UNIVERSITY OF MICROCHIP
UofM
MASTERs 2007

11089 PS4

SMPS Basics Using the dsPIC® SMPS Digital Signal Controllers (DSCs)

UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

Class Objective

When you finish this class you will:

- Understand the Intelligent Power Peripherals on SMPS devices
- Learn how digital PID control loops are implemented
- Understand the functional blocks of SMPS firmware
- Understand Power-up Sequencing

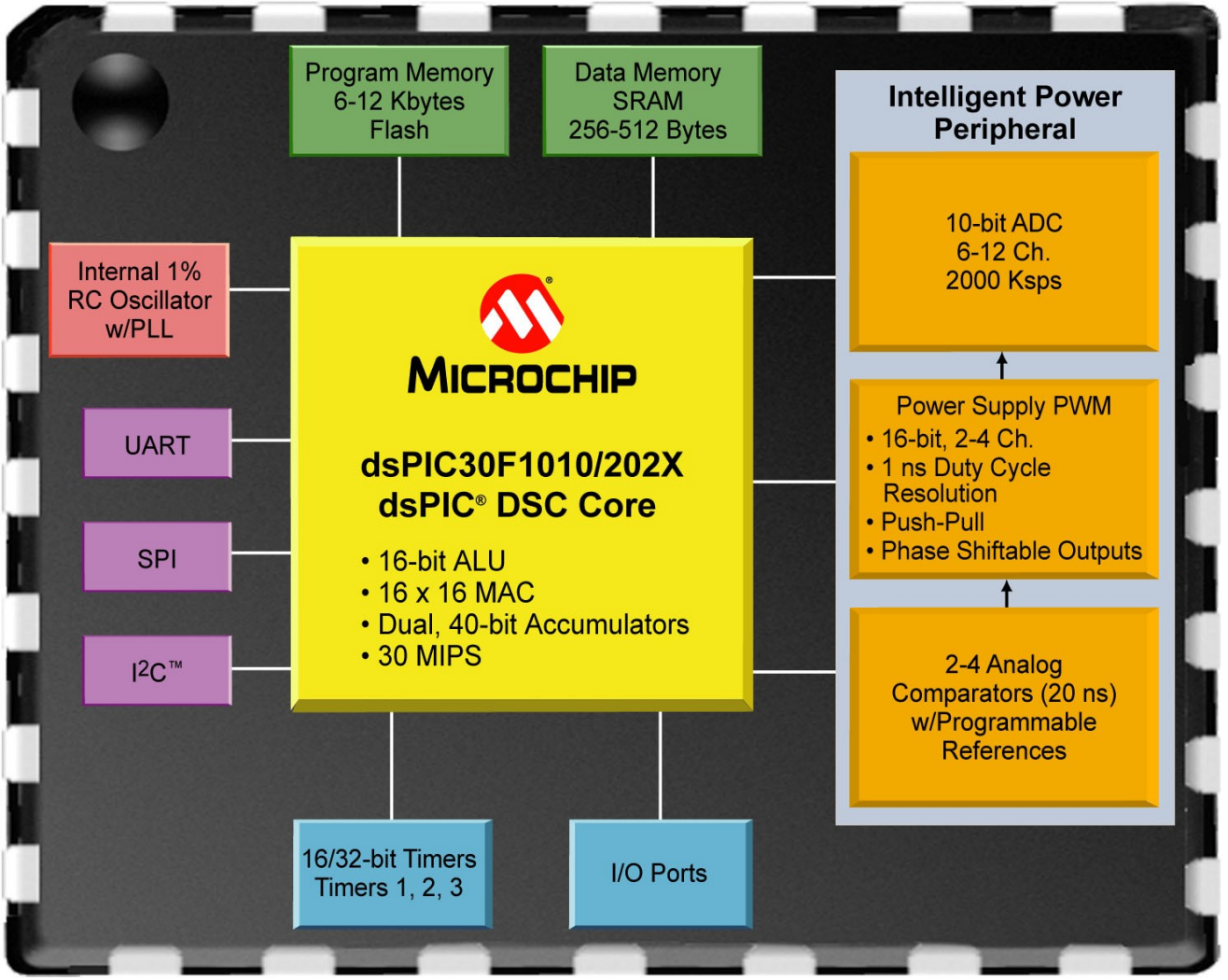
Agenda

- General Overview of the SMPS dsPIC30F Processors
  - IPP PWM: Architecture and Programming
  - Data Monitor and Control Interface
    - Lab 1
  - IPP ADC: Architecture and Programming
  - Faults and Current Limiting Features
    - Lab 2
  - Buck Demo Board: PID control loop and firmware
    - Lab 3
  - Sequencing

General Overview

University of Microchip UofM MASTERs 2007

The SMPS dsPIC® DSC Family



The SMPS dsPIC® DSC Family

- 30 MIPs MCU + DSP core
- High Speed A/D: 10-bit, 2 MSPS
- High Resolution PWM – 1.05 nsec resolution
- High Speed Analog Comparators
- Internal FRC + PLL
- Extended Temp (125ºC) Operation
- Flash based controller
- Small footprint package - 6 x 6 mm

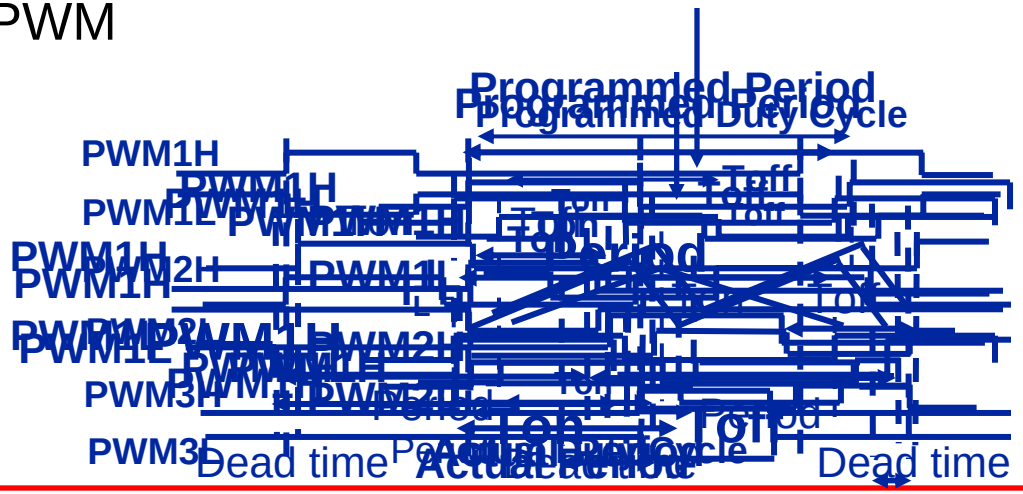
IPP PWM Architecture and Programming

IPP PWM

- High resolution PWM: 12 bits at 234 kHz
  - 1.05 ns duty cycle resolution
- Individual deadtime values for each PWM
  - Positive and negative deadtime
  - 4.4 nsec resolution
- Extensive ADC triggering options
- Four PWM generators with eight I/O and Four independent time bases
  - Duty cycle resolution of 1.05 nsec @ 30 MIPS
  - Dead-time resolution of 4.2 nsec @ 30 MIPS
  - Frequency resolution of 8.4 nsec @ 30 MIPS

IPP PWM

- Supported PWM modes:
  - Standard Edge-Aligned PWM
  - Complementary PWM
  - Push-Pull PWM
  - Multi-Phase PWM
  - Variable Phase PWM
  - Fixed Off-Time PWM
  - Current Reset PWM
  - Current-Limit PWM
  - Independent Time Base PWM

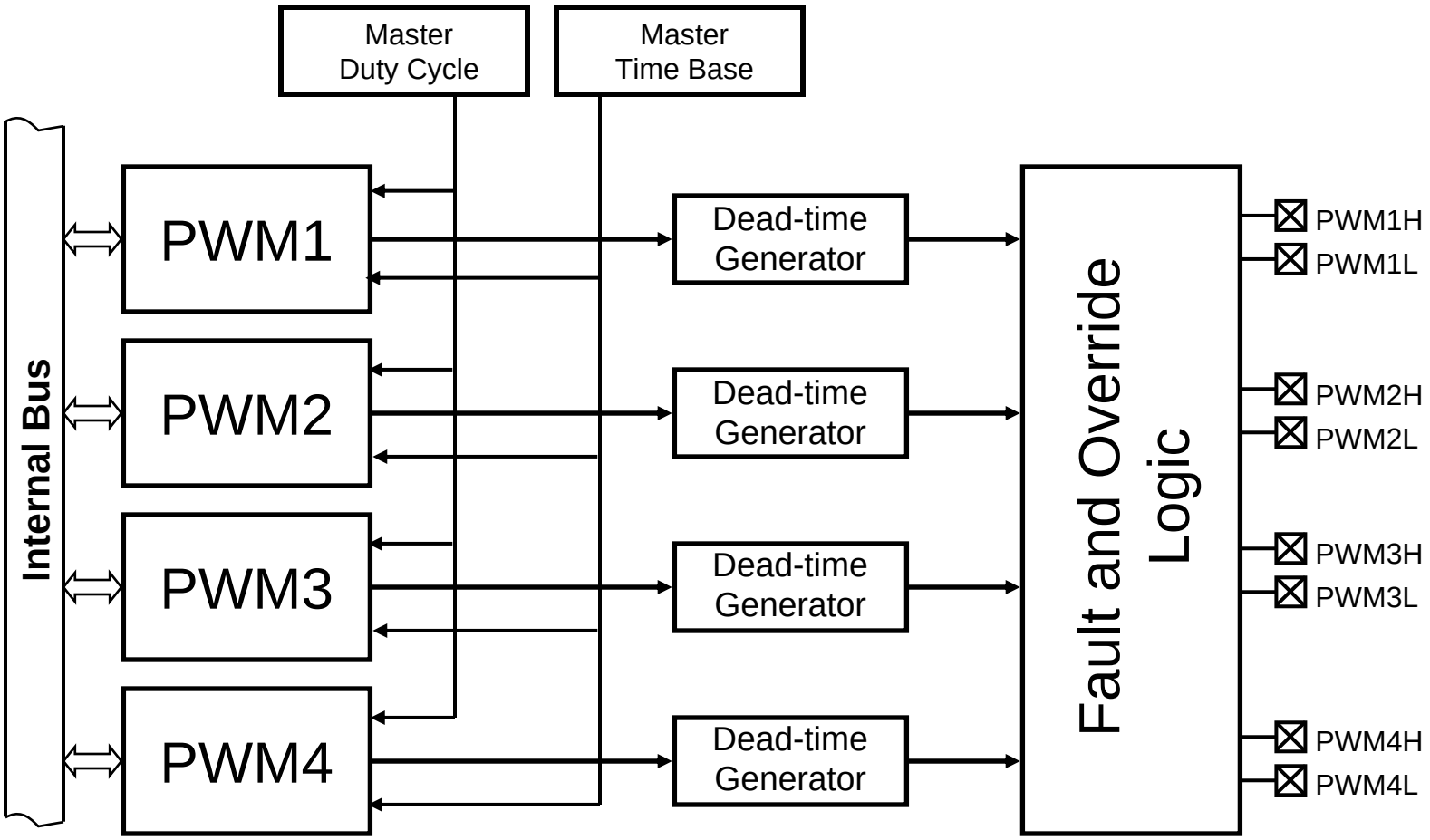


IPP PWM

- On-the-Fly changes to:
  - PWM frequency
  - PWM duty cycle
  - PWM phase shift
- Output override control
- Independent current-limit and fault inputs
- Special event triggers

IPP PWM

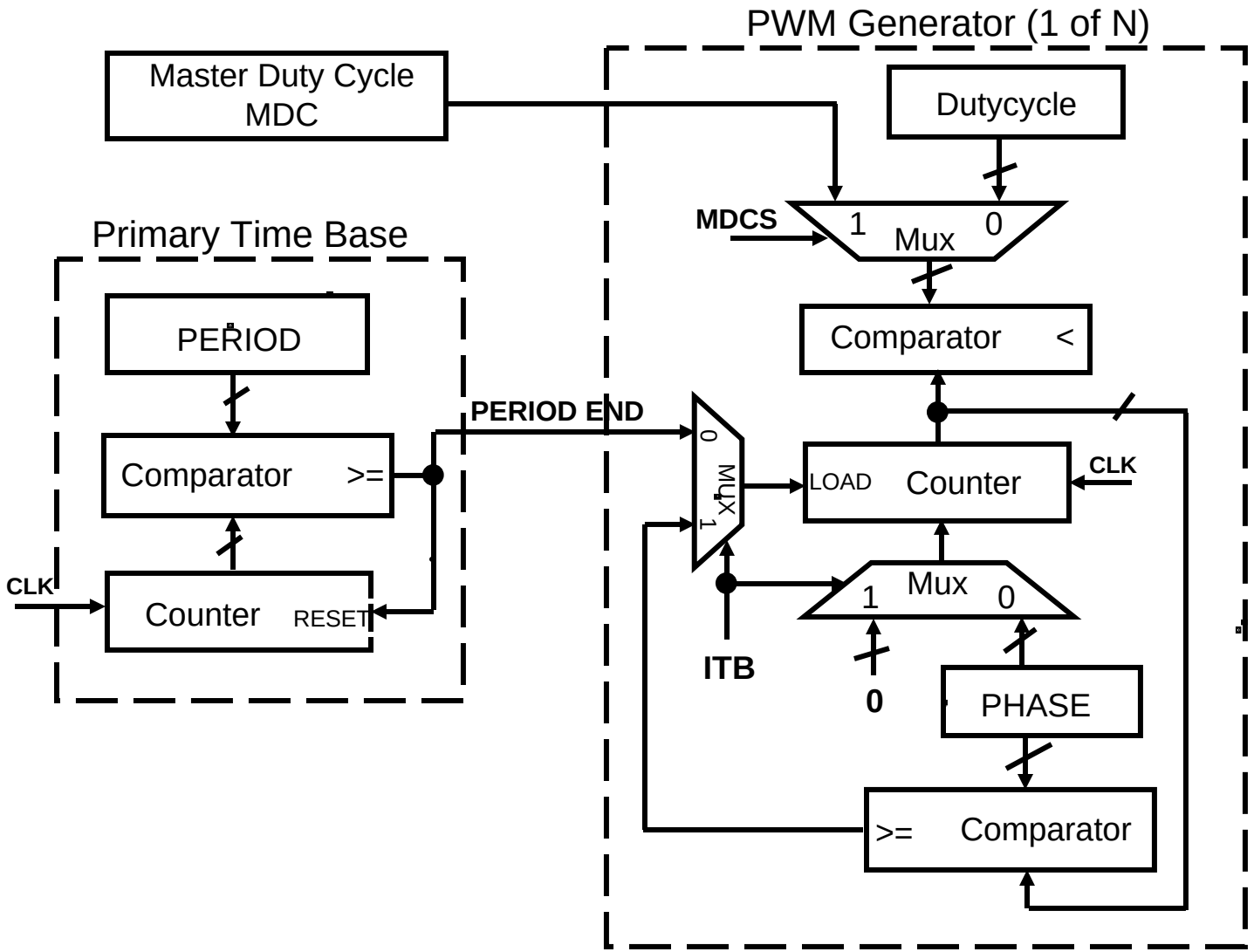
- **Basic PWM Architecture**



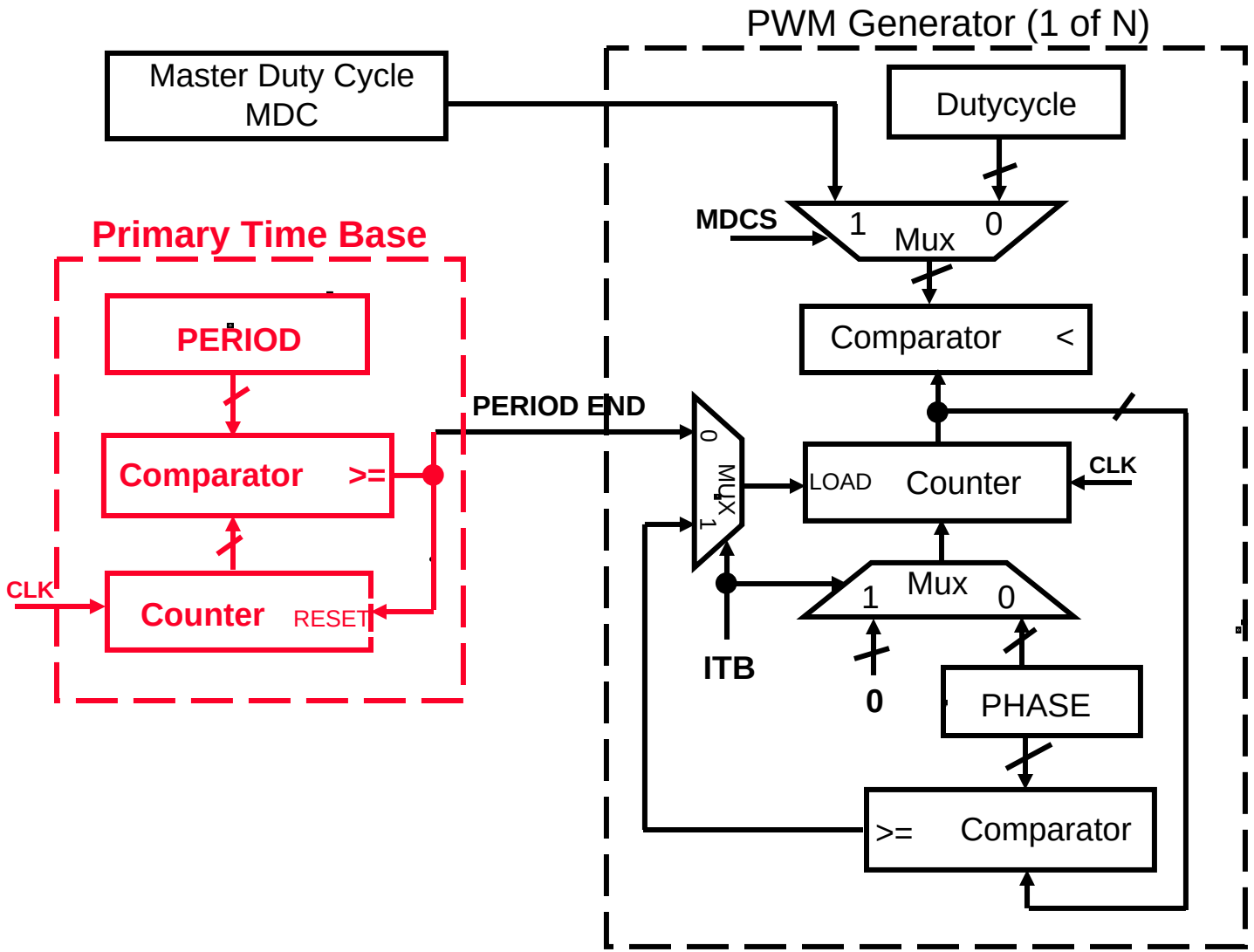
IPP PWM

- PWM Frequency Generation
  - Primary Time Base (PTMR) counter
  - Individual Time Base Counter (PHASEx)
- PWM Duty Cycle Generation
  - Master Duty Cycle (MDC) Register
  - Individual Duty Cycle Register

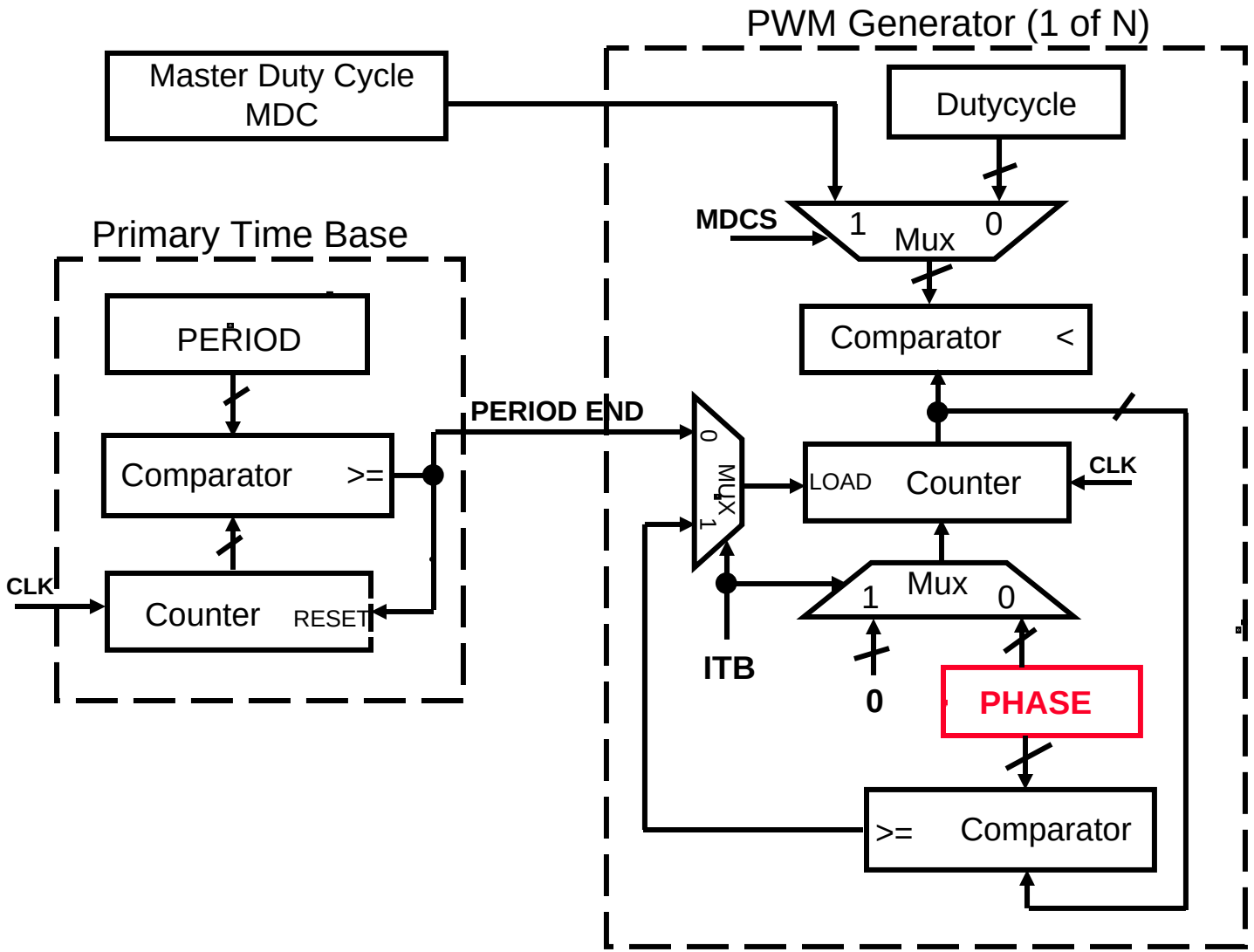
IPP PWM



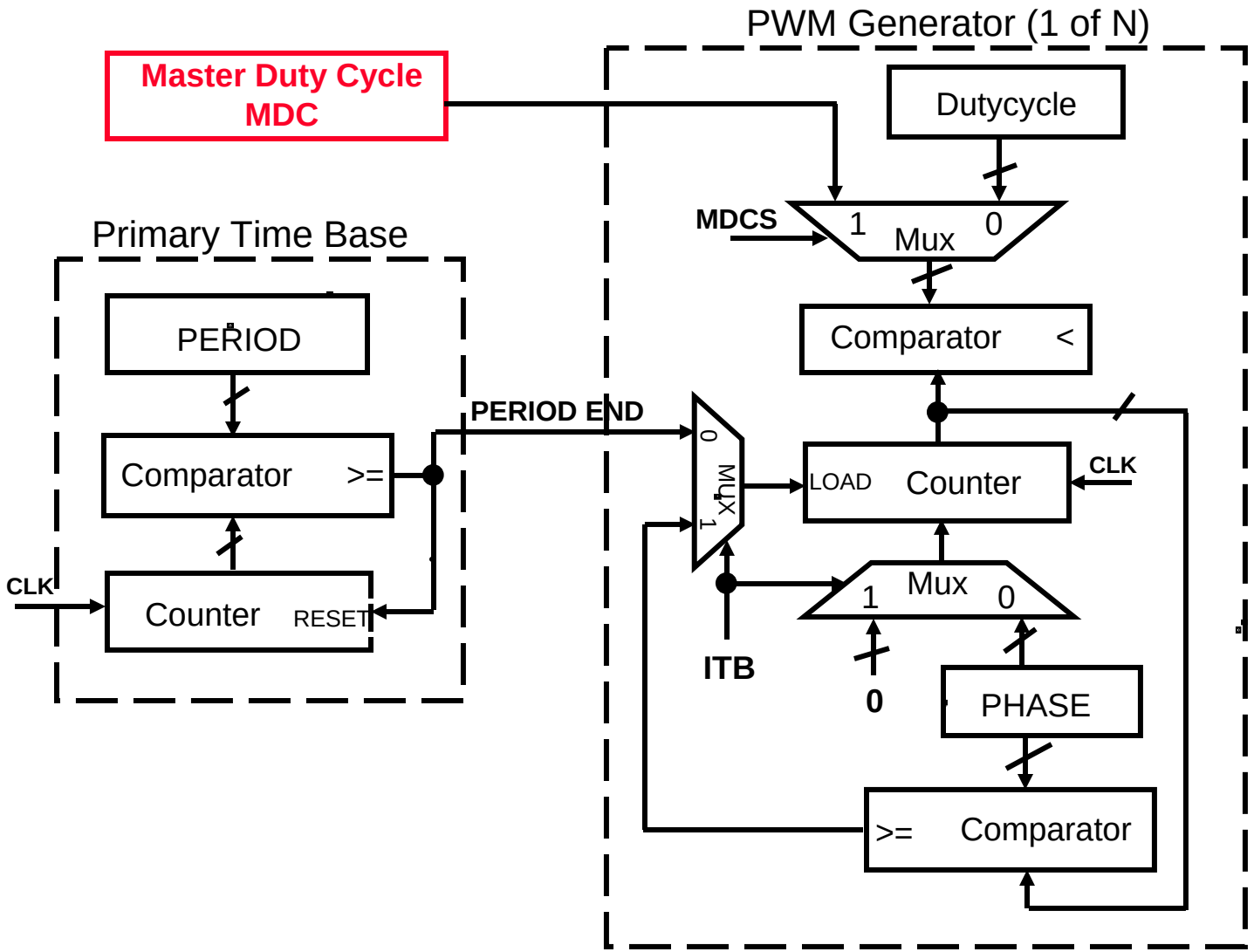
IPP PWM



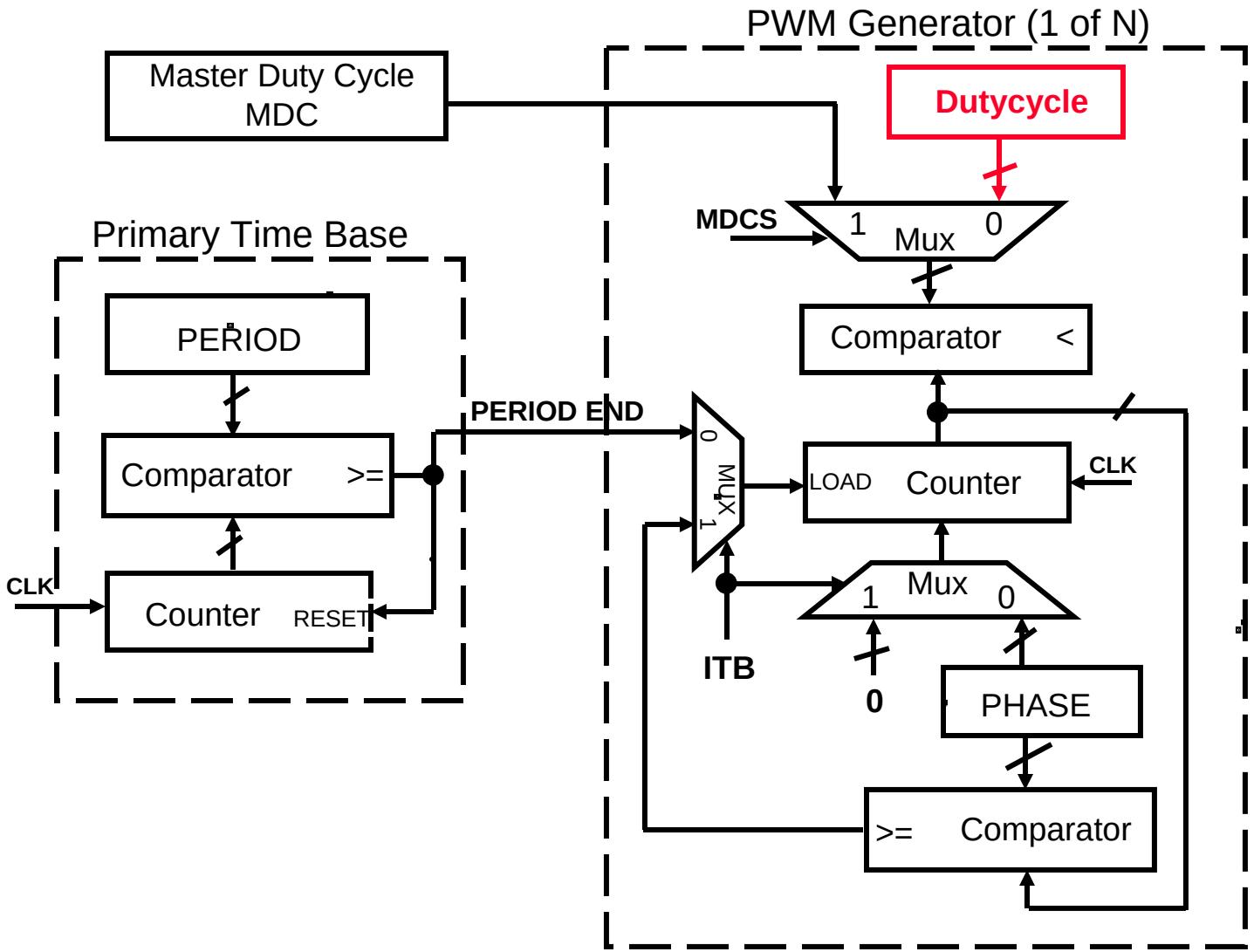
IPP PWM



IPP PWM



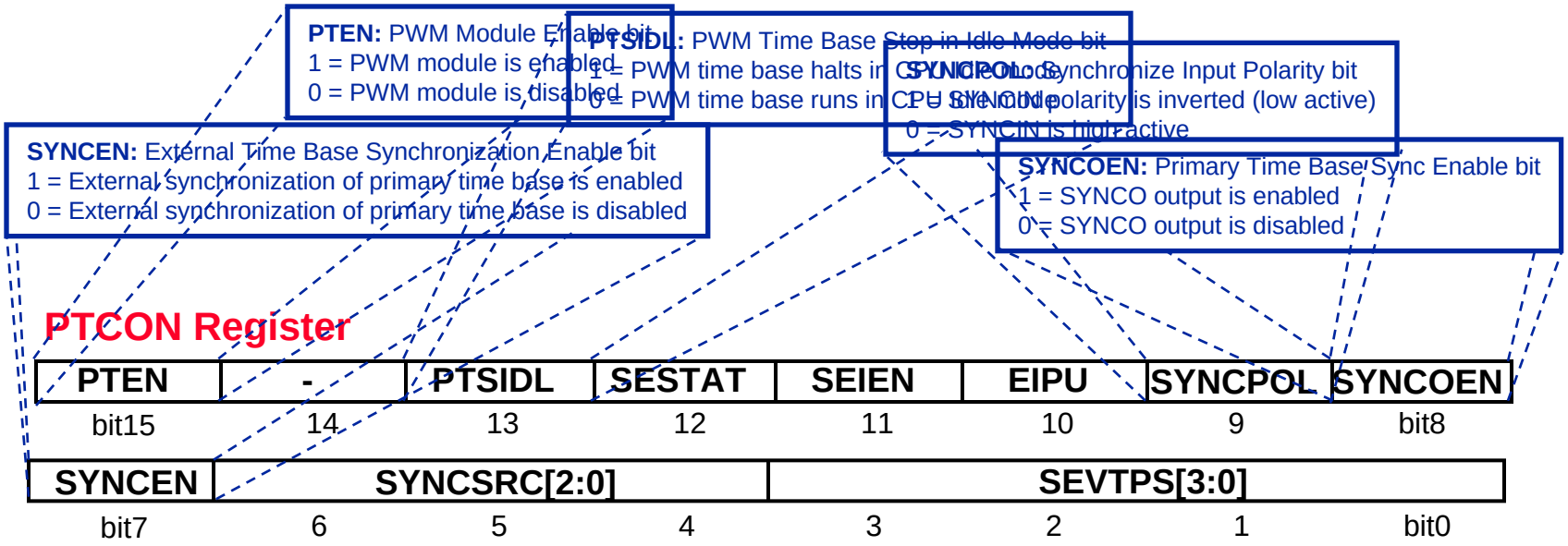
IPP PWM



IPP PWM

The PTCON Register: controls the general operation of the peripheral

- Enables the module
- Controls IDLE behavior
- Controls external synchronization features

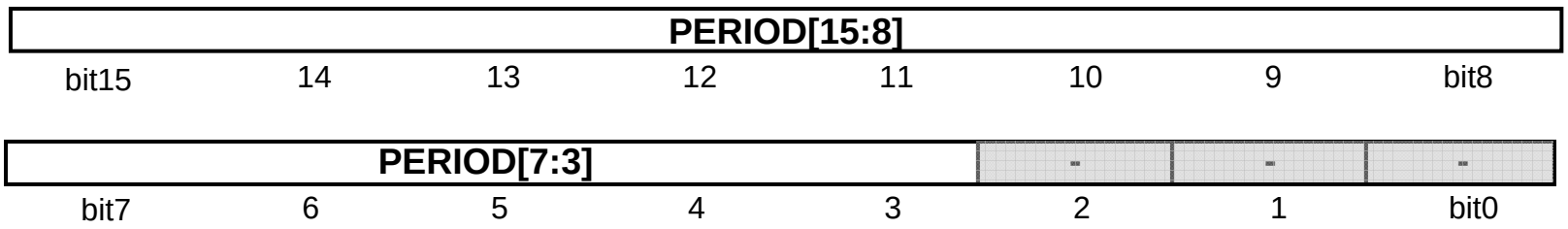


IPP PWM

The PTPER register defines the primary time base period for the PWM generators

- The least significant 3 bits are not implemented
- Bit 3 (LS implemented bit) represents 8.4ns @ 30 MIPS

PTPER Register



IPP PWM

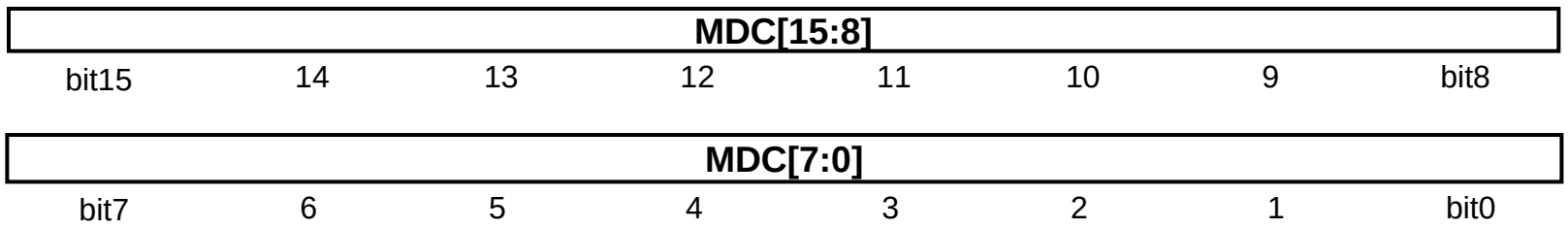
The Master Duty Cycle register defines the Duty Cycle common to all modules

- May be used by any PWM as a source for the duty cycle

Duty Cycle Computation

To get a 50% Duty Cycle: MDC = PTPER >> 1

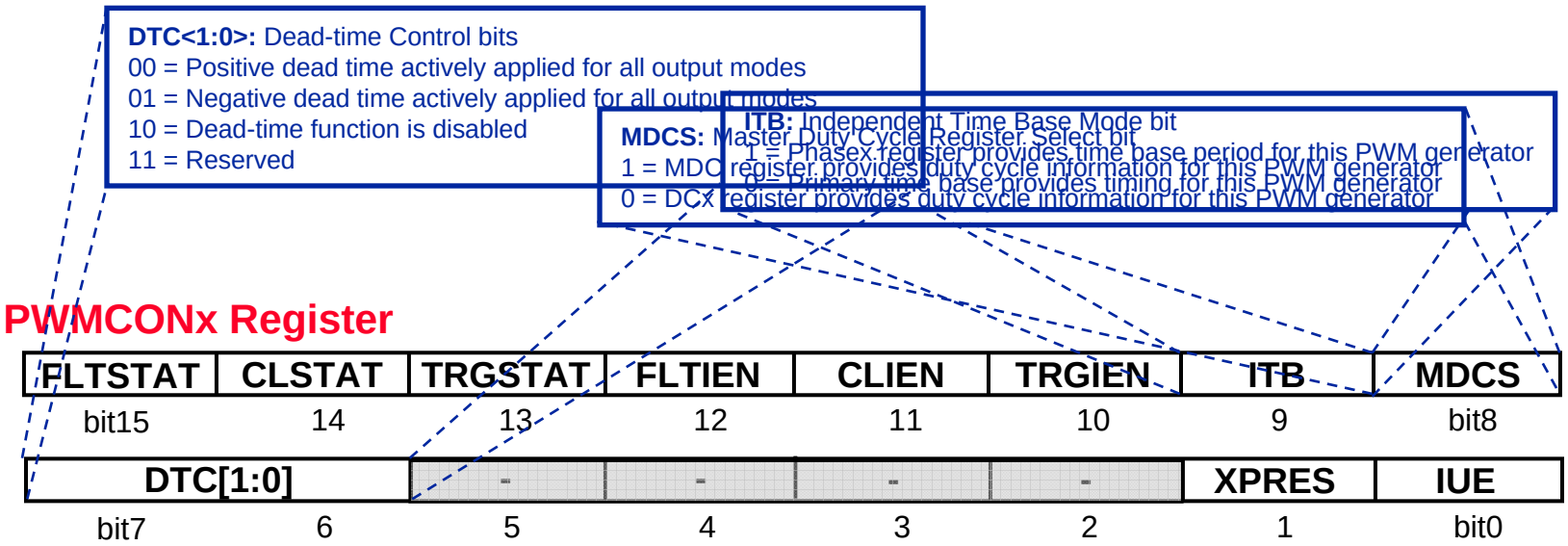
MDC Register



IPP PWM

The PWMCONx register controls the general operation of each PWM module

- Enables independent time base mode
- Selects the source of the duty cycle
- Defines the operation of the deadtime logic



IPP PWM

The PDCx register sets the Duty Cycle for each PWM module

- The dutycycle resolution is 1.05 nsec @30 MIPS
- The dutycycle value may be updated at anytime during the PWM cycle, if the IUE bit is set in the PWMCONx register

PDCx Register

PDC[15:8]							
bit15	14	13	12	11	10	9	bit8

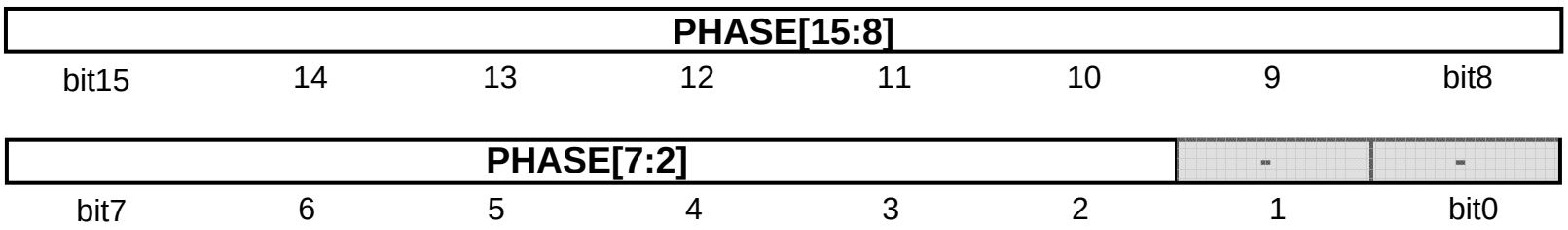
PDC[7:0]							
bit7	6	5	4	3	2	1	bit0

IPP PWM

The PHASEx register:

- Each PWM generator has its own PHASE register
- It can be used either as a phase-shift value or as the independent Time Base Period
- The phase resolution is 4 nsec @30 MIPS
- The value moves the PWM signal “earlier” in time
- The phase value may be updated at anytime during the PWM cycle; the new phase value takes effect at the end of the PWM cycle

PHASEx Register



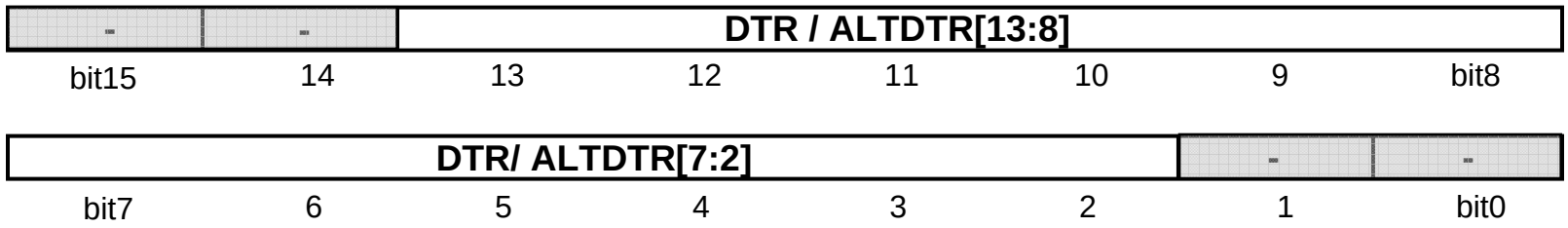
IPP PWM

The DTRx/ALTDTRx registers:

- Each PWM generator has its own Dead Time Registers for rising and falling PWM edges
- The dead time resolution is 4.2 nsec @30 MIPS

MIPS	Resolution	Dead-time Range
30	4.16 ns	0-17.03 μs
20	6.25 ns	0-25.59 μs

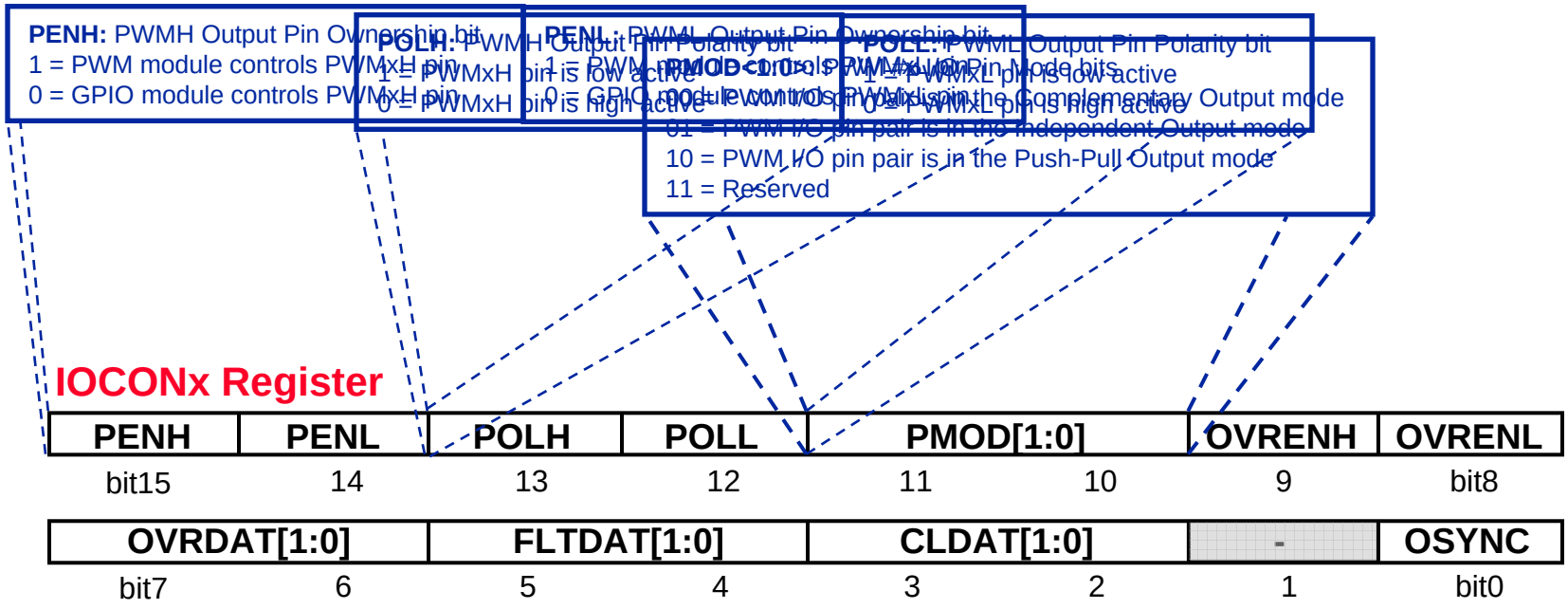
DTRx / ALTDTRx Register



IPP PWM

The IOCONx register is the I/O Control register

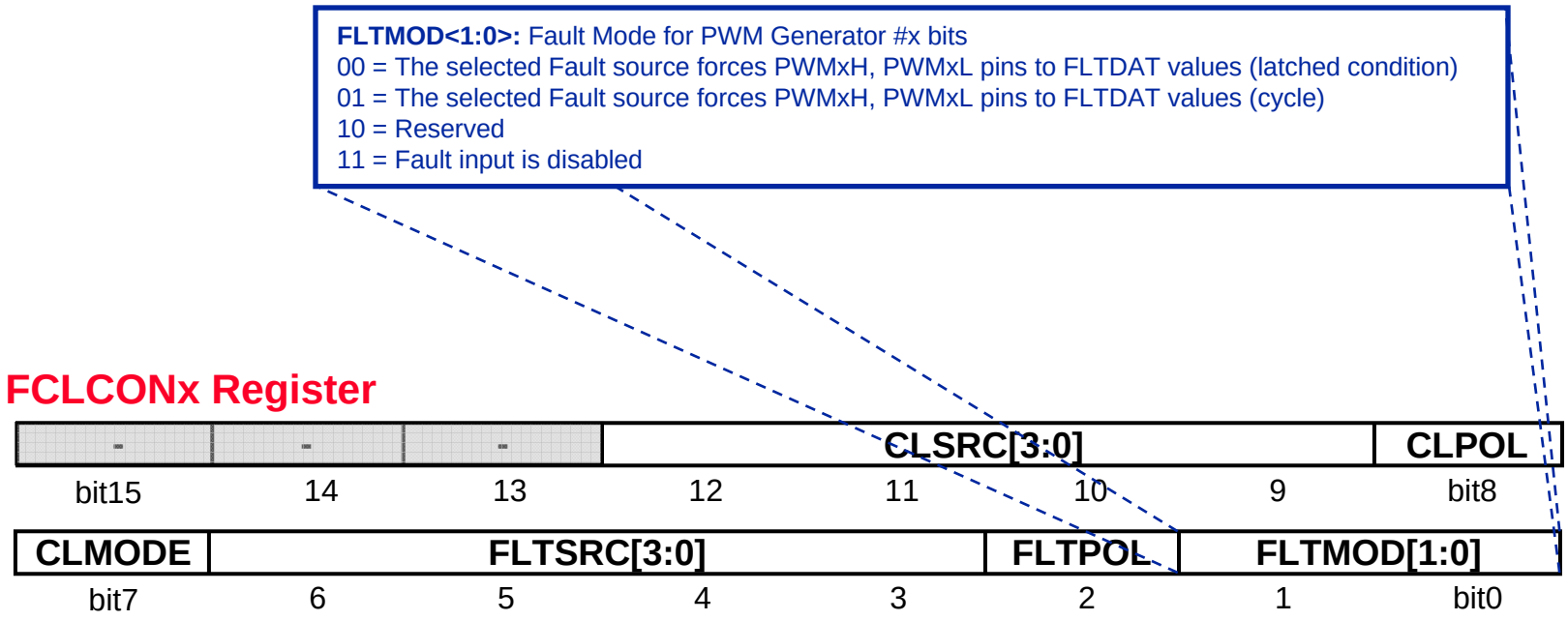
- Controls the ownership of the PWM pins
- Controls the polarity of the PWM pins
- Controls the pwm output mode



IPP PWM

The FCLCONx register controls the Fault and Current Limit Operations

- Enables/disables the Fault operation



Oscillator System

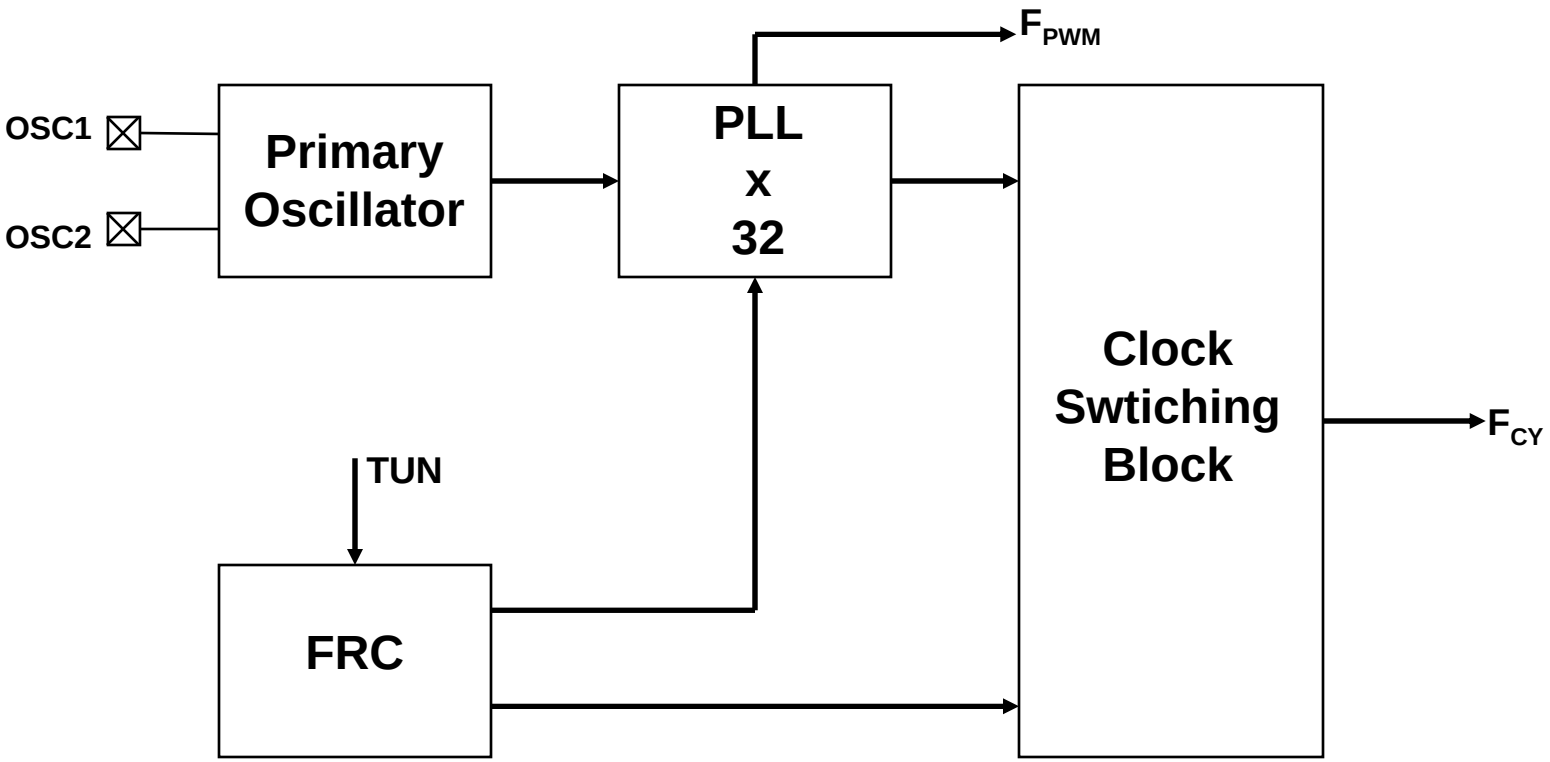
Oscillator System

Oscillator System

- Dual Internal RC
  - 9.7 and 14.55 MHz Industrial Temp
  - 6.4 and 9.7 MHz Extended Temp
- 32x PLL with 480 MHz VCO
- External Source
  - EC clock 6.0 to 14.55 MHz
  - HS Crystal mode 6.0 to 14.55 MHz

IPP PWM

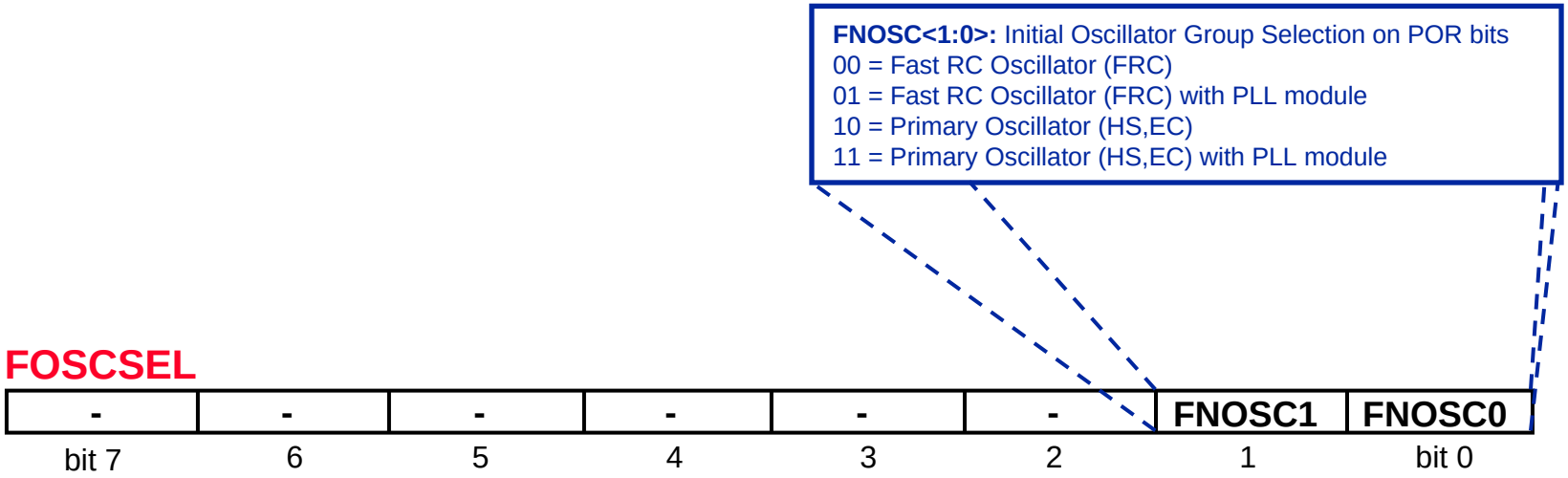
Simplified Clock Block Diagram



Oscillator System

FOSCSEL: Oscillator Selection Configuration Bits

- Oscillator Selection



Oscillator System

FOSC: Osillator Selection Configuration Bits

- Available Frequency Ranges

FRANGE	Temp. Range	FRC Frequency (Nominal)	PLL VCO (Nominal)
1 = High Range	Industrial Extended	14.55 MHz 9.7 MHz	466 MHz (480 MHz max) 310 MHz (320 MHz max)
0 = Low Range	Industrial Extended	9.7 MHz 6.4 MHz	310 MHz (320 MHz max) 205 MHz (211 MHz max)

FOSC

FCKSM [1:0]		FRANGE	-	-	OSCIOFNC	POSCMD [1:0]	
bit 7	6	5	4	3	2	1	bit 0

FOSCSEL

-	-	-	-	-	-	FNOSC1	FNOSC0
bit 7	6	5	4	3	2	1	bit 0

IPP PWM

- Summary. We have seen:
  - The basic Architecture of the IPP PWM
  - The registers used to program the PWM functionalities
  - The clock generation system

DMCI

DMCI

- New Graphical Tool in MPLAB® IDE
- Allows us to:
  - Input variable values
  - Display results graphically (up to four plots)

- The DMCI provides dynamic user configuration of:
  - 9 slider controls
  - 9 boolean (on/off) controls
  - 35 input controls (7 groups of 5)
  - 4 graphs

DMCI

- A typical series of operational steps with the interface would be as follows:
  - Compile the application software
  - Set up the interface as desired
  - Run the application
  - Halt the application using the control button at the bottom of the tuning interface; the data will upload into the interface

DMCI

- A typical series of operational steps with the interface would be as follows:
  - If you are not satisfied with the system performance, you can change the various values on the Dynamic Data Control/Input tabs of the interface; once you are satisfied with the values, click the Start button on the dialog to update parameter values and resume the code execution
  - Repeat the previous steps as required
  - Once the application operation is satisfactory, you can use the Save button in the interface to save the desired parameter values to a file

DMCI

- Buttons
  - Run/Halt: "Run“ write all values currently displayed on the Dynamic Data Control tab to the target RAM and continue code execution

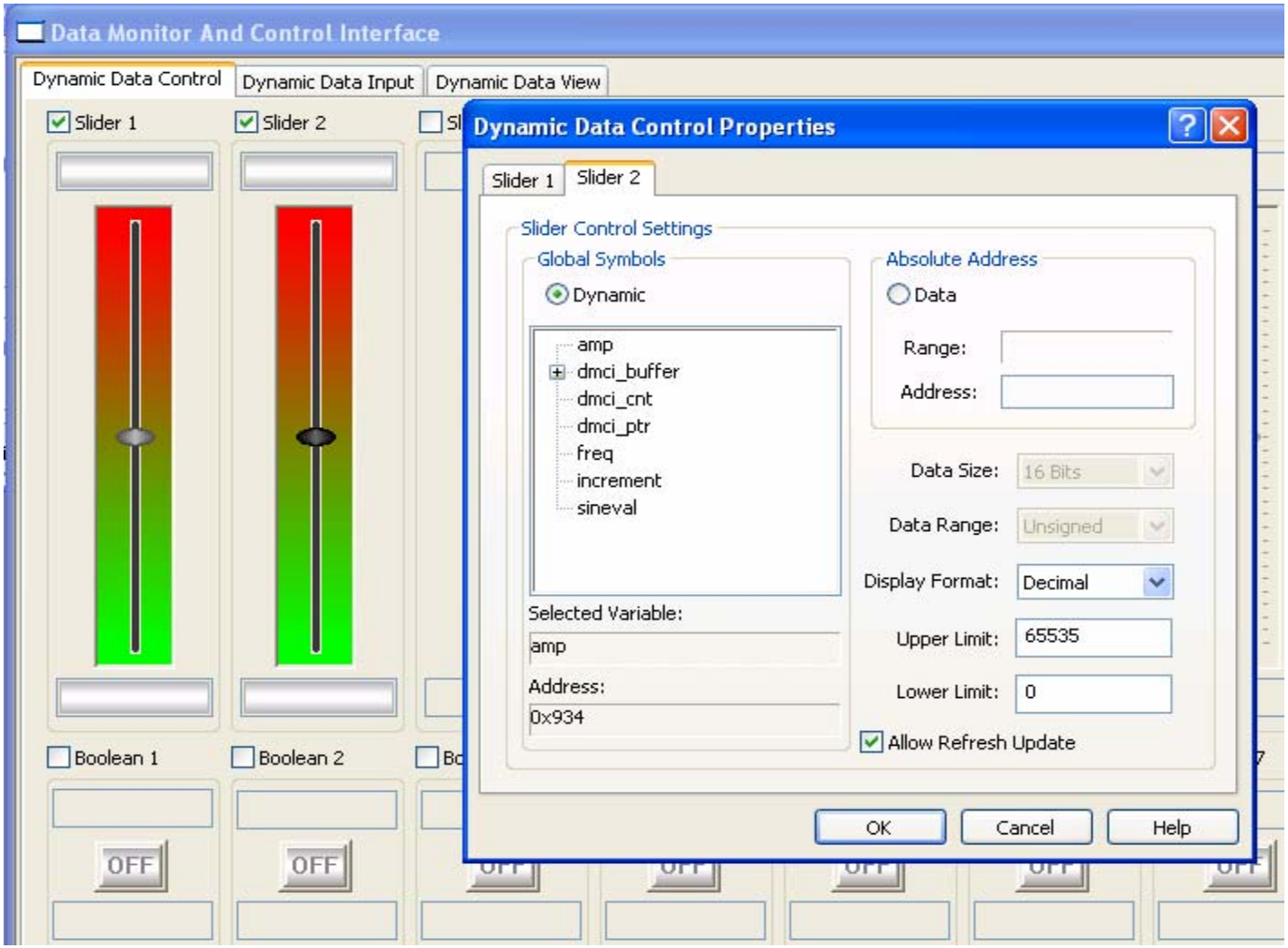
    "Halt" - Halt code execution, read all values from the target RAM and reflect those values on the dialog tabs
  - Save: Save the dialog settings to a *.dmci file.
  - Load: Load the dialog settings from a *.dmci file.
  - Refresh: Read all values from the target device RAM and reflect those values on the dialog tabs.

DMCI

- Buttons
  - Exit: Exit from the interface dialog. Note: The dialog always opens with default values. To save your settings, click Save. Then load your settings after you reopen the dialog at another time.
  - Help: On-line help information

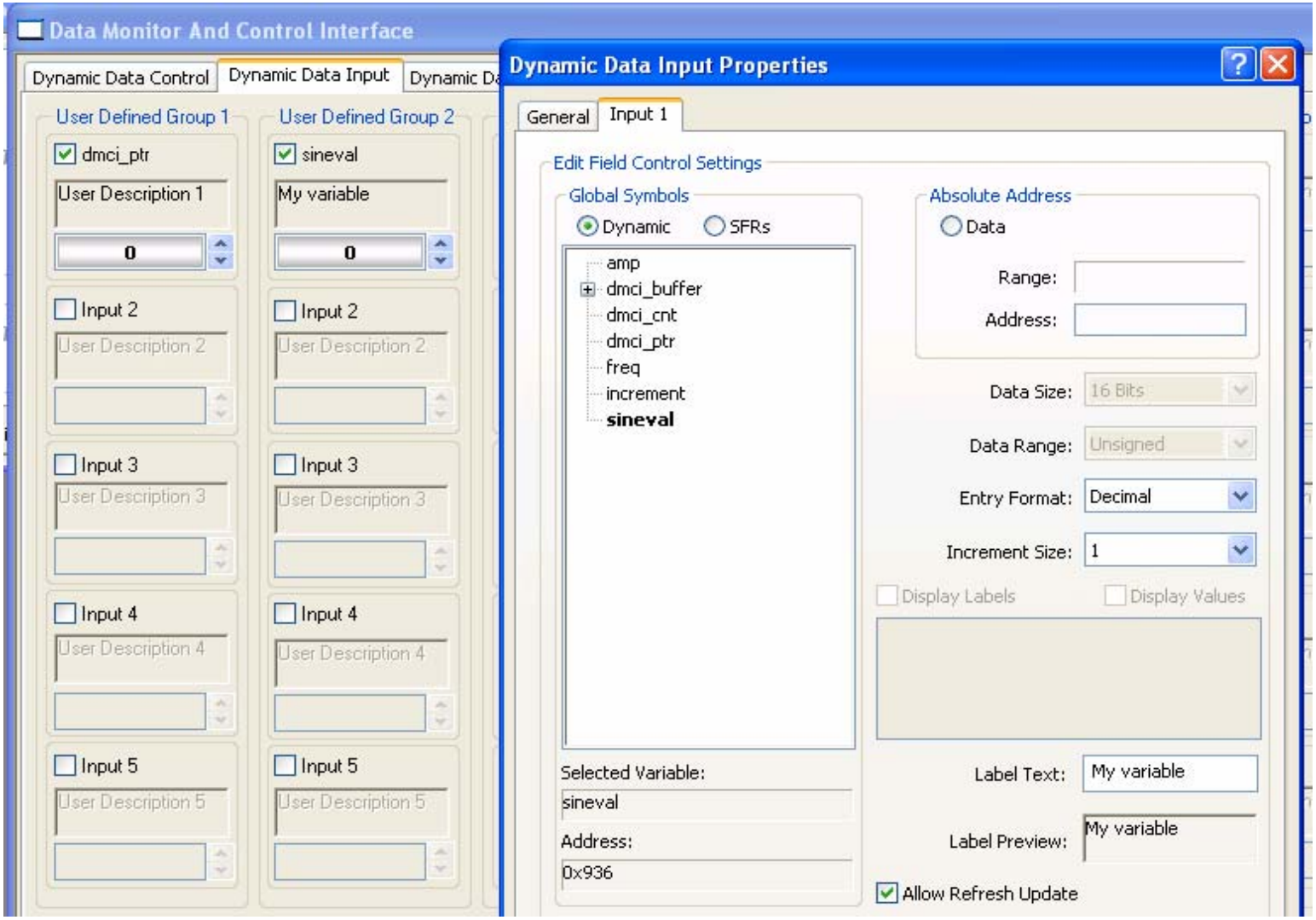
DMCI

Dynamic Data Control



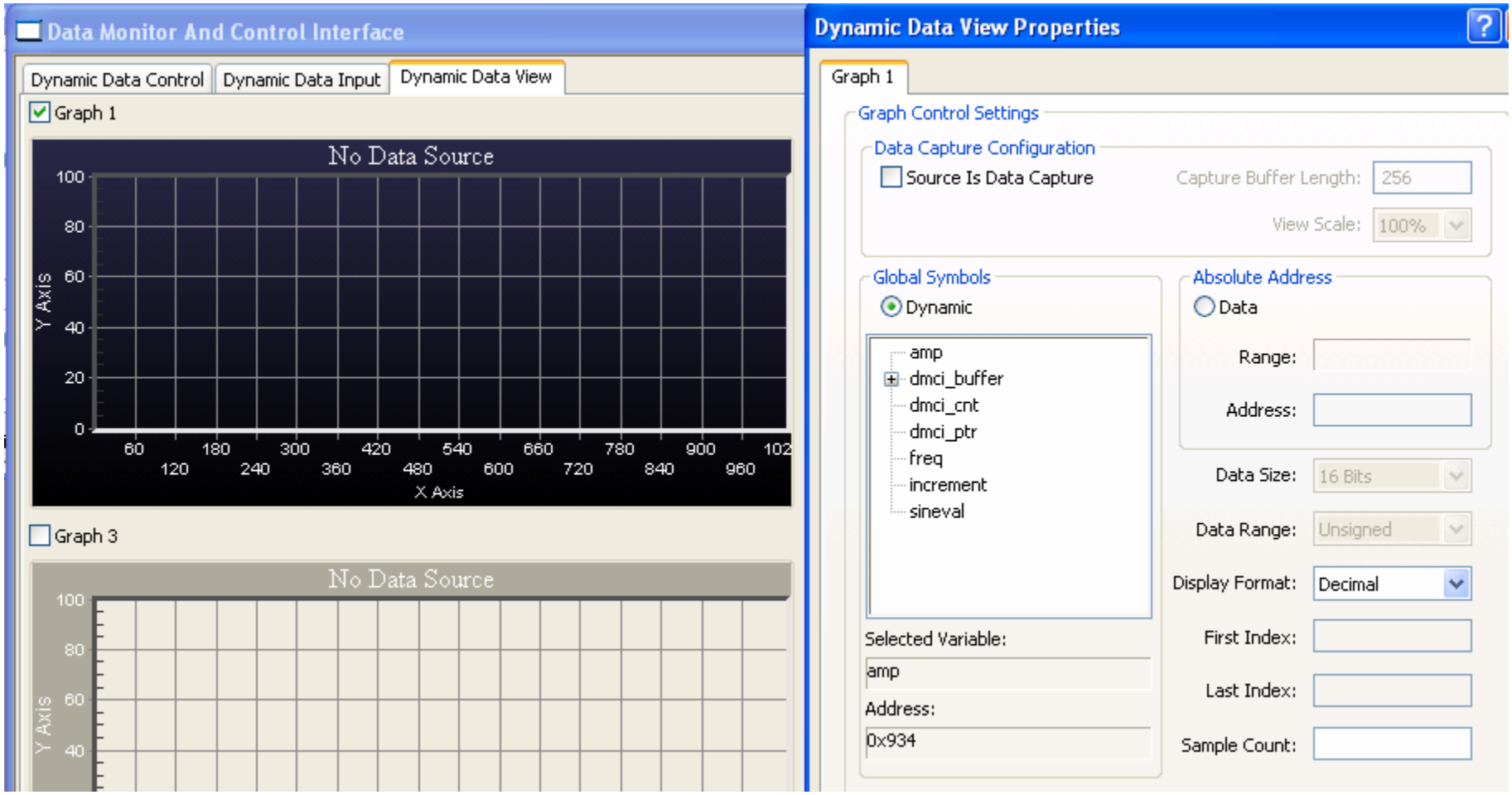
DMCI

Dynamic Data Input



DMCI

Dynamic Data View



DMCI

- Summary. We have seen:
  - What the DMCI is
  - How we can use it to change parameter values
  - How we can use it to monitor the behavior of our system

Lab 1

Lab 1

- Lab 1 is intended to:
  - Review the IPP PWM achitecture
  - Initialize the PWM peripheral
  - Use the DMCI

Lab 1

- In this Lab, you will:
  - Part 1
    - Initialize PWM 2
    - Initialize PWM 3
  - Part 2
    - Experiment with the DMCI Interface to change the features of the duty cycle of PWM 1

PWM1 is already initialized

Lab 1

- **Solution**

```
// ===================================================================
//              PWM 2
//              Independent time base
//              Independent duty cycle
//              Frequency 150 KHz
//              Complimentary output
//              Duty cycle = 25 %
//              Dead band of 50 nsec rising edge
//              Dead band of 100 nsec falling edge
// -------------------------------------------------------------------

PWMCON2 = ( unsigned int ) 0x0200;
PHASE2 = ( unsigned int ) PWM2_TBASE;
PDC2 = ( unsigned int ) ( PWM2_TBASE >> 2 );
DTR2 = ( unsigned int ) DTIME_50NS;
ALTDTR2 = ( unsigned int ) DTIME_100NS;
TRGCON2 = ( unsigned int ) 0x0000;
IOCON2 = ( unsigned int )0xC000;
FCLCON2 = ( unsigned int ) 0x0003;
TRIG2 = ( unsigned int ) 0x0000;
LEBCON2 = ( unsigned int ) 0x0000
```

Lab 1

- **Solution**

```
// ===========================================
//              PWM 3
//              Independent time base
//              Independent duty cycle
//              Frequency 150 KHz
//              Push-pull output
//              Duty cycle = 50 %
//              No dead band
// ---------------------------------------------------------------------

PWMCON3 = ( unsigned int ) 0x0280;
PHASE3 = ( unsigned int ) PWM3_TBASE;
PDC3 = ( unsigned int ) ( PWM3_TBASE >> 1 );
DTR3 = ( unsigned int ) 0x0000;
ALTDTR3 = ( unsigned int ) 0x0000;
TRGCON3 = ( unsigned int ) 0x0000;
IOCON3 = ( unsigned int )0xC800;
FCLCON3 = ( unsigned int ) 0x0003;
TRIG3 = ( unsigned int ) 0x0000;
LEBCON3 = ( unsigned int ) 0x0000;
```

A/D Converter

IPP ADC

- General Features
  - 10 bit Resolution, +/- 1 bit accuracy
  - 2 Million Samples / sec conversion rate
  - 6 to 12 input channels
  - Analog Input Range: 0 to 5V
  - Low latency improves control loop stability
  - Advanced Sampling Capability:
    - Individual triggers for each S&H
    - Samples may be simultaneous
    - Samples may be uniquely timed
    - Sample acquisitions are precisely timed

IPP ADC

- The SMPS ADC samples and converts inputs independently, and asynchronously of each other
- Data capture from time critical events is simplified with sophisticated triggering capabilities
- Conversions are ALWAYS performed in “pairs” of analog inputs: (AN0, AN1), (AN2, AN3), etc …
- Pairs represent voltage and current measurements

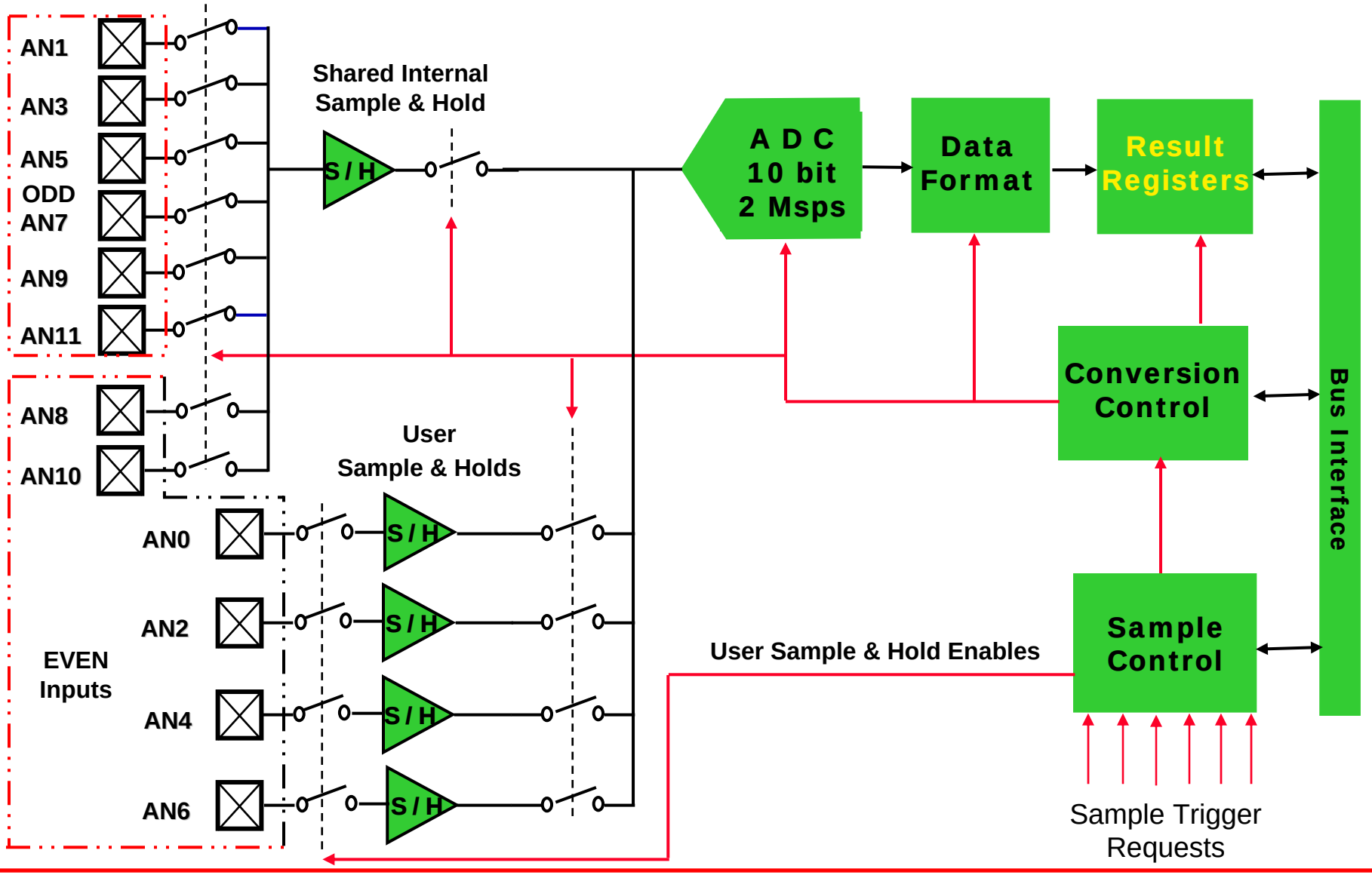
IPP ADC

- Output Buffers
  - Each analog input pin has an associated output data register
  - The data buffers (registers) are NOT implemented as a FIFO
  - They are uniquely owned by each input

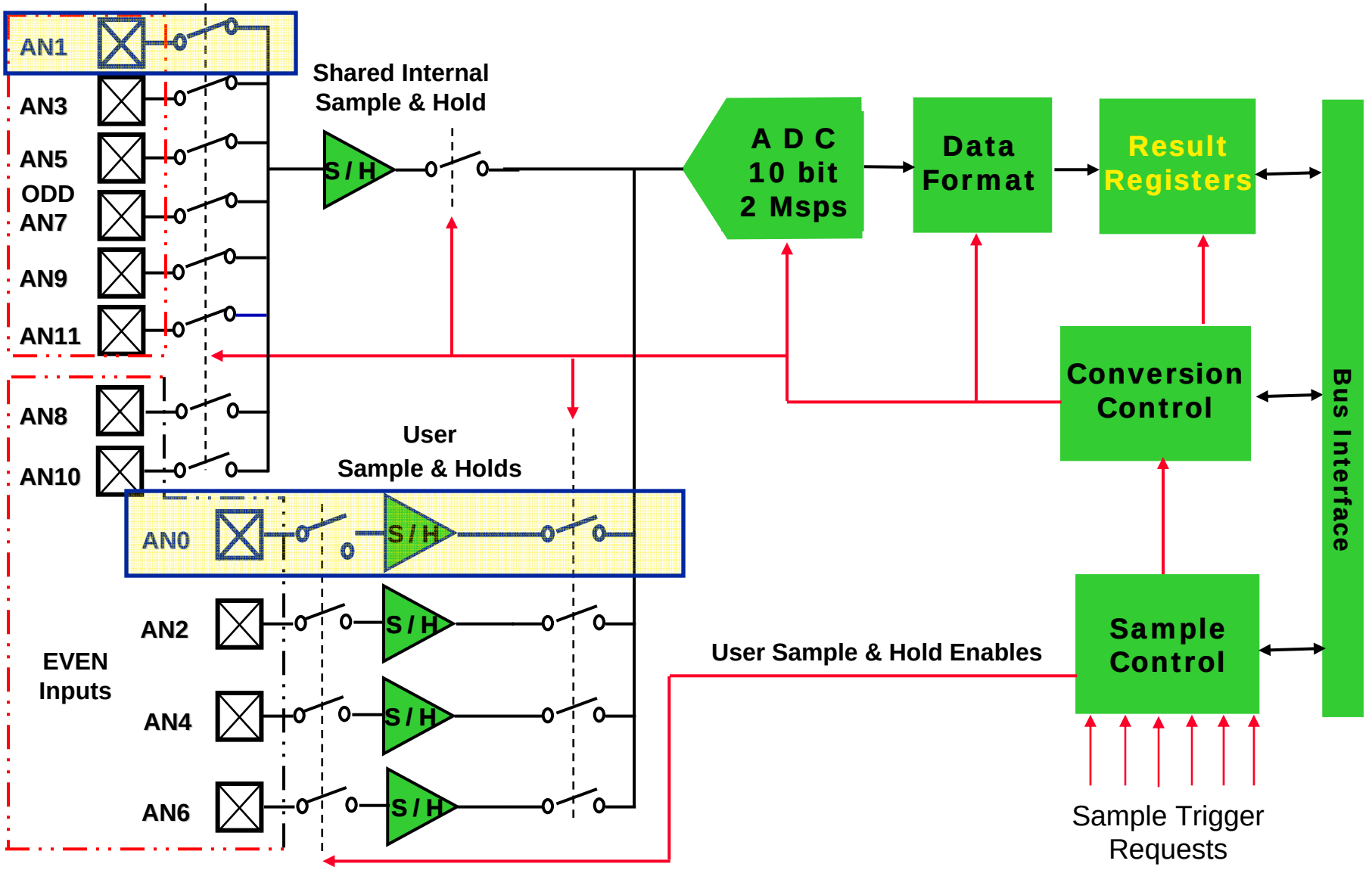
IPP ADC

- Interrupt Management
  - Interrupt requests are generated on a per pair basis
  - Interrupts may be generated at the completion of first or second conversion of the pair

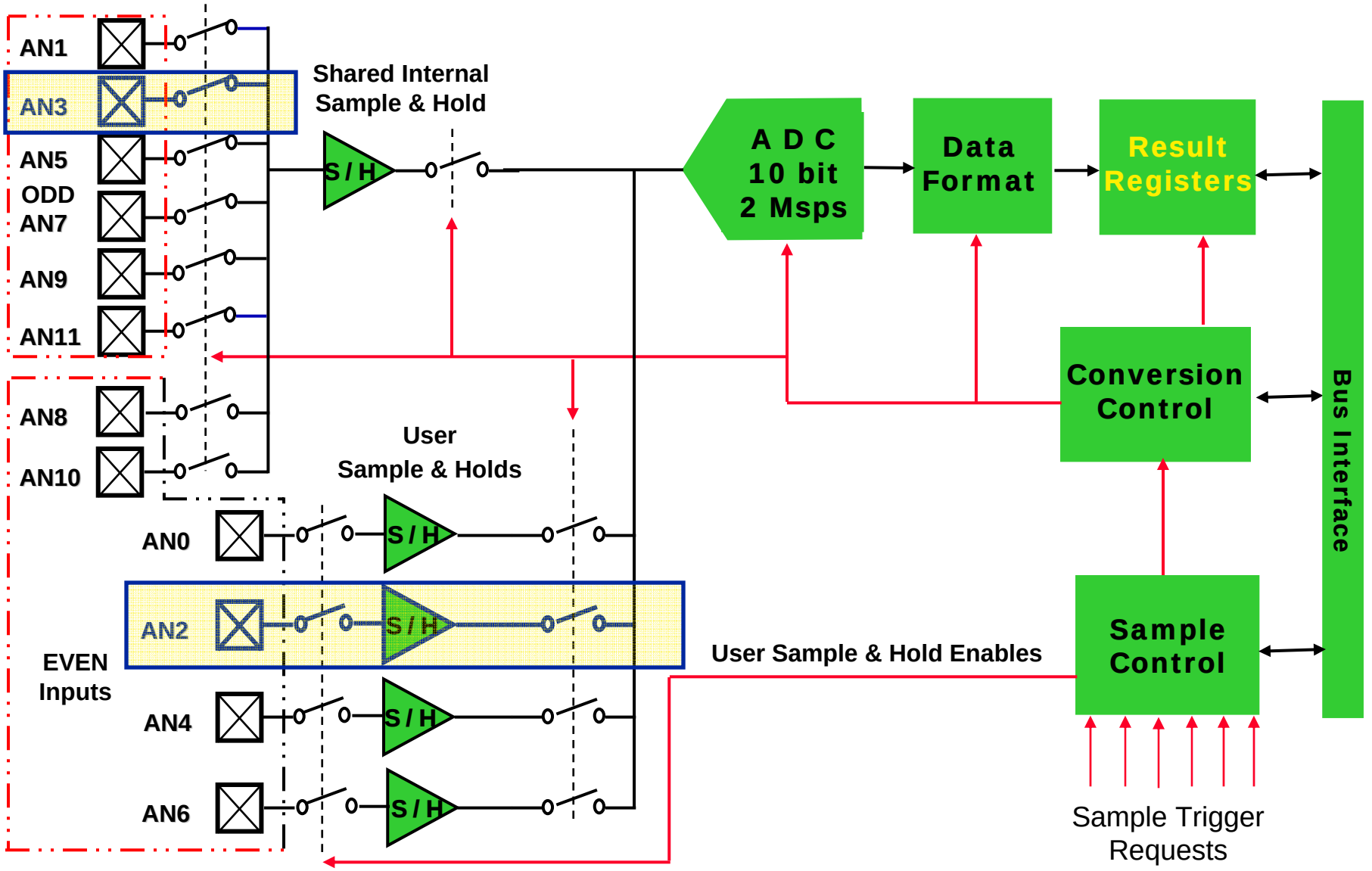
IPP ADC



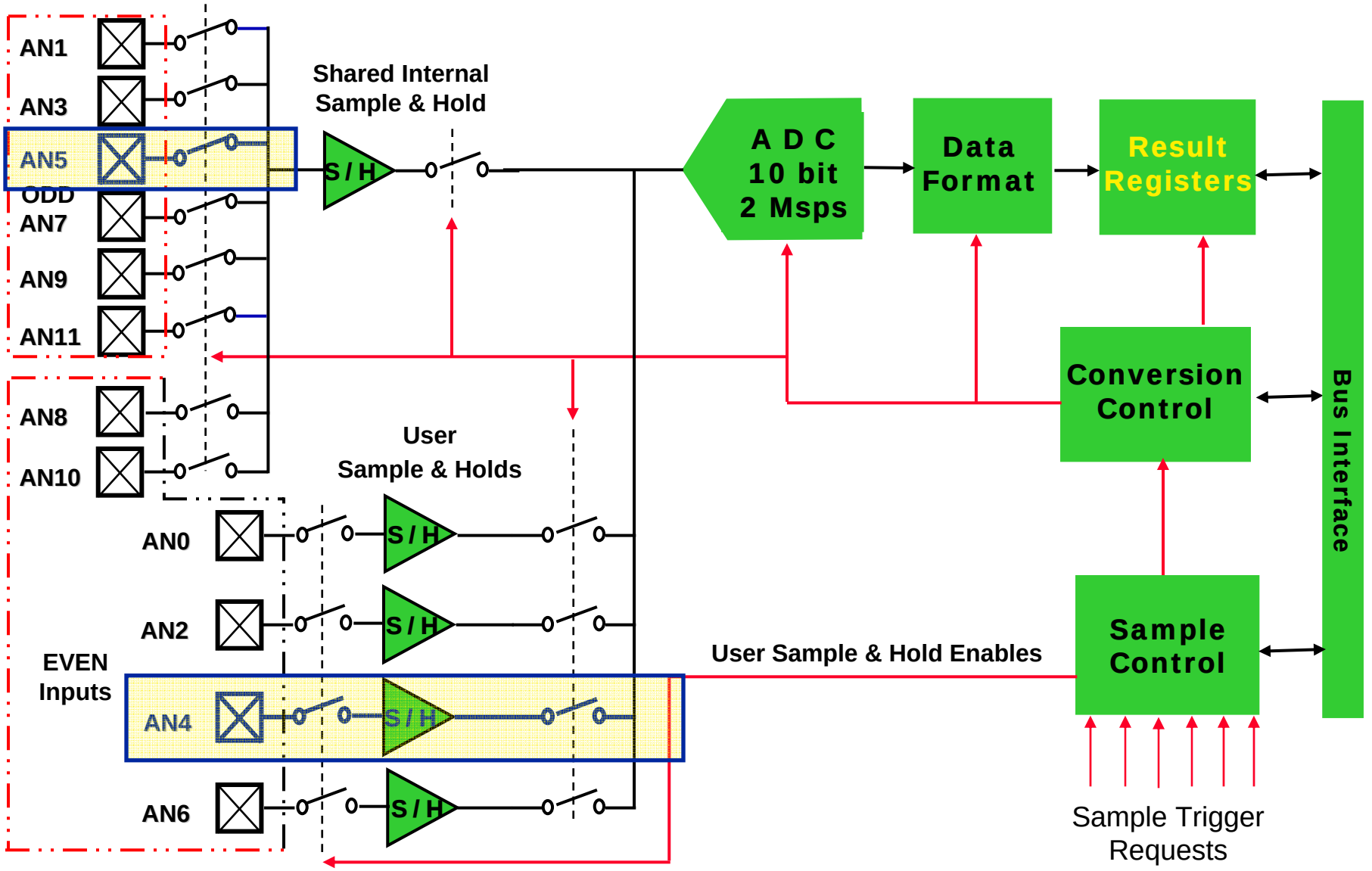
IPP ADC



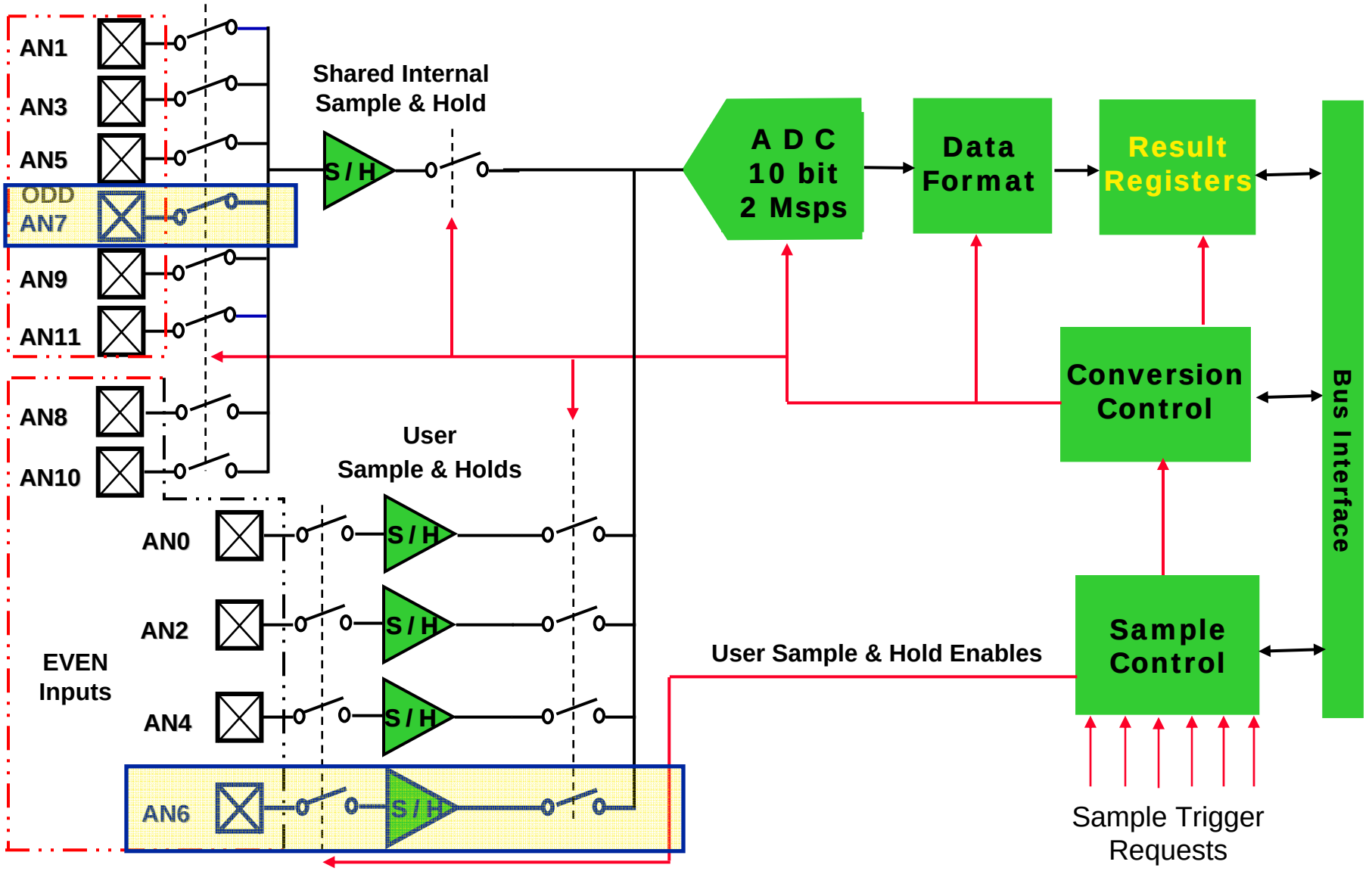
IPP ADC



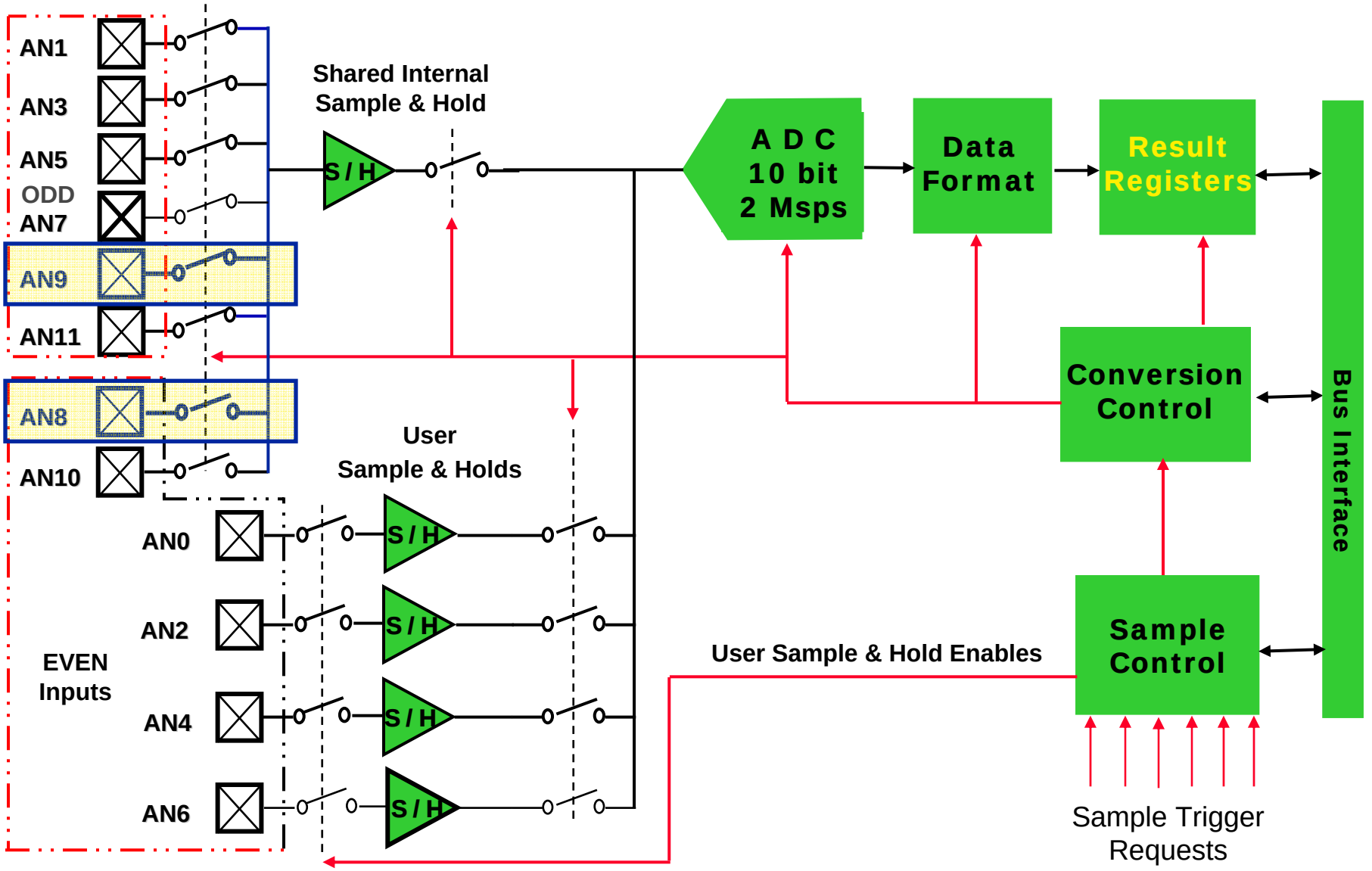
IPP ADC



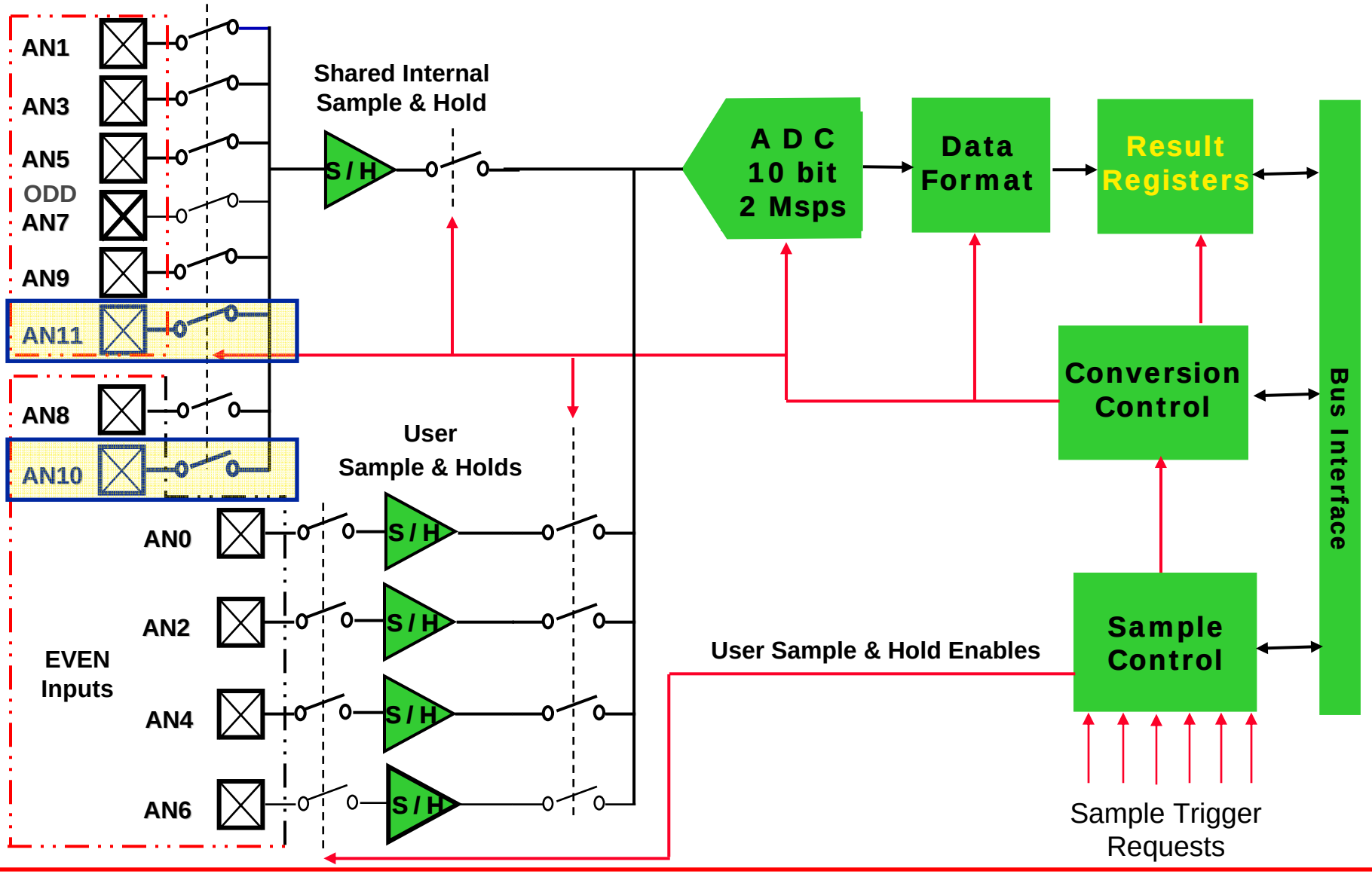
IPP ADC



IPP ADC



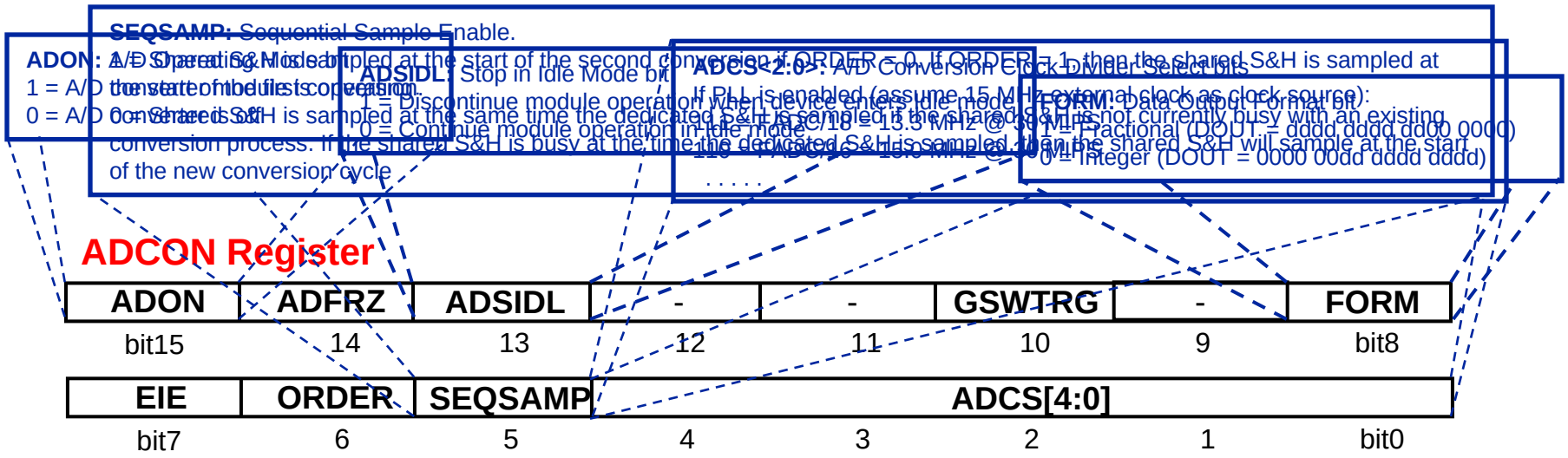
IPP ADC



IPP ADC

ADCONx Register

- Enables the peripheral
- Controls its operation in idle
- Defines the data output format
- Enables sequential sampling
- Defines the ADC clock



IPP ADC

ADPCFG<11:0>: A/D Port Configuration Control bits

- 1 = Port pin in Digital mode, port read input enabled
- 0 = Port pin in Analog mode, port read input disabled

ADPCFGx Register

-	-	-	-	PCFG11	PCFG10	PCFG9	PCFG8
bit15	14	13	12	11	10	9	bit8

PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit7	6	5	4	3	2	1	bit0

IPP ADC

ADCPCx register:

- Interrupt request enable (for the pair)
- Pending conversion status bit
- Software trigger
- Selects trigger source for conversion of analog channel pair

TRGSRC0<4:0>: Trigger 0 Source Selection bits
Selects trigger source for conversion of analog channels AN1 and AN0.
00000 = No conversion enabled
00001 = Individual software trigger selected
00010 = Global software trigger selected
00011 = PWM Special Event Trigger selected
00100 = PWM generator #1 trigger selected
00101 = PWM generator #2 trigger selected
00110 = PWM generator #3 trigger selected
00111 = PWM generator #4 trigger selected
01100 = Timer #1 period match
01101 = Timer #2 period match
01110 = PWM GEN #1 current-limit ADC trigger
01111 = PWM GEN #2 current-limit ADC trigger
10000 = PWM GEN #3 current-limit ADC trigger
10001 = PWM GEN #4 current-limit ADC trigger
10110 = PWM GEN #1 fault ADC trigger
10111 = PWM GEN #2 fault ADC trigger
11000 = PWM GEN #3 fault ADC trigger
11001 = PWM GEN #4 fault ADC trigger

SWTRGy: Software Trigger 0 bit
1 = Start conversion of AN1 and AN0 (if selected by TRGSRC bits). If other conversions are in progress, then conversion will be performed when the conversion resources are available. This bit will be reset when the PEND bit is set
0 = Conversion is complete

IRQEN0: Interrupt Request Enable 0 bit
1 = Enable IRQ generation when requested conversion of AN1 and AN0 is complete
0 = IRQ is not generated

PEND0: Pending Conversion Status 0 bit
1 = Conversion of requested AN1 and AN0 is pending. Set when selected trigger is asserted
0 = Conversion is complete

ADCPCx Register

IRQENz	PENDz	SWTRGz	TRGSRCz[4:0]				
bit15	14	13	12	11	10	9	bit8

IRQENy	PENDy	SWTRGy	TRGSRCy[4:0]				
bit7	6	5	4	3	2	1	bit0

IPP ADC

- The ADCPCx registers:
  - Association to the channels pairs

ADCPC2 Register

Channels AN10 and AN11	**IRQENz**	**PENDz**	**SWTRGz**	**TRGSRCz[4:0]**				
	bit15	14	13	12	11	10	9	bit8
Channels AN8 and AN9	**IRQENy**	**PENDy**	**SWTRGy**	**TRGSRCy[4:0]**				
	bit7	6	5	4	3	2	1	bit0

ADCPC1 Register

Channels AN6 and AN7	**IRQENz**	**PENDz**	**SWTRGz**	**TRGSRCz[4:0]**				
	bit15	14	13	12	11	10	9	bit8
Channels AN4 and AN5	**IRQENy**	**PENDy**	**SWTRGy**	**TRGSRCy[4:0]**				
	bit7	6	5	4	3	2	1	bit0

ADCPC0 Register

Channels AN2 and AN3	**IRQENz**	**PENDz**	**SWTRGz**	**TRGSRCz[4:0]**				
	bit15	14	13	12	11	10	9	bit8
Channels AN0 and AN1	**IRQENy**	**PENDy**	**SWTRGy**	**TRGSRCy[4:0]**				
	bit7	6	5	4	3	2	1	bit0

IPP ADC

- Interrupts
  - The ADC module provides a common or “Group” interrupt request that is the OR of all of the enabled interrupt sources within the module

Global Interrupt Flag Status Register	IFS0: ADIF
Global Interrupt Enable Control Register	IEC0: ADIE
Interrupt Priority Control Register	IPC2: ADIP[2:0]

IPP ADC

- Interrupts
  - Each ADCPC register has two IRQENx bits, one for each analog input pair; if the IRQEN bit is set, an interrupt request is made to the interrupt controller when the requested conversion is completed
  - When an interrupt is generated, an associated PxRDY bit in the ADSTAT register is set

ADC Pair x Interrupt Flag Status Register	IFS2: ADCPxIF
ADC Pair x Interrupt Enable Control Register	IEC2: ADCPxIE
ADC Pair x Interrupt Priority Control Register	IPC9: ADCPxIP IPC10: ADCPxIP

IPP ADC

- Sample and Conversion
  - The ADC module assigns two ADC clock periods for the sampling process
  - When operating at 24 MHz clock, the sampling period is:

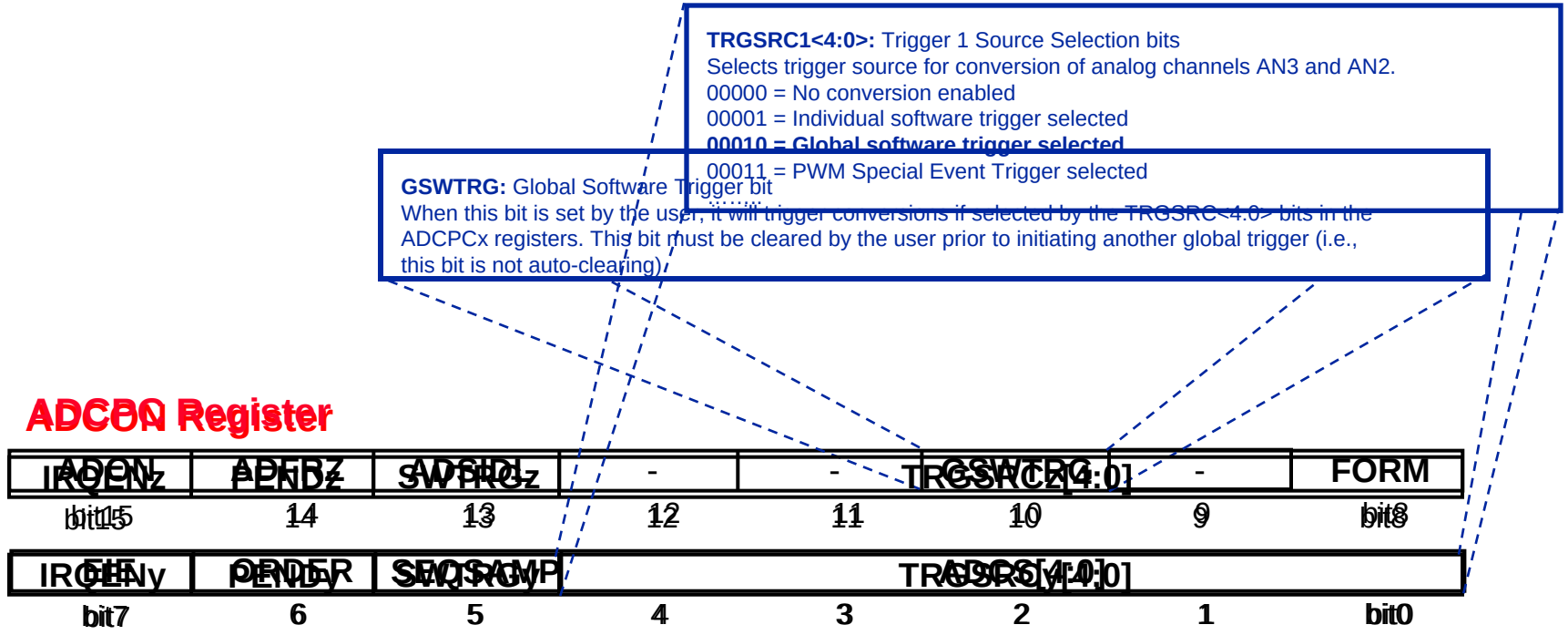
    2 x 41.6 nsec = 83.3 nsec.

IPP ADC

- Sample and Conversion
  - Each ADC pair specified in the ADCPCx registers initiates a sample operation when the selected trigger event occurs
  - The conversion of the sampled data occurs as resources become available
  - The actual conversion process requires 10 additional ADC clocks

IPP ADC

- Global Software Trigger
  - Start Conversion Setting a Bit
  - Select the Trigger Source per Pair



IPP ADC

- Individual Software Trigger
  - Select the Trigger Source per Pair
  - Enable the Interrupt Request
  - Start Conversion Setting a Bit
  - ISR Code: ADCInterrupt

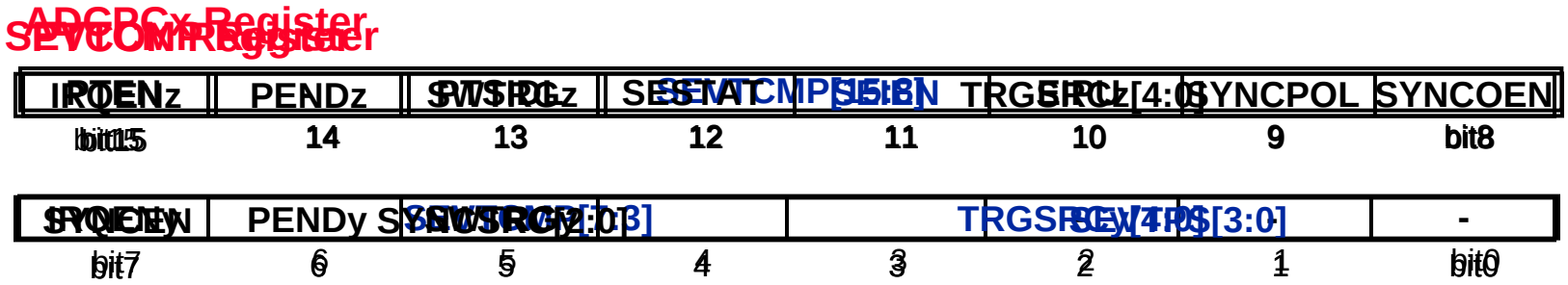
ADCPCx Register

IRQENz	PENDz	SWTRGz	TRGSRCz[4:0]				
bit15	14	13	12	11	10	9	bit8

IRQENy	PENDy	SWTRGy	TRGSRCy[4:0]				
bit7	6	5	4	3	2	1	bit0

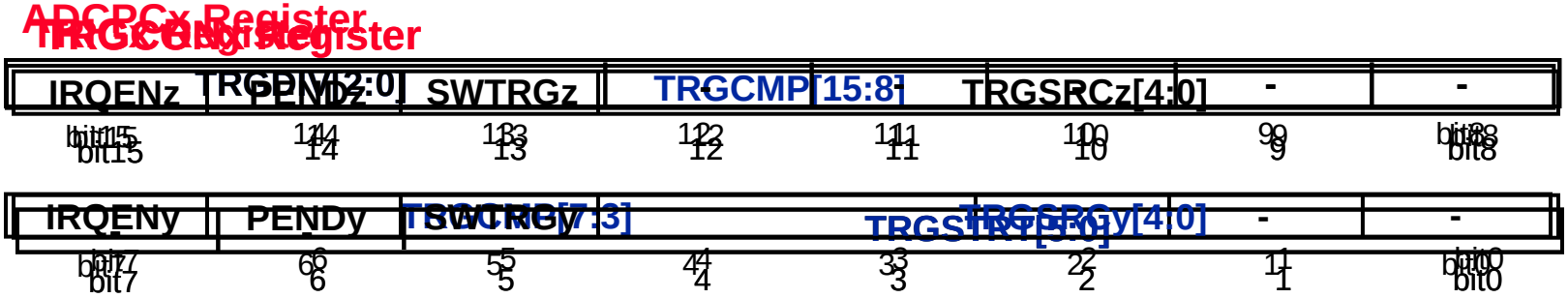
IPP ADC

- Special Event Trigger (Master Period/Duty)
  - Set Value in PWM Peripheral Register SEVTCMP
  - Set Interrupt Enable bit and Postscaler in PWM Register PTCON
  - Select ADC Trigger Source in ADC Register ADCPCx
  - ISR Code: PWMSpEventMatchInterrupt



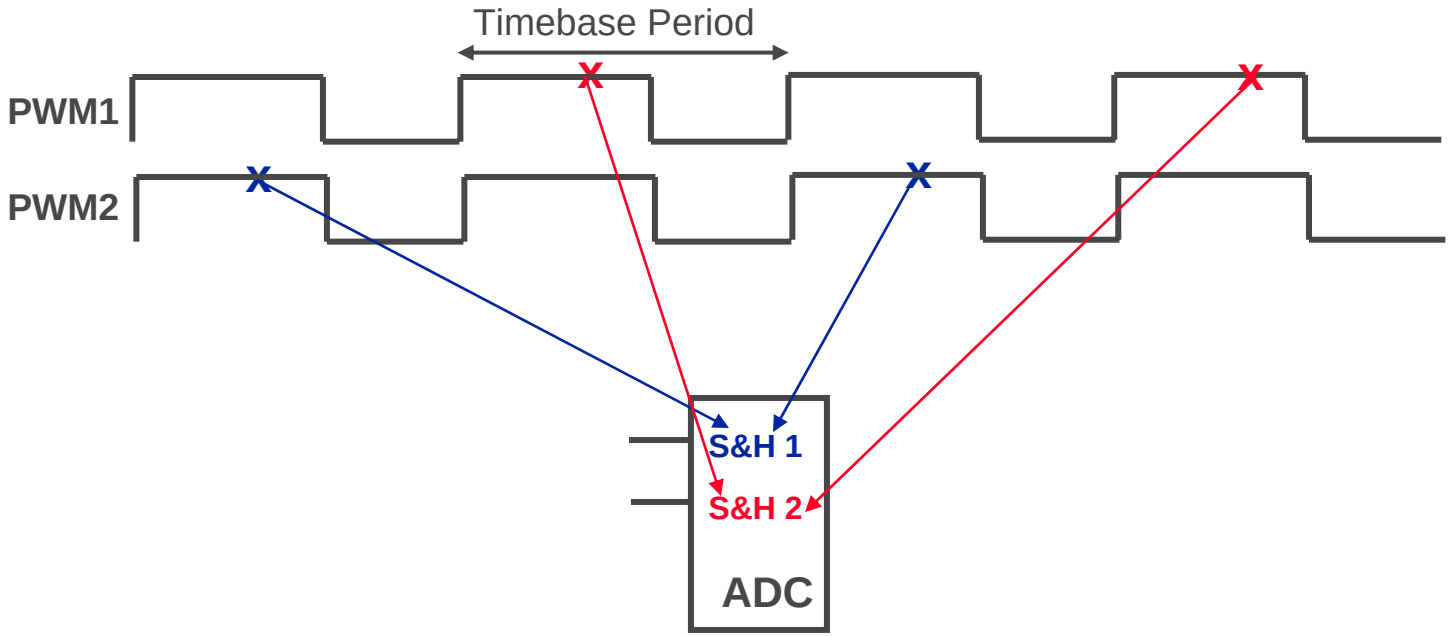
IPP ADC

- Individual Triggers (Individual Period/Duty)
  - Set Value in PWM Peripheral Register TRIGx
  - Select the Trigger Output Divider in PWM Peripheral Register TRGCONx
  - Select the Postscaler for Staggering Operation in PWM Peripheral Register TRGCONx
  - Select ADC Trigger Source in ADC Register ADCPCx



IPP ADC

Triggers can be staggered relative to other PWM generated triggers on a PWM Period Basis



IPP ADC

- Summary. We have seen:
  - The basic architecture of the IPP ADC
  - How to program it to get the best performances
  - The triggering capabilities

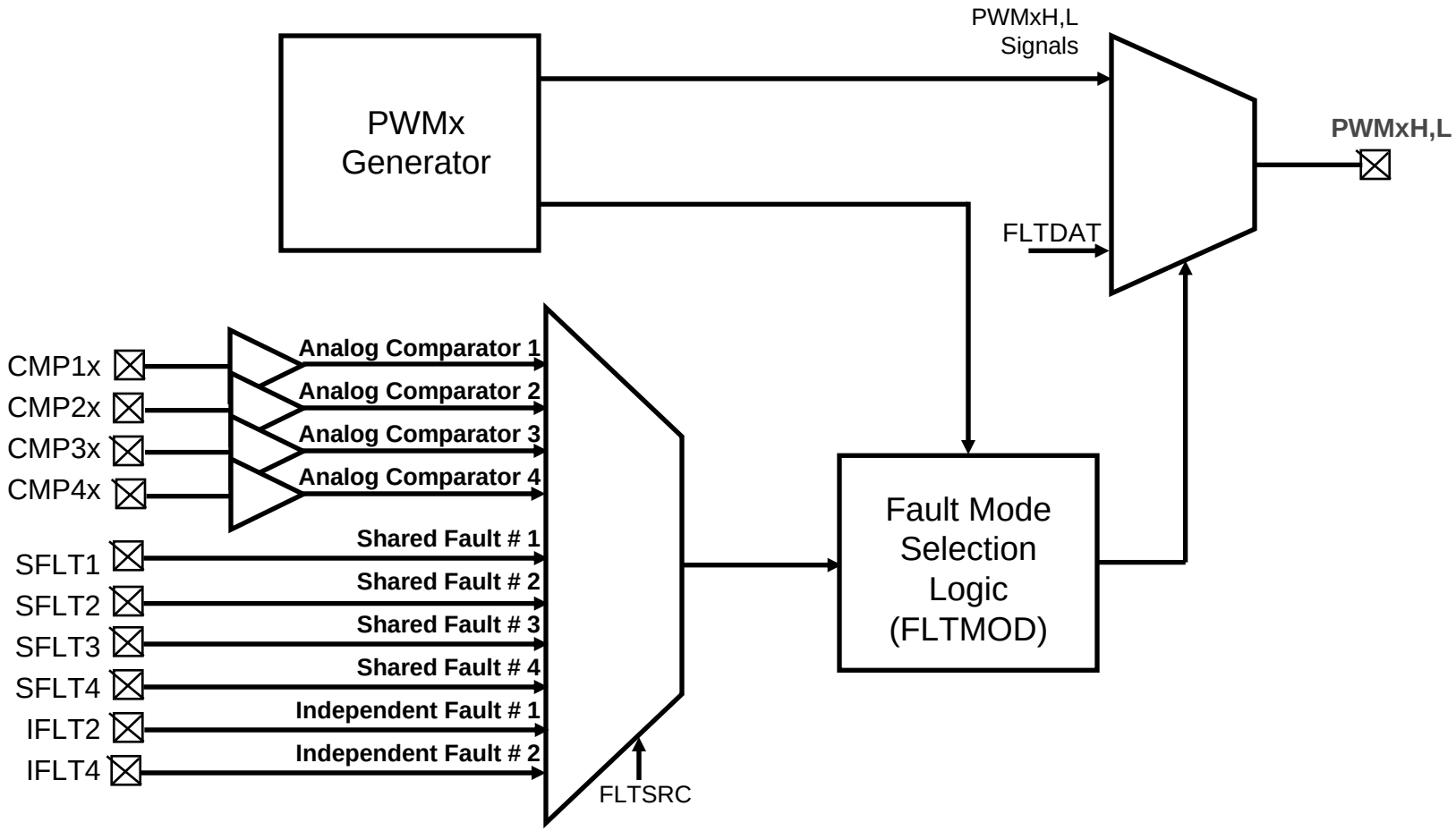
Fault Generation

Faults

- The Fault pins actually serve two different purposes:
  - Generation of Fault overrides for the PWM outputs; the action of overriding the PWM outputs is performed asynchronously in hardware
  - The Fault pin inputs can be used to implement either Current-Limit PWM mode

Faults

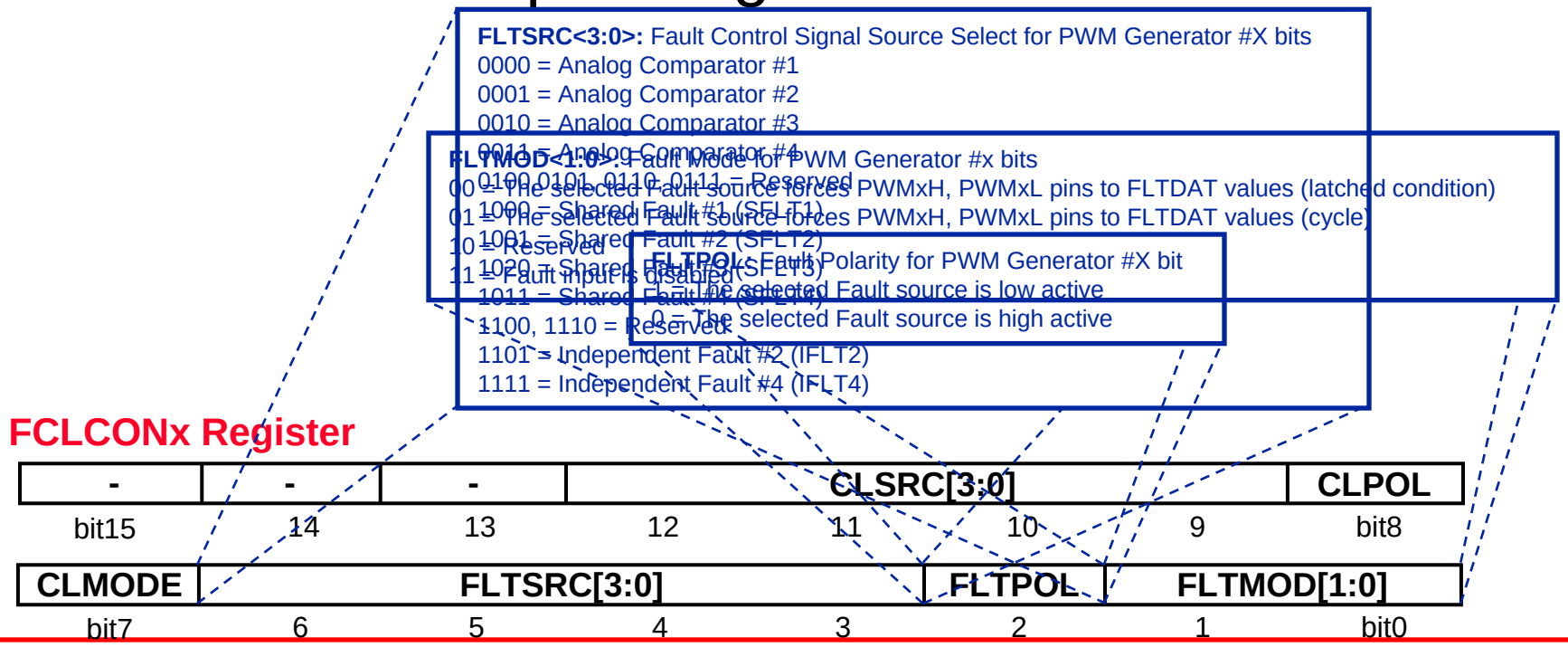
- Bock Diagram



Faults

FCLCONx register:

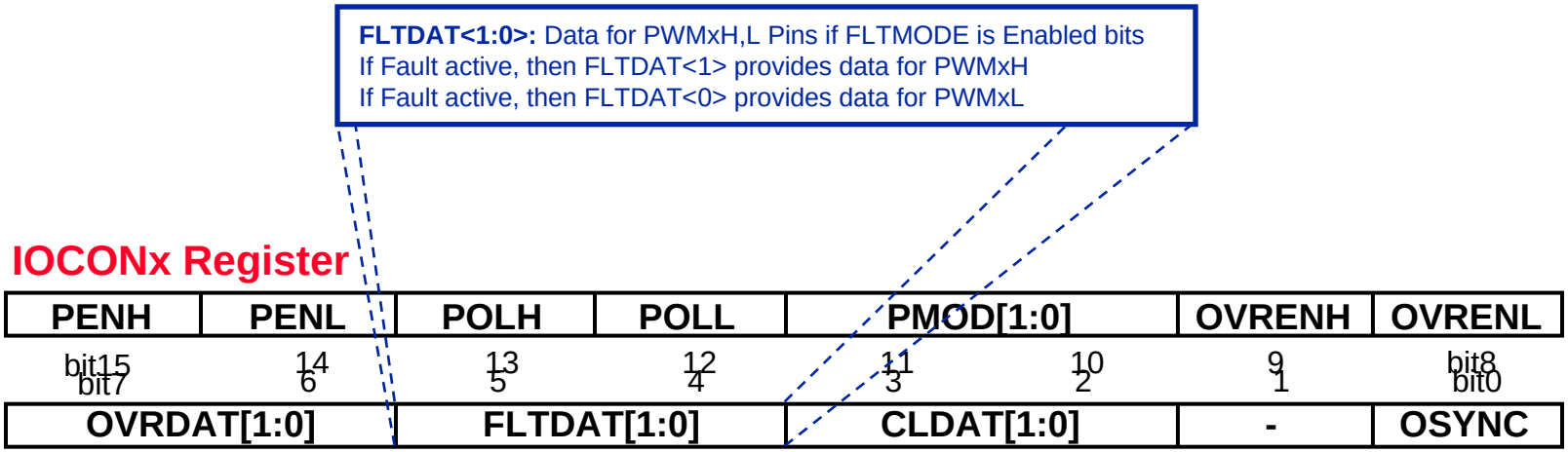
- Selects the fault input source for each PWM
- Selects the fault polarity
- Selects the operating mode



Faults

The IOCONx register:

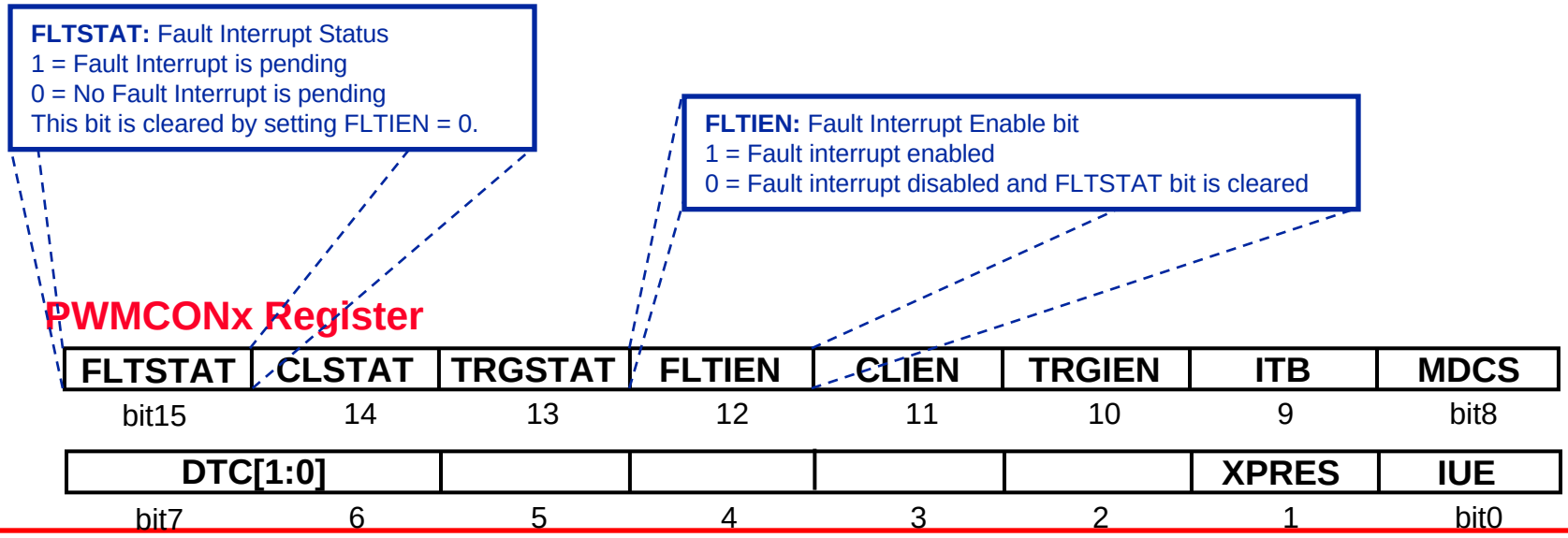
- Select the values output on PWMH and PWML when a Fault Event Occurs



Faults

The PWMCON register:

- Enables the fault interrupt
- Pending Interrupt Status Bit



Current Limit

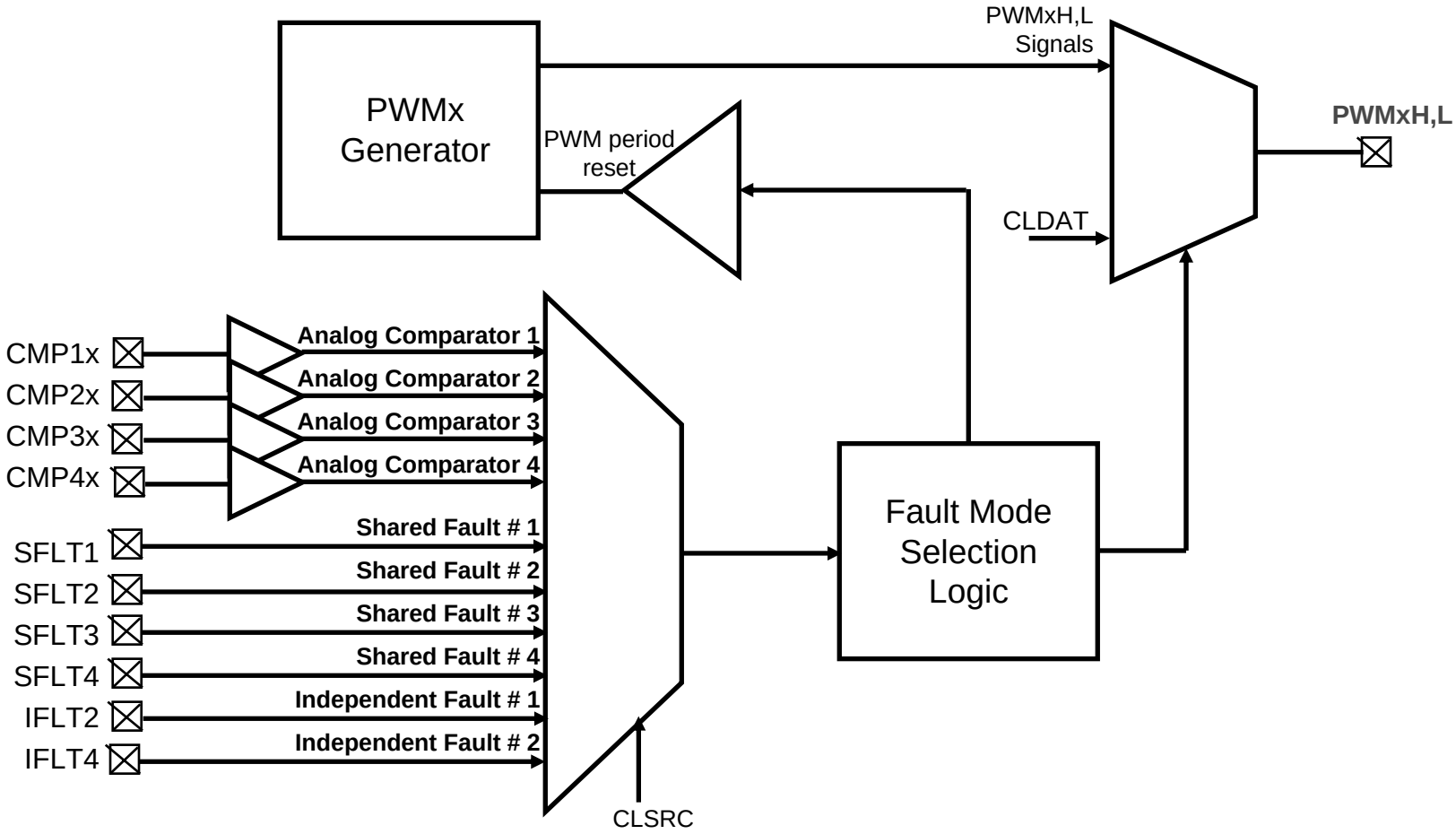
UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

Current Limit

- The current-limit pins actually serve two different purposes
- Current-Limit PWM mode
  - If the selected current-limit input signal is asserted the PWMxH,L outputs are forced to the values specified by the CLDAT<1:0> bits in the IOCONx register
- Current Reset PWM mode
  - A current-limit signal resets the time base if:
    - CLMOD bit is zero (current limit disabled)
    - XPRES bit in the PWMCONx register is '1'
    - The PWM generator is in Independent Time Base mode
  - This is used in some PFC applications

Current Limit

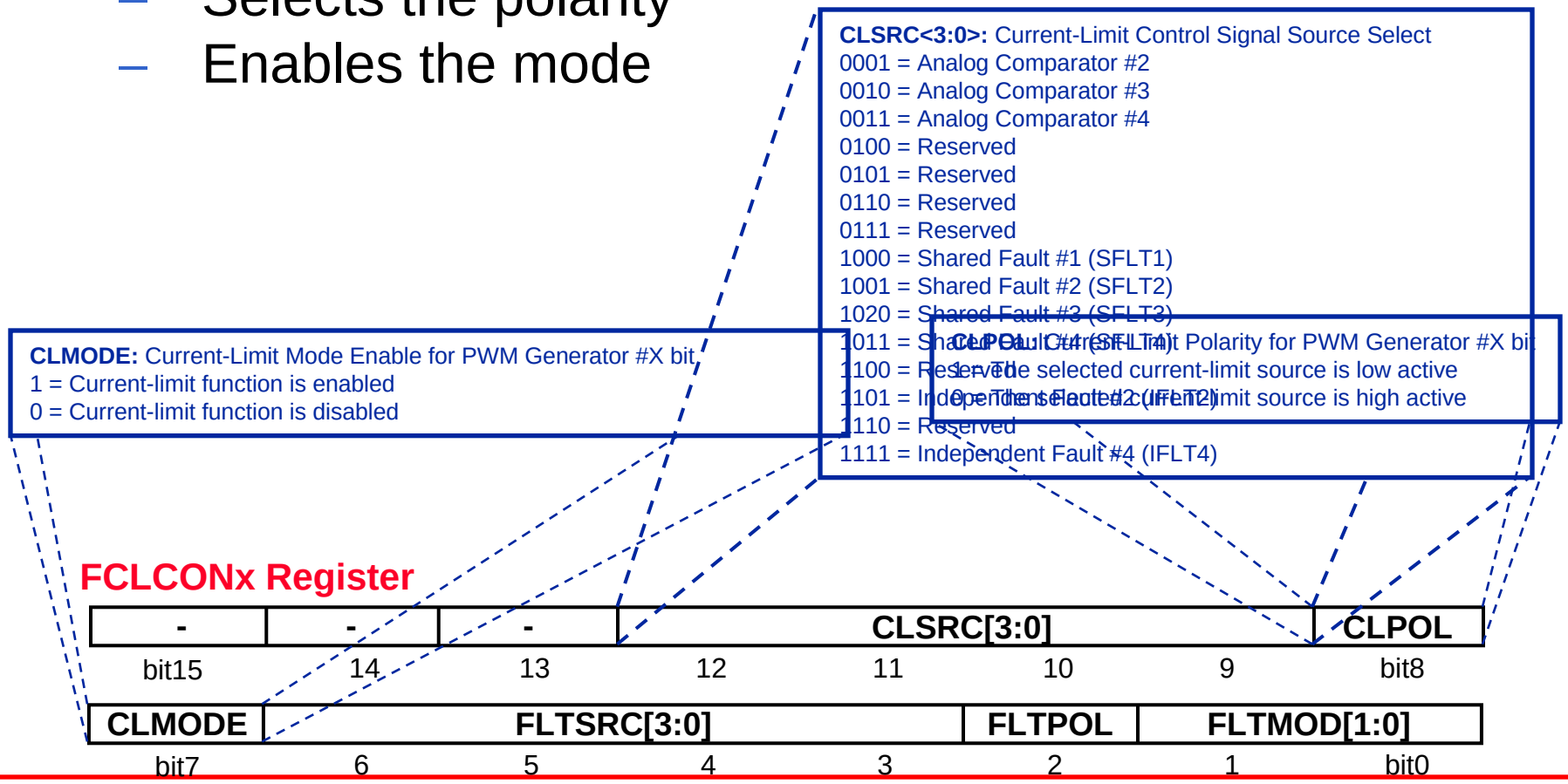
Block Diagram



Current Limit

FCLCONx register:

- Selects the Current limit input signal
- Selects the polarity
- Enables the mode



Current Limit

IOCONx register:

- Select the values output on PWMH and PWML when a Fault Event Occurs

CLDAT<1:0>: Data for PWMxH,L Pins if CLMODE is Enabled bits
If current limit active, then CLDAT<1> provides data for PWMxH
If current limit active, then CLDAT<0> provides data for PWMxL

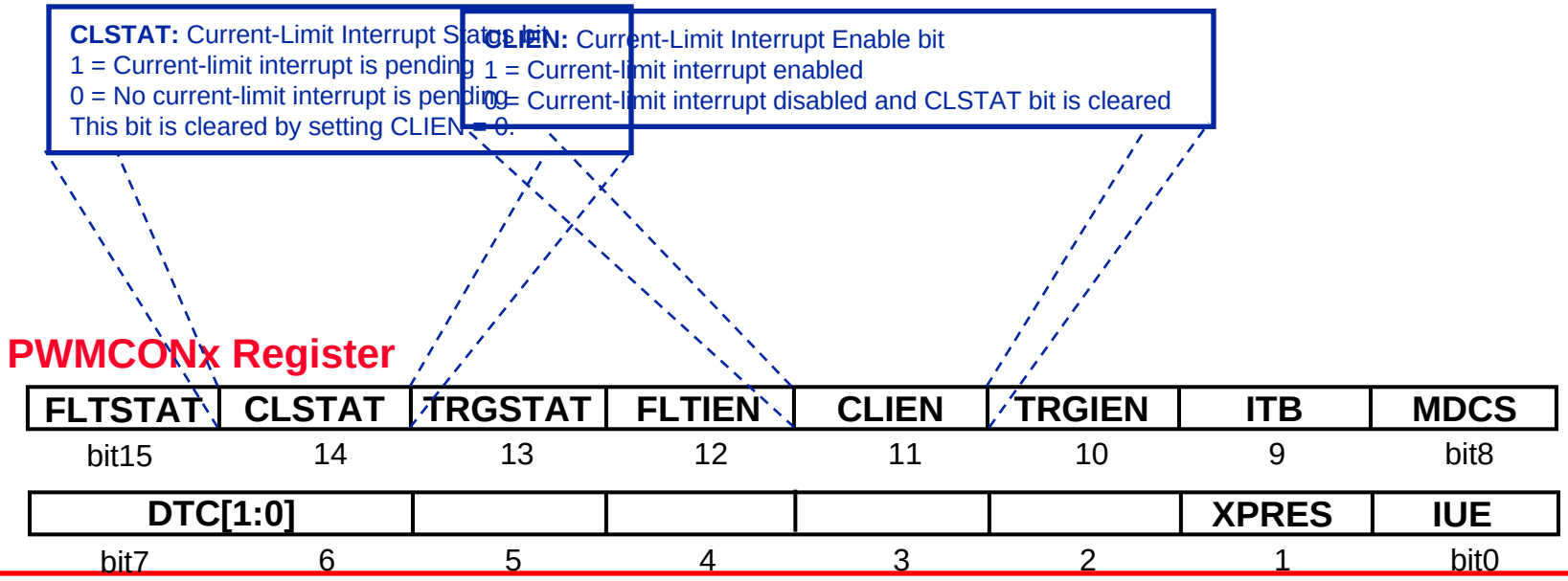
IOCONx Register

PENH	PENL	POLH	POLL	PMOD[1:0]		OVRENH	OVRENL
bit15 bit7	14 6	13 5	12 4	11 3	10 2	9 1	bit8 bit0
OVRDAT[1:0]		FLTDAT[1:0]		CLDAT[1:0]		-	OSYNC

Current Limit

The PWMCON register:

- Enables the current-limit interrupt
- Pending Interrupt Status Bit



Current Limit

- Summary. We have seen:
  - Fault operation
  - Current limiting operation

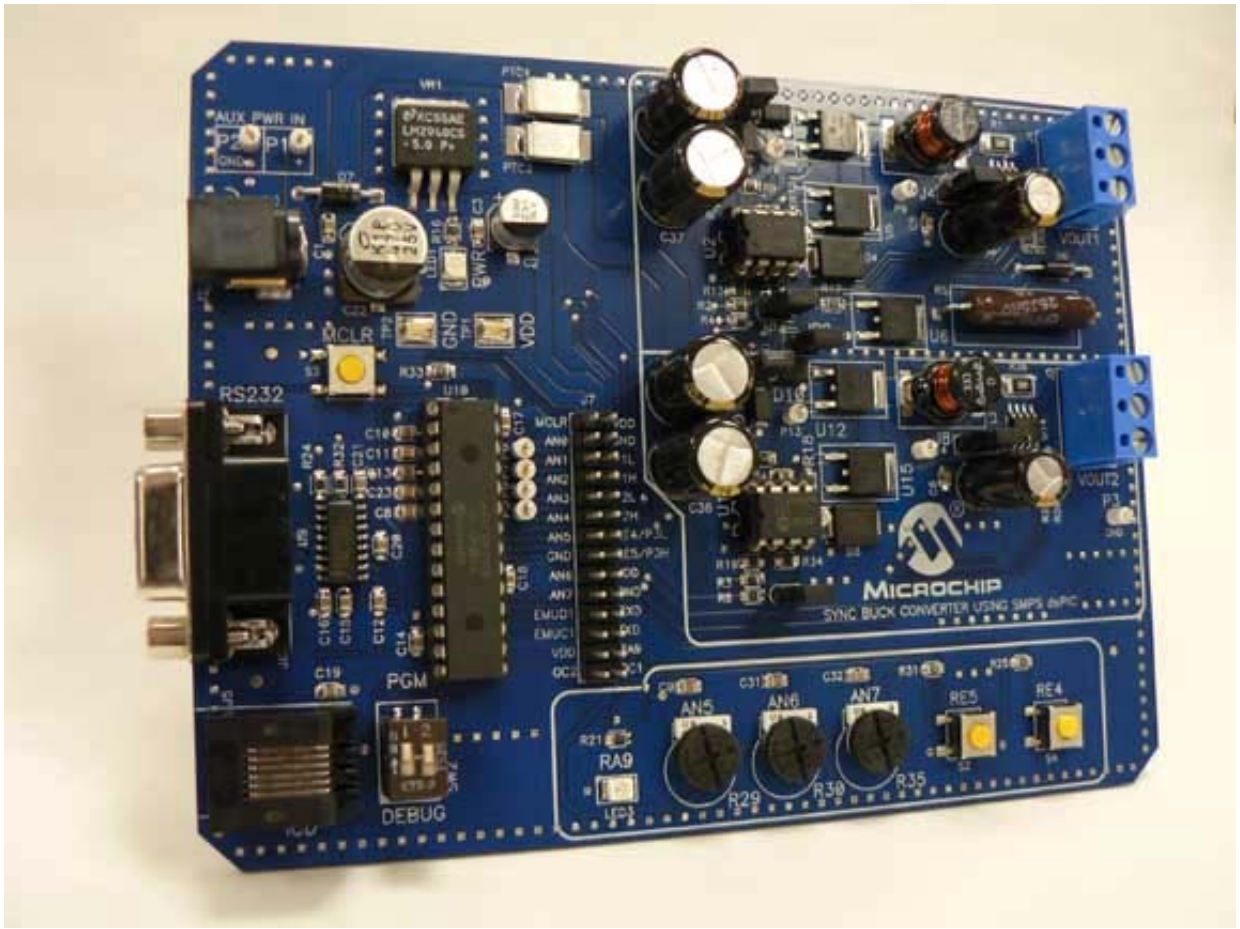
SMPS and ICD2

- SMPS Devices and ICD2
  - When the ICD 2 halts the device the PWM pins takes the characteristics of the GPIO that is muxed on that pin
  - PWM1L and PWM1H are muxed with RE0 and RE1
  - If TRISE = 0x00FF then the pins will be tri-stated on a halt

    If TRISE = 0x00FC and PORTE = 0x0000 then the pins will drive low on a halt
  - We can configure how the PWM pins behave on an ICD2 halt

University of Microchip UofM MASTERs 2007

Buck Board



Lab 2

Lab 2

- **Lab 2 is intended to:**
  - Review the PWM Fault
  - Review Current Limit capabilities

Lab 2

- In this Lab, you will:
  - Part A
    - Initialize PWM 1: fault (cycle condition)
    - Initialize PWM 2: fault (latched condition)
    - Cycle Condition Operation
  - Part B
    - Latched Condition Operation
  - Part C
    - Optional

Lab 2

- **Solution**

```
// PWM 1: Fault control
// Independent Time Base (ITB = 1)
// Independent duty cycle (MDCS = 0)
// Complimentary output
// Dead time disabled
// Fault input at SFLT1
// Output low in fault
// Cycle condition
// ---------------------------------------------------------------------

PWMCON1 = ( unsigned int ) 0x0280;
PDC1 = ( unsigned int ) ( PWM1_TBASE >> 1 );
PHASE1 = ( unsigned int ) ( PWM1_TBASE );
DTR1 = ( unsigned int ) 0x0000;
ALTDTR1 = ( unsigned int ) 0x0000;
TRGCON1 = ( unsigned int ) 0x0000;
IOCON1 = ( unsigned int ) 0xC000;
FCLCON1 = ( unsigned int ) 0x0041;
TRIG1 = ( unsigned int ) 0x0000;
LEBCON1 = ( unsigned int ) 0x0000;
```

Lab 2

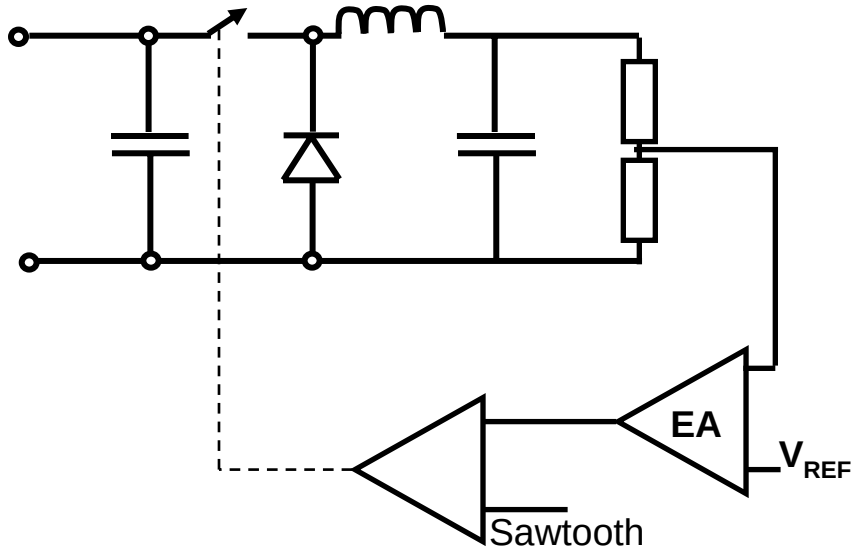
- **Solution**

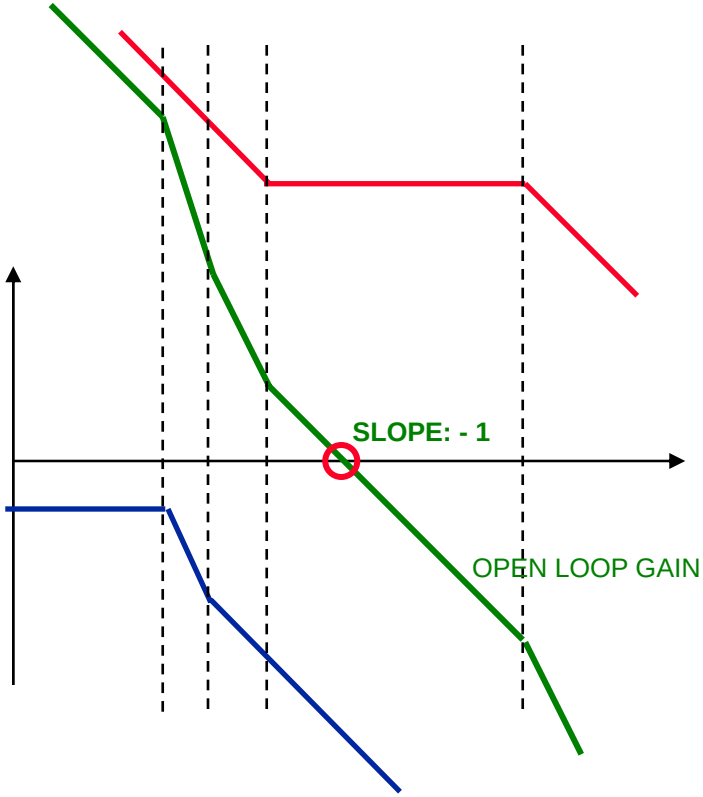
```
// PWM 2: Fault control
// Master Time Base (ITB = 0)
// Master duty cycle (MDCS = 1)
// Complimentary output
// Dead time disabled
// Fault input at SFLT1
// Output low in fault
// Latched condition
// ---------------------------------------------------------------------

PWMCON2 = ( unsigned int ) 0x0180;
PDC2 = ( unsigned int ) 0x0000;
PHASE2 = ( unsigned int ) 0x0000;
DTR2 = ( unsigned int ) 0x0000;
ALTDTR2 = ( unsigned int ) 0x0000;
TRGCON2 = ( unsigned int ) 0x0000;
IOCON2 = ( unsigned int ) 0xC000;
FCLCON2 = ( unsigned int ) 0x0040;
TRIG2 = ( unsigned int ) 0x0000;
LEBCON2 = ( unsigned int ) 0x0000;
```

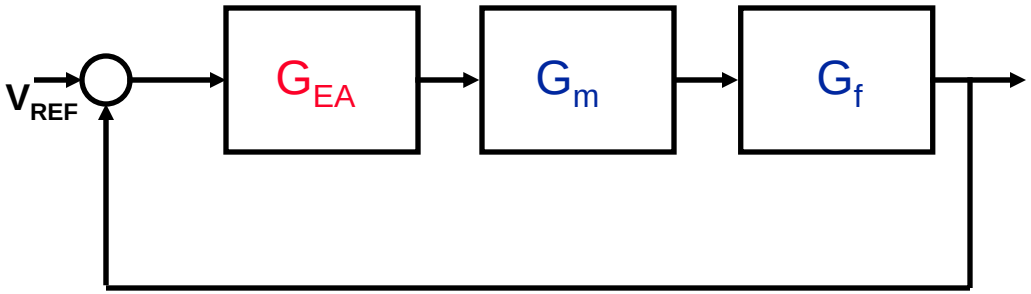
PID Control Loop

PID Control Loop

Classical Feedback-Loop Stabilization







PID Control Loop

From Analog to Digital

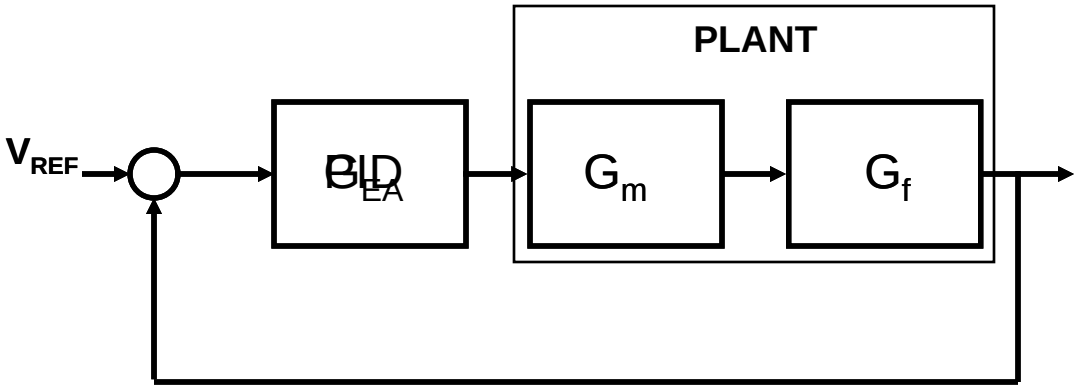
PID

G_{EA} = Error Amplifier Gain

P = Proportional

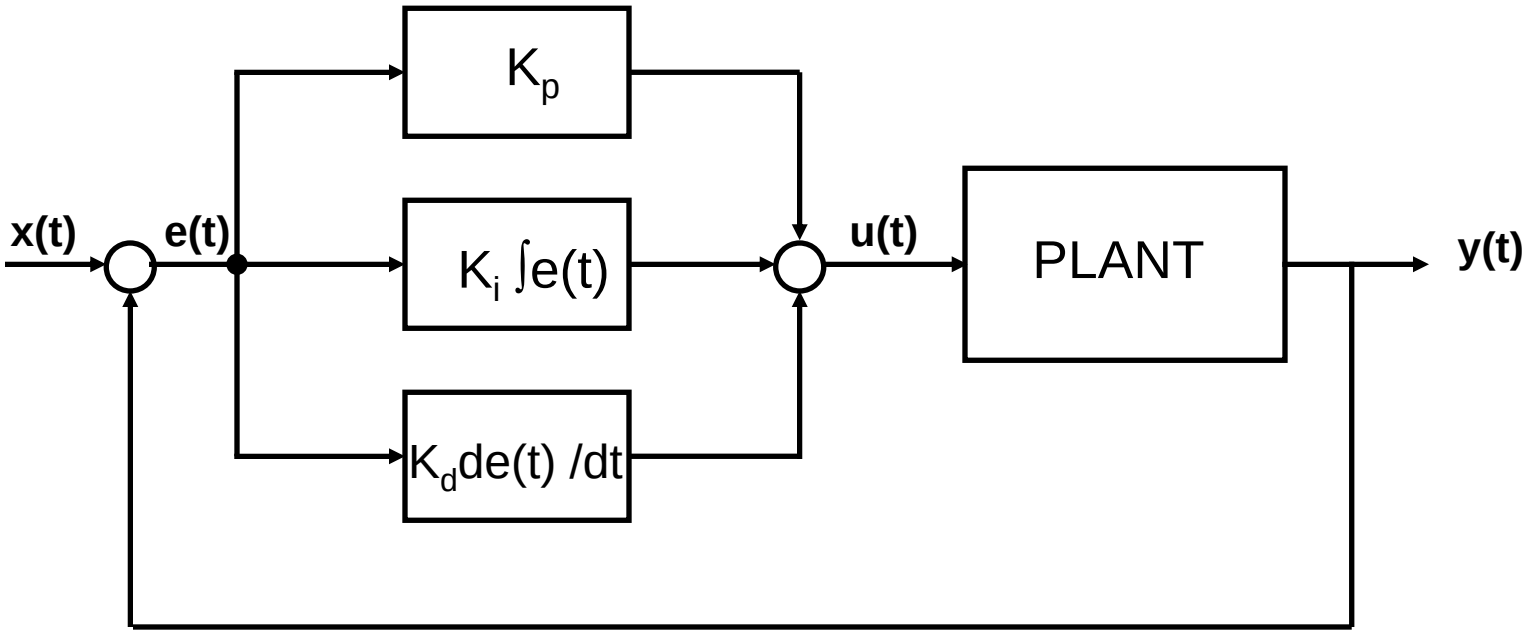
I = Integrative

D = Derivative



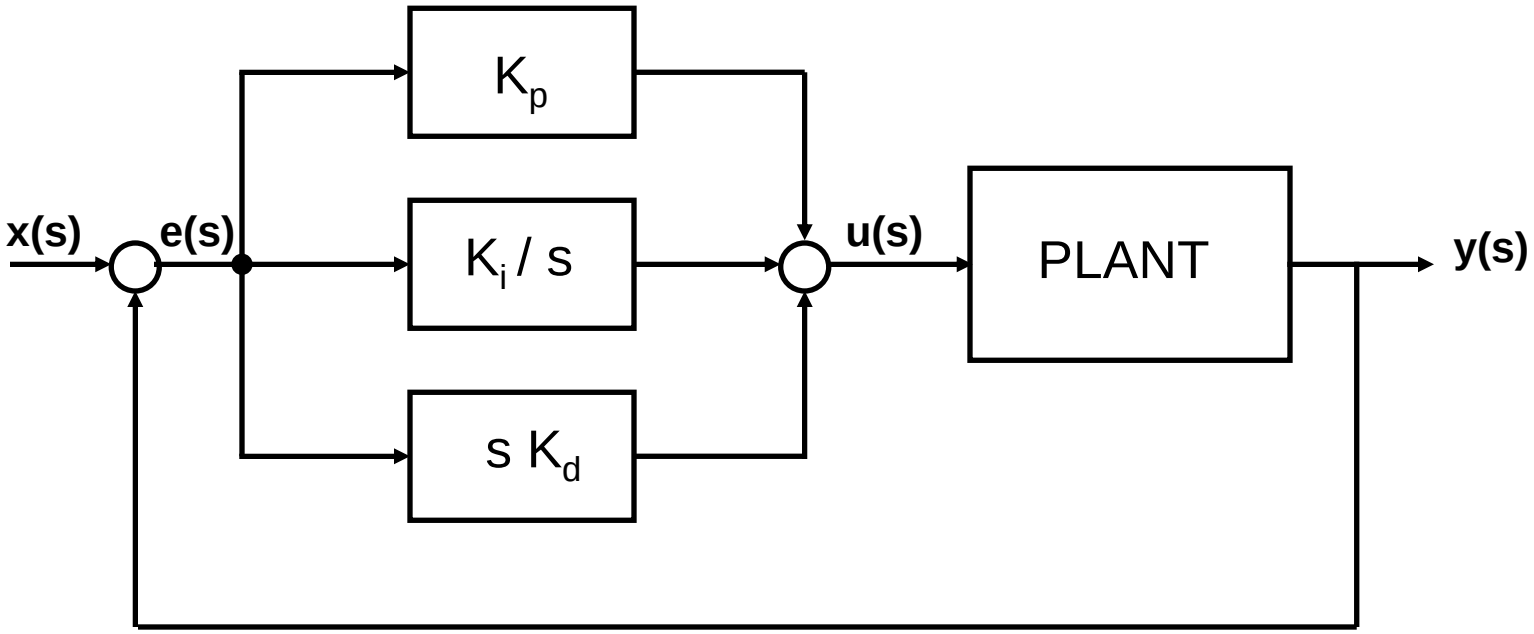
PID Control Loop

PID Structure in Analog: TimeDomain



PID Control Loop

PID Structure Analog: Frequency Domain (Laplace Transform)

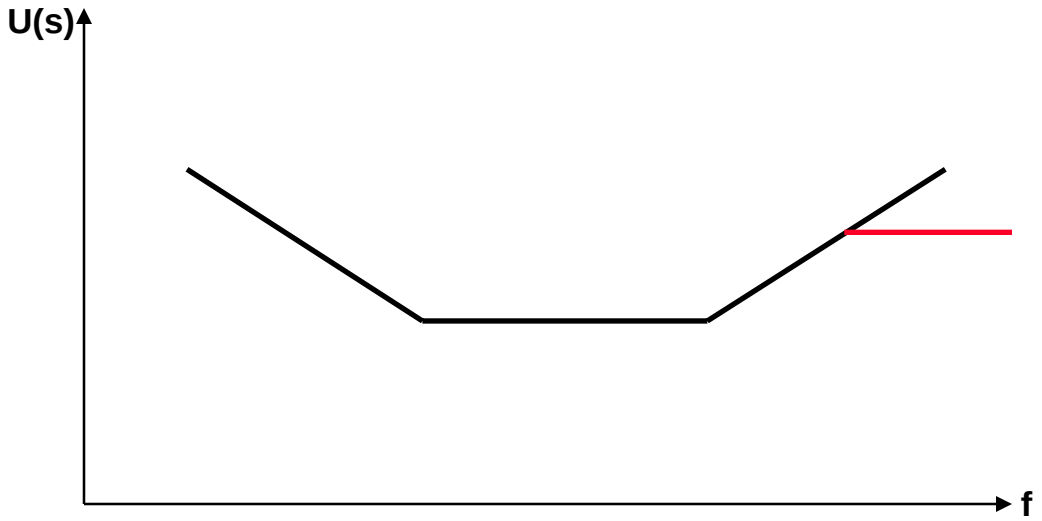


PID Control Loop

Output in the Analog Domain

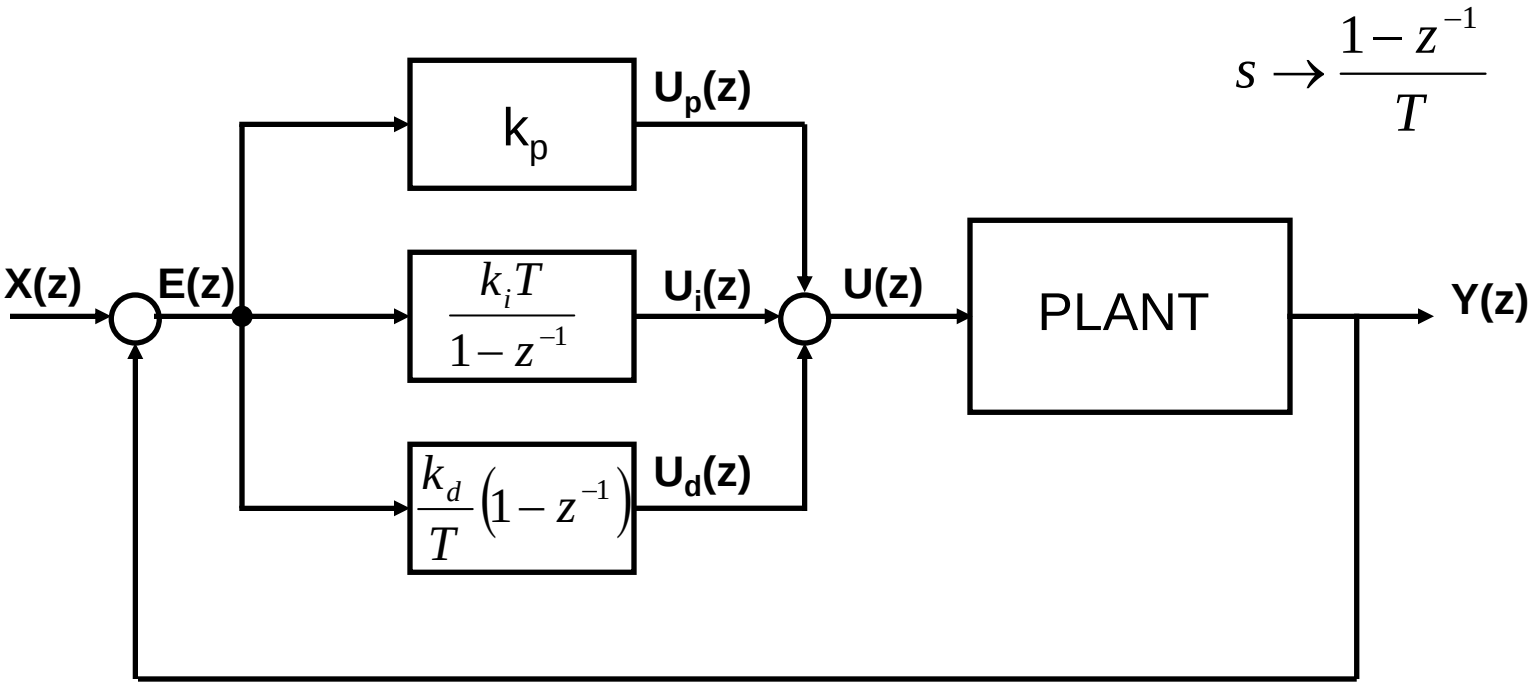
$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

$$U(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$



PID Control Loop

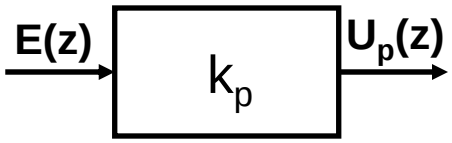
PID: Derivation from Analog Counterpart
(Z Transform)



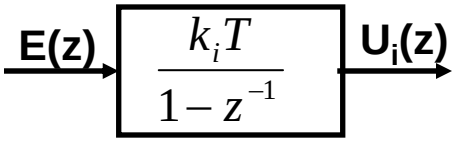
PID Control Loop

PID Equations

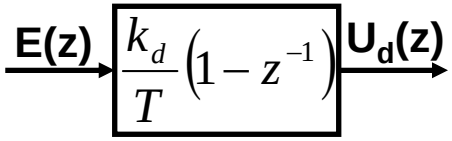
$$U_p(z) = k_p E(z)$$



$$U_i(z) = \frac{k_i T}{1 - z^{-1}} E(z)$$



$$U_d(z) = \frac{k_d}{T}(1 - z^{-1})E(z) \Rightarrow$$



$$U(z) = \left[k_p + \frac{k_i T}{1 - z^{-1}} + \frac{k_d}{T}(1 - z^{-1}) \right] E(z)$$

PID Control Loop

PID Equations

$$U(z) = \frac{\left(k_p T + k_i T^2 + k_d\right) - \left(k_p T + 2k_d\right) z^{-1} + k_d z^{-2}}{T\left(1 - z^{-1}\right)} E(z) \Rightarrow$$

$$U(z)\left(1 - z^{-1}\right) = \left[K_A + K_B z^{-1} + K_C z^{-2}\right] E(z)$$

where

$$K_A = k_p + k_i T + \frac{k_d}{T}; K_B = -(k_p + 2\frac{k_d}{T}); K_C = \frac{k_d}{T}$$

PID Control Loop

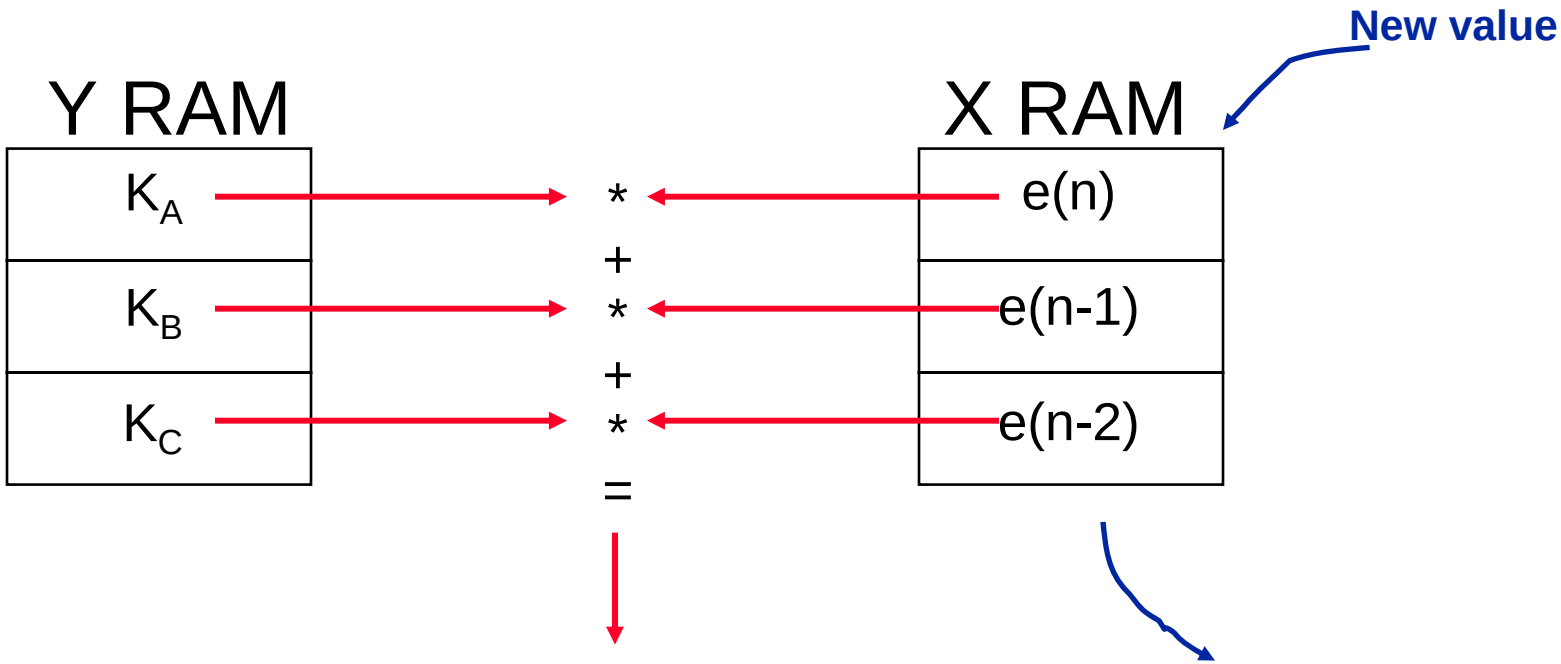
PID Equations

$$U(z)\left(1 - z^{-1}\right) = \left[K_A + K_B z^{-1} + K_C z^{-2}\right] E(z)$$

$$\Rightarrow$$

$$u(n) = u(n-1) + K_A e(n) + K_B e(n-1) + K_C e(n-2)$$

PID Control Loop



$$u(n) = u(n-1) + K_A e(n) + K_B e(n-1) + K_C e(n-2)$$

PID Control Loop

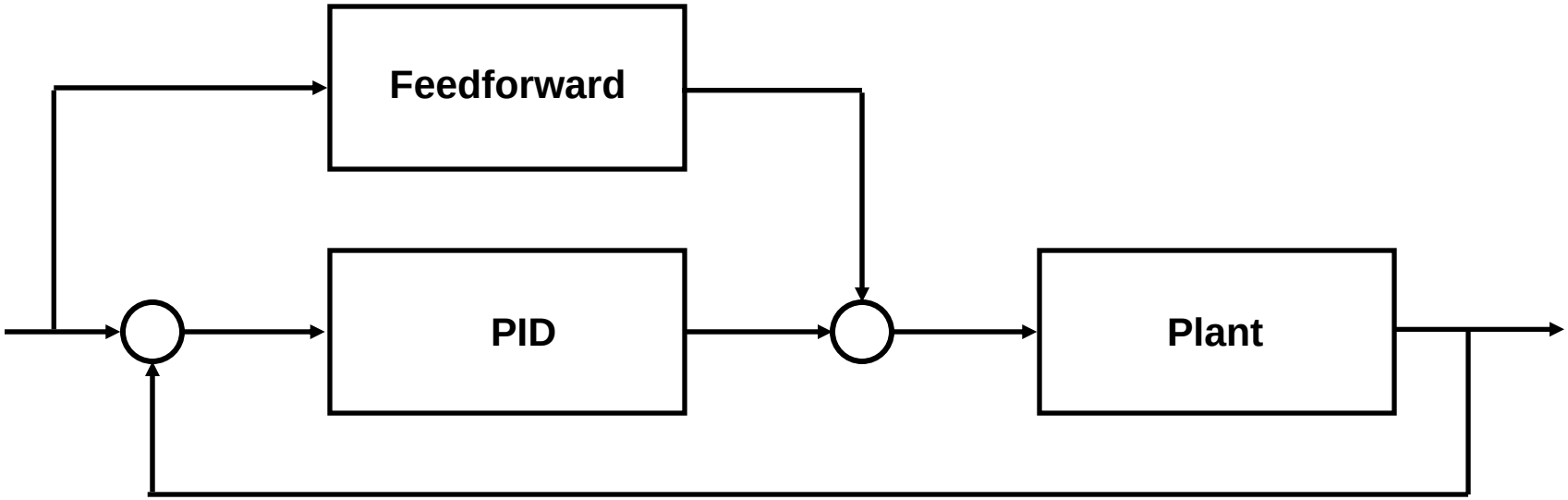
Influence of P,I,D on Performances

Closed Loop Response	Rise Time	Overshoot	Settling Time	Steady-State Error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	Small Change

PID Control Loop

Improvements to the Basic PID Design

FeedForward



PID Control Loop

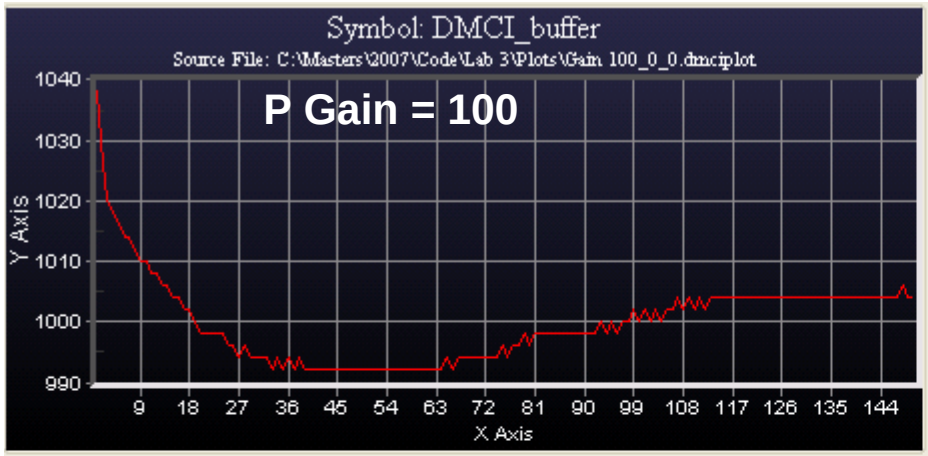
Additional Improvements

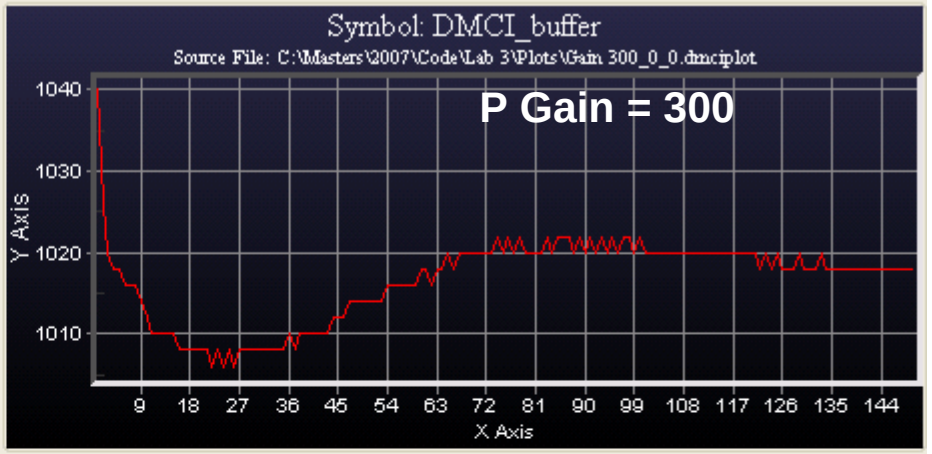
Also consider as errors:

- Second derivative (jerk): improves transient response
- Excess Current: reduce command if limit exceeded
- Dead Time Delay: removes offset caused by dead-time

PID Control Loop

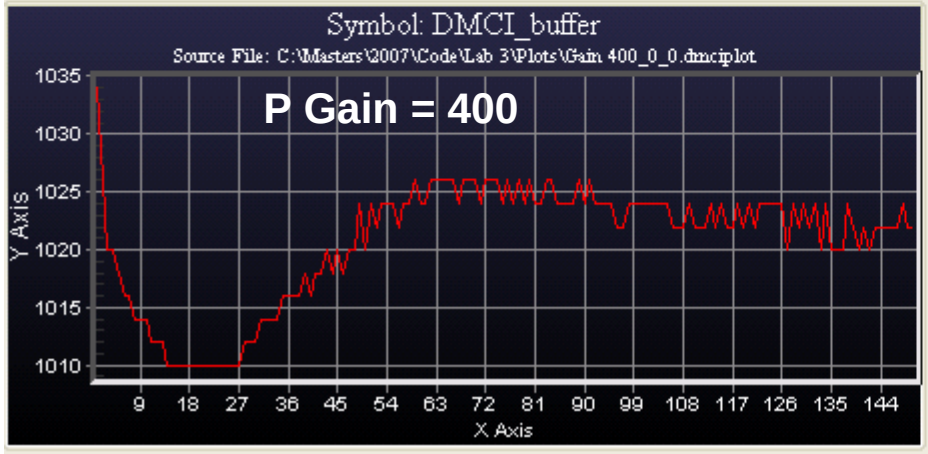
Proportional Gain Effects

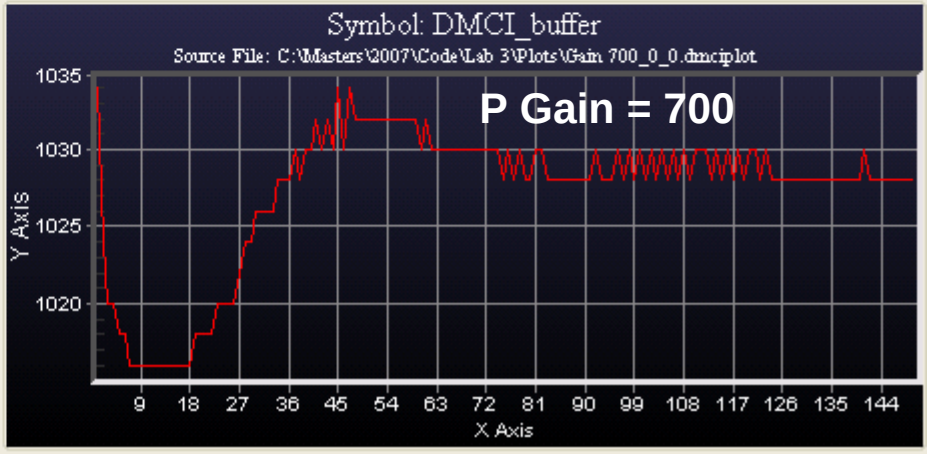




Graph 3

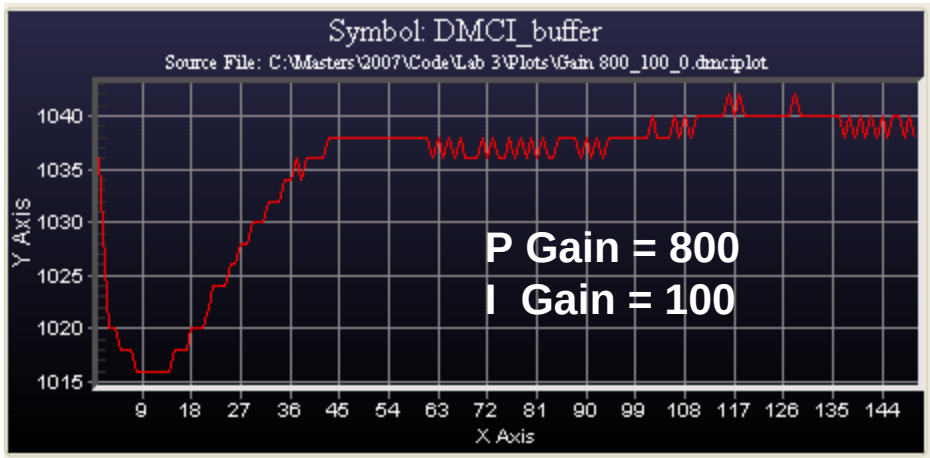
Graph 4

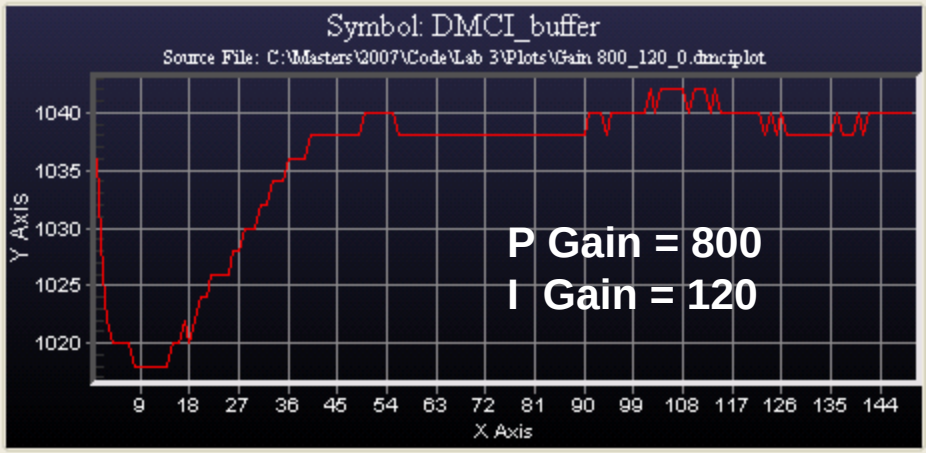




PID Control Loop

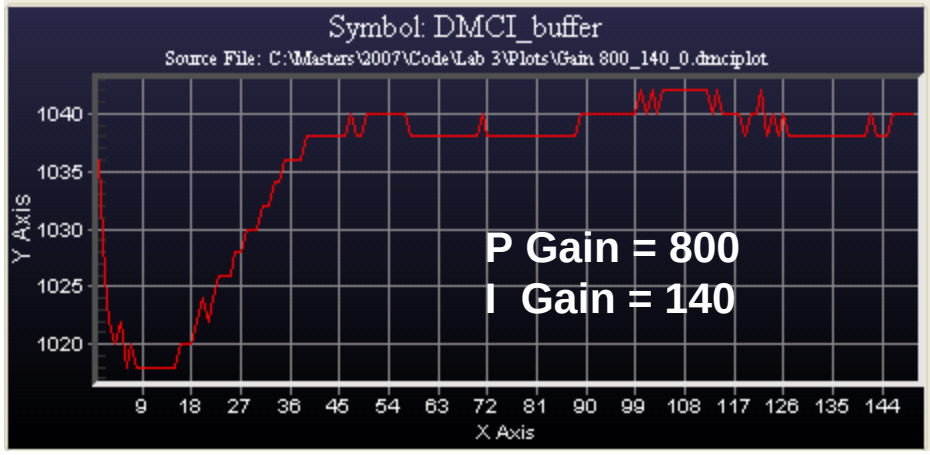
Integrative Gain Effects

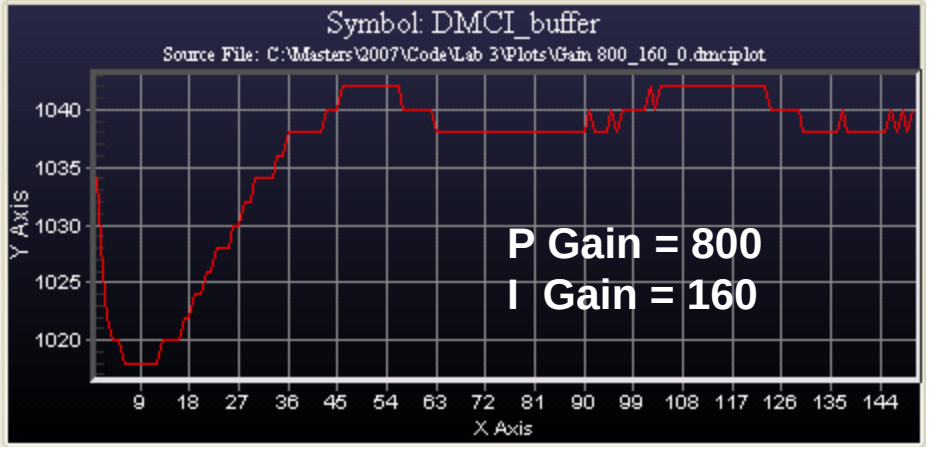




Graph 3

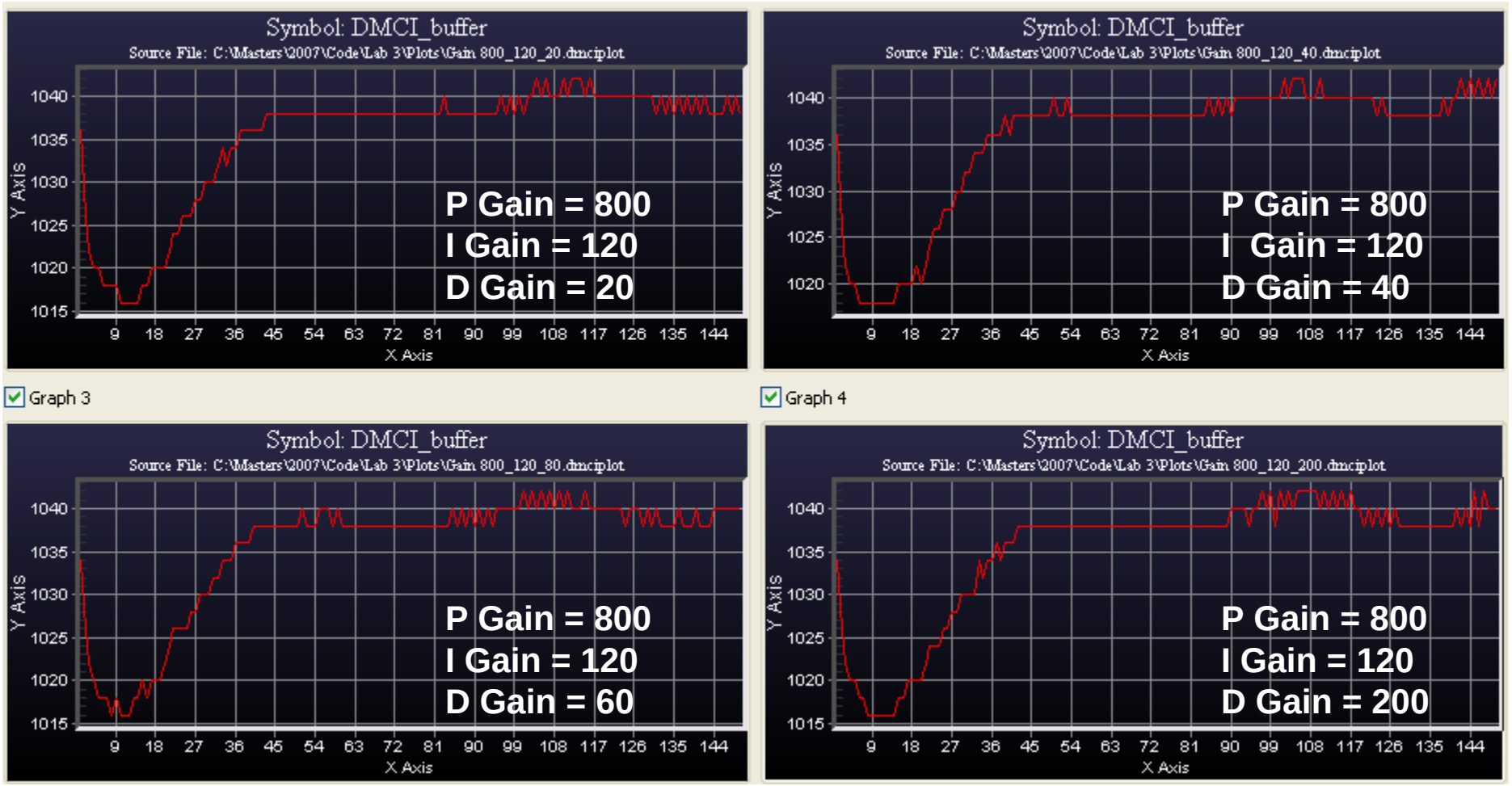
Graph 4





PID Control Loop

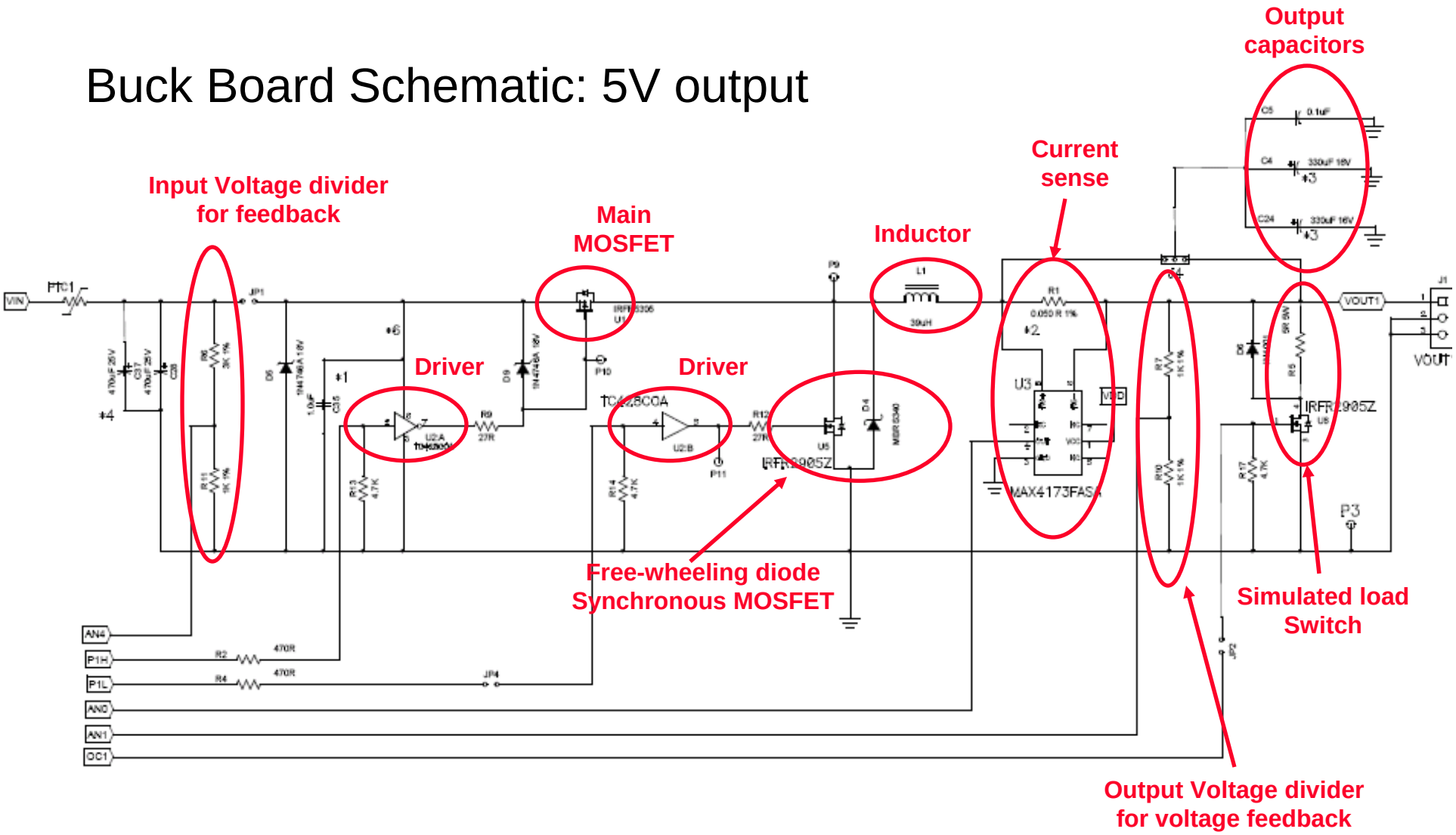
Derivative Gain Effects



Buck Board Hardware

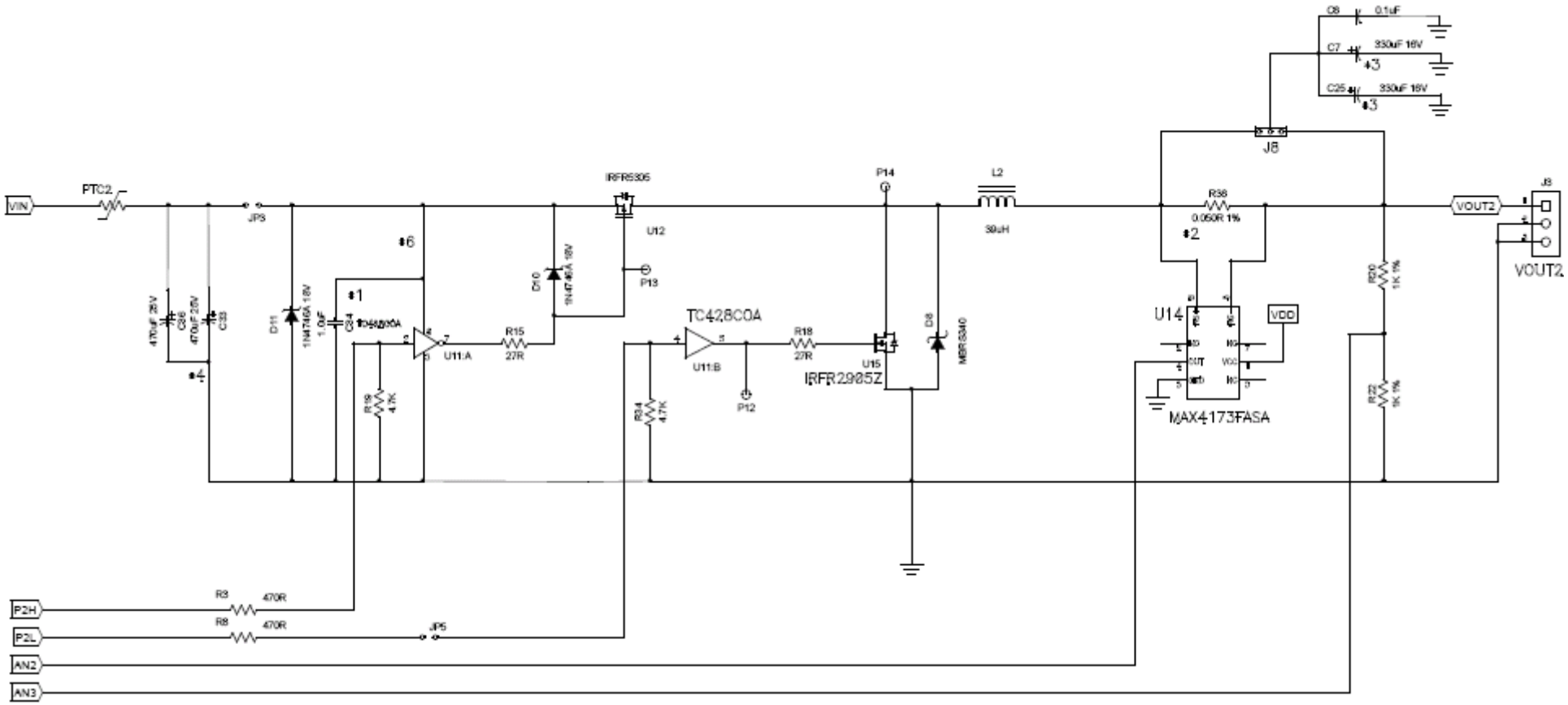
Buck Board Hardware

Buck Board Schematic: 5V output



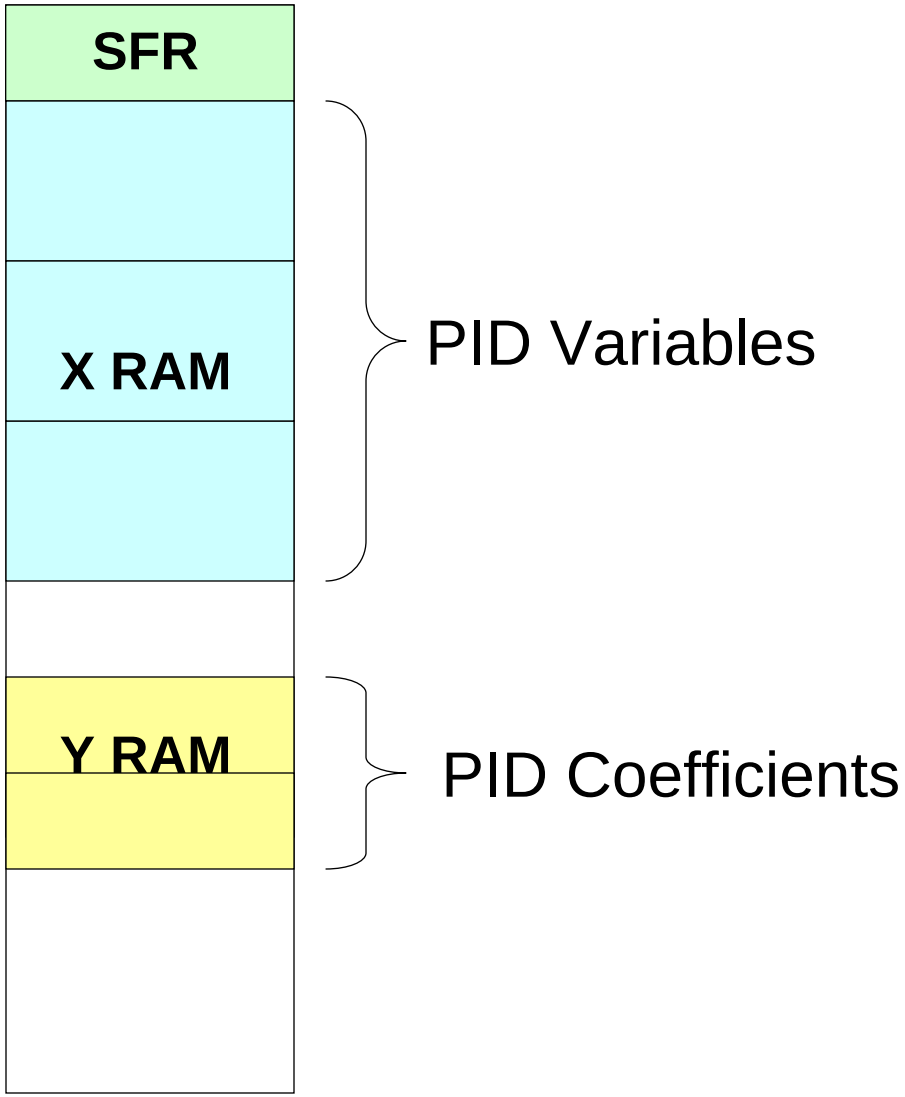
Buck Board Hardware

Buck Board Schematic: 3.3V output



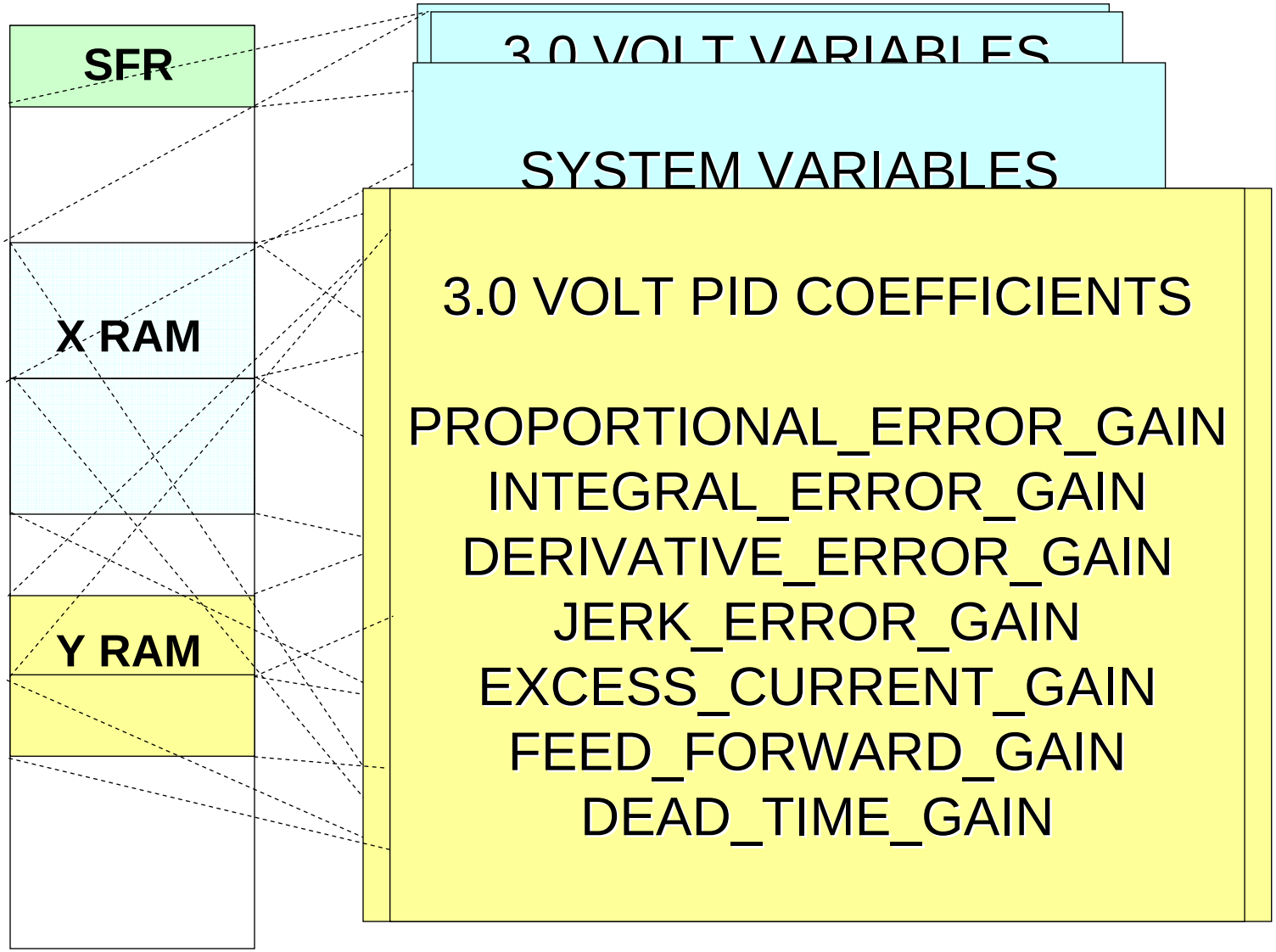
Buck Board Firmware

Buck Board Firmware



Variable Allocation in RAM

Buck Board Firmware



Variable Allocation in RAM

Buck Board Firmware

PID computations performed in main idle loop

EXCESS-CURRENT =

ACTUAL CURRENT – MAXIMUM ALLOWED CURRENT
(If there is an excessive current issue, do not update integral error)

FD FWD VOLT CMD =

Vout = Vin * DUTY_CYCLE

Buck Board Firmware

PID computations performed in ADC interrupt function

PROPORTIONAL ERROR =

COMMANDED OUTPUT – CURRENT OUTPUT VOLTAGE

INTEGRAL ERROR =

PROPORTIONAL ERROR > 0 ⇨ INCREMENT INTEGRATOR

PROPORTIONAL ERROR < 0 ⇨ DECREMENT INTEGRATOR

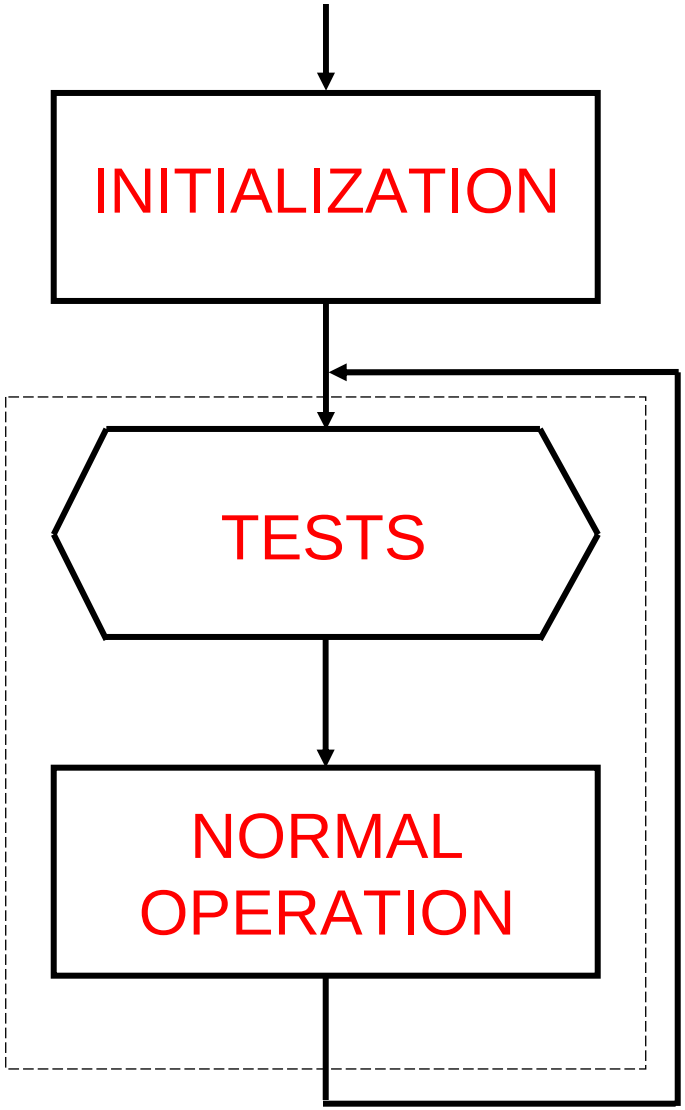
DERIVATIVE ERROR =

CURRENT VOLTAGE ERROR – PREVIOUS VOLTAGE ERROR

JERK ERROR =

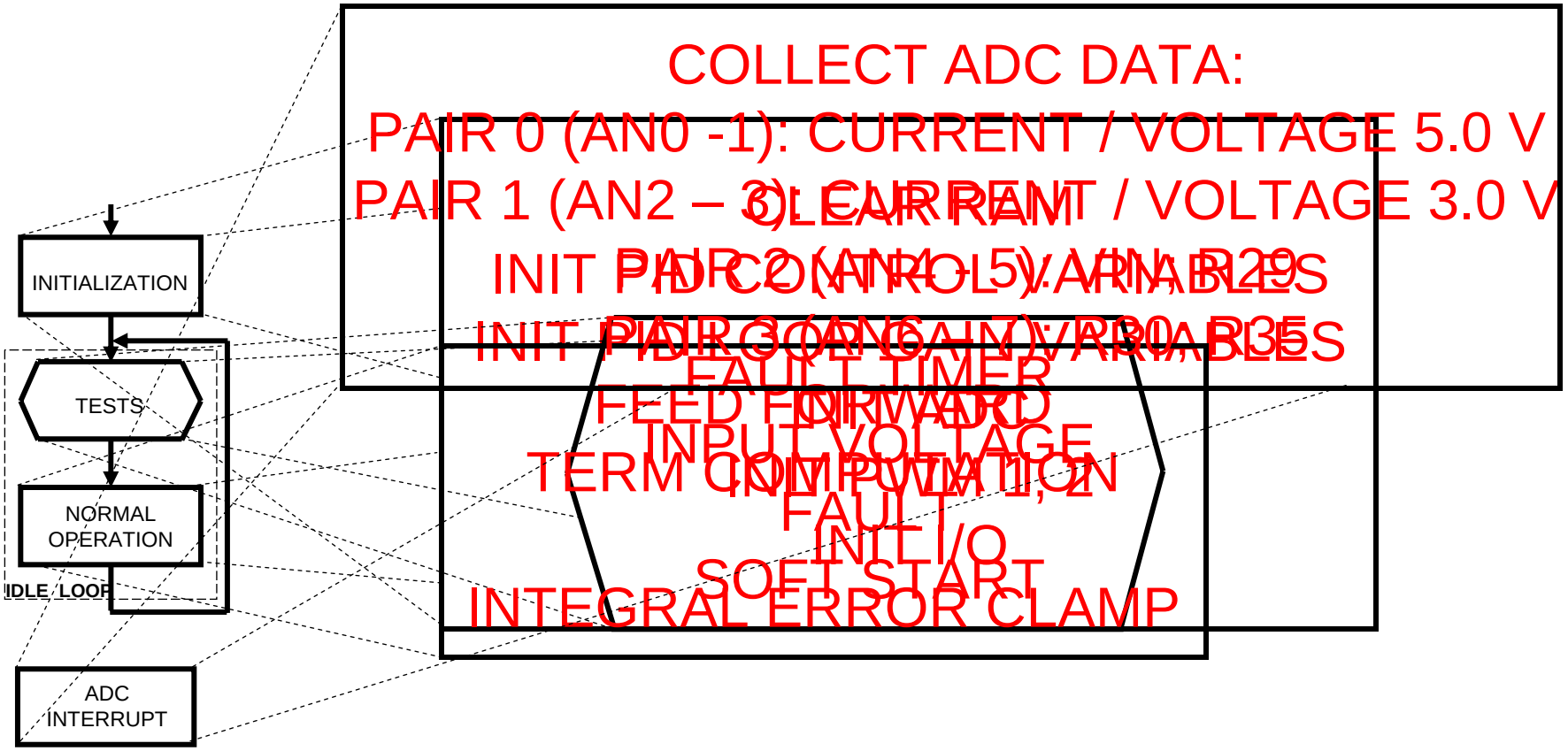
CURRENT DERIVATIVE ERROR – PREVIOUS DERIVATIVE ERROR

Buck Board Firmware



ADC INTERRUPT

Buck Board Firmware



Buck Board Firmware

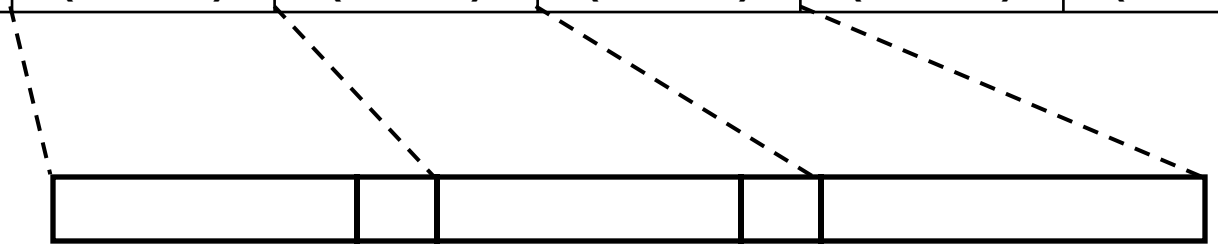
A voltage control loop is implemented

PID OUTPUT =

PROPORTIONAL_ERROR	*	PROPORTIONAL_ERROR_GAIN	+
INTEGRAL_ERROR	*	INTEGRAL_ERROR_GAIN	+
DERIVATIVE_ERROR	*	DERIVATIVE_ERROR_GAIN	+
JERK_ERROR	*	JERK_ERROR_GAIN	+
EXCESS_CURRENT	*	EXCESS_CURRENT_GAIN	+
FD_FWD_VOLT_CMD	*	FD_FWD_VOLT_CMD_GAIN	+
DEAD_TIME	*	DEAD_TIME_GAIN	

Buck Board Firmware

PWM Period #	PWM #1	PWM #2	PWM #3	PWM #4	PWM #5	PWM #6
Control Loop on	5V buck	3.3 V buck	None	5V buck	3.3 V buck	None
Exec time for control loop	2.0 us	2.0 us	0	2.0 us	2.0 us	0
Period length	2.5 us (400 Khz)	2.5 us (400Khz)	2.5 us (400 Khz)	2.5 us (400 Khz)	2.5 us (400 Khz)	2.5 us (400 Khz)



- 4us out of 7.5 are used in the loop computation (53%)
- There is enough room for the main loop

PID Control Loop

- Summary. We have seen:
  - What a PID is
  - What are its performances
  - How the PID has been inmplemented in the software of the Buck Demo Board
  - What is the associated timing

Lab 3

Lab 3

You can experience how the PID coefficients influence the system behavior

- Start with I and D gain = 0
- Increase P gain until you get an output value close to the desired one
- Increase I gain in order to decrease the residual error
- Increase D gain to reduce overshoot and settling time

Sequencing

Sequencing

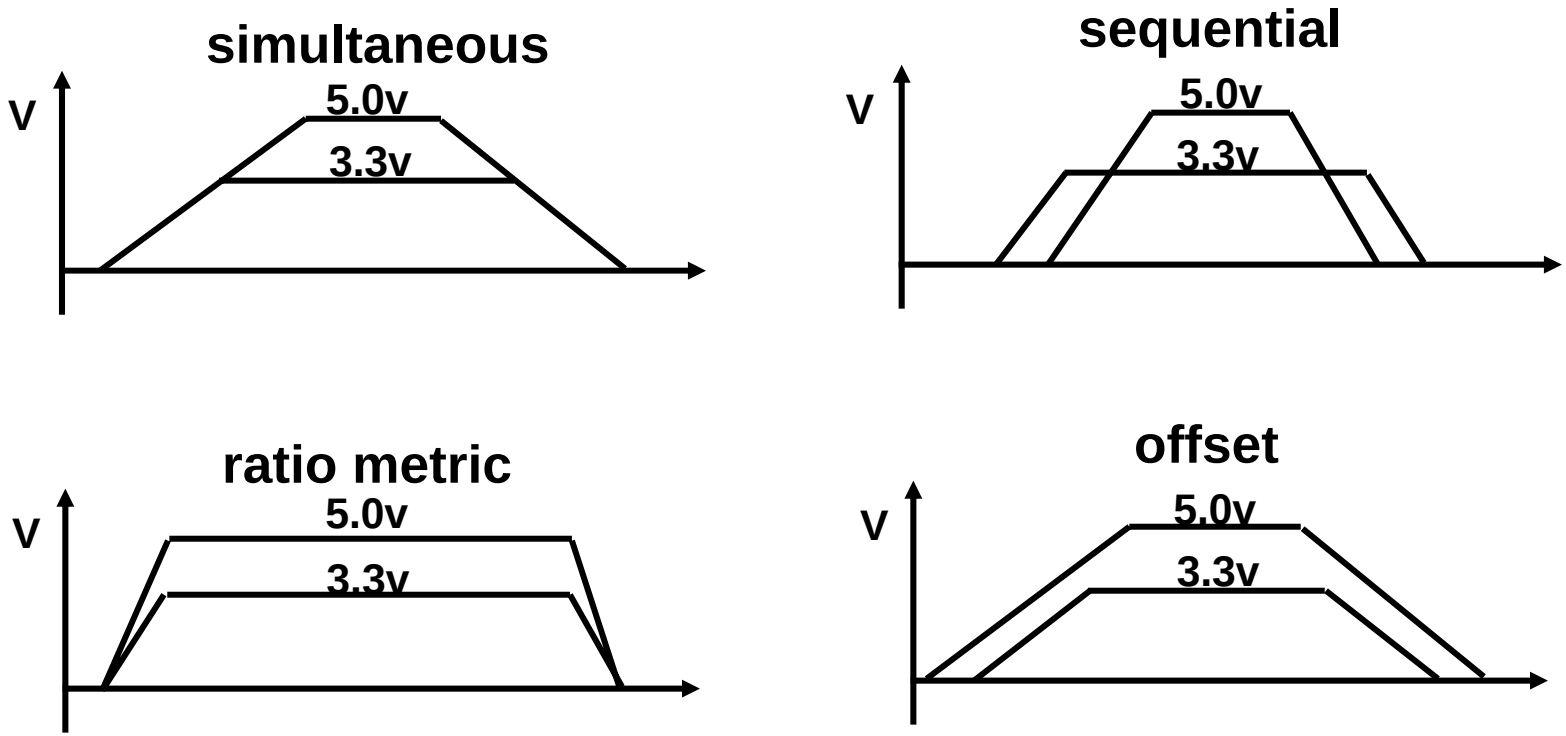
- Sequencing is the process of generating two or more voltages with a specific profile and a specific time relation between them
- Many electronic devices require their multiple supply voltages be coordinated at power on and off to protect their circuitry
- If the supply voltages are improperly applied, many integrated circuits can experience “Latch-up”

Sequencing

- Other systems require that specific circuitry be powered prior (to put it in a known safe state) to the rest of the load circuitry
- The most common power supply sequencing method is the simultaneous ramp up and down
- The choice of power sequencing is very dependent on the system requirements

Sequencing

Choose the method that meets system requirements



Dev Tools Used in This Class:

dsPICDEM™ SMPS Buck Board (DM300023)

MPLAB® ICD2 In-Circuit Debugger (DV164005)

Thank You

UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLoq, KeeLoq logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.