8K RAM	T_a = 25 °C	—	71	—	µA	—	71	—	µA
		T_a = 85 °C	—	175	800		—	125	700	
		T_a = 105 °C	—	338	1725		—	195	1225	
		T_a = 125 °C	—	750	2775		—	314	2100	
STANDBY1	STANDBY with 64K RAM	T_a = 25 °C	—	72	—	µA	—	72	—	µA
		T_a = 85 °C	—	176	815		—	140	715	
		T_a = 105 °C	—	350	1775		—	225	1275	
		T_a = 125 °C	—	825	3000		—	358	2250	
STANDBY2	STANDBY with 128K RAM	T_a = 25 °C	—	75	—	µA	—	75	—	µA
		T_a = 85 °C	—	182	830		—	155	730	
		T_a = 105 °C	—	366	1825		—	255	1350	
		T_a = 150 °C	—	900	3250		—	396	2600	
STANDBY3	STANDBY with 256K RAM	T_a = 25 °C	—	80	—	µA	—	80	—	µA
		T_a = 85 °C	—	197	860		—	180	800	
		T_a = 105 °C	—	400	1875		—	290	1425	
		T_a = 125 °C	—	975	3500		—	465	2900	
STANDBY3	FIRC ON	T_a = 25 °C	—	500	—	µA	—	500	—	µA

Hardware Implications

Interrupt / wakeup – Signal identification

- See IO Signal Table in spreadsheet attached to MPC5748G Reference Manual
- WKPU[x] corresponds to wakeup / interrupt source (also called channel) in WKPU registers:
  - Status (WISR)
  - Interrupt Req. Enable (IRER)
  - Wakeup Req. Enable (WRER)
  - Rising Edge Enable (WIREER)
  - Falling Edge Enable (WIFEER)
  - Filter Enable (WIFER)
  - Pull Enable (WIPER)

6	Port	LVDS Pair Port	SIUL MSCR#	MSCR SSS	Function	Modul	D
7	PA[0]		0	0000_0000	GPIO[0]	SIUL2	
8	PA[0]			0000_0001	E0UC_0_X	EMIOS0	
9	PA[0]			0000_0010	CLKOUT0	CGM	
10	PA[0]			0000_0011	E0UC_13_H	EMIOS0	
11	PA[0]			-	WKPU[19]	WKPU	
12	PA[0]		512	0000_0010	E0UC_0_X	EMIOS0	
13	PA[0]		525	0000_0010	E0UC_13_H	EMIOS0	
14	PA[0]		701	0000_0001	CAN1RX	FlexCAN_1	
15	PA[1]		1	0000_0000	GPIO[1]	SIUL2	
16	PA[1]			0000_0001	E0UC_1_G	EMIOS0	
17	PA[1]			-	WKPU[2]	WKPU	
18	PA[1]			-	NMI[0]	WKPU	
19	PA[1]		513	0000_0010	E0UC_1_G	EMIOS0	
20	PA[1]		703	0000_0001	CAN3RX	FlexCAN_3	
21	PA[2]		2	0000_0000	GPIO[2]	SIUL2	
22	PA[2]			0000_0001	E0UC_2_G	EMIOS0	
23	PA[2]			0000_0010	E2UC_0_X	EMIOS2	
24	PA[2]			0000_0011	ADC0_MA[2]	ADC_0	
25	PA[2]			-	WKPU[3]	WKPU	
26	PA[2]		514	0000_0010	E0UC_2_G	EMIOS0	

Interrupt / wakeup - pads list

- List is filtered from MPC5748G IO Signals spreadsheet attached to reference manual

Port	Function	Module
PA[0]	WKPU[19]	WKPU
PA[1]	WKPU[2]	WKPU
PA[2]	WKPU[3]	WKPU
PA[4]	WKPU[9]	WKPU
PA[15]	WKPU[10]	WKPU
PB[1]	WKPU[4]	WKPU
PB[3]	WKPU[11]	WKPU
PB[8]	WKPU[25]	WKPU
PB[9]	WKPU[26]	WKPU
PB[10]	WKPU[8]	WKPU
PC[7]	WKPU[12]	WKPU
PC[9]	WKPU[13]	WKPU
PC[11]	WKPU[5]	WKPU
PD[0]	WKPU[27]	WKPU
PD[1]	WKPU[28]	WKPU
PE[0]	WKPU[6]	WKPU
PE[3]	WKPU[29]	WKPU
PE[5]	WKPU[30]	WKPU
PE[9]	WKPU[7]	WKPU
PE[11]	WKPU[14]	WKPU
PF[9]	WKPU[22]	WKPU
PF[11]	WKPU[15]	WKPU
PF[13]	WKPU[16]	WKPU
PG[3]	WKPU[17]	WKPU
PG[5]	WKPU[18]	WKPU
PG[7]	WKPU[20]	WKPU
PG[9]	WKPU[21]	WKPU
PI[1]	WKPU[24]	WKPU
PI[3]	WKPU[23]	WKPU
PJ[13]	WKPU[31]	WKPU

Wakeup Pin Terminations

- All wakeup pins, whether used in the application or not, must be terminated for STANDBY mode
- Terminations can be either
  - internal, configured by software
  - External, with hardware
- Do not have BOTH internal and external terminations with opposite directions.

MPC574xG Internal Pullups/Pulldowns (per MPC574xG Ref. Man. Rev6)

Has priority
Over WKPU

Table 24-2. Pull control of wakeup pads

Device Mode	SIUL2_MSCRn[PUE]	SIUL2_MSCRn[PUS]	WKPU_WIPUER[IPUE]	Affect on WKPU PAD
All except STANDBY modes	1	0	x	Pull down
	1	1	x	Pull up
	0	x	0	Highz
	0	0	1	Pull down
	0	1	1	Pull up

Determines STANDBY
pull up or down state

Table 24-3. Pull control of wakeup pads

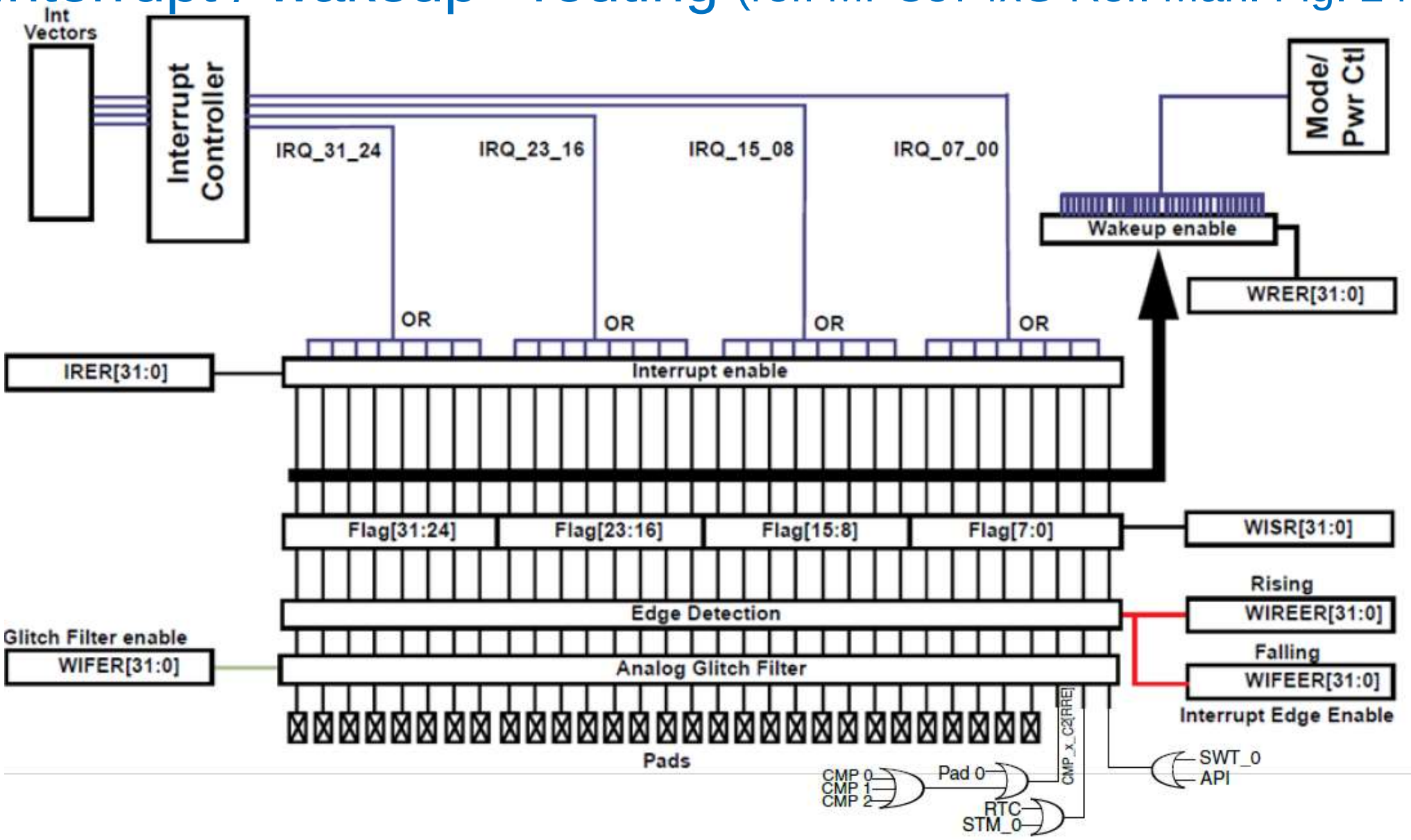
Device Mode	State of SIUL2_MSCRn[PUS] before entering Standby	WKPU_WIPER[IPUE]	Affect on WKPU PAD
STANDBY modes/ LPU_STANDBY mode	x	0	Highz
	0	1	Pull down
	1	1	Pull up

Pad keeping feature in LPU/STANDBY mode

(Reference: MPC5748G Ref. Manual section 4.2)

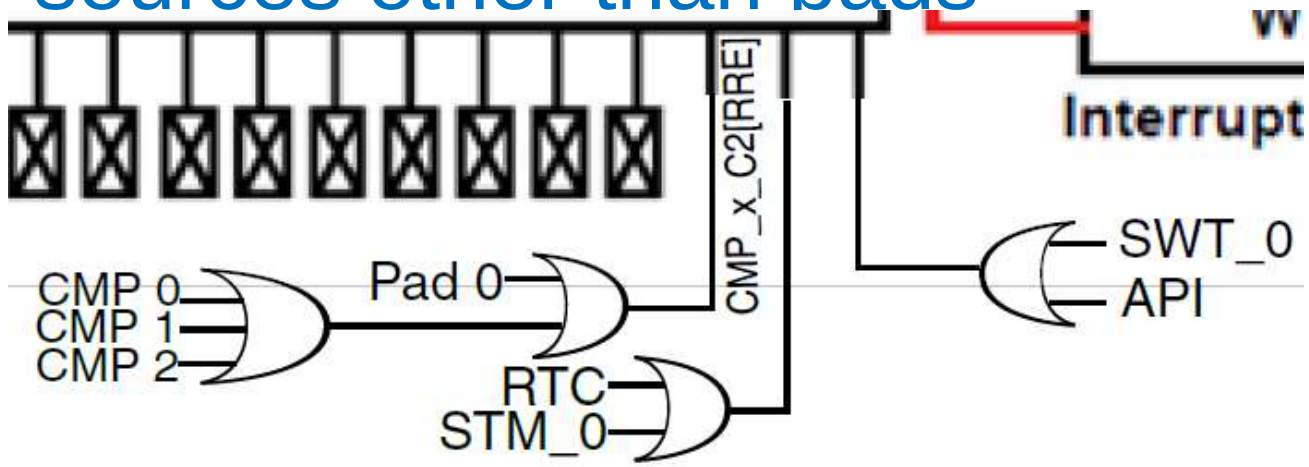
- The device is capable to retain the output value of some selected pads when device is power gated i.e. in LPU or STANDBY mode.
  - These pads reside in always alive power domain(PD0).
  - Table 4-1 in the reference manual lists which pads support PAD Keeping feature.
- To enable this feature, set PMCDIG_RDCR[PAD_KEEP_EN] to '1'.
  - **NOTE**: Upon LPU or STANDBY mode exit, you must clear this bit or else the pads output values will be retained!
- **Debugger notes**
  - By default, debugger connection will break after STANDBY exit if Pad Keeper functionality is enabled in STANDBY modes. (per reference manual)
  - For a debugger to maintain connection through STANDBY, PASS_LCSTAT[CNS] must be cleared by programming the Censorship DCF client. For example, DCF record 0x000055AA, 0x001000B0

Interrupt / wakeup - routing (ref: MPC574xG Ref. Man. Fig. 24-2)



Interrupt / wakeup – sources other than pads

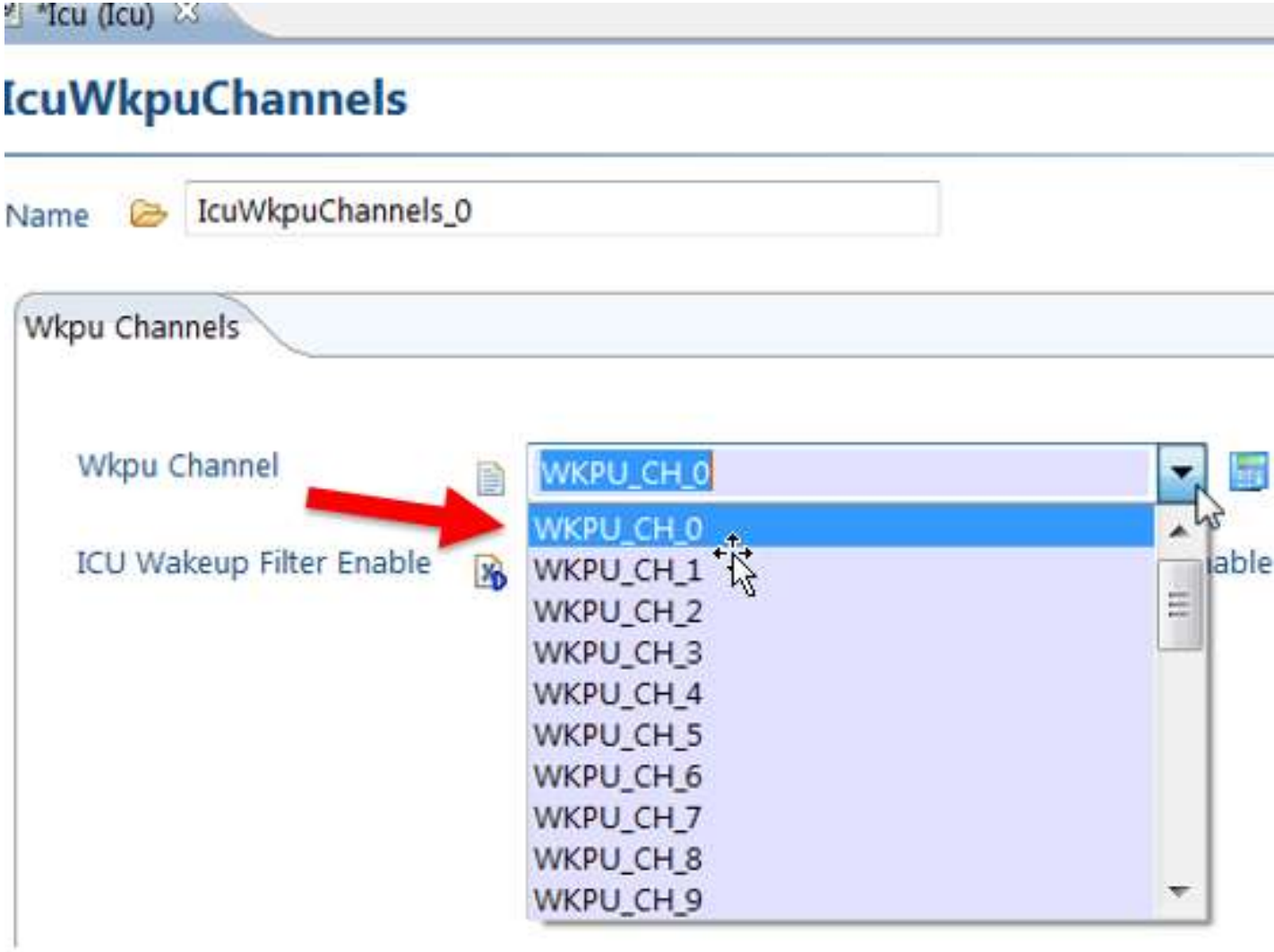
- MPC5748G channels 0, 1 and 2 have multiple options



Channel	Source	MCAL 4.2 STOP/STANDBY Support
0	Software Watchdog Timeout 0 (SWT_0) RTC Autonomous Periodic Interrupt (API)	Y Y
1	RTC Real Time Counter interrupt (RTC) System Timer Module 0 (STM_0)	N (RTC not supported by GPT driver) Y
2	Pad 0 ADC comparators 0, 1, 2 (CMP 0,1,2)	Y N (CMP not supported by ADC driver)

MCAL Wakeup Unit Channel Selection

- Channels are in the Input capture unit (Icu) module



Software Implications

Startup Code

- Determine why micro is starting:
  - Reset event - If a reset status flag is set (in Functional or Destructive Event Status Registers):
    - Normal initializations including wakeup logic such as wakeup unit, wakeup timers like RTC.
    - Normally clear all SRAM
    - Clear reset flags
  - Wakeup event (STANDBY mode low power exit) - If a wakeup unit status flag is set (in WISR register):
    - Normal initializations excluding wakeup logic such as wakeup unit, wakeup timers like RTC
      - Wakeup logic's power was maintained during STANDBY
    - Only clear SRAM sections not powered during STANDBY
    - Clear wakeup flags

Wakeup logic initialization

- Mode Entry module:
  - Configure STANDBY mode
  - Enable STANDBY mode
- Wakeup Module
  - Enable desired pads for wakeup
  - Configure whether pads wakeup on rising and/or falling edges
  - Configure desired timers
  - Configure desired ADC comparators

Entering STANDBY

- Ensure there are no pending interrupts or wakeup events

- Write to Mode Control register for entering STANDBY mode

```
MC_ME.MCTL.R = 0xD0005AF0;      /* Attempt STANDBY Mode (0xD) & Key */
MC_ME.MCTL.R = 0xD000A50F;      /* Attempt STANDBY Mode (0xD) & Inverted Key*/
```

- Wait for mode transition. (Suggest adding timeout logic)

```
while (MC_ME.GS.B.S_MTRANS) {}  /* Wait for mode transition to complete */
```

Wakeup to Flash or SRAM?

- Wakeup to flash advantages
  - Less complexity
- Wakeup to SRAM advantages
  - Less power
  - Less time
- Wakeup address is in MC_ME_CADDR1[ADDR]
  - To wakeup to SRAM, program desired SRAM start address into MC_ME_CADDR1
  - To wakeup to flash, leave original flash start address in register that was loaded at power on reset by BAF.

38.3.93 CORE1 Address Register (MC_ME_CADDR1)

This register contain the boot address of each Z4A and a bit for controlling whether the Z4A core is to be reset on the next mode change that has the core configured to be running in the target mode.

This register can be written only as words and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = '1'). A write access to any of this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

ME_CADDR1 is pre-loaded with the value provided by the device configuration just before reset exit in order to ensure that Z4A boots from the correct address after any chip reset if the DRUN bit in the ME_CCTL1 register has a default value of '1'.

The value of the ADDR field just before entering STANDBY0 mode is the value of the ADDR field on STANDBY0 exit.

Address: FFFB_8000h base + 1E4h offset = FFFB_81E4h

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

Low Power Wake up with Multicore Guidelines

- Before entering STANDBY mode, make sure the other cores will not automatically start on STANDBY exit.
  - Reason: software on other cores may depend on startup code completing by main core on STANDBY exit
  - Prevent other cores automatically starting on exit by disbling DRUN in ME_CCTL2, ME_CCTL3.
- After STANDBY wakeup make sure the prior wakeup to DRUN completed
  - Wait for ME_GS[S_MTRANS]
  - Reason: if you exit STANDBY to, for example, turn on the external oscillator and use it to start the PLL, then the clocks may not be stable for code that needs them.

Debugging and when things go wrong

DCF Record Required for MPC574xG Low Power Debug

- **PASS_LCSTAT [CNS] must be cleared** for low power debug mode to work across low power mode STANDBY transitions.
  - Only a DCF record in UTEST can clear CNS.
  - If CNS =1 (default per factory shipped) then no DCF record has been written to affect its value.

Mode Interrupts

- Mode transition issues can be enabled to generate an interrupt.
  - Example: MPC5748G Ref Manual v6, Mode Entry's Interrupt Status Register:

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
244	13D0	SIUL	EIRQ [8-15] - SIUL Combined External Interrupt
245	13D4	SIUL	EIRQ [16-23] - SIUL Combined External Interrupt
246	13D8	SIUL	EIRQ [24-31] - SIUL Combined External Interrupt
250	13E8	LPU_CTL	LPU_ICR[NEM] | LPU_ICR[MRIG]
251	13EC	MC_ME	ME_IS[I_SAFE]
252	13F0	MC_ME	ME_IS[I_MTC]
253	13F4	MC_ME	ME_IS[I_IMODE]
254	13F8	MC_ME	ME_IS[I_ICONF]
255	13FC	MC_RGM	MC_RGM Functional and destructive reset

26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) — This bit is set if a write access to one of the ME_CADDRn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
29 I_IMODE	Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt — This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes STOP0, or STANDBY0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

Invalid Mode Possible Causes

- Mode transitions are aborted when they are invalid.
- Example: Attempt to enter STANDBY mode but a wakeup event exists. IMTS[S_MRIG] is set

MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	Mode Request Ignored status — This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. It is cleared by writing a '1' to this bit. This bit is also set when a transition to STANDBY0/STOP0 mode is requested while a wakeup event is active. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
27 S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other non-SAFE mode transition process is active (S_MTRANS is '1'). Please refer to Mode Transition Interrupts for the exceptions to this behavior. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status — This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

Bare code debugging for incomplete mode transitions

- Incorporate a timer for mode transitions.
  - Timer expiration can be used to call error recovery function or simply trap for debug
- Examine Mode Entry status registers:
  - ME_GS
  - ME_IS
  - ME_IMTS
  - ME_DMTS

MCAL debugging for incomplete mode transitions

- C:\NXP\AUTOSAR\MPC574XG_MCAL4_2_RTM_1_0_0\eclipse\plugins\Mcu_TS_T2D35M10I0R0\src\Mcu_MC_ME.c file has the function Mcu_MC_ME_ApplyMode:

```
FUNC( void, MCU_CODE) Mcu_MC_ME_ApplyMode( VAR( Mcu_PowerModeType, AUTOMATIC) ePowerMode )
```

- In this function there is a timeout to wait for the mode transition to complete.

```
while (((uint32)MC_ME_GS_S_MTRANS_MASK32 == (REG_READ32(MC_ME_GS_ADDR32) &
(uint32)MC_ME_GS_S_MTRANS_MASK32)) && (u32Timeout > (uint32)0x0U))
    {
        /* (ME_GS[S_MTRANS]=1) <=> (Transition ongoing). */
        u32Timeout--;
    }
```

- Modify code so before timeout expires you branch somewhere and have a breakpoint there.

MCAL behavior on standby entry failure

- MCAL McuSetMode function implements a timeout on attempting mode entry
  - If a timeout occurs, a diagnostic error is reported to the diagnostic manager with dem_seteventstatus function.
- When dem_eventstatus function is called, the first parameter is “timeout from Mcu driver”.
- After that, the McuSetMode function returns and execution continues.

- Customers must implement handling this error with their own code.

MCAL LPU support

- NXP MCAL
  - Supports transition to LPU_RUN mode
    - User must write LPU application code and place in RAM
    - MCAL will not run in LPU_RUN mode. (e200z4 cores do not have power in LPU_RUN.)
  - Does not support LPU_STOP, LPU_SLEEP, LPU_STANDBY

- Rational: LPU_RUN mode is intended for small tasks to be performed while executing from RAM (when the AUTOSAR stack is not needed).
  - So the LPU_STOP, LPU_SLEEP and LPU-STANDBY are intentionally not supported.

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com