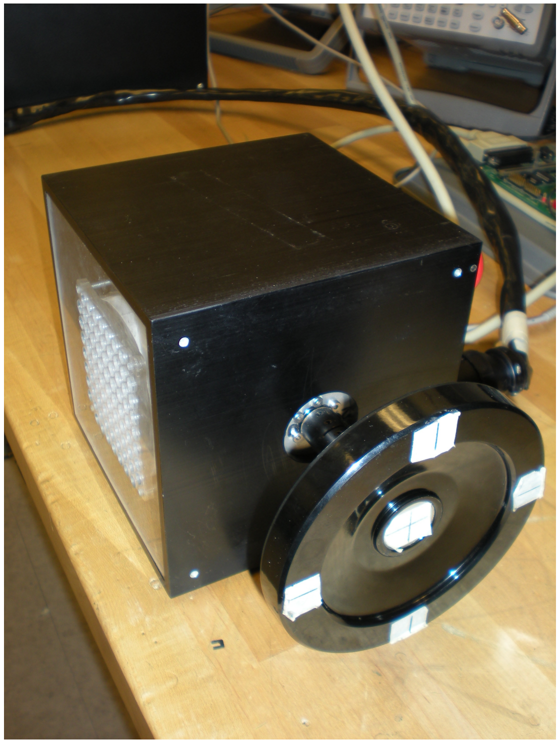
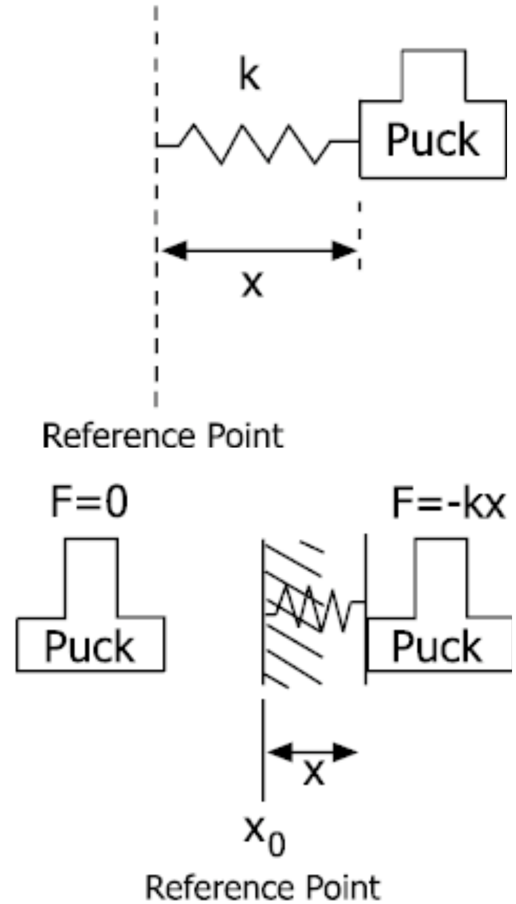
Lab 4: Pulse Width Modulation and Introduction to Simple Virtual Worlds (PWM)



Virtual Wall and Virtual Spring-Mass

- Virtual Spring-Mass
  - Puck attached to a reference point by a virtual spring with constant k
  - If the puck is moved to either side, spring exerts a restoring force Fs = -kx
  - We will use a motor and encoder to create a virtual *torsional* spring
- Virtual Wall
  - On one side of a virtual wall (x < xo), wheel spins freely (motor applies no force)
  - Once the wheel rotates into (x > xo), motor applies a force



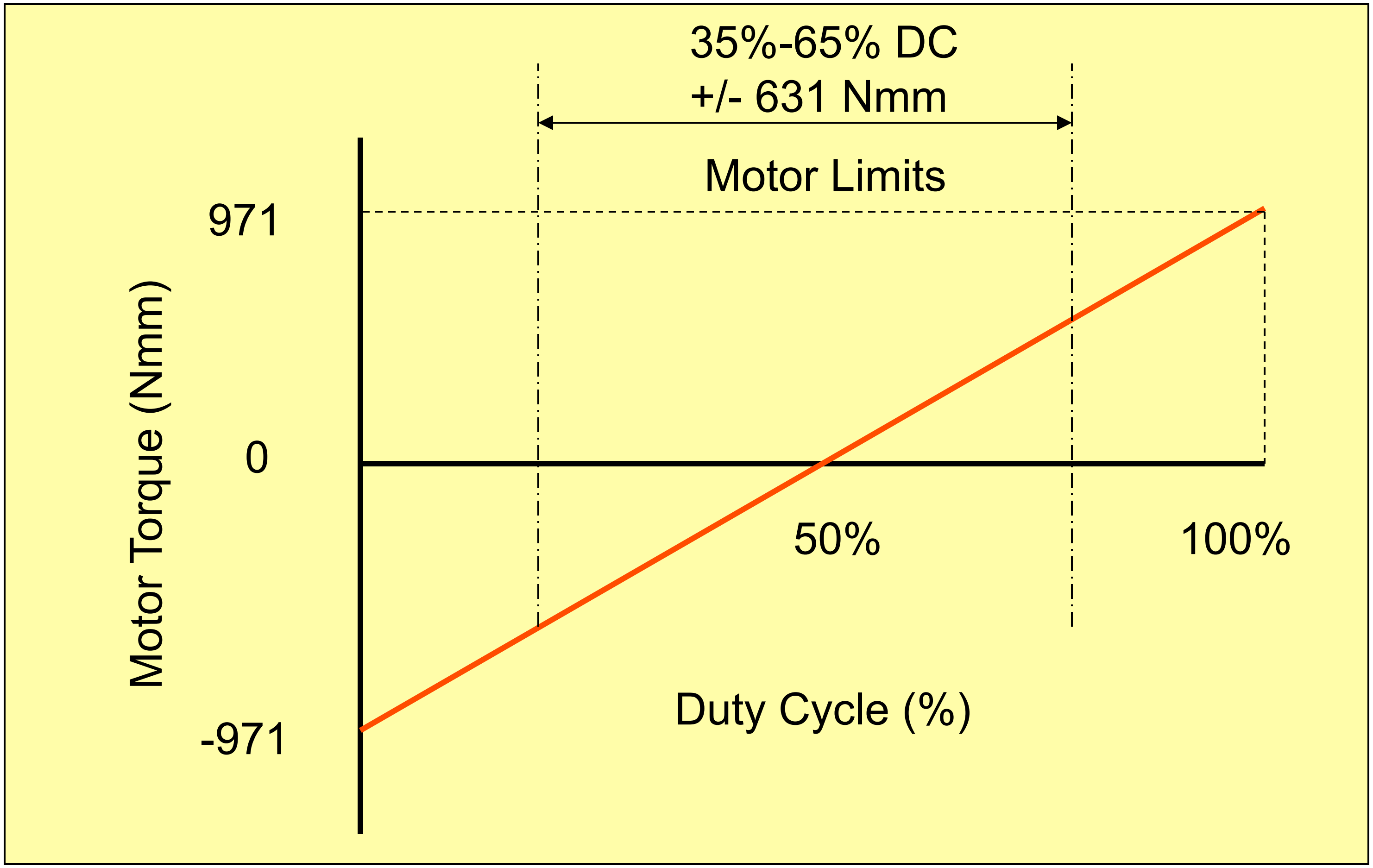
Virtual Spring-Mass (top) and Virtual Wall

Equations

- $T_w = K\Theta_w$
- T_w = Wheel Torque, *Nmm*
- K = Spring Constant, *Nmm/degree*
- Θ_w = Displacement, *degrees*

- Embedded system units are encoder counts and PWM duty cycle!
  - (Counts/Encoder Rev)(Wheel Rev/Degree) = Counts/Degree

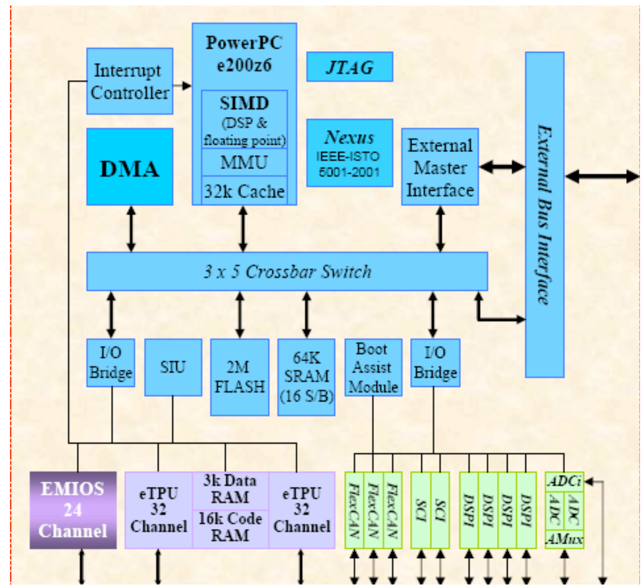
Duty Cycle-to-Motor Torque



Tm = 1.942(DC-.5) Nm

Enhanced Modular Input/Output Subsystem (eMIOS)

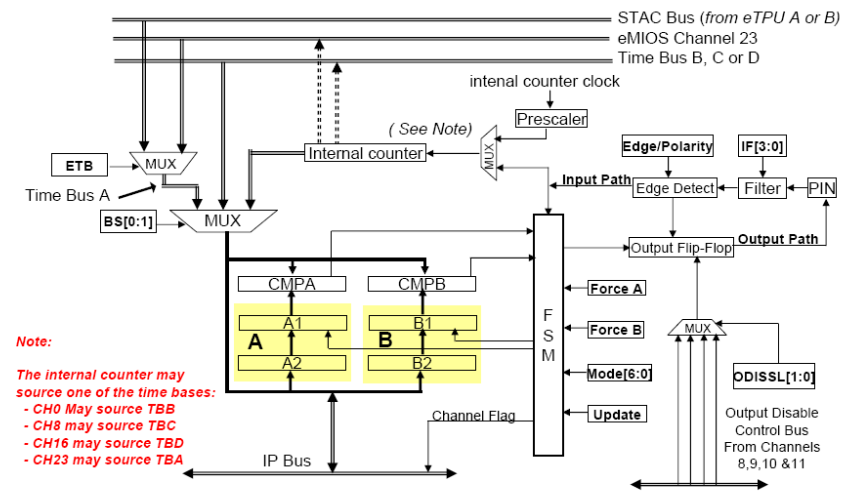
- Use eMIOS to generate Pulse Width Modulation (PWM) signal to the motor
  - 24 channels with many different operating modes
  - See Chapter 17 MPC5553-RM



- eMIOS Operation Modes
  - Timer Mode
  - Input Channel Modes
    - Single Action Input Capture
    - Input Pulse Width Measurement
    - Input Period Measurement
    - Pulse/Edge Accumulation
    - Pulse Edge Counting
    - Quadrature Decode
  - Output Channel Modes
    - Single Action Output Compare
    - Double Action Output Compare
    - Output Pulse Width Modulation
    - **Output Pulse Width and Frequency Modulation**
    - Center Aligned Output Pulse Width Modulation

eMIOS PWM

- Programming data registers A and B configure PWM duty cycle
  - Example: 10% DC:
    - A = 10; B = 100
    - Resolution = 1%
- Note that the value in register B is the pulse width (in clock ticks)
  - Resolution and frequency are related



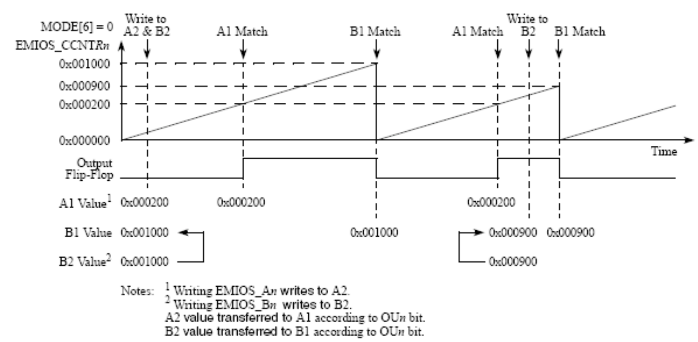


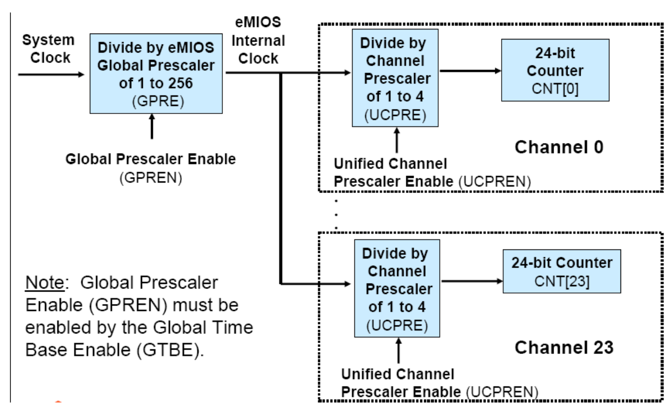
Figure 17-31. OPWFM with Immediate Update

PWM Frequency Configuration

- 2 “prescalers” located in the module control register (MCR) and the channel control register (CCR) determine the PWM frequency
  - Global Prescaler
    - GPRE: eMIOS_MCR[16:23] global prescaler divides system clock by 1 to 256 (see Table 17-7)
      - System clock is 40MHz
      - We want PWM frequency = 20000 HZ
  - Channel Prescaler
    - UCPRE: eMIOS_CCR[4:5]
    - Additional timebase scaling (divide by 1 to 4)

Table 17-7. Global Prescaler Clock Divider

GPRE[0:7]	Divide Ratio
00000000	1
00000001	2
. . . .	. . . .
11111111	256



Programming the eMIOS

- Like other peripherals, the eMIOS must be configured by writing commands to special purpose registers
  - eMIOS Module Configuration Register (MCR)
  - eMIOS Channel Control Register (CCR)
  - eMIOS Channel A/B Data Registers (CADR, CBDR)
- Structure to access these registers is contained in `MPC5553.h`

EMIOS_MCR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	FRZ	GTBE	ETB	GPREN	0	0	0	0	0	0	SRV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPRE								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	Base + 0x00															

Figure 17-2. eMIOS Module Configuration Register (EMIOS_MCR)

- GPRE: Global prescaler - selects the clock divider as shown in Table 17-7
- GPREN: Prescaler enable (enabled = 1)
- GTBE: Timebase enable (enabled = 1)

EMIOS_CCR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BSL		EDSEL	EDPOL	MODE						
W			FOR CMA	FOR CMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x0C															

Figure 17-8. eMIOS Channel Control Register (EMIOS_CCR*n*)

- See Table 17-10
- **UCPRE**: Selects clock divider
  - 0b00 = divide by 1
  - 0b11 = divide by 4
- **UCPREN**: Prescaler enable (enabled = 1)
- **BSL**: Bus select (use internal counter, BSL = 0b11)
- **EDPOL**: Edge polarity (trigger on falling edge = 0)
- **MODE**: Selects the mode of operation. See Table 17-11 (we want **output pulse width and frequency modulation with next period update**)

Lab 4 Software

- As usual, you are given `mios.h` with function prototypes; you will write the functions in `mios.c`, plus application code in `lab4.c`
- Four functions are required:
  - `Init_MIOS_clock`
  - `Init_PWM`
  - `Set_PWMPeriod`
  - `Set_PWMDutyCycle`

Lab 4 Software

- `Init_MIOS_clock, Init_PWM`:
  - Configure the MCR, CCR and set initial values for the data registers
  - Use the structure defined in MPC5553.h to access the registers
  - Initialize the data registers to 50% duty cycle (zero torque output)
  - Don't forget to turn on the output pads for the PWM channel

```
/* Init data registers A and B for 50% duty cycle */
EMIOS.CH[miosChannel].CADR.R = newPeriod>>1; /* divide by 2 */
EMIOS.CH[miosChannel].CBDR.R = newPeriod;
```

```
 /* Turn on the output pads for our PWM channel */
  SIU.PCR[179 + miosChannel].B.PA = 0b11;
  SIU.PCR[179 + miosChannel].B.OBE = 0b1;
```

Lab 4 Software

- `Set_PWMPeriod, Set_PWMDutyCycle`
  - 24 bit values written to data registers CADR*n*, CBDR*n* determine period and duty cycle
  - Values are NOT units of time
    - "Clock Ticks" per period
    - For 40MHz system clock counts_per_period = 40000000/PWM_FREQ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	A							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x00															

Figure 17-5. eMIOS Channel A Data Register (EMIOS_CADR*n*)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x04															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reg Addr	UC*n* Base + 0x04															

Figure 17-6. eMIOS Channel B Data Register (EMIOS_CBDR*n*)

Lab 4 Assignment

- Use everything you've learned so far:
  - Read a duty cycle value from a QADC pin and output a PWM signal to the oscilloscope
  - Drive the motor and haptic wheel with the PWM signal
    - Experiment with different frequencies and observe motor response
      - What do you expect to happen at 2Hz? 20KHz?
    - Output a constant 200 Nmm torque
  - Implement the virtual spring and virtual wall using FQD function of the eTPU and the eMIOS PWM
    - Experiment with different values of the spring constant and observe the effect

Lab 4 Assignment

- You will need to write the following code (template files are provided)
  - worlds.h and worlds.c
    - Code for the virtual spring and virtual wall
    - As usual, prototypes are contained in worlds.h; you write the code for these functions in worlds.c
  - motor.h and motor.c
    - Code to generate motor output torque
  - lab4.c
    - Read the encoder, calculate the restoring torque and output the appropriate PWM to the motor