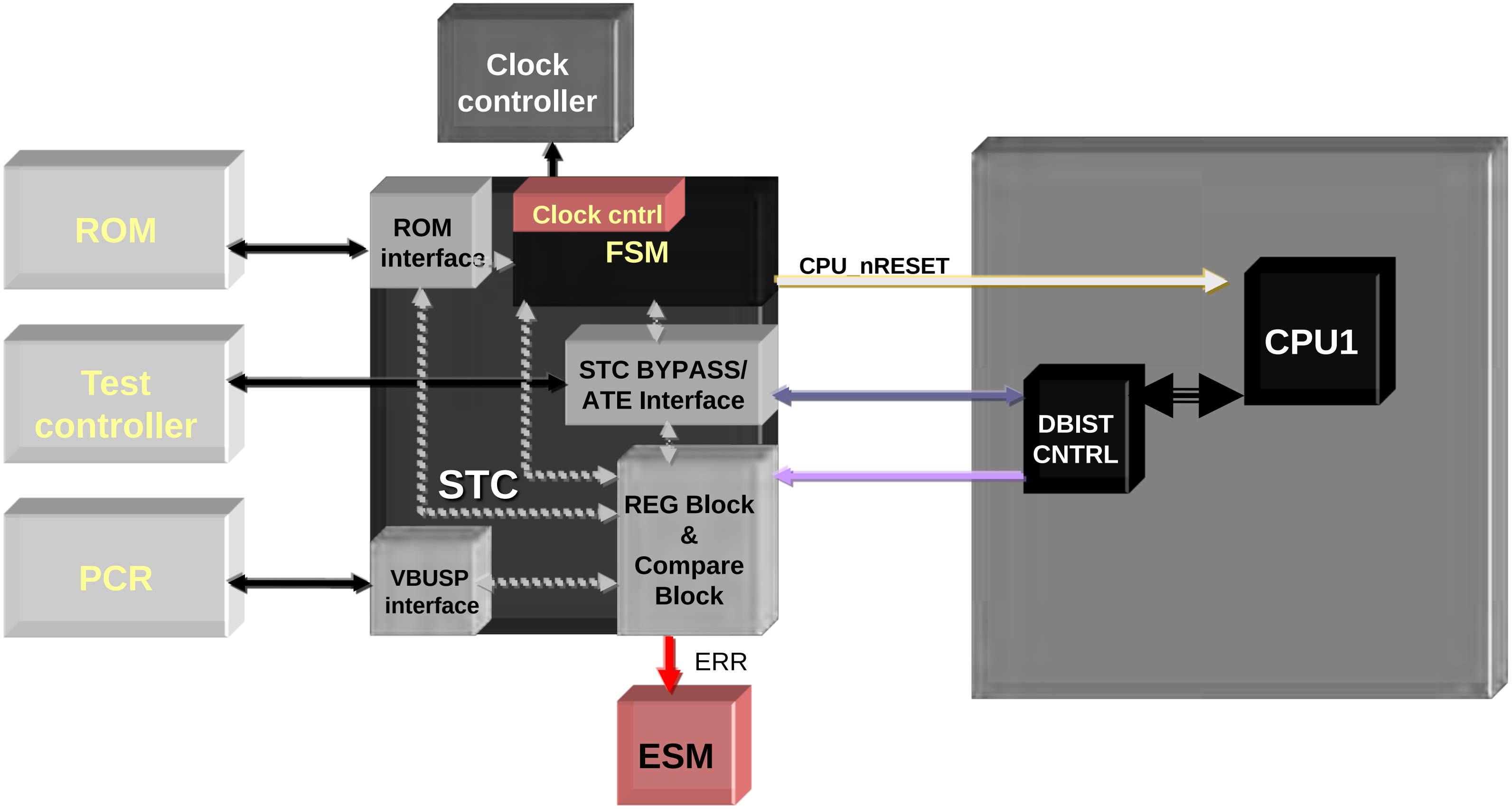
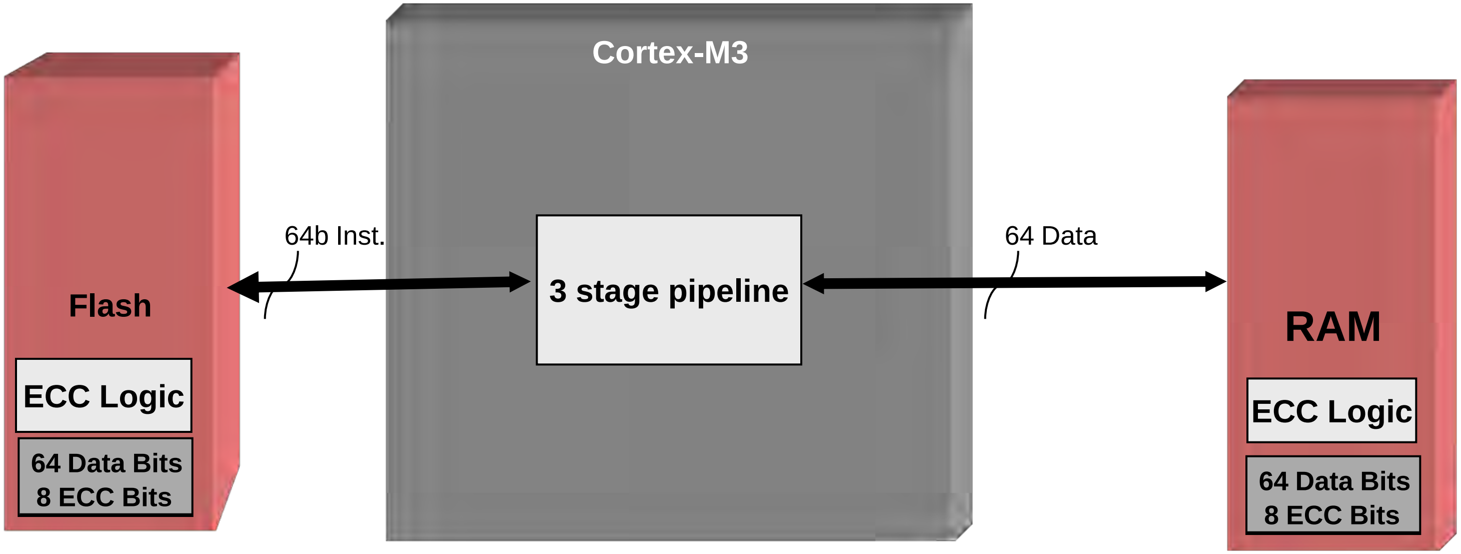
TMS470M Safety Features

CPU Self Test Controller (STC/LBIST)



- Provides High Diagnostic Coverage
- Significantly Lowers S/W and Runtime Overhead
- No SW BIST (Built In Self Test) Code overhead in Flash
- Simple to configure and start BIST via register

Flash / RAM ECC Protection

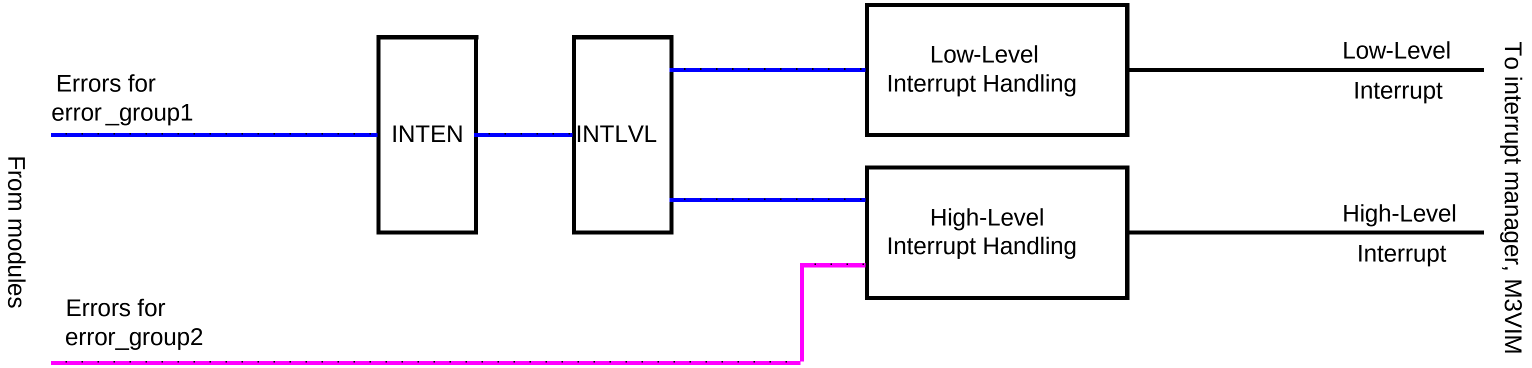


- ECC evaluated in the Memory Wrapper Logic
  - Single Bit Error Correction and Double Bit Error Detection

Safety Aspects of Network Interfaces

- Networked peripherals (DCAN, and SCI/LIN) are considered grey-channel / black-channel communications
- In such communications application level protocols (time redundancy, CRC in data packet, etc.) are necessary
- When such assumption is made, the Dangerous Undetected Failure from the network is effectively not measurable (<0.001 Failure In Time (FIT))
- Detailed fault data available upon request

Error Signaling Module (ESM)



TMS470M ESM Features

- ESM functions
  - Up to 64 error channels, divided into 2 different groups
    - 32 channels with configurable output for interrupt and error behavior
    - 32 channels with predefined output for interrupt and error behavior
  - Error forcing capability for self test
- ESM hardware
  - Hardware assistance for prioritizing error sources

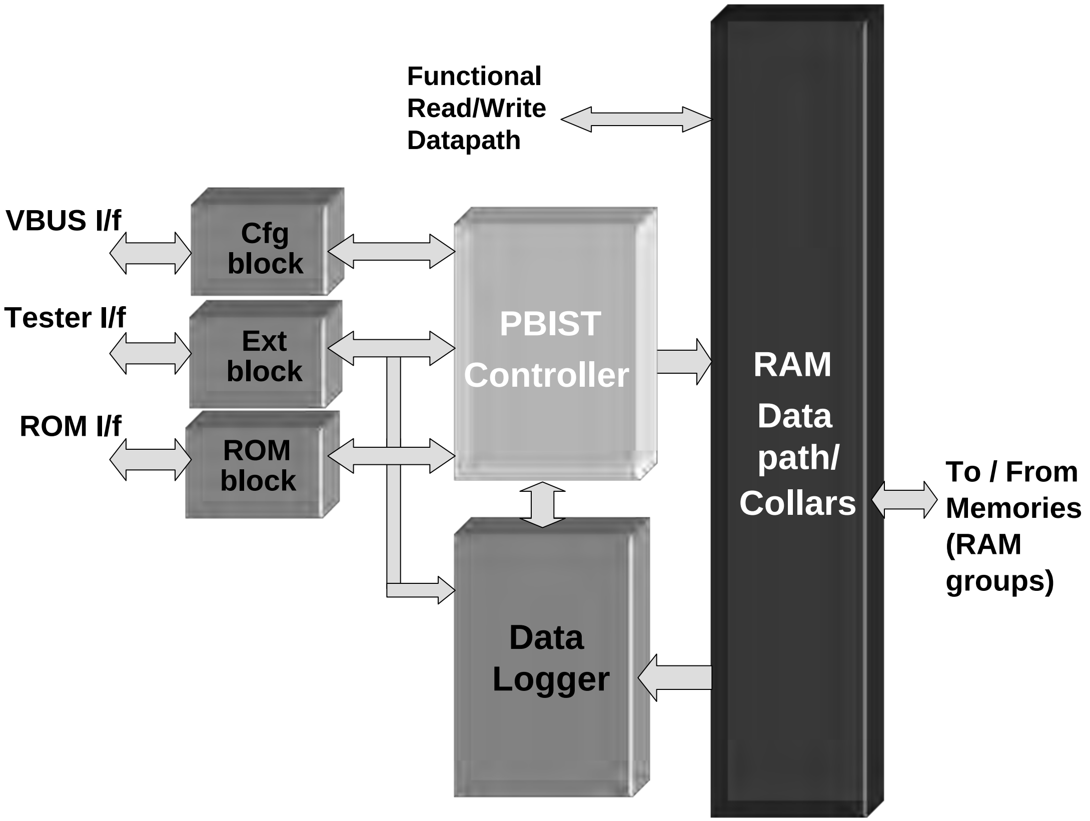
Additional Safety Features

- On chip voltage regulator (VReg)
  - Regulates core supply voltage (VCC) from digital I/O supply (VCCIOR)
  - Required operating range
    - 3.0-3.6 V (nom=3.3 V) for VCCIOR
    - 1.4-1.7 V (nom=1.55 V) for VCC

- Clock monitoring
  - Oscillator monitor
    - Detects failure if oscillator frequency exceeds defined min/max thresholds
    - Selectable hardware response on oscillator fail
  - PLL slip detector
    - Indicates PLL slip if phase lock is lost
    - Selectable hardware response on PLL slip
    - Internal backup ‘low power oscillator’ (LPO) clock source
  - External clock prescaler
    - Allows external monitoring of CPU clock frequency

- CPU with dedicated Memory Protection Unit (MPU)

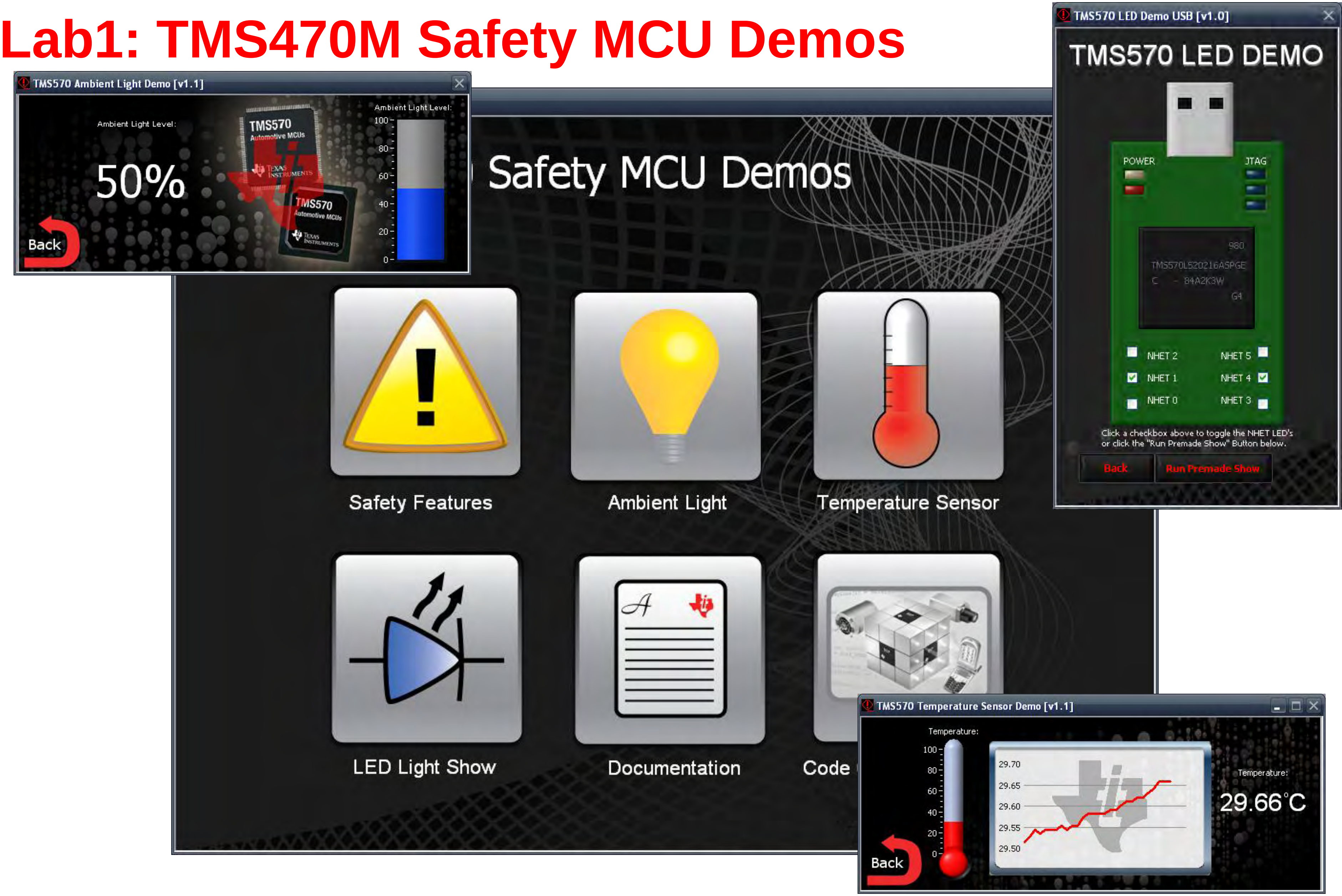
Programmable Memory BIST (MBIST)



- All on-chip RAMS can be tested
- Run at startup
- Multiple Memory Test Algorithms
- Detects multiple failure modes

LAB 1: TMS470M Safety Features Demo

Lab1: TMS470M Safety MCU Demos



TMS570 Ambient Light Demo [v1.1]

Ambient Light Level: 50%

Safety MCU Demos

TMS570 LED Demo USB [v1.0]

TMS570 LED DEMO

☐ NHET 2 NHET 5 ☐

☑ NHET 1 NHET 4 ☑

☐ NHET 0 NHET 3 ☐

Click a checkbox above to toggle the NHET LED's or click the "Run Premade Show" Button below.

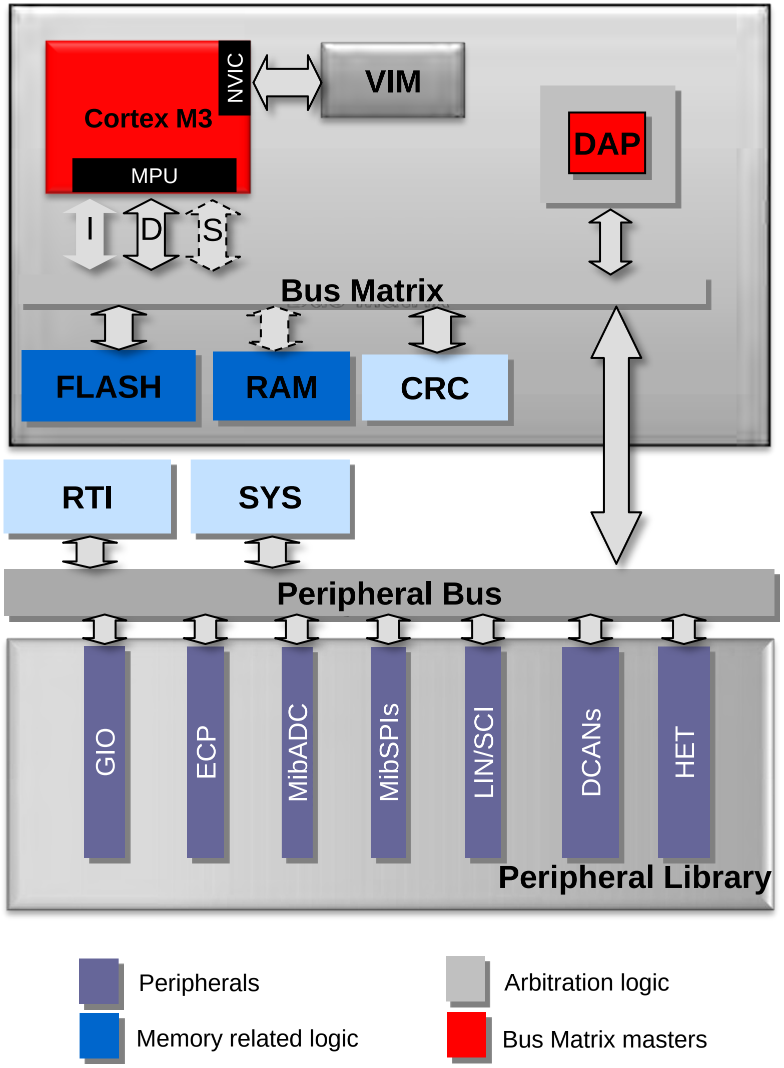
TMS570 Temperature Sensor Demo [v1.1]

Temperature: 29.66°C

TMS470M Architecture Overview: Memory Map, Clocking, Exceptions

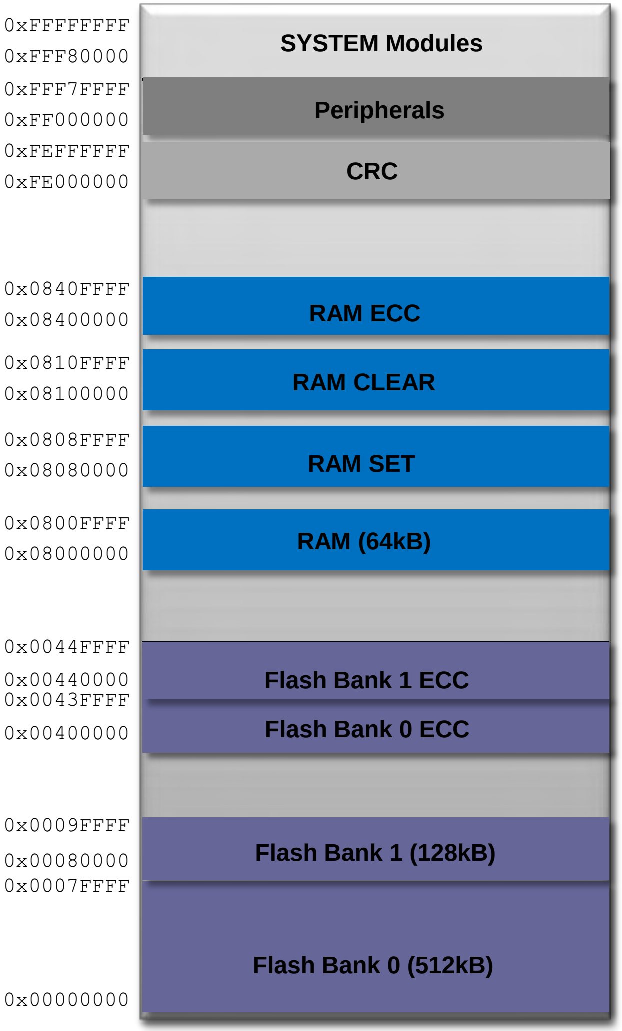
Architecture Overview

- Cortex M3 Features
  - 32 bit CPU
  - 3 stage pipeline
  - Harvard Architecture
  - ARM v7M instruction set
    - Thumb-2 mode (16/32-bit instruction)
  - 32x32 single cycle multiplier
  - Single cycle Shift and ALU operation
  - Hardware Divider
  - 3 x 32-bit bus interface (AHB)
  - Built in MPU
    - 1 background region
    - 8 memory regions
  - Built in NVIC
    - 2 interrupt levels
      - NMI
      - Vectorized IRQ
- Performance
  - 1.25 DMIPS/MHz



Memory Map

TMS470MF066xx



0xFFFFFFFF
0xFFF80000
SYSTEM Modules

0xFFF7FFFF
0xFF000000
Peripherals

0xFEFFFFFF
0xFE000000
CRC

0x0840FFFF
0x08400000
RAM ECC

0x0810FFFF
0x08100000
RAM CLEAR

0x0808FFFF
0x08080000
RAM SET

0x0800FFFF
0x08000000
RAM (64kB)

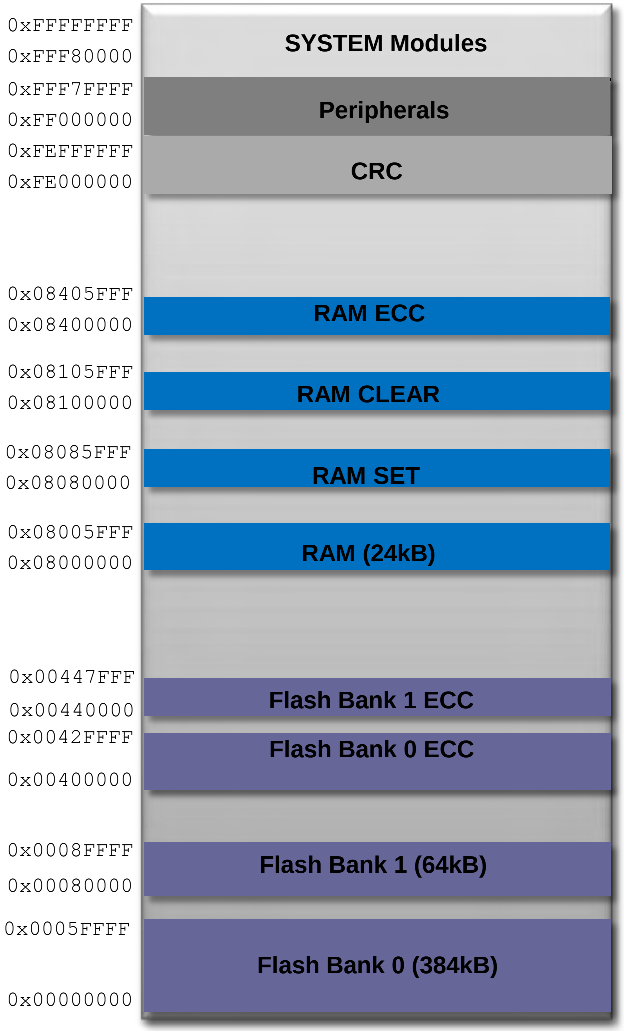
0x0044FFFF
0x00440000
Flash Bank 1 ECC

0x0043FFFF
0x00400000
Flash Bank 0 ECC

0x0009FFFF
0x00080000
Flash Bank 1 (128kB)

0x0007FFFF
0x00000000
Flash Bank 0 (512kB)

TMS470MF04xxx/03xxx



0xFFFFFFFF
0xFFF80000
SYSTEM Modules

0xFFF7FFFF
0xFF000000
Peripherals

0xFEFFFFFF
0xFE000000
CRC

0x08405FFF
0x08400000
RAM ECC

0x08105FFF
0x08100000
RAM CLEAR

0x08085FFF
0x08080000
RAM SET

0x08005FFF
0x08000000
RAM (24kB)

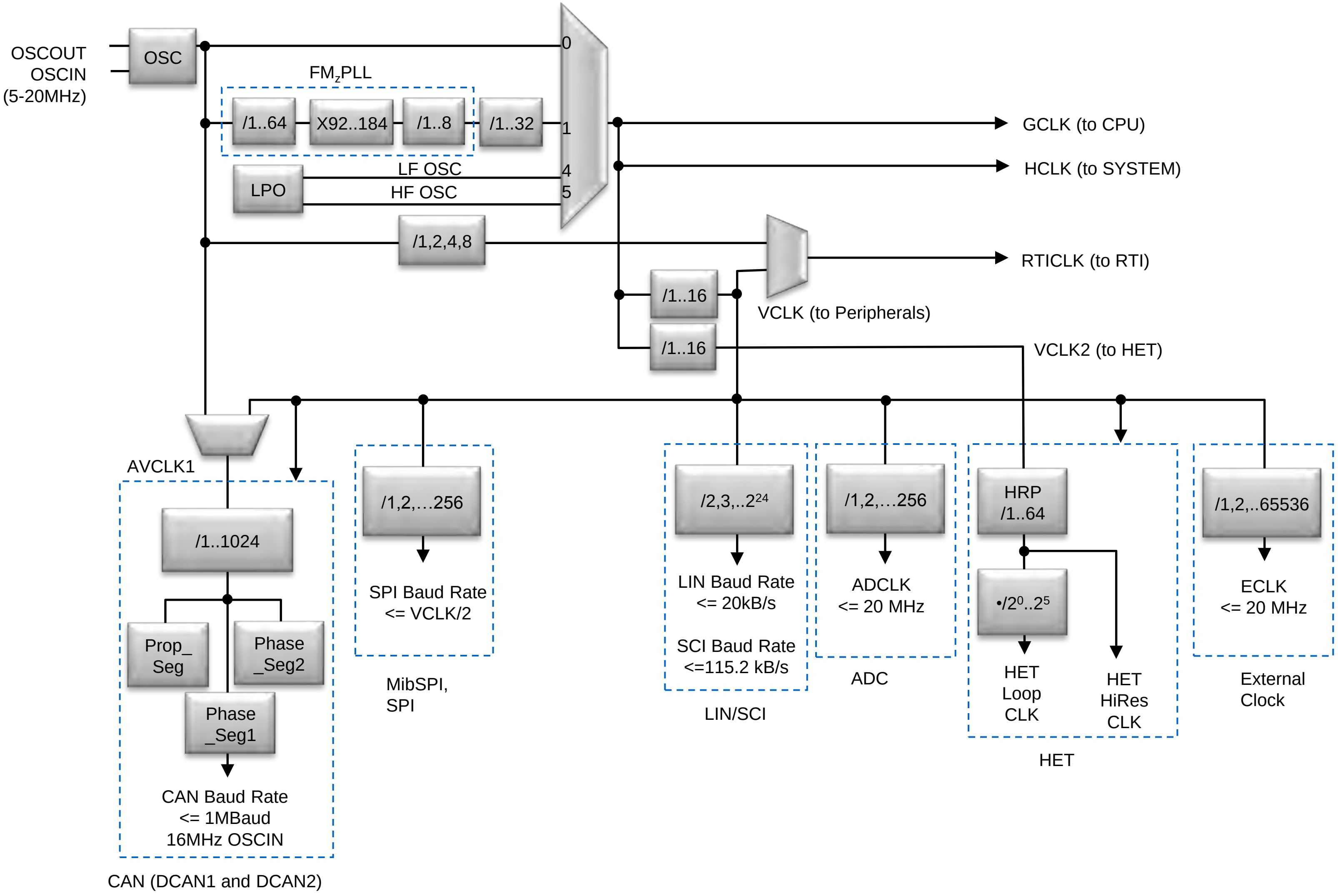
0x00447FFF
0x00440000
Flash Bank 1 ECC

0x0042FFFF
0x00400000
Flash Bank 0 ECC

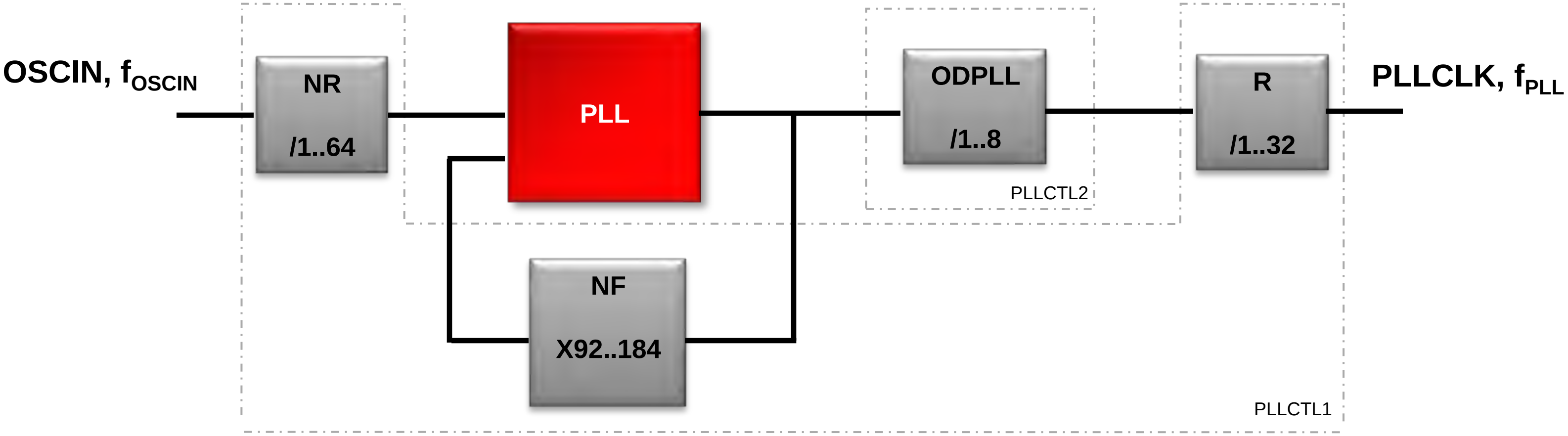
0x0008FFFF
0x00080000
Flash Bank 1 (64kB)

0x0005FFFF
0x00000000
Flash Bank 0 (384kB)

Clock Sources and Domains



Frequency Modulated PLL (FMzPLL)



The output frequency of the FMzPLL is given by:

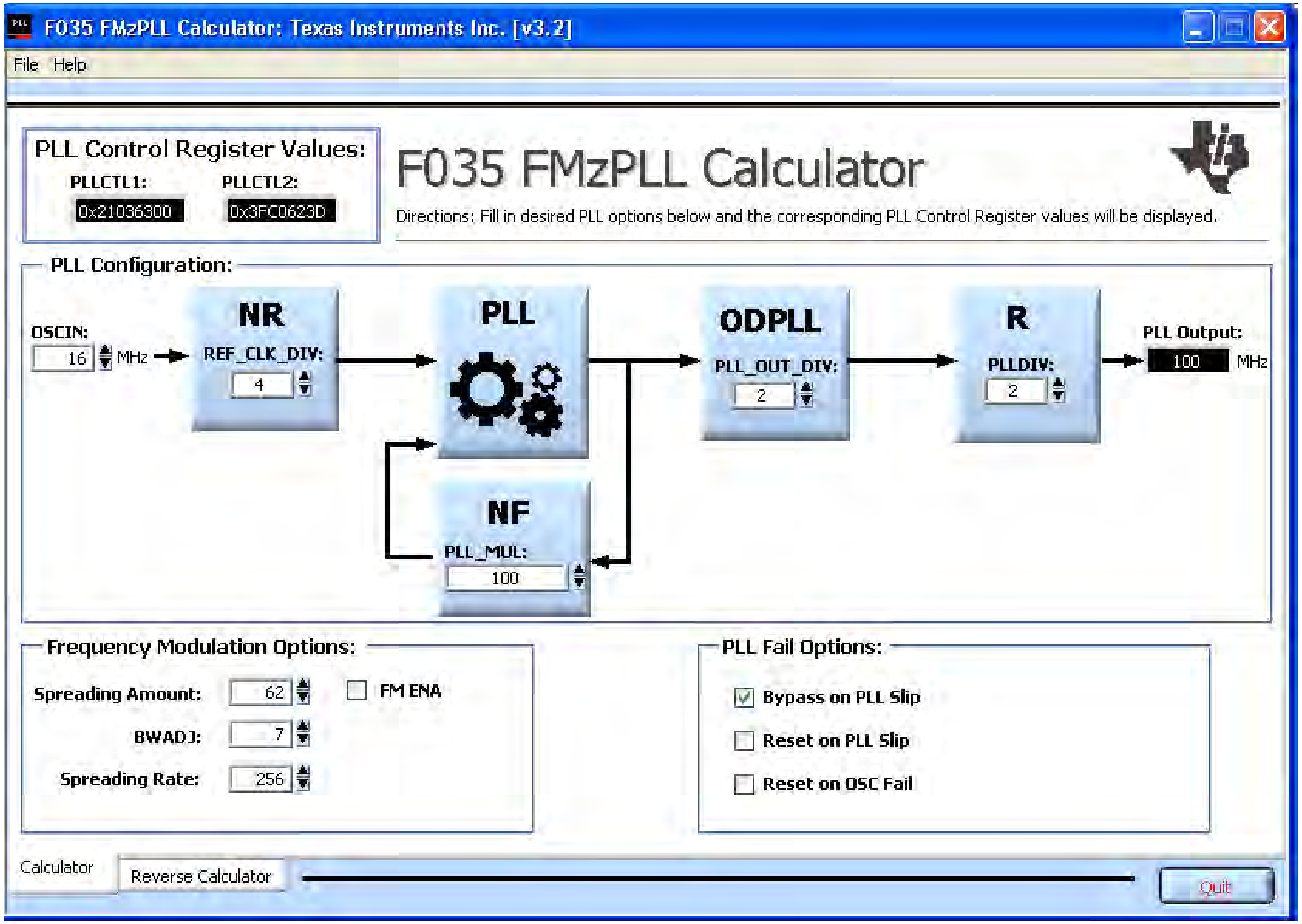
$$f_{PLL} = (f_{OSCIN} / NR) *NF/(ODPLL * R)$$

Where

NR – Reference Clock Divider ratio (REFCLKDIV)
NF – PLL feedback divider ratio (PLLMUL), 92 .. 184
ODPLL – PLL output divider ratio (ODPLL), 1 .. 8
R – PLL Divider Ratio (PLLDIV), 1 .. 32

Configure FMzPLL using PLLCTL1(0x70) and PLLCTL2 (0x74) registers of System frame1 (0xFFFF_FF00)

FMzPLL Calculator



PLL Control Register Values:

PLLCTL1: 0x21036300

PLLCTL2: 0x3FC0623D

F035 FMzPLL Calculator

Directions: Fill in desired PLL options below and the corresponding PLL Control Register values will be displayed.

PLL Configuration:

OSCIN: 16 MHz

NR REF_CLK_DIV: 4

PLL

ODPLL PLL_OUT_DIV: 2

R PLLDIV: 2

PLL Output: 100 MHz

NF PLL_MUL: 100

Frequency Modulation Options:

Spreading Amount: 62

☐ FM ENA

BWADJ: 7

Spreading Rate: 256

PLL Fail Options:

☑ Bypass on PLL Slip

☐ Reset on PLL Slip

☐ Reset on OSC Fail

- Can be used to determine PLLCTL1 and 2 register values based on the entered PLL and FM option settings
- Can be used to determine the PLL and FM settings based on the entered PLLCTL1 and 2 register values

Low Power Modes

- Doze Mode
  - Highest power consumption among low power modes
  - Fastest from wake up event to full-speed operation
  - Main oscillator remains active and clocks RTI module
  - Wake up from RTI or external sources (CAN, LIN, SCI, GIO, reset)
  - On chip Vreg in LPM1

- Sleep Mode
  - Lowest power consumption among low power modes
  - Slowest from wake up event to full-speed operation
  - All clock sources and clock domains are disabled
  - Wake up only from external sources (CAN, LIN, SCI, GIO, reset)
  - On chip Vreg in LPM1

Low Power Modes Summary

Doze Mode:

Vreg	High freq. oscillator	Internal low freq. clock	RTI clock	GCLK	HCLK	VCLKP	VCLK2	VCLKA1	PLL	Flash banks	Flash pumps
LPM1	**on**	**on**	**on**	**off**	**off**	**off**	**off**	**off**	**off**	**off**	**off**

Sleep Mode:

Vreg	LPO	Internal low freq. clock	RTI clock	GCLK	HCLK	VCLKP	VCLK2	VCLKA1	PLL	Flash banks	Flash pumps
LPM1	**off**	**off**	**off**	**off**	**off**	**off**	**off**	**off**	**off**	**off**	**off**

Reset Sources

- Power-on Reset
  - Asserted by external voltage supervisor or by internal voltage regulator
- Oscillator fail
  - Asserted by internal clock monitor when enabled by software
- CPU Reset
  - Asserted by CPU self-test controller after LBIST operation completes
- Software Reset
  - Asserted by software writing to the exception control register
- External Reset
  - Asserted by external circuitry driving the warm reset (nRST) signal LOW
- Debug Reset
  - Asserted by ICEPICK JTAG module

TMS470M: Flash Tools

nowECC

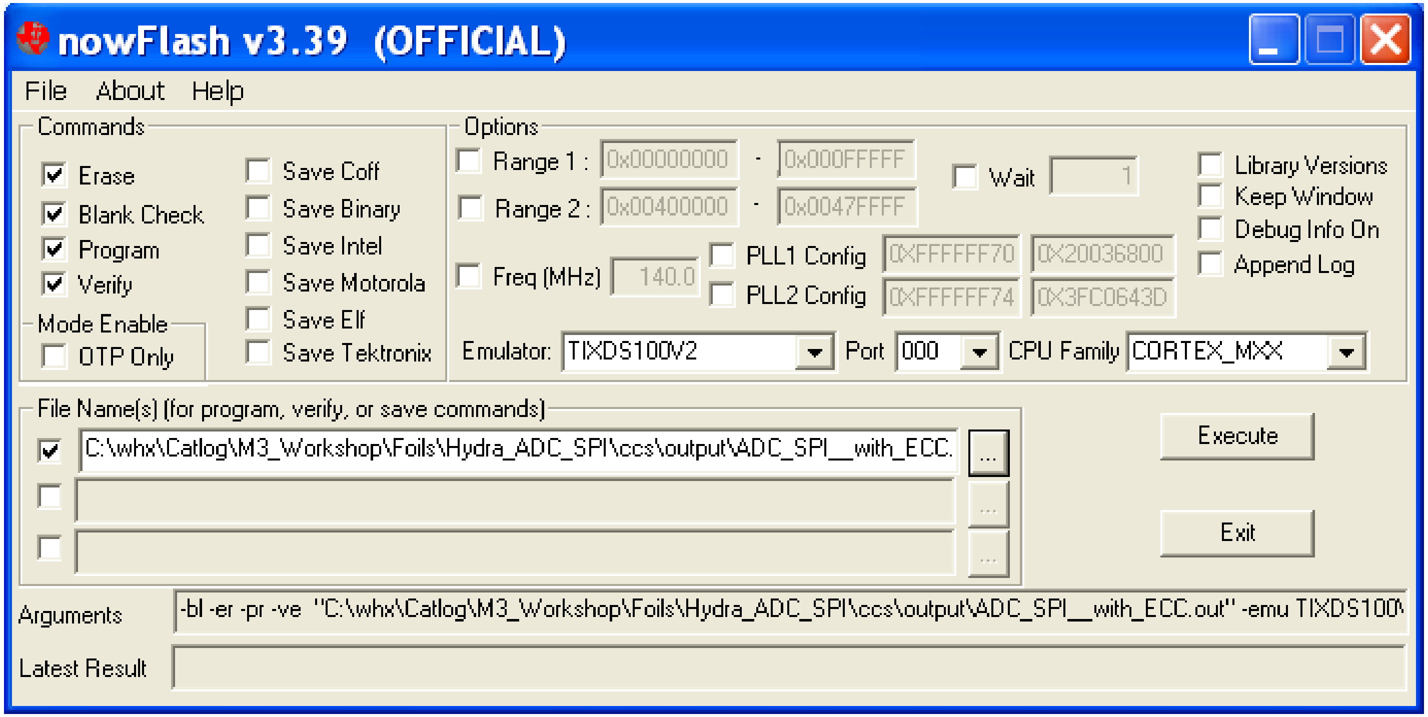
```
<return_value> nowECC [options] -i <input_file> [-o <output_file>]
```

- Generates ECC data for program flash
- Command-line executable
- Return value = 0 indicates no error during operation
  - Separate error codes to differentiate each type of error
- Input_file is only required parameter
  - Can be Extended Tektronix, Intel Hex, Motorola-S, COFF or ELF format
- Output_file specifies the name of the output file to be generated
  - If no name is specified, ECC is appended to input file specified

Options for Flash Programming

- On-board programming using nowFlash/Code Composer Studio v4.x
  - Requires JTAG connection
  - Emulators Supported:
    - Blackhawk BHUSB560M
    - Spectrum Digital XDS510PP, XDS510PP+, XDS510USB, XDS560RUSB
    - Signum JTAGjet
    - Texas Instruments SPI525, XDS100v2, XDS560
- On-board programming via customer boot-loader code
  - Must use Texas Instruments released API routines
  - Multiple communication interfaces can be used
  - Necessary to validate program and erase routines
- Off-board programming
  - Single-device or Concurrent programming
  - Supports high degree of automation

nowFlash



Commands

☑ Erase

☑ Blank Check

☑ Program

☑ Verify

Mode Enable

☐ OTP Only

☐ Save Coff

☐ Save Binary

☐ Save Intel

☐ Save Motorola

☐ Save Elf

☐ Save Tektronix

Options

☐ Range 1 : 0x00000000 - 0x000FFFFF

☐ Range 2 : 0x00400000 - 0x0047FFFF

☐ Wait 1

☐ Library Versions

☐ Keep Window

☐ Debug Info On

☐ Append Log

☐ Freq (MHz) 140.0

☐ PLL1 Config 0XFFFFFF70 0X20036800

☐ PLL2 Config 0XFFFFFF74 0X3FC0643D

Emulator: TIXDS100V2 Port 000 CPU Family CORTEX_MXX

File Name(s) (for program, verify, or save commands)

☑ C:\whx\Catlog\M3_Workshop\Foils\Hydra_ADC_SPI\ccs\output\ADC_SPI__with_ECC.

☐

☐

Arguments -bl -er -pr -ve "C:\whx\Catlog\M3_Workshop\Foils\Hydra_ADC_SPI\ccs\output\ADC_SPI__with_ECC.out" -emu TIXDS100\

Latest Result

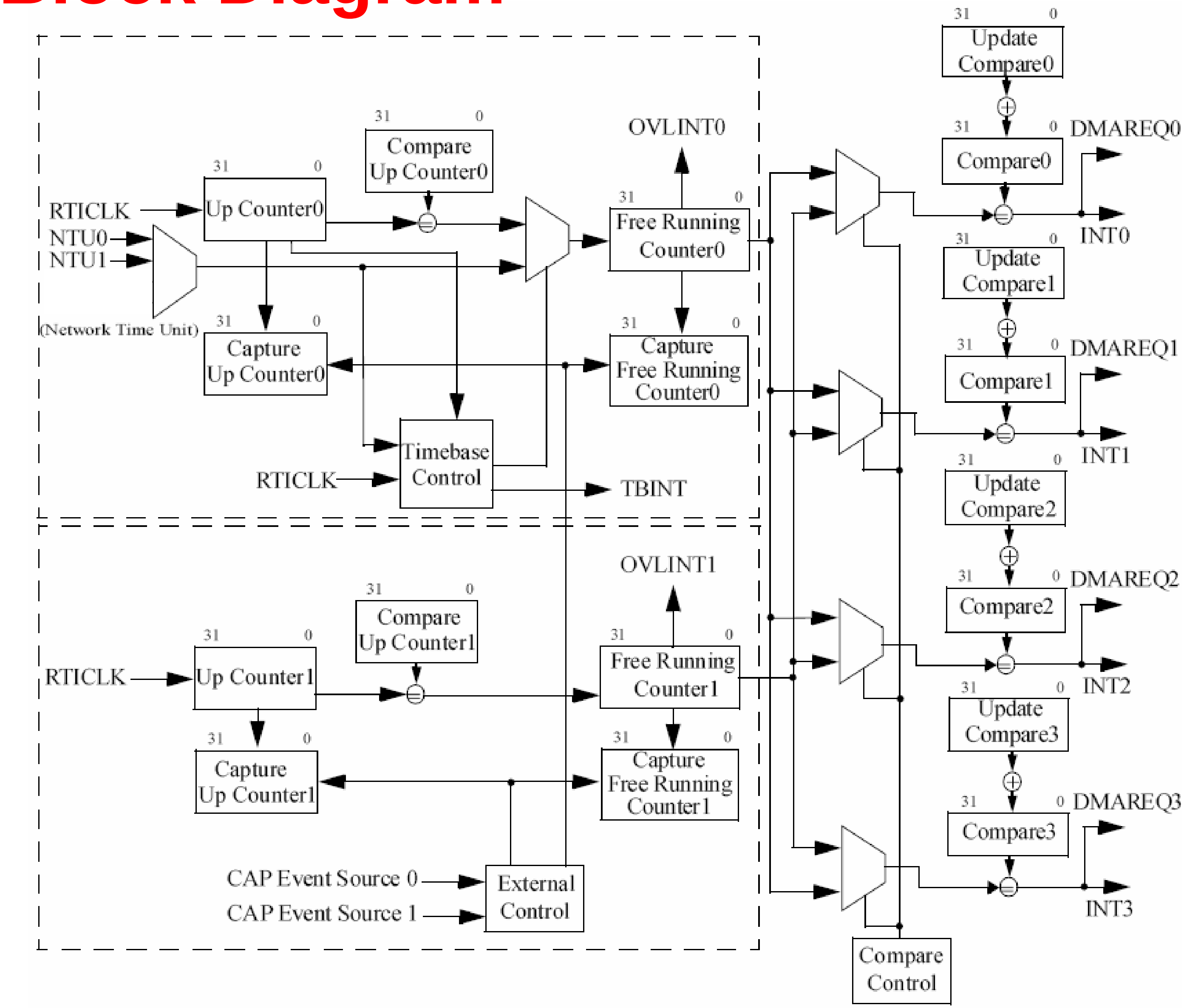
- PC-based software tool: GUI + command line executable
- Communicates with microcontroller via JTAG
- Can be used to program, erase, read, or verify flash memory
- Also supports execution of custom code out of RAM (from command line only)

Flash Application Programming Interface

- Distributed only as an object library file
- Supports flash operations out of on-chip RAM
- Supports operations at max specified device clock frequency
- Library routines for
  - Blank check
  - Compaction
  - Erase
  - Program zeros
  - Program data
  - Calculate checksum
  - Verify
- Routines also manage ECC

TMS470M: Real-Time Interrupt Module (RTI)

RTI: Block Diagram

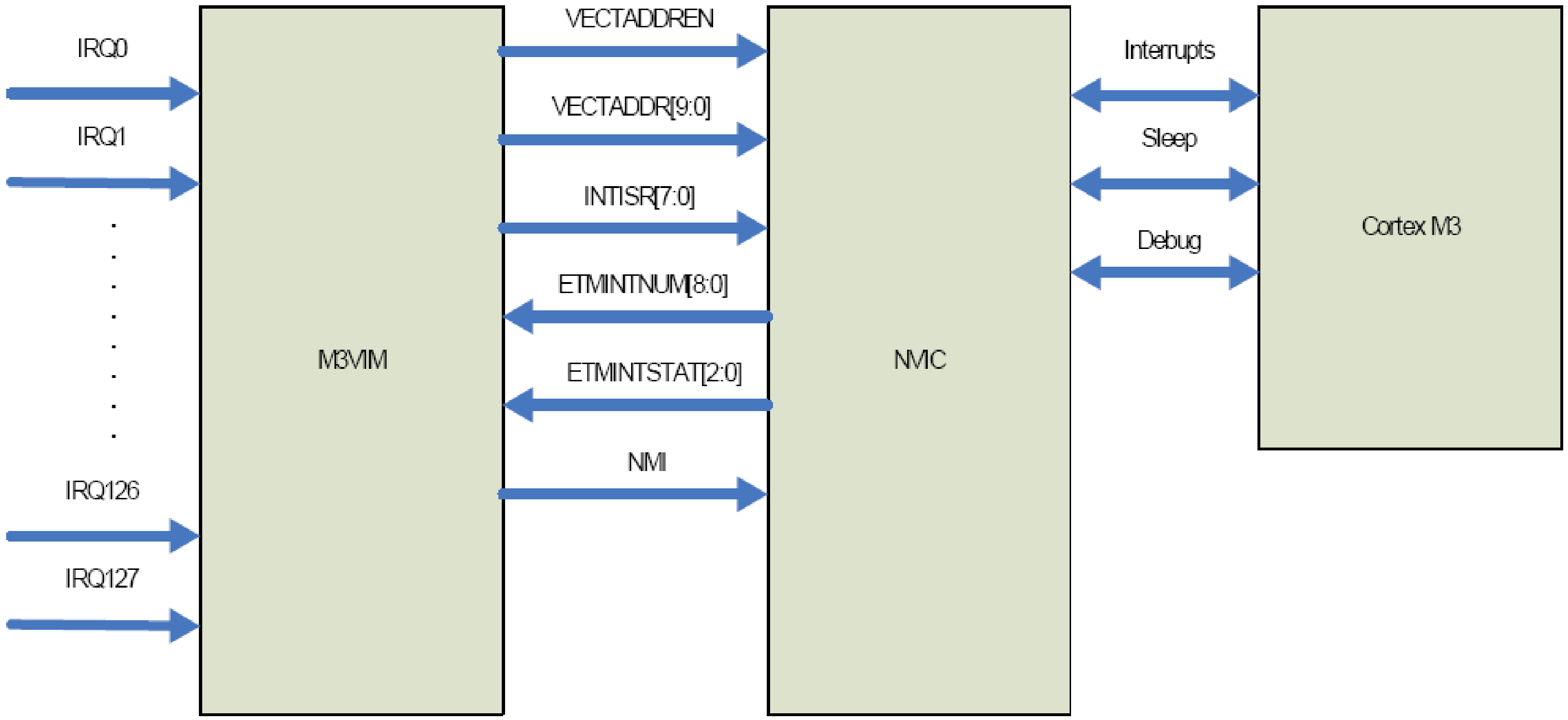


RTI: Main Features

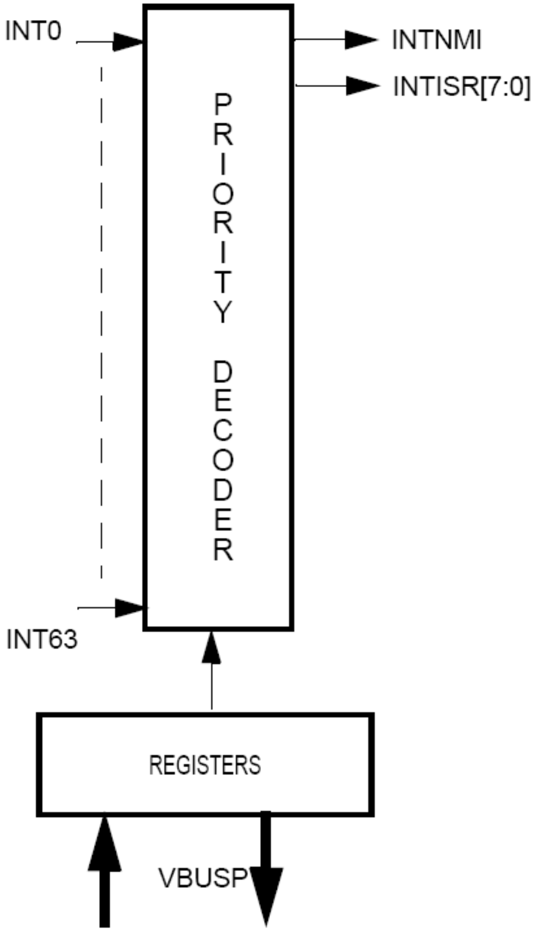
- Two independent counter blocks for generating different time bases
- Each block consists of
  - One 32-bit prescale counter
  - Two 32-bit free-running counters
  - Two capture registers for capturing the prescale and free-running counters
- Four compare interrupts
  - Each can use either of the two available free-running counters
  - Automatic update of compare values to minimize CPU intervention
- Two counter-overflow interrupts
  - Generated when a free-running counter overflows and goes to zero
- Four compare interrupts
- Digital Watchdog

TMS470M: Cortex M3 Vectored Interrupt Manager (M3VIM)

M3VIM: Interrupt Handling Block Diagram



M3VIM: Block Diagram



Minimum 32 lines, maximum 128, TMS470M → 64

M3VIM: Main Features

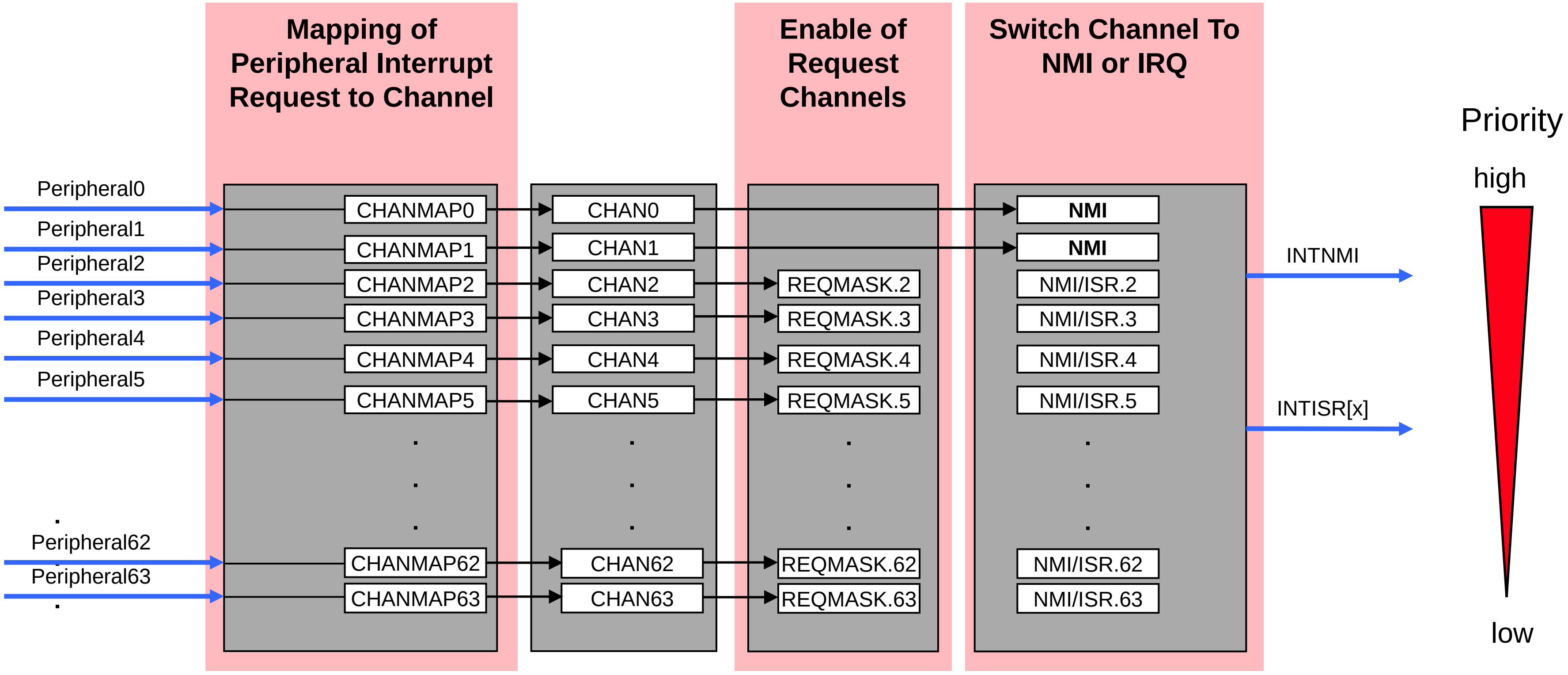
- **VIM Hardware**
  - Dedicated Vector Interrupt interface to Cortex M3 CPU
  - INTNMI (non-maskable interrupt) and INTISR (maskable interrupt) connection on Cortex M3.
  - 48 interrupt request lines
  - Up to 8 levels of nesting.

- **VIM Functions**
  - Map interrupt request to interrupt channel via programming.
  - Provides programmable priority through interrupt request mapping
  - Prioritizes the interrupt channels to the CPU
  - Provides the CPU with the address of the interrupt service routine (ISR)

M3VIM: Why M3VIM needed (M3VIM vs NVIC)?

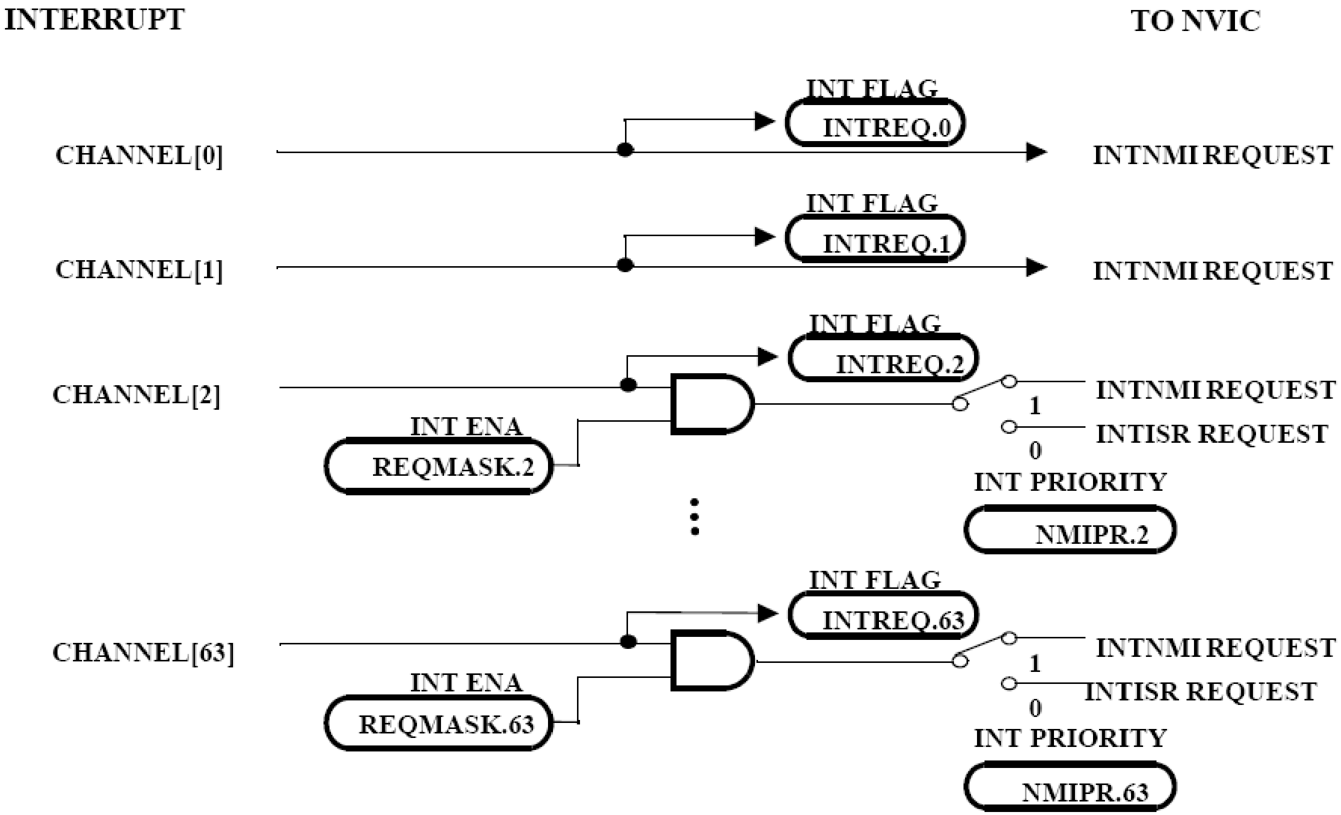
- NVIC is part of M3 core. Therefore its configuration can’t be changed without redesigning the entire core
- NVIC doesn’t support non-nested mode (required by some applications)
- NVIC doesn’t support SW mode of previous VIMs. M3VIM is needed to do this
- M3VIM is more flexible than M3 NVIC

M3VIM: Channel Mapping - Default State



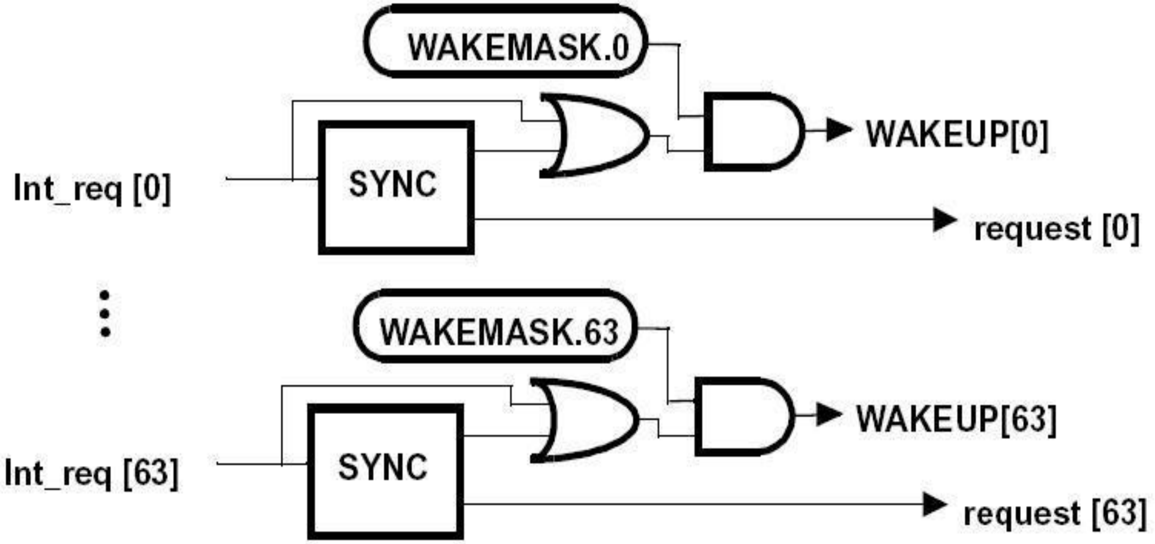
- Peripherals are mapped to corresponding channel
- All channels are disabled except for Channel 0 and Channel 1
- Note - INTNMI (Non-Maskable Interrupt): Channel 0 and Channel 1 are called NMI Channels and are non-maskable. These channels receive INT0 and INT1 and can not be changed

M3VIM: Input Channel Management



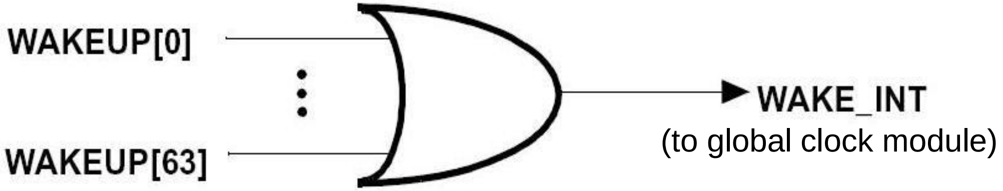
Write to NMIPR.0 and NMIPR.1 have no effect as channels 0 and 1 are not maskable

M3VIM: Wake-up Generation



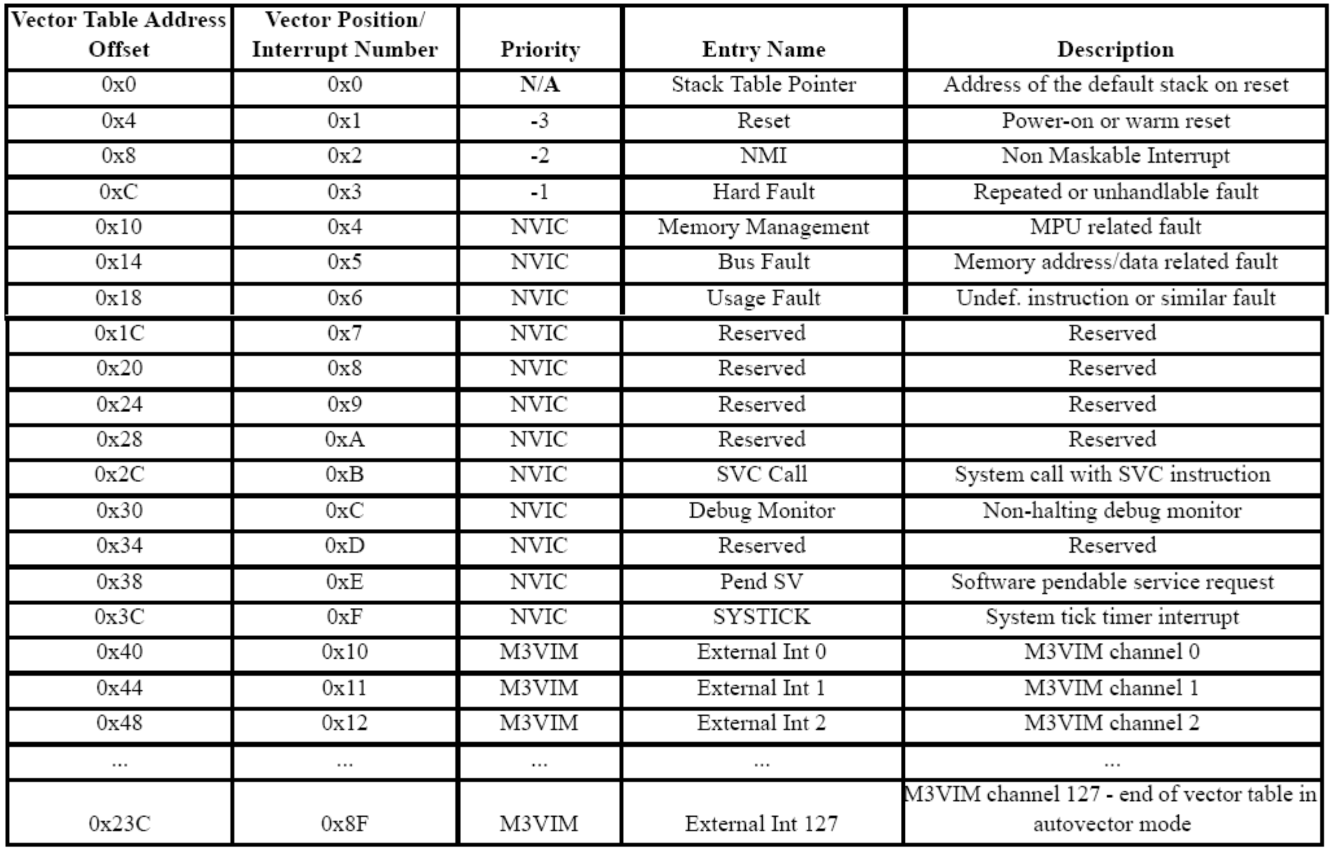
detail of the interrupt request input

Wake-up interrupt generation



(to global clock module)

M3VIM: Vector Table



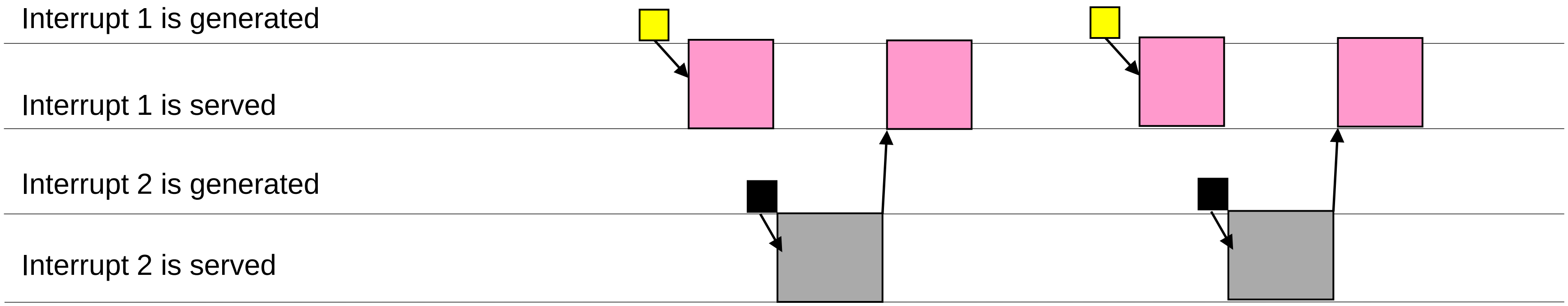
Vector Table Address Offset	Vector Position/ Interrupt Number	Priority	Entry Name	Description
0x0	0x0	N/A	Stack Table Pointer	Address of the default stack on reset
0x4	0x1	-3	Reset	Power-on or warm reset
0x8	0x2	-2	NMI	Non Maskable Interrupt
0xC	0x3	-1	Hard Fault	Repeated or unhandlable fault
0x10	0x4	NVIC	Memory Management	MPU related fault
0x14	0x5	NVIC	Bus Fault	Memory address/data related fault
0x18	0x6	NVIC	Usage Fault	Undef. instruction or similar fault
0x1C	0x7	NVIC	Reserved	Reserved
0x20	0x8	NVIC	Reserved	Reserved
0x24	0x9	NVIC	Reserved	Reserved
0x28	0xA	NVIC	Reserved	Reserved
0x2C	0xB	NVIC	SVC Call	System call with SVC instruction
0x30	0xC	NVIC	Debug Monitor	Non-halting debug monitor
0x34	0xD	NVIC	Reserved	Reserved
0x38	0xE	NVIC	Pend SV	Software pendable service request
0x3C	0xF	NVIC	SYSTICK	System tick timer interrupt
0x40	0x10	M3VIM	External Int 0	M3VIM channel 0
0x44	0x11	M3VIM	External Int 1	M3VIM channel 1
0x48	0x12	M3VIM	External Int 2	M3VIM channel 2
...	...	...	...	...
0x23C	0x8F	M3VIM	External Int 127	M3VIM channel 127 - end of vector table in autovector mode

Vector table starts at 0x0000 0000 by default. This address can be changed in NVIC

M3VIM: Interrupt nesting

- Disabled at reset
- Up to 8 nested interrupts allowed
- Can be disabled/enabled by writing NEST_ENABLE in NESTCTRL register
- Nesting statistics is reported in NETSTAT register. This is including highest level reached and overrun.
- If disabled, only INTISR[0] can interrupt core. No other ISRs can interrupt currently executed ISR.
- INTNMI will still interrupt INTISR[0] even if M3VIM is in nesting disabled mode.
- If enabled, 8 levels of nesting are supported through INTISR[7:0] signals.

M3VIM: Interrupt nesting example



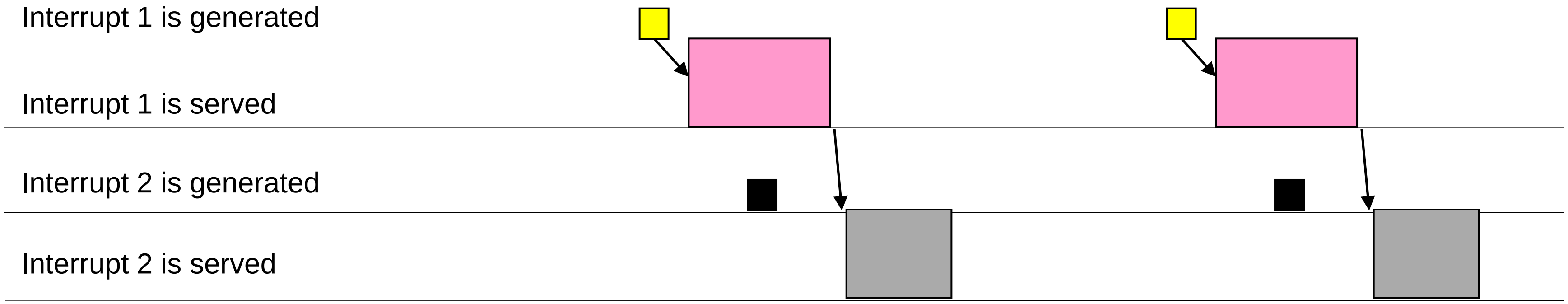
Interrupt 1 is generated

Interrupt 1 is served

Interrupt 2 is generated

Interrupt 2 is served

Interrupt 2 is higher priority than Interrupt 1. Nesting enabled



Interrupt 1 is generated

Interrupt 1 is served

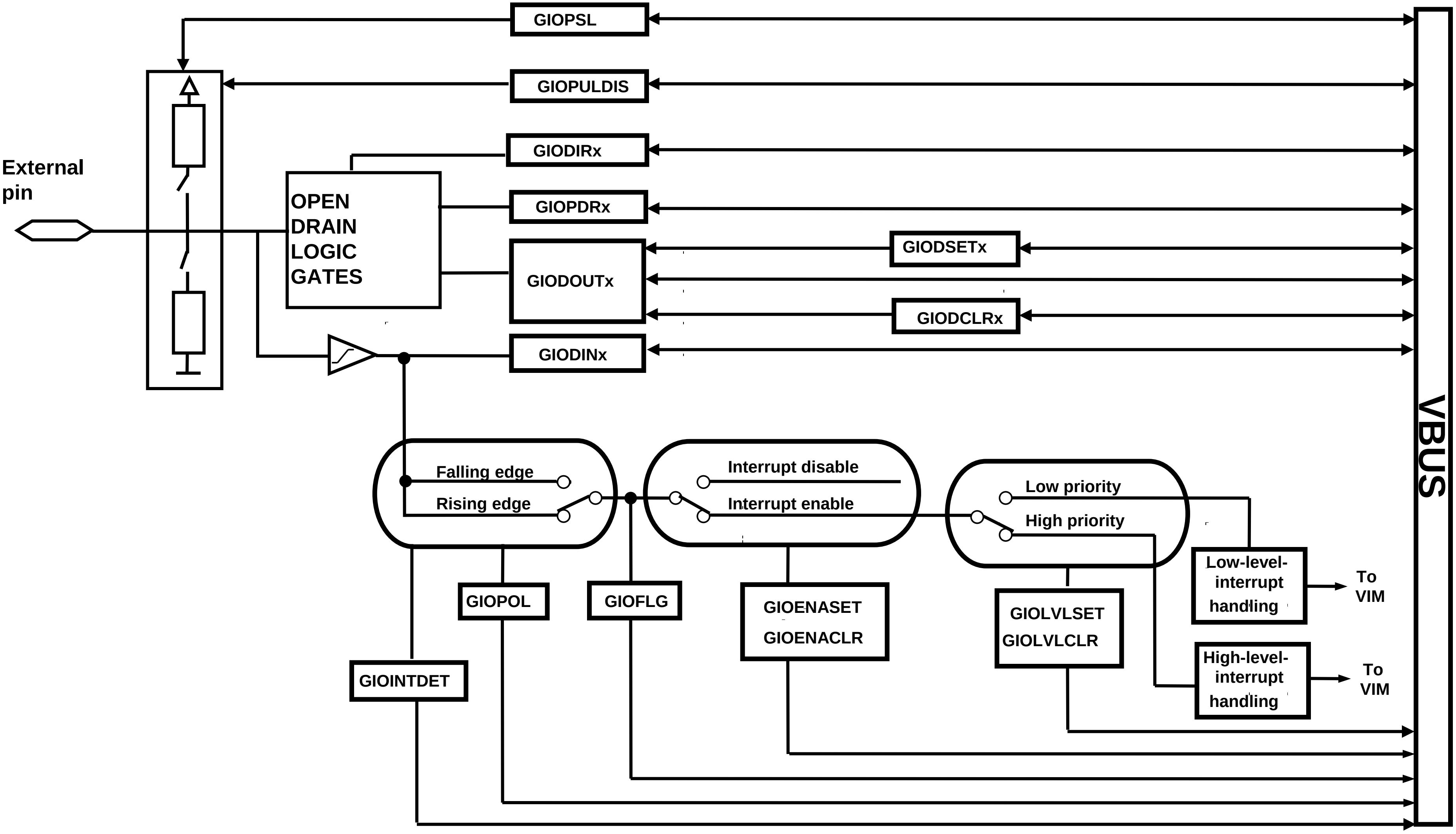
Interrupt 2 is generated

Interrupt 2 is served

Interrupt 2 is higher priority than Interrupt 1. Nesting disabled

TMS470M: General-Purpose I/O (GPIO)

GPIO: Block Diagram

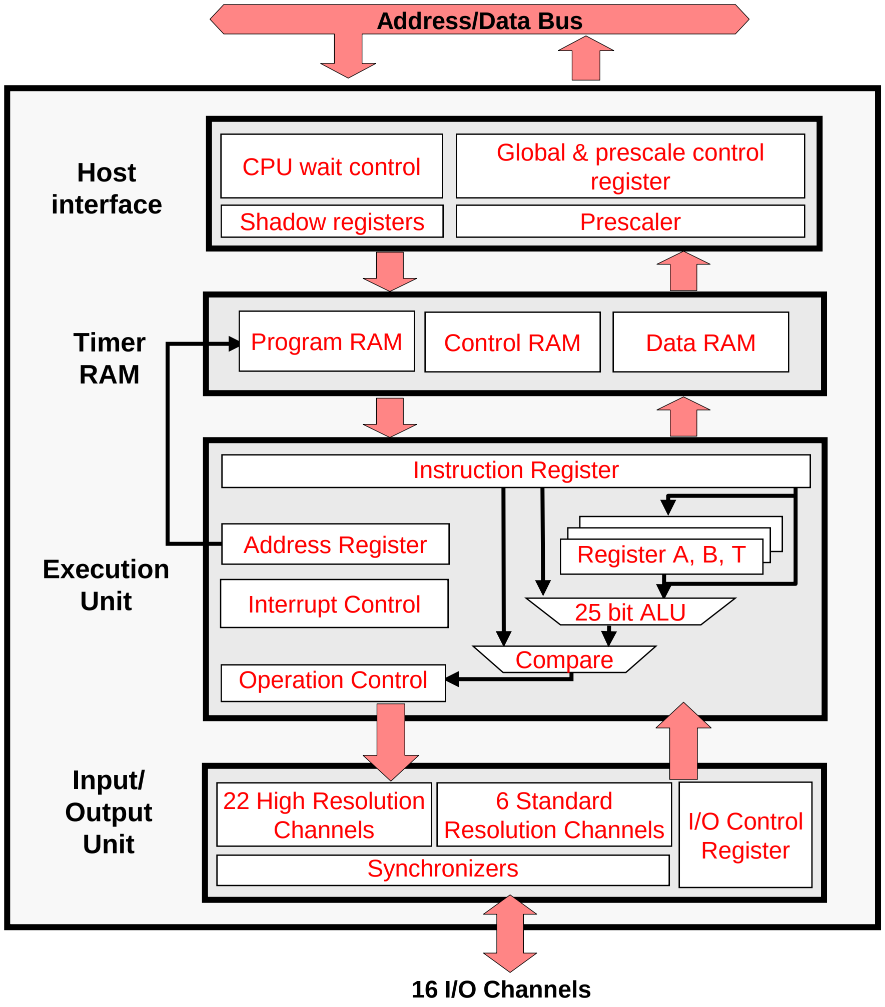


GPIO: Main Features

- Configurable as Input or Output via Direction Register
- Read/Write Registers
- Set Registers
- Clear Registers
- Separate Input Register
- Pull-up/Pull-down Configurability
- Open Drain Capability
- Interrupt Capability
  - Configurable Priority
  - Configurable Polarity: rising edge, falling edge, or both edges

TMS470M: High-End Timer (HET)

High End Timer (HET)



- User-programmable Timing Co-Processor
- Provides high level and complex timing functions with low CPU overhead
- 96-bit instruction word RAM with Parity protection
  - 64 Word Instruction RAM for TMS470M066xx
  - 128 Word Instruction RAM for TMS470M04x/03x
- Conditional program execution based on pin conditions and compares
- 16 input/output (I/O) channels (pins) for complex or classical timing functions such as capture, compare, PWM, GPIO
  - 12 additional internal channels
- Multiple 20-bit virtual counters for timers, event counters, and angle counters
- High Resolution I/Os and coarse resolutions implemented by sub loops for multiple resolution capability

HET: Application Examples

Pulse Width Modulation

- Single / multi channel PWMs
- PWM with synchronous / asynchronous duty cycle update
- PWM with synchronous period update
- Phase shift PWM's using RADM64 instruction

Frequency and Pulse Measurements

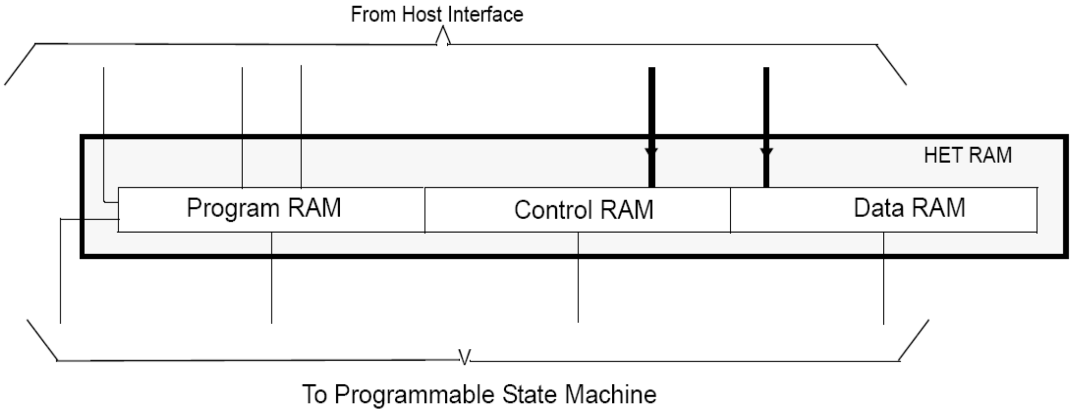
- Pulse width and period measurement (using PCNT)
- Period measurement using PCNT in HR mode, HRshare feature and 64 bit read access with “auto read/clear” bit set

Other Features

- Frequency Modulated Output
- Pulse width count (using PWCNT)
- Time stamp (using WCAP)
- Event counter (using ECNT)
- Pulse accumulator example (using ECNT)
- Multi-resolution scheme

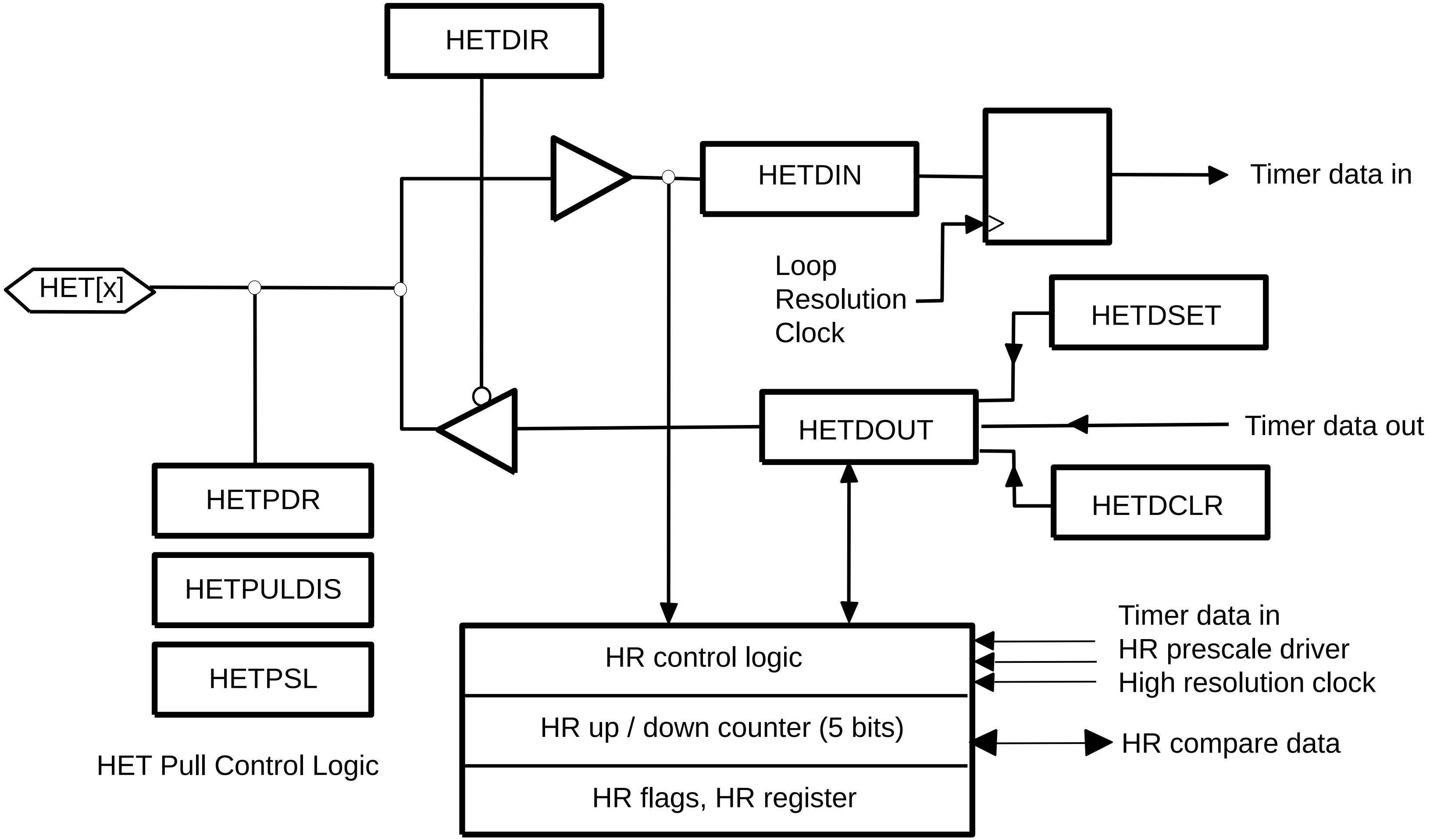
HET: Timer RAM

- Timer RAM supports Dual Port Access (one RAM address may be written while another address is read)
- RAM words are 96-bits wide (3*32bit, program, control, data)
- Write access by word (32bit) only



Instruction Address	Program Field Address	Control Field Address	Data Field Address	Reserved Address
00h	XX000h	XX004h	XX008h	XX00Ch
01h	XX010h	XX014h	XX018h	XX01Ch
02h	XX020h	XX024h	XX028h	XX02Ch
:	:	:	:	:
:	:	:	:	:
3Fh	XX3F0h	XX3F4h	XX3F8h	XX3FCh
40h	XX400h	XX404h	XX408h	XX40Ch
:	:	:	:	:
FFh	XXFF0h	XXFF4h	XXFF8h	XXFFCh

HET: I/O Structure



HET: Instruction Set Overview

Mnemonic	Instruction Name	Cycles
ACMP	Angle Compare	1
ACNT	Angle Count	2
ADCNST	Add Constant	2
ADM32	Add Move 32	1 or 2
APCNT	Angle Period Count	1 or 2
BR	Branch	1
CNT	Count	1 or 2
DADM64	Data Add Move 64	2
DJNZ	Decrement and Jump if Non-Zero	1
ECMP	Equality Compare	1
ECNT	Event Count	1
MCMP	Magnitude Compare	1
MOV32	Move 32	1 or 2
MOV64	Move 64	1
PCNT	Period/Pulse Count	1
PWCNT	Pulse Width Count	1
RADM64	Register Add Move 64	1
SCMP	Sequence Compare	1
SCNT	Step Count	3
SHFT	Shift	1
WCAP	Software Capture Word	1

HET: Command Line Assembler

- Invoking the **NHET assembler (hetp.exe):** **hetp** [options] input file

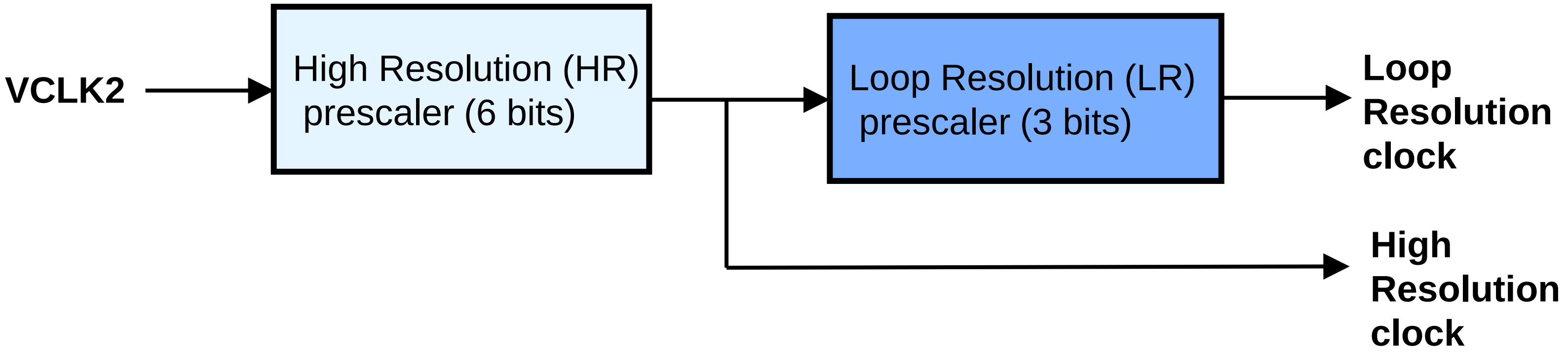
- Options:
  - -c32 produces an output file containing assembler directives for the TMS570 CodeGen Tools
  - -hc32 produces a **C file** and a **header file**. (used together with the -nx option)
  - -nx specifies the x-th HET module on the device (used together with -hc32 option)
  - -l (lowercase L) produces a listing file with the same name as the input file with a .lst extension.
  - -x produces a cross-reference table and appends it to the end of listing file.

- Example: **hetp -hc32 -n0 pwm.het**
- Input: pwm.het contains the assembly source of the HET program
- Output: pwm.c provides a C array, which contains the HET program opcode

  pwm.h provides a C structure, which allows a simple access to the NHET fields from other C code

HET: Time Base

- VCLK2 is used as base clock for the High End Timer
- A 6-bit prescaler dividing the system clock by a user-defined high-resolution (HR) prescale divide rate (hr) stored in the 6-bit HR prescale factor code (with a linear increment of codes).
- A 3-bit prescaler dividing the HR clock by a user-defined loop-resolution prescale divide rate (lr) stored in the 3-bit loop-resolution prescale factor code (with a power of 2 increment of codes).


VCLK2
High Resolution (HR) prescaler (6 bits)
Loop Resolution (LR) prescaler (3 bits)
Loop Resolution clock
High Resolution clock

Exercise: Using HET as GIO

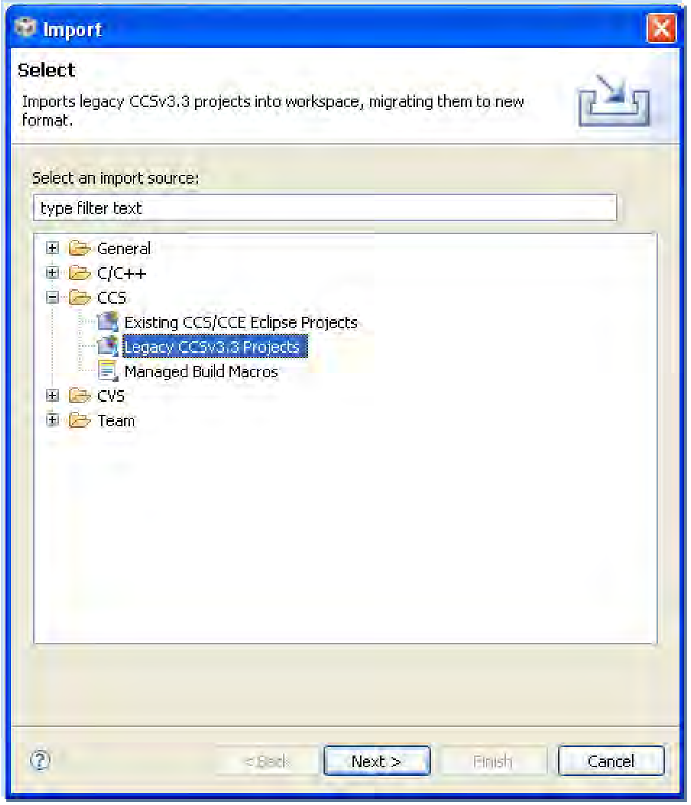
Overview

- In this project we will:
  - Set up the workspace in Code Composer Studio 4 (CCS4)
  - Import legacy code into CCS4
  - Write code to turn on the LED on HET pin 1
  - Build and Deploy our code to the microcontroller

Setting up Code Composer Studio 4 (CCS4)

- Launch CCS4
  - Start > Programs > Texas Instruments > Code Composer Studio v4 > Code Composer Studio v4
- When it launches, CCS will ask you to select a workspace, we will chose “C:\myWorkspace”
- Once it loads, go to:

  File > Import > CCS > Legacy CCSv3.3 Projects
- Browse to select the project file (.\workspace\HET31_gpio_M3\devices\TMS470M\M3\HET1\HET\HET31_gpio.pjt) →NEXT
- Select Code Generation Tools (default should be ARM, 4.6.3) →NEXT
- Select Advanced Options (nothing to select) →Finish



Import

Select

Imports legacy CCSv3.3 projects into workspace, migrating them to new format.

Select an import source:

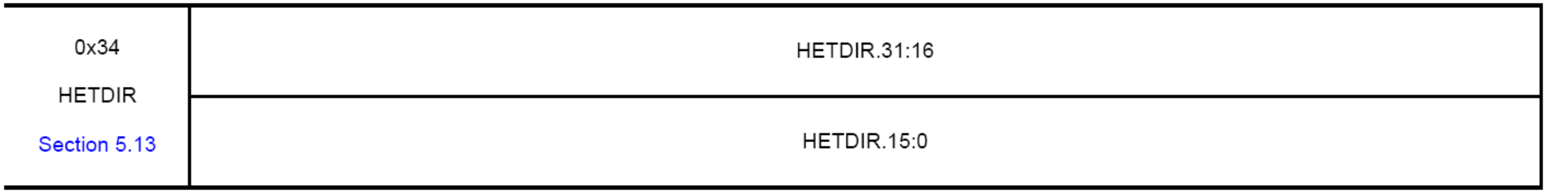
type filter text

Writing the Code

- On the left hand side in the “C/C++ Projects” explorer, open “HET31_gpio.c”
- Fill in HET register entries to match the comments

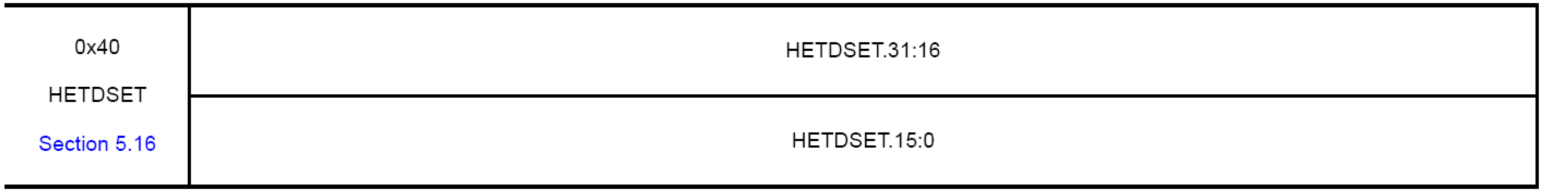
```
void main ()
{

   /* HET 1 Pin Direction - o/p*/
   HET_Ptr->CCDIR   = ______________;

    while (1)  /* Loop forever... */
    {
         /* Set HET1 output - HET1 --> high*/
         HET_Ptr->CCDSET   = ______________;

         delay(10000);

         /* Clear HET 1 output - HET1 --> Low */
         HET_Ptr->CCDCLR=______________;

         delay(10000);
    }

}
```

Writing the Code



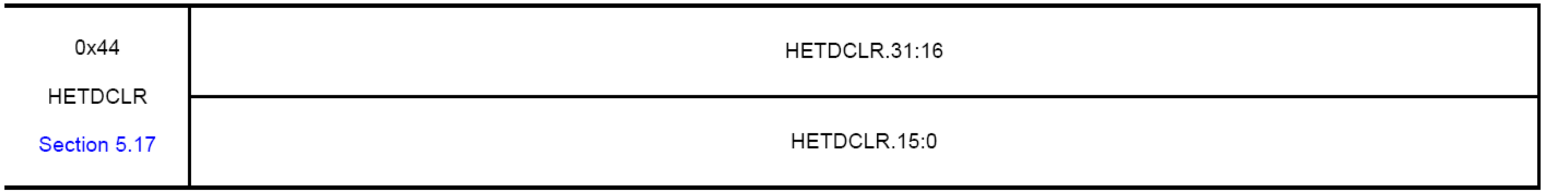
0x34 HETDIR Section 5.13	HETDIR.31:16
	HETDIR.15:0

- Writing a 1 to corresponding channel number will set the channel/pin to an output. Writing as 0 sets the pin as an input. Bits for channels without associated IO pins have no affect.



0x40 HETDSET Section 5.16	HETDSET.31:16
	HETDSET.15:0

- Writing a 1 to corresponding channel number will set the channel/pin output high. Writing as 0 has no affect. Bits for channels without associated IO pins have no affect.



0x44 HETDCLR Section 5.17	HETDCLR.31:16
	HETDCLR.15:0

- Writing a 1 to corresponding channel number will clear the channel/pin output to a low. Writing as 0 has no affect. Bits for channels without associated IO pins have no affect.

Writing the Code

- Complete the code per the register definitions:

```
void main ()
{

   /* HET 1 Pin Direction - o/p*/
   HET_Ptr->CCDIR    = 0x00000002;

    while (1)  /* Loop forever... */
    {
          /* Set HET1 output - HET1 --> high*/
          HET_Ptr->CCDSET    = 0x00000002;

          delay(10000);

          /* Clear HET 1 output - HET1 --> Low */
          HET_Ptr->CCDCLR= 0x00000002;

          delay(10000);
    }

}
```

Compiling the Project

- The code is now complete and we are ready to build our project.
  - Go to Project > Build Active Project

- Now that we have our .out file, we need to program the microcontrollers Flash memory.

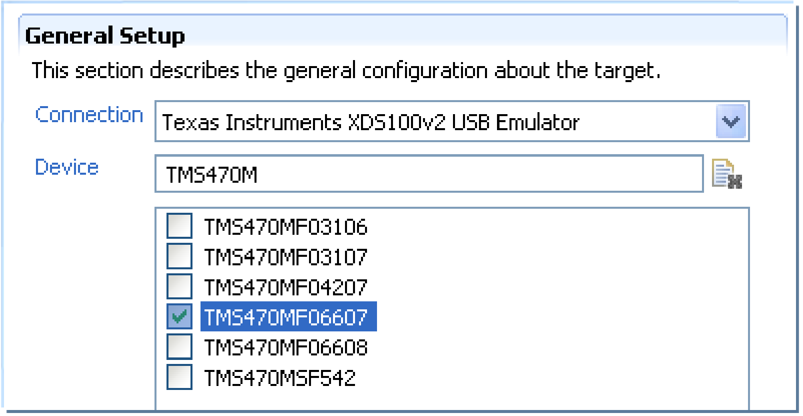
Creating a Target Configuration

- Before we begin, we must make a new target configuration, this tells CCS4 what device this project is designed for.
  - Target > New Target Configuration

- A new window will appear, we will make our file name “TMS470M.ccsxml”
- Click Finish

Creating a Target Configuration…

- A new tab will appear with a list of emulators and devices.
  - Connection: Texas Instruments XDS100v2 USB Emulator
  - In the text box labeled “Type Filter Text”, type “TMS470M”.
    - This will narrow the search down to just TMS470M devices, select TMS470MF06607
  - Click “Save” on the right



General Setup

This section describes the general configuration about the target.

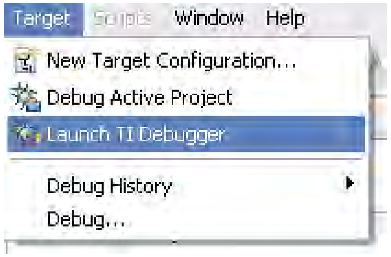
Connection: Texas Instruments XDS100v2 USB Emulator

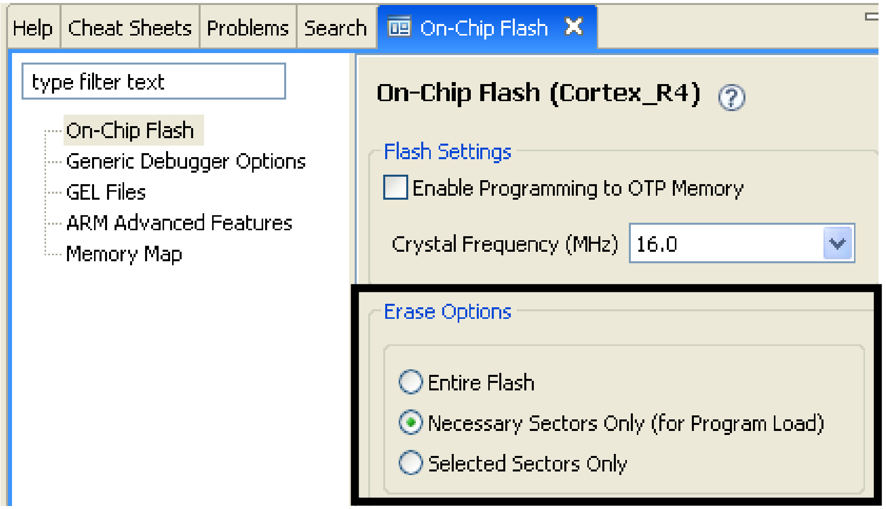
Device: TMS470M

- ☐ TMS470MF03106
- ☐ TMS470MF03107
- ☐ TMS470MF04207
- ☑ TMS470MF06607
- ☐ TMS470MF06608
- ☐ TMS470MSF542

Flash Programming Configuration

- It is possible to make the flash programming process much faster by only erasing and programming the necessary regions of flash memory.
  - To do so, go to Tools > On-Chip Flash
    - If ‘On-Chip Flash’ option is not available, select Target > Launch TI Debugger. Then go to Tools > On-Chip Flash
  - In the window that appears on the right, under Erase Options, check “Necessary Sectors Only (for Program Load)”





On-Chip Flash (Cortex_R4)

Flash Settings

□ Enable Programming to OTP Memory

Crystal Frequency (MHz) 16.0

Erase Options

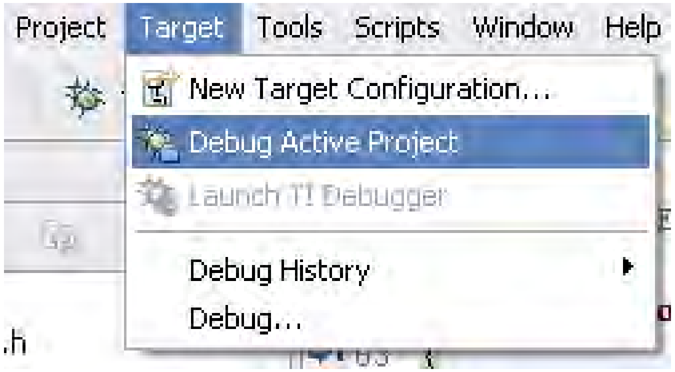
○ Entire Flash

● Necessary Sectors Only (for Program Load)

○ Selected Sectors Only

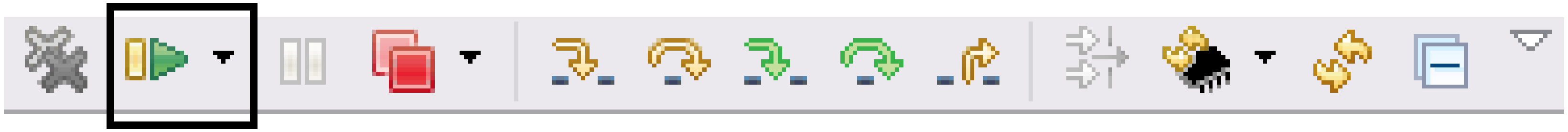
Programming the Flash

- We are now ready to program the flash.
  - Go to Target > Debug Active Project
  - A new window should appear as it programs the flash memory.
    - This may take a few moments.



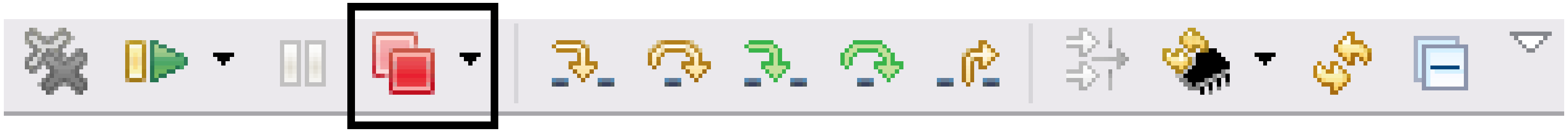
Testing our Program

- Click the green arrow on the debug tab to run our program



Alternatively the program can be run without the debugger connected by:

- Clicking the red square on the debug tab to terminate the debugger’s connection



- Hit the reset button on the board and the NHET[1] LED should illuminate.
- Congratulations! You have completed the lab.

TMS470M: Multi-Buffered Serial Peripheral Interface (MibSPI)

SPI / MibSPI Features

- The SPI / MibSPI has the following attributes:
  - 16-bit shift register
    Receive buffer register
  - 8-bit baud clock generator
    Serial Clock (SPICLK) I/O pin
  - SPI Enable (SPIENA) I/O pin (4 or 5-pin mode only)
  - Slave Chip Select (SPISCS[7:0]) I/O pin (4 or 5-pin mode only)

- The SPI / MibSPI allows software to program the following options:
  - SPISOMI / SPISIMO pin direction configuration
  - SPICLK pin source (external/internal)
  - MibSPI pins as functional or digital I/O pins

SPI / MibSPI Features

- For each Buffer, following features can be selected from 4 different combinations of Formats using the control fields in the buffer:
  - SPICLK frequency ([VBUSPCLK]/2 through /256)
  - Character length (2 to 16 bits)
  - Phase (delay/no delay), Polarity (high or low)
  - Enable/Disable Parity for transmit & receive
  - Enable/Disable timers for ChipSelect Hold & Setup timers
  - Direction of shifting, MSBit first or LSBit first

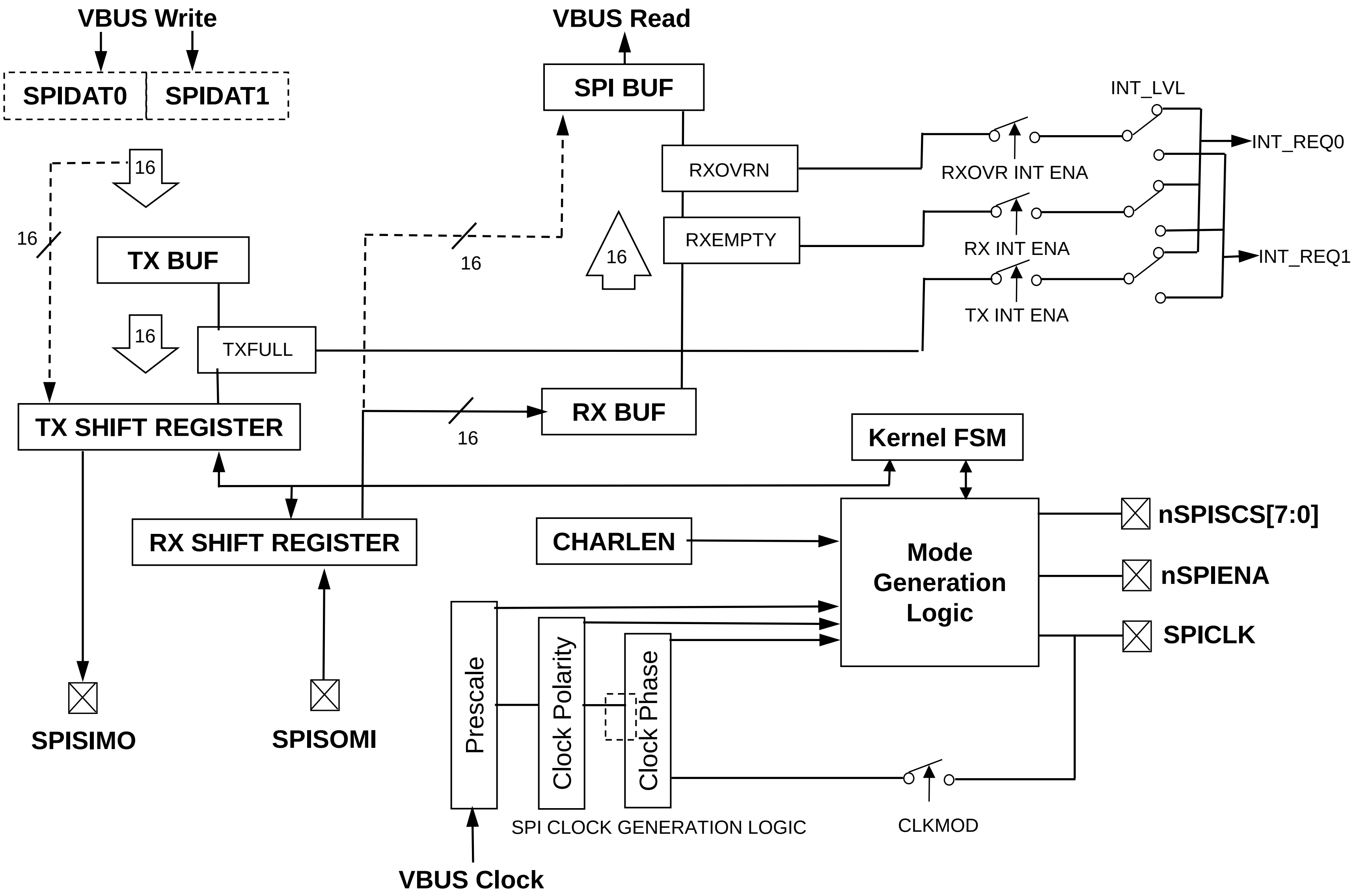
SPI / MibSPI Features

- In Multibuffer Mode (uses the Multibuffer RAM (up to 64 Buffers)), in addition to the above, many other features are configurable:
  - Number of buffers for each peripheral (or data source/destination) or group (up to 16 transfer groupings)
  - Triggers for each groups, trigger types, trigger sources for individual groups (up to 14 external trigger sources & 1 internal trigger source)

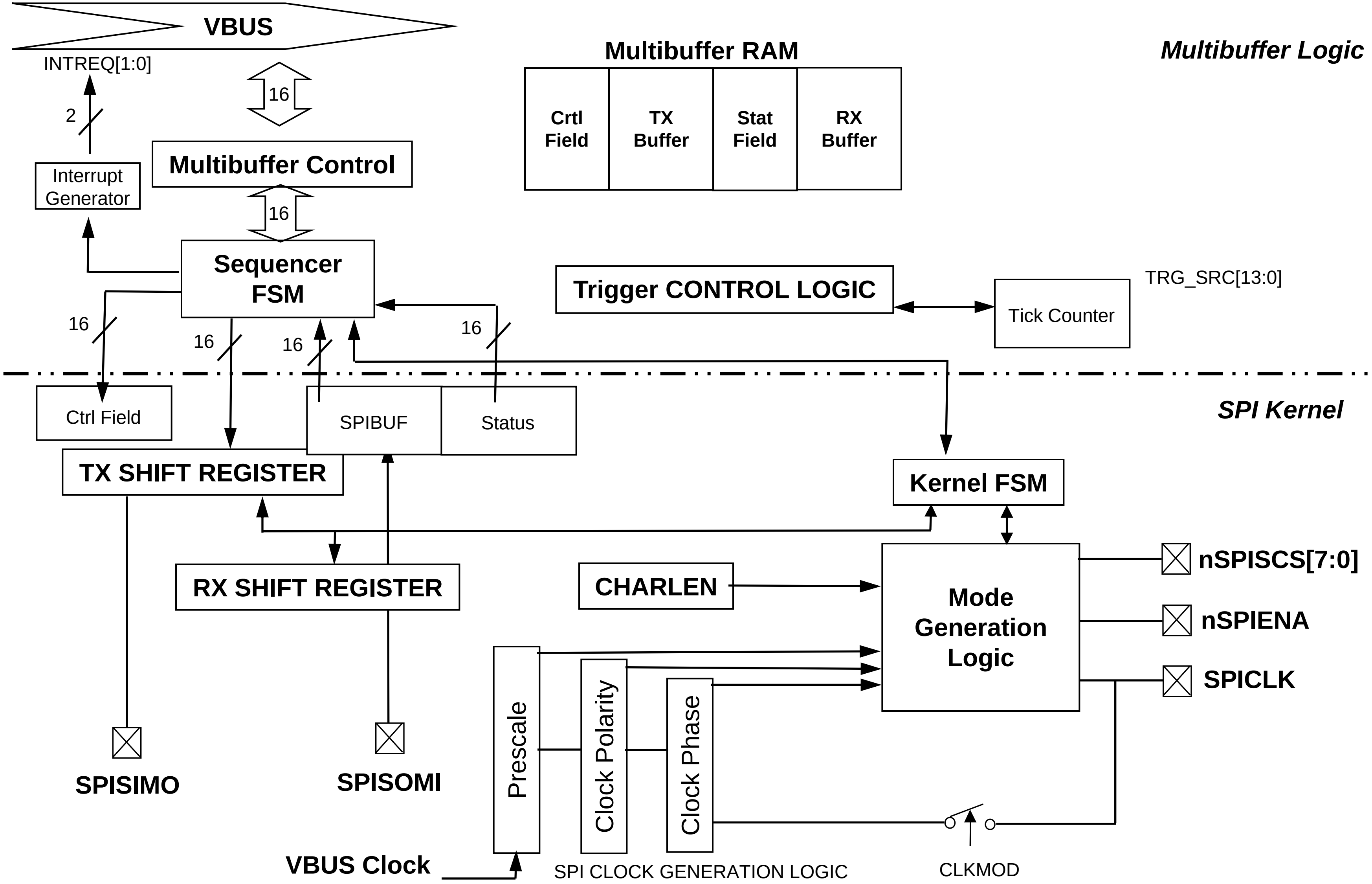
SPI / MibSPI Additional Features

- Multibuffer RAM Fault Detection (MibSPI only):
  - Parity circuit to detect memory faults
- SPI / MibSPI Multiple Chip Select (Master only):
  - The SPI / MibSPI supports multi chip select (multiCS) modes
- SPI / MibSPI Internal Loop-Back Test Mode (Master only):
  - To test the SPI / MibSPI transmit path and receive path including the buffers and the parity generator
- SPI / MibSPI Transmission continuous self-test (Master/Slave):
  - During data transfer SPI / MibSPI compares its own internal transmit data with its transmit data on the bus
- SPI / MibSPI Variable Chip Select Setup and Hold Timing (Master only):
  - To support slow slave devices a counter can be configured to delay the data transmission after the chip select is activated
- MibSPI Lock transmission (Multibuffer mode Master only):
  - To enable consecutive transfer without being interrupted by another higher priority group transfer
- SPI /MibSPI Detection of Slave de-synchronization (Master only)
- SPI / MibSPI ENA Signal Time-out (Master only):
  - To avoid stalling the MibSPI by a non-responsive slave device
- SPI/MibSPI Data Length Error:
  - To detect any mismatch in length of received/ transmitted data with the programmed character length
- Modulo Count Parallel Mode (Optional mode):
  - Special parallel mode to accommodate some specific slave devices

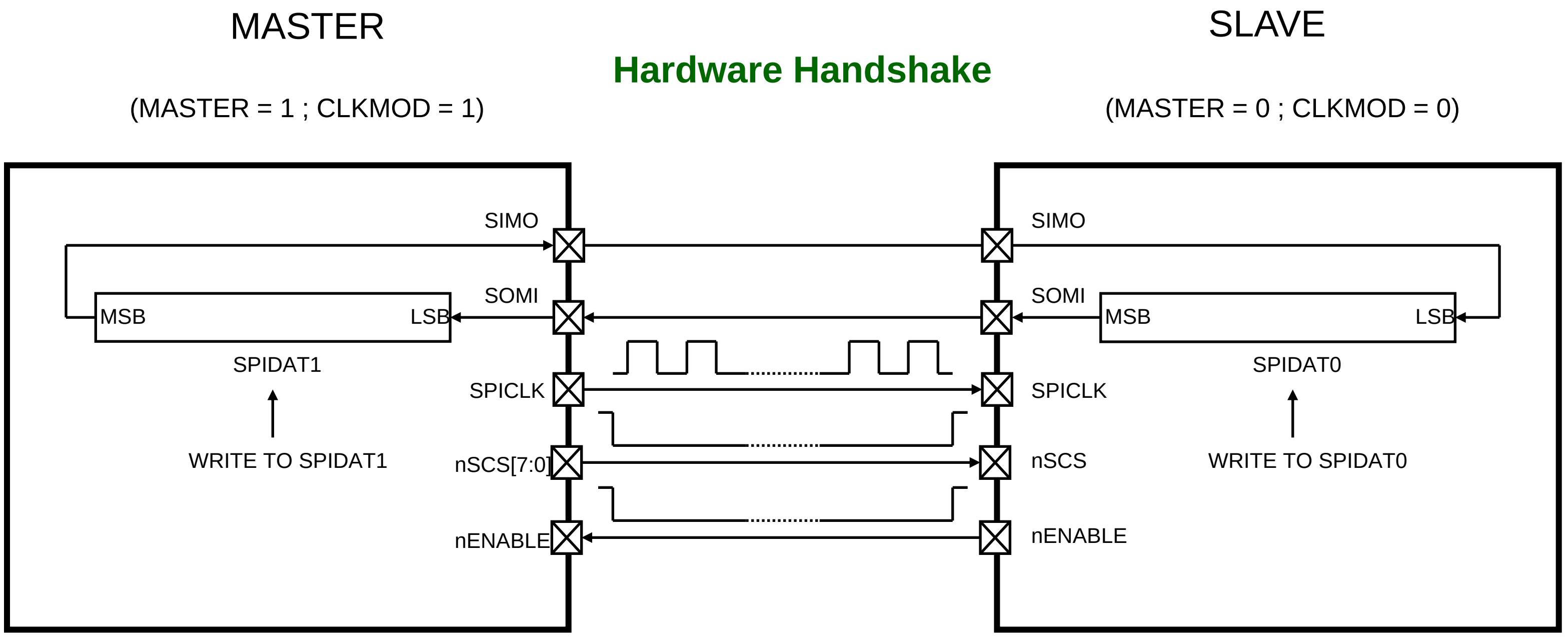
Standard SPI Mode - Block Diagram

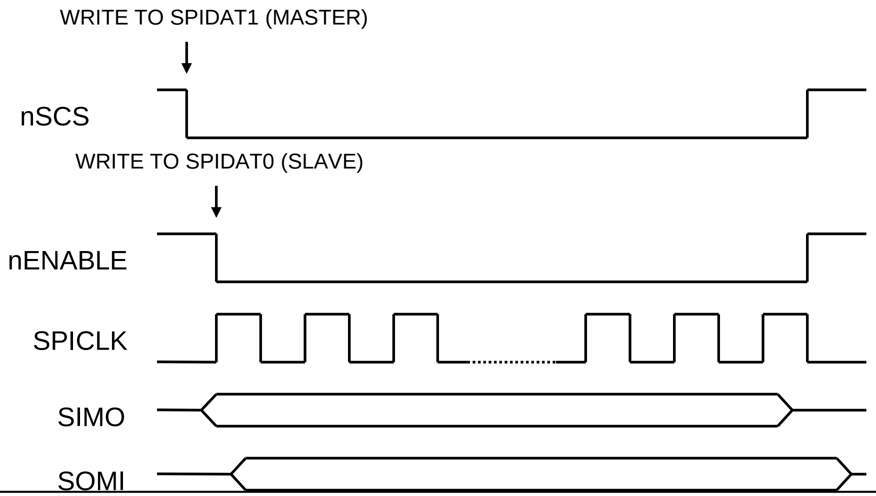


MibSPI Multibuffer Mode - Block Diagram



Transfer Mode - Five Pin Option:





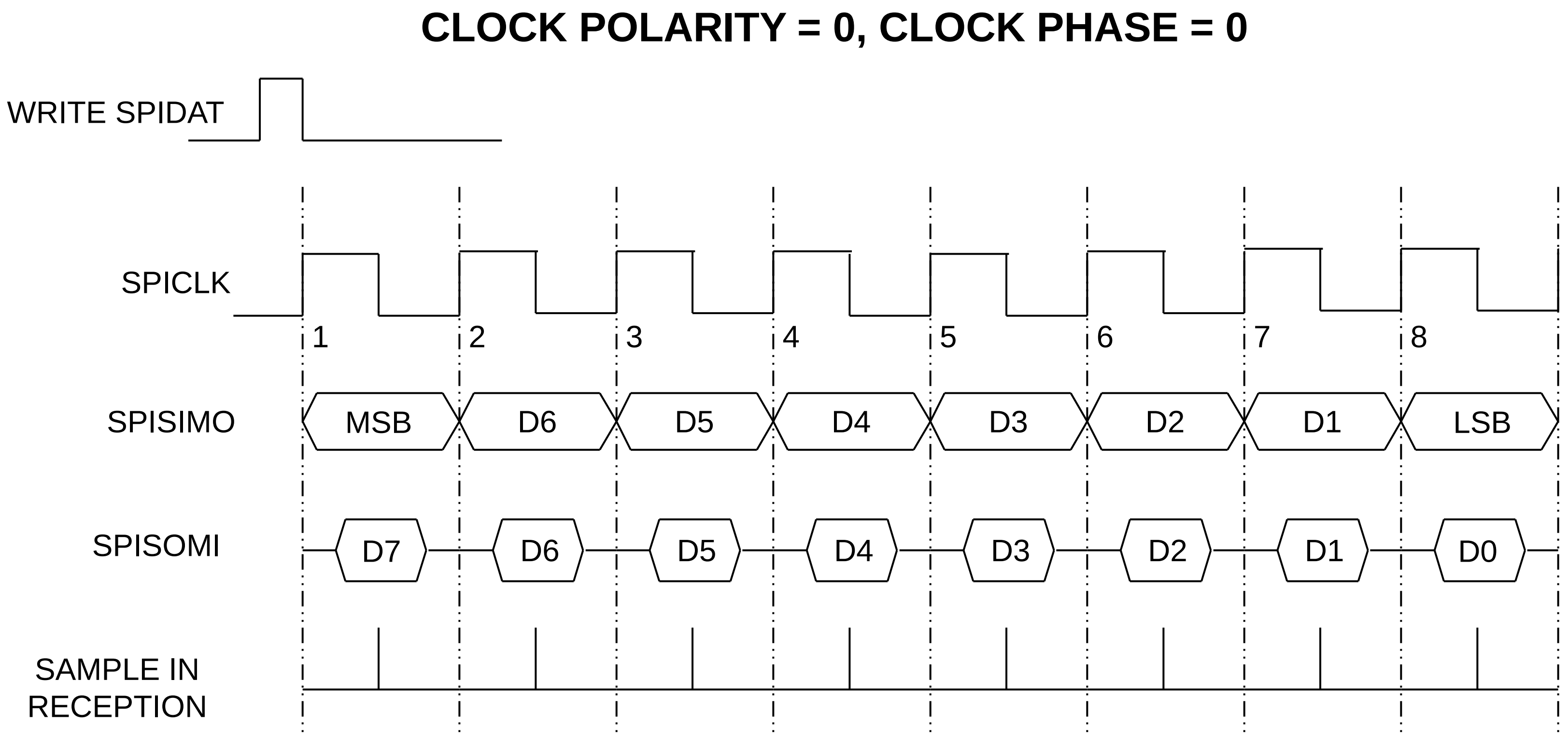
SPI / MibSPI Data Formats

- Data word length must be programmed in master mode and in slave mode
- For words with fewer than 16 bits Data must be right-justified when it is written to the MibSPI for transmission irrespective of its character length or word length
- The figure below shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer in order to be transmitted correctly:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	1	1	1	0	1	1	0	0	1	0	0	1

- The Received data is always stored right justified irrespective of the character length or direction of Shifting and is padded with leading ‘0’s when character length is less than 16.
- SPI / MibSPI supports automatic right-alignment of receive data independent from shift direction and data word length
- 2 consecutive accesses to the same slave are possible

Clock Options

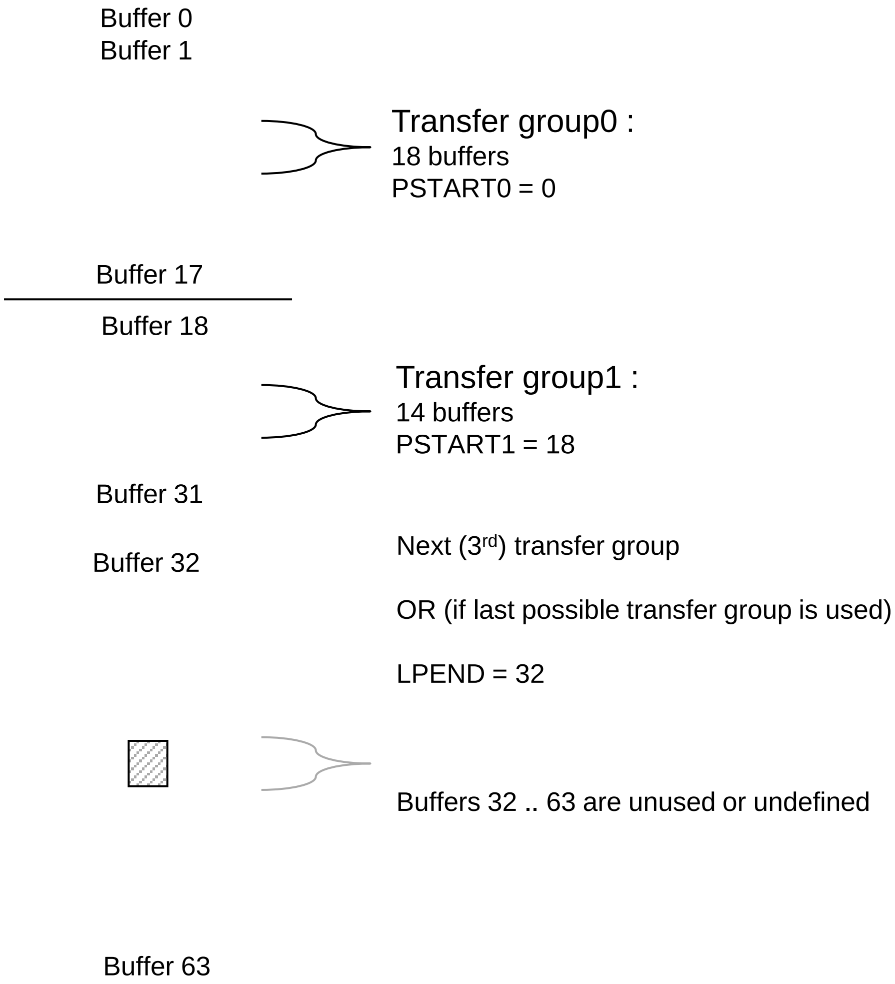


CLOCK PHASE = 0 (SPICLK WITHOUT DELAY)

- DATA IS OUTPUT ON THE RISING EDGE OF SPICLK
- INPUT DATA IS LATCHED ON THE FALLING EDGE OF SPICLK
- A WRITE TO THE SPIDAT REGISTER STARTS SPICLK

Multibuffer RAM

- Size depends on the implementation
  - Number of buffers: 0..63
- Divided into different groups with individual configuration
- For each group a trigger event can be chosen
  - One shot or continuous option
  - Trigger event conditions can be set up (e.g. rising edge)
- Up to 15 trigger sources are available
  - External events, tick counter, buffer array management
- Interrupt generation
  - Upon finishing transfer group
  - Suspend (provide new data or consume received data)



Buffer 0
Buffer 1

Transfer group0 :
18 buffers
PSTART0 = 0

Buffer 17

Buffer 18

Transfer group1 :
14 buffers
PSTART1 = 18

Buffer 31

Buffer 32

Next (3rd) transfer group

OR (if last possible transfer group is used)

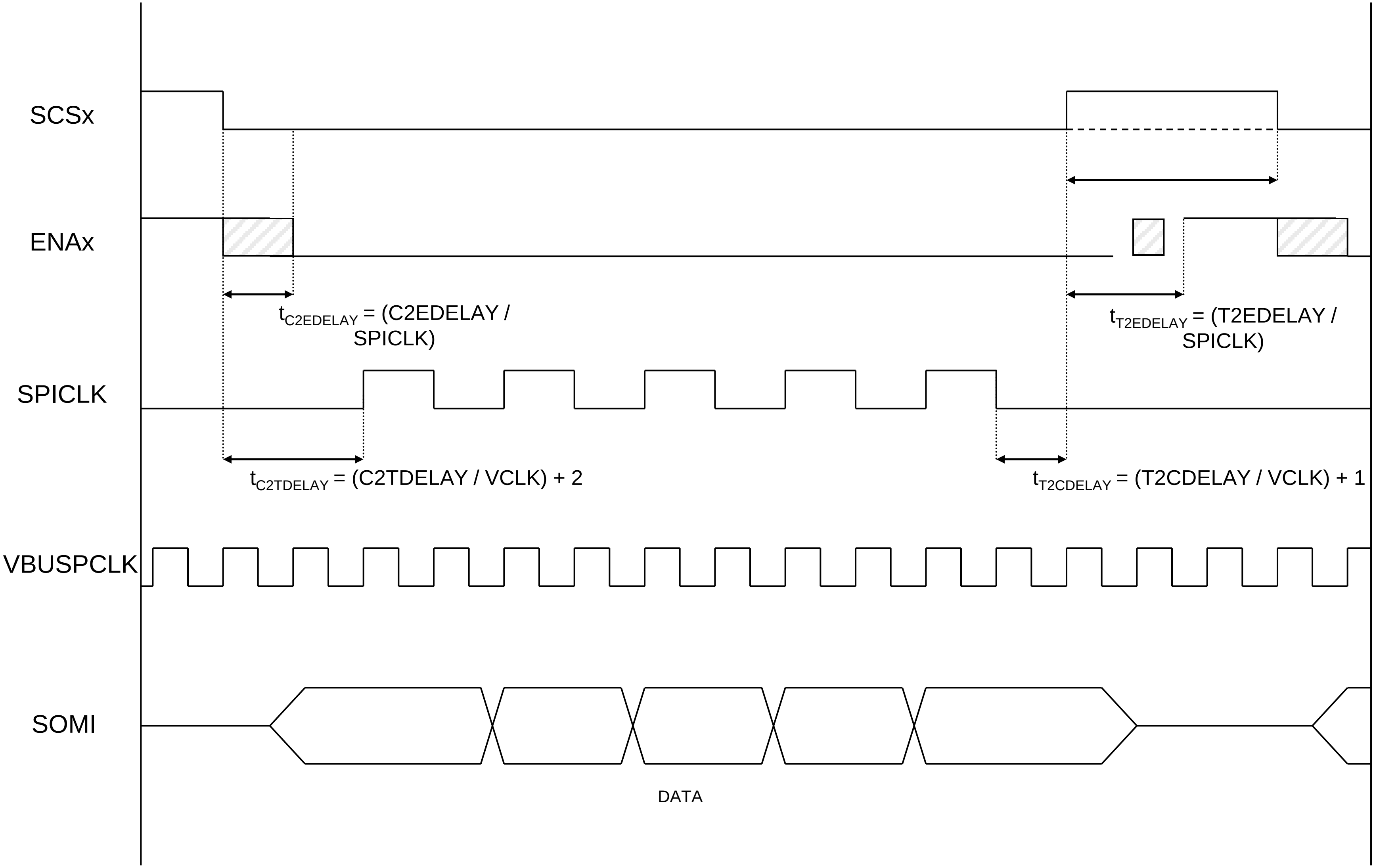
LPEND = 32

Buffers 32 .. 63 are unused or undefined

Buffer 63

Example for a 32 buffers dual transfer group (64buffer RAM size)

Timing Setup – Delay Register (SPIDELAY)



SCSx
ENAx
$t_{C2EDELAY}$ = (C2EDELAY / SPICLK)
$t_{T2EDELAY}$ = (T2EDELAY / SPICLK)
SPICLK
$t_{C2TDELAY}$ = (C2TDELAY / VCLK) + 2
$t_{T2CDELAY}$ = (T2CDELAY / VCLK) + 1
VBUSPCLK
SOMI
DATA