



845 SNS

Sensor and Sensorless Control of a BLDC Motor Using a dsPIC[®] DSC Hands-on Class



Agenda

- BLDC Motor Basics
- dsPIC[®] DSC Peripherals for BLDC control - PWM and ADC
 - lab 1 - “How to get started” by blinking an LED
 - lab 2 - using the ADC
 - lab 3 - using the PWM and multiple ADC channels
- Control of a BLDC motor with Position Sensors
 - Lab 4 - Spinning a Sensored BLDC Motor
- Review of BLDC Sensorless Techniques
- The Back EMF “zero crossing” technique
- Implementation on the dsPIC[®] DSC (AN901)
- Commutation and other control loops
 - Lab 5 - Setting Up and Tuning the Sensorless BLDC Reference Design



Learning Objectives

- dsPIC[®] DSC Peripherals used for BLDC Control:
 - 10-bit ADC
 - Motor Control PWM
- BLDC Basics
- BLDC Control Techniques
 - Sensored Techniques
 - Sensorless Techniques
- Write Code to Spin a Sensored and Sensorless BLDC Motor



BLDC Motor Basics

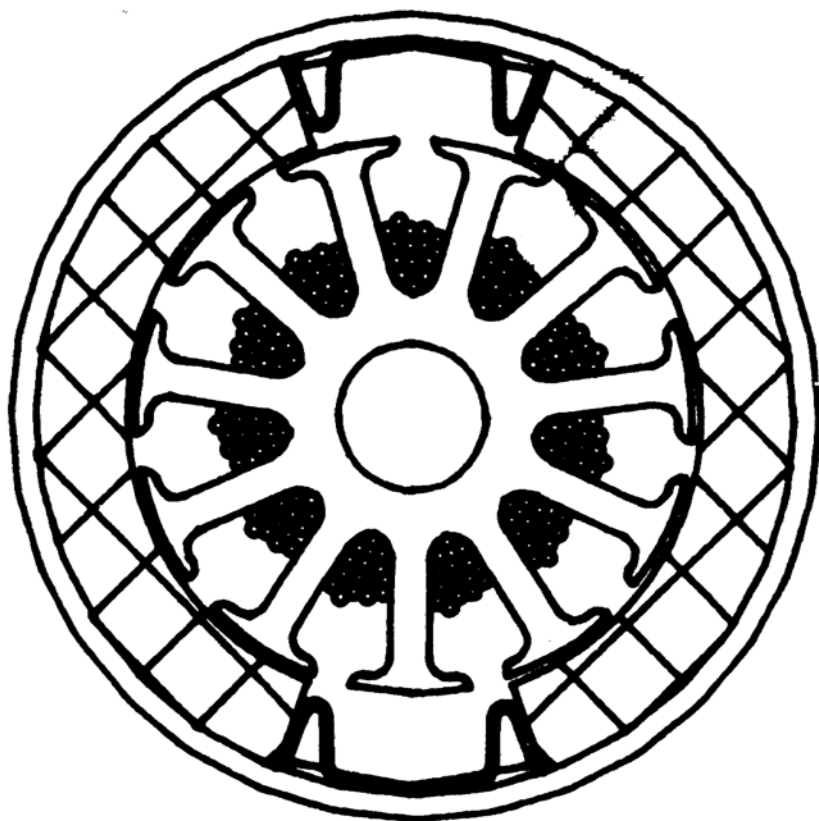


The BrushLess DC (BLDC) Motor

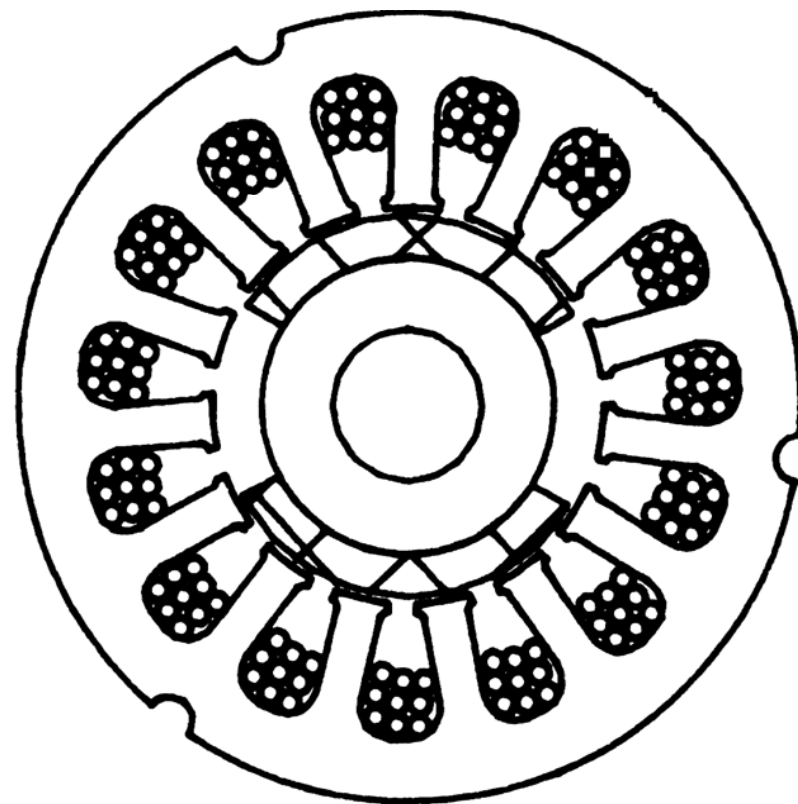
- An inside out brushed DC motor with electronic commutation
- A modern, much improved, version of the traditional brushed DC motor.
- Field, which has relatively low loss, is generated on the rotor using permanent magnets.
- Armature, which causes the majority of the loss, is on the stator which has good cooling.
- The Back EMF is usually somewhere between being sinusoidal and trapezoidal.

Brushed & Brushless DC Motor Construction

PERMANENT MAGNET BRUSHED DC MOTOR



PERMANENT MAGNET BRUSHLESS DC MOTOR





BLDC Advantages Over Brushed DC Motor

- Smaller size for same power owing, mainly, to the improved cooling of the armature.
- Operation to higher speeds.
- Lower inertia as no commutator or rotor windings.
- Therefore much higher acceleration rates possible.
- No brushes to maintain.
- No sparking on Commutator.



BLDC Control

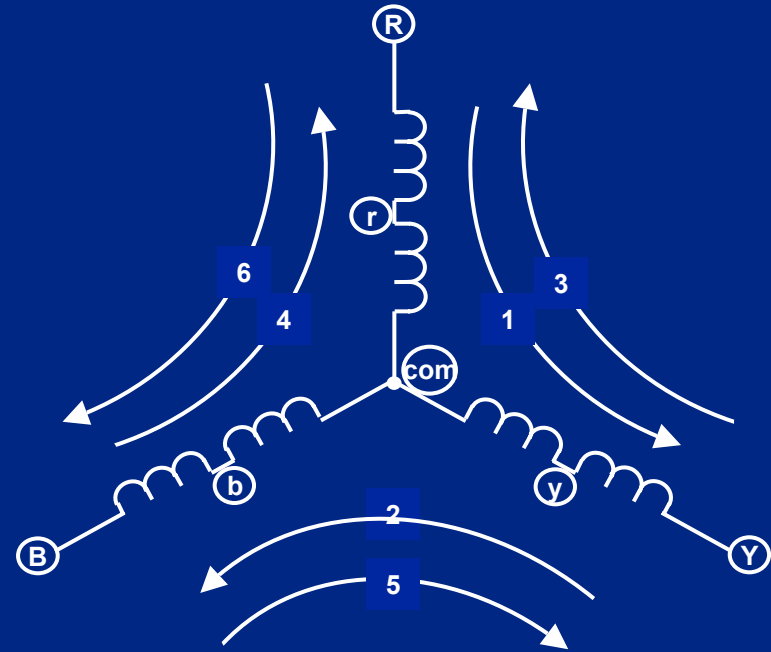
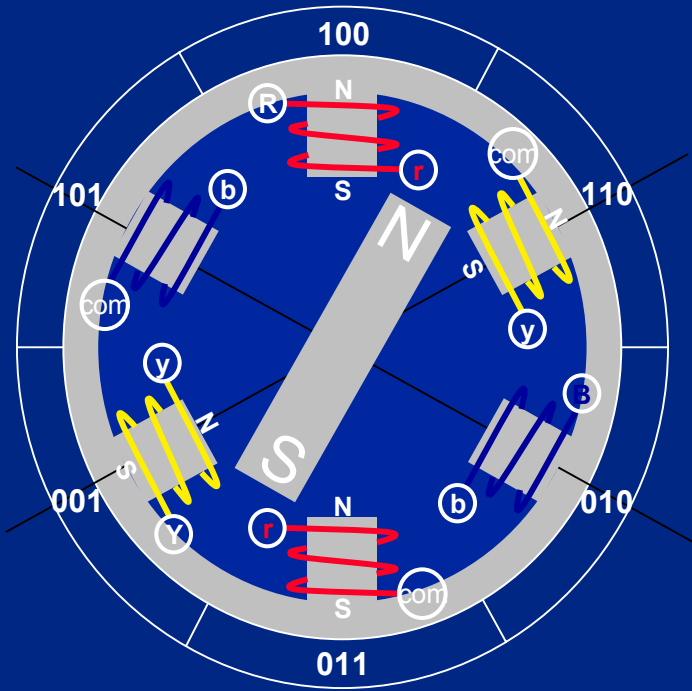
- Mechanical commutator of brushed motor replaced by electronic switching of the phases.
- Switching must be correctly synchronised to absolute rotor position to ensure smooth and efficient torque production.
- The BLDC is a synchronous motor with no damping or starting windings.
- All discussions will focus on Y connected 3 phase motors with 3 leg inverters and square wave (120°) conduction.



Standard BLDC Position Sensing

- A sensing disk is attached to the rotor which provides a ~ 50% duty pattern aligned to the rotor magnets. The repetition rate of the pattern will follow the number of rotor poles.
- The disk is monitored by three optical or hall sensors, displaced by the equivalent of 120° , located on the stator.
- In the case of hall sensors, the rotor magnets themselves may be sensed directly.

Brushless DC Motor Construction

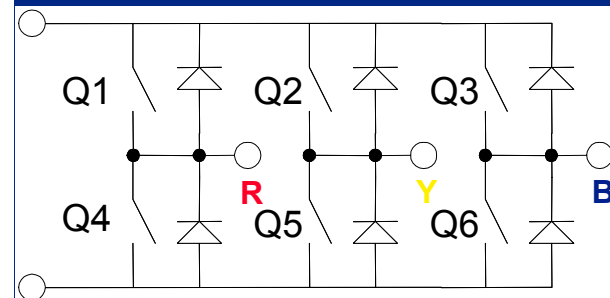
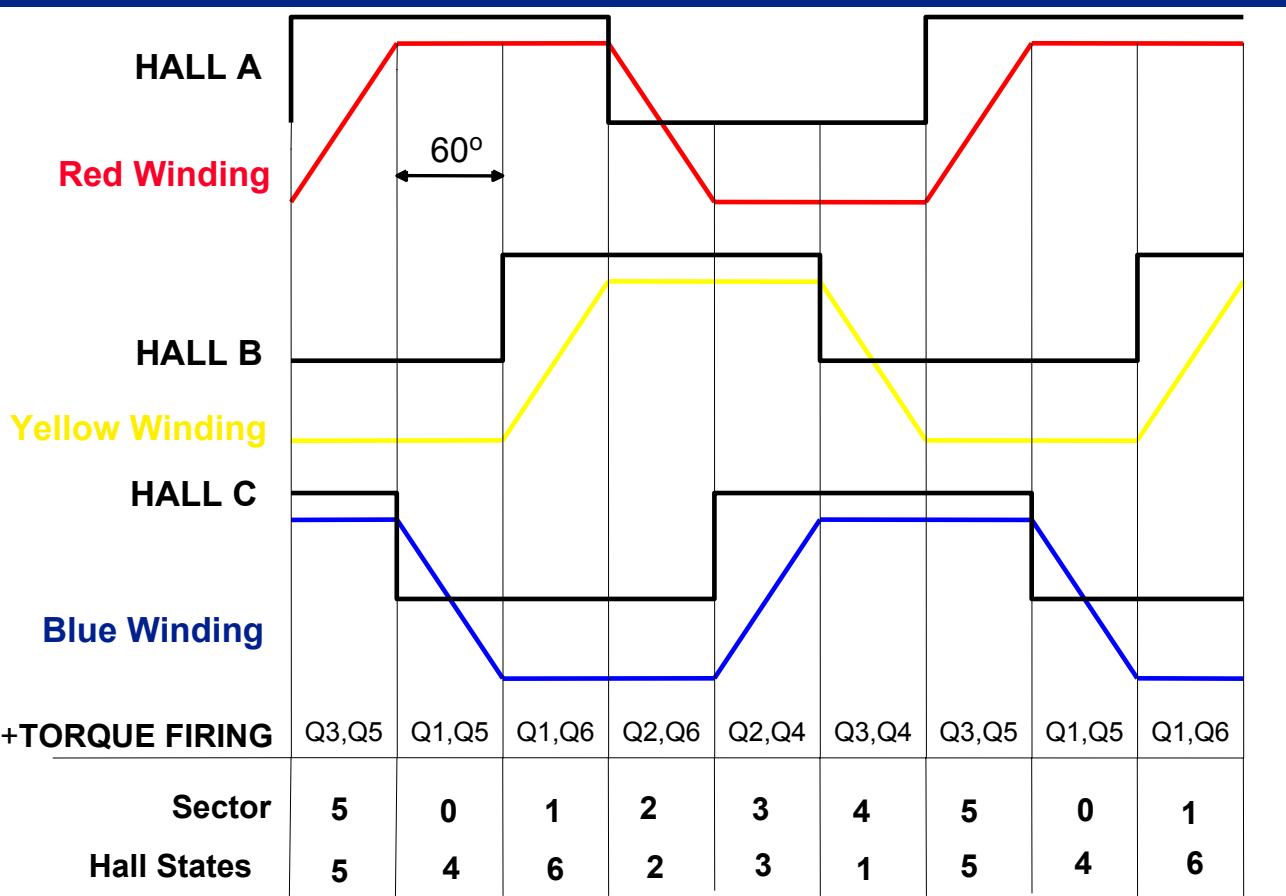




Standard Sensored BLDC Control

- The 3 logic signals are decoded to determine which windings should be energized.
- There are 6 valid states and 2 states that should never be seen (000, 111).
- Use Lookup table to drive the 3 windings, high or low or no-drive.
- The 6 different valid states directly translate to the 6 different 60° electrical cycle sectors.
- The states should only transition by one at a time. Missing transitions or invalid states should be detected for robust operation.

Standard BLDC Control





16-bit Digital Signal Controller Architecture



**MICROCHIP
M A S T E R S**

dsPIC30F Controller Family

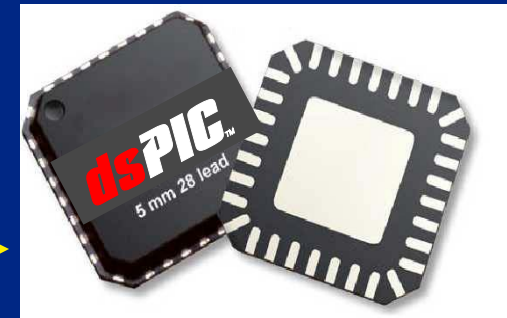
| | |
|--------------------------|--------------------------------------|
| Flash Program Memory: | 12 K - 144 K Bytes |
| RAM | 512 - 8 K bytes |
| Data EEPROM | 1K - 4K Bytes |
| Pin-count | 18 - 80 pins |
| Timers 16-bit | Up to 5 |
| Input Capture | Up to 8 |
| Output Compare / PWM | Up to 8 (individual time bases) |
| Motor Control PWM | 6 or 8 with shutdown pins |
| A/D converter | 10-bit, 500 KSPS, up to 16 ch |
| A/D converter | 12-bit, 100 KSPS, up to 16 ch |
| UART | 1 - 2 |
| SPI™ (8/16-bit) | 1 - 2 |
| I ² C™ | 1 |
| QEI | 1 |
| CODEC interface | 1 |
| CAN | 1 - 2 |

Operating Parameters

- ➔ Operating Speed @ 5V (-40°C to 85°C) : 30 MIPS
- ➔ VDD: 2.5 to 5.5V
- ➔ Temp: -40° C to 125° C
- ➔ Program Memory: Flash
- ➔ Data Memory: SRAM, EEPROM
- ➔ Analog: 10-bit & 12-bit Precision

- ➔ Package sizes

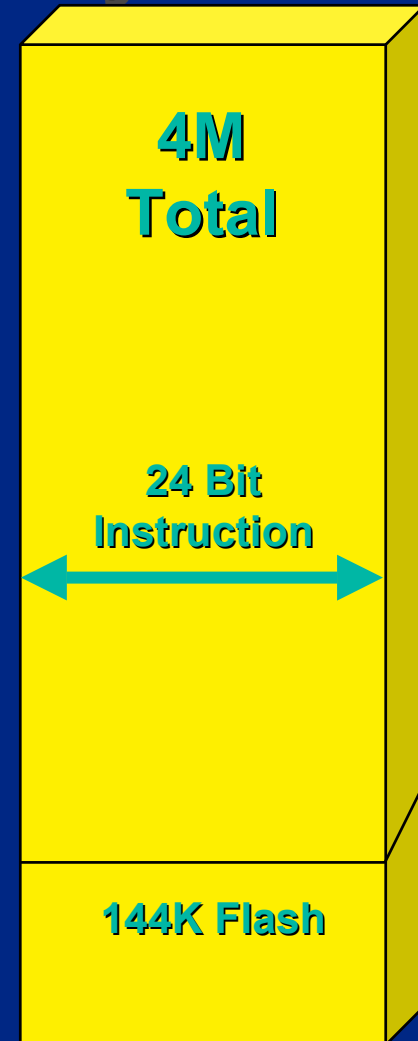
- 18-pin SO & SP
- 28-pin SO, SP and QFN
- 40-pin SP; 44-pin TQFP
- 64- and 80-pin TQFP





Program Memory

- Modified Harvard Bus Architecture
- Single Core : MCU + DSP
- Instruction is 24-bit wide
- Total Architecture Reach:
 - 4M x 24-bit Linear Program Space
- Devices contain up to 144K byte Flash Memory
 - No Paging or Segmentation





Data EEPROM Memory

- Up to 4K bytes Data EE Memory
- Run-Time programmable
- Row and Word erasable
- Row and Word programmable
- Modify a Row of 16 words in 2 milliseconds
- Can access Data EEPROM for 16-bit data read operations

```
MOV [++w4], [w6++]
```

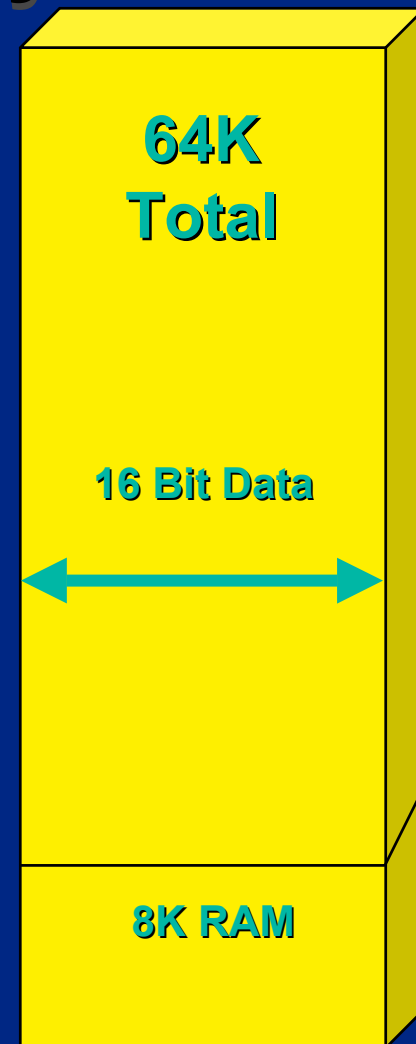


Data EE Memory used as source address for data read operation



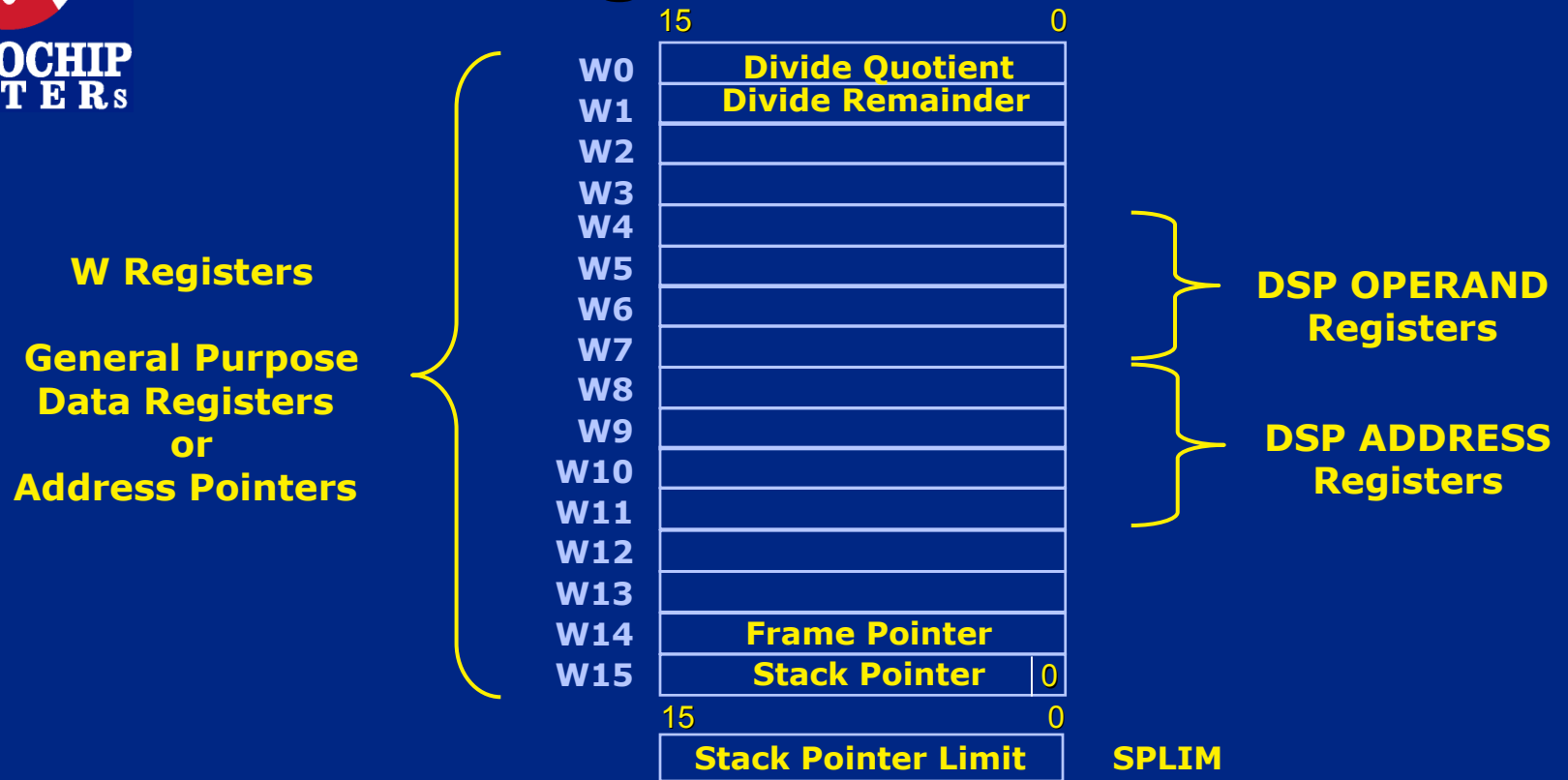
Data Memory

- Data is 16-bit wide
 - Byte addressable
- Total Space:
 - 64K Bytes Linear Data Space
 - No Banking
- Devices contain up to 8K Byte User RAM
- Addressable Indirectly or with Memory Direct '**MOV**' Instruction





Programmers Model



DSP Accumulators (40-bit)

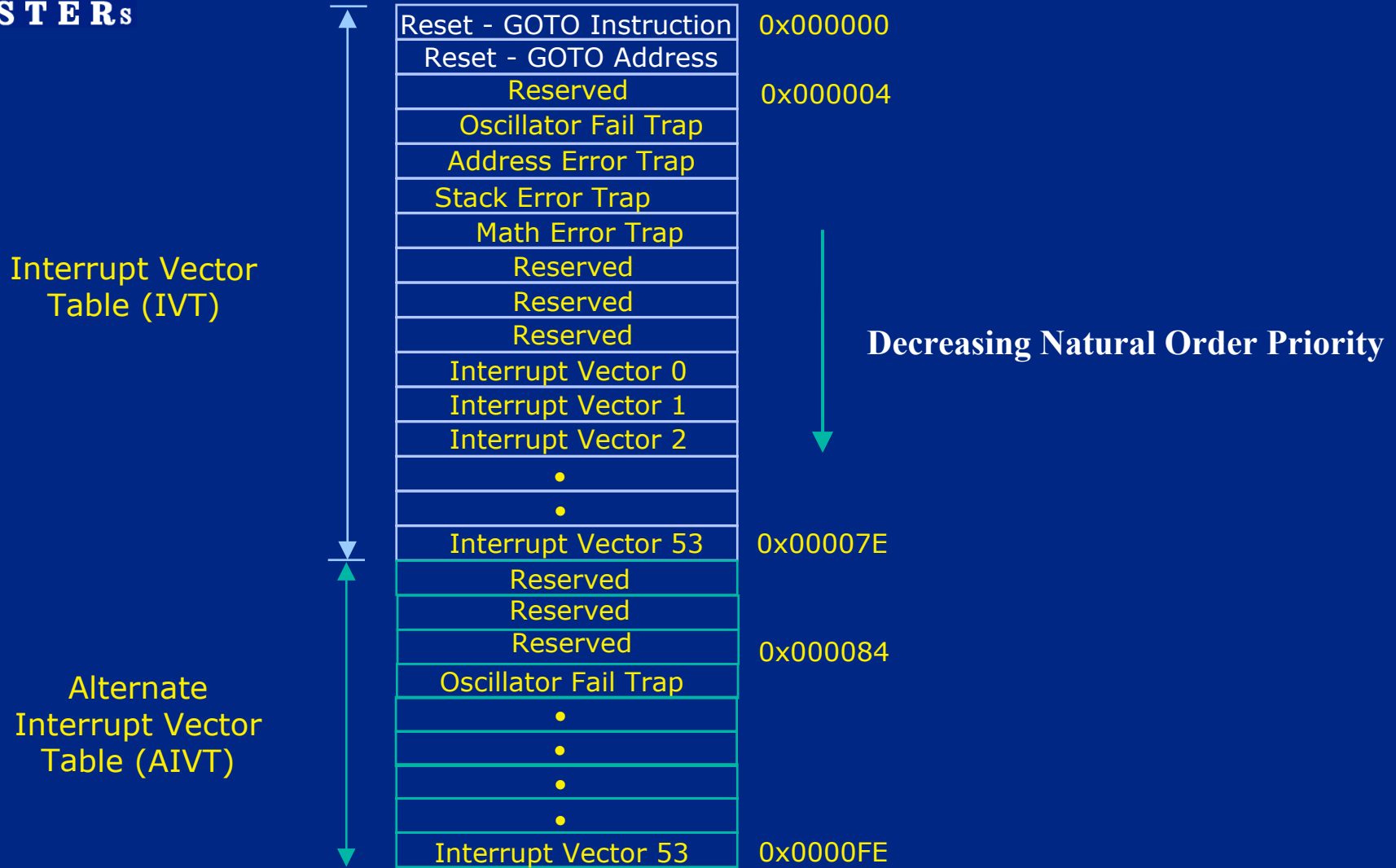




Interrupt Subsystem



Interrupt Vector Table





Traps for Robust Operation

- Oscillator Failure Trap (level 14)
- Address Error Trap (level 13)
 - Instruction fetch from illegal program space
 - Data fetch from unimplemented data space
 - Unaligned word access from data space
- Stack Error Trap (level 12)
 - Stack overflow or underflow
- Arithmetic Error Trap (level 11)
 - Divide by Zero
 - Unsaturated Accumulator Overflow (A or B)
 - Catastrophic Accumulator Overflow (either)
 - Accumulator Shift Overflow



System Management Features



System Management Features

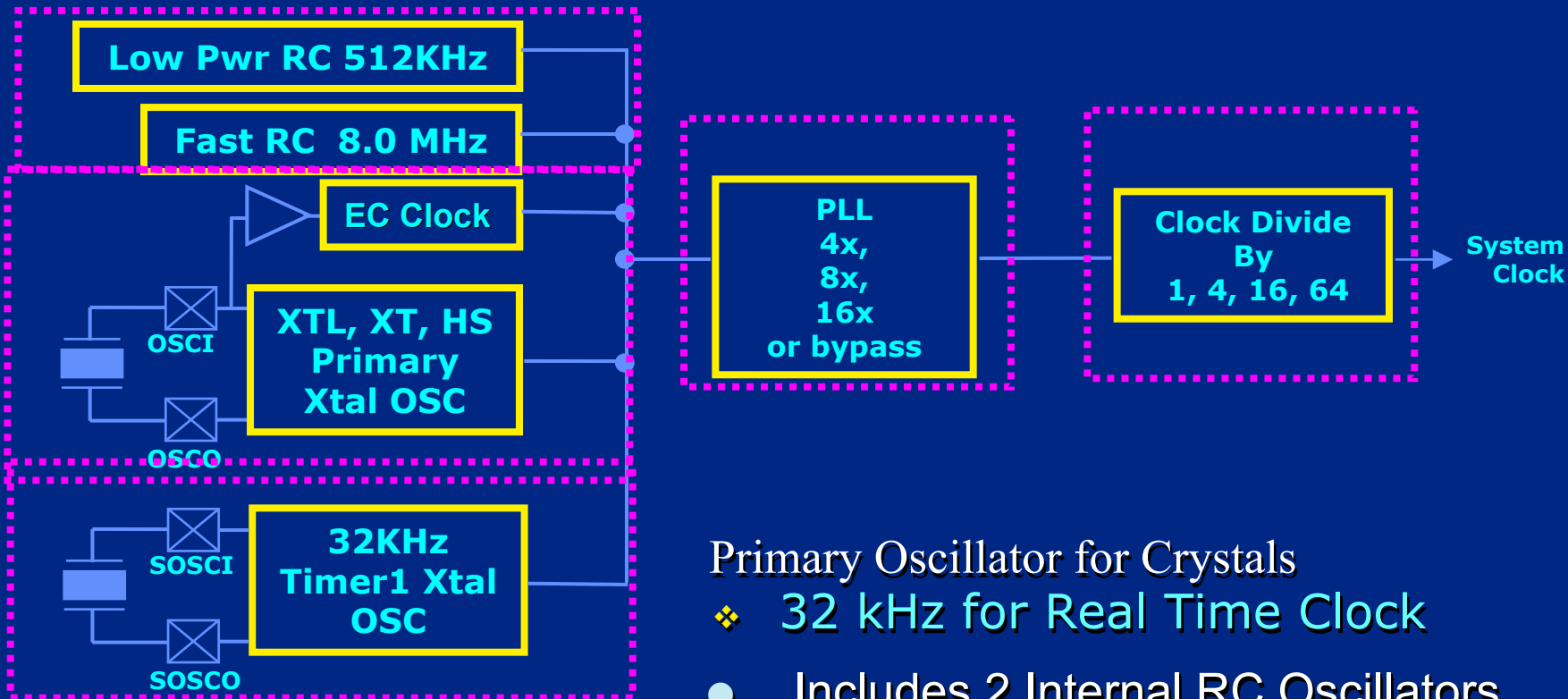
dsPIC[®] DSC has the same system management features that PIC[®] MCU users love

- ♥ Configurable **Watchdog Timer** with its own RC oscillator
Programmable Time out: 2 ms - 16 sec
- ♥ **Power-On-Reset** with a programmable delay 0, 4, 16, 64 ms
- ♥ **Brown-out Reset** with programmable levels
- ♥ **Low VDD Detect Interrupt** with programmable levels



**MICROCHIP
MASTERS**

Clock sources



Primary Oscillator for Crystals

- ❖ 32 kHz for Real Time Clock
 - Includes 2 Internal RC Oscillators
 - ❖ PLL multiplies oscillator source for high frequency operation
- Clock divide can optionally slow clock to conserve power



Fail-Safe Operation

- Features that make the target system safe!
 - Clock monitor detects oscillator failure
 - Automatic switch to an internal RC clock
 - Illegal Program Instruction
 - Device Resets
 - Traps let software handle error conditions
 - Oscillator Fail
 - Address out of range
 - Stack out of range
 - Math errors



LAB 1

- Getting Started with the dsPIC[®] DSC BLDC board
- Objectives for Lab:
 - Configure board hardware connections
 - Open a workspace in MPLAB[®] IDE
 - Compile or Build the project in MPLAB IDE
 - Follow procedure to Program the dsPIC DSC using ICD 2
 - Follow procedure to Run the program using ICD 2

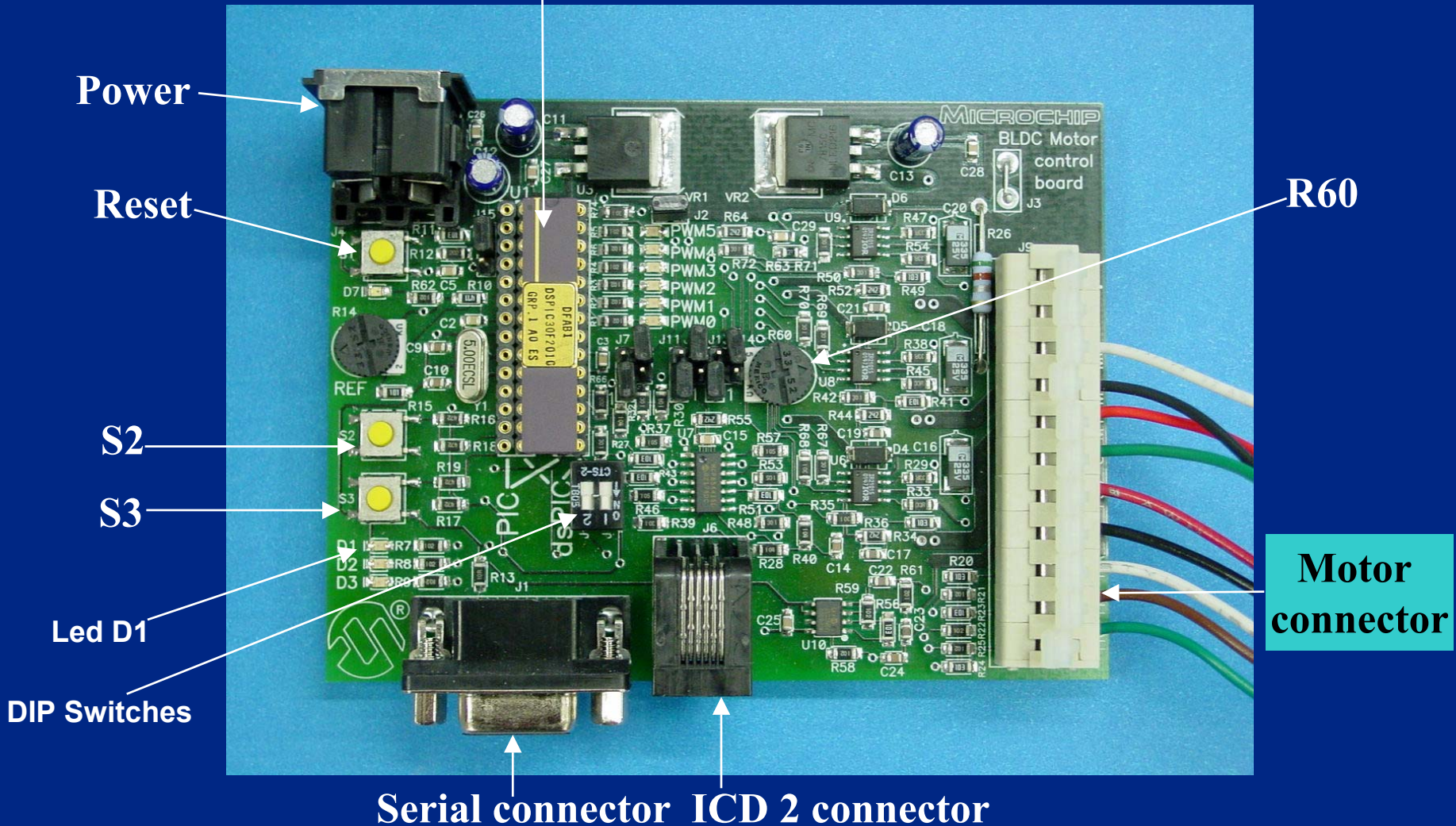


You should have....

- 1) MPLAB[®] IDE V6.5 or higher installed
- 2) Complete MPLAB[®] ICD 2 setup
- 3) BLDC motor control board(05-60001)
- 4) 24V power supply for the above board
- 5) Hurst(NTDynamo) BLDC motor with
 - Power cable (4 wires with white square connector) and
 - Hall sensor cable (5 wires with 8-pin inline connector)
- 6) Circuit schematic BLDC motor control board

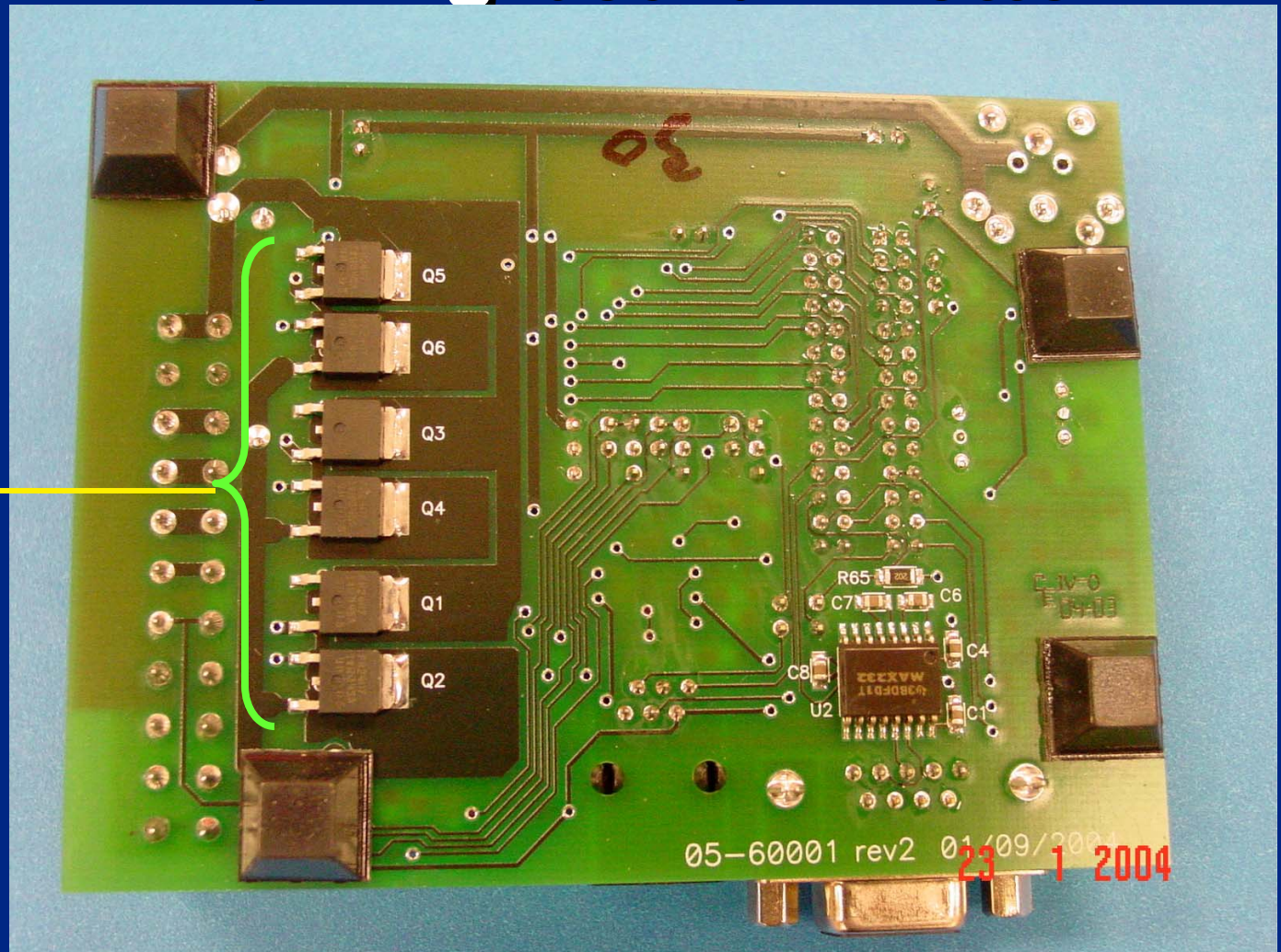
Training board - Top

dsPIC30F2010



Training board - Bottom

**Power
MOSFETs**



Jumpers and LEDs

dsPIC30f2010

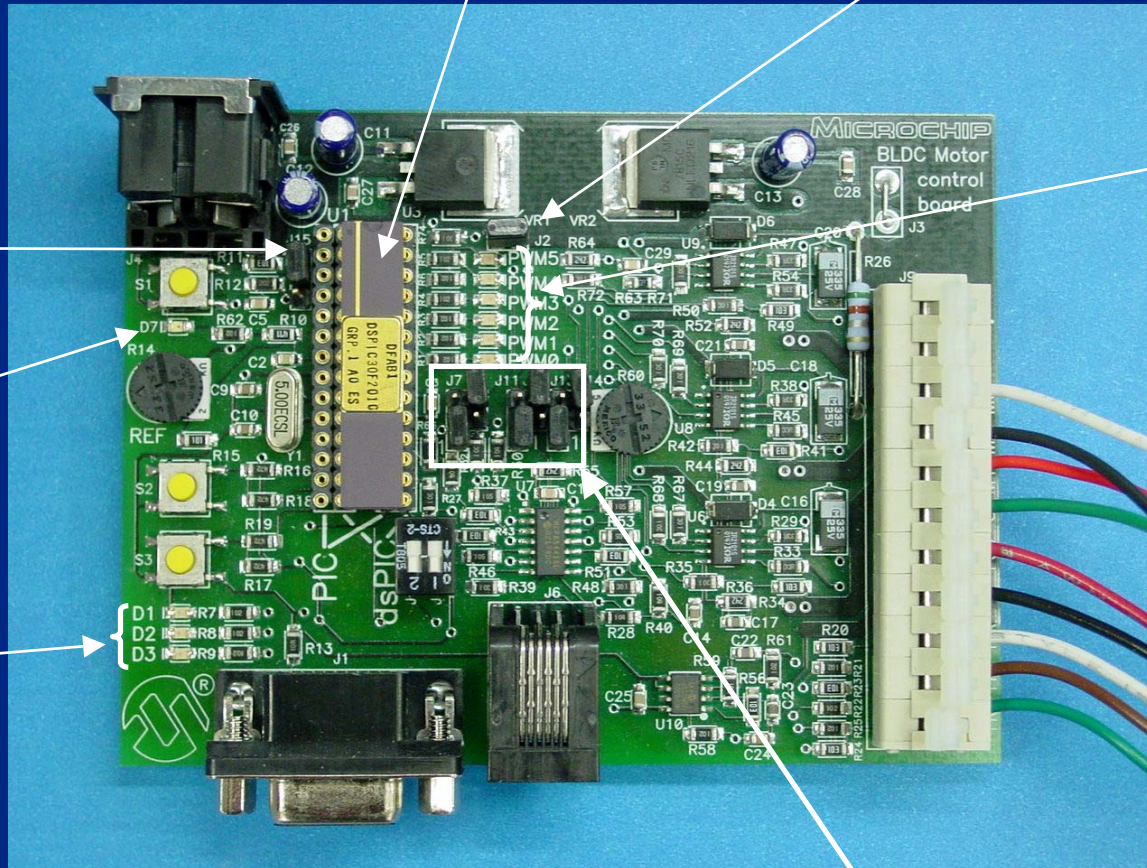
Jumper J2

LEDs
PWM0
to
PWM5

Jumper
J15

Power LED
D7

LEDs
D1
D2
D3



**Jumpers
J7, J8, J11, J12, J13, J14**



Default Jumper settings

- **J2**(2 pin) - shorted.
- **J15**(3 pin) - shorted between pin 2 & 3
(short link towards the crystal oscillator)
- **J7, J11 and J13** - shorted between pin 3 & 2
(short link away from ICD 2 connector)
- **JP8, JP12, JP14, JP16 and JP17** keep open
- Keep Potentiometer **REF(R14) and R60** in center position



Lab1

- Instructions for Lab1:
 - Make sure the DIP switches are in the CLOSE position (not OPEN)
 - Connect power to System
 - Open MPLAB[®] IDE by double clicking on icon
 - Select “File -> Open Workspace”
 - Browse to
“C:\MASTERS\845\Lab1\Lab1.mcw”
 - Select “Lab1.mcw” to open workspace

Contd ...



Lab 1 (contd.)

- Instructions for Lab1 (contd):
 - Select “Debugger -> Build All”
 - IF NO errors then ...
 - Select “Debugger -> Program” to program
 - Move DIP switches to OPEN position
 - Select “Debugger -> Run” and D1 should blink



Lab1 Results

- This same procedure has to be followed for programming and running software:
 - When programming, move DIPs to CLOSE position (not OPEN)
 - When running, move DIPs in OPEN position
- Each Lab has a workspace in the appropriate folder already created
- Use the workspace for each lab
- If D1 is blinking, your system is functional - Congratulations!!!!



10-bit A/D Converter

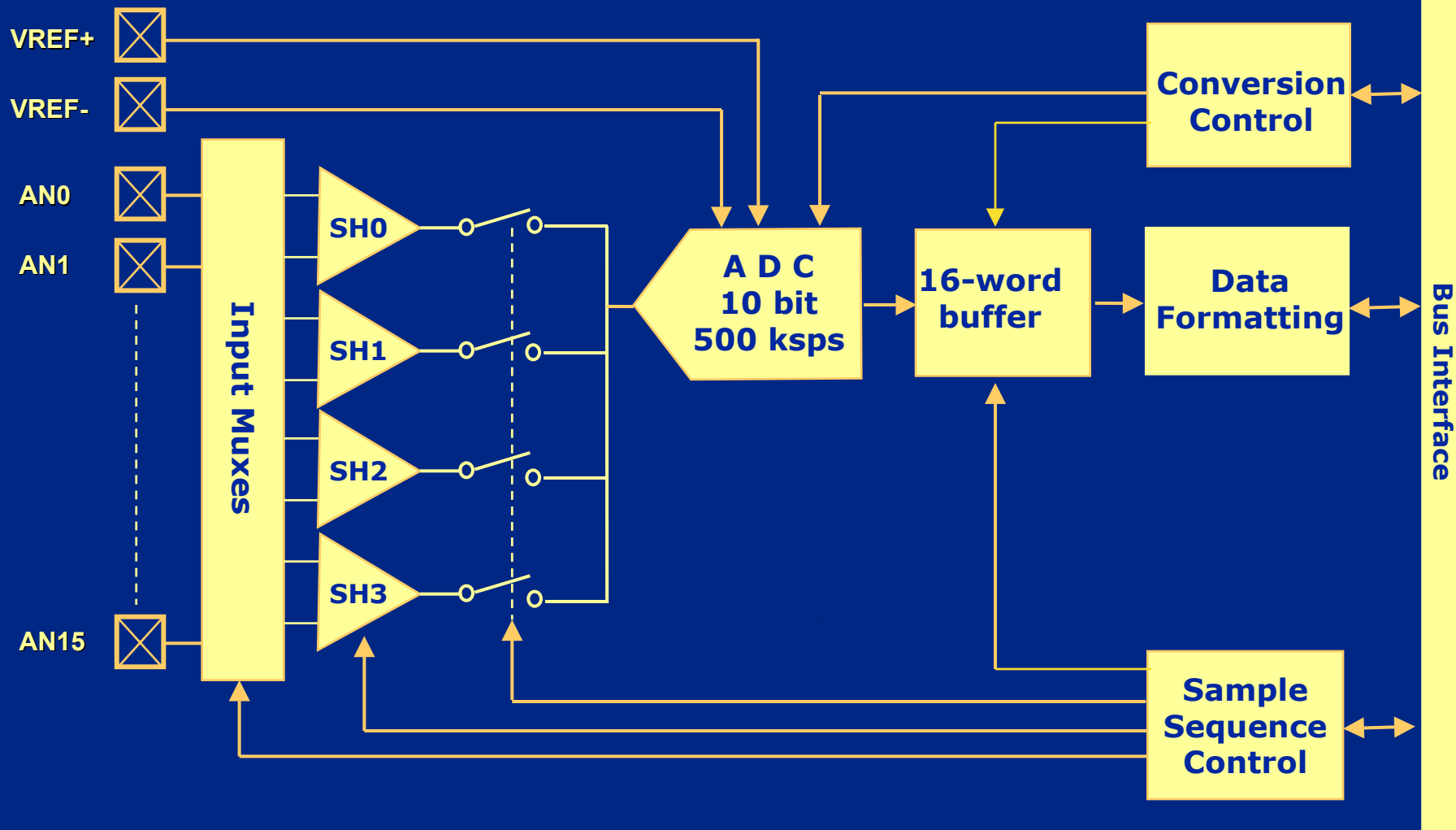


dsPIC™ 10-bit A/D Converter

- Feature Summary
 - 10-bit Resolution with +/- 1 bit accuracy
 - 500K Samples/Sec conversion rate
 - Up to 16 input channels, 4 S/H Amplifiers
 - Unipolar differential measurements
 - External VREF+ and VREF-
 - Many sampling sequences can be programmed
 - Multiple conversion trigger sources



10-bit A/D Block Diagram





10-bit A/D Converter

- A/D Configuration for Labs
 - Lab 2 - Sample one channel, manual sampling, manual conversion
 - Lab 3 - Simultaneously sample 4 channels, auto sample, trigger conversion from PWM
- Keep these configurations in mind as we look at the A/D control registers



10-bit A/D Converter

- ADCON1 Register Bits
 - DONE - A/D conversion status
 - SAMP - A/D sampling status/control
 - ASAM - Auto sample control bit
 - SIMSAM - Simultaneous sample control
 - SSRC - Conversion trigger source
 - FORM - Result format
 - ADSIDL - Stop in IDLE control bit
 - ADON - A/D enable bit (**set last**)



10-bit A/D Converter

- A/D Conversion Trigger Options
 - Auto convert (internal counter)
 - *Motor Control PWM special event trigger*
 - allows synchronized shunt current measurement
 - Timer3 period match event
 - Edge input on I/O pin
 - Manual trigger
 - user clears SAMP bit to begin CONV



10-bit A/D Converter

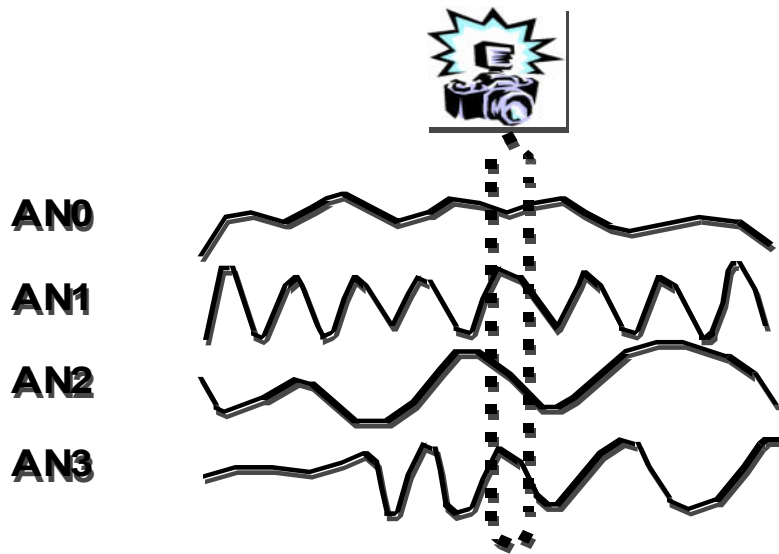
- **ADCON2 Register Bits**
 - **ALTS** - Alternate MUX sample mode
 - **BUFM** - Buffer mode control bit
 - **SMPI** - Samples per interrupt
 - **BUFS** - Buffer status
 - **CHPS** - S/H Channels per Sample
 - **CSCNA** - MUX A Scan enable bit
 - **VCFG** - Voltage Ref configuration



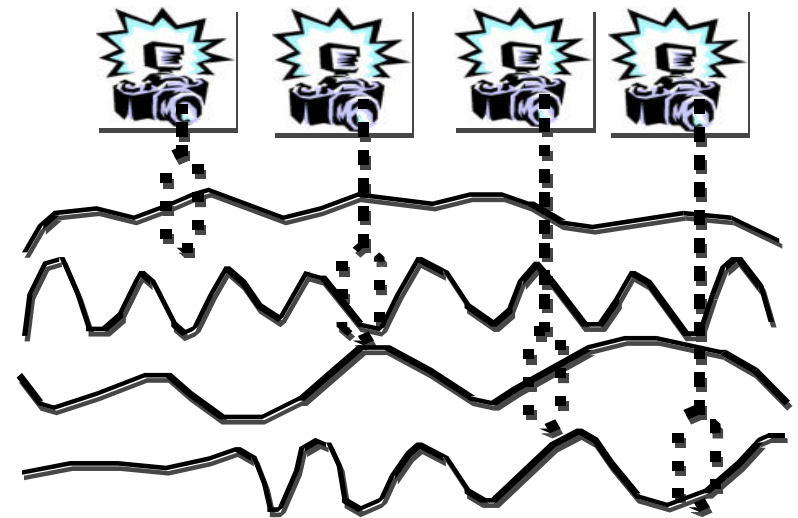
10-bit A/D Converter

- A/D Sampling Options
 - 1, 2, or 4 S/H amplifiers for conversions
 - Simultaneous or sequentially sampled
 - Scan mode with enable for each input
 - Two multiplexer settings can be programmed
 - Alternate between conversions
 - Selectable number of samples per interrupt
 - Selectable buffer fill mode
 - Auto or manual sample

Sampling Options



**SIMULTANEOUS
SAMPLING**



**SEQUENTIAL
SAMPLING**



10-bit A/D Converter

- ADCON3 Register Bits
 - ADCS - A/D Clock setting (T_{ad})
 - $T_{cy}/2$ increments, 167ns min
 - ADRC - A/D RC oscillator
 - SAMC - Auto Sample Time bits



10-bit A/D Converter

- ADPCFG Register
 - Sets port pin for analog (default) or digital mode
- ADCSSL - A/D Input Scan Select Register
 - Use when CSCNA bit set



Lab2 - Single channel ADC Conversion

- Instructions for Lab2:
 - Use workspace
“C:\MASTERS\845\Lab2\Lab2.mcw”
 - Write code to setup ADC:
 - Convert single Pot REF on AN2(RB2)
 - Use Manual Sample by setting SAMP bit every 100 mS (use Delay100 mS routine)
 - Manually check conversion complete
 - Use Internal RC for Tad

..... Contd.



Lab2 - Single channel ADC Conversion (contd.)

- Code to be modified clearly marked:
“Replace text with Code”
 - Compile the program using “Project -> Build All”
 - If no errors then Program the part but ...
 - MAKE SURE DIPs are in CLOSE position
 - Use “Debugger -> Program” to program part
 - MAKE SURE DIPs are in OPEN position
- Contd ...



Lab2 - Single channel ADC Conversion (contd.)

- In Windows[®]:
 - Open Hyperterm “Lab2.ht” in Lab2 directory.
 - If Bottom Status reads “Disconnected” then connect by selecting: “Call -> Call”
- In MPLAB[®] IDE, Run the program: “Debugger -> RUN”
- Turn POT REF to view different values in Hyperterm.



Lab 2 Result

```
Lab2 - HyperTerminal
File Edit View Call Transfer Help
Pot Ref=0x22C
Pot Ref=0x1F7
Pot Ref=0x1F0
Pot Ref=0x1F1
Pot Ref=0x1F2
Pot Ref=0x1EF
Pot Ref=0x1F1
Pot Ref=0x1F0
Pot Ref=0x1F1
Pot Ref=0x1EF
Pot Ref=0x1F2
Pot Ref=0x1F2
Pot Ref=0x1EF
Pot Ref=0x1F0
Pot Ref=0x1F0
Pot Ref=0x1F2
Pot Ref=0x1EF
Pot Ref=0x1F2
Pot Ref=0x1F1
Pot Ref=0x1EE
Pot Ref=0x1F0
Pot Ref=0x1F1
Pot Ref=0x1F0
Pot Ref=0x1EF
Connected 0:00:10  ANSIW  9600 8-N-1  SCROLL  CAPS  NUM
```

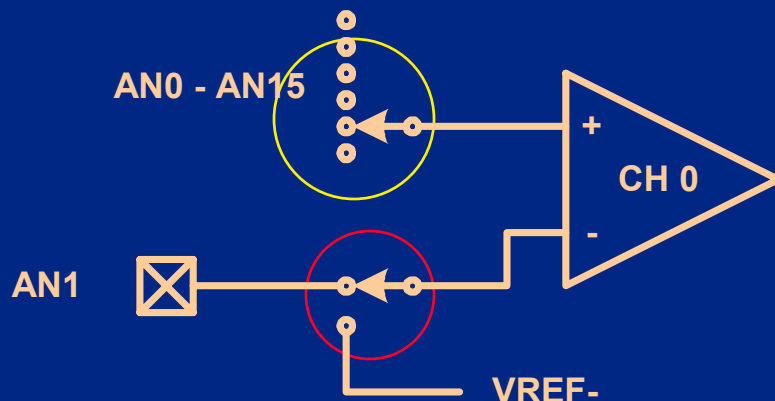


Lab 2 Results

- Configuration of 10-bit ADC
- Serial communications using dsPIC[®] DSC

10-bit A/D Converter

● ADCHS Register - CH0 S/H Input Selection

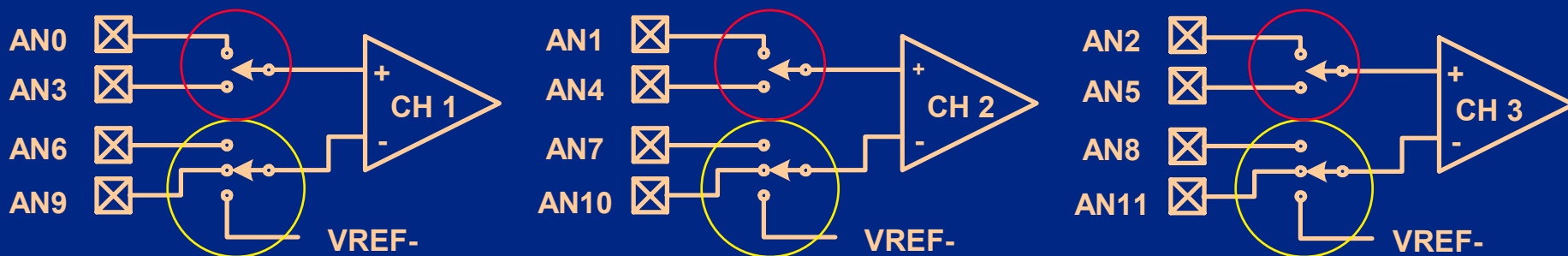


ADCHS Register

| | | | | | | | |
|------------|-------|-------|------------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| CHXNB<1:0> | CHXSB | CH0NB | CH0SB<3:0> | | | | |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| CHXNA<1:0> | CHXSA | CH0NA | CH0SA<3:0> | | | | |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |

10-bit A/D Converter

- ADCHS - CH1, CH2, CH3 Input Select



ADCHS Register

| | | | | | | | |
|-------------------------|--------------|--------|------------|--------|--------|--------|-------|
| R/W -0 | R/W -0 | R/W -0 | R/W -0 | R/W -0 | R/W -0 | R/W -0 | R/W-0 |
| CHXNB<1:0> | CHXSB | CH0NB | CH0SB<3:0> | | | | |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| CHXNA<1:0> | CHXSA | CH0NA | CH0SA<3:0> | | | | |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |



Motor Control PWM Module

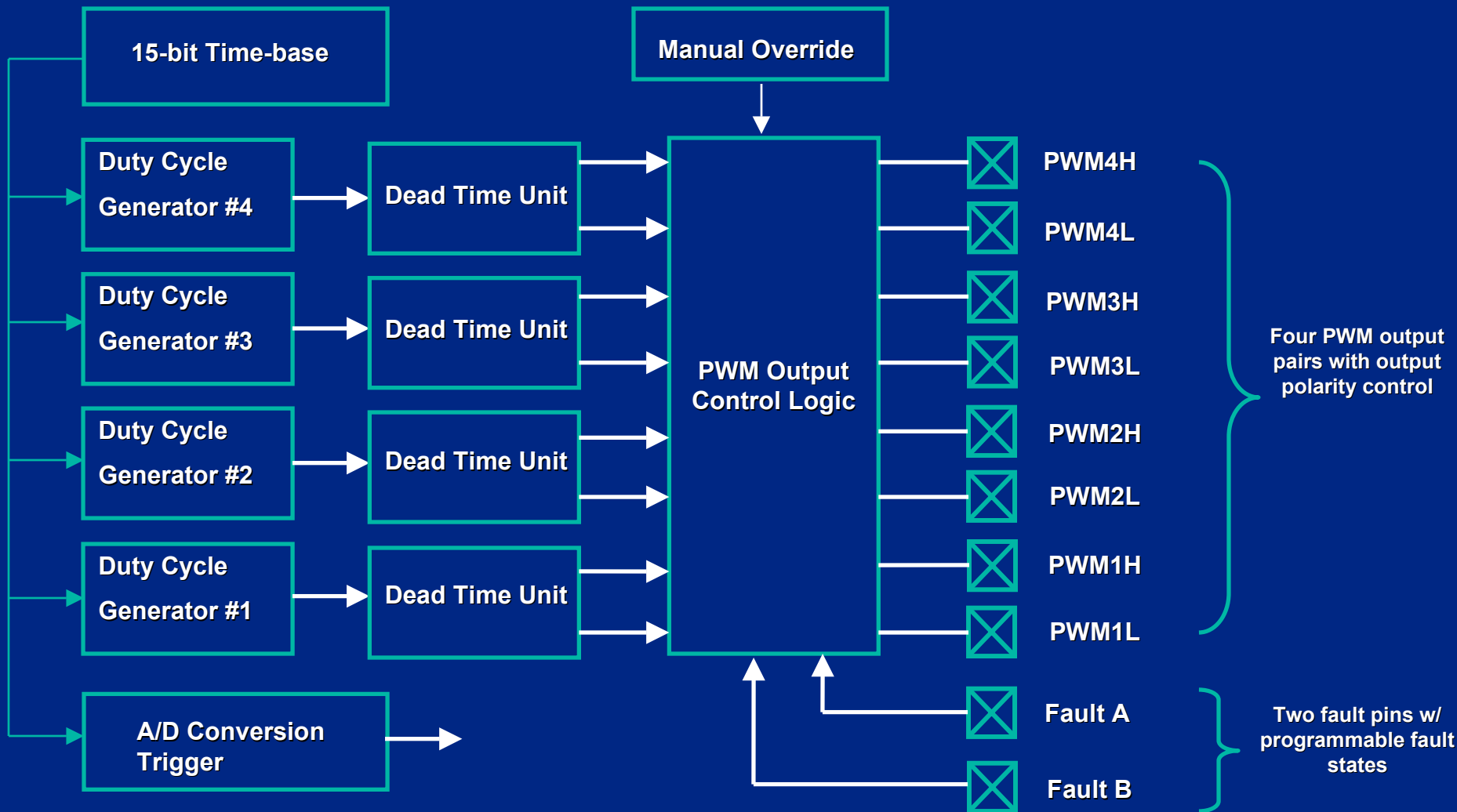


Motor Control PWM

- Features Overview
 - Designed for ACIM, BLDC, SR, UPS
 - Complementary PWM signals w/ dead time
 - Independent output mode
 - Special PWM generation modes
 - Fault pins for protection of power circuits
 - Override register for commutation applications
 - Synchronized A/D conversions



Motor Control PWM





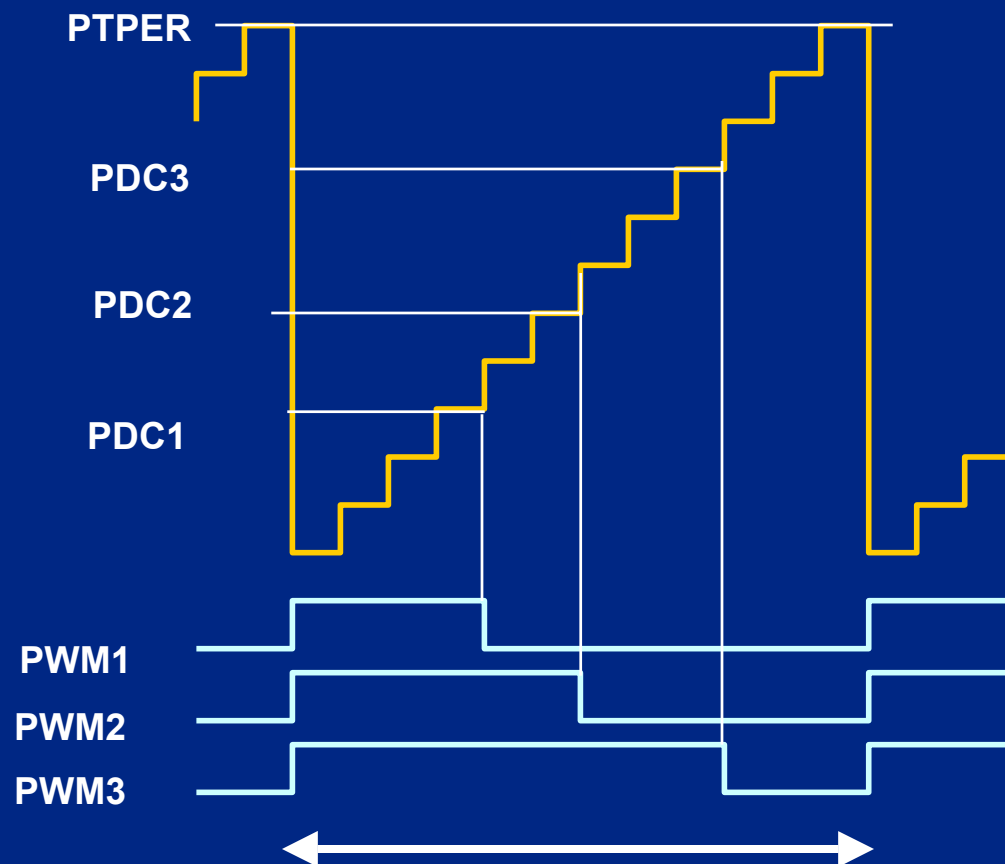
Motor Control PWM

- PTCON register
 - PTMOD - Timebase Mode
 - PTCKPS - Timebase clock pre-scale
 - PTOPS - Timebase interrupt post-scale
 - PTSIDL - Stop in IDLE mode
 - PTEN - Timebase Enable

| | | | | | | | |
|------------|-------|--------|-------------|-------|------------|-------|-------|
| R/W-0 | U-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| PTEN | - | PTSIDL | - | - | - | - | - |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PTOPS<3:0> | | | PTCKPS<1:0> | | PTMOD<1:0> | | |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |

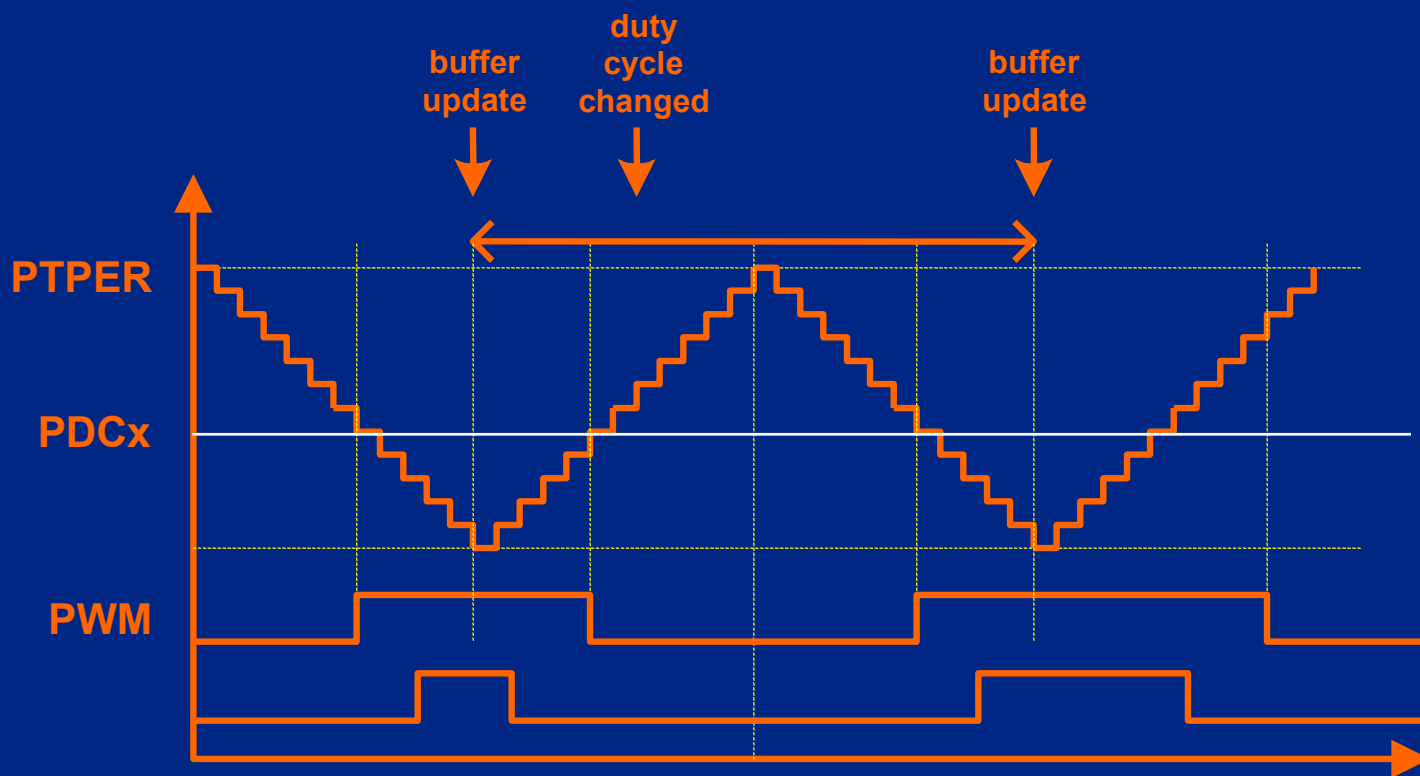
Motor Control PWM

- Edge Aligned PWM



Motor Control PWM

- Center Aligned PWM



We will use center aligned PWM for today's labs.

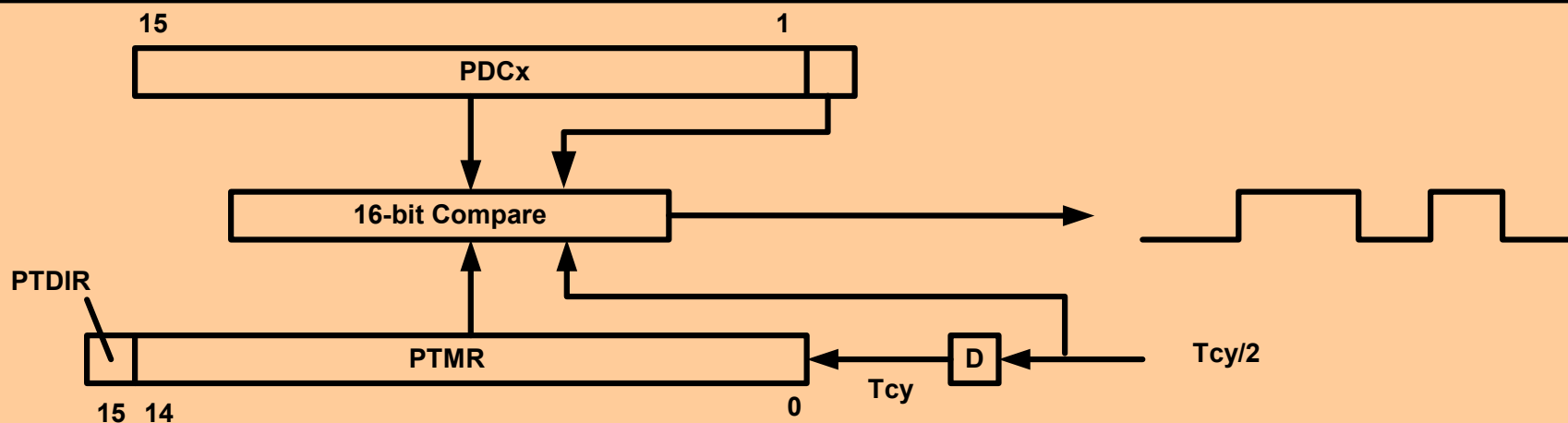


Motor Control PWM

- PWM Frequency and Counting Resolution
 - The 15-bit PTMR increments every T_{cy}
 - PTPER sets the counting period in T_{cy}
 - 16-bit duty cycles allow $T_{cy}/2$ edge resolution
 - Upper 15 bits of d.c. compared to PTMR
 - LSbit of duty cycle specifies $T_{cy}/2$ boundary
 - Divide count period value by 2 to get PTPER
 - PWM frequency is halved for center-aligned

Motor Control PWM

- Duty Cycle Comparison Block Diagram





Motor Control PWM

- PWM Formula for Center Aligned PWM

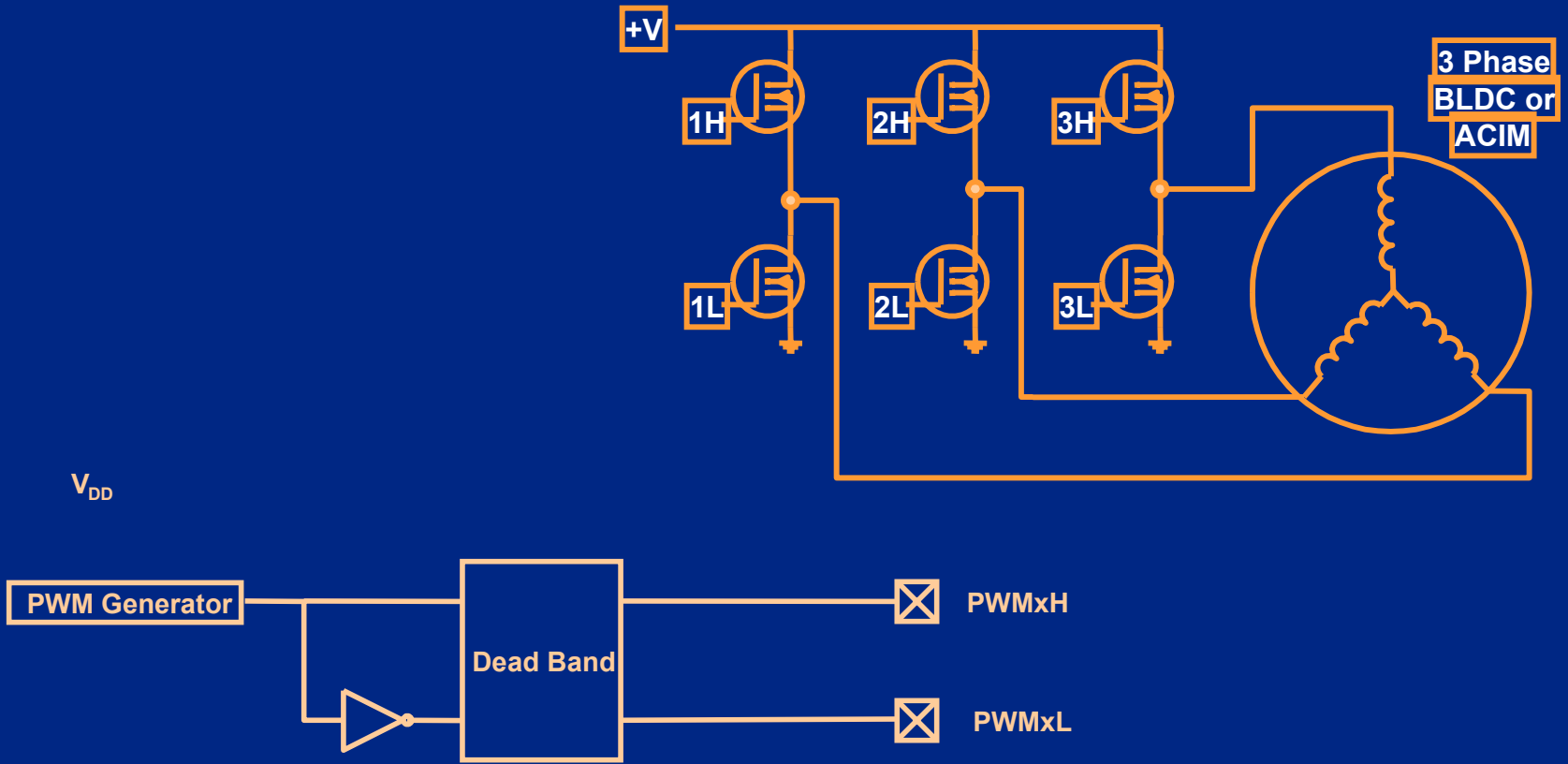
$$PTPER = (F_{cy}/F_{pwm} - 1)/2$$

- PWM Formula for Edge Aligned PWM

$$PTPER = (F_{cy}/F_{pwm} - 1)$$

Motor Control PWM

- Complementary output mode





Motor Control PWM

- PWMCON1 register
 - PWM output enable bits
 - PWM output mode select

- For labs, enable PWM1, PWM2, and PWM3 pairs for complementary mode

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| - | - | - | - | PMOD4 | PMOD3 | PMOD2 | PMOD1 |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PEN4H | PEN3H | PEN2H | PEN1H | PEN4L | PEN3L | PEN2L | PEN1L |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |



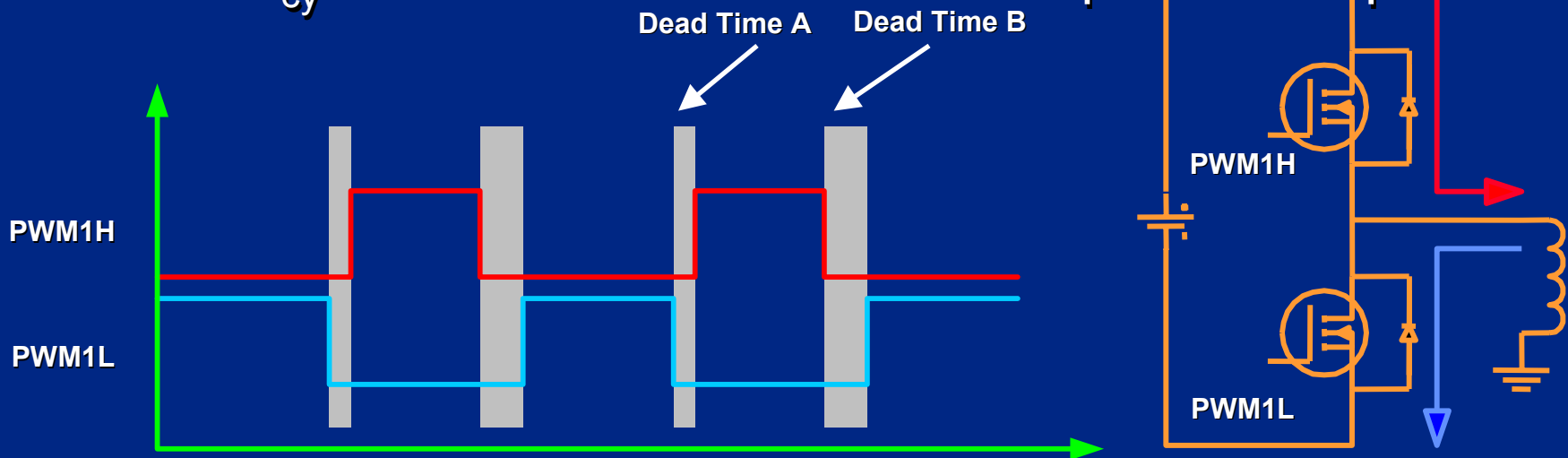
Motor Control PWM

- Dead Time Generators
 - Two programmable dead times
 - one for dsPIC2010
 - Control bits choose dead time for active/inactive events
 - Two ways to use DTA and DTB
 - Tcy minimum resolution
 - Prescalers for increased dead time range



MCPWM Dead Time Insertion

- Applies only to pin pairs in complimentary mode
- Two programmable dead times
- One dead time per pair for multiple inverters OR
- Two dead times per pair for distortion optimization
- T_{cy} minimum resolution with four pre-scale options



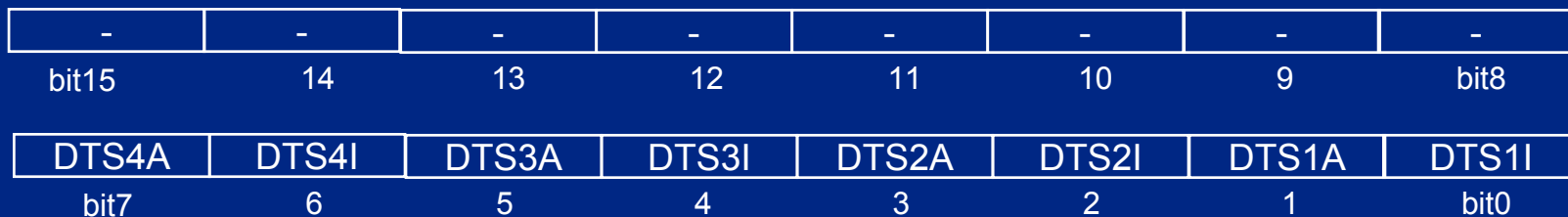


Motor Control PWM

- DTCON1 selects dead time value for A and B



- DTCON2 selects which dead time value is assigned to the active(A) or inactive(I) edge of each generator





MCPWM Fault Inputs

- Two programmable fault pins: Fault A, Fault B
 - dsPIC2010 only has Fault A
- Fault pins can be assigned to each output pair
- Fault state for each output is programmable
- Automatic or latched fault protection
- Interrupt vector for each fault input
- Fault A has priority over Fault B
- Fault condition overrides all other pin control
- Fault pins can be used as interrupt pins when not used for PWM module



Motor Control PWM

- FLTACON Register
 - FAENx - Enables pin pair for control by fault pin
 - FLTAM - Latched or cycle-by-cycle mode
 - FAOVxx - Override value bits

- FLTBCON register is identical
- **For labs - FLTA drives all outputs inactive, latched mode**

| | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|
| FAOV4H | FAOV4L | FAOV3H | FAOV3L | FAOV2H | FAOV2L | FAOV1H | FAOV1L |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |

| | | | | | | | |
|-------|---|---|---|-------|-------|-------|-------|
| FLTAM | - | - | - | FAEN4 | FAEN3 | FAEN2 | FAEN1 |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |



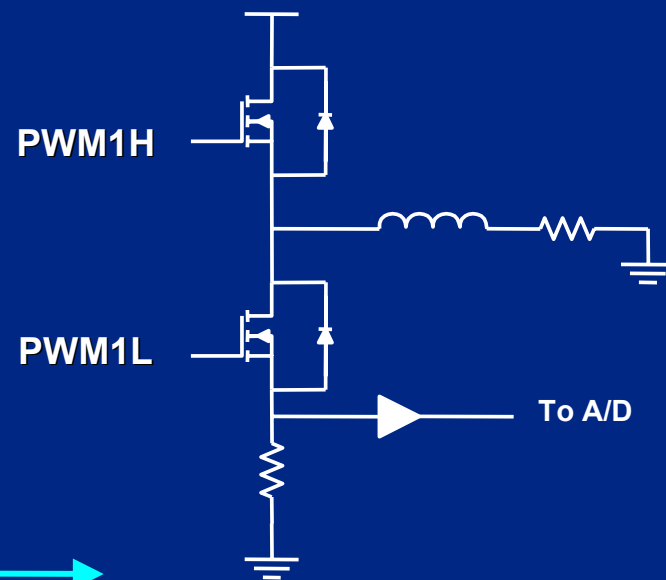
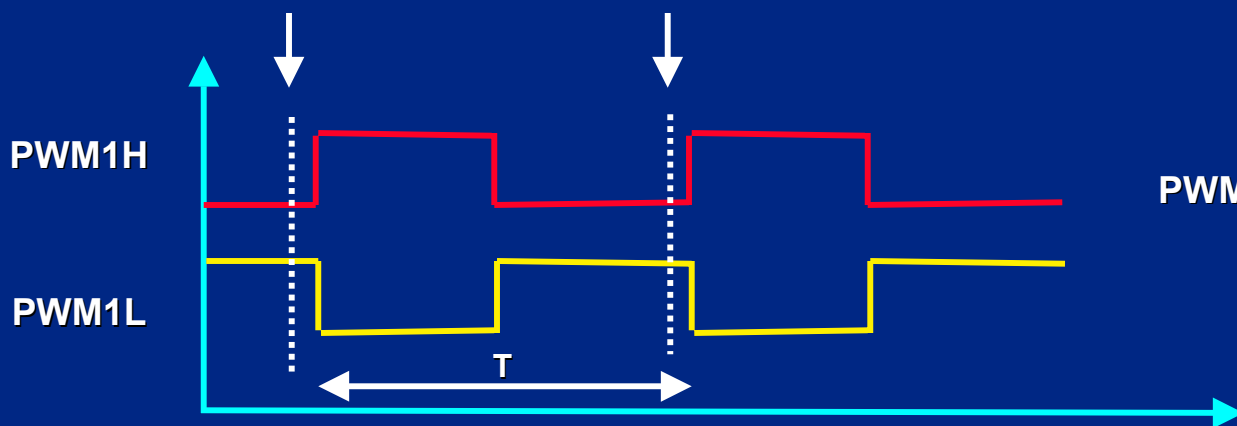
MCPWM A/D Synchronization

SEVTCMP register sets A/D conversion start time in PWM cycle

Ensure A/D properly captures shunt current

Can also use to minimize control loop update delay

Trigger conversion at end of bottom transistor on-time

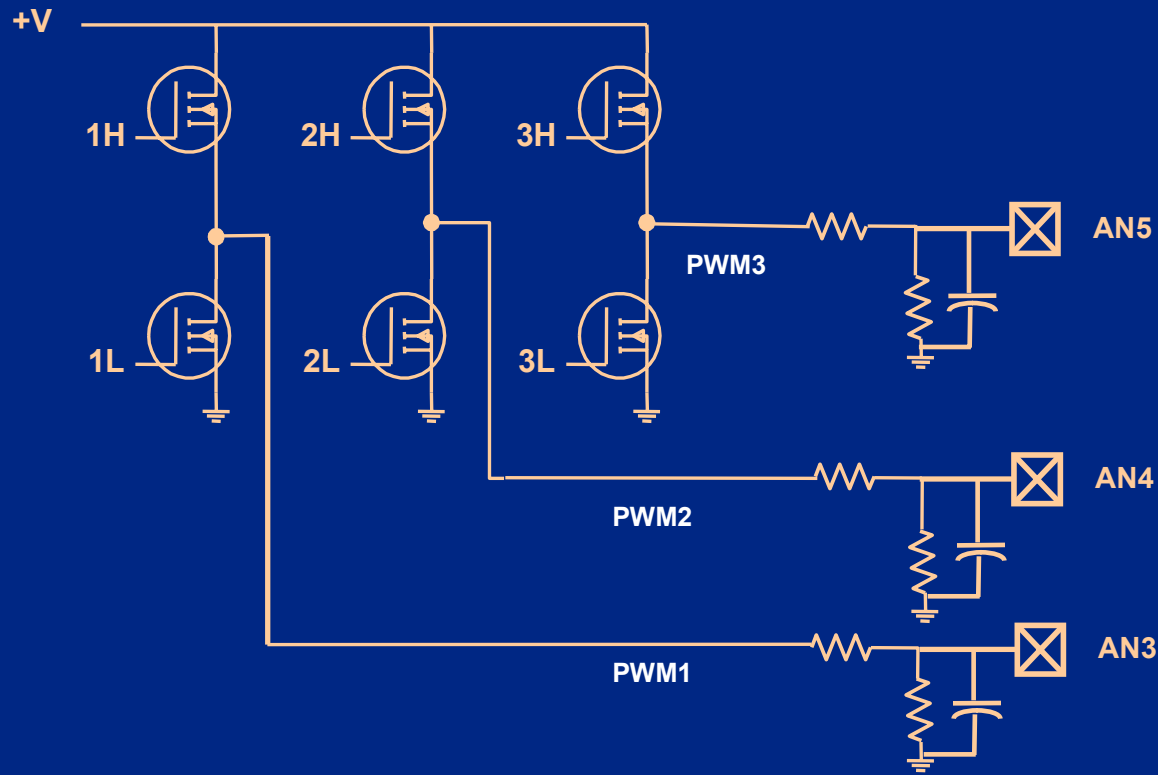




Lab 3 Objectives

- 3 MC PWM pairs are connected to 3 ADC inputs
- MC PWMs will be driven with a 25%, 50% and 75% duty cycle
- The average of the duty cycle will be measured by each ADC
- If the configuration is correct then the 3 ADC value displayed will be 1.8V, 2.8V and 3.75V using Hyperterm

Lab 3 Hardware





Lab3 Jumper settings

- **J2**(2 pin) - shorted.
- **J15**(3 pin) - shorted between pin 2 & 3
(short link towards the crystal oscillator)
- **J7, J11** and **J13** - shorted between pin 2 & 3
(short link away from ICD 2 connector)
- **JP8, JP12, JP14, JP16** and **JP17** keep open
- Keep Potentiometer **REF(R14)** and **R60** in center position



Lab 3 Objectives

- Learn to setup the MC PWM period, duty cycle, dead time, ADC trigger etc.
- Learn to setup the ADC for simultaneous sampling and conversion on multiple channels



Lab3 Setup

- Lab 3:
 - Part 1: Setup ADC
 - ADC to simultaneously sample 4 channels
 - Set CH0 - AN2(Pot), CH1 - AN3, CH2 - AN4 and CH3 - AN5. $V_{ref-} = GND$
 - Trigger Conversion with 16 PWM Periods
 - Set $T_{ad} = 4T_{cy}$
 - Modify Registers ADCON1, ADCON2, ADCON3, ADCHS and ADPCFG
 - Code to be modified is clearly marked



Lab 3 Setup (contd.)

- Lab3 :
 - Part 2: setup MC PWM
 - Setup for continuous center aligned PWM
 - Complementary pairs
 - $F_{cy} = 20\text{Mips}$, PWM Freq = 80khz
 - PDC1 = 25%, PDC2 = 50% PDC3 = 75%
 - Dead time = 1 μS (active and inactive)
 - Code to be modified is clearly marked



Lab 3 Setup (contd.)

- Lab 3 General:
 - Use workspace in:
“C:\MASTERS\845\Lab3\Lab3.mcw”
 - Do not connect the motor to the board
 - Use Hyperterm setup: “Lab2.ht” in Lab3 directory to view results of the ADC
 - Make sure that DIPs are in CLOSE position when programming
 - Make sure that DIPs are in OPEN position when running

Lab3 Result

```

Lab3 - HyperTerminal
File Edit View Call Transfer Help
Pot = 0x26A PWM1 = 0x16A PWM2 = 0x22D PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x16C PWM2 = 0x22E PWM3 = 0x2EE
Pot = 0x26C PWM1 = 0x167 PWM2 = 0x22A PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x16A PWM2 = 0x22D PWM3 = 0x2ED
Pot = 0x26B PWM1 = 0x16D PWM2 = 0x22E PWM3 = 0x2EF
Pot = 0x26A PWM1 = 0x16E PWM2 = 0x22F PWM3 = 0x2EF
Pot = 0x267 PWM1 = 0x16C PWM2 = 0x22E PWM3 = 0x2EF
Pot = 0x26C PWM1 = 0x169 PWM2 = 0x22B PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x168 PWM2 = 0x22B PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x16C PWM2 = 0x22D PWM3 = 0x2EE
Pot = 0x26C PWM1 = 0x168 PWM2 = 0x22B PWM3 = 0x2EE
Pot = 0x26A PWM1 = 0x16D PWM2 = 0x22F PWM3 = 0x2EF
Pot = 0x26B PWM1 = 0x16D PWM2 = 0x22F PWM3 = 0x2EF
Pot = 0x26A PWM1 = 0x169 PWM2 = 0x22C PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x16C PWM2 = 0x22D PWM3 = 0x2EE
Pot = 0x26B PWM1 = 0x169 PWM2 = 0x22B PWM3 = 0x2EE
Pot = 0x26A PWM1 = 0x168 PWM2 = 0x22B PWM3 = 0x2ED
Pot = 0x26B PWM1 = 0x167 PWM2 = 0x22A PWM3 = 0x2EC
Pot = 0x26A PWM1 = 0x16A PWM2 = 0x22C PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x169 PWM2 = 0x22C PWM3 = 0x2ED
Pot = 0x26C PWM1 = 0x169 PWM2 = 0x22B PWM3 = 0x2ED
Pot = 0x26B PWM1 = 0x16D PWM2 = 0x22E PWM3 = 0x2EF
Pot = 0x26A PWM1 = 0x16A PWM2 = 0x22C PWM3 = 0x2ED
Pot = 0x26A PWM1 = 0x168 PWM2 = 0x22B PWM3 = 0x2ED

```

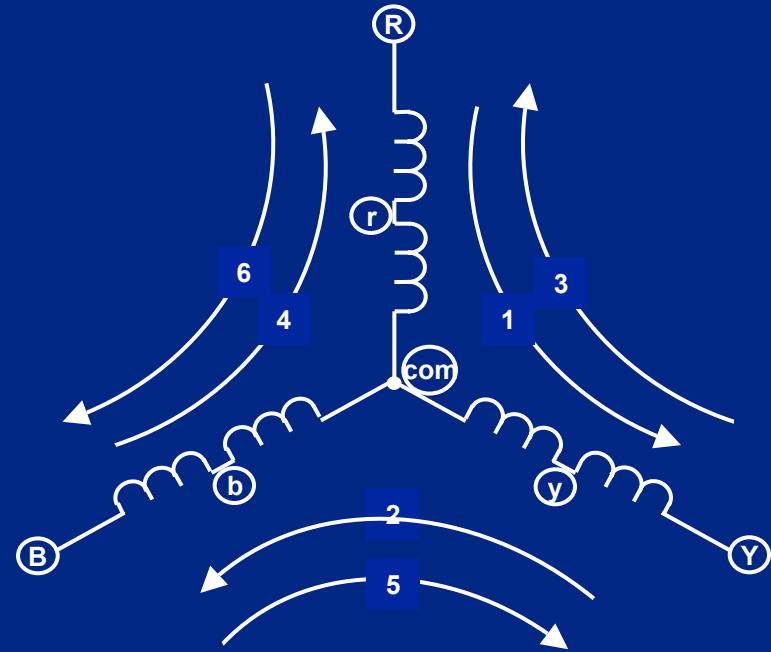
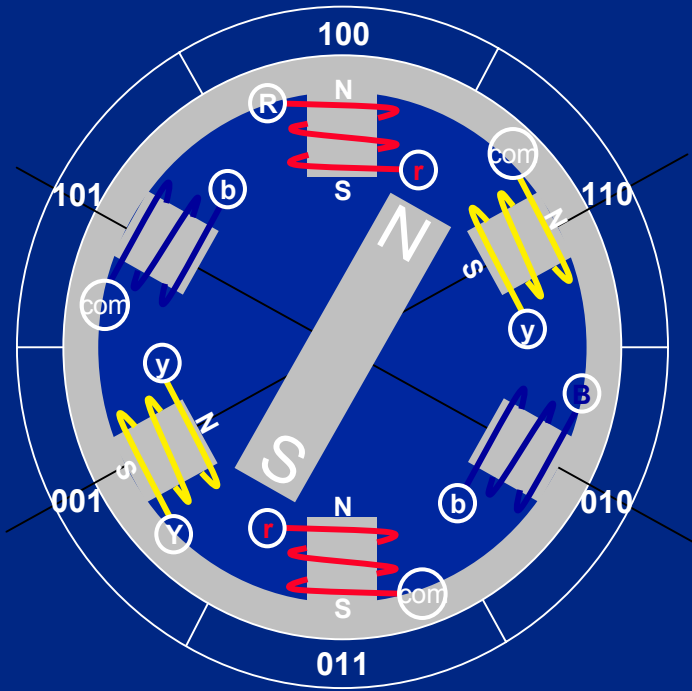
Connected 0:14:43 ANSIW 9600 8-N-1 SCROLL CAPS NUM Capture Print echo



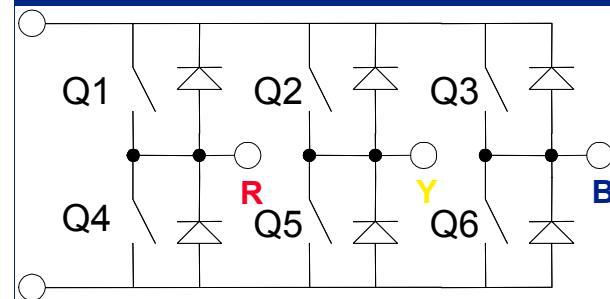
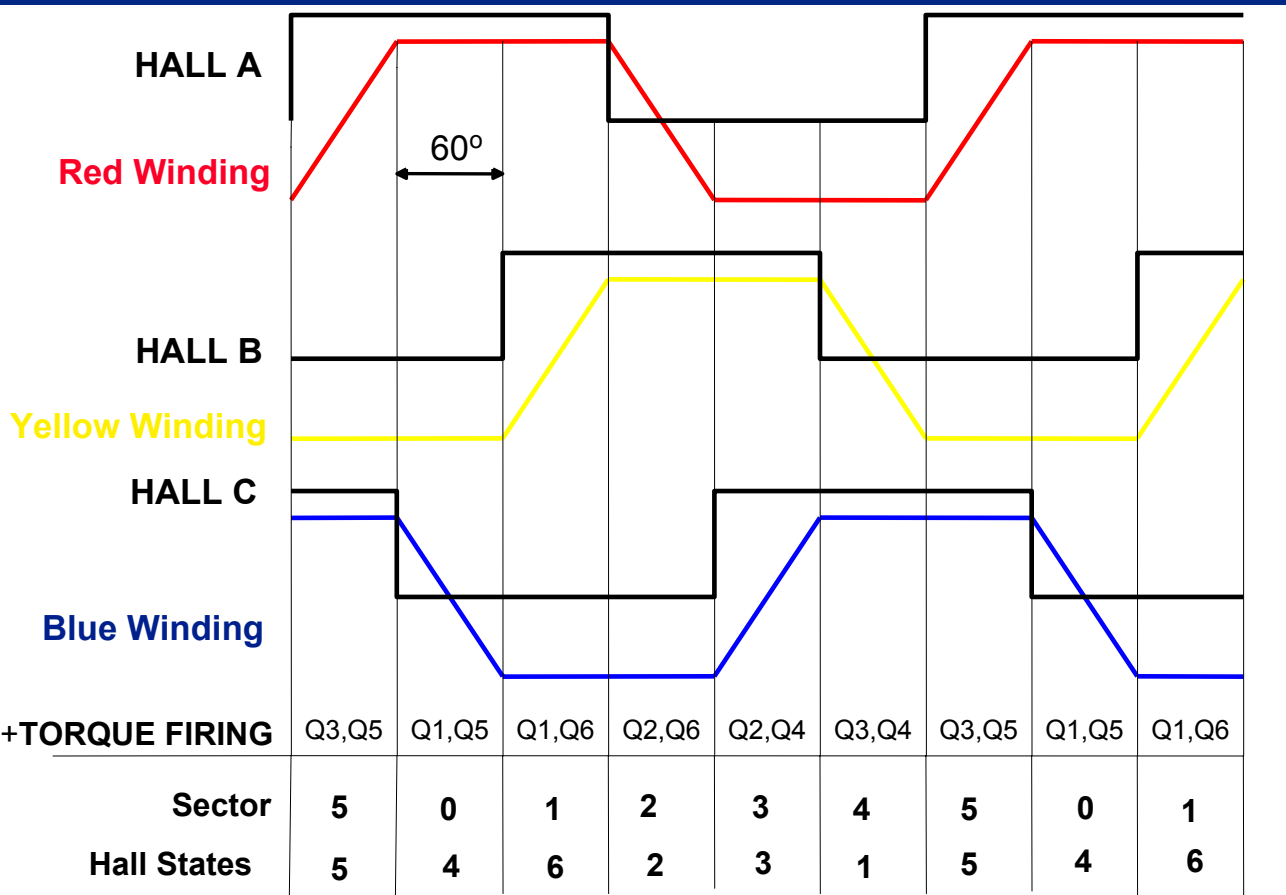
Lab 3 Results

- Learned how to setup the ADC
- Learned how to setup the PWM
- We should now have sufficient information to setup a simple sensored control of a BLDC motor

Brushless DC Motor Construction



Standard BLDC Control

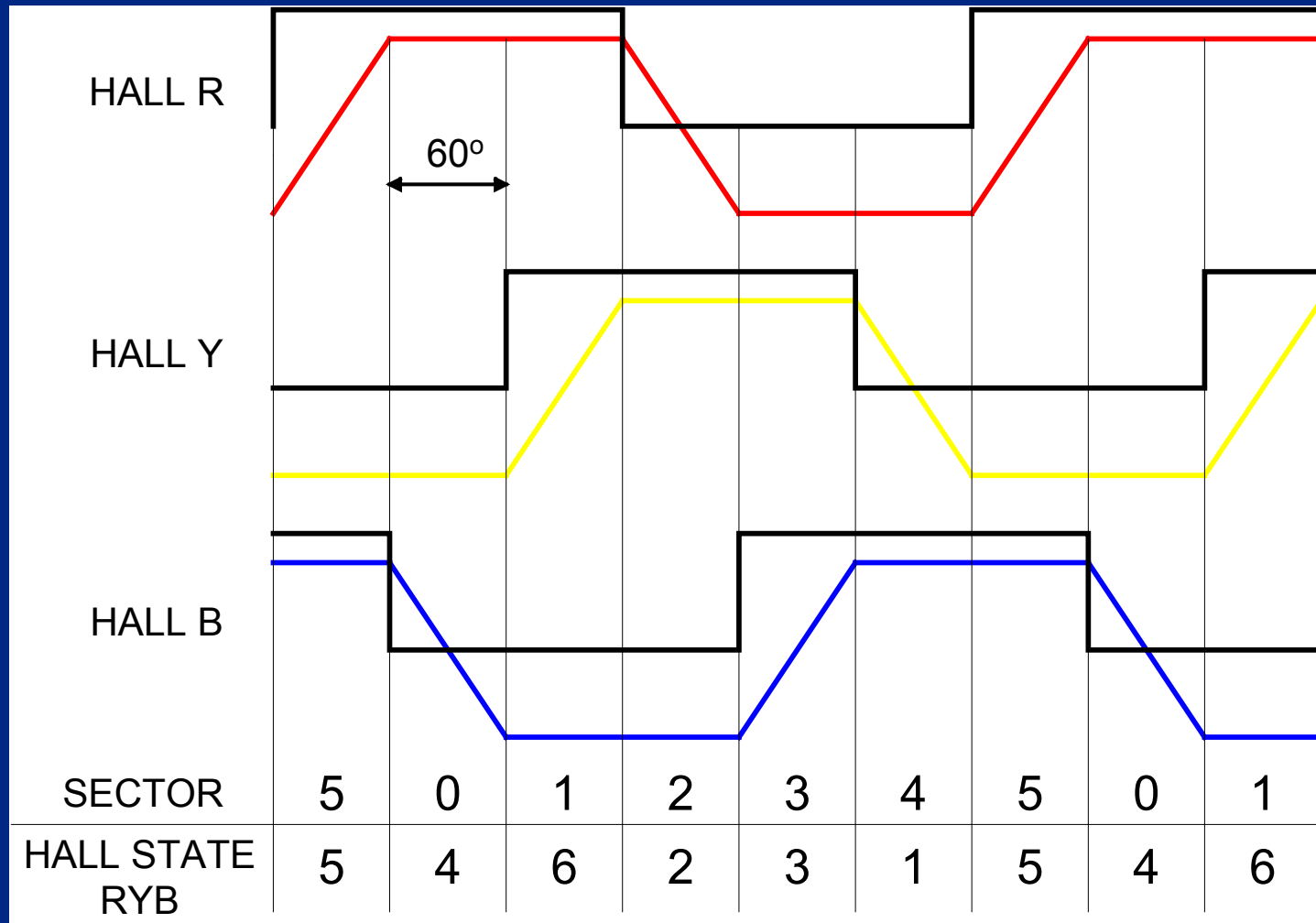




Standard Sensored BLDC Control

- The 3 logic signals are decoded to determine which windings should be energized.
- There are 6 valid states and 2 states that should never be seen (000, 111).
- Use Lookup table to drive the 3 windings, high or low or no-drive.
- The 6 different valid states directly translate to the 6 different 60° electrical cycle sectors.
- The states should only transition by one at a time. Missing transitions or invalid states should be detected for robust operation.

Standard BLDC Position Sensing



BLDC Motor Construction



Rotor magnets

Stator winding

Hall sensors



Control of Sensored BLDC

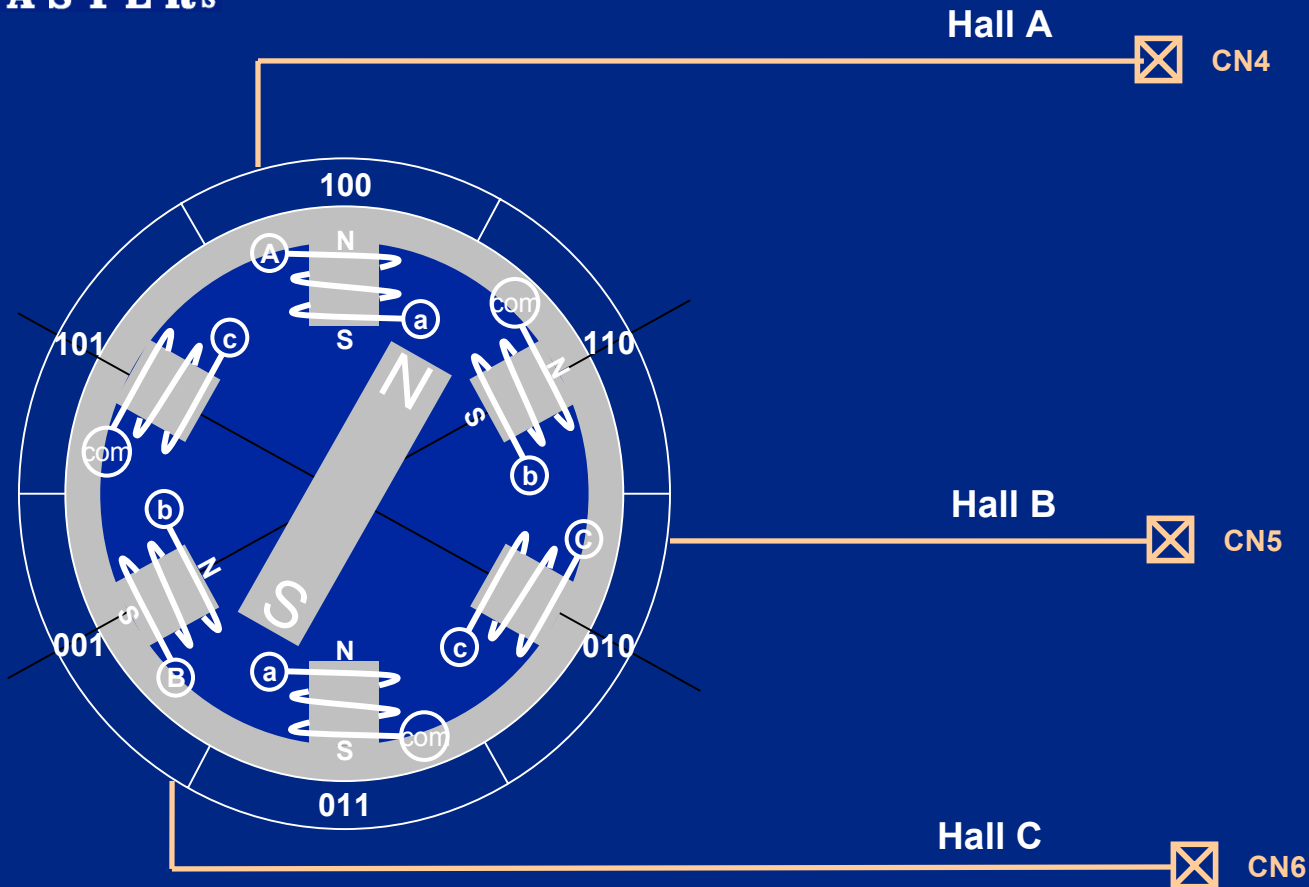
- Simple Control Technique:
 - Sense change in state of position
 - Read Hall sensors to determine the new state
 - Use lookup table for commutation
 - Use PWM for speed control



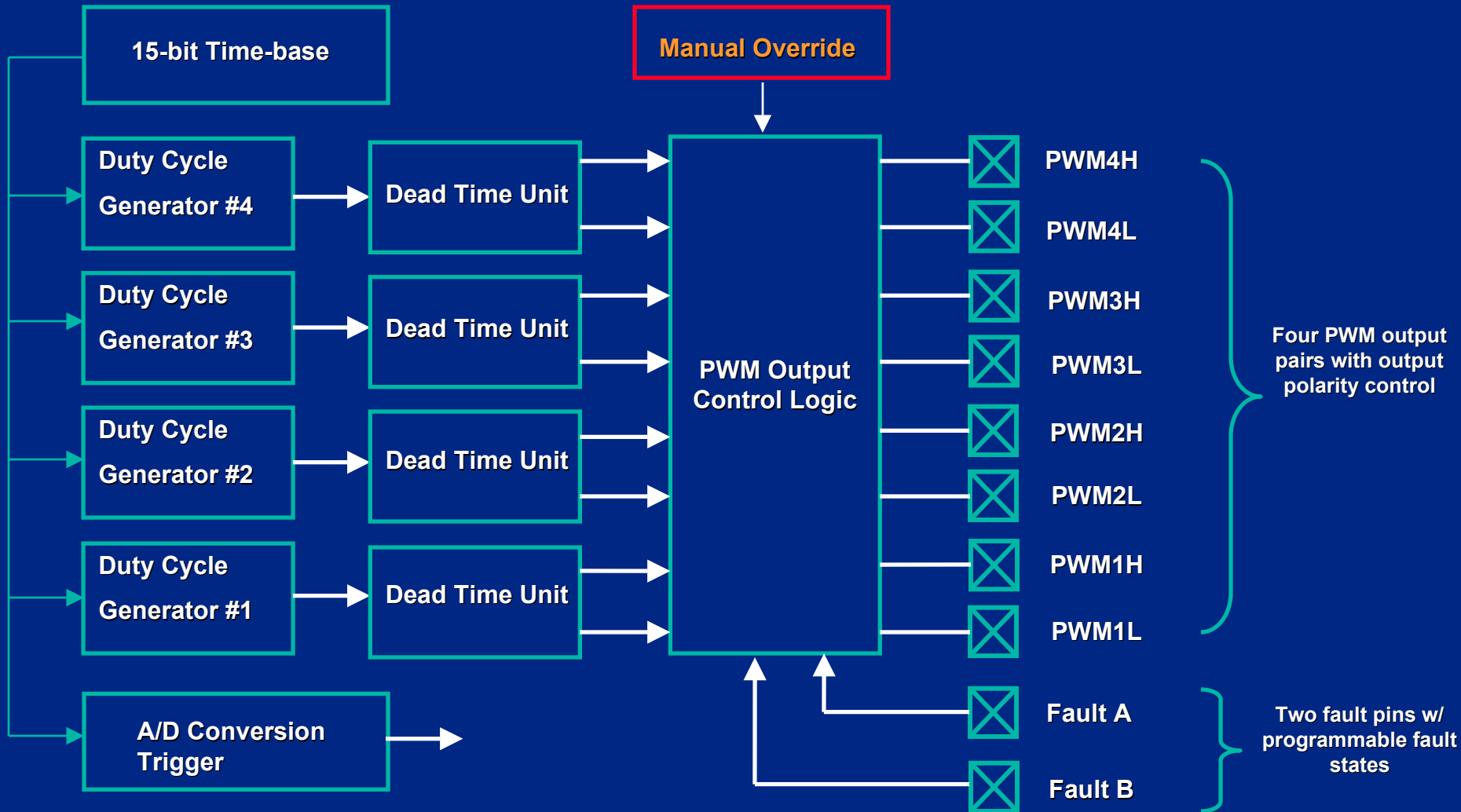
Change Notification (CN)

- dsPIC[®] DSC has Change Notification inputs:
 - Detect digital changes on a specific input pin and generates an interrupt
 - Hall sensors A, B and C are connected to RB3, 4 and 5 or CN4, 5 and 6 respectively.
 - When CNxInterrupt occurs, all 3 Hall inputs are read and a lookup table is used to control the BLDC motor

CN Hardware



Motor Control PWM - Manual Override Control





Motor Control PWM

- OVDCON (override control) register
 - Used for motor commutation
 - I/O pin can be PWM, active, or inactive
 - POVD = 0, I/O pin is controlled manually
 - POUT bits set pin state for manual control
 - Dead time requirements are always satisfied in complementary mode

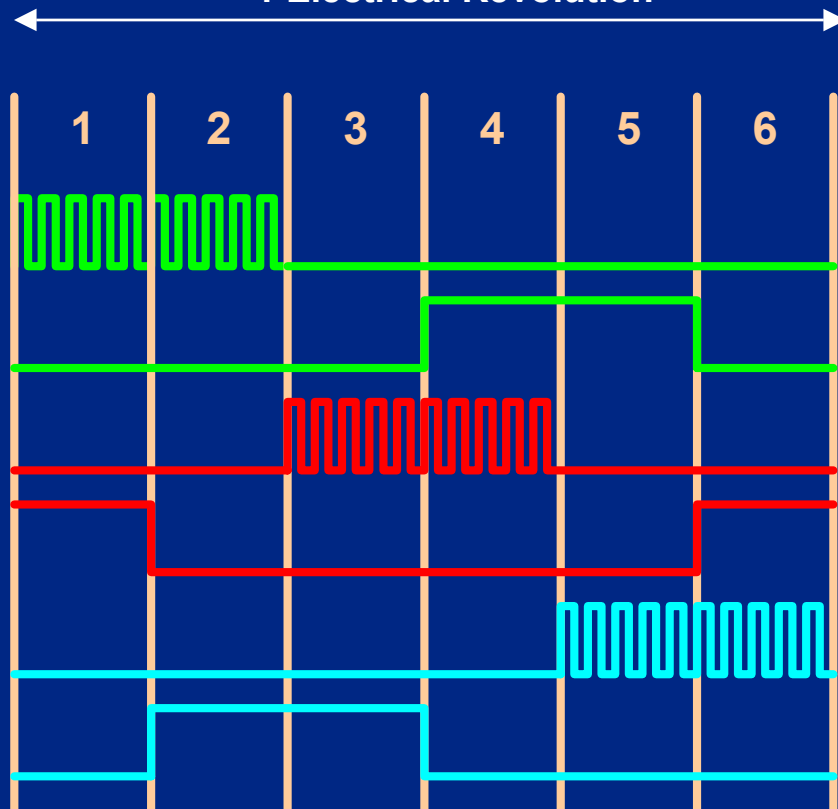
| | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| POVD4H | POVD4L | POVD3H | POVD3L | POVD2H | POVD2L | POVD1H | POVD1L |
| bit15 | 14 | 13 | 12 | 11 | 10 | 9 | bit8 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| POUT4H | POUT4L | POUT3H | POUT3L | POUT2H | POUT2L | POUT1H | POUT1L |
| bit7 | 6 | 5 | 4 | 3 | 2 | 1 | bit0 |



Motor Control PWM

- Using OVDCON for BLDC commutation

1 Electrical Revolution



OVDCON Value

Step

POVD<7:0>

POUT<7:0>

1

00000010

00000100

2

00000010

00010000

3

00001000

00010000

4

00001000

00000001

5

00100000

00000001

6

00100000

00000100



Lab4 Jumper settings

- **J2**(2 pin) - shorted.
- **J15**(3 pin) - shorted between pin 2 & 3
(short link towards the crystal oscillator)
- **J7, J11 and J13** - shorted between pin 2 & 1
(short link towards ICD 2 connector)
- **JP8, JP12, JP14, JP16 and JP17** keep open
- Keep Potentiometer **REF(R14) and R60** in center position



Lab 4: Sensored Control

- Lab 4:
 - Part A: Setup CNxInterrupts
 - Set up CNx interrupt vectors to sense changes in Hall sensors
 - Read PORTB and check States (3-bits)
 - Use Lookup table to set up the Override Control Register
 - Lookup Table already Provided as “StateLoTable[0,1,2,3,4,5,6,7]”
 - Note: States 0 and 7 are illegal



Lab 4: Sensored Control (contd.)

- Lab 4:
 - Part B: Setup PWM
 - Set period to 10-bit, 4.9 kHz @ 5 Mips
 - Set Duty Cycle = ADC input on Pot REF
 - Set for Edge Aligned PWM
 - Dead Time not required
 - Initialize the OVDCON register for override control



Lab 4: Sensored Control (contd.)

- Lab 4:
 - Part C: Setup ADC:
 - Convert single channel on AN2(POT REF)
 - Use $T_{ad} = ADRC$
 - Use Manual Sample by setting SAMP bit every 100 mS (use Delay100mS routine)
 - Manually check conversion complete
 - Load ADC result as Duty Cycle for all PWMs



Lab 4: Sensored BLDC (contd.)

- Lab 4:
 - General:
 - Use workspace in “C:\MASTERS\845\Lab4\Lab4.mcw”
 - Position REF to center position
 - Connect Motor’s Black and white connector
 - Build project and program part ...
 - DIPs in CLOSED position for programming
 - DIPs in OPEN position for running



Lab 4 Results

- We learned how to control a sensored BLDC motor
- We controlled speed of the motor by using a potentiometer
- Now let us look at Sensorless Control of a BLDC motor



Sensorless BLDC Motor



Why sensorless?

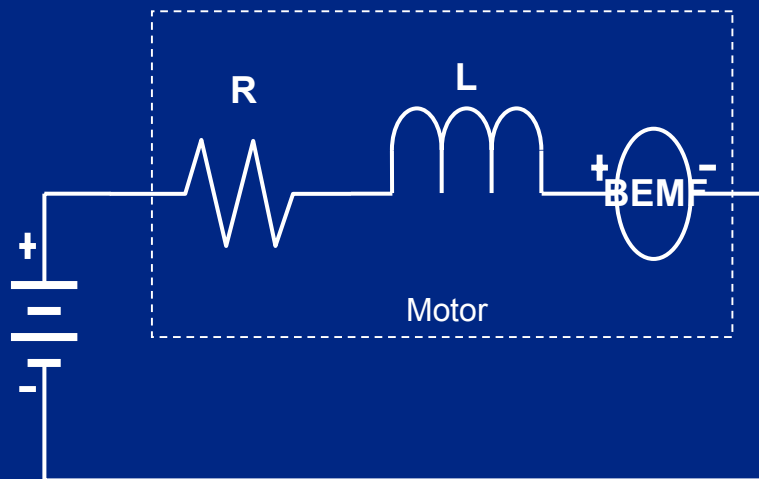
- Reliability – especially aerospace, military
- Physical space restrictions – axial length
- Issues surrounding sealing of connections
- Applications where rotor runs “flooded”
- Manufacturability – alignment and duty cycle tolerance
- Cost – especially on low power systems
 - Even at high volumes, position sensing can add \$3 to system cost



BLDC Sensorless Techniques

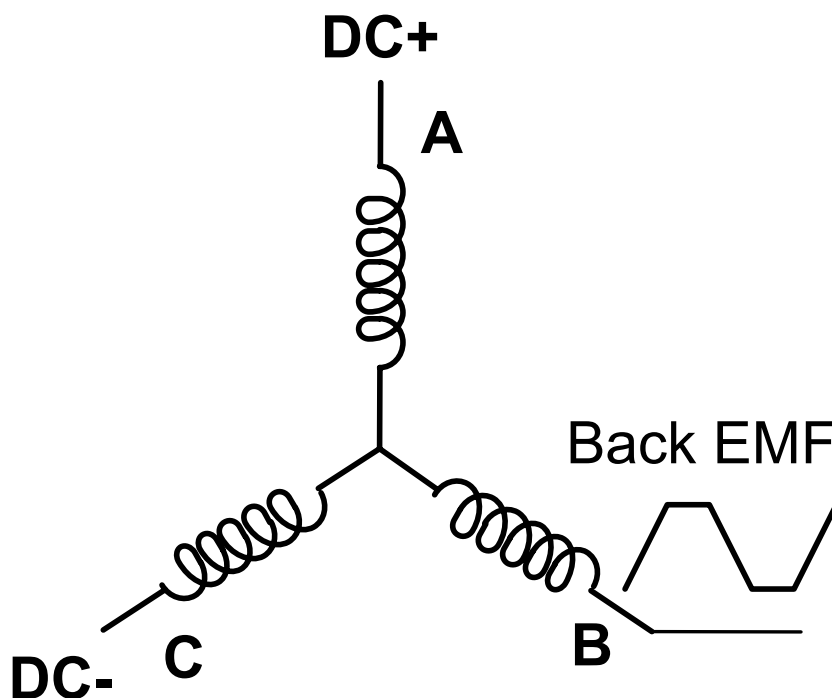
- Many method to sense rotor position
- AN901 uses Back EMF sensing
 - Reliable
 - Varies linearly with speed
 - Works over a wide range of BLDC Motors
 - Relatively easy to implement
 - Works well for applications like Fan or pump speed control
- Method used is called Back EMF “zero crossing” method
 - Consists of monitoring the voltage of the inactive winding

What is BEMF?



- When a DC motor spins, the PM rotor, moving past the stator coils induces a electrical potential in the coils called Back EMF
- BEMF is directly proportional to speed
- $BEMF = RPM/K_v$
- In order to sense BEMF we have to spin the motor

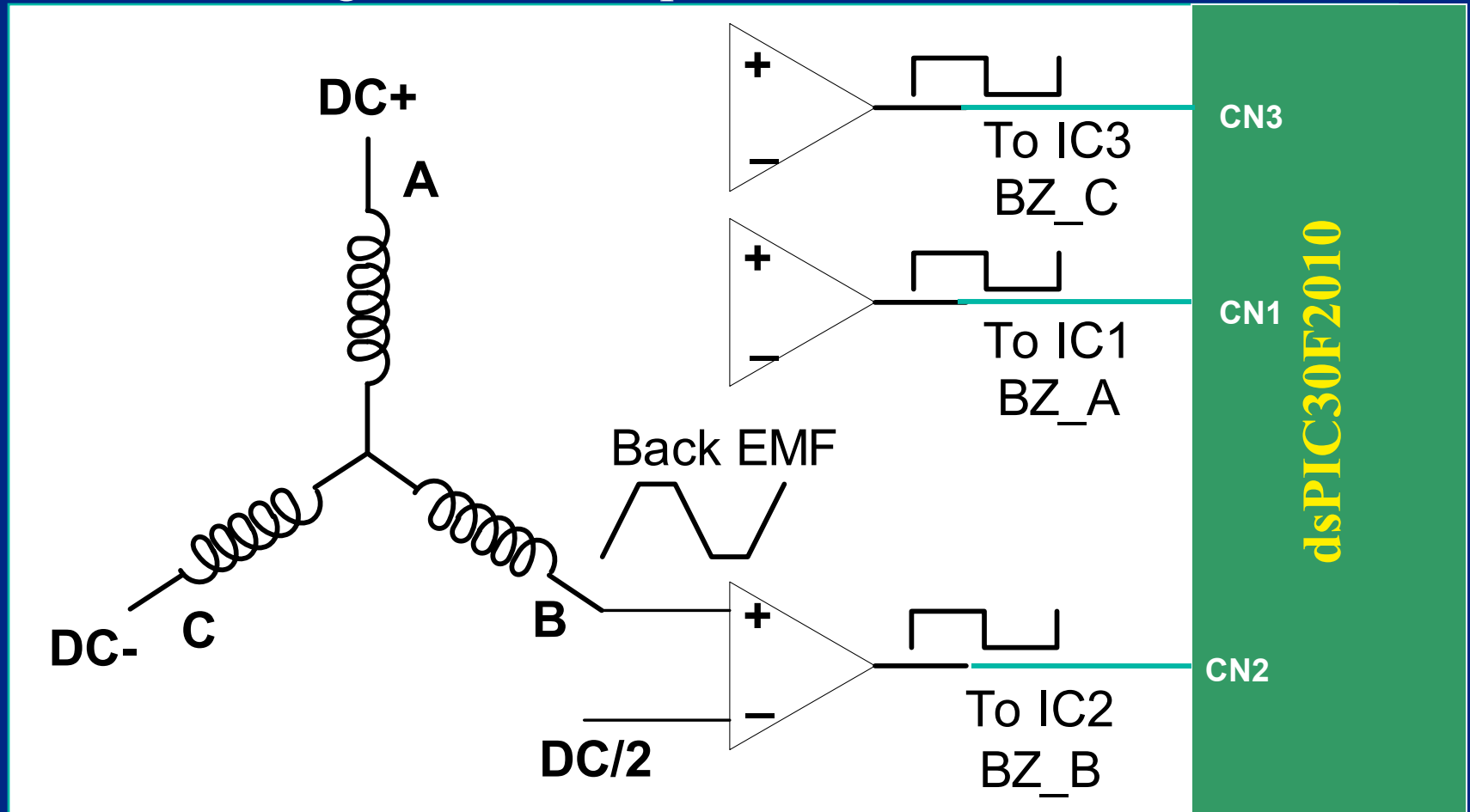
BLDC Motor Back EMF



- Phase A and C are energized
- Inactive Phase B has induced Back EMF
- Normally the phase which is not energized, is monitored for Back EMF
- **Important: Motor has to be spinning**

Technique to Monitor Back EMF

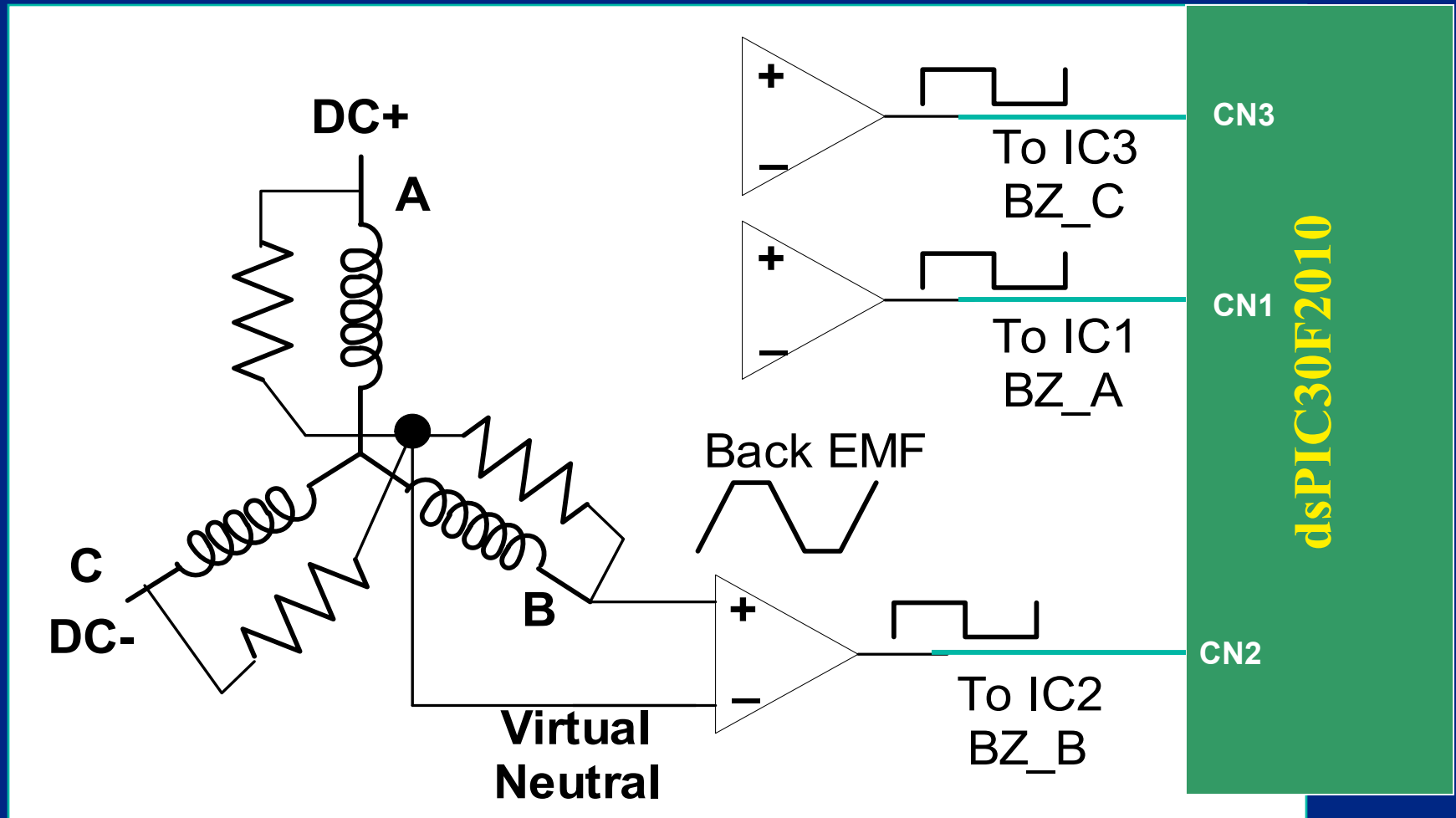
- Back EMF signal compared with DC bus/2 using external comparators



dsPIC30F2010

Technique to Monitor Back EMF

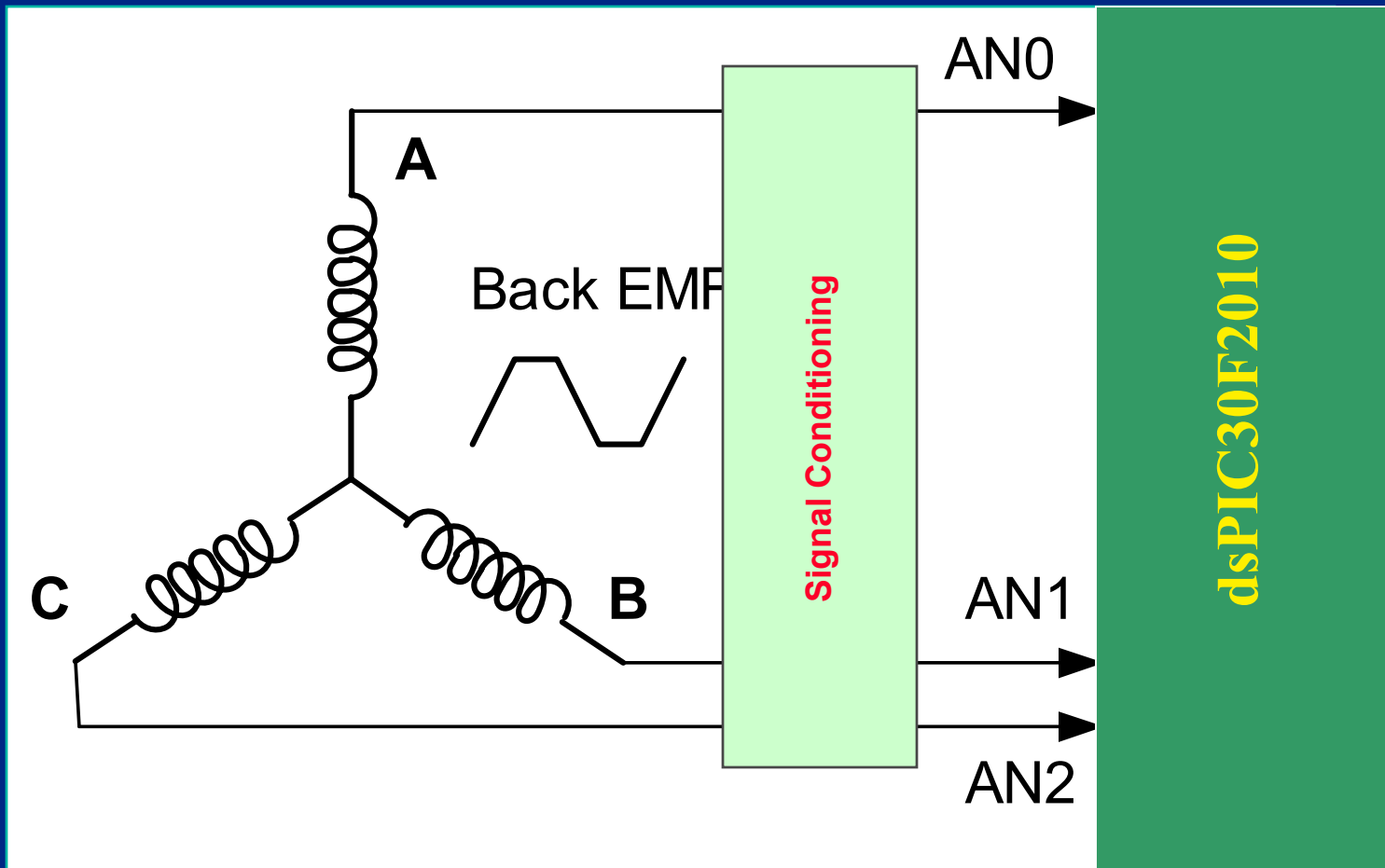
- Back EMF signal compared with virtual neutral



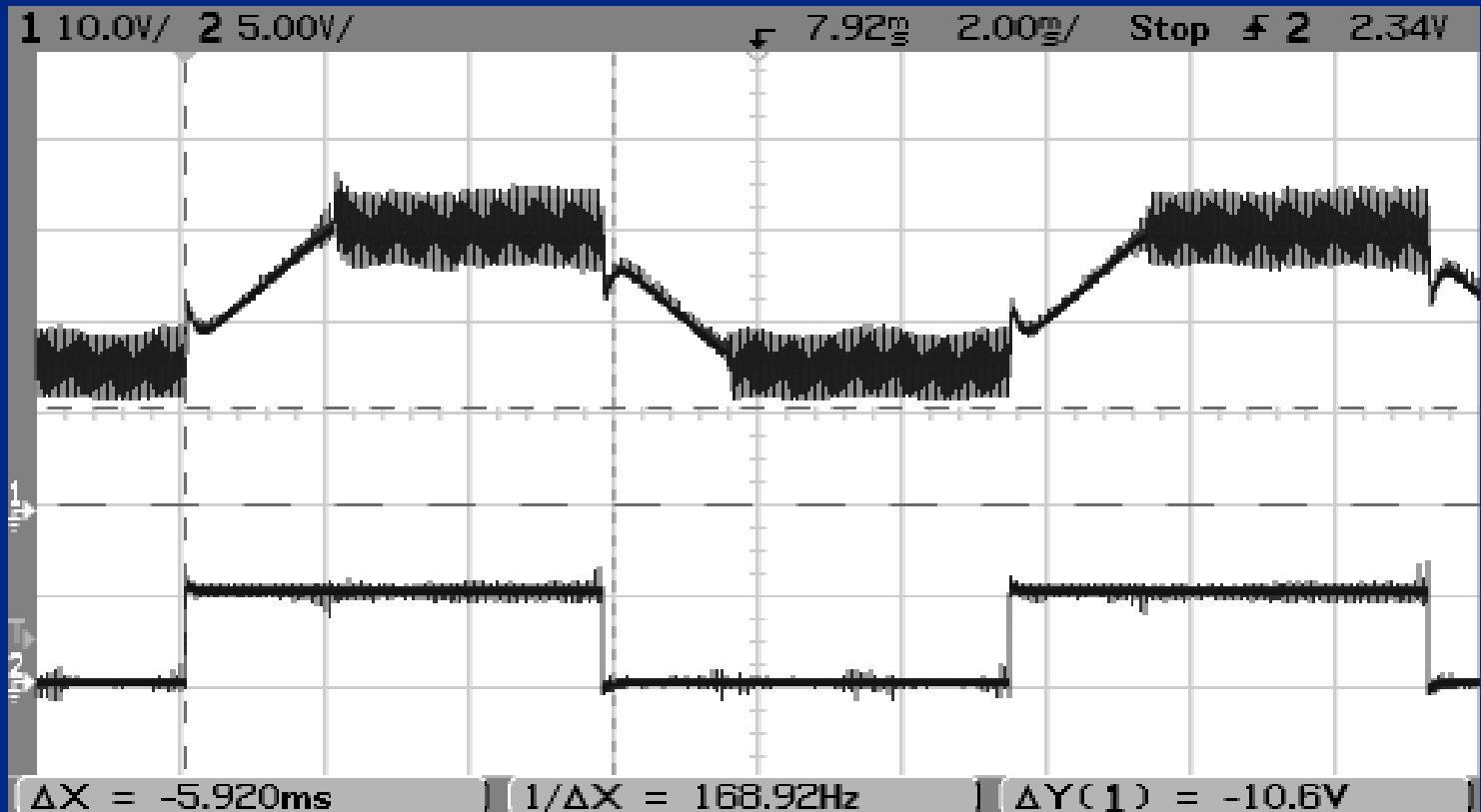
dsPIC30F2010

Technique to Monitor Back EMF

- Back EMF signal read using A/D Channels



BEMF vs. Hall sensors



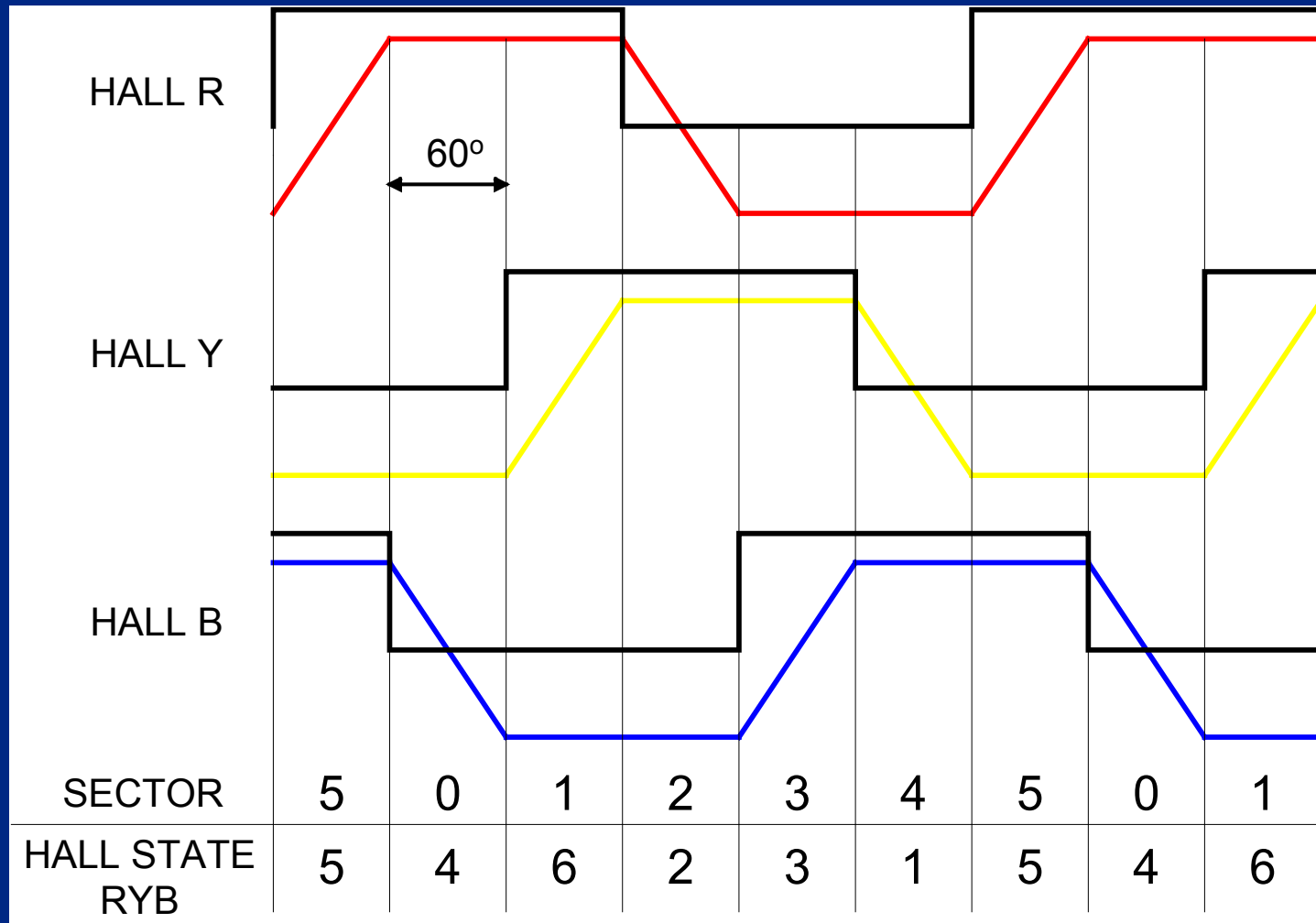


How to “Start Spinning”

- The motor is energized Open Loop (no feedback)
- The speed is ramped up to a programmable value
- The windings are then de-energized, so the rotor will coast
- The windings are then monitored for two rising edges (120 degree information)
- From the time and sequencing of the edges we can determine the speed and rotation direction
- The BEMF sensing algorithm is now applied to rotate the motor



Standard BLDC Position Sensing

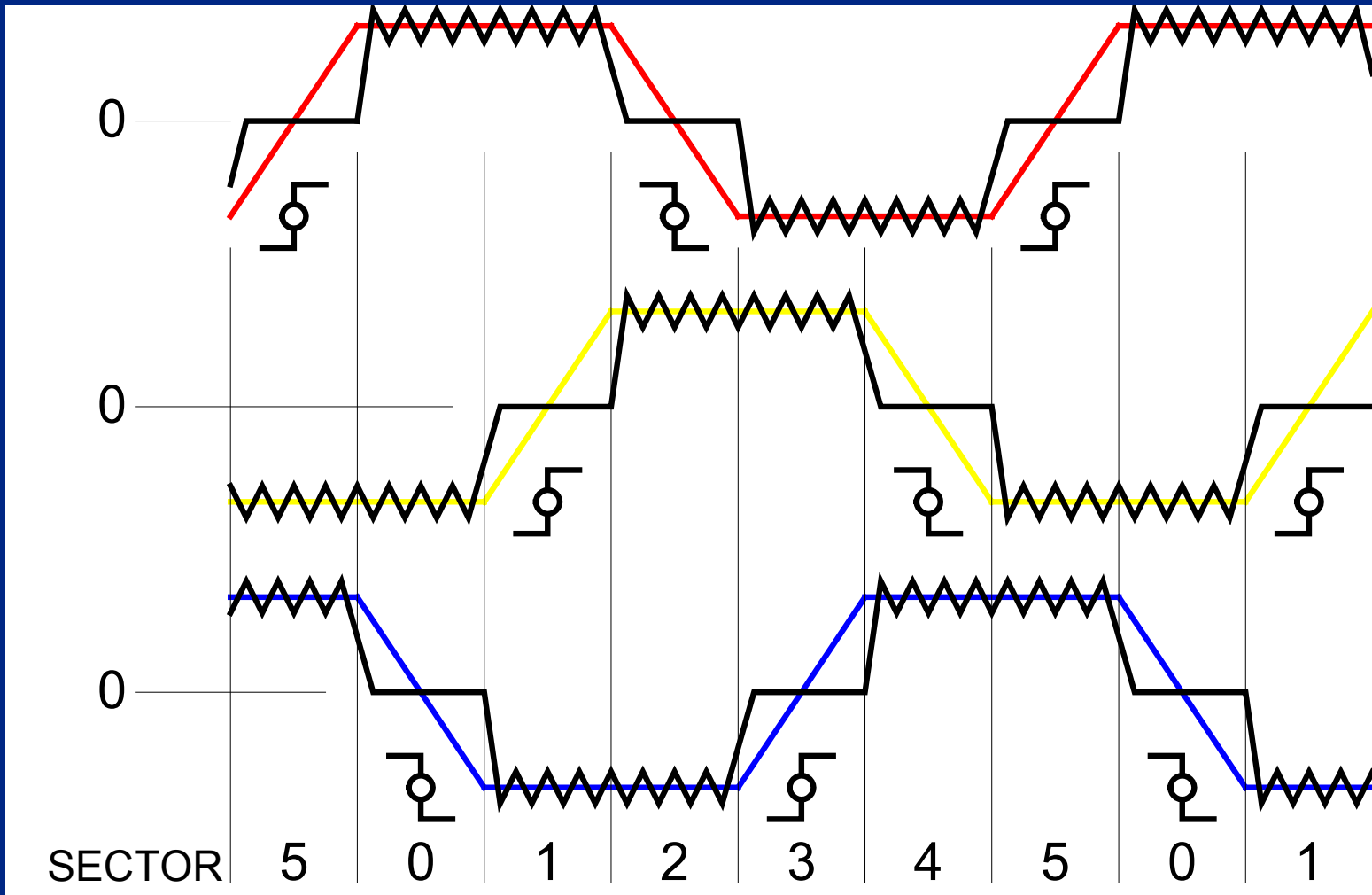




The Back EMF “zero crossing” method in detail

- In every electrical cycle, there are periods when each phase is not being driven.
- During these regions one end of the inactive phase is referenced to the star point and the other is monitored.
- The monitored voltage will cross the $1/2 V_{dd}$ point at 30 electrical degrees.
- Knowing the last “zero crossing” time we now know the 60 electrical degree time.
- That value is divided by 2 and loaded in a second timer.
- The ISR of the 2nd timer then commutes the windings at 30 electrical degrees later.

Back EMF Crossing Diagram





dsPIC[®] DSC Implementation

- High speed ADC of dsPIC DSC essential for high sampling rate
- Voltage monitor is done by resistor dividers
 - Lower cost for customer
 - no external comparators used
- Single chip solution for customer
 - cost advantage
- PID control done using the dsPIC DSC engine
- AN901 is used to discuss the Sensorless BLDC technique using a dsPIC DSC



Starting Algorithm Parameters used in AN901

- Lock Position 1 Time
 - Before start motor is rotated to a known position
 - The time is amount of time that the rotor is held in that position
- Lock Position 1 Demand
 - speed at which the rotor moves to the lock position
 - if value is too high then rotor may overshoot position



Starting Algorithm Parameters used in AN901 (contd.)

- Ramp Start/End Speed:
 - Open loop speed to get the rotor moving before back EMF is monitored
 - Too low a speed will not generate enough back EMF
 - Too high a speed may cause an over-current stall
 - After Ramp Start speed the system increases speed to the End over the Ramp Duration
 - Simply put: the acceleration profile



Starting Algorithm Parameters used in AN901 (contd.)

- Ramp Start/End Demand:
 - The amount of “torque” required to spin the motor without slipping
 - If the rotor appears to be spinning slowly as the ramp time proceeds then the ramp demand needs to be increased
 - If the whole motor vibrated when the ramp time increases then the demand is too high and most likely the over current will trip.



Lab5 Jumper settings

- **J2**(2 pin) - shorted
- **J15**(3 pin) - shorted between pin 2 & 3
(short link towards the crystal oscillator)
- **J7, J11 and J13** - shorted between pin 2 & 3
(short link away from ICD 2 connector)
- **JP8, JP12, JP14, JP16 and JP17** keep open
- Keep Potentiometer **REF(R14) and R60** in center position
- Disconnect Hall Sensors from Motor (Black connector)



Lab 5: Spinning a Sensorless BLDC Motor

- Lab 5:
 - Use workspace
“C:\MASTERS\845\Lab5\Lab5.mcw”
 - Disconnect Black Connector from Motor
 - Change Setup parameters to start BLDC motor
 - Lock Position 1 Time
 - Lock Position 1 Demand
 - Ramp Start/End Speed
 - Ramp Start/End Demand
 - Ramp Duration



Lab 5: Spinning a Sensorless BLDC Motor (contd.)

- Lab5:
 - Change values as per handout
 - Compile program for no errors
 - Program part with ...
 - DIPs in ON position
 - Run program with DIPs in OFF position
 - Press S2 to run the motor



Summary

- BLDC motor basics
- dsPIC[®] DSC Architecture and Peripherals
- Programmed the dsPIC DSC ADC peripheral
- Programmed the dsPIC DSC PWM peripheral
- Controlled a Sensored BLDC Motor
- Controlled a Sensorless BLDC Motor



Thank you for your attendance

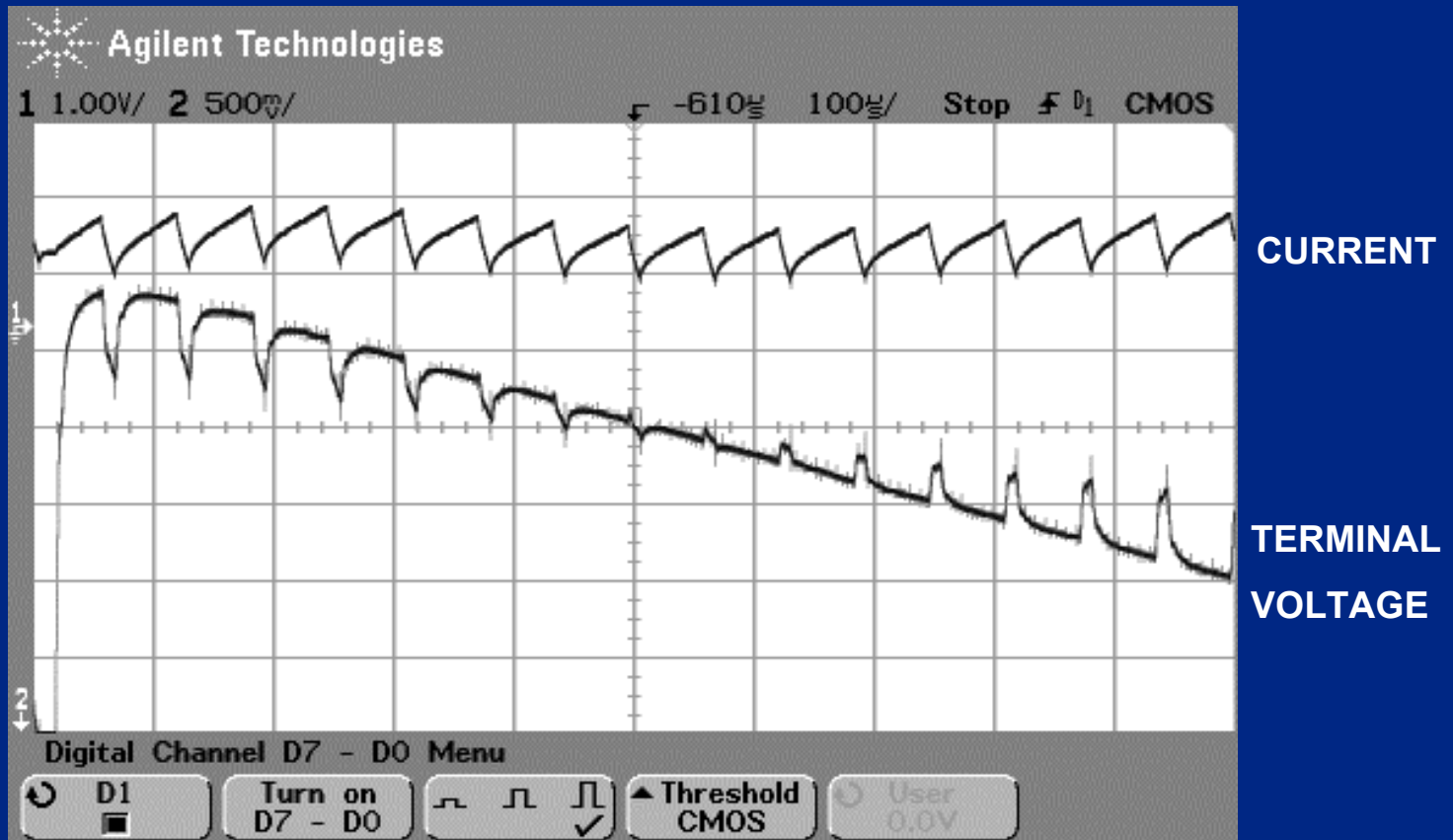


Sensorless BLDC Issues

- Phase winding demagnetisation
 - Stored energy in winding is automatically returned to the supply via the inverter diodes in a finite time.
 - BEMF sampling is blanked just after commutation
- Coupling from active phases giving PWM “noise”
 - Adjacent phases induce PWM noise on the inactive phase
 - ADC sampling is done asynchronously with the PWM in order to avoid the “PWM” noise spikes.



Trace of PWM “Noise” on Inactive Phase Terminal Voltage



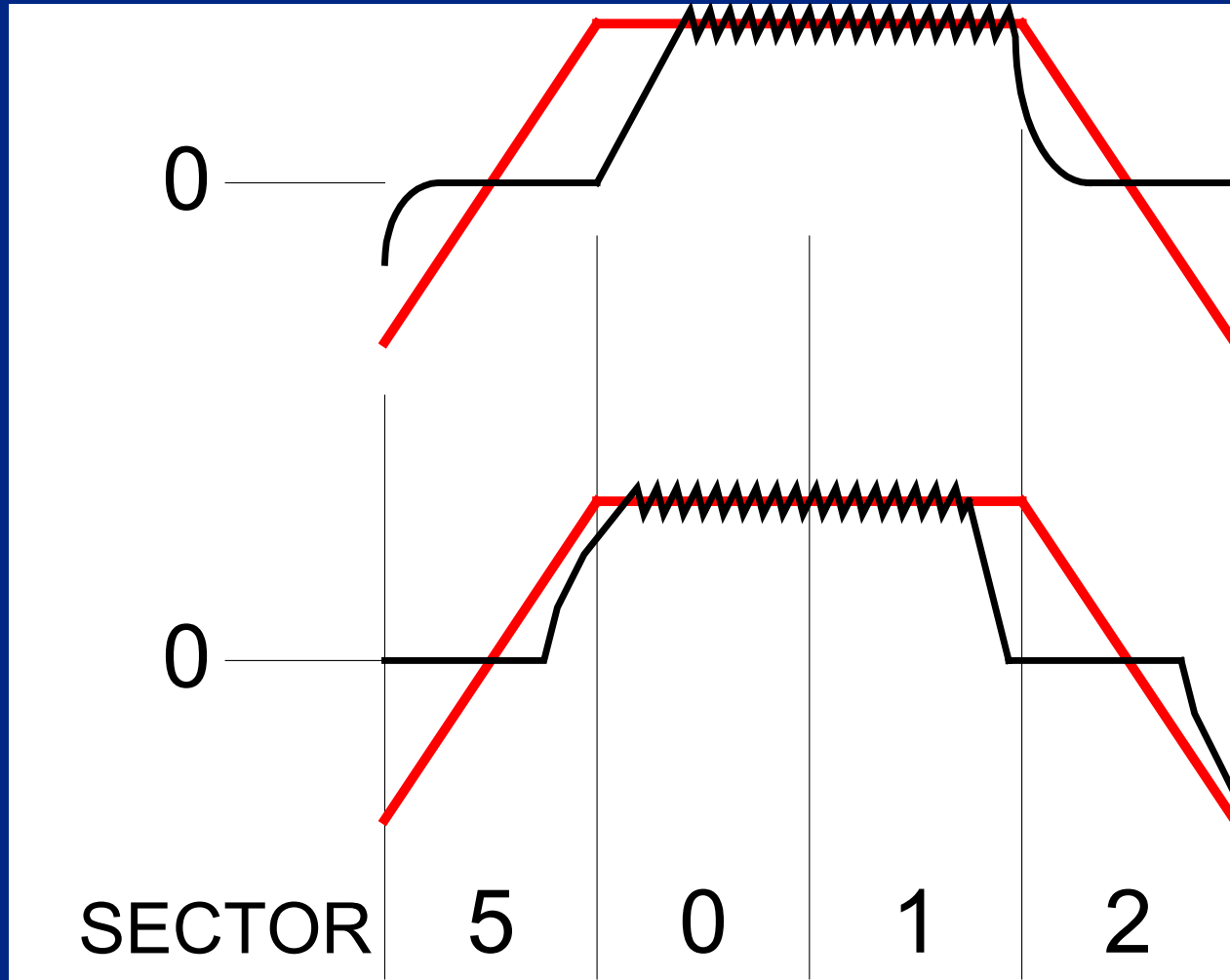


Phase Advance

- The 30 electrical degree commutation time is determined in the BEMF algorithm
- Since current lags the voltage, energizing the winding at the precise 30 deg. time may not give us the right or adequate torque
- Phase advance allows us to pre-energize the winding slightly ahead of the 30 electrical degree commutation



Plots showing the effect of Phase Advance at High Speed



NO ADVANCE

15° ADVANCE