Putting the New 8-bit PIC® MCU Peripherals to Work in the Real World

1610 CLC

Objectives

At the end of this class you will be able to describe the function of and implement designs using the PIC16F1509

- **Configurable Logic Cell (CLC)**
- **Numerically Controlled Oscillator (NCO)**
- **Complimentary Waveform Generator (CWG)**

Agenda

- Overview of the new Peripherals
  - CLC
  - CLC Design Tool
    - Lab 1 Manchester Encoder
  - NCO
    - Lab 2 Manchester Decoder
  - CWG
    - Lab 3 Driving a motor using the PIC Communicator
    - Lab 4 Combining all three modules to send motor speed

Configurable Logic Cell (CLC)

Configurable Logic Cell

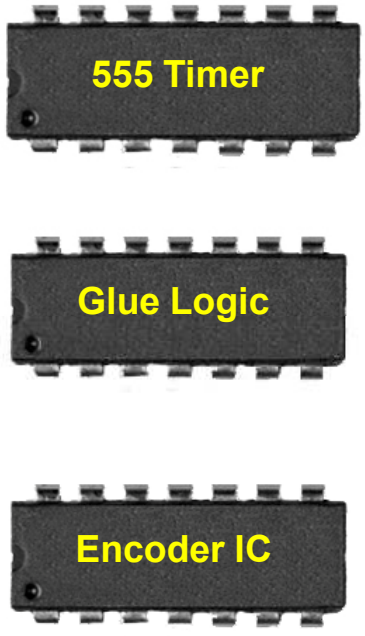
What does it do?

Performs one of eight single output logic functions on a selection of 4 of 16 inputs. Operates outside the speed limitations of software execution.

Benefits:

- **Increases on chip interconnection**
- **Replaces External Components**
- **Saves Code Space - function as a programmable logic device**

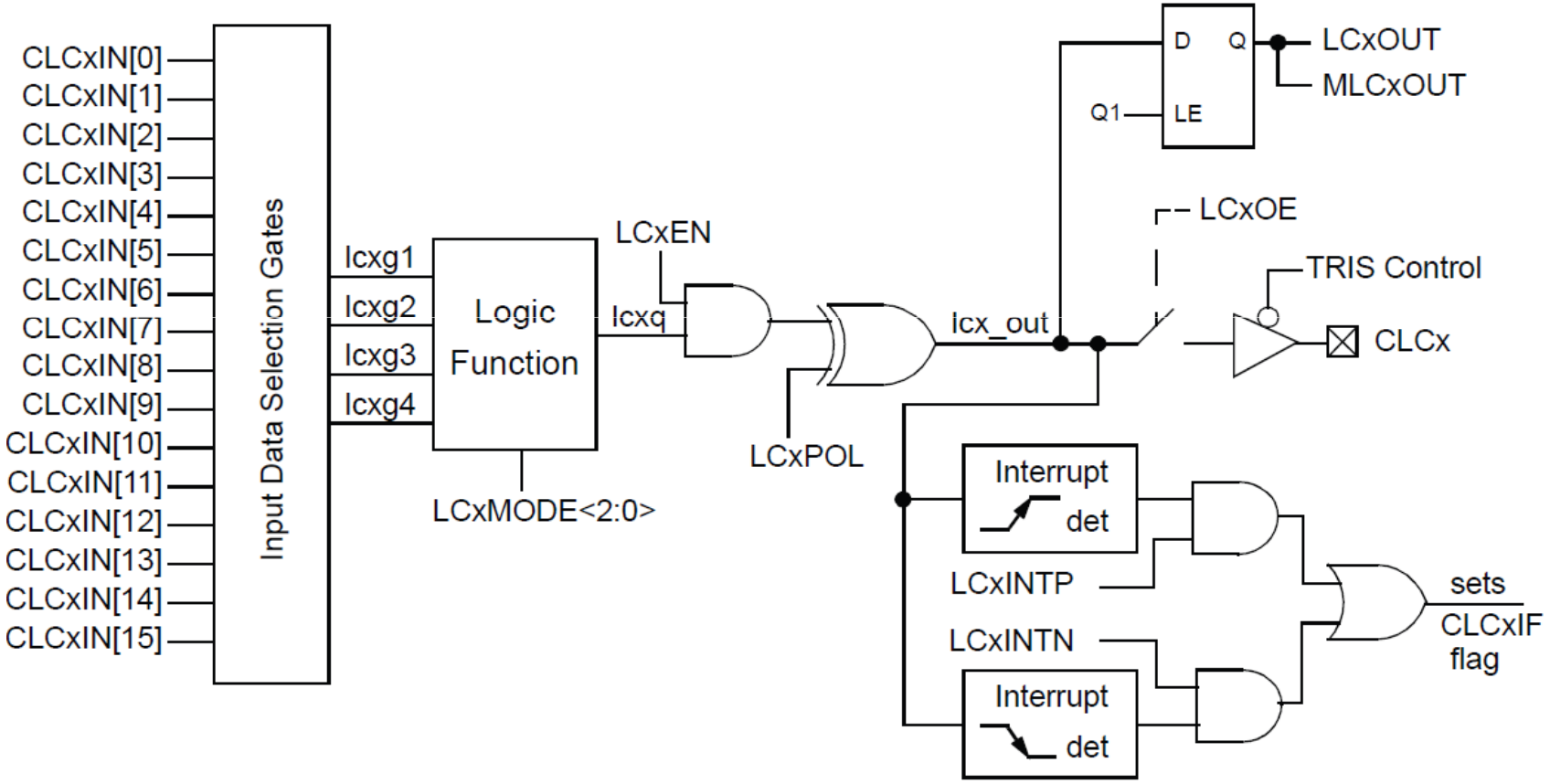
Configurable Logic
= Consolidation



PIC® MCU

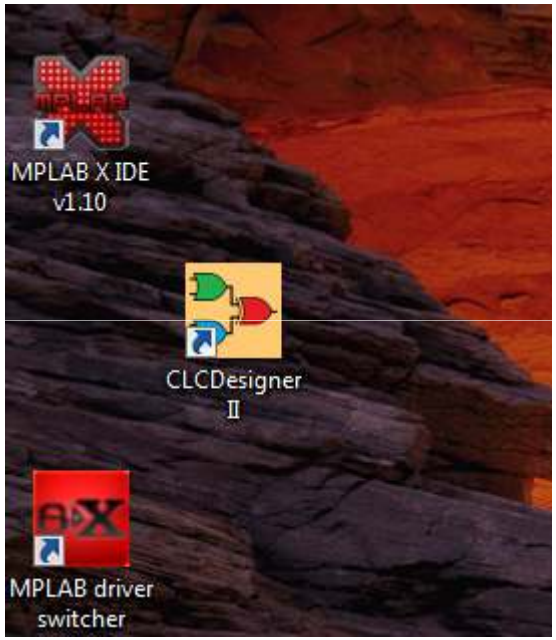
Code-free implementation with many PIC® MCUs

Configurable Logic Cell

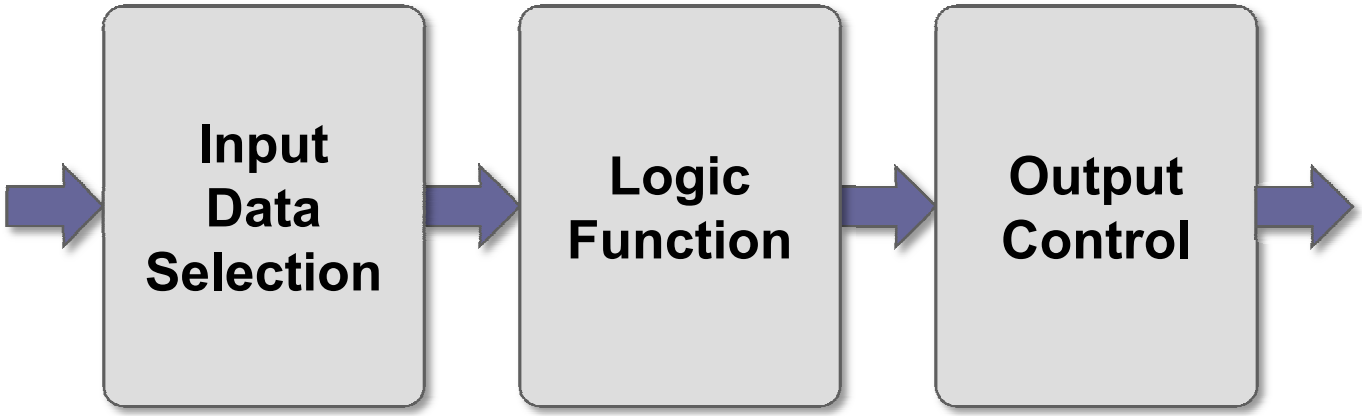


CLC Designer

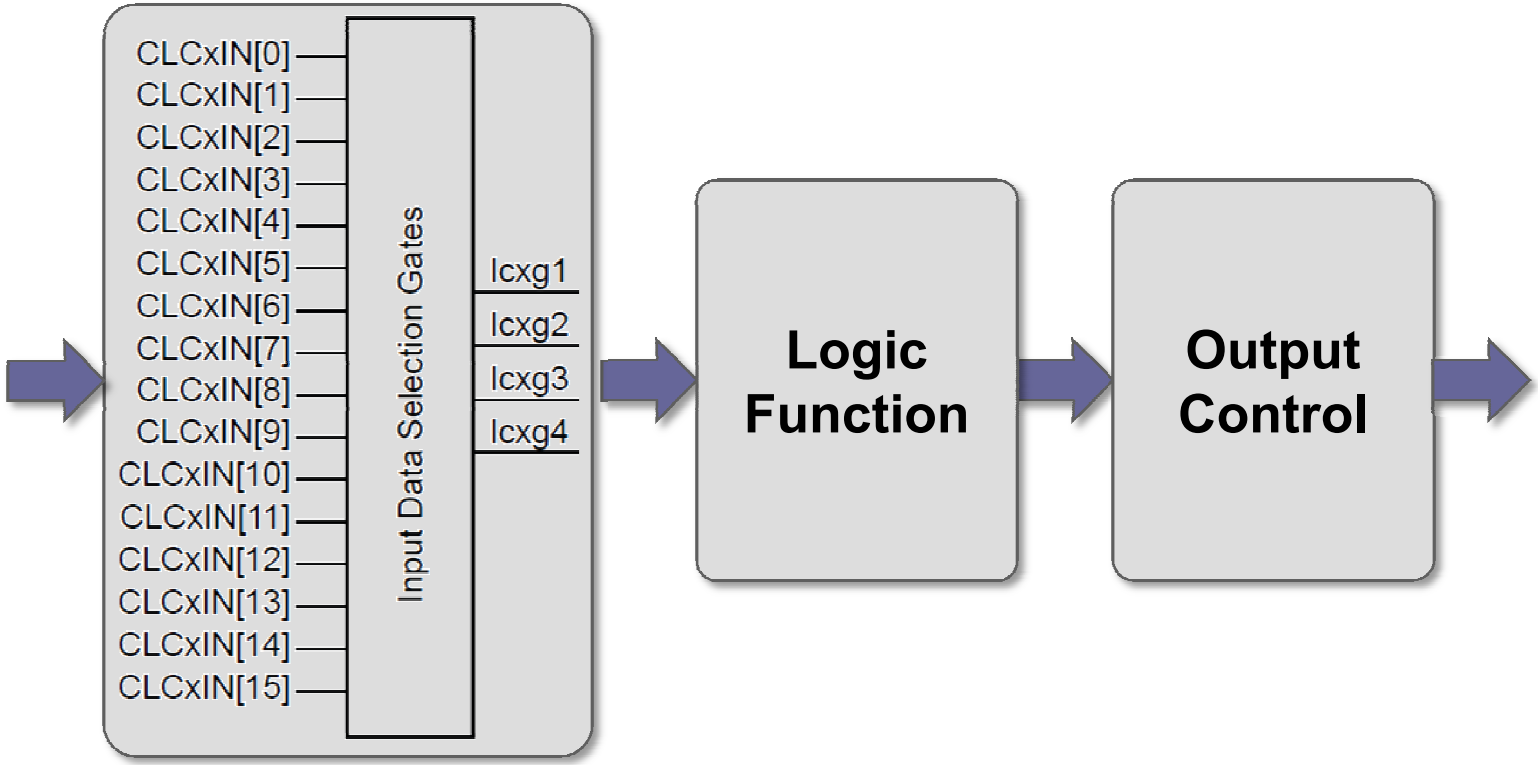
CLC Designer



Configurable Logic Cell
Peripheral Setup

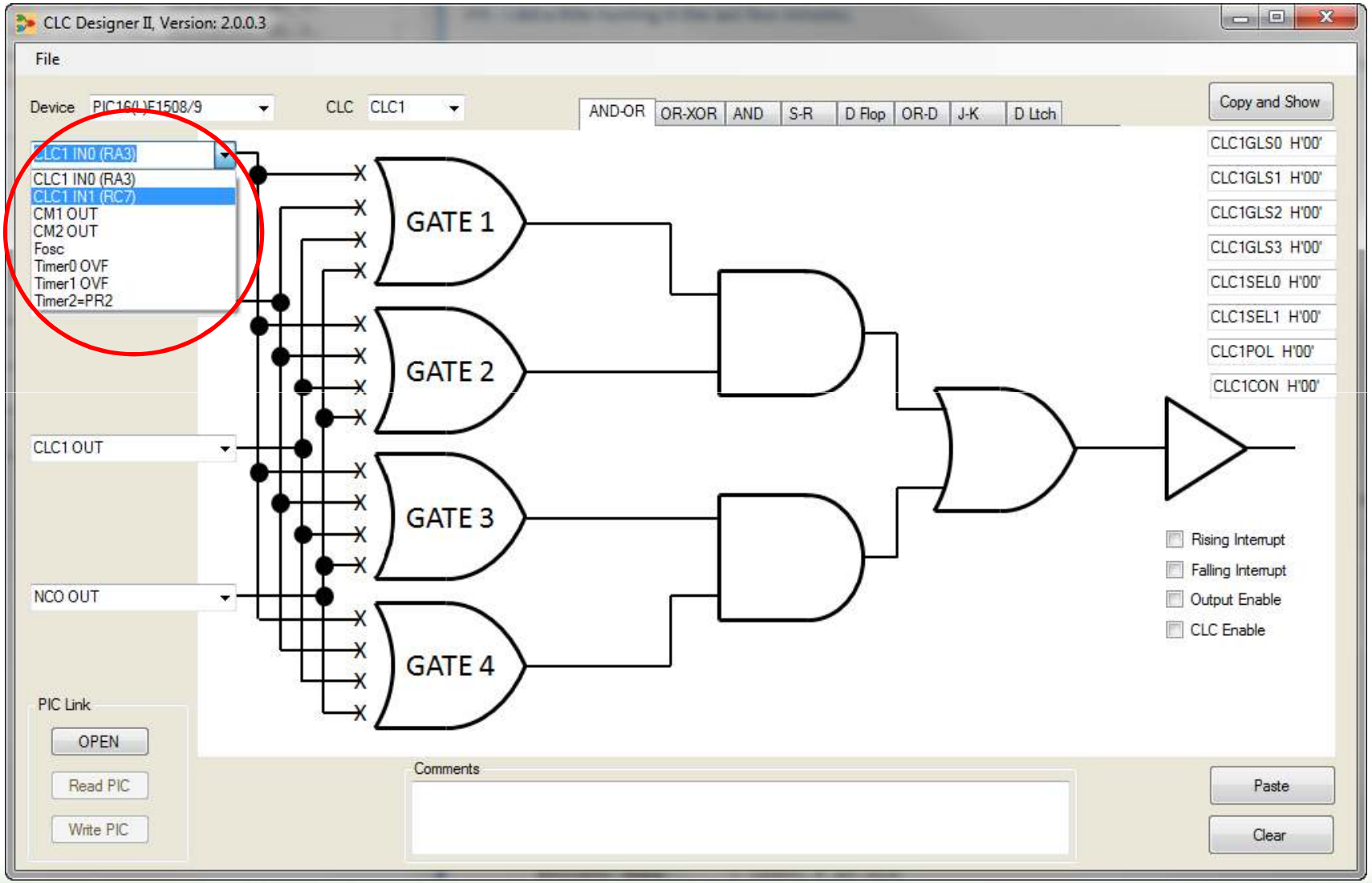


Configurable Logic Cell
Peripheral Setup

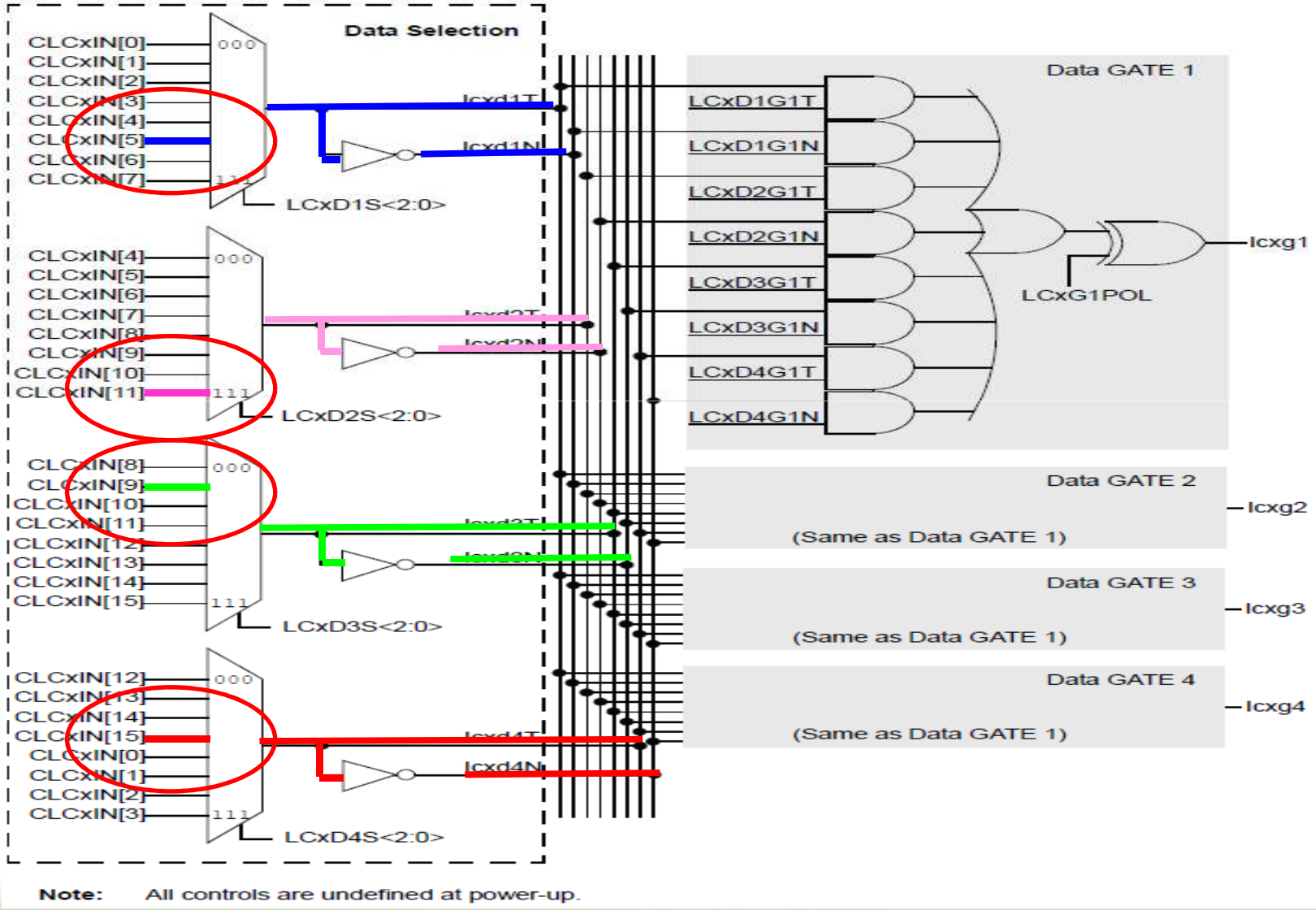


Input Data Selection and Data Gating

CLC Design Tool



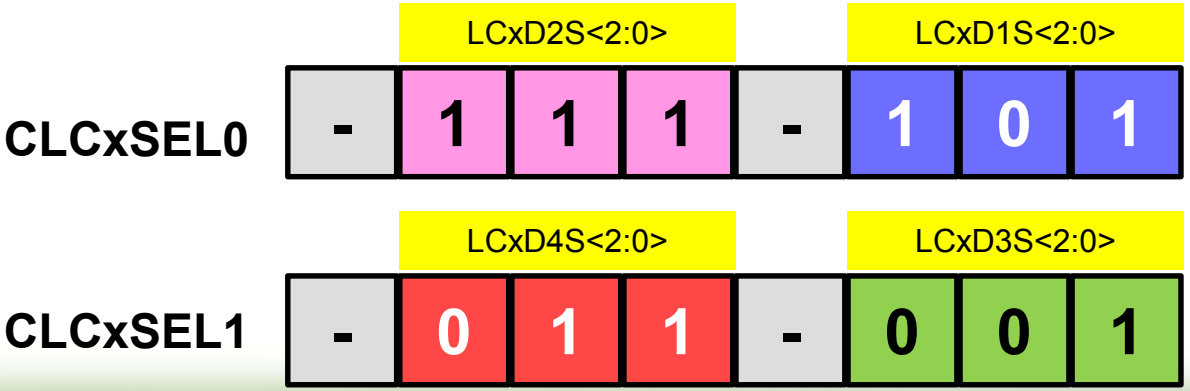
Configurable Logic Cell



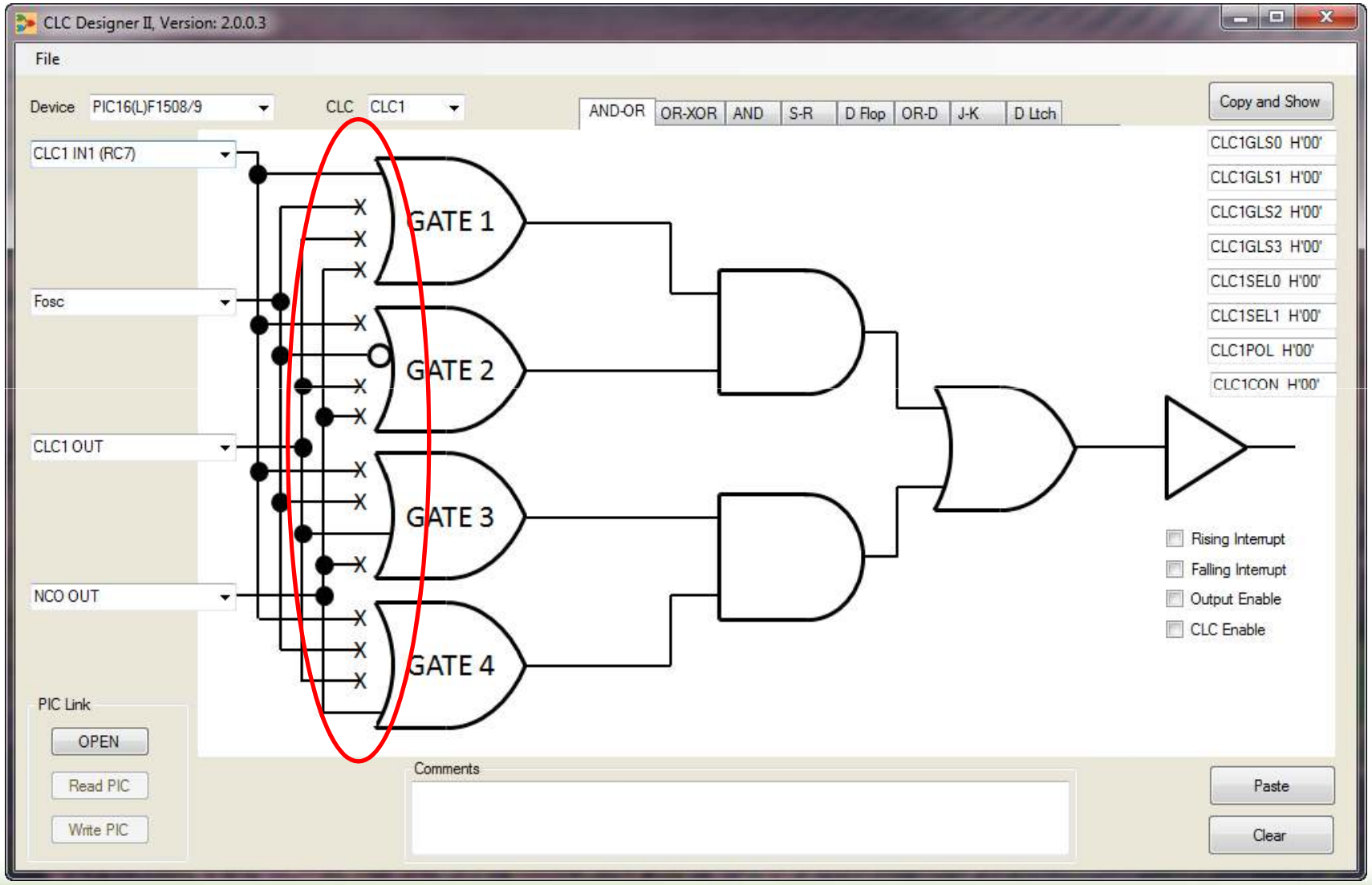
Note: All controls are undefined at power-up.

Configurable Logic Cell

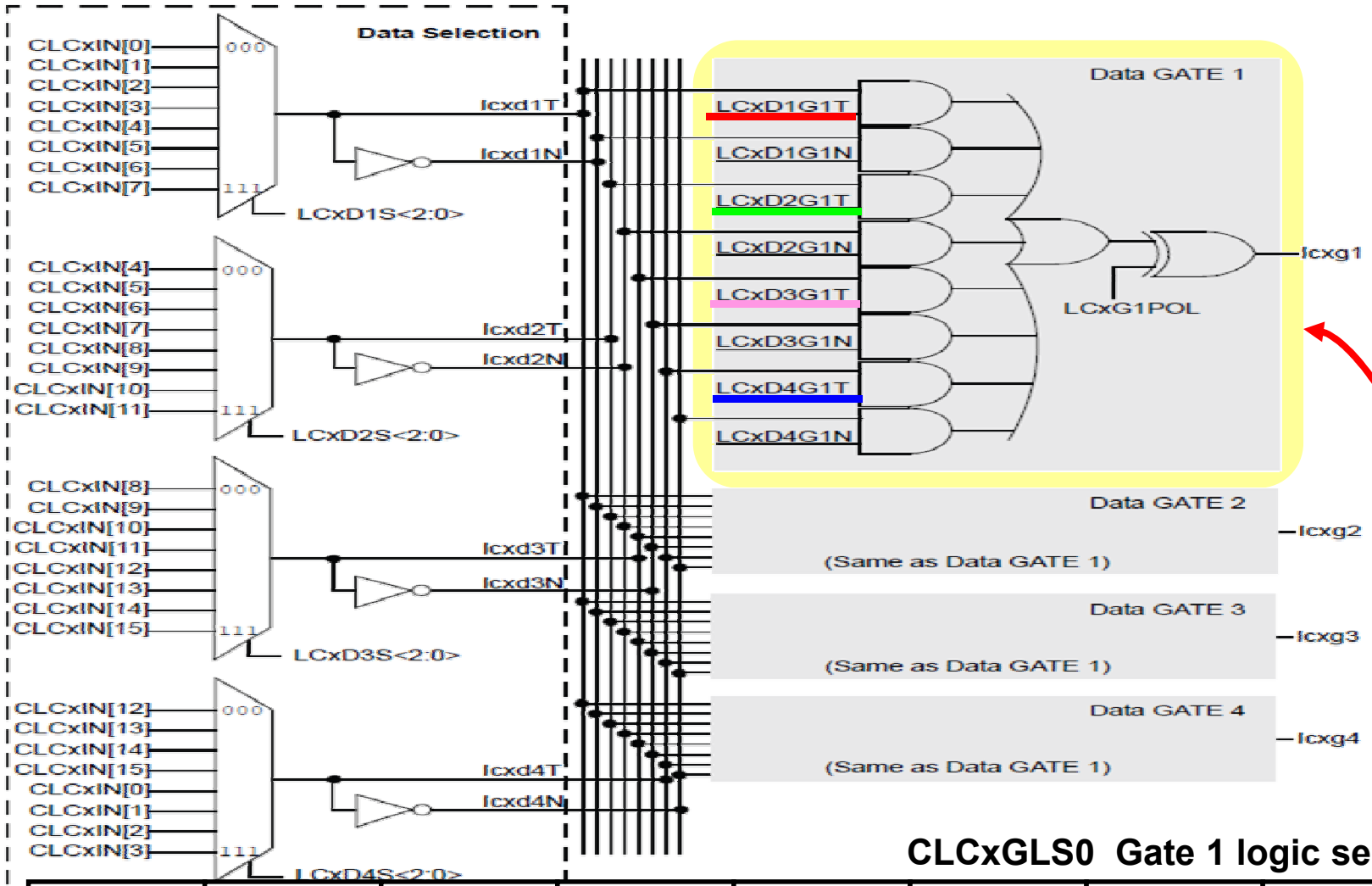
Data Input	lcxd1 D1S	lcxd2 D2S	lcxd3 D3S	lcxd4 D4S	CLC 1	CLC 2	CLC 3	CLC 4
CLCxIN[0]	000	—	—	100	CLC1IN0	CLC2IN0	CLC3IN0	CLC4IN0
CLCxIN[1]	001	—	—	101	CLC1IN1	CLC2IN1	CLC3IN1	CLC4IN1
CLCxIN[2]	010	—	—	110	SYNCC1OUT	SYNCC1OUT	SYNCC1OUT	SYNCC1OUT
CLCxIN[3]	011	—	—	111	SYNCC2OUT	SYNCC2OUT	SYNCC2OUT	SYNCC2OUT
CLCxIN[4]	100	000	—	—	FOSC	FOSC	FOSC	FOSC
CLCxIN[5]	101	001	—	—	TMR0IF	TMR0IF	TMR0IF	TMR0IF
CLCxIN[6]	110	010	—	—	TMR1IF	TMR1IF	TMR1IF	TMR1IF
CLCxIN[7]	111	011	—	—	TMR2 = PR2	TMR2 = PR2	TMR2 = PR2	TMR2 = PR2
CLCxIN[8]	—	100	000	—	lc1_out	lc1_out	lc1_out	lc1_out
CLCxIN[9]	—	101	001	—	lc2_out	lc2_out	lc2_out	lc2_out
CLCxIN[10]	—	110	010	—	lc3_out	lc3_out	lc3_out	lc3_out
CLCxIN[11]	—	111	011	—	lc4_out	lc4_out	lc4_out	lc4_out
CLCxIN[12]	—	—	100	000	NCO1OUT	LFINTOSC	TX (EUSART)	SCK (MSSP)
CLCxIN[13]	—	—	101	001	HFINTOSC	ADFRC	LFINTOSC	SDO (MSSP)
CLCxIN[14]	—	—	110	010	PWM3OUT	PWM1OUT	PWM2OUT	PWM1OUT
CLCxIN[15]	—	—	111	011	PWM4OUT	PWM2OUT	PWM3OUT	PWM4OUT



CLC Design Tool



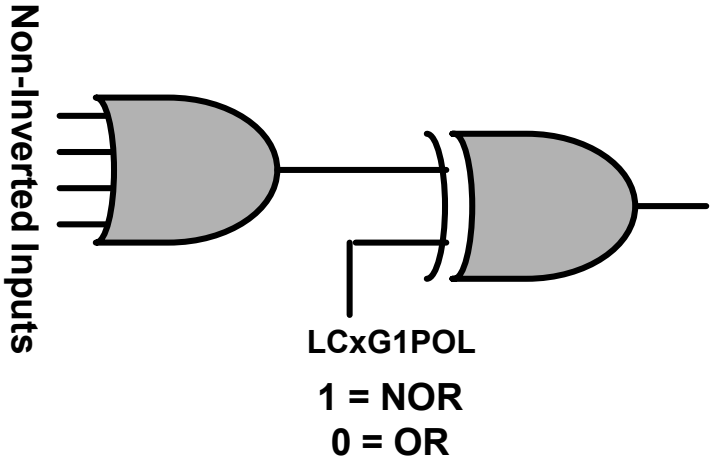
Configurable Logic Cell



CLCxGLS0 Gate 1 logic select

LCxD1G1T	LCxD1G1N	LCxD2G1T	LCxD2G1N	LCxD3G1T	LCxD3G1N	LCxD4G1T	LCxD4G1N
1	0	1	0	1	0	1	0

Configurable Logic Cell



CLCxGLS0	LCxGyPOL	Gate Logic
0x55	1	AND
0x55	0	NAND
0xAA	1	NOR
0xAA	0	OR
0x00	0	Logic 0
0x00	1	Logic 1

CLCxPOL

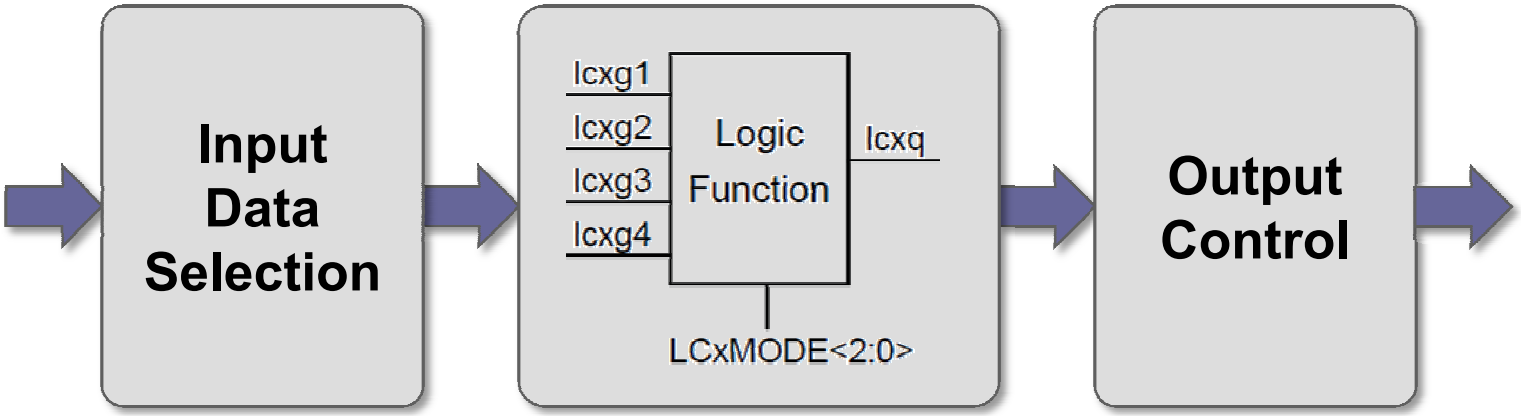
LCxPOL	–	–	–	LCxG4POL	LCxG3POL	LCxG2POL	LCxG1POL
							0

CLCxGLS0 Gate 1 logic select

LCxG1D4T	LCxG1D4N	LCxG1D3T	LCxG1D3N	LCxG1D2T	LCxG1D2N	LCxG1D1T	LCxG1D1N
1	0	1	0	1	0	1	0

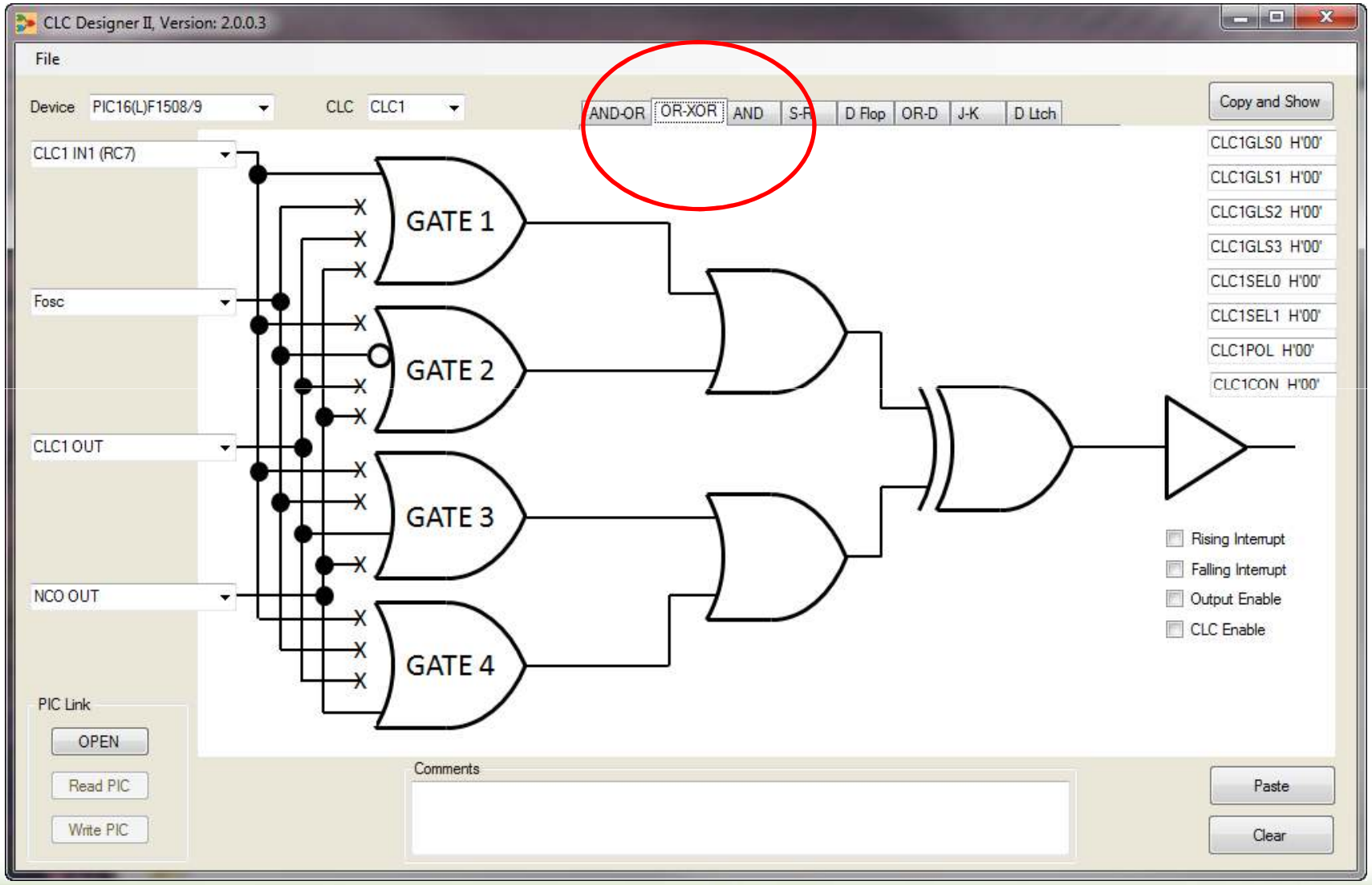
Non-Inverted Inputs (T) Selected

Configurable Logic Cell
Peripheral Setup



Logic Function

CLC Design Tool

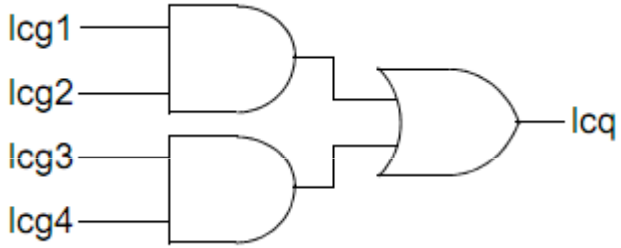


Configurable Logic Cell

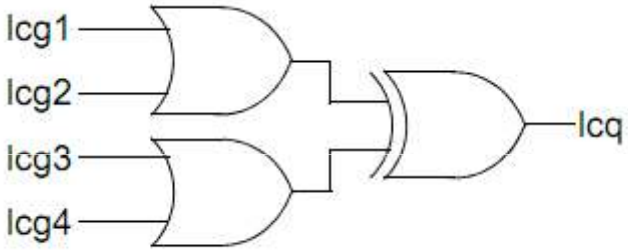
Logic Function Selection

Select the desired logic function using the **LCxMODE<2:0>** bits of the **CLCxCON** register.

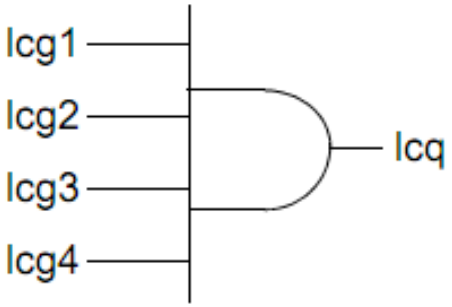
000 = AND – OR



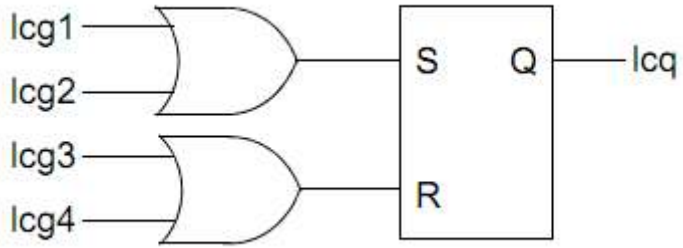
001 = OR – XOR



010 = 4-Input AND



011 = S-R Latch

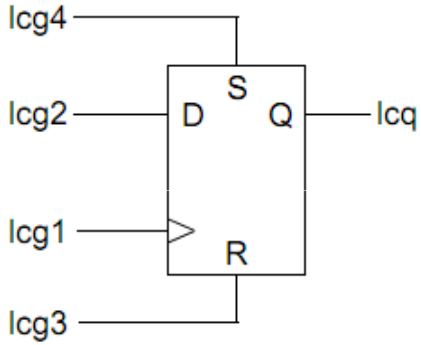


Configurable Logic Cell

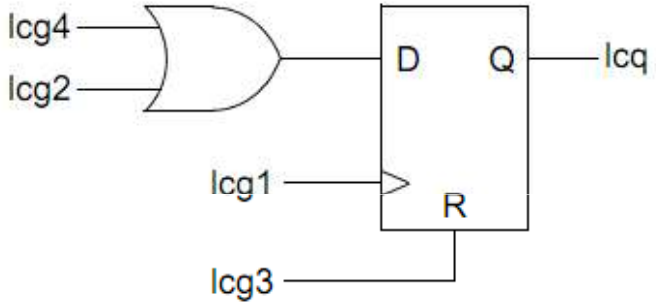
Logic Function Selection

Select the desired logic function using the **LCxMODE<2:0>** bits of the **CLCxCON** register.

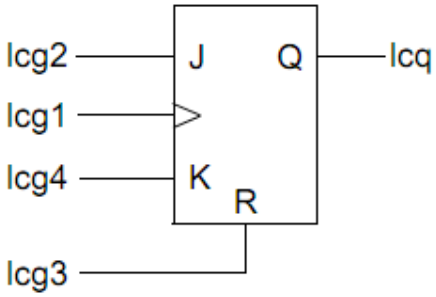
100 = 1-Input D Flip-Flop with S and R



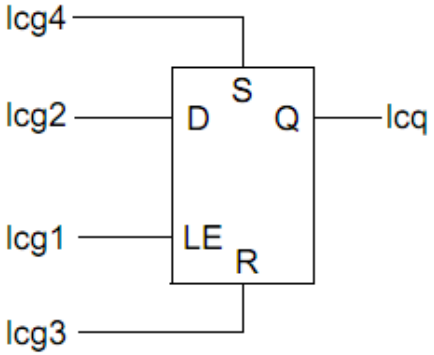
101 = 2-Input D Flip-Flop with R



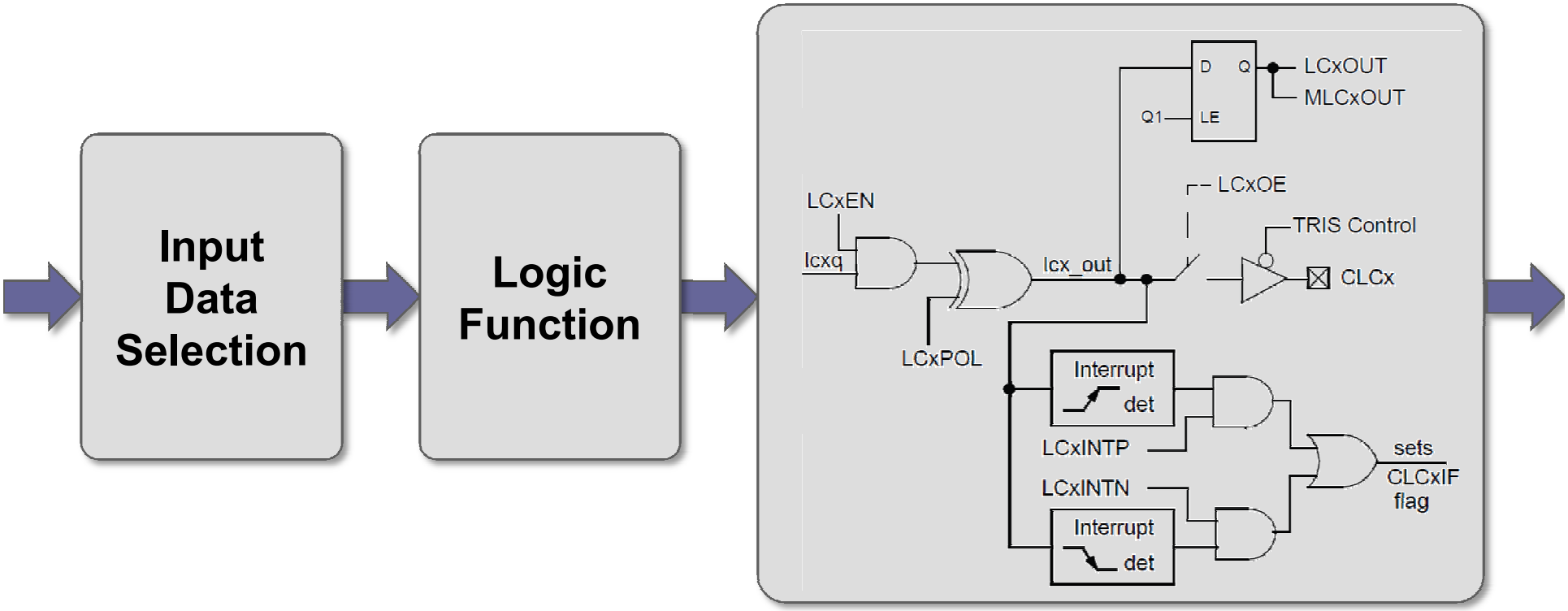
110 = J-K Flip-Flop with R



111 = 1-Input Transparent Latch with S and R

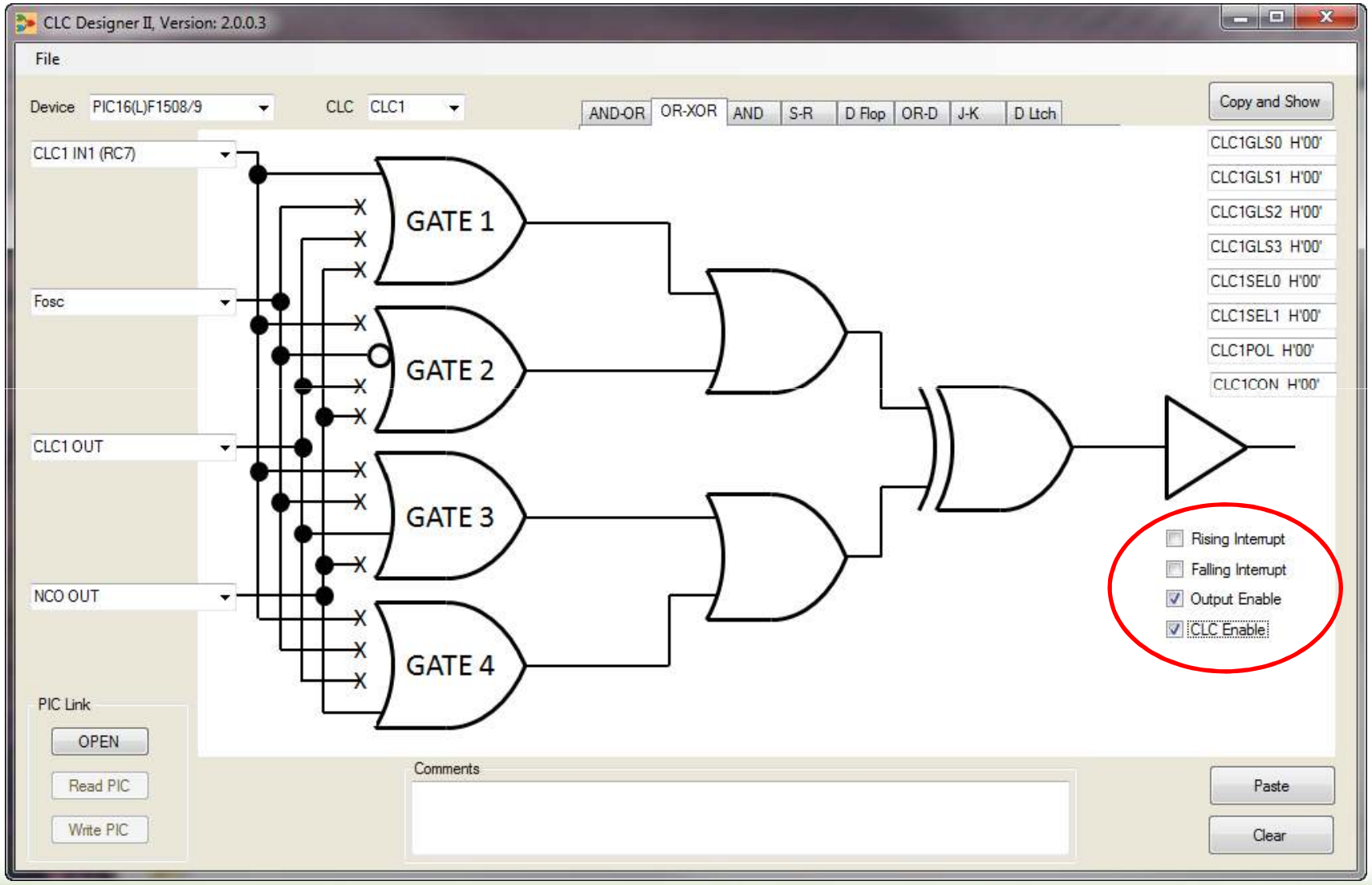


Configurable Logic Cell
Peripheral Setup

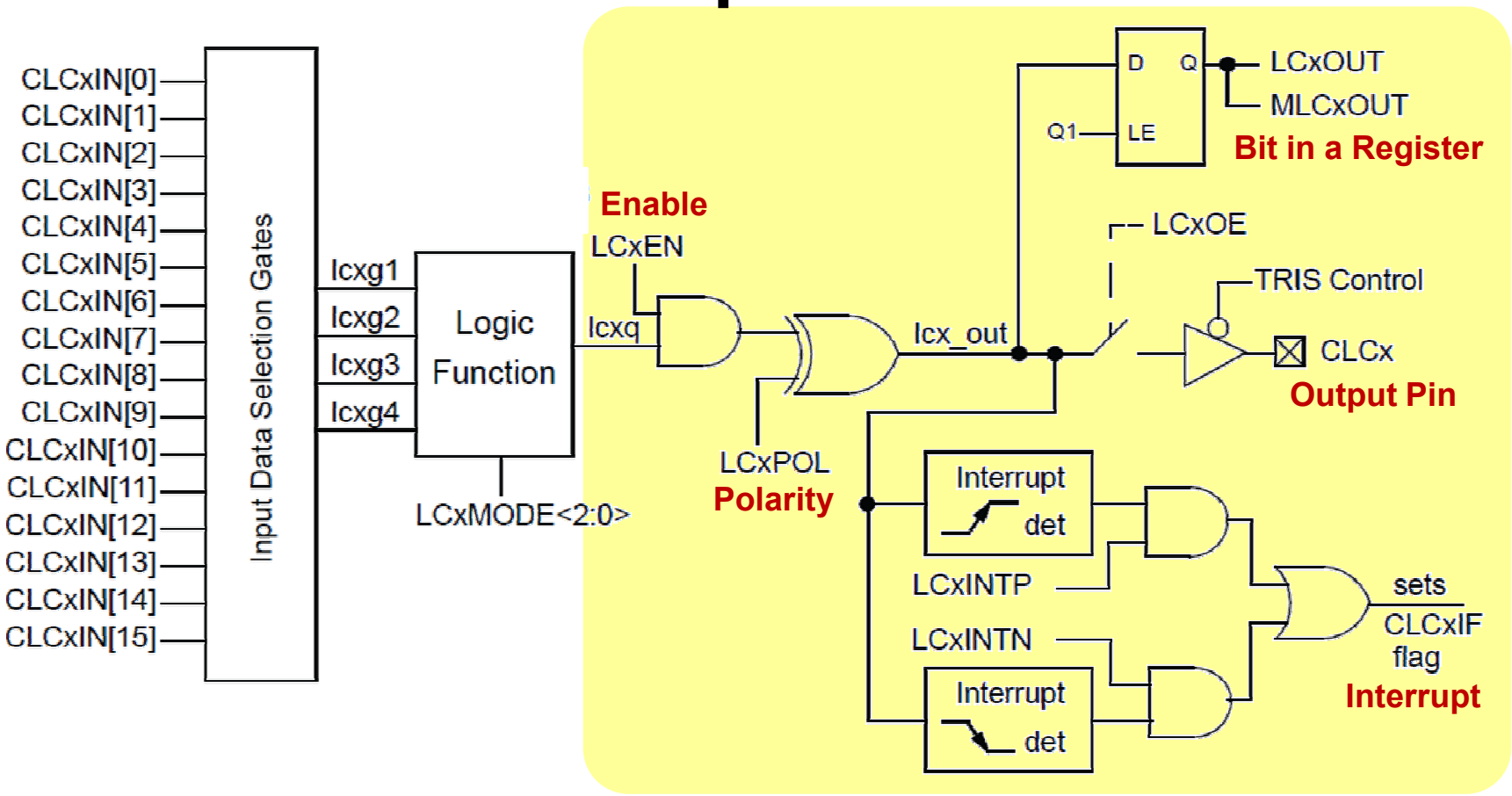


Output Control

CLC Design Tool



Configurable Logic Cell
Output Control

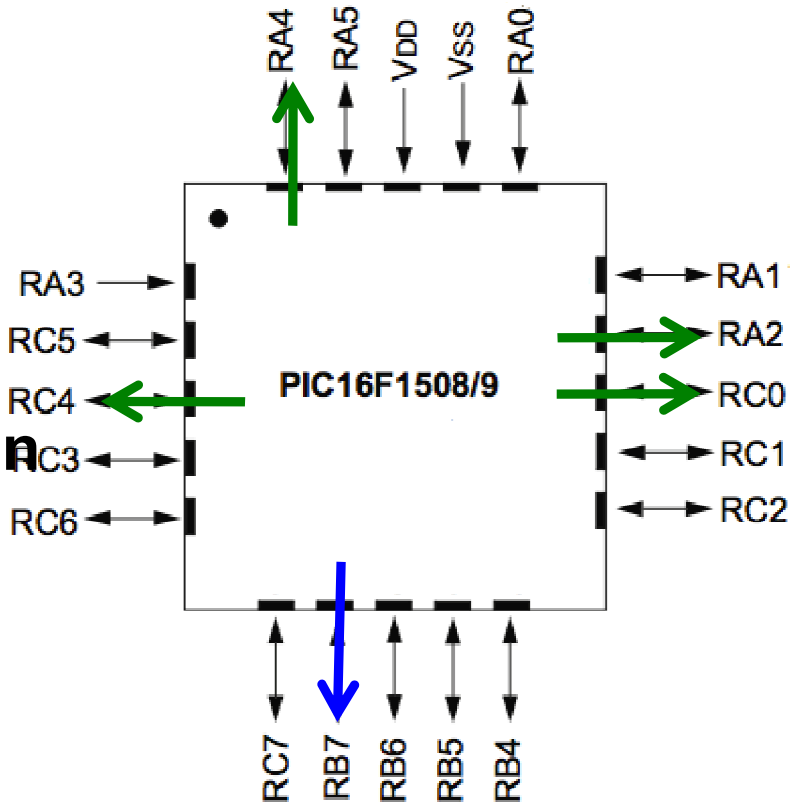


REGISTER 24-1: CLCxCON: CONFIGURABLE LOGIC CELL CONTROL REGISTER

R/W-0/0	R/W-0/0	R-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LCxEN	LCxOE	LCxOUT	LCxINTP	LCxINTN	LCxMODE<2:0>		
bit 7							bit 0

Application Example:
Output Pin Re-Routing

- Route signals without adding additional layers
- Cost effective solution
- Crucial in small form factor applications

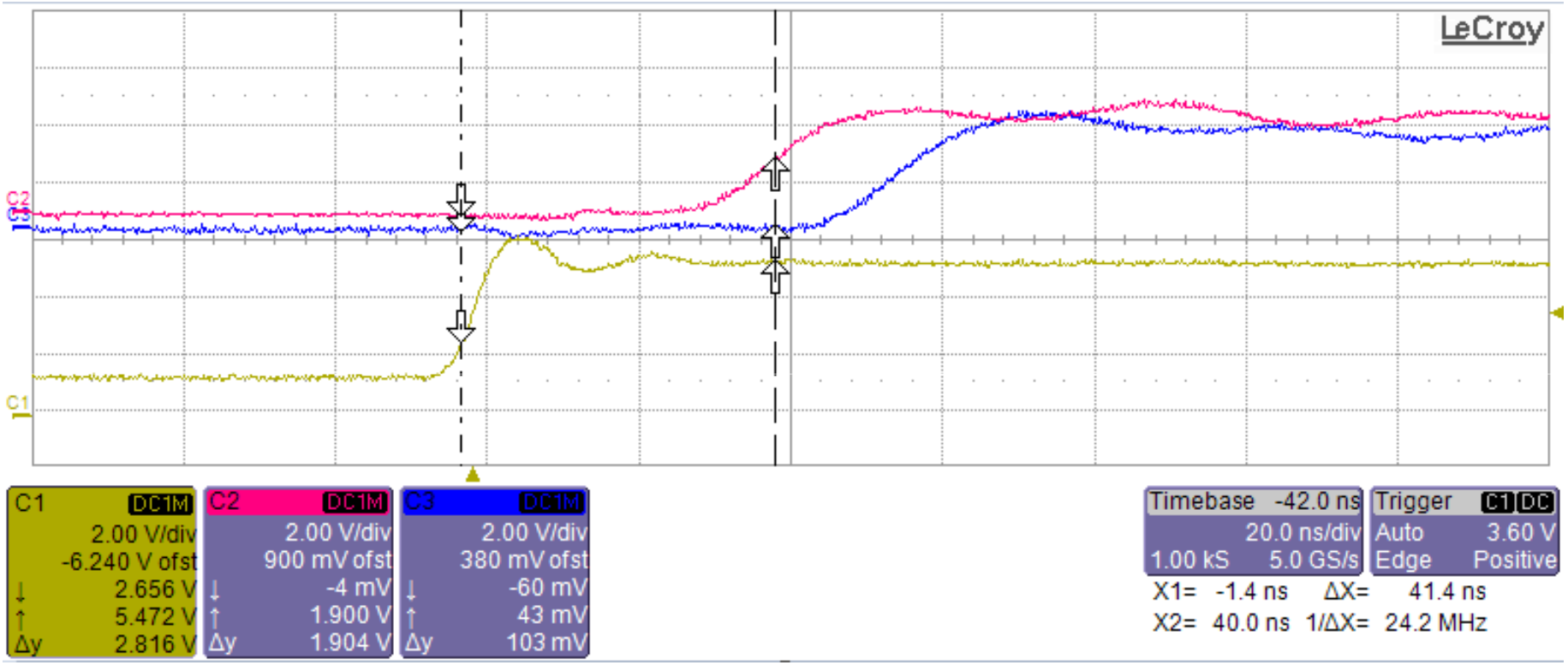


Example:

- RB7 = EUSART transmit pin
  - Re-route using CLC
  - Signal can be routed to:

    RC4, RC0, RA4 & RA2

CLC1 -> CLC4 : AND-OR

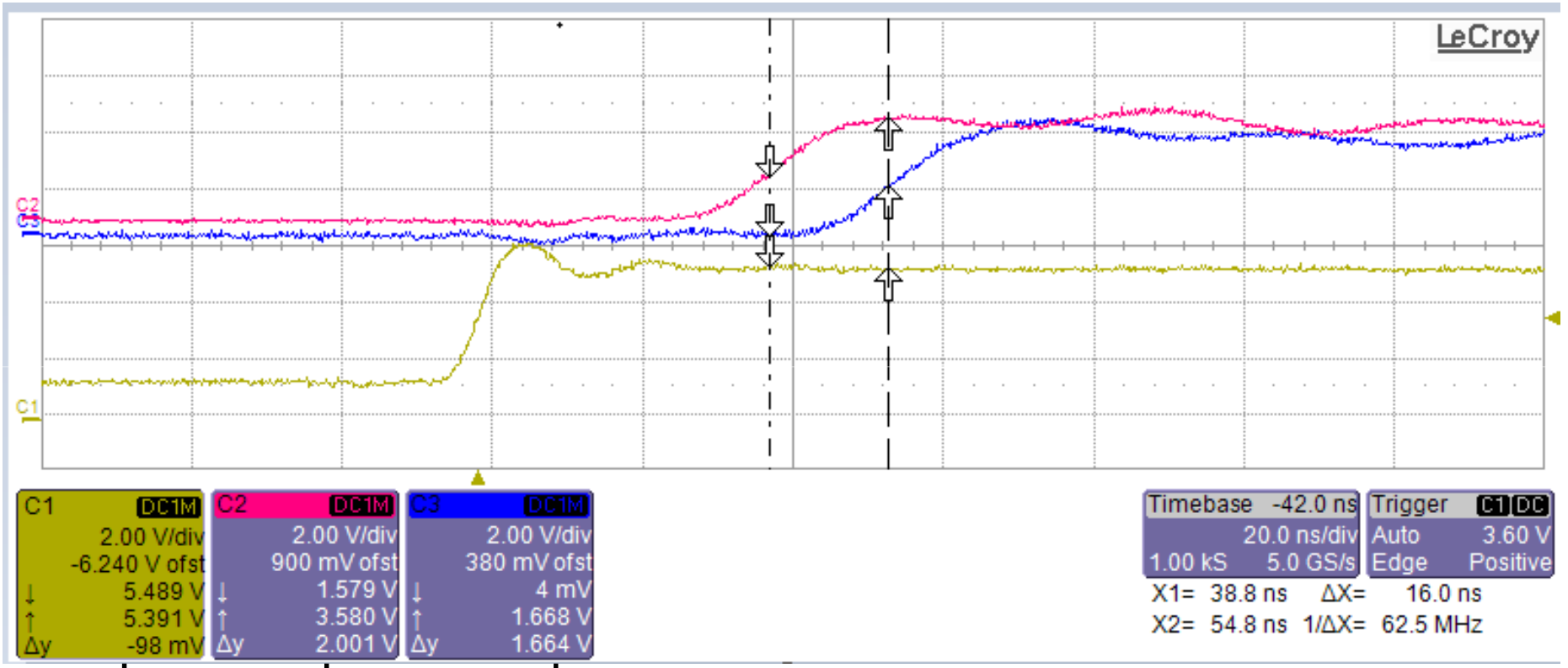


Pulse **CLC1 Out** **CLC4 Out**

DeltaX = CLC1 Out - Pulse

DeltaX ~= 40nS

CLC1 -> CLC4 : AND-OR



Pulse CLC1 Out CLC4 Out

DeltaX = CLC4 Out – CLC1 Out

DeltaX ~= 16nS

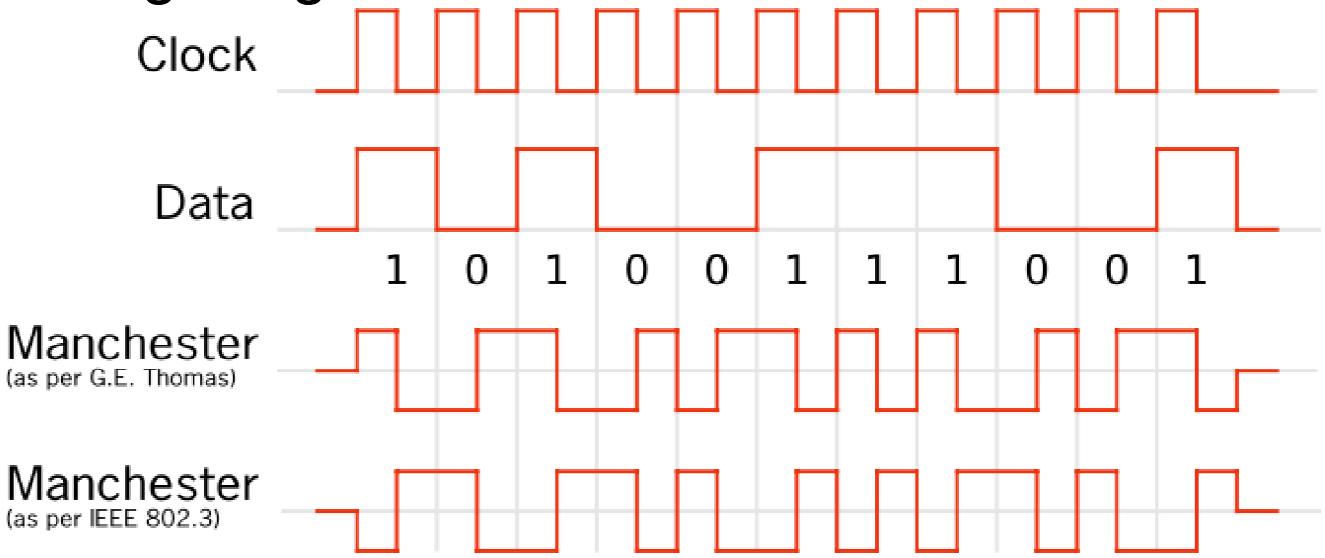
CLC Prop Delay = 16nS / 4 = **4nS**

Lab 1
Manchester Encoder

Lab 1 - Manchester Encoder

What is Manchester Encoding?

Manchester encoding is a method of encoding data by combining the clock and data lines using an exclusive OR into a single signal.

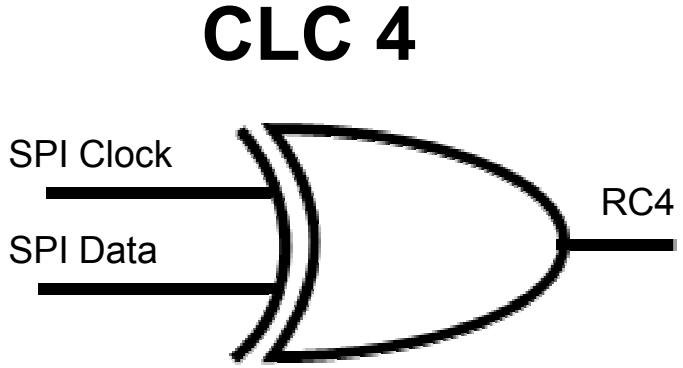


Lab 1 - Manchester Encoder

Setup

Use the CLC design tool to build and implement this circuit.

Inputs: SPI Clock, SPI Data
Output: CLC 4 Out (RC4)

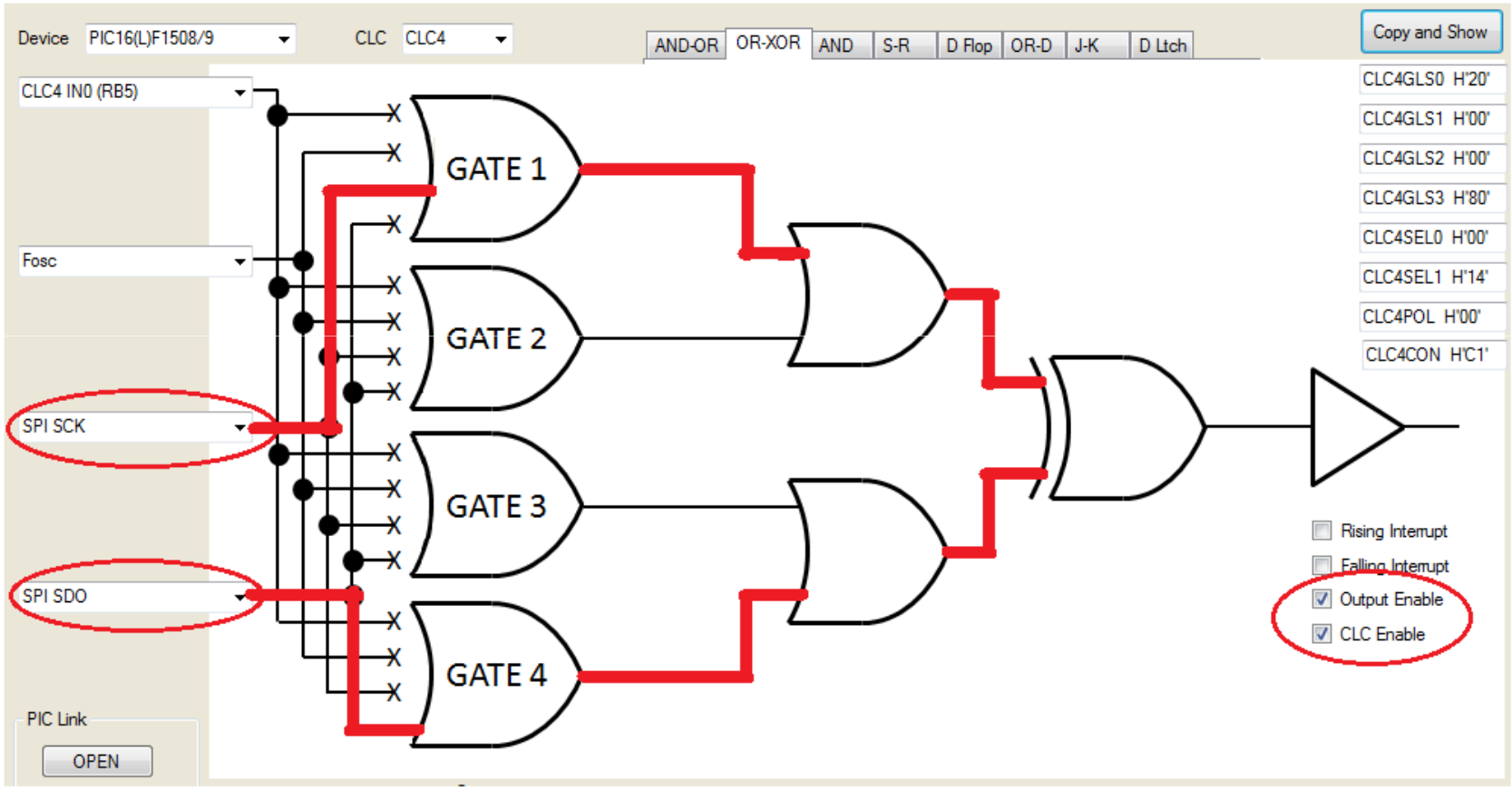


Note:

- **Make sure POT 2 is turned all the way clockwise!**
- **"S3" sends your message, not "S2"**
- **Open main.c and edit line 167 to send a custom Message!**

Lab 1 - Manchester Encoder

Use CLC4



- Make sure POT 1 is turned all the way clockwise!
- "S3" sends your message, not "S2"
- Save as "lab1.inc" in: C:\MASTERs\1610\Manchester Encoder.X\src

Numerically Controlled Oscillator (NCO)

Numerically Controlled Oscillator

What does it do?

Provides an adjustable output frequency pulse based on an input clock and increment value that with true linear frequency control and increased frequency resolution over standard PWM.

NCO Features

- **Two modes of operation**
  - Fixed 50% duty cycle (FDC)
    - **500 KHz max output**
  - Pulse Frequency Modulation (PFM)
    - **1MHz max output**

Numerically Controlled Oscillator

Pulse Frequency Mode

fmax = 1MHz

$$f_{NCO} = \left(\frac{f_{CLK}}{\text{Accumulator}} \right) \left(\text{Increment value} \right)$$

(Constant)

Fixed Duty Cycle Mode

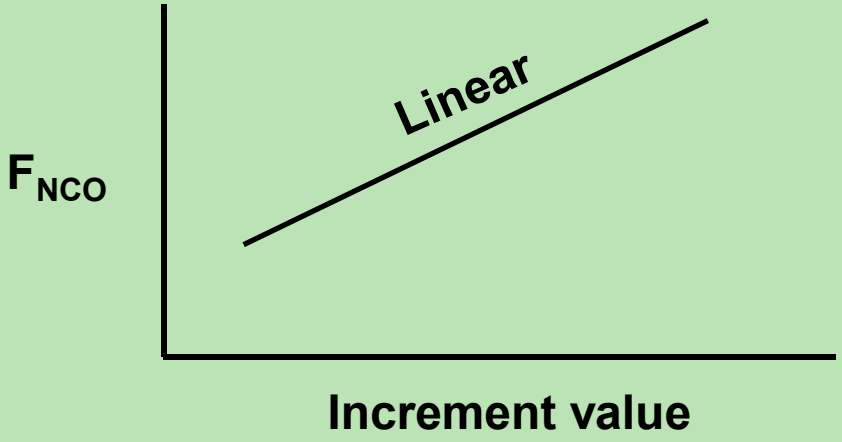
fmax = 500KHz

$$f_{NCO} = \left(\frac{f_{CLK}}{2*\text{Accumulator}} \right) \left(\text{Increment value} \right)$$

(Constant)

Accumulator = 20-bit
Increment value = 16-bit

Numerically Controlled Oscillator

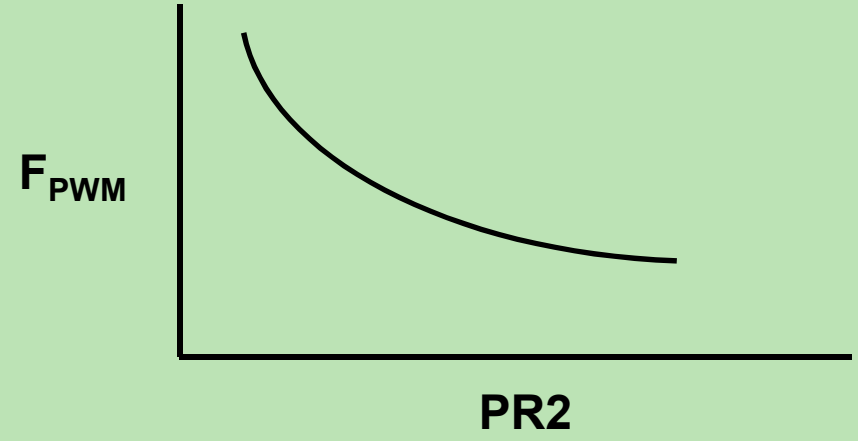


Increment value = 2^{16} = 0 → 65,535

$$F_{NCO} = \left(\frac{F_{OSC}}{\text{Accumulator}} \right) \left(\text{Increment value} \right)$$

Increment value ↑, F_{NCO} ↑

Linear change in NCO Frequency.



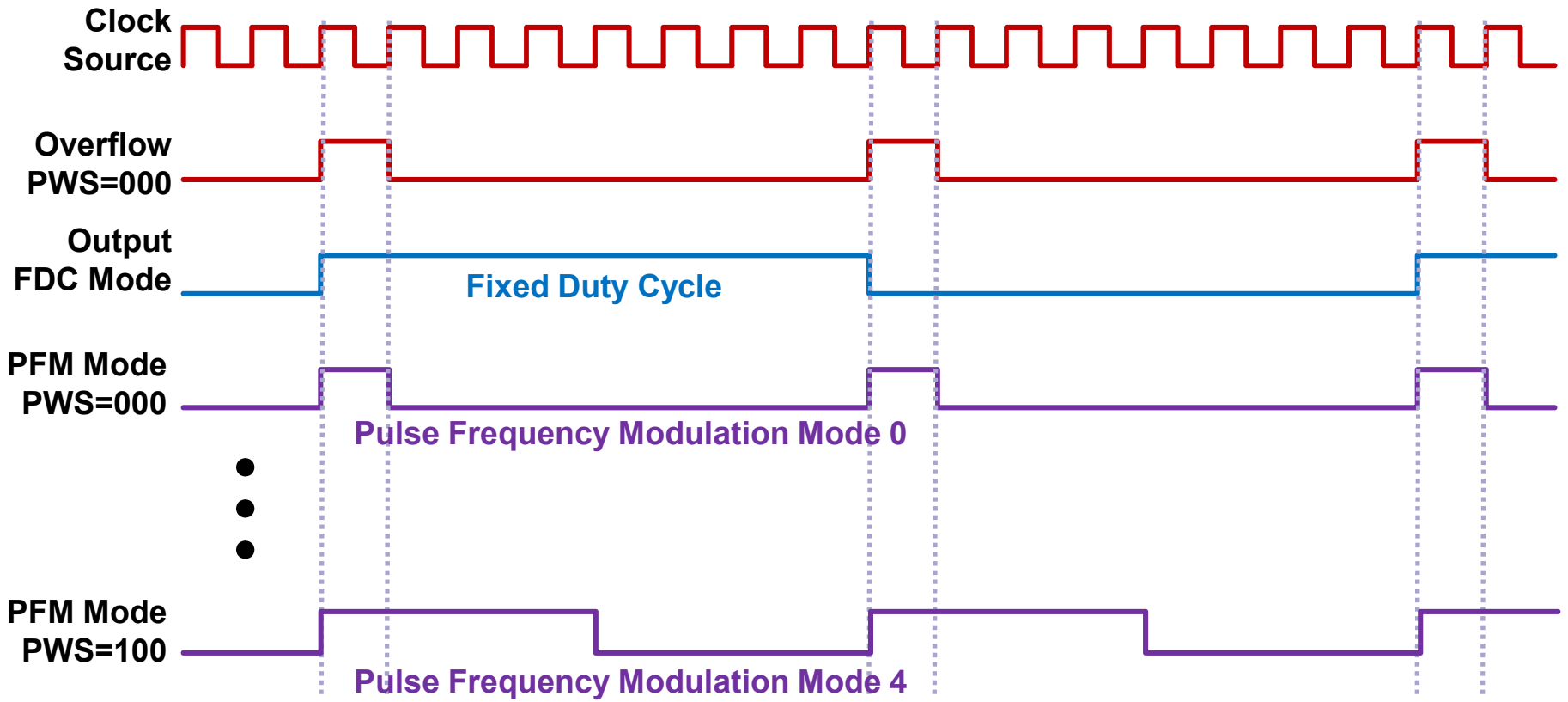
PR2 = 2^{8} = 0 → 255

$$F_{PWM} = \left(\frac{F_{OSC}}{4 \left(PR2 + 1 \right)} \right)$$

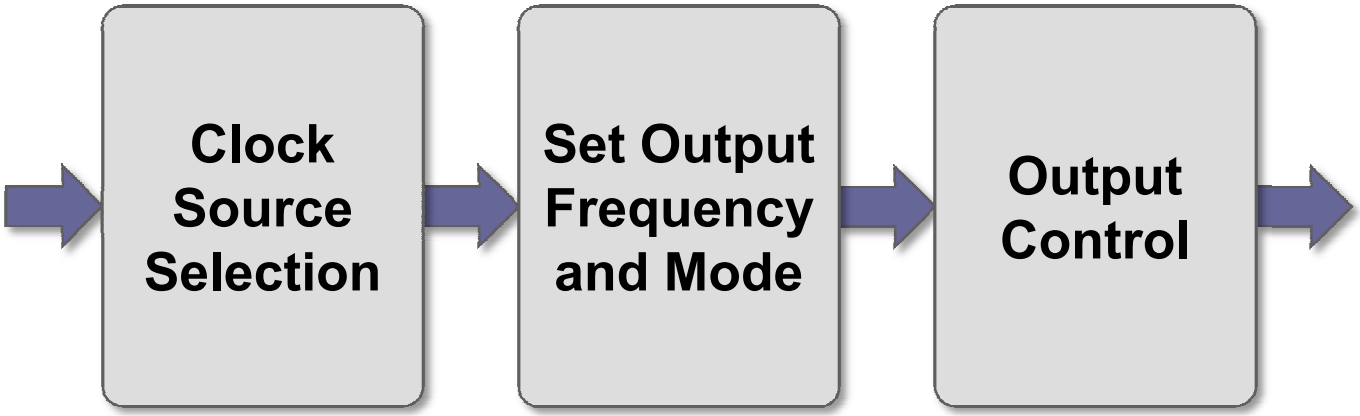
PR2 ↑, Fosc ↓

Nonlinear change in PWM Frequency.

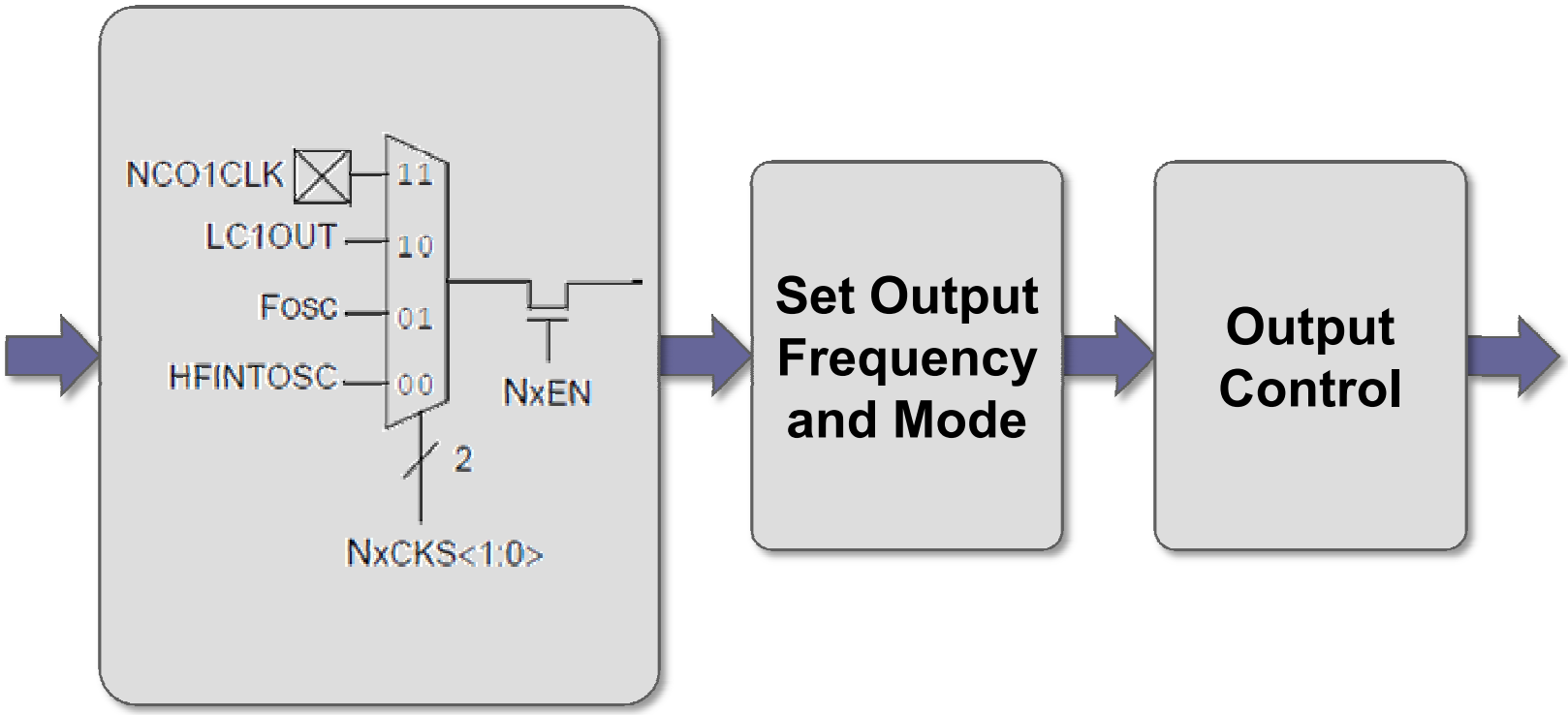
Numerically Controlled Oscillator



Numerically Controlled Oscillator

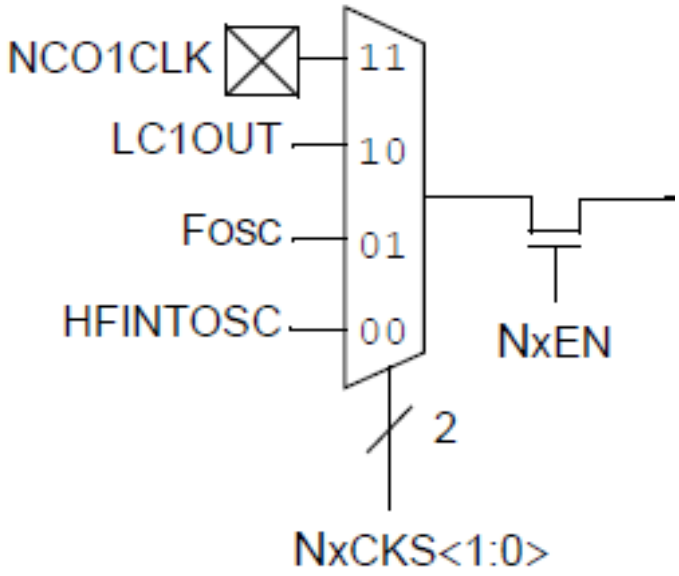


Numerically Controlled Oscillator



Clock Source Selection

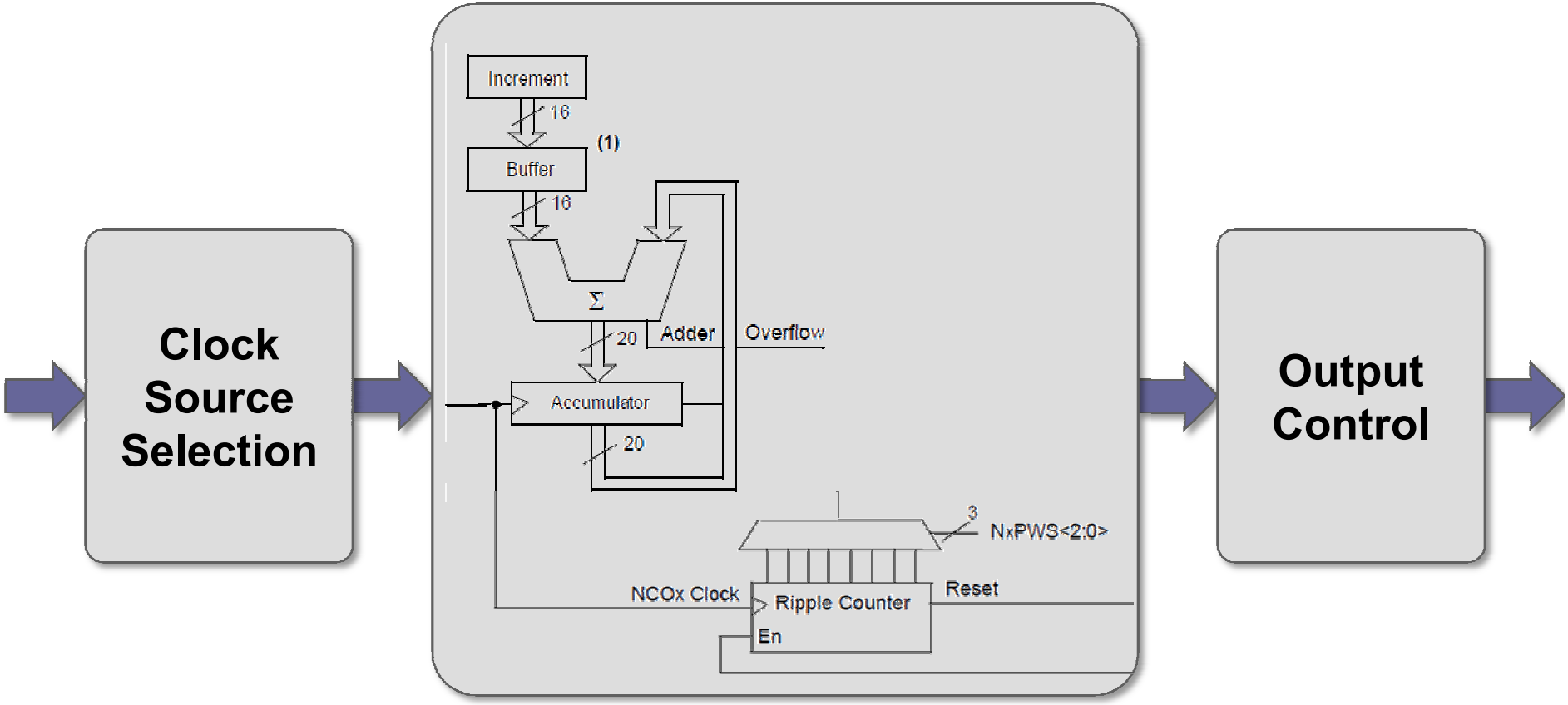
Clock Source Selection



NCOxCLK Register

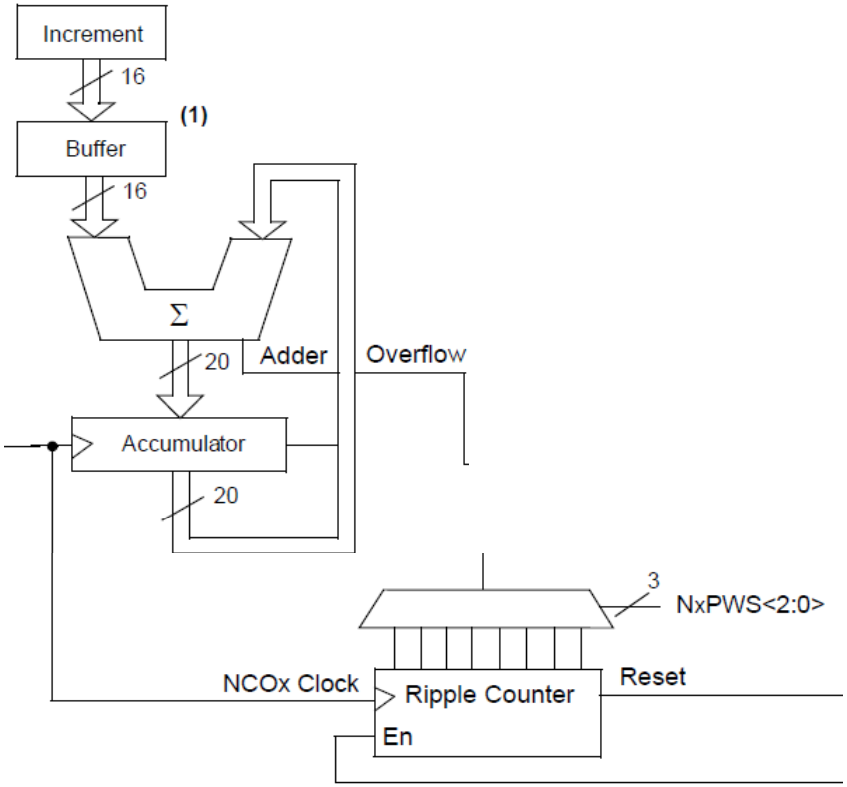
NxPWS<2:0>	-	-	-	NxCKS<1:0>

Numerically Controlled Oscillator



Set Output Frequency and Mode

Mode & Frequency



NCOxCLK Register

NxPWS<2:0>	-	-	-	NxCKS<1:0>

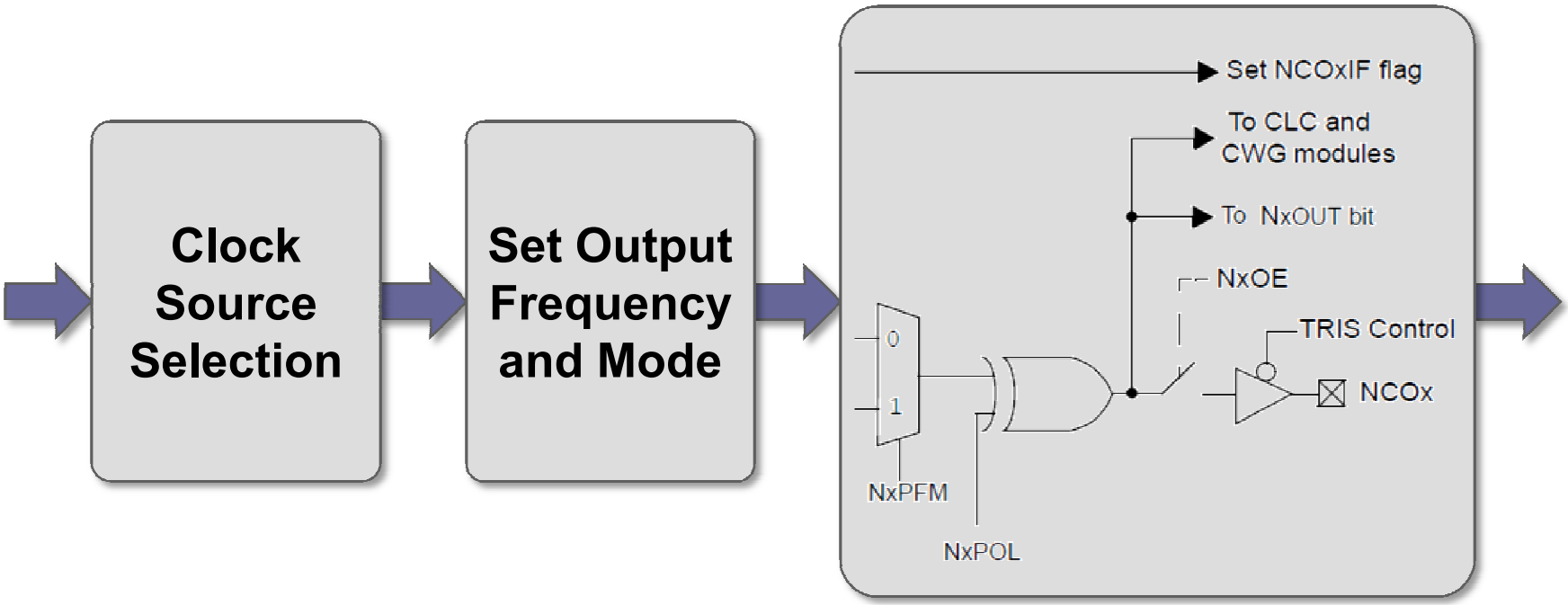
NCOxINCH Register

Increment <15:8>

NCOxINCL Register

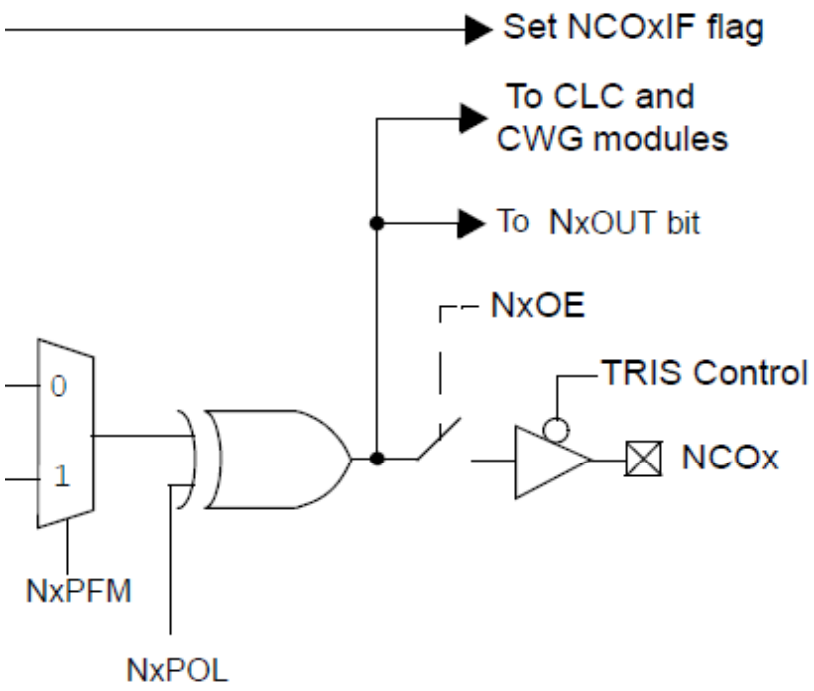
Increment <7:0>

Numerically Controlled Oscillator



Output Control

Output Control



NCOxCON Register

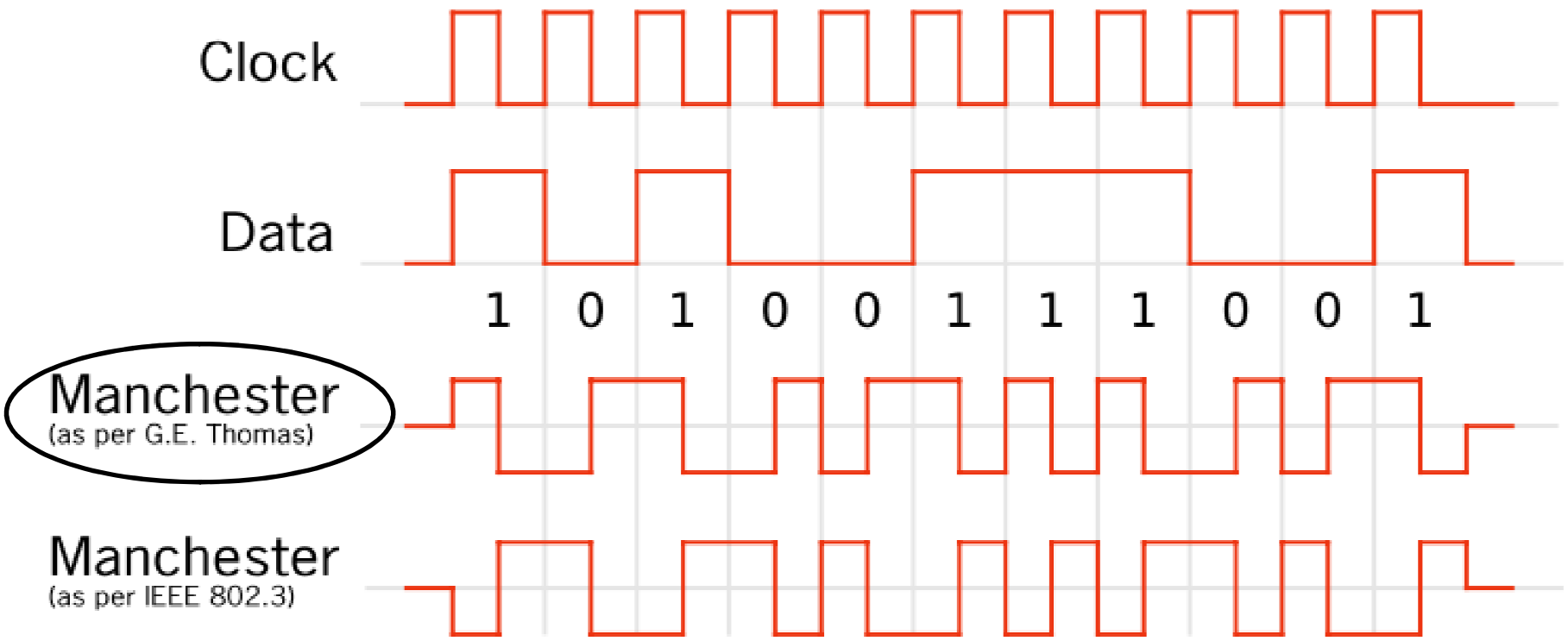
NxEN	NxOE	NxOUT	NxPOL	-	-	-	NxPFM

NCO Controlled PWM
How does it work?

- The NCO can produce some pulses that are 1 clock period smaller than the regular pulse.
- By this averaging (dithering/spreading), the NCO period increments achievable are much smaller than one CPU clock.
- If we pulse 3 clocks wide, but every second pulse is one system clock shorter the average pulse width would be exactly 2½ T_{SYS}, an increment of less than 1 clock period
- By using this period for timing the PWM pulse width we can adjust it by increments effectively much smaller than the CPU clock (on Average)

Lab 2
Manchester Decoder

Lab 2 - Manchester Decoder



Lab 2 - Manchester Decoder

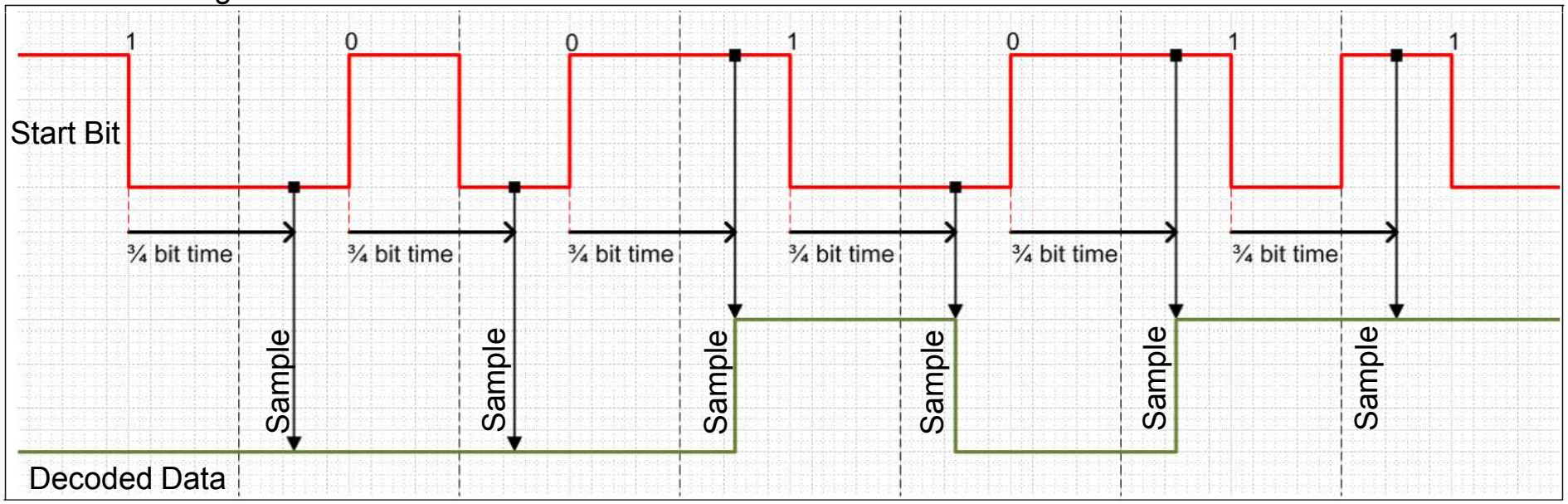
NCO Timing

If we synchronize our signal on the middle transition then we must use NCO in pulse frequency mode to generate an overflow at ¾ bit time to best capture the correct data.

$$F_{overflow} = \frac{NCO\ Clock\ Frequency\ *\ Increment\ Value}{2^n}$$

n = accumulator width in bits

Manchester Signal



Lab 2 - Manchester Decoder

NCO Timing

The Manchester signal is running at 25 kHz, which gives us a 40 us cycle time

¾ of that is 30 us, which is 33.33 kHz.

Plugging it in to our equation gives us:

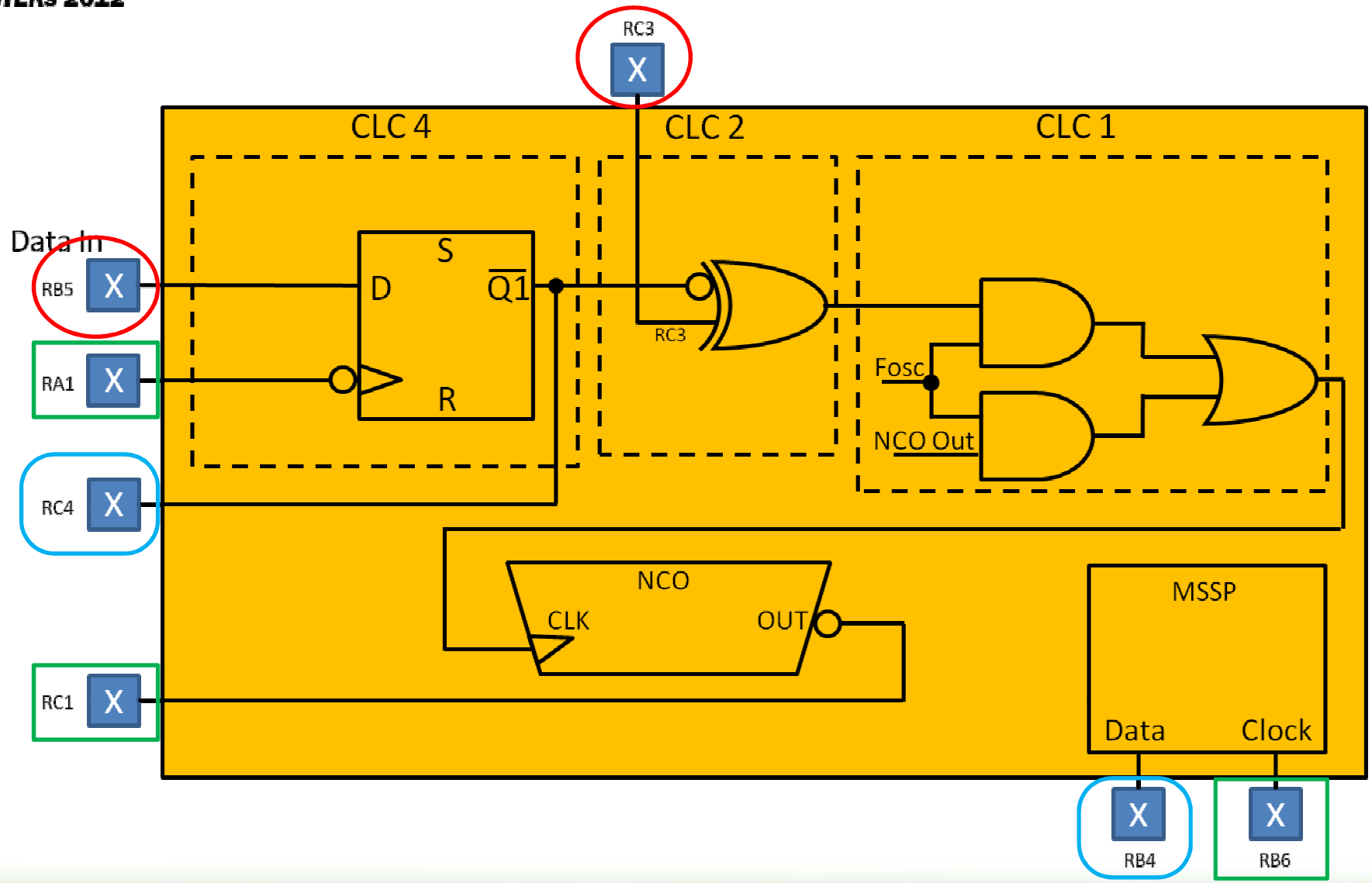
$$F_{overflow} = \frac{NCO\ Clock\ Frequency\ *\ Increment\ Value}{2^n} = \frac{16\ MHz(Internal\ Clock\ F_{osc})\ *\ Increment\ Value}{2^{20}}$$

n = accumulator width in bits

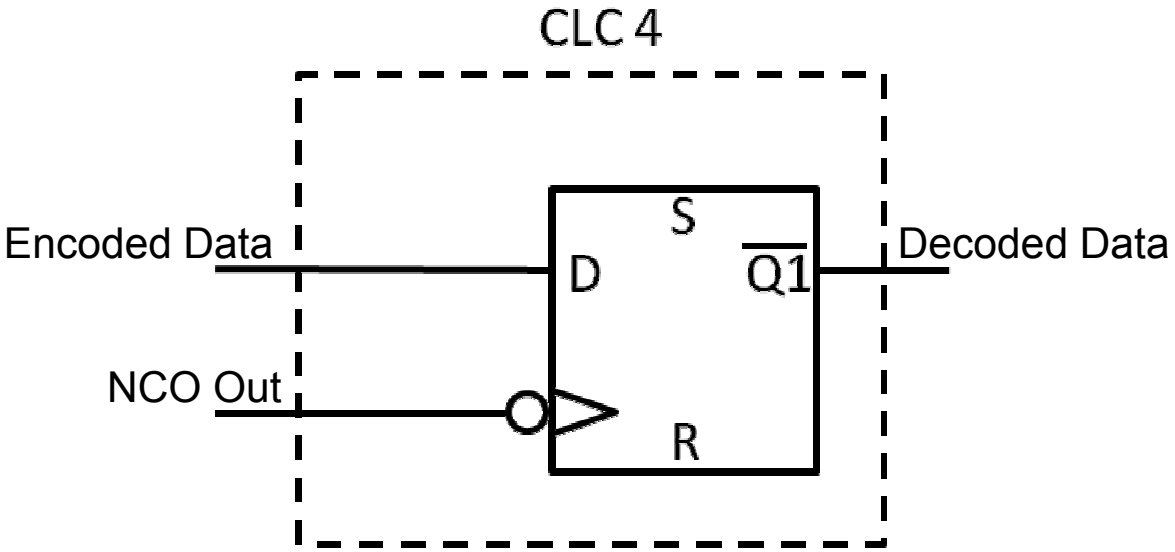
$$30us\ pulse\ =\ 33.33\ kHz\ =\ \frac{16\ MHz\ *\ Increment\ Value}{2^{20}} \Rightarrow Increment\ Value\ =\ 2184\ =\ 0x0888$$

NCO Increment = 0x0888

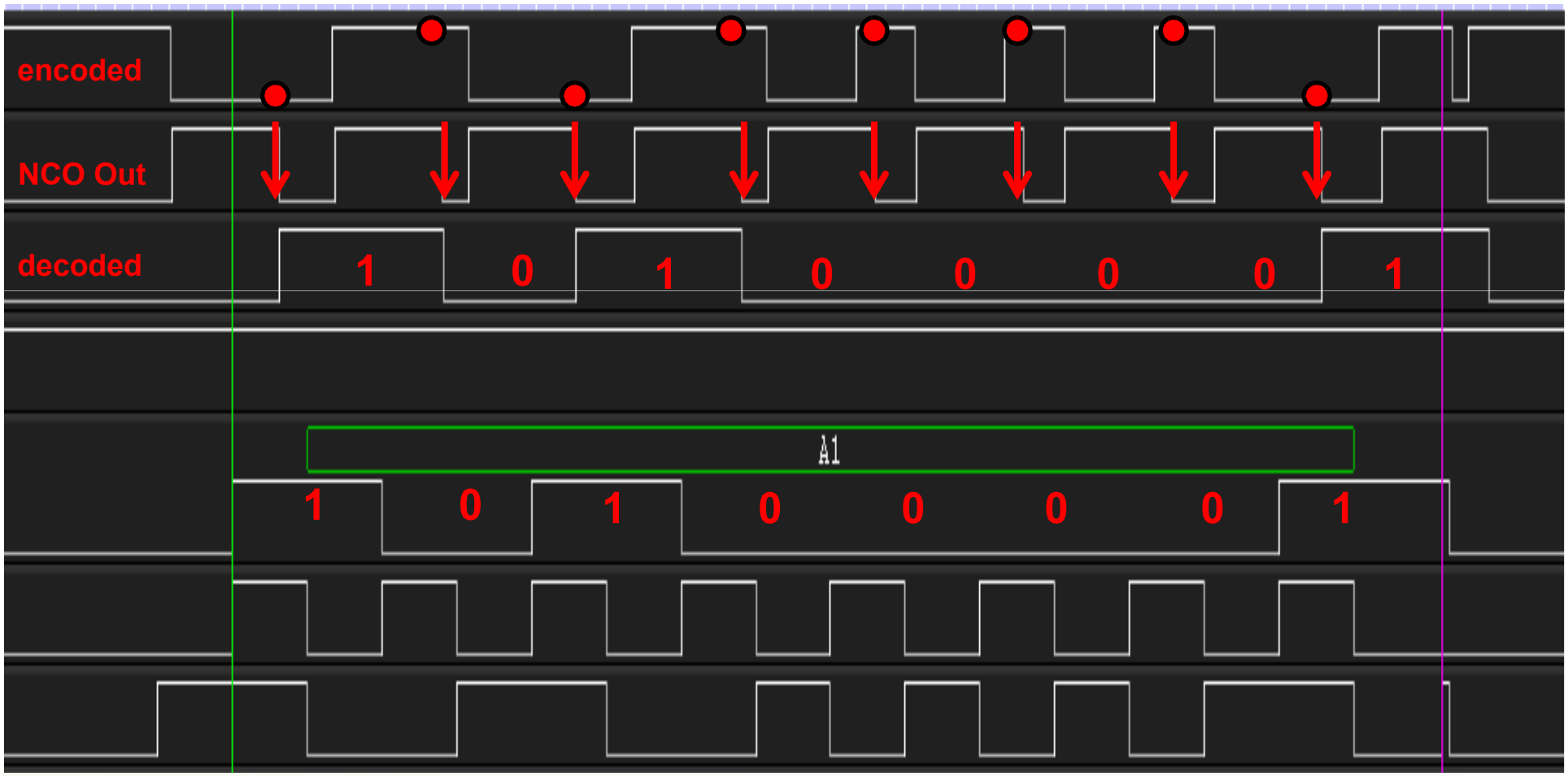
Lab 2 - Manchester Decoder



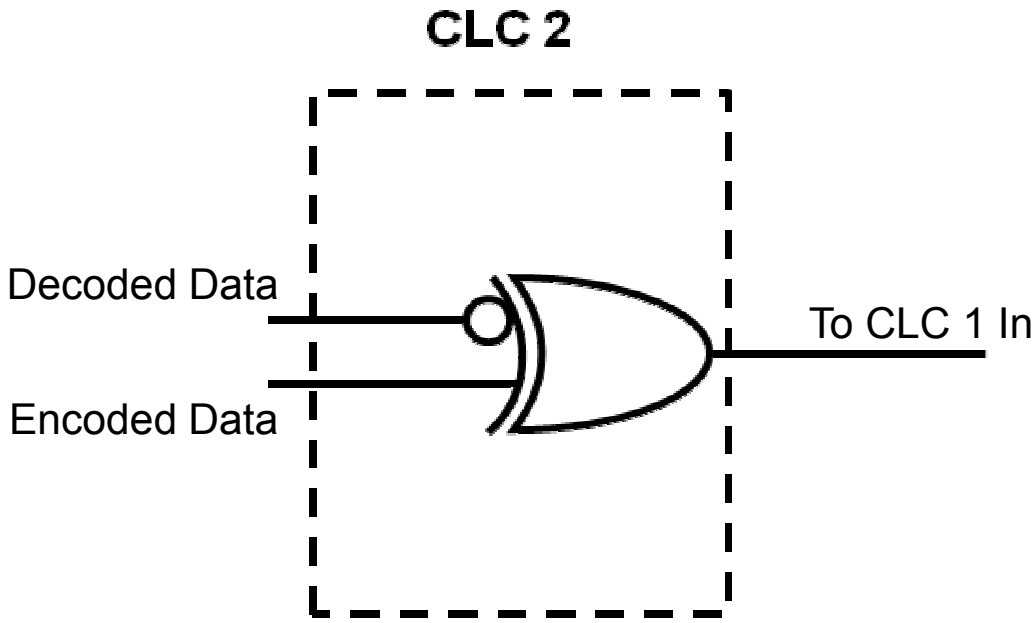
Lab 2 - Manchester Decoder



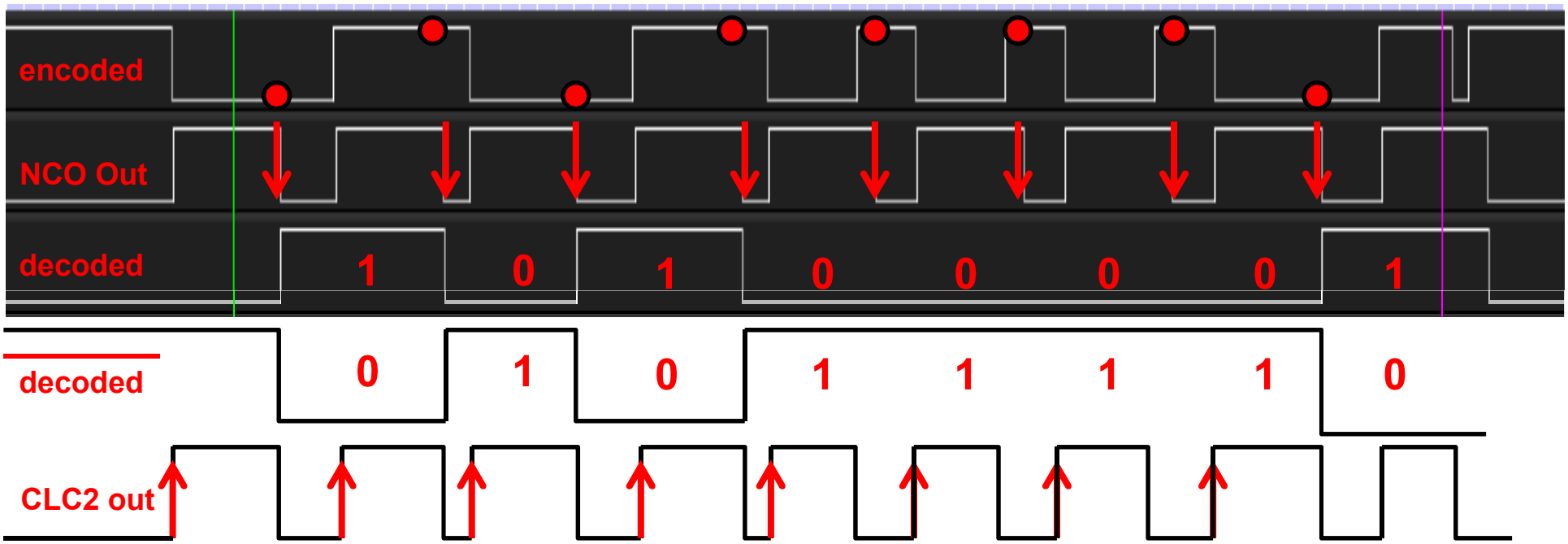
This stage captures the incoming encoded data and samples it every ¾ bit time to produce the decoded data



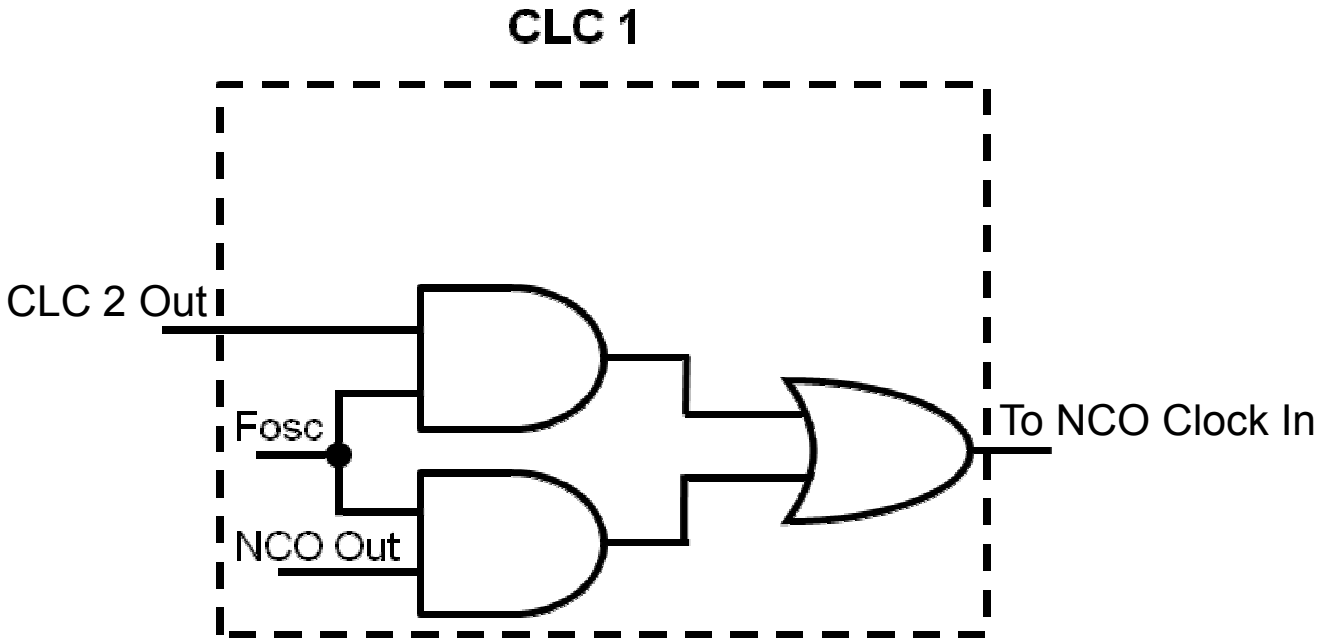
Lab 2 - Manchester Decoder



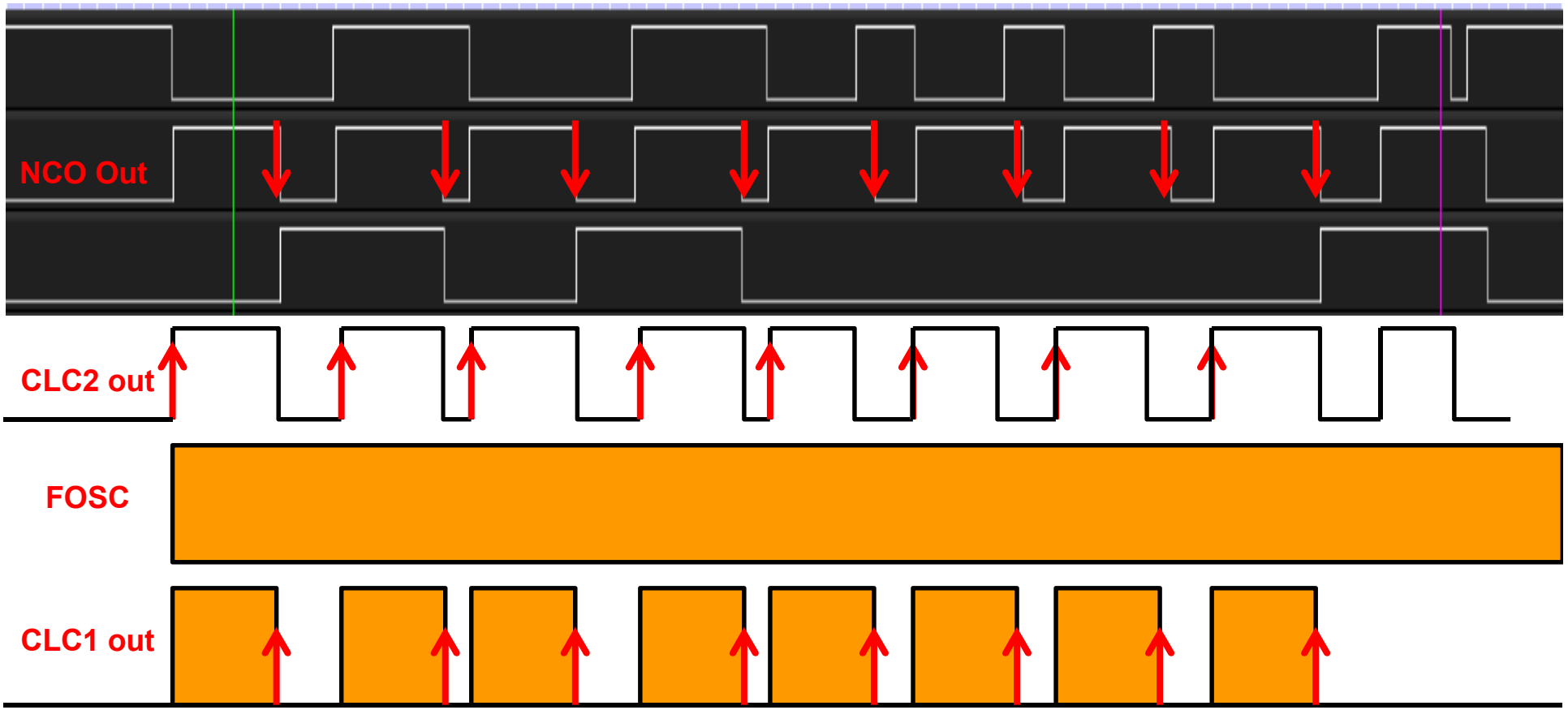
This stage ensures that every encoded mid-bit transition produces a rising edge for the next stage (Synchronization)



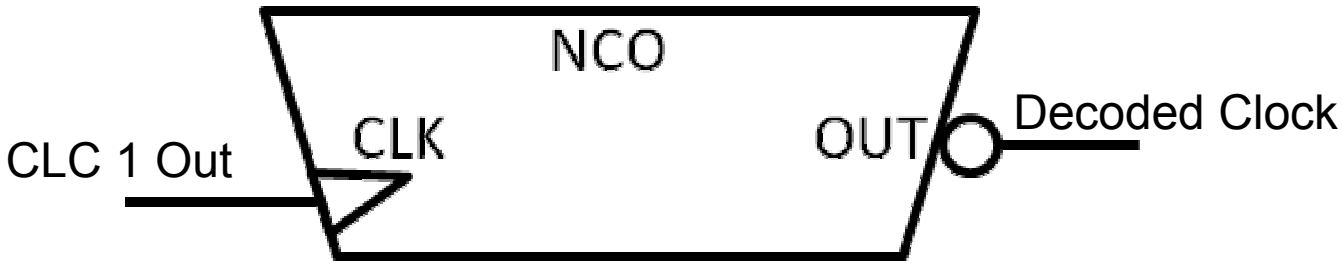
Lab 2 - Manchester Decoder



This stage takes the previous rising edge and starts clocking the NCO. It also ensures that the NCO continues clocking until it overflows if the input to this stage goes to zero

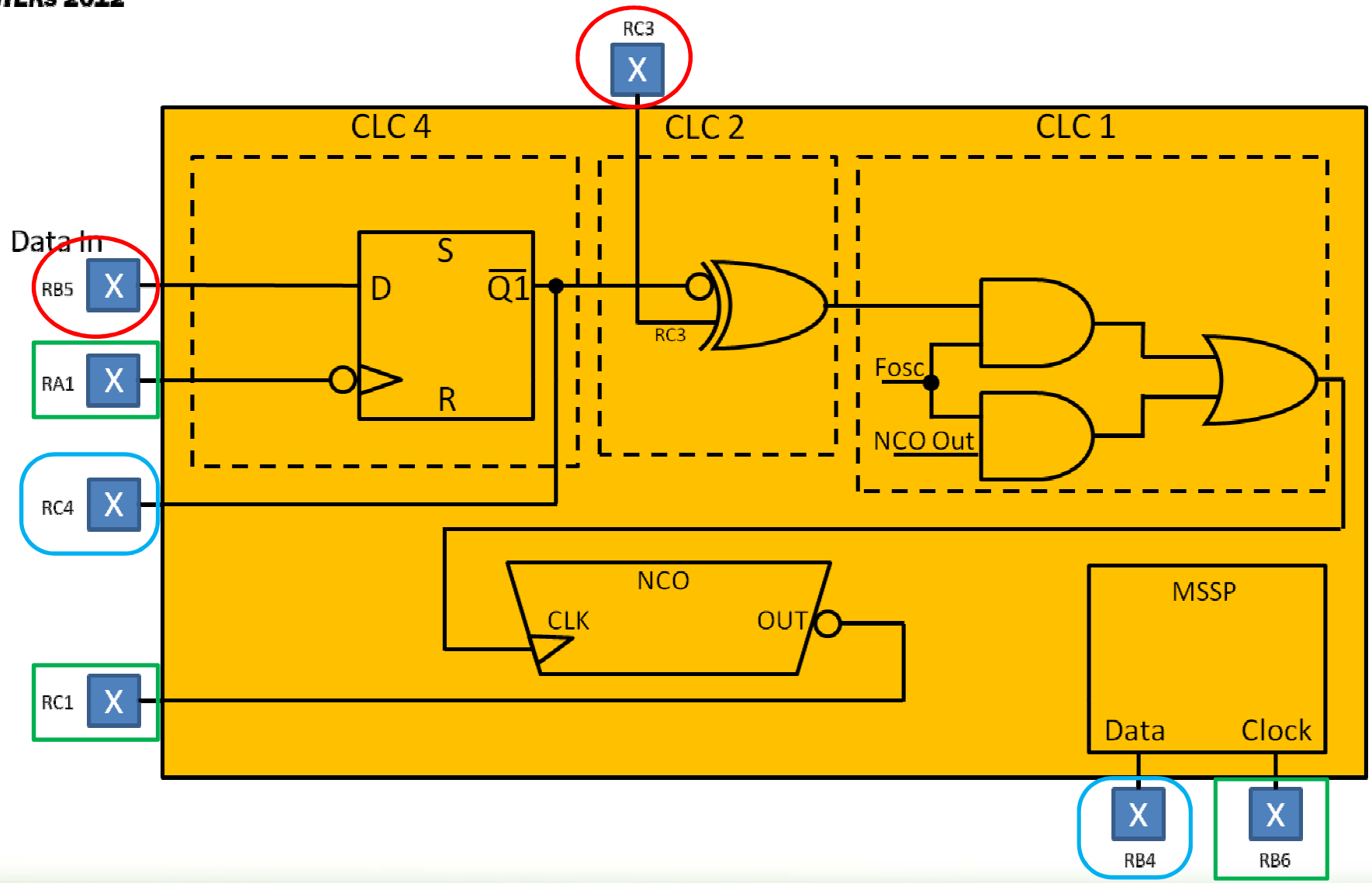


Lab 2 - Manchester Decoder



This stage generates the ¾ bit time to create the clock. Combined with the previous stage, it also keeps the input clock present until the output goes low (overflow)

Lab 2 - Manchester Decoder

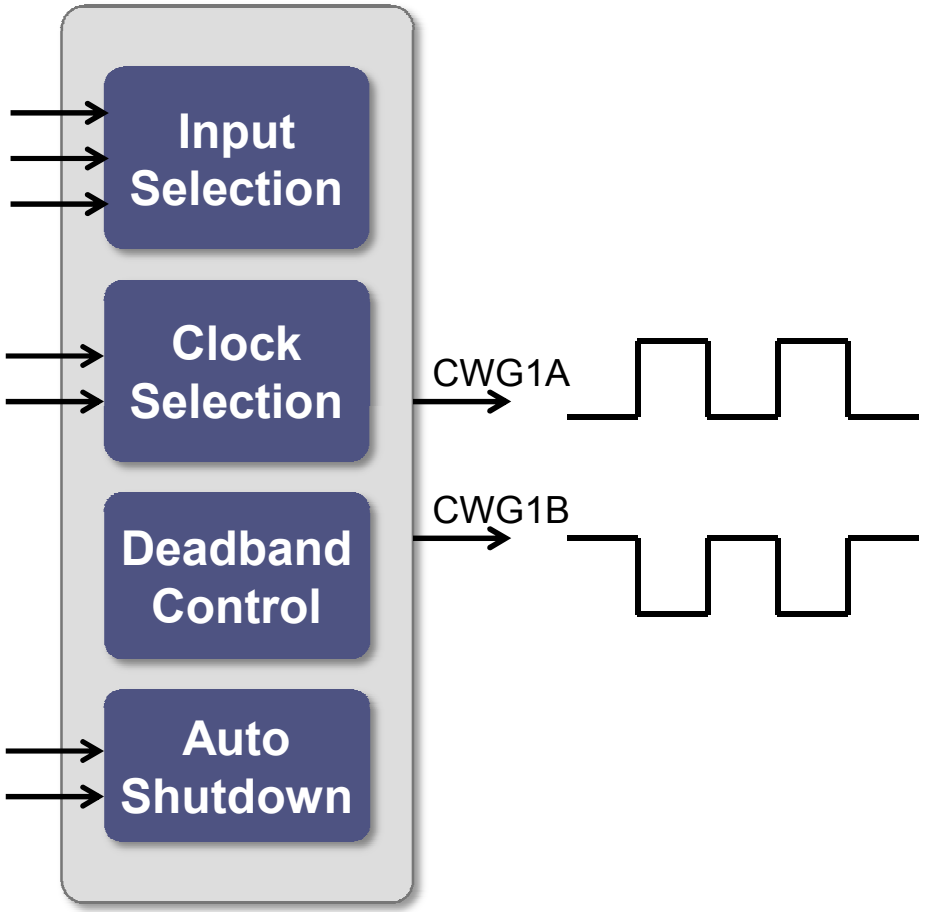


Complementary Waveform Generator (CWG)

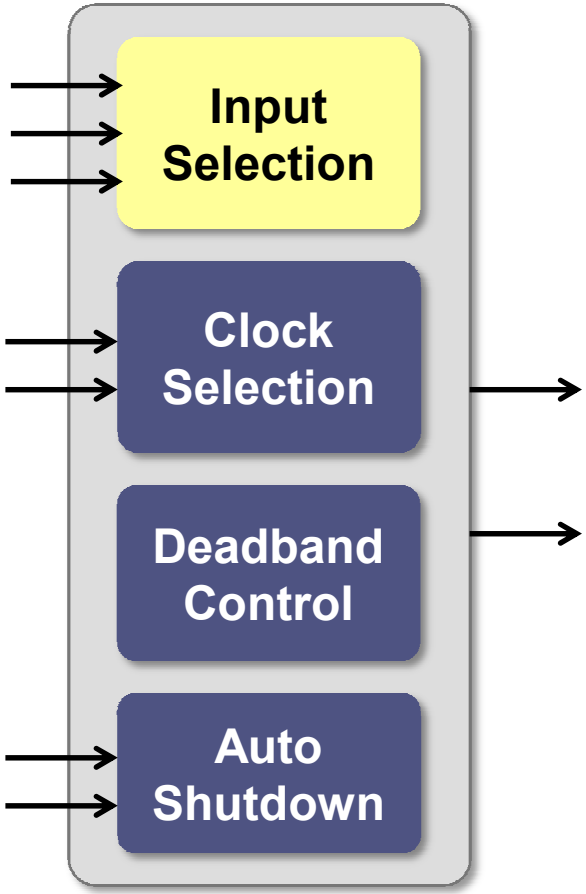
Complementary Waveform Generator

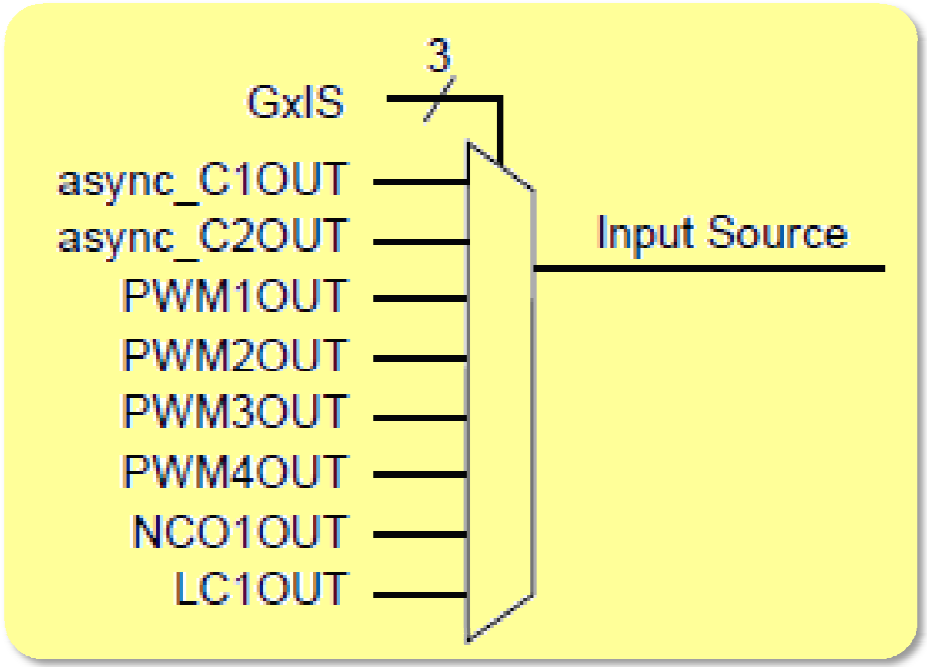
What does it do?

generates a complementary waveform with dead-band delay and auto-shutdown capabilities from a selection of input sources.



Input Selection

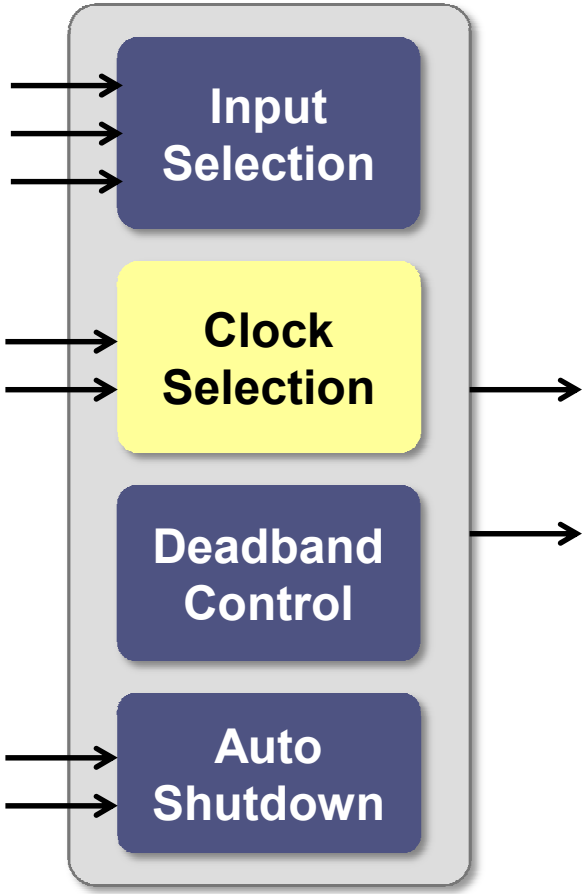


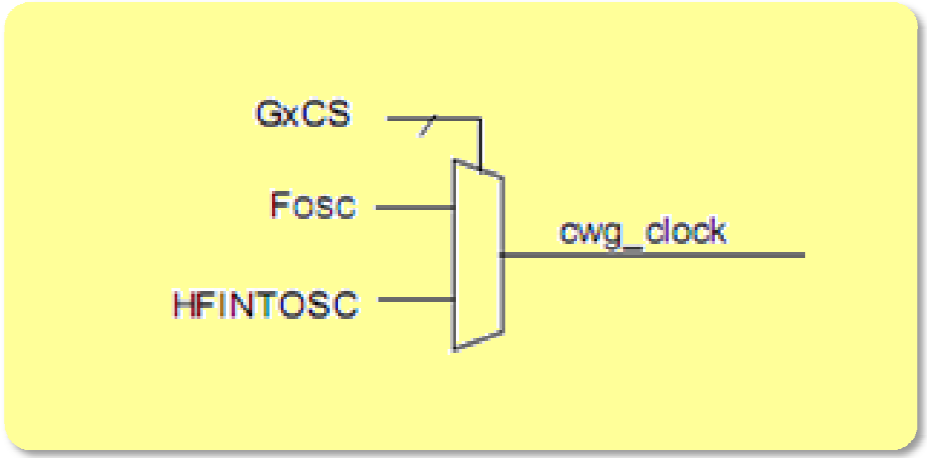


CWGxCON1 Register

GxASDLB<1:0>	GxADSLA<1:0>	-	GxIS2	GxIS1	GxIS0

Clock Selection

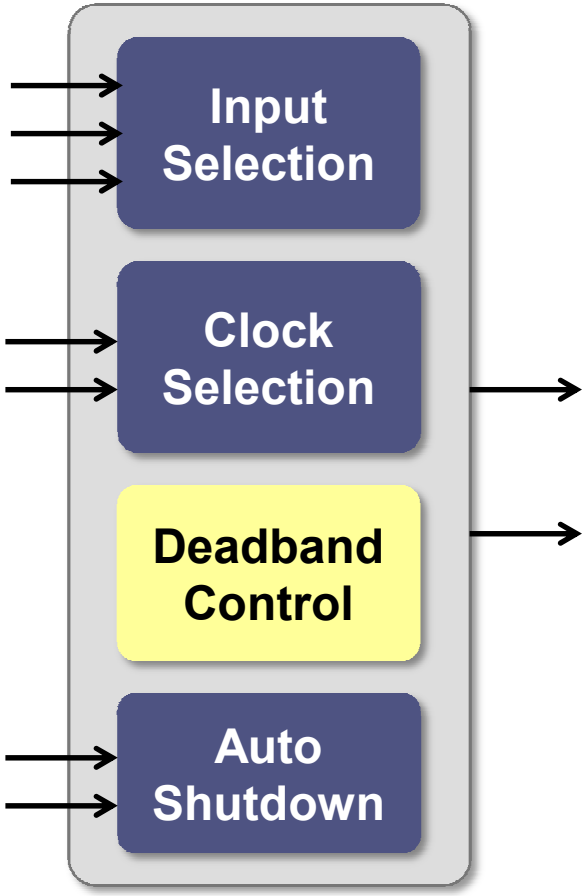




CWGxCON0 Register

GxEN	GxOEB	GxOEA	GxPOLB	GxPOLA	-	-	GxCS

Clock Selection



What does Dead-Band Control do?

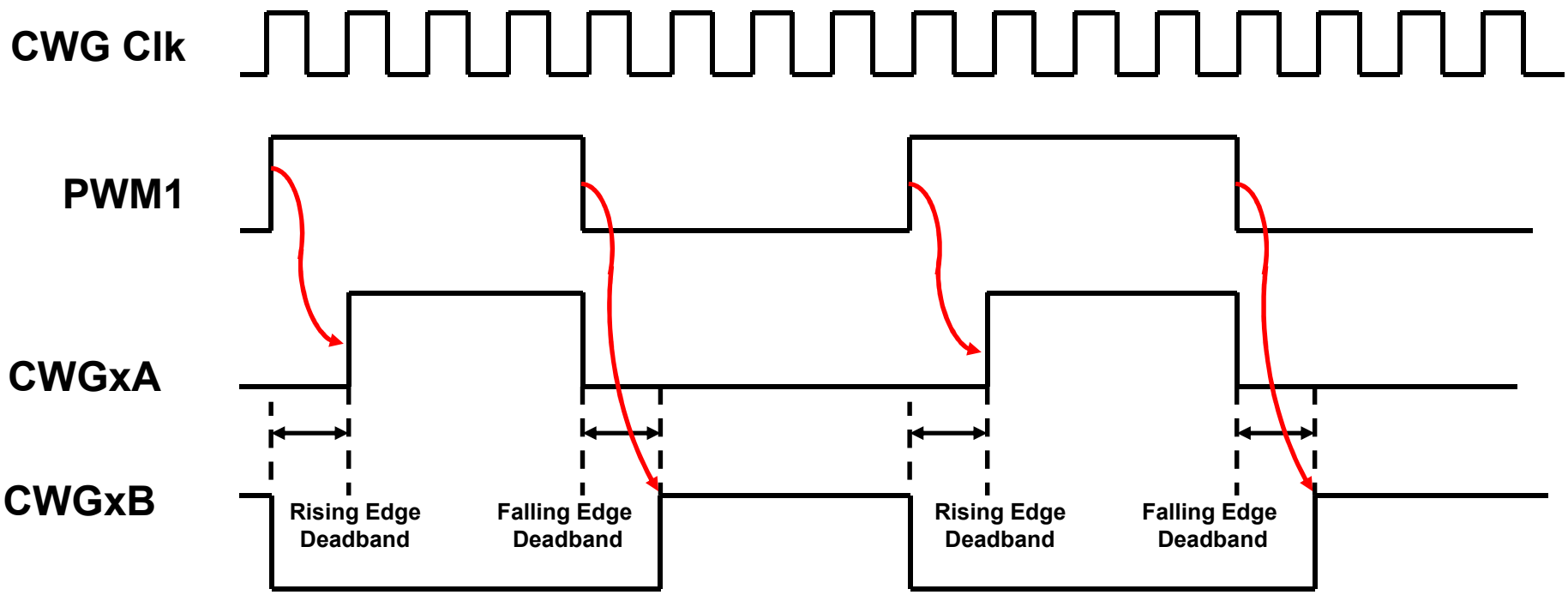
It provides for non-overlapping output signals to prevent shoot-through current in power switches.

- Two 6-bit counters (64 counts of delay)
- Configurable for rising or falling edge of input source

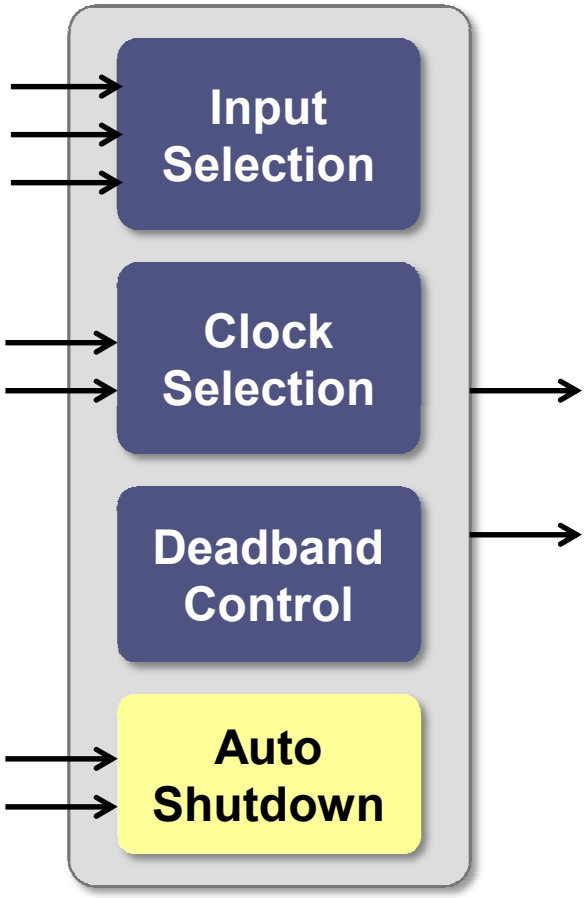
CWGxDBR and CWGxDBF Registers

-	-	CWGxDBR<5:0> or CWGxDBF<5:0>

Complementary Waveform Generator



Auto Shutdown



What does Auto-Shutdown Control do?

Forces an immediate override of the CWG outputs allowing a safe shutdown of external circuitry.

Complementary Waveform Generator

Selectable shutdown conditions:

- Software generated
- External input pin (CWG1FLT ► **GxASDFLT**)
- Output of CLC (LC1OUT ► **GxASDCLC2**)

The shutdown state can be cleared automatically or held until cleared by software.

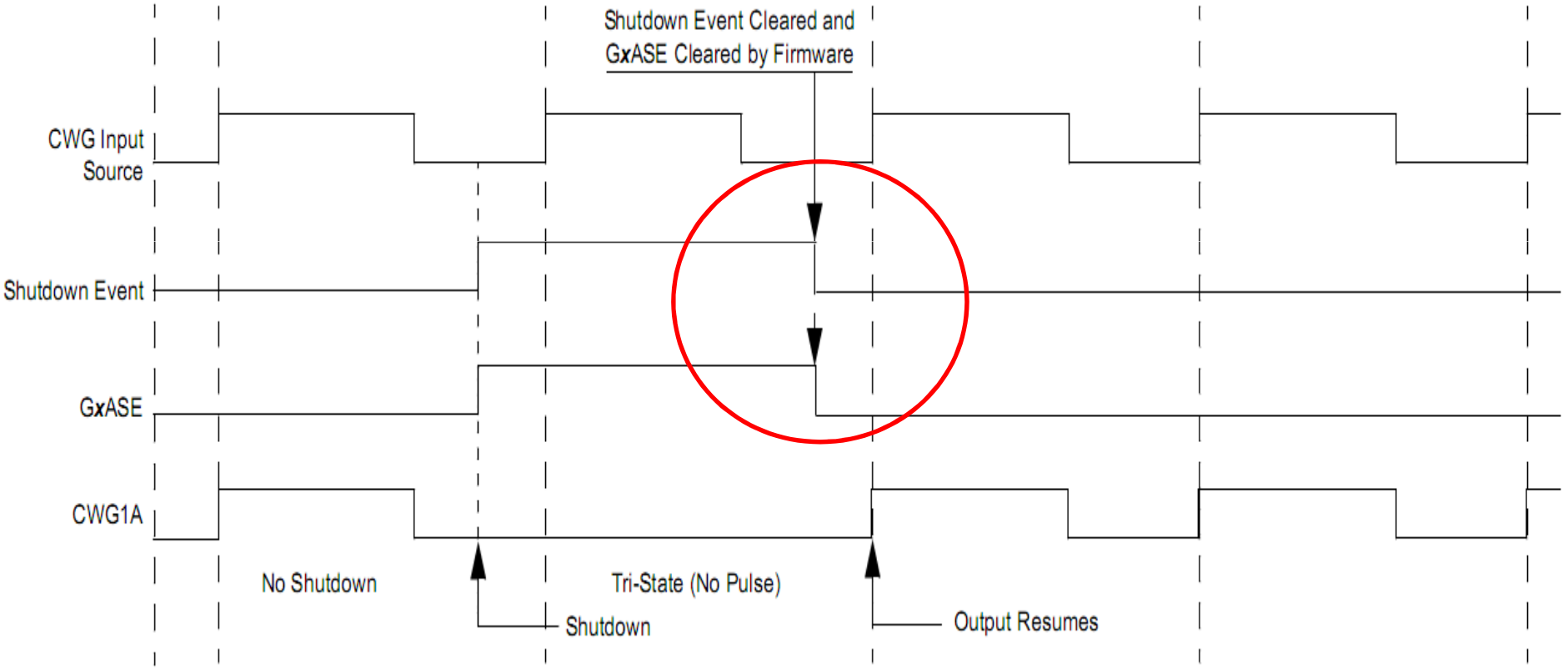
- **GxARSEN** Enables automatic hardware restart
- **GxASE** indicates if shutdown has occurred

CWGxCON2 Register

GxASE	GxARSEN	-	-	-	-	GxASDFLT	GxASDCLC2

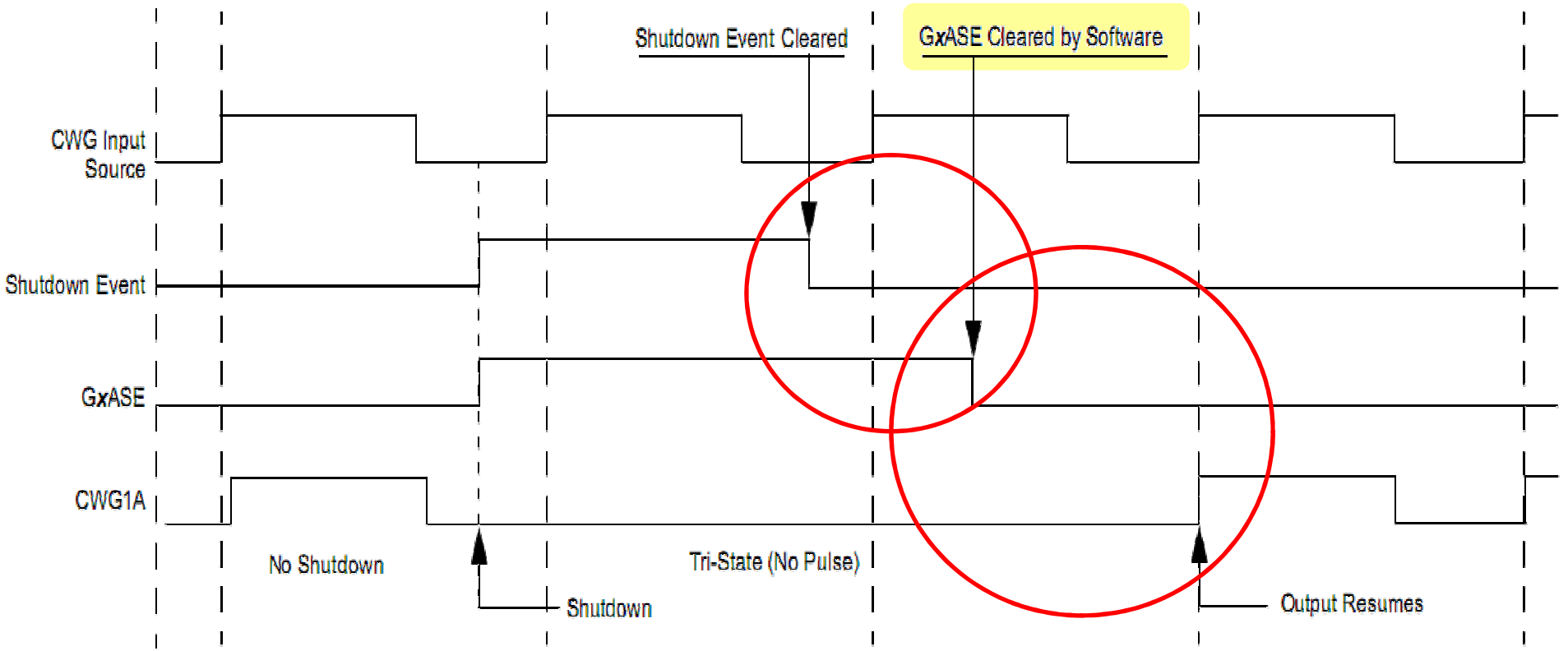
Complementary Waveform Generator

Auto-Shutdown using Auto-Restart



Complementary Waveform Generator

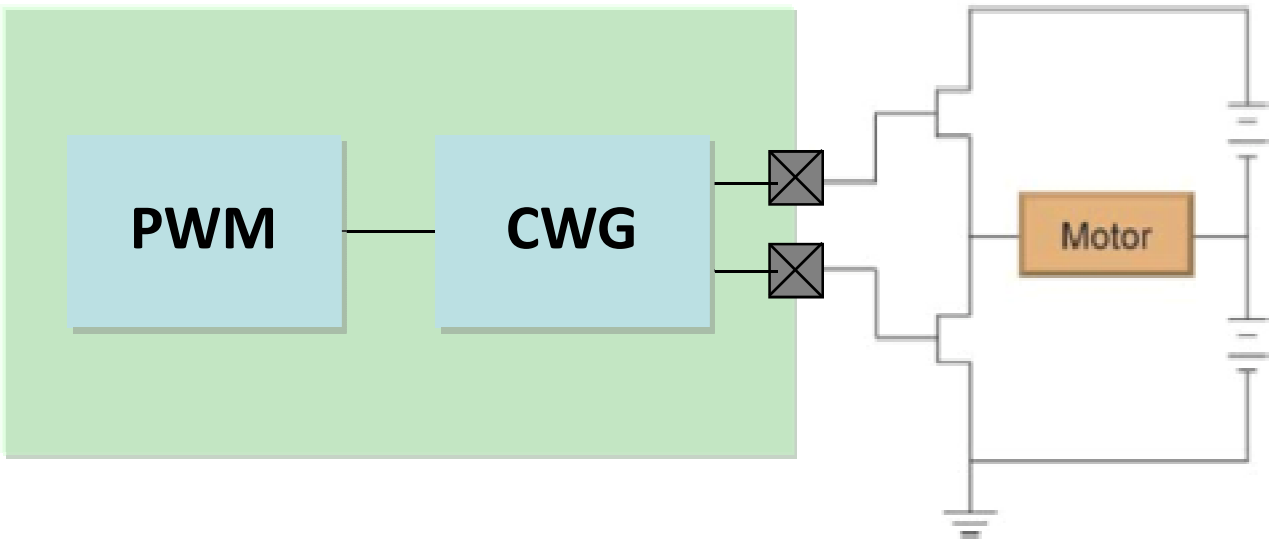
Auto-Shutdown without using Auto-Restart



CWG Applications

Example Applications

- Switch Mode Power Supplies
- LED / Fluorescent lighting
- Battery Charger
- Motor Drive
- Power Factor Correction
- Class D Audio Amplifiers



Ex: Half Bridge Circuit using CWG

Use CWG advanced features to provide dead-band control, auto shut down

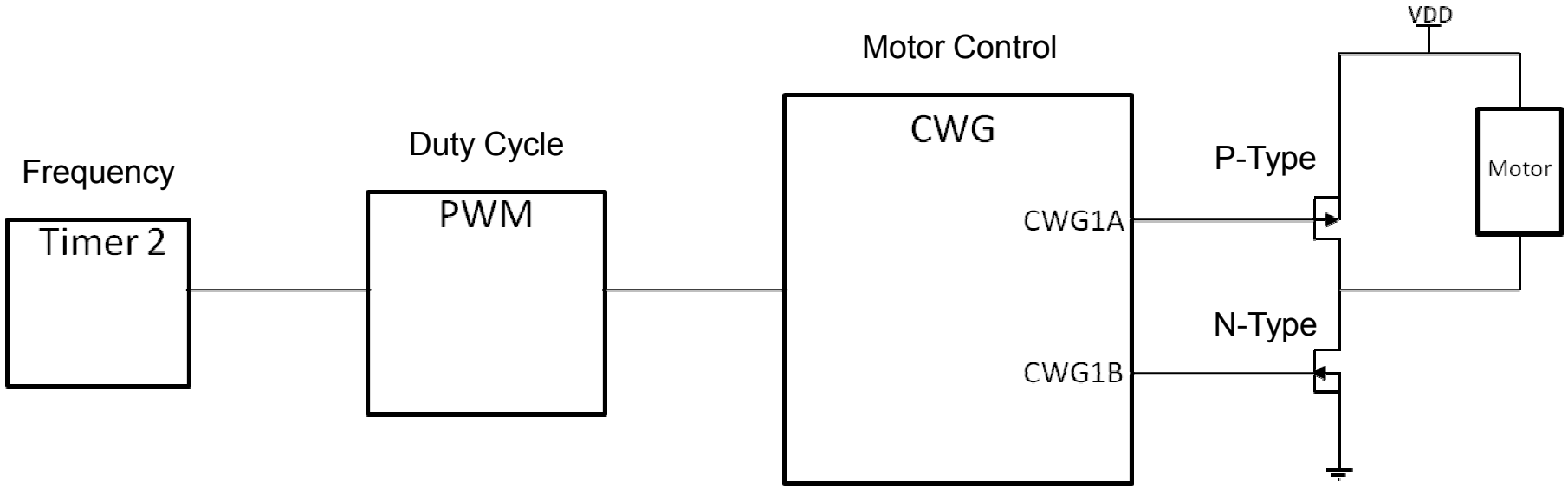
Lab 3
Driving a Motor with the CWG

Lab 3 – CWG Motor Control

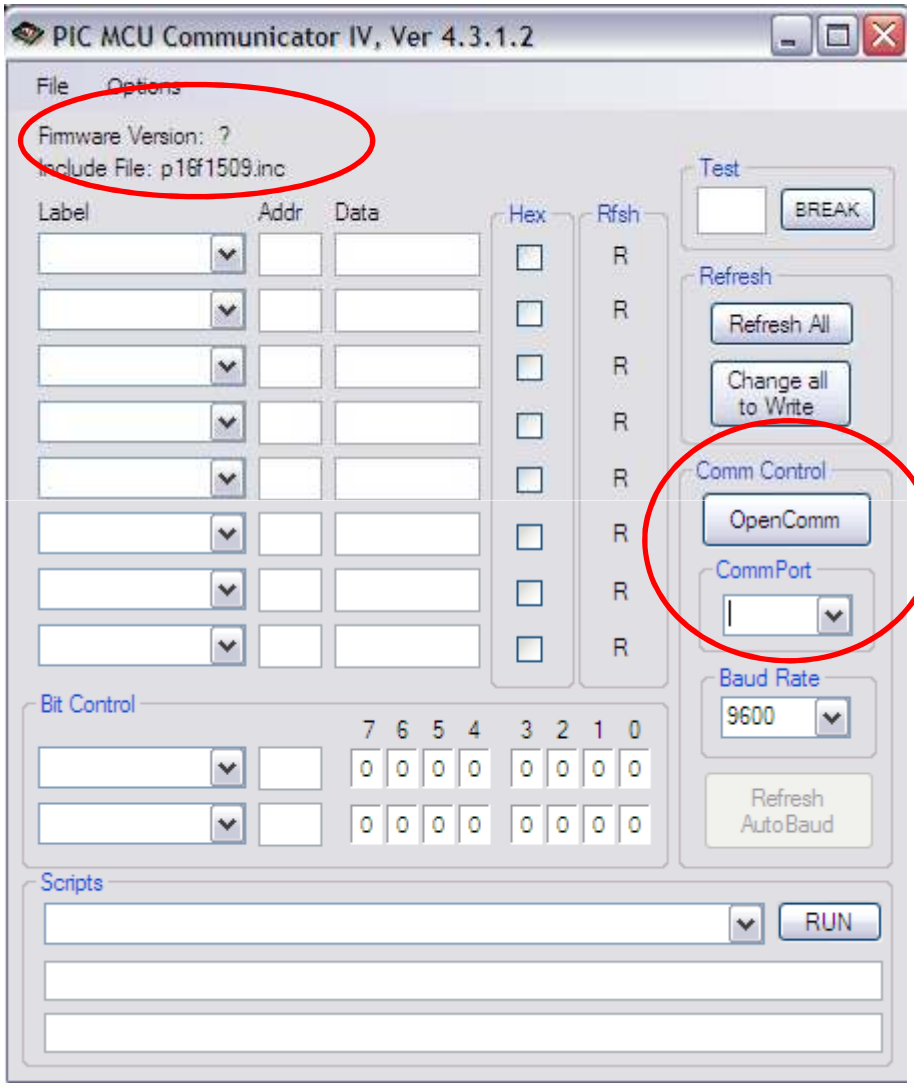
CWG Setup

Converts PWM signal into complementary form

P-type for efficiency



Lab 3 – CWG Motor Control



How to setup PIC communicator

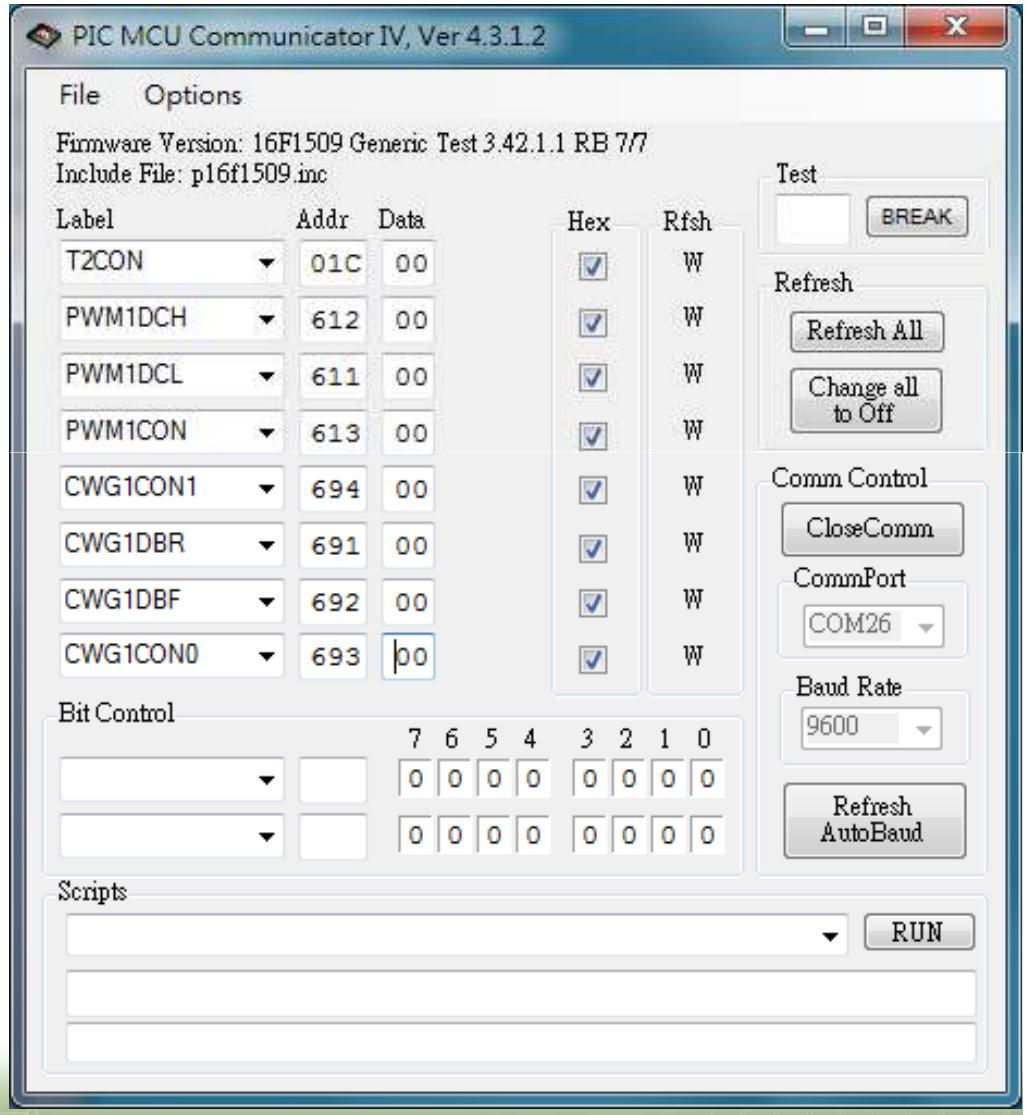
1. Close jumpers J3 and bottom one on J4 on daughter board
2. Open CLC designer tool
3. In lower left hand corner, press “OPEN” button
4. Connect USB cord to board
5. Select the COM Port (most likely COM8) or whichever is the greatest number
6. Click “Open comm”
7. If successful, the firmware version will no longer be a question mark “?”

Lab 3 – CWG Motor Control

PIC Communicator

The registers needed are (in order):
T2CON
PWM1DCH
PWM1DCL
PWM1CON
CWG1CON1
CWG1DBR
CWG1DBF
CWG1CON0

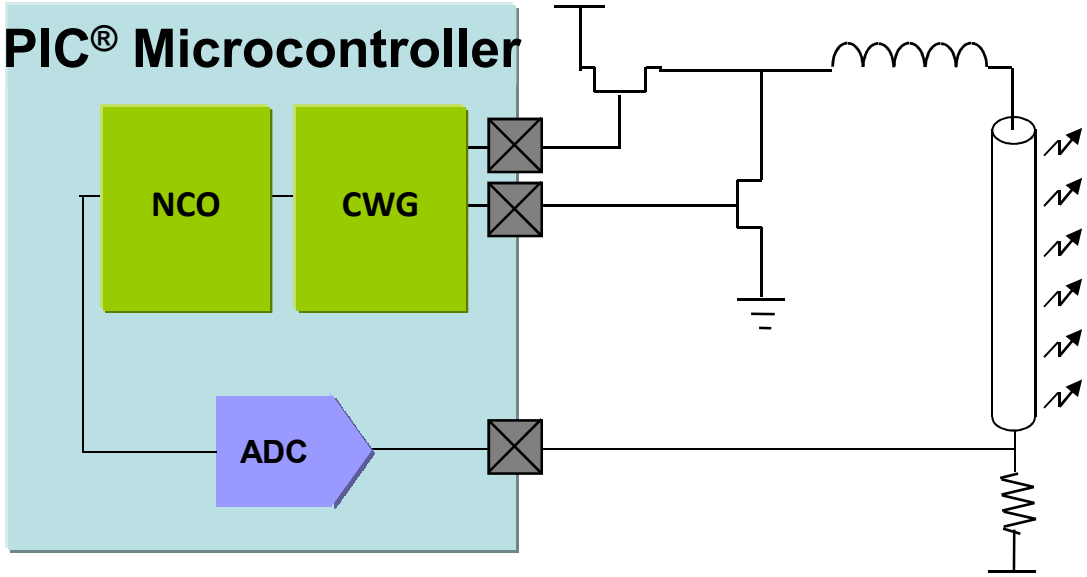
Note:
Remove all jumpers from J4 and J6 (daughter)
Close both horizontal jumpers on J5 (daughter)

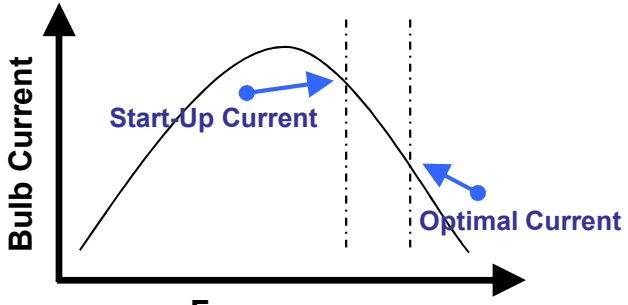


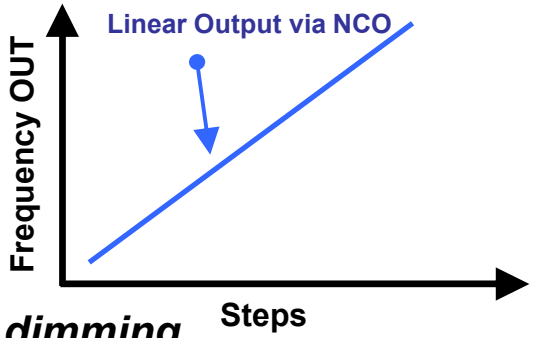
NCO and CWG
Applications

Example Applications

- Fluorescent & LED Lighting Control
- Neon Lighting Control
- Lighting Ballast
- Resonant Power Supplies
- Motor Drivers
- Radio Tuning Circuitry
- Class D Audio Amplifiers







Ex: Fluorescent Lighting Control

NCO used to create linear frequencies for start-up and dimming control for lower power and extended lifetime of the fluorescent bulb

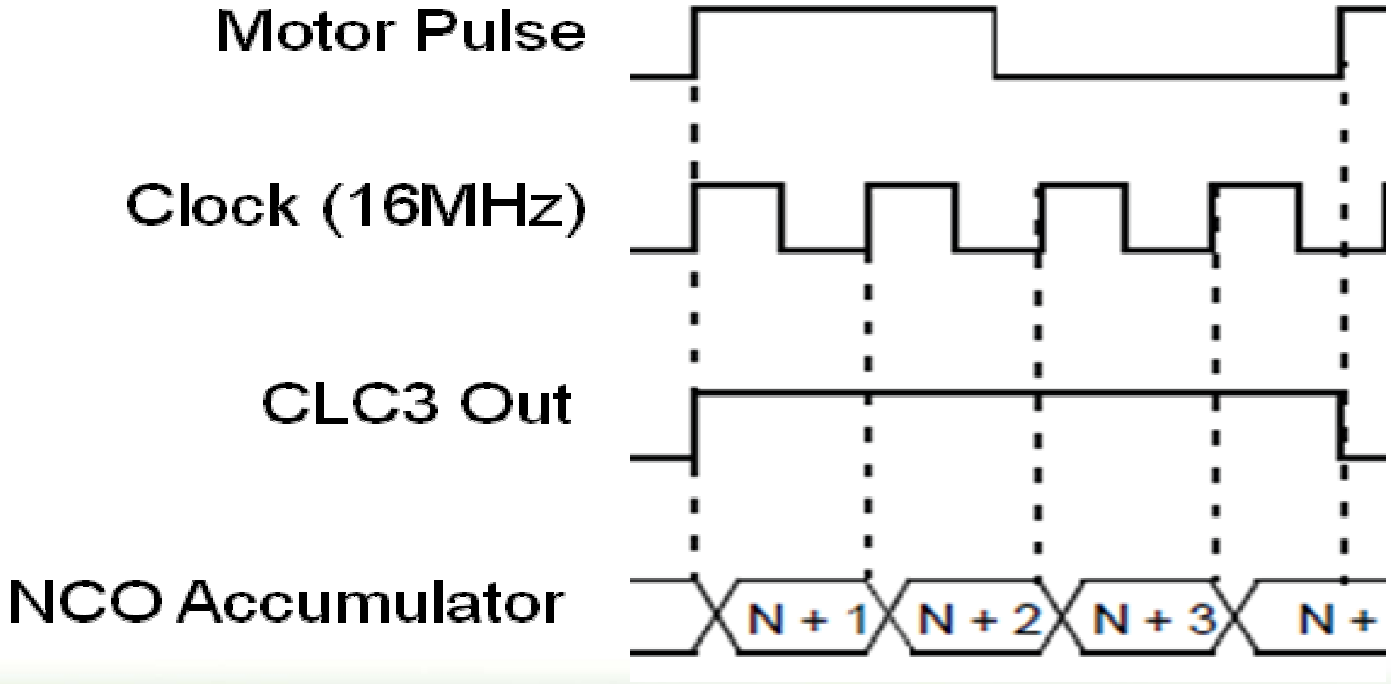
Lab 4
Sending Motor Speed Through Manchester Encoder

Lab 4 – Motor Speed

Motor Speed Measurement

Gives a small pulse every half revolution

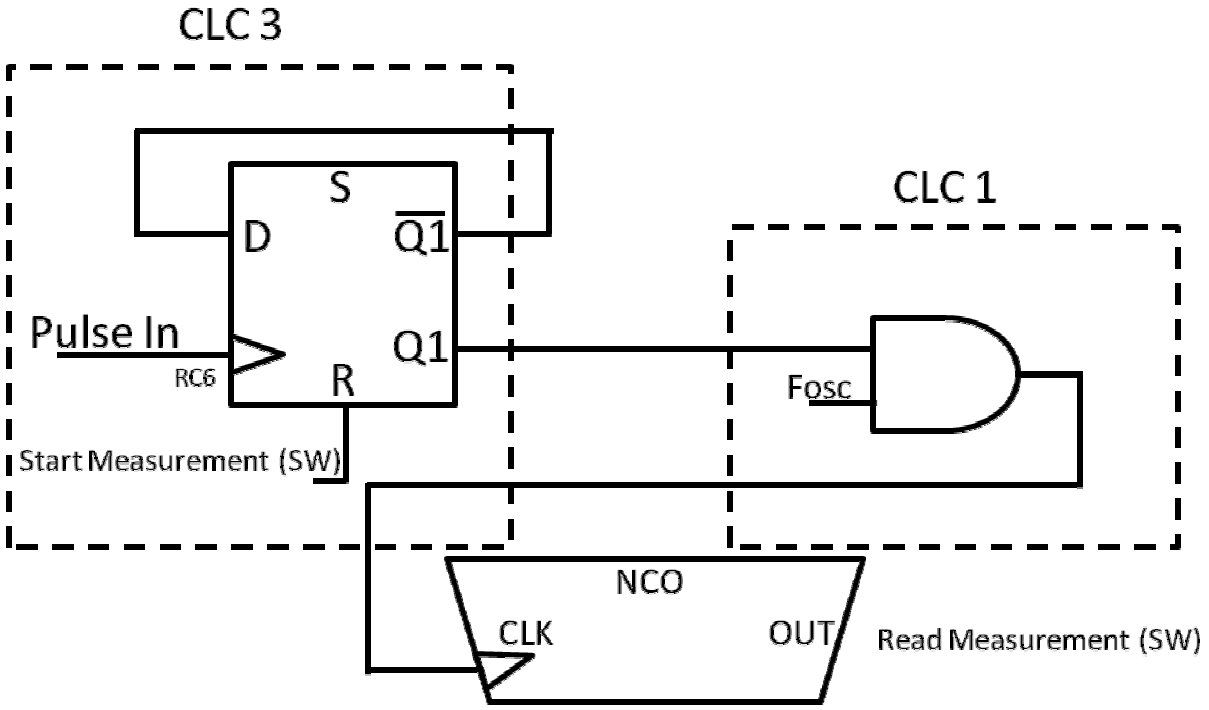
Design a method using two CLC blocks and the NCO to measure the period of the two pulses



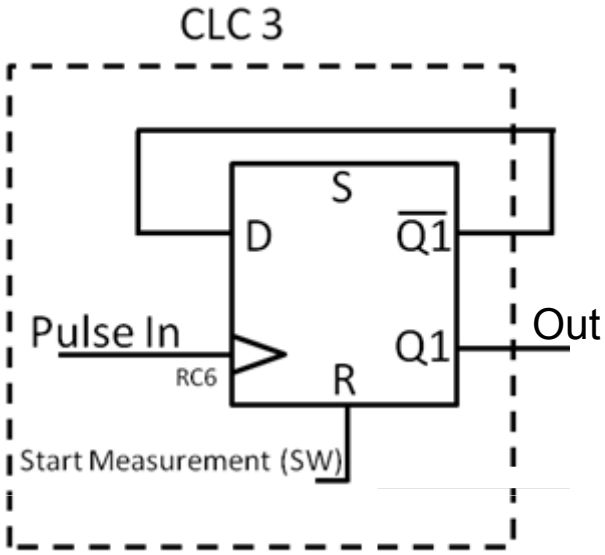
Lab 4 – Motor Speed

Period Measurement

Measures the time between two rising edges



Lab 4 – Motor Speed



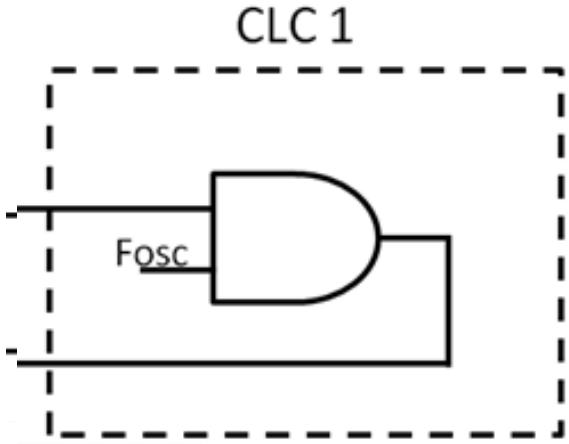
Clock Divider

This stage toggles the output for every rising edge of the clock (clock divider)

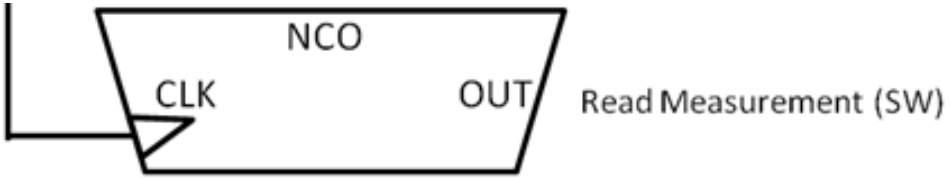
The reset line is cleared in software when you want to measure the pulse, and then set when the pulse is captured.

The interrupt flag must be configured on the output falling edge in order to signal the end of the pulse. The flag can then be polled in software.

Lab 4 – Motor Speed



This stage passes the clock to the next stage when the input is high

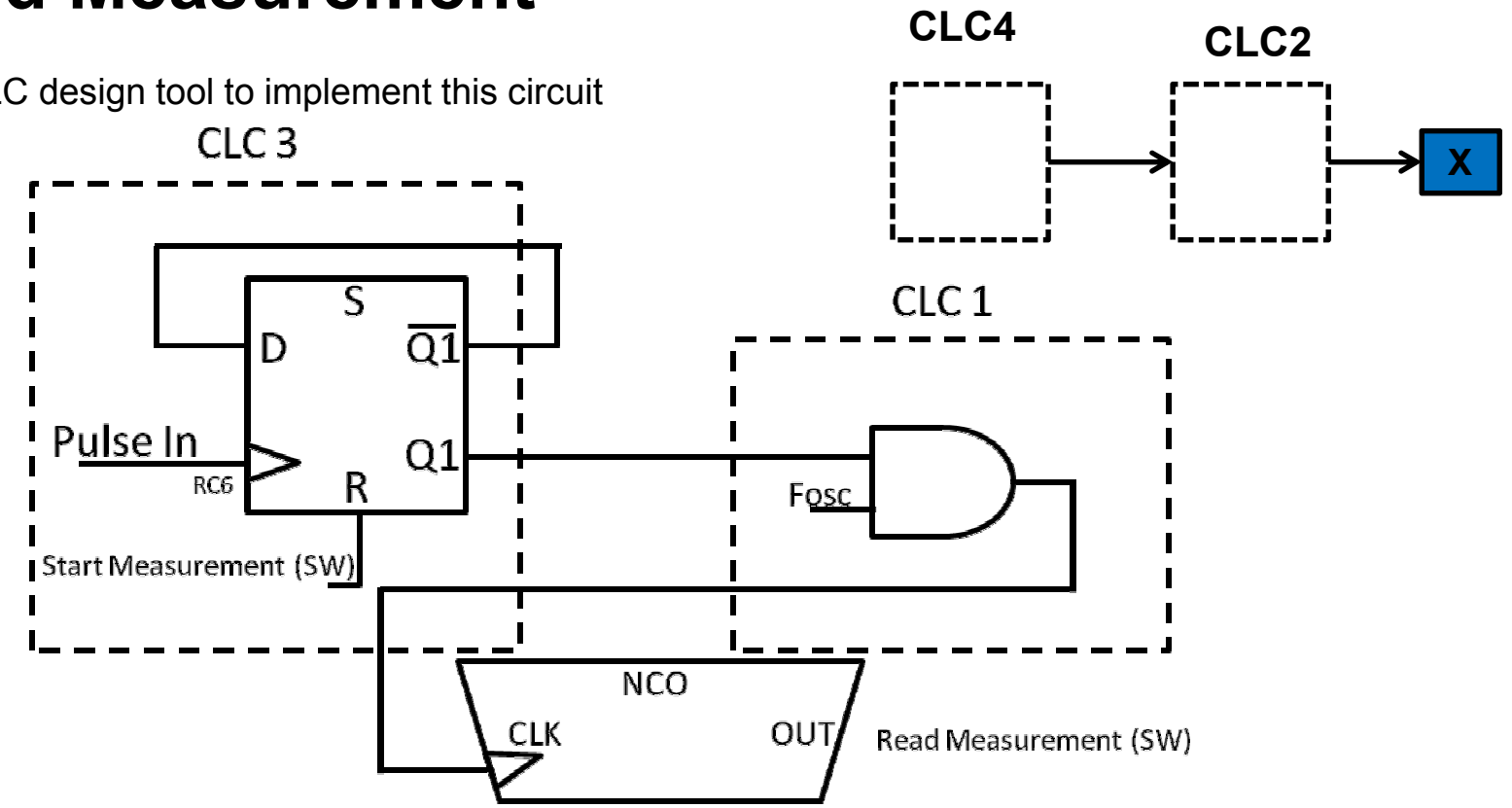


This stage times the pulse. The value is read out in software.

Lab 4 – Motor Speed

Period Measurement

Use the CLC design tool to implement this circuit



Note:
Connect POT2 to the PIC by placing a jumper on J5 on the base board.
Keep the same jumper connections from Lab 3.
Pin 2 of J6 is the top pin. Pin 2 of J7 is the right pin.

•Save as "lab4.inc" in:
C:\MASTERs\1610\Manchester Encoder.X\src

Summary

Summary

CLC Benefits

- Increases on chip interconnection of peripherals and I/O
- Integrates hardware functions & saves board area
- Save program code space & frees up resources

NCO Benefits

- Linear frequency control
- Increased frequency resolution

CWG Benefits

- Save Hardware space
- Improved switching efficiencies
- Save program code space & frees up resources

MICROCHIP MASTERs 2012

Current Parts

Device	FLASH (Bytes)	RAM (Bytes)	I/O Pins	ADC	Comp.	5-bit DAC	Comms.	CLC	CWG, NCO, Temp Ind	PWM	8-/16-bit Timers
6-pin											
PIC10(L)F320	448	64	4	3x 8-bit	-	-	-	1	☑	2	2/0
PIC10(L)F322	896	64	4	3x 8-bit	-	-	-	1	☑	2	2/0
8-pin											
PIC12(L)F1501	1.75K	64	6	4x 10-bit	1	‼	-	2	☑	4	2/1
14-pin											
PIC16(L)F1503	3.5K	128	12	8x 10-bit	2	‼	MI²C™ , SPI	2	☑	4	2/1
20-pin											
PIC16(L)F1507	3.5K	128	18	12x 10-bit	-	-	-	2	☑	4	2/1
PIC16(L)F1508	7K	256	18	12x 10-bit	2	‼	EUSART, MI²C™, SPI	4	☑	4	2/1
PIC16(L)F1509	14K	512	18	12x 10-bit	2	‼	EUSART, MI²C™ , SPI	4	☑	4	2/1

Trademarks