NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

MA4825

Robotics

Xie Ming, PhD (France)

mmxie@ntu.edu.sg

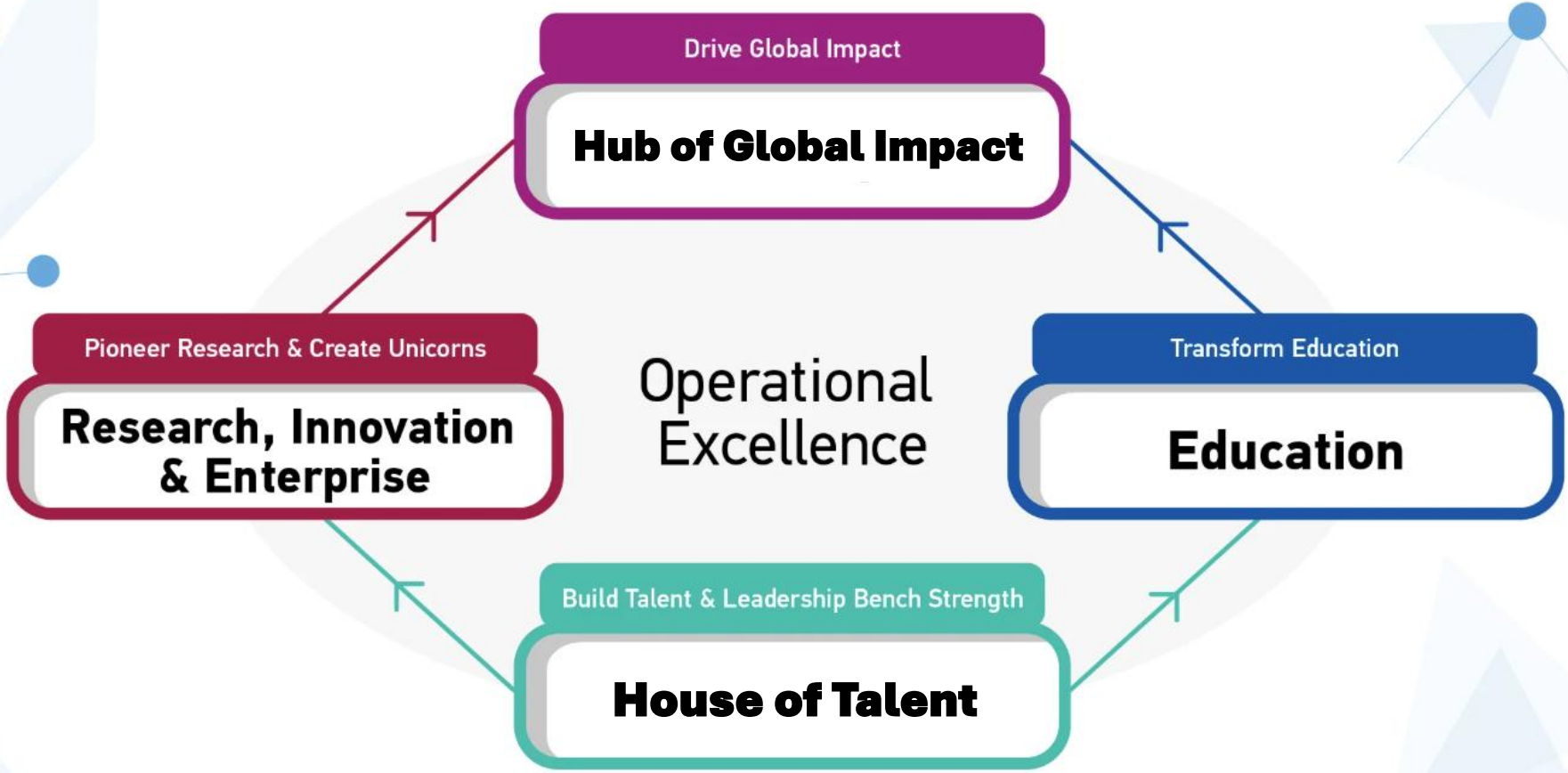
http://personal.ntu.edu.sg/mmxie

Outline

- Module 1: Robot's Advanced Body
- Module 2: Robot's Advanced Perception
- Module 3: Robot's Advanced Planning
- Module 4: Robot's Advanced Control

About NTU

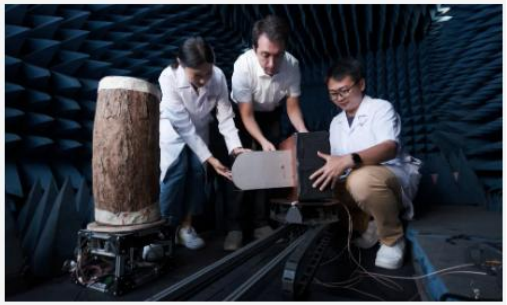
Remember NTU's Vision …



Remember NTU's Mission …

Education

We deliver transformative educational experiences that make our students both future- and AI-ready, so they are sought after by employers.



House of Talent

We attract, develop, and retain the very best people to drive excellence across the University.

Research, Innovation and Enterprise

We pursue breakthrough discoveries. We integrate technology and the humanities to address global challenges. We accelerate cutting edge innovation and create promising new enterprises.

Hub of Global Impact

We drive global impact in all that we do. We pursue long-lasting global partnerships with like-minded institutions across the world.

Education is to help citizens to fulfill their missions on Earth, which include: to understand the world and to improve the world …

0月18日 Friday
农历癸巳年九月十四
版／售价80￠
obao | zaobao.com
加坡报业控股

联合早报

南洋 ○ 星洲

部分时间托管残疾者
明年也可获80%津贴
3 焦点 〉

拒载德士增加
今年210人受罚 4 新加坡 〉

华乐团招募4000人
明年在国家体育场办音乐会
7 新加坡 〉

李总理：排名不应是办大学终极考量

黄伟曼 报道
ngwaimun@sph.com.sg

本地大专学府的主要服务对象是新加坡学生，大学除了追求学术表现之外，也扮演重要的国家与社会角色，李显龙总理提醒本地大学不应为追求名次而忽略所肩负的重任，必须持续改善本地教育，为新加坡人提供精益求精的教育。

李总理昨晚主持新加坡国立大学大学城（University Town，简称UTown）开幕仪式时指出，本地大学的世界排名一直受到关注，但在衡量大学的表现时，不应只看名次先后，排名也不应是办大学的终极考量。

他向现场近1000名国大学生、教师与嘉宾解释说，许多在世界大专教育排行榜上名列前茅的大国顶尖大学收生广泛，例如美国哈佛大学、斯坦福大学、英国牛津大学、剑桥大学、中国北京大学和清华大学等每年录取的学生中，只有一小部分是当地学生，而本地的情况却不同，大专学府主要收本地生，必须履行更广泛的国家任务。

他说："本地大学的责任在于发展学生的品格与社会良知、鼓励他们在求学时期建立长久的友谊与默契。大学要以灌输新加坡价值观和精神特质为出发点，教导学生取之社会用之社会的精神，让他们知道自己有责任推动新加坡前进。"

"要达到这个宗旨，每一所大学都应该建立自己独特的教育模式，不要一味复制其他大学。"

占地19公顷的大学城是国大搬到肯特岗校园现址后，最大型的一次扩展计划。设于大学城的四所寄宿型学院和为研究生所设的大学城宿舍，将为4000多名学生提供住宿，将教学融入住宿环境里，让来自不同背景和科系的学生一起生活、一起学习。

在大学城学院课程中，学生需在所住的学院里修读五个单元，包括两个写作单元和三个研讨课单元。他们也有机会聆听外国学者发表专题演讲，并参加各种不同的课外活动。

李总理昨天肯定国大为提升学生学习体验所做的努力，他认为大学城为学生营造了一个有利于相互学习的空间。他指出，大学城是国大的"新聚集点"，它将国大社群缩小，进一步拉近学生与同学和教授的关系。

他也以陈爱丽丝与彼德学院（College of Alice & Peter Tan)学生之前在"整合式课程"（Capstone Programme）下为乐龄人士制定健康饮食计划为例，表示希望看到大学城继续推行类似社会计划，鼓励学生积极参与这些关怀社会的活动。

国大校长陈祝全教授致辞时指出，除了设施与空间设计吸引人之外，大学城为国大社群以及国大的运作和成长带来了深刻且积极的影响。"大学城可说已成为国大校园的核心，在一些方面补足了之前肯特岗校园所缺少的生气。"

相关新闻刊第6页

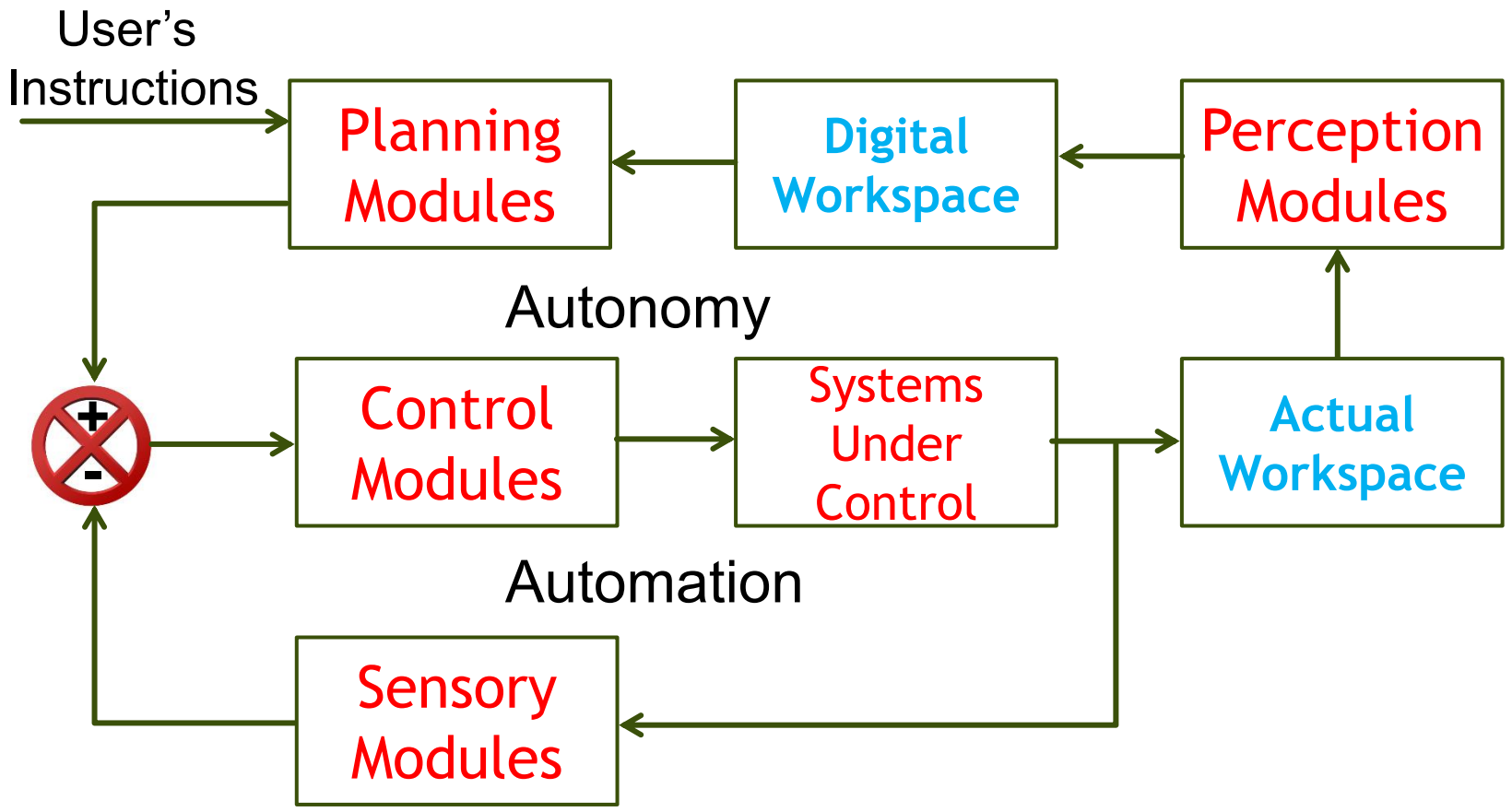
About You

Remember your mission as MAE undergraduates …

- You are here to grow your knowledge and skills so as to be able to design machines with controllable behaviors and hopefully in some intelligent ways.

How to fulfill your mission?

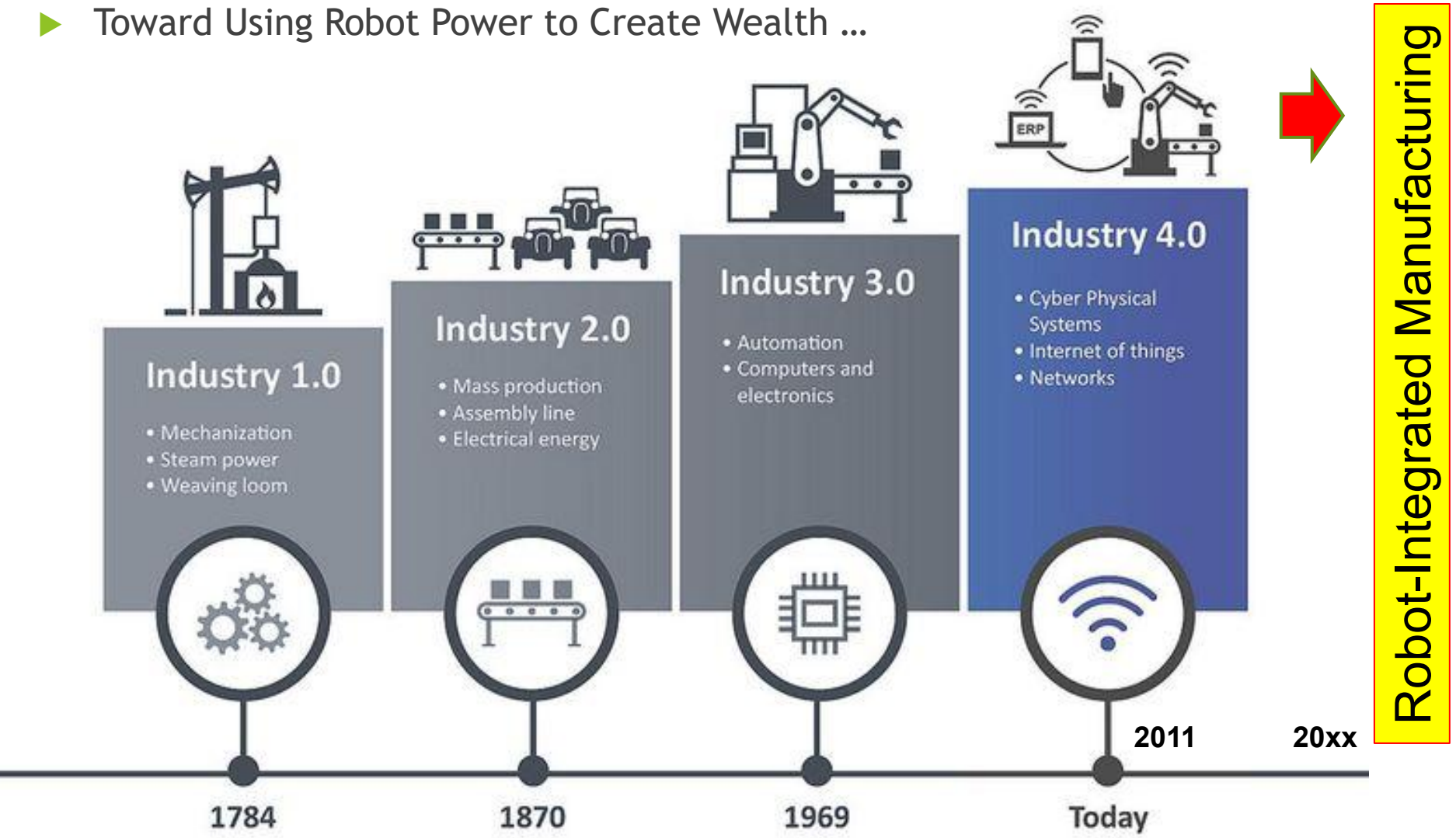
- To apply learnt knowledge and skills into the implementation of the following universal blueprint underlying all the intelligent machines or systems.



About Course

Why to study this course?

- Toward Using Robot Power to Create Wealth …



How to study this course?

- To put yourselves into the mindset of designers of robots as products:
  - Who are the users?
  - What are the needs of users?
  - What are your robots which could meet the needs of your users or buyers?
  - What are the solutions behind the design of your robots?

Market Demands or Needs

↓

Product Specifications

↓

Design Specifications

↓

Conceptual Design

↓

Selection of Materials/Components/Devices

↓

Embodiment Design

↓

Prototyping

↓

Optimizing

↓

Production

↓

Marketing

What to Learn?

Q1: What is the energy flow?

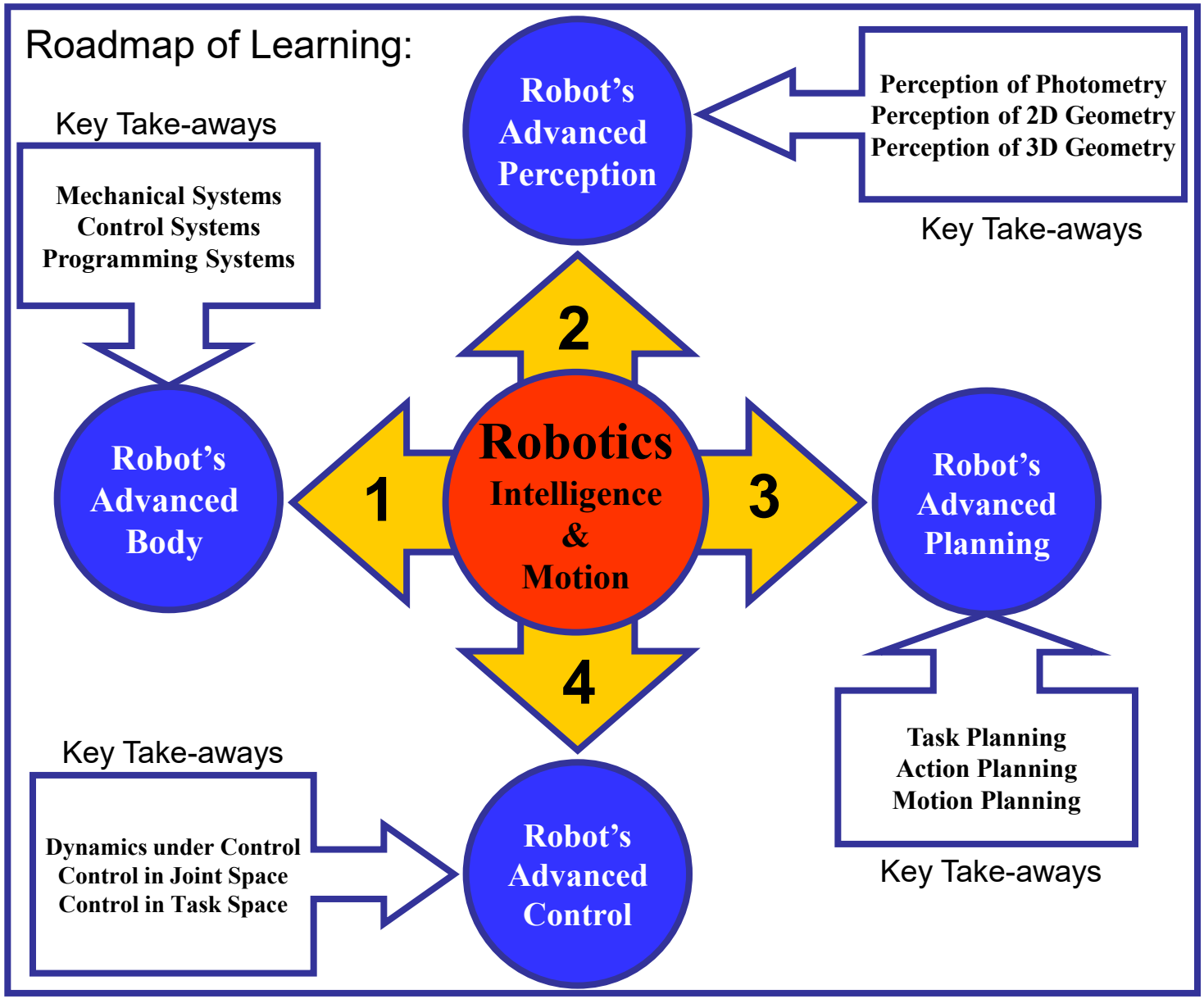
Q2: What is the signal flow?

Q3: What is the knowledge flow?

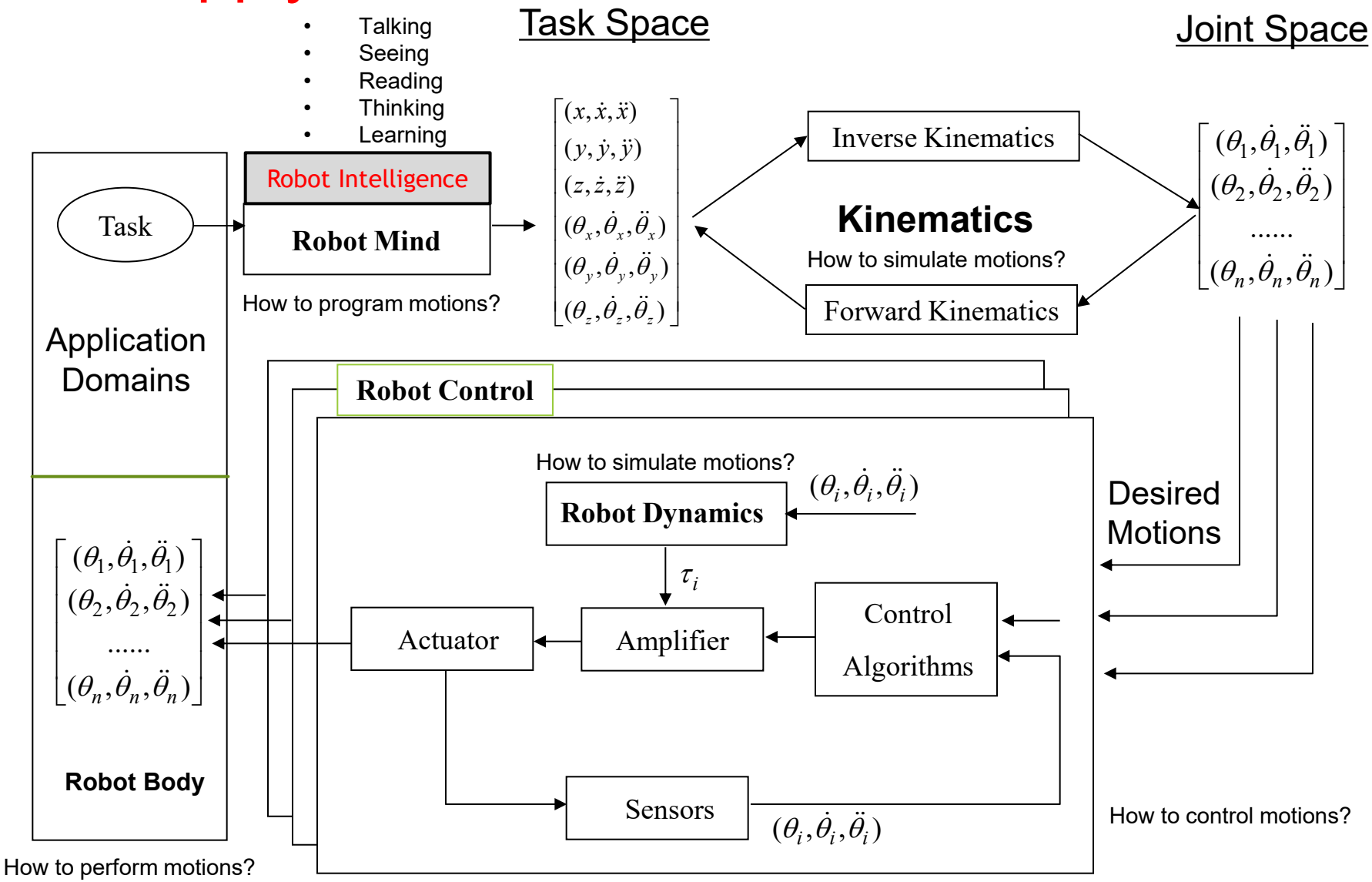
Q4: What is the relationship between energy flow and signal flow?

Q5: What is the relationship between signal flow and knowledge flow?

1. One Machine
2. Two Capabilities
3. Three Benefits
4. Four Pillars



How to Apply?


Talking
Seeing
Reading
Thinking
Learning
Task Space
Joint Space
Robot Intelligence
Robot Mind
Task
How to program motions?
$(x,\dot{x},\ddot{x})$
$(y,\dot{y},\ddot{y})$
$(z,\dot{z},\ddot{z})$
$(\theta_x,\dot{\theta}_x,\ddot{\theta}_x)$
$(\theta_y,\dot{\theta}_y,\ddot{\theta}_y)$
$(\theta_z,\dot{\theta}_z,\ddot{\theta}_z)$
Inverse Kinematics
Kinematics
How to simulate motions?
Forward Kinematics
$(\theta_1,\dot{\theta}_1,\ddot{\theta}_1)$
$(\theta_2,\dot{\theta}_2,\ddot{\theta}_2)$
.....
$(\theta_n,\dot{\theta}_n,\ddot{\theta}_n)$
Application Domains
Robot Control
How to simulate motions?
Robot Dynamics
$(\theta_i,\dot{\theta}_i,\ddot{\theta}_i)$
τ_i
Desired Motions
Actuator
Amplifier
Control Algorithms
Sensors
$(\theta_i,\dot{\theta}_i,\ddot{\theta}_i)$
Robot Body
How to perform motions?
How to control motions?

Terminology Alert

- Advanced Robotics is about the study of advanced robots which could perform tasks in some intelligent ways.

- Advanced Robot is a machine which has
  - two capabilities (automatic control and autonomous control),
  - three benefits and
  - four pillars.

Today's Lectures …

- Module 1: Robot's Advanced Body
- Module 2: Robot's Advanced Perception
- Module 3: Robot's Advanced Planning
- Module 4: Robot's Advanced Control

Module 1

MA4825 Robotics

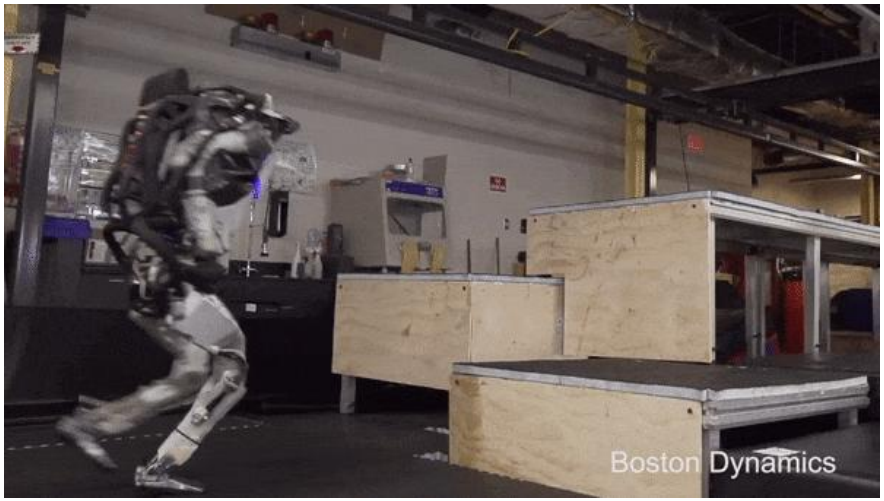
Robot's Advanced Body

Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

Outline of Module 1

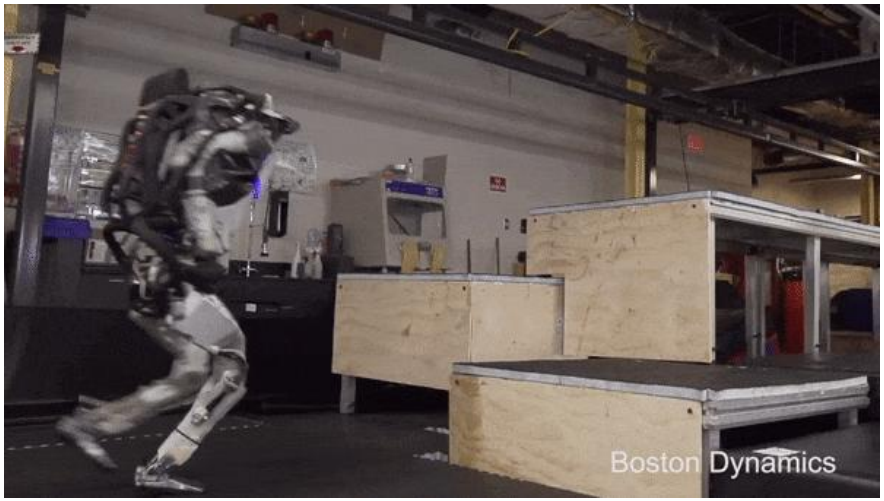
- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

Outline of Module 1



- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems

What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

Lecture 1

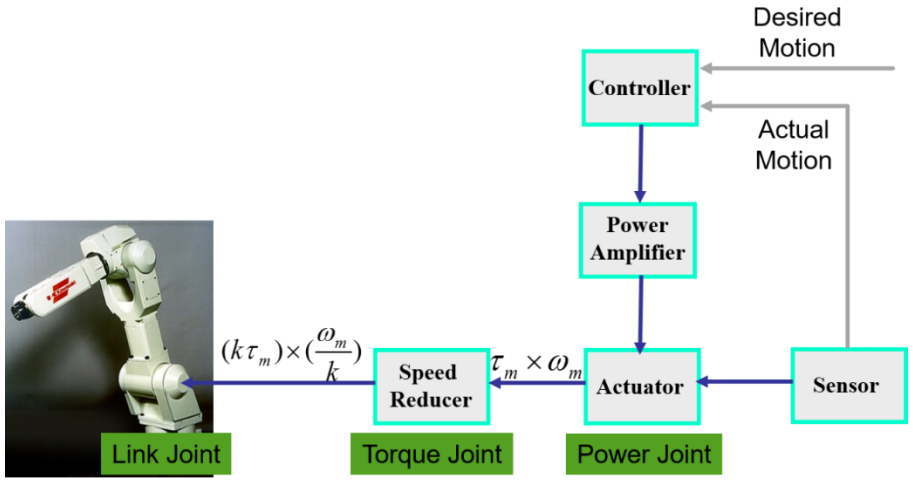
Robot Systems

Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

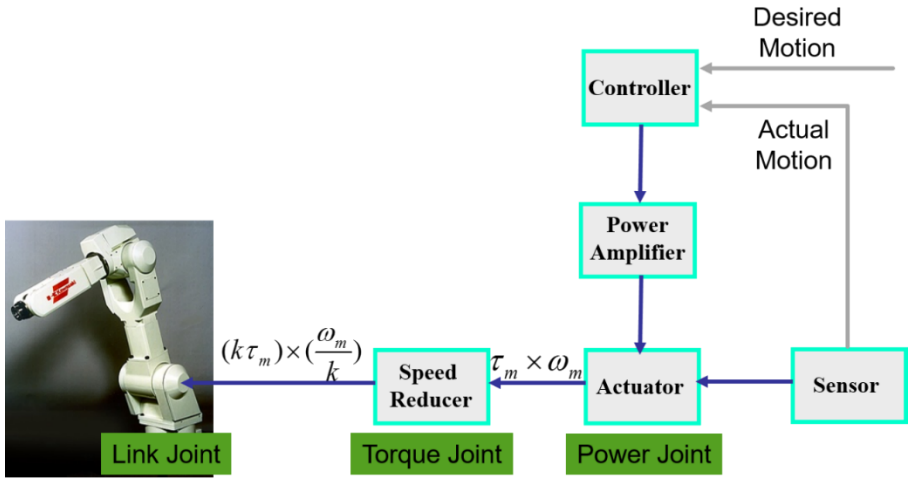
Outline of Lecture 1

- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems



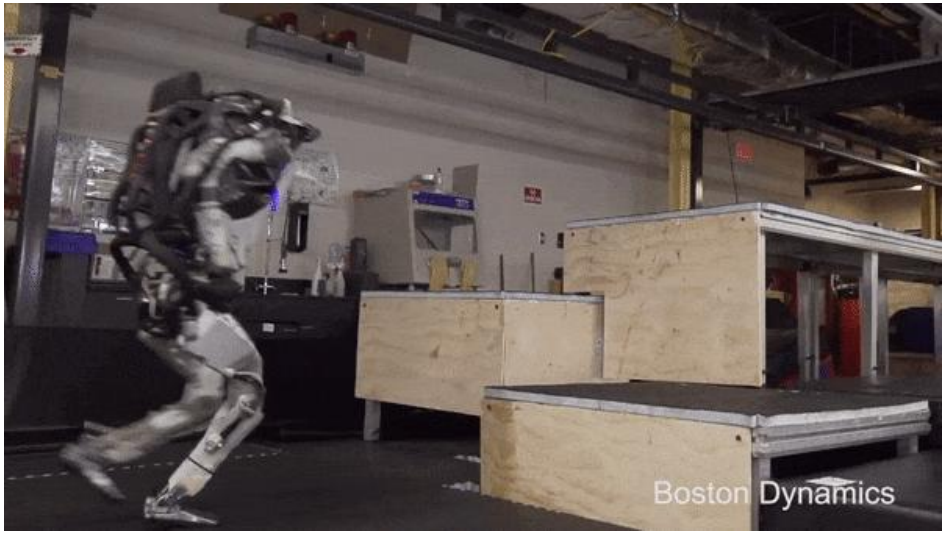
Outline of Lecture 1

- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems



Definition of System

- A system is a set of entities which are interconnected and act together for achieving common goals or outcome.



Name five systems that you know

- 1
- 2
- 3
- 4
- 5

Characteristics of Systems

- A system has input.
- A system has responses.
- A system has transfer function.

Illustration of Systems without Error Control (i.e. Open-loop Systems)

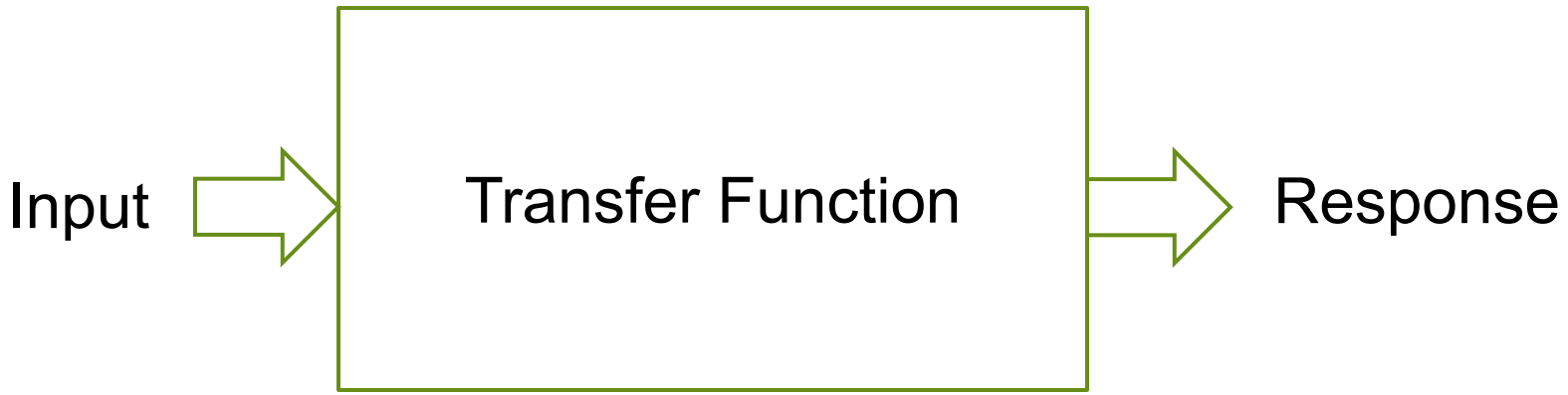
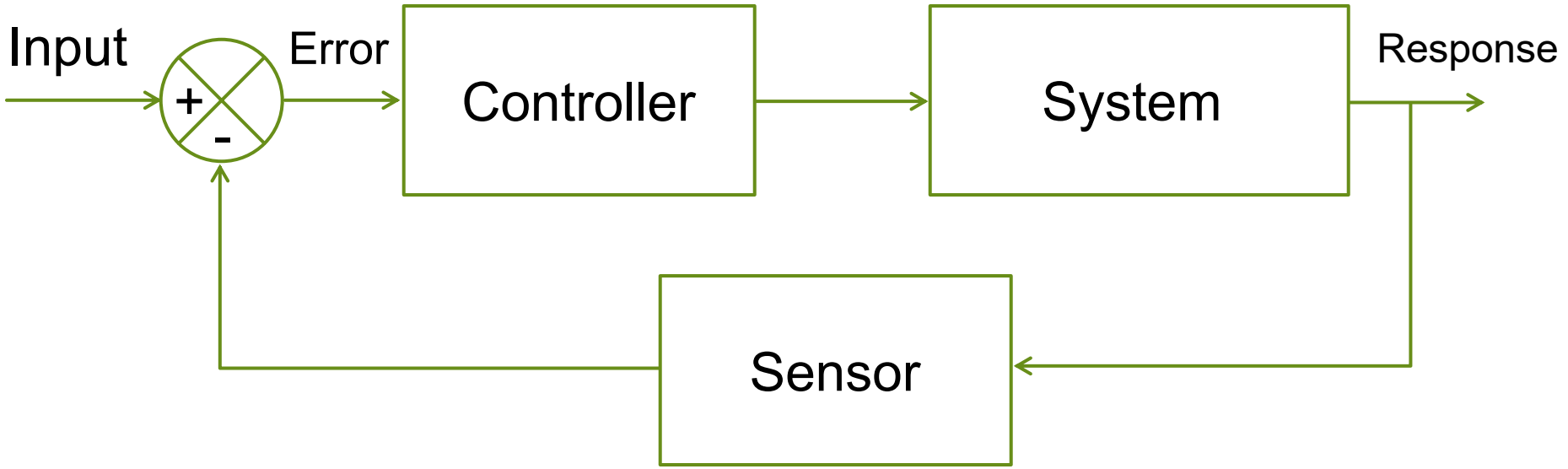


Illustration of Systems with Error Control (i.e. Closed-loop Control Systems)



This is a system which responds to error

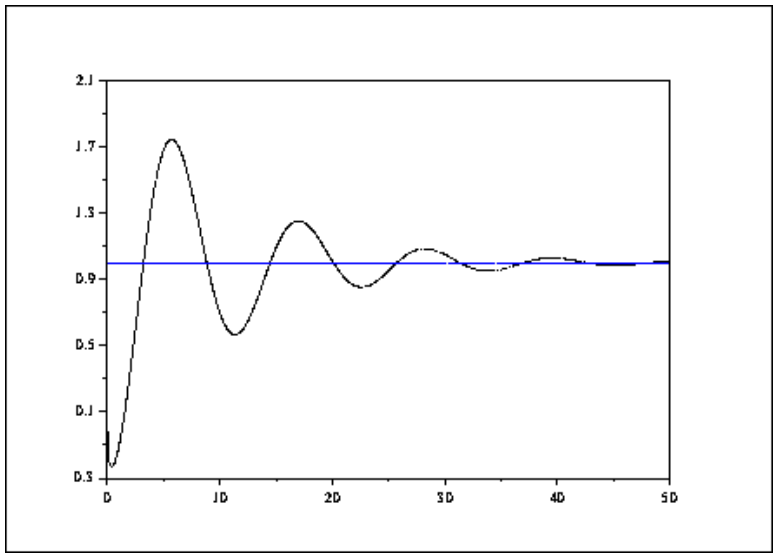
Characteristics of Systems' Responses

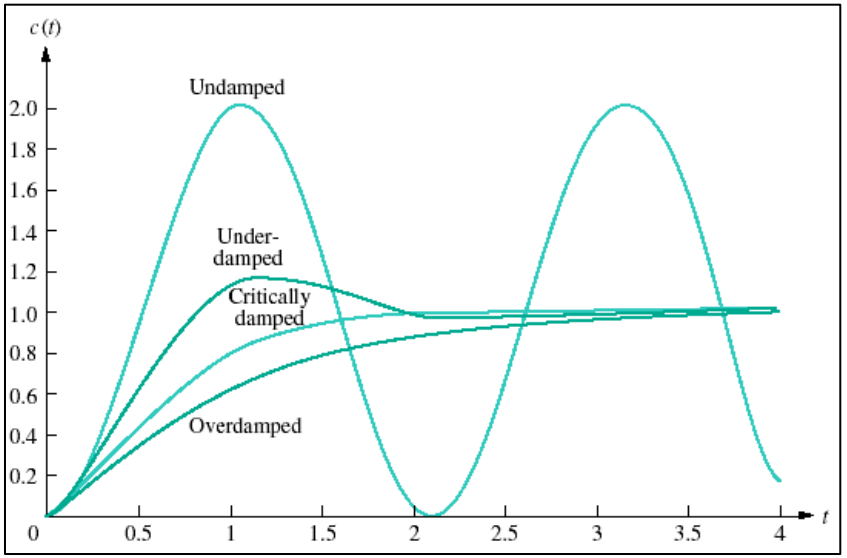
- Transient Responses:
  - Transition from one steady-state response to another steady-state response.

- Steady-state Responses:
  - Output which stabilizes at steady states.

Static systems only have steady-state responses

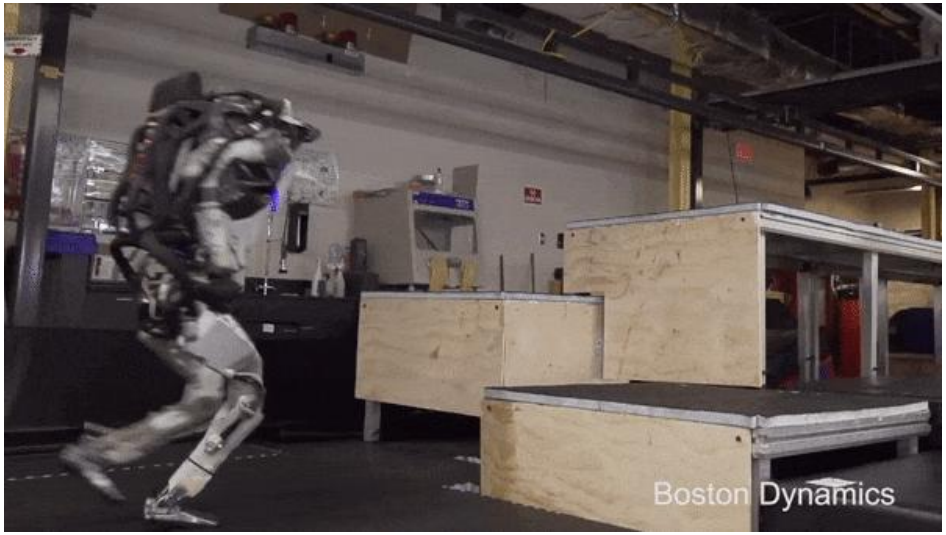
Illustration of Systems' Responses





Performance of Dynamic Systems

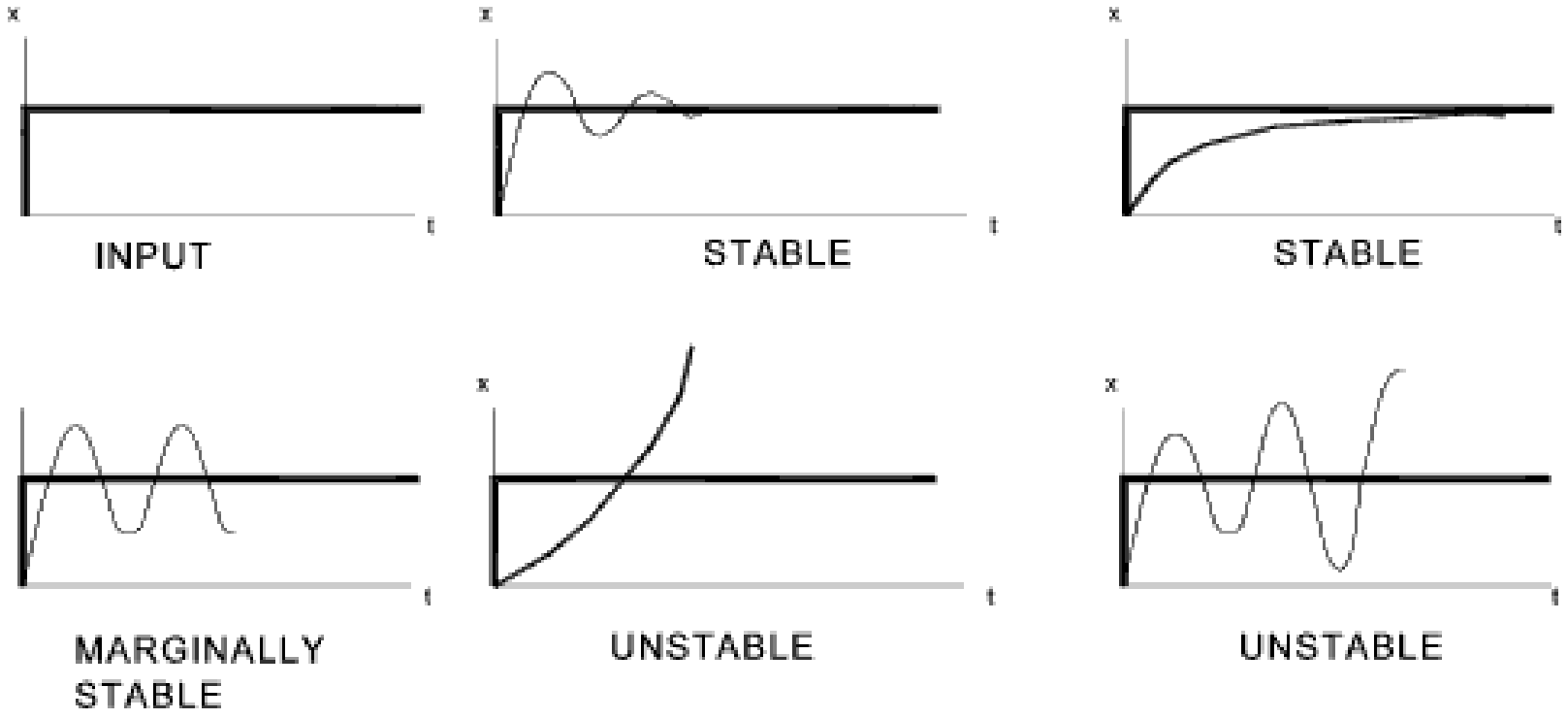
- Stability
- Response Time
- Response Accuracy



Static Systems:
- 100% stable
- Zero response time
- 100% accurate

Dynamic Systems:
- 100% stable: Possible
- Zero response time: Not possible
- 100% accurate: Possible

Illustration of Stability



In time domain, stability means bounded-input bounded output

Illustration of Response Times

- Rise Time
- Peak Time
- Settling Time

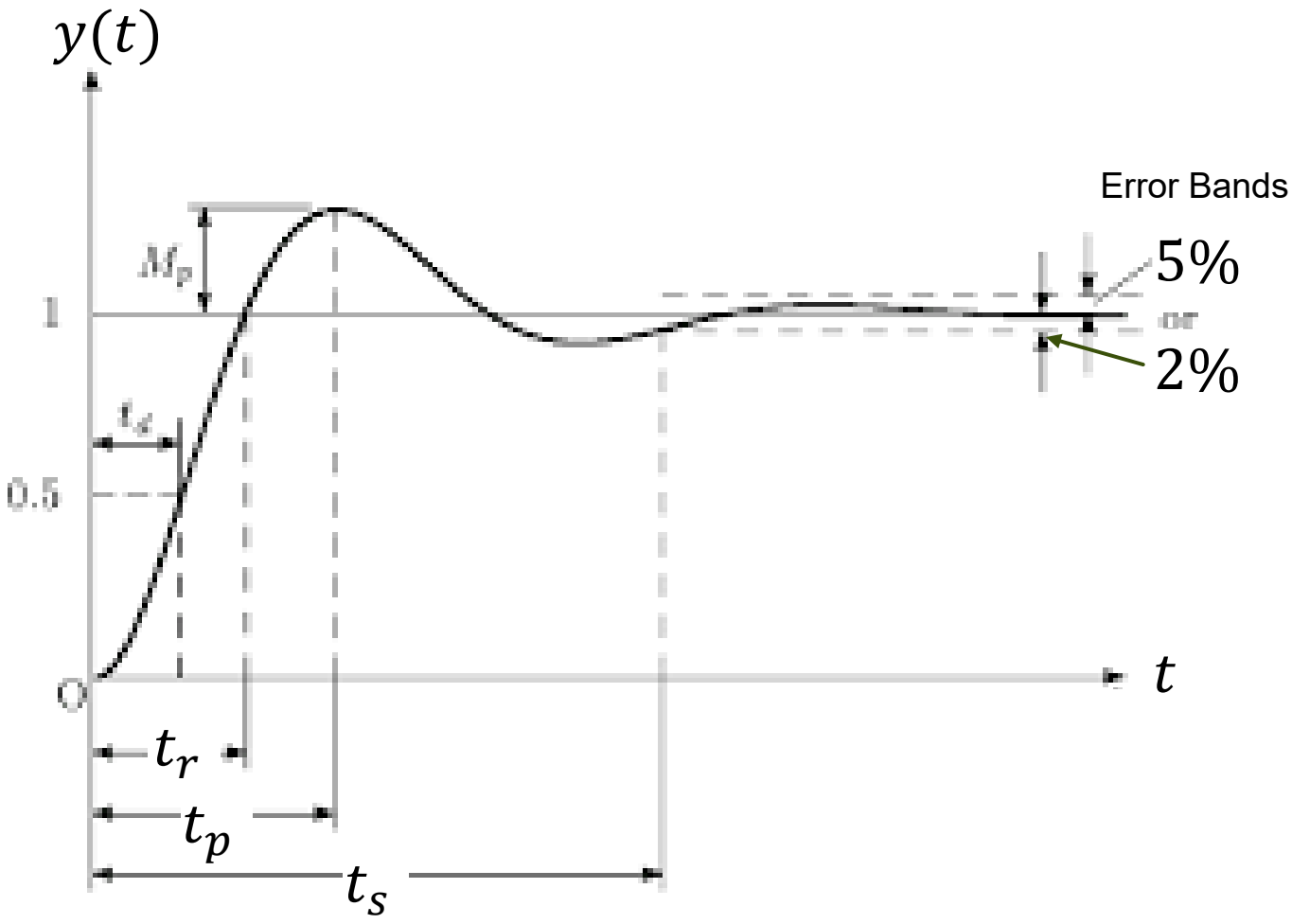
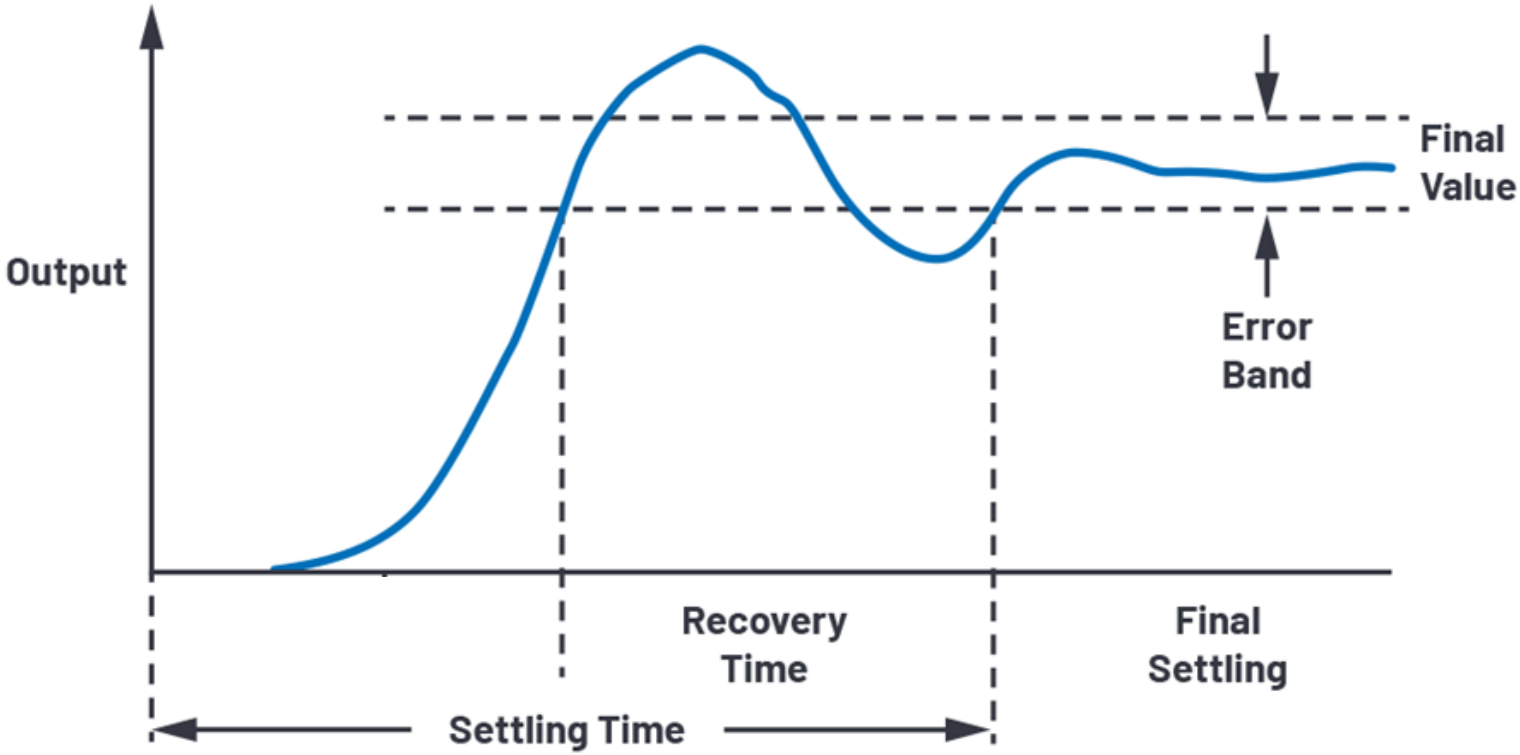


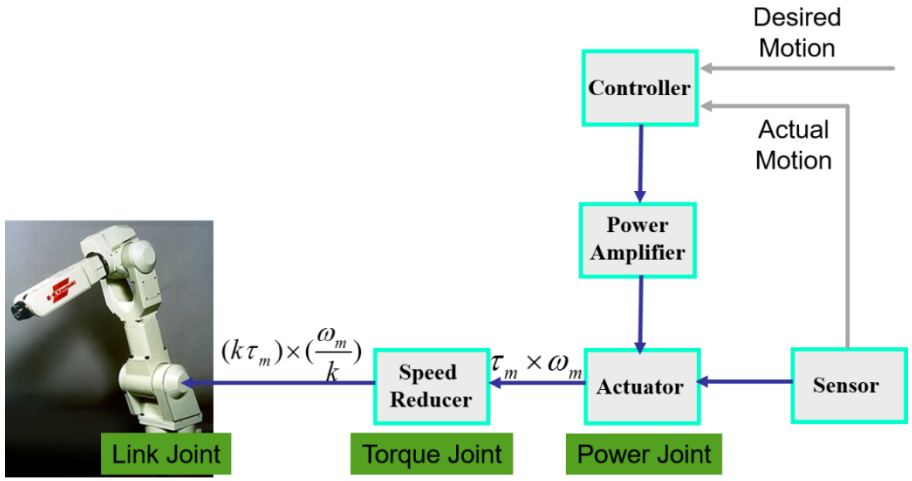
Illustration of Response Accuracy

- Error Band: 2%
- Error Band: 5%
- etc



Outline of Lecture 1

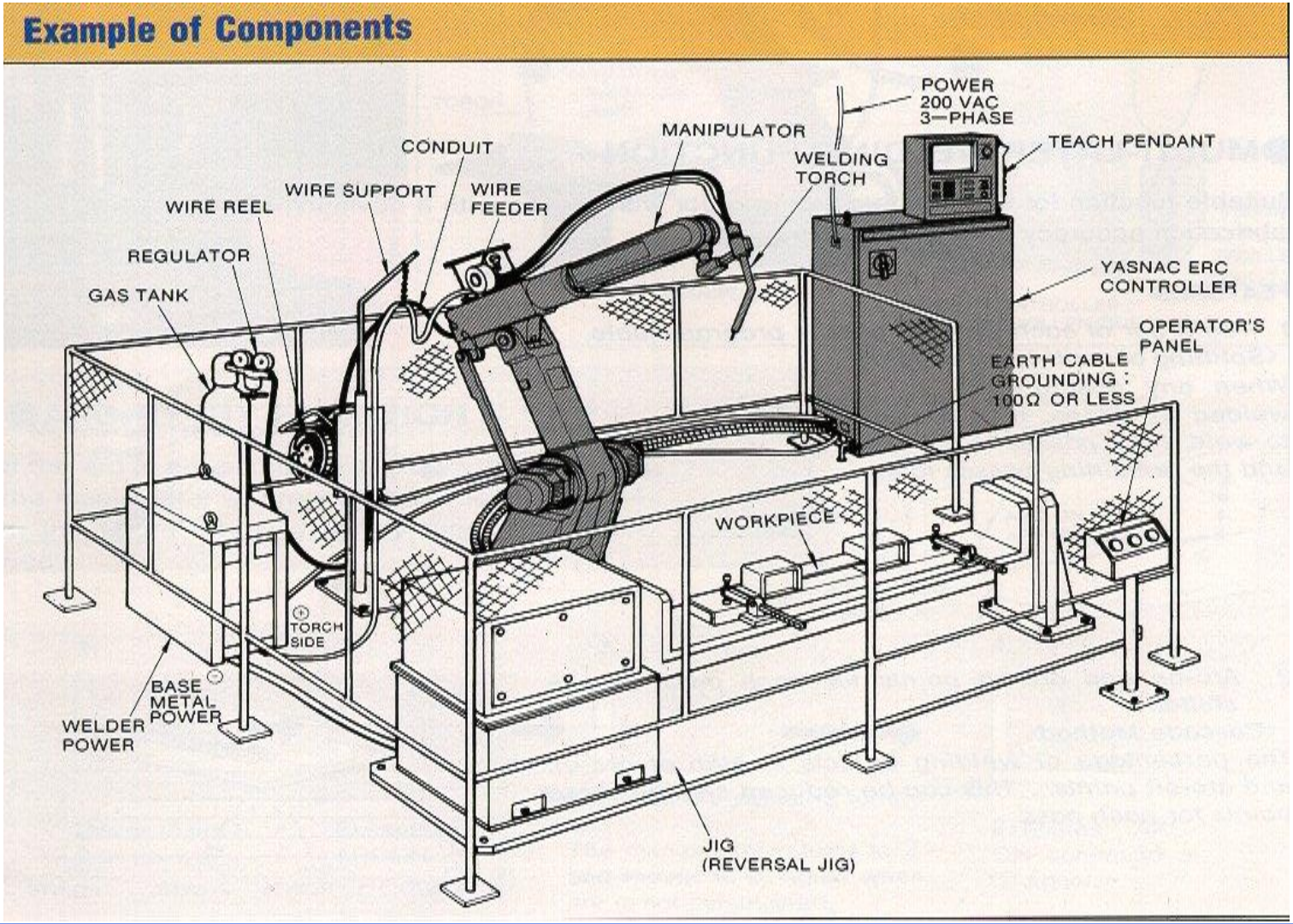
- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems



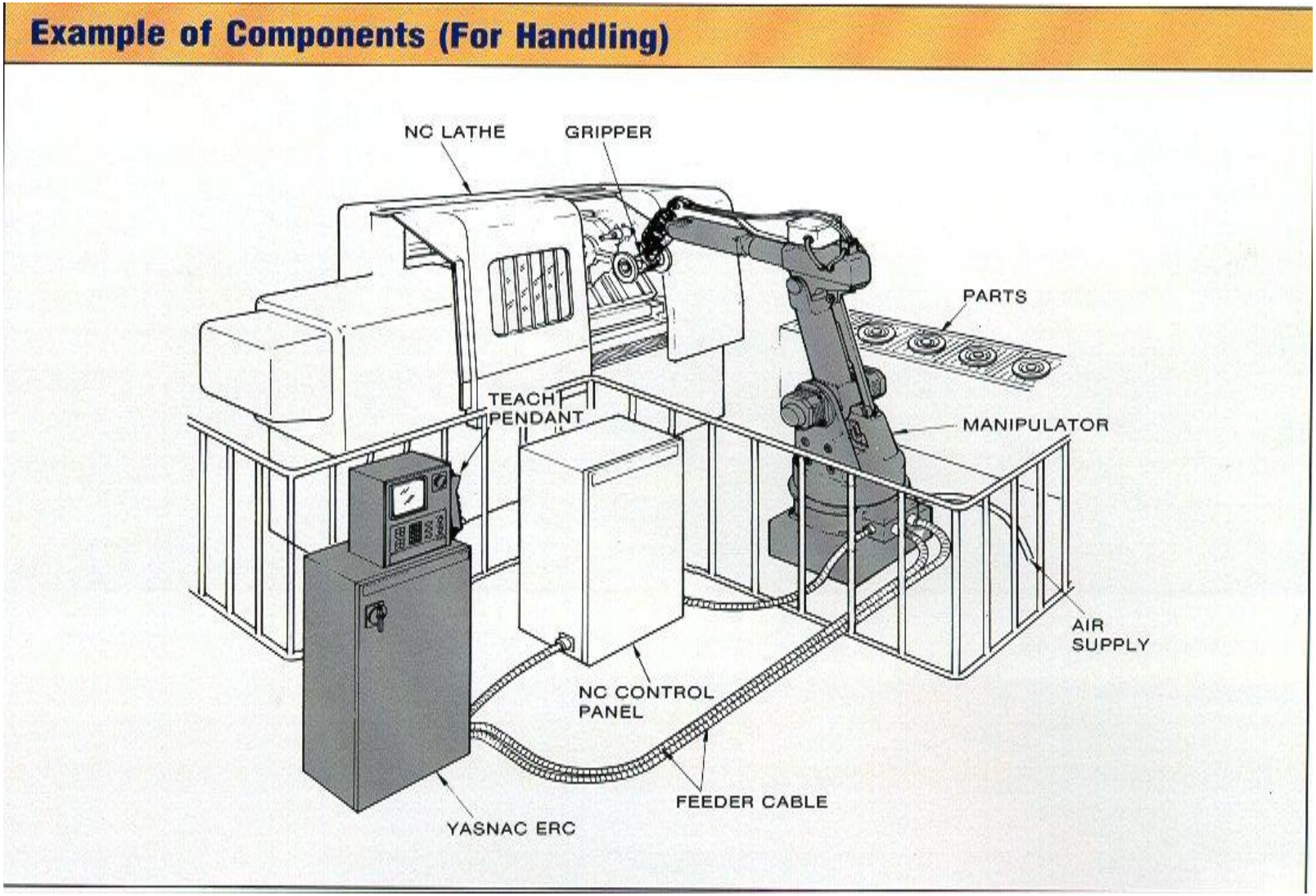
Definition of Robot System

- A robot system is a set of modules which are interconnected and act together for achieving common goals such as:
  - Imitating the skills of human beings
  - Imitating the intelligence of human beings

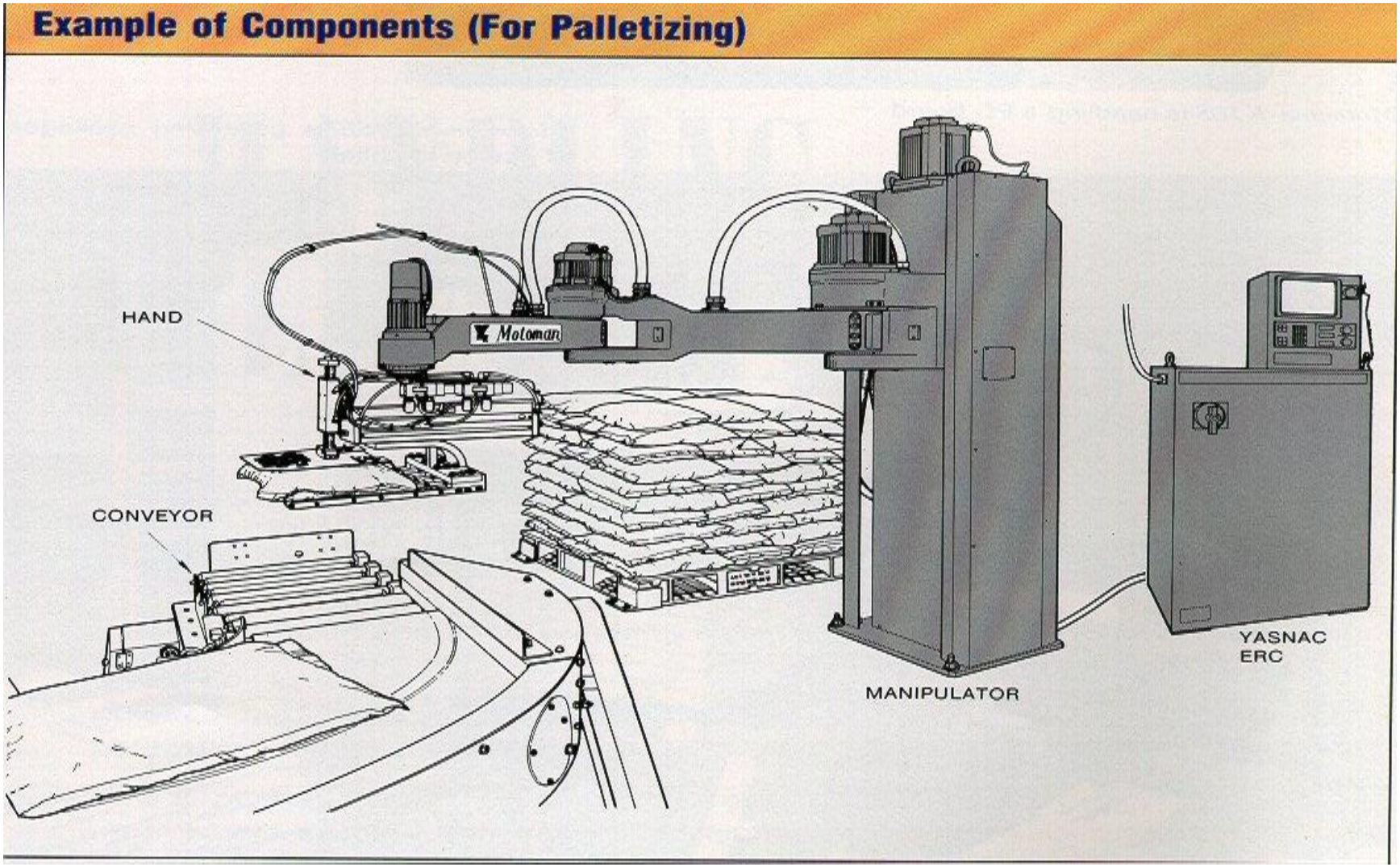
Example: Robot Systems for Welding



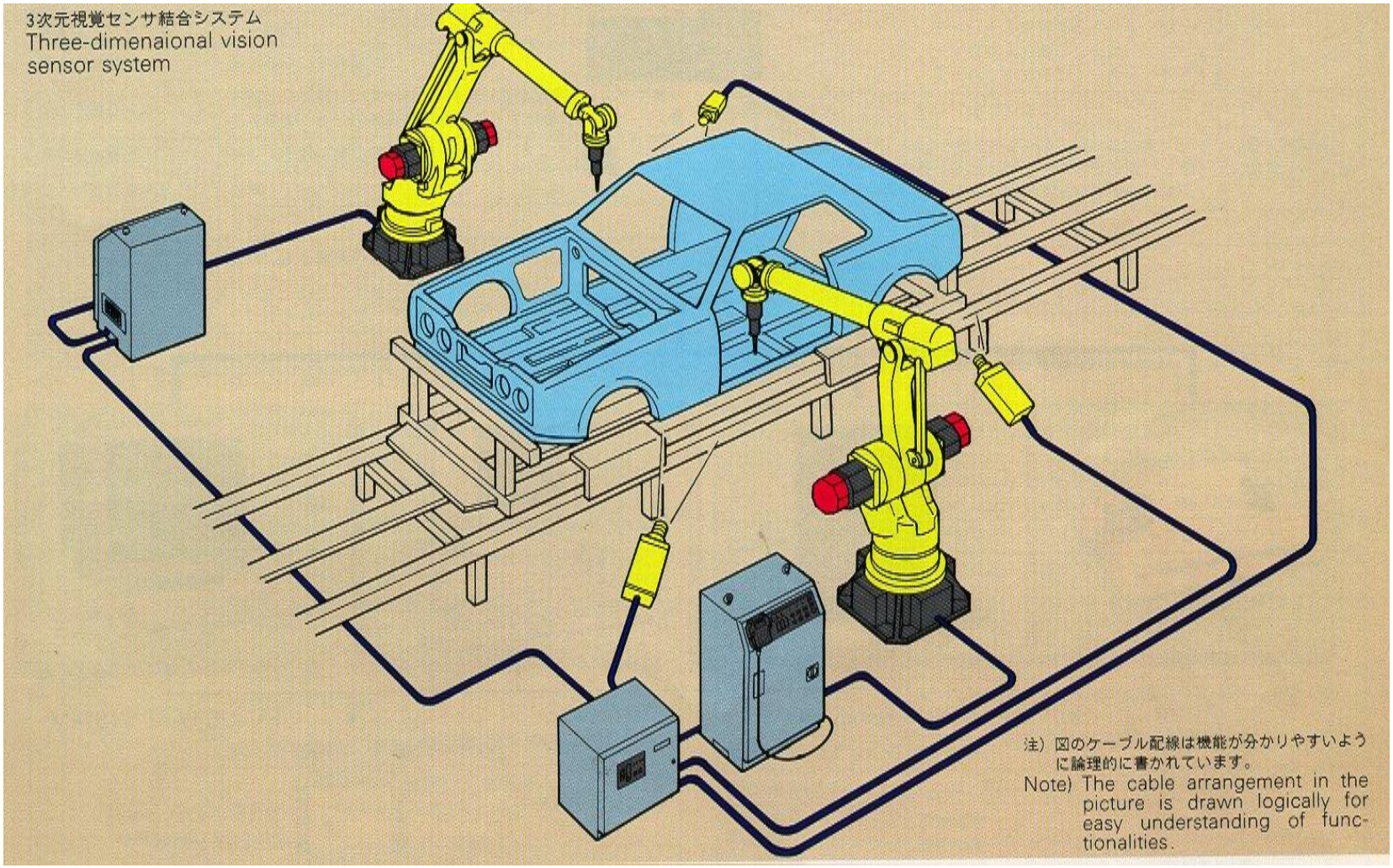
Example: Robot Systems for Handling



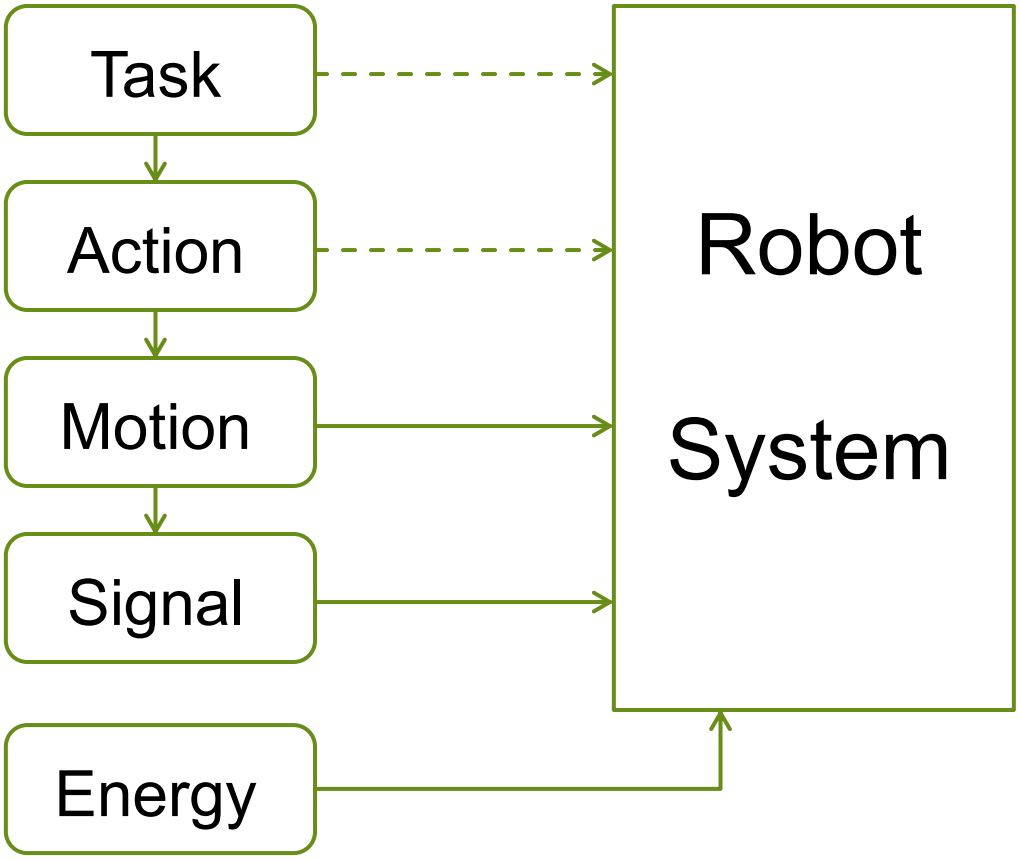
Example: Robot Systems for Palletizing



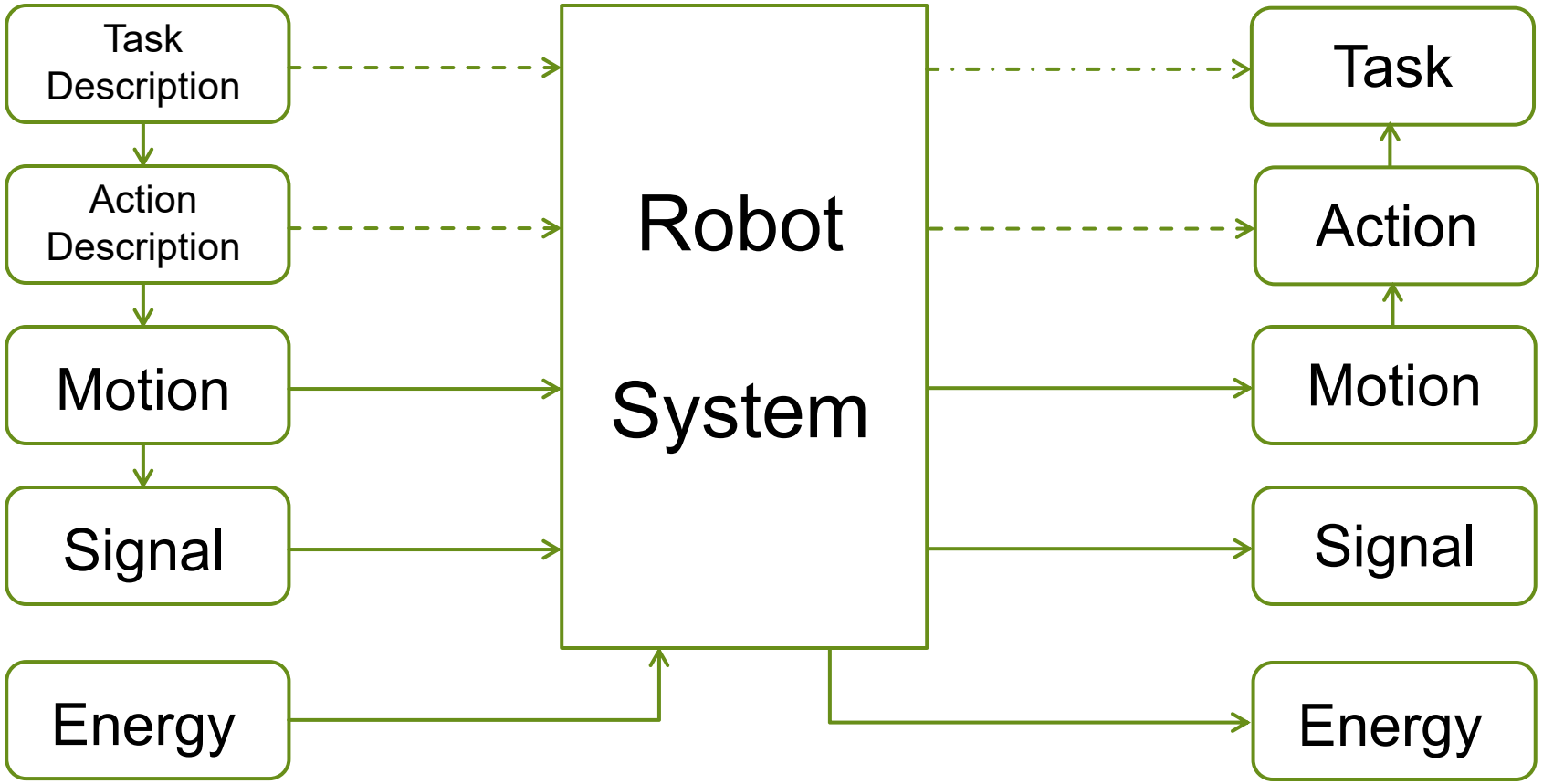
Example: Robot Systems for Welding



Inputs to a Robot's System

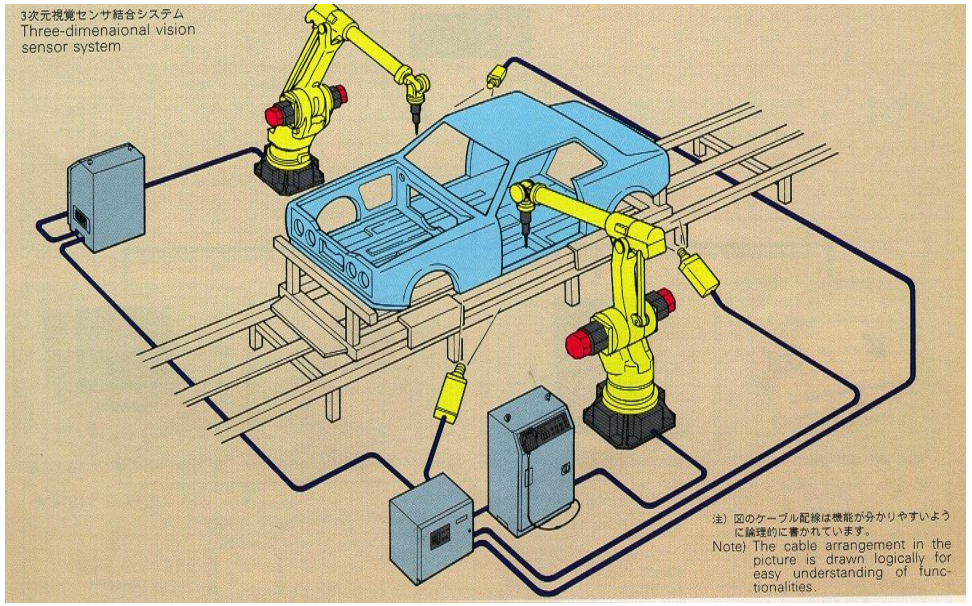


Responses from a Robot's System



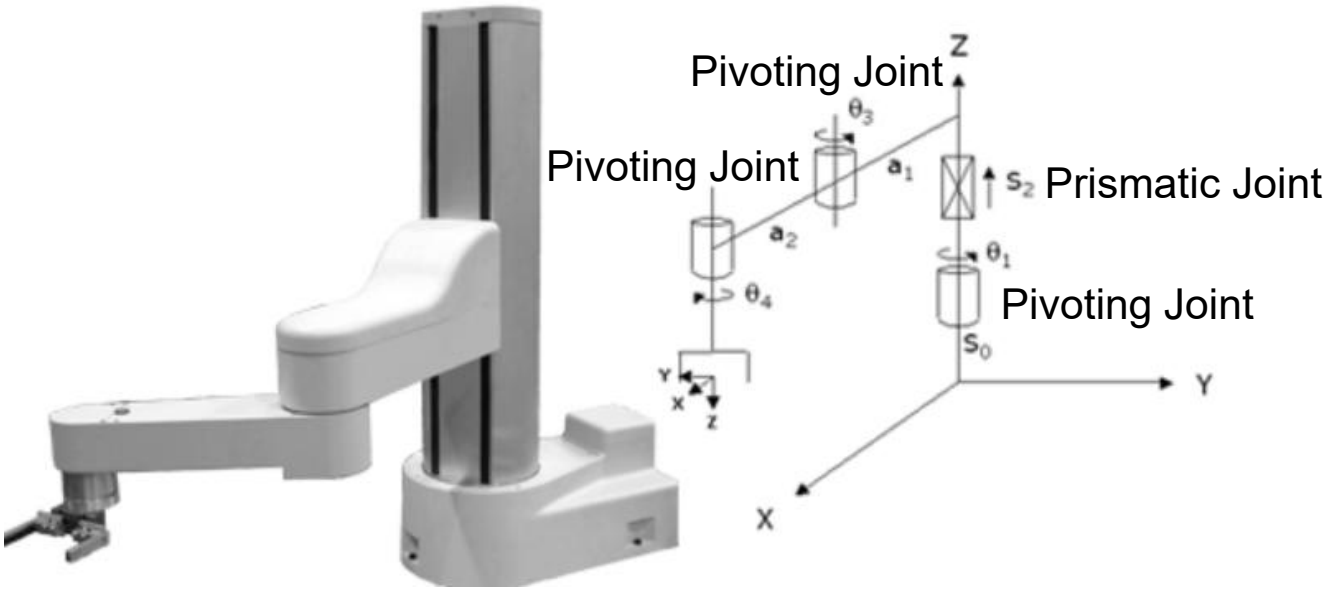
The inside of a robot's system consists of a set of subsystems such as

- Mechanical system
- Control system
- Programming system
- Power system
- Communication system
- Vision system
- Speech system
- etc



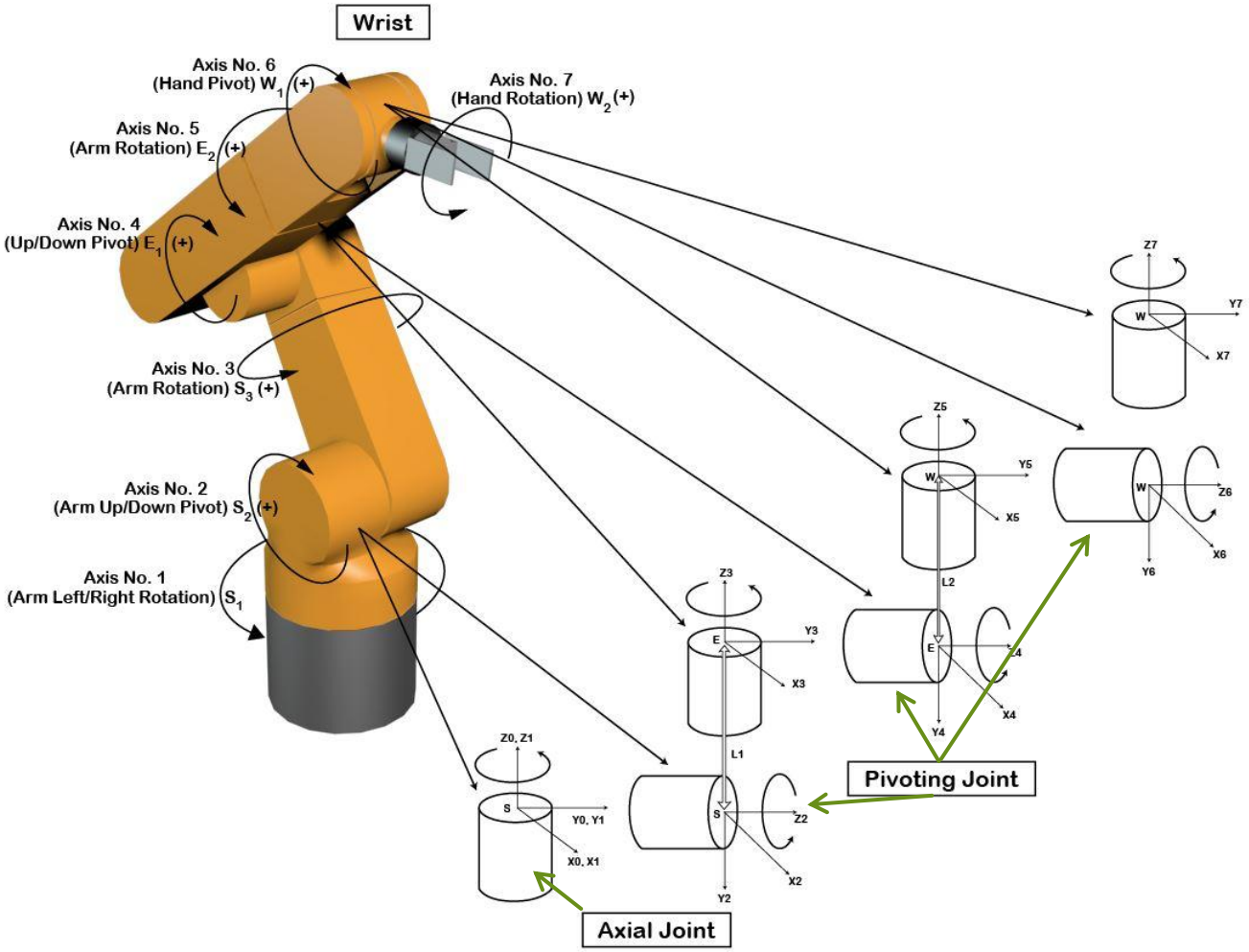
1. Mechanical System of Robot

- It consists of links and joints. A joint could be a pivoting joint, axial joint, or prismatic joint.



Example of Robot Arm with Revolute Joints

Example of Robot Arm with Revolute Joints

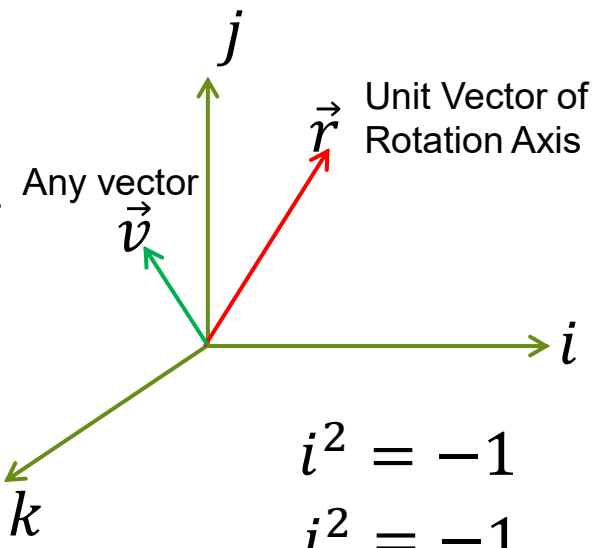


Application of Quaternion for Motion Analysis

- Euler Equation: $e^{j\omega} = \cos(\omega) + j\,sin(\omega)$

$$j = \sqrt{-1}$$

Generalized Euler Equation: Equation of Unit Quaternion



- Unit Vector of Rotation Axis:

$$\vec{r} = r_x i + r_y j + r_z k$$

$$i^2 = -1$$
$$j^2 = -1$$
$$k^2 = -1$$
$$ijk = -1$$

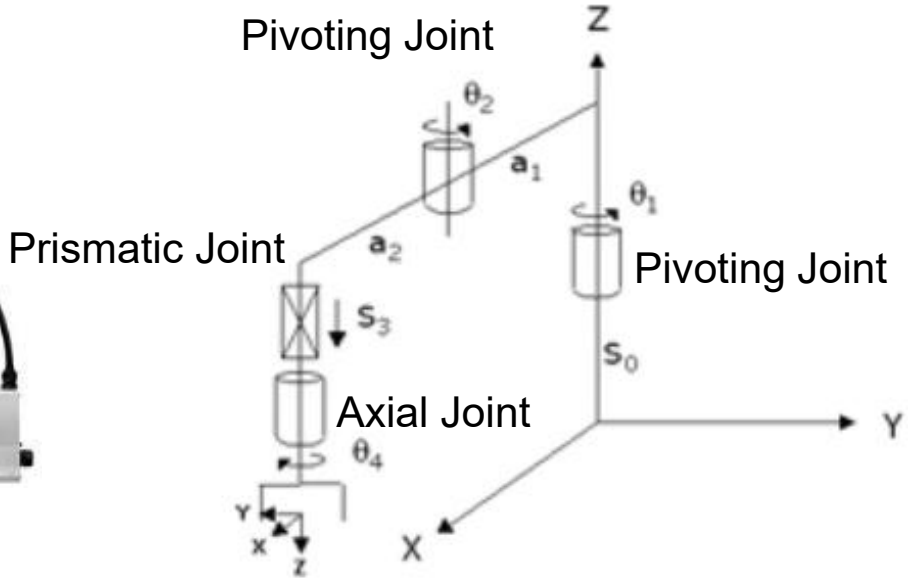
- Angle of Rotation: θ

- Hamilton Equation: $q = e^{\theta\vec{r}} = \cos(\theta) + \sin(\theta)\vec{r}$

- Equation of Rotation: $\vec{v}_{after} = q\vec{v}_{before} = [\cos(\theta) + \sin(\theta)\,\vec{r}]\vec{v}_{before}$

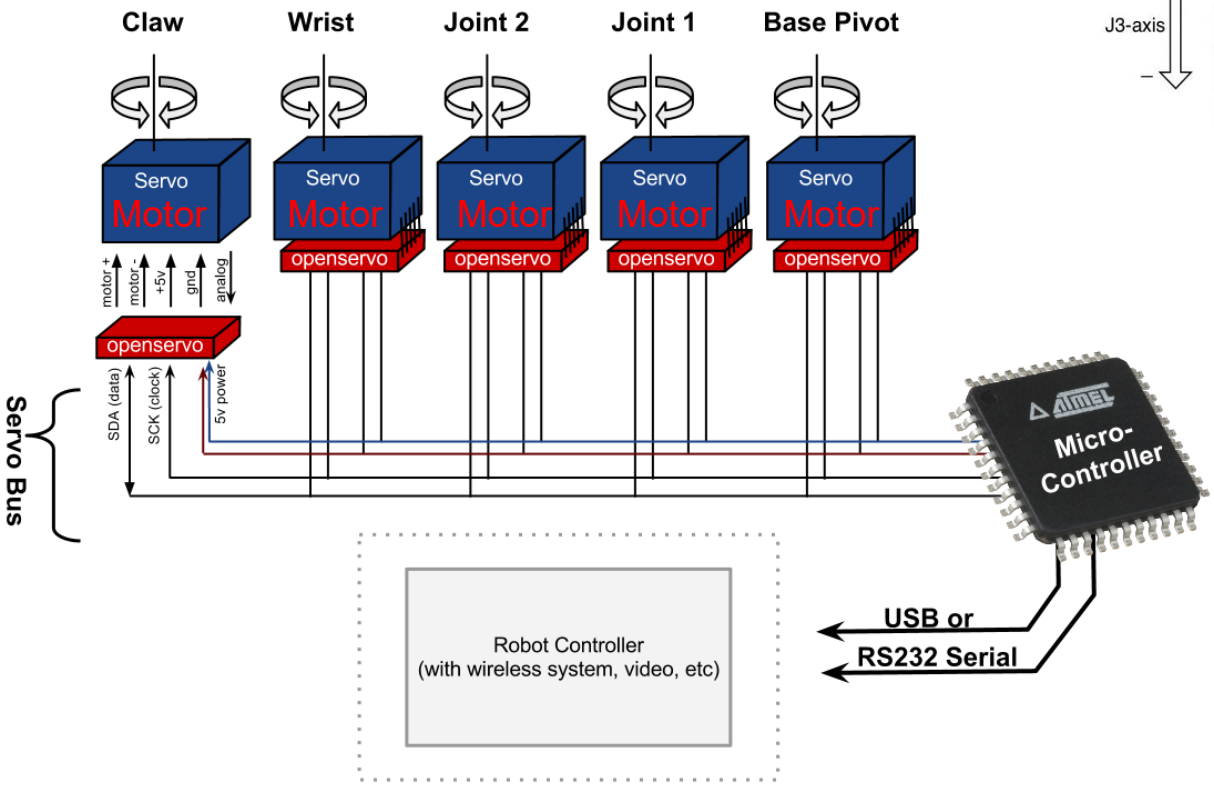
$\vec{u}\vec{v} = \vec{u} \times \vec{v} - \vec{u} \cdot \vec{v}$

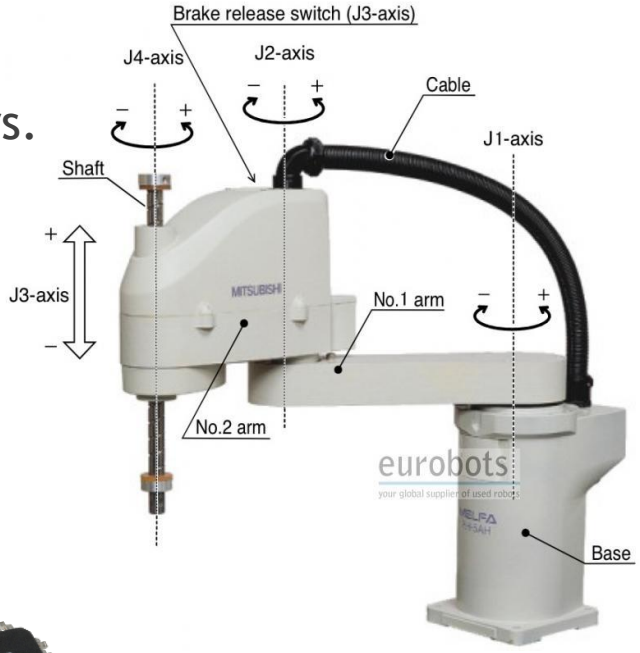
Example of SCARA Robot



2. Control System of Robot

- It consists of actuators, sensors, and controllers.



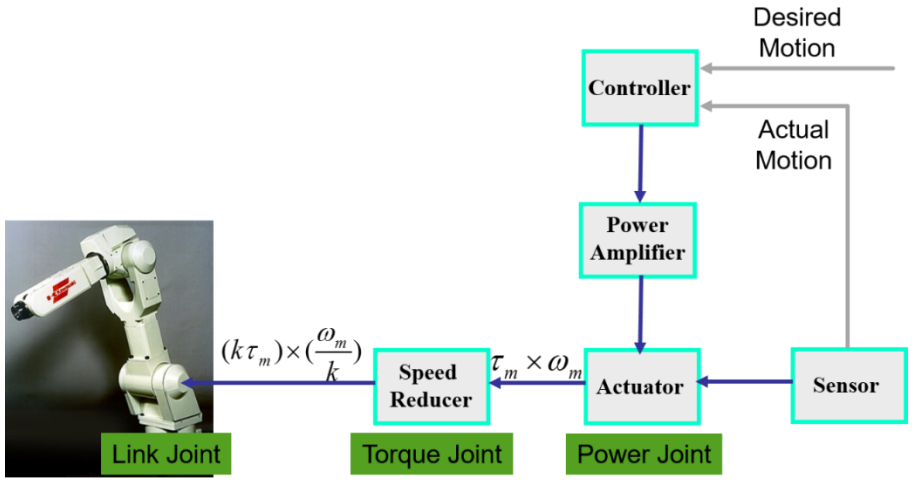


3. Programming System of Robot

- It consists of programmable brain, software tools and teachable mind.

Outline of Lecture 1

- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems

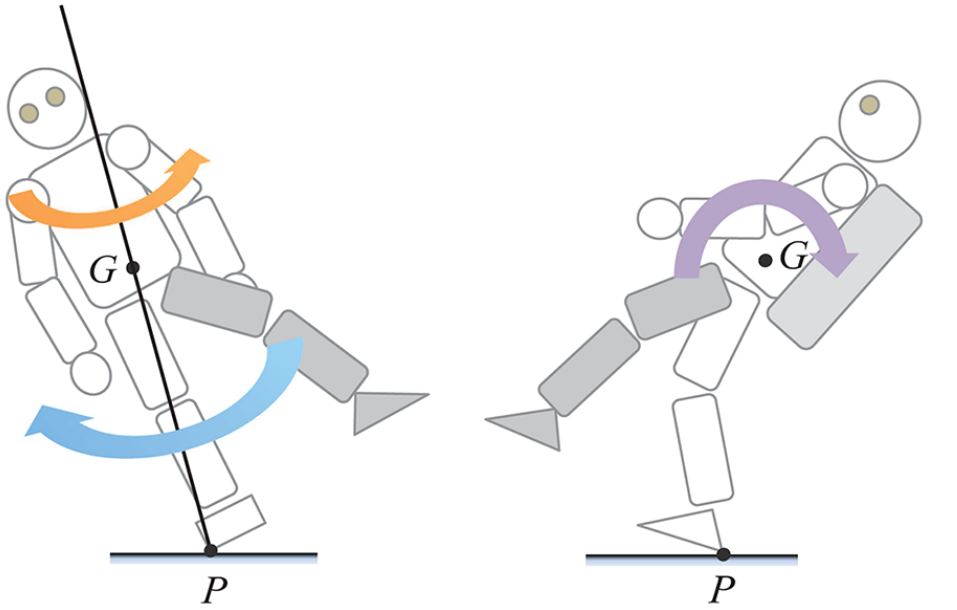


Three Criteria for Judging Performance

- Stability
- Response Time
- Response Accuracy

1. Stability of Robot Systems

- A robot is stable as long as it maintains its position and orientation in the presence of external forces or induced forces.



(a) Spinning with swing leg (b) Bending the trunk backward

Example of Robots with Weak Stability

Example

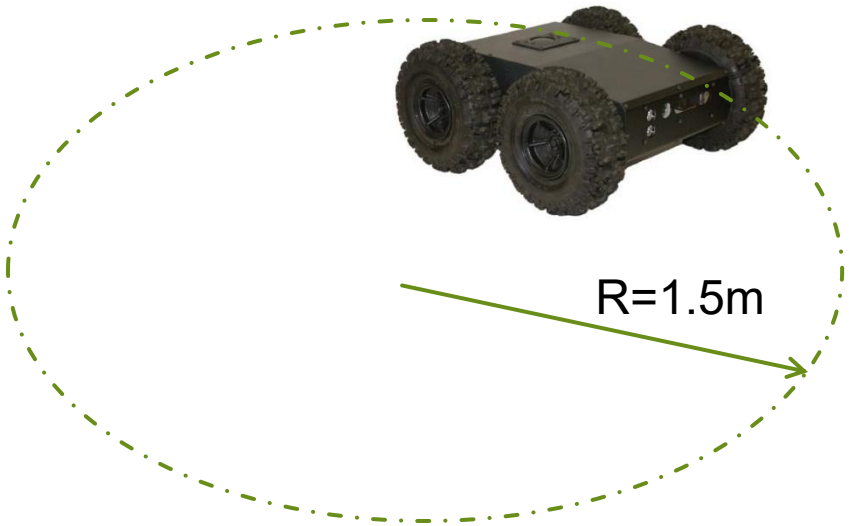
- A humanoid robot kicks off a ball and stands with the support of a single leg. What is the condition of stability in this case?
- Answer:
- The robot is stable if the projection of its centre of gravity (CG) onto the ground is within the area of the supporting leg's foot.

Example

- A mobile robot's mass is 20.0 kg. Its centre of gravity is located at the centre of the robot's body and is 0.3 meters above the ground. The distance between its left wheels and right wheels is 0.5 meters. The robot moves along a circular path. And, the radius of the circle followed by the outer wheel is 1.5meters. What is the maximum circular speed that the robot could reach before it becomes unstable?
- Answer:

$$F_c \times 0.3 = m\frac{v^2}{1.5} \times 0.3 < mg \times 0.25$$

$$v < \sqrt{\frac{0.25 \times 1.5 \times 9.8}{0.3}}\ \text{m/s}$$

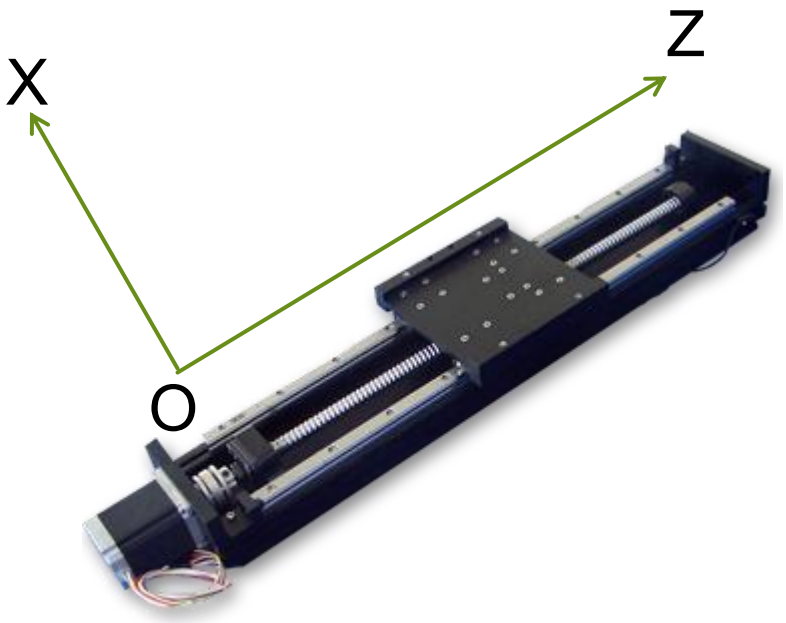


2. Response Time of Robot System

- It is the time for a robot to complete a motion.

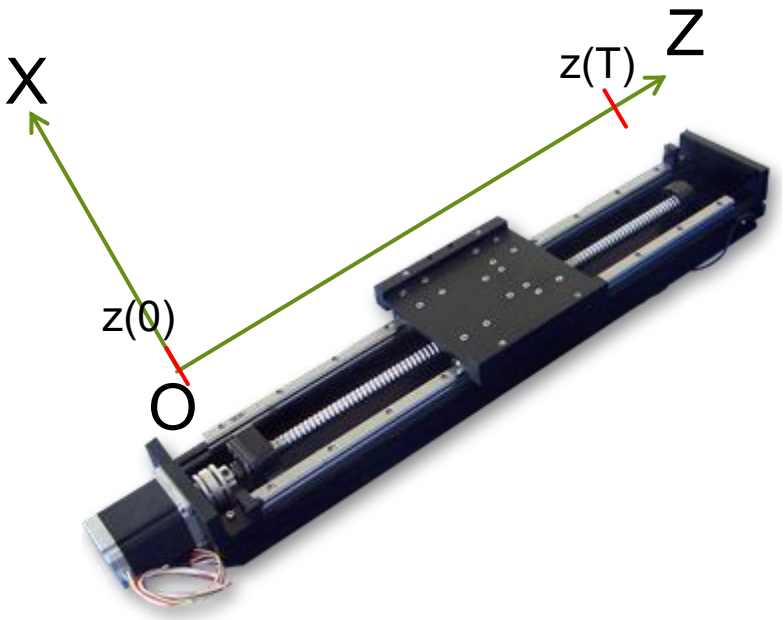
Example

- A single axis robot has the maximum acceleration of +/- 5.0 cm/s2. Before it moves, the robot is at rest. What is the shortest response time for it to travel 20.0 cm and stop?



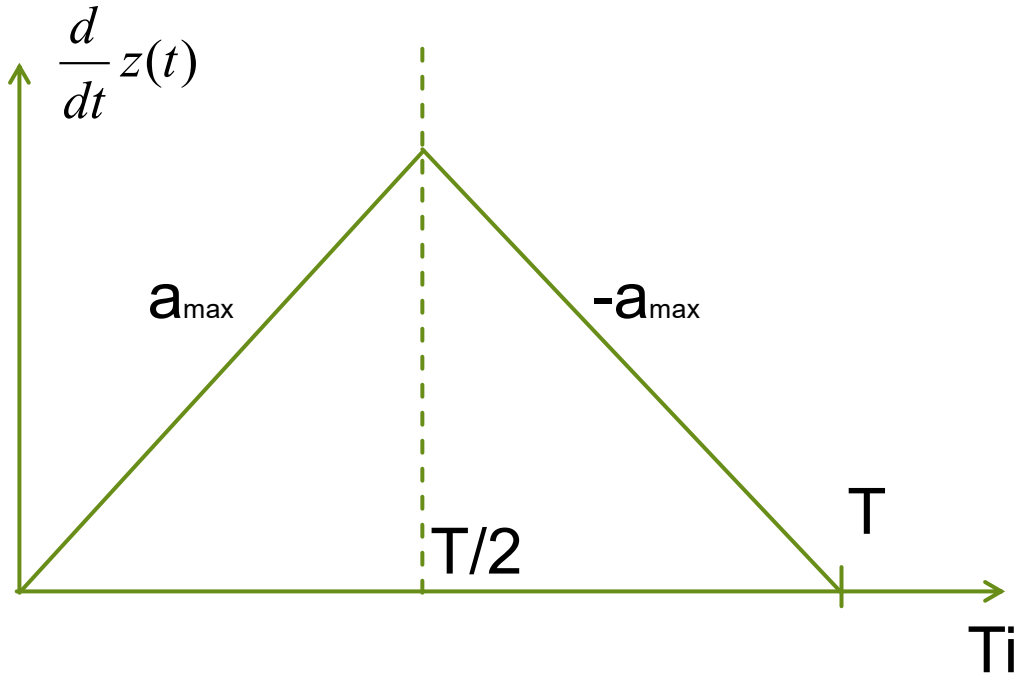
Solution

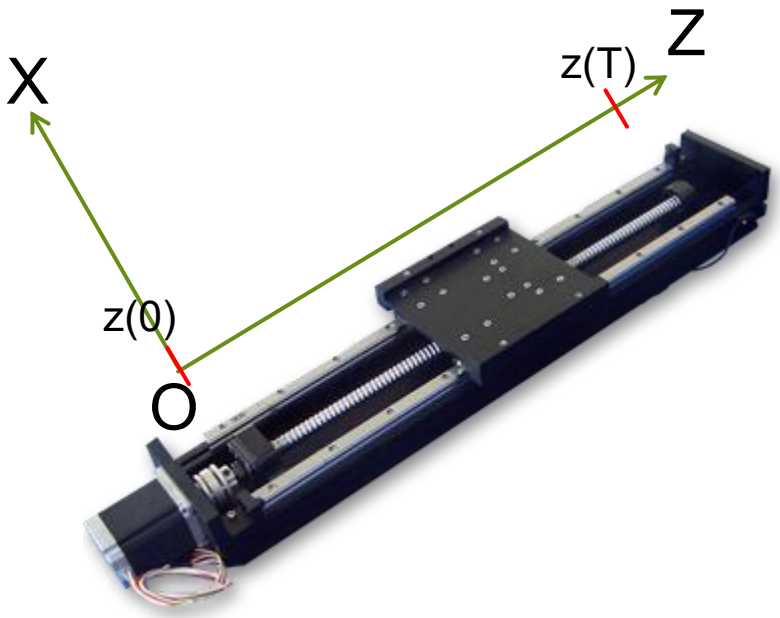
- Analysis:
  - Place X axis at the position before motion: z(0) = 0.0cm
  - Maximum acceleration: (+/-) 5.0 cm/s2
  - Travelled distance: z(T) = 20.0cm
  - To determine the shortest response time: T



Solution

- Step 1: Design of Trajectory of Shortest Response Time
  - Speed-up Motion: from 0.0 to T/2 (seconds)
  - Slow-down Motion: from T/2 to T (seconds)



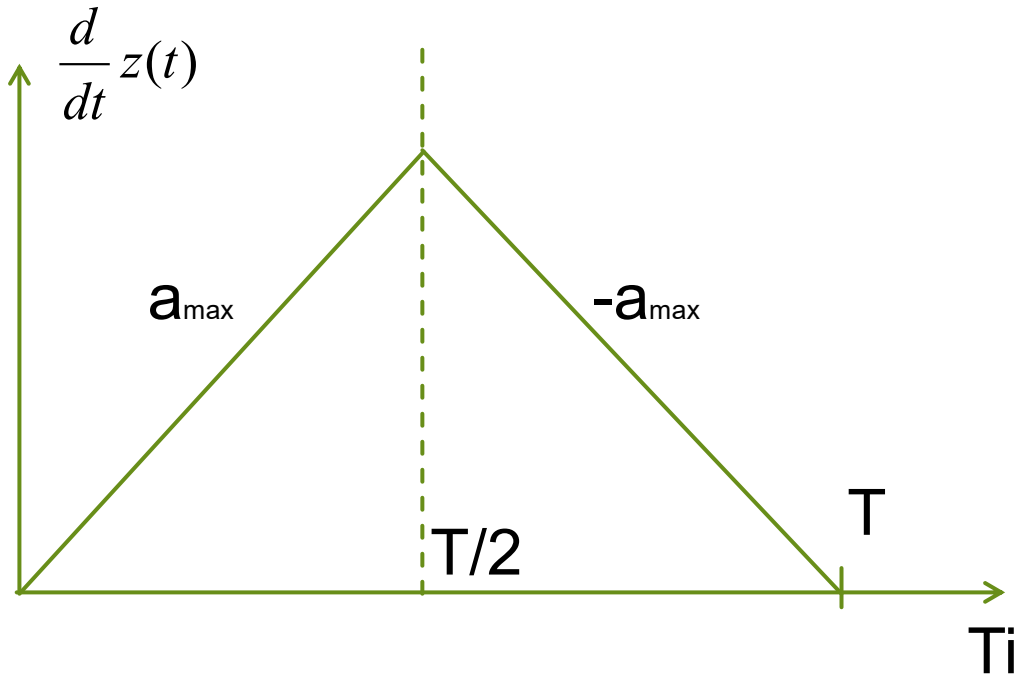


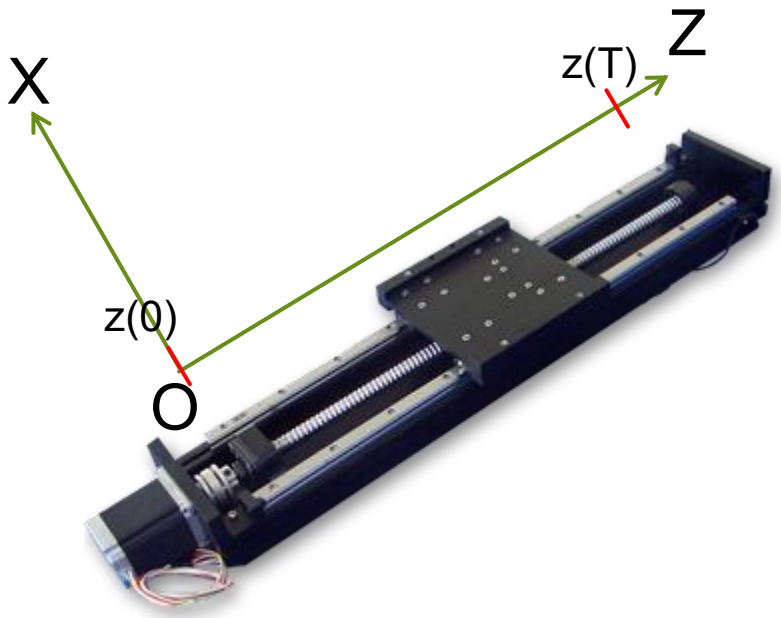
Solution (continued)

- Step 2: Trajectory for speed-up motion: [0, T/2]

$$z(t) - z(0.0) = \frac{1}{2} a_{\max} t^2$$

$$v(t) = v(0.0) + a_{\max} t = a_{\max} t$$





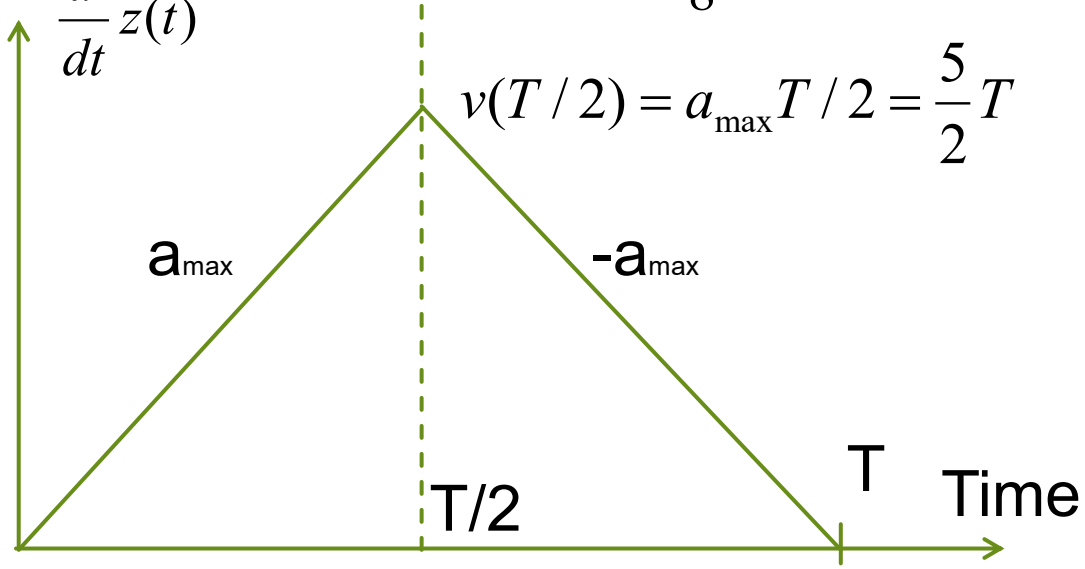
Solution (continued)

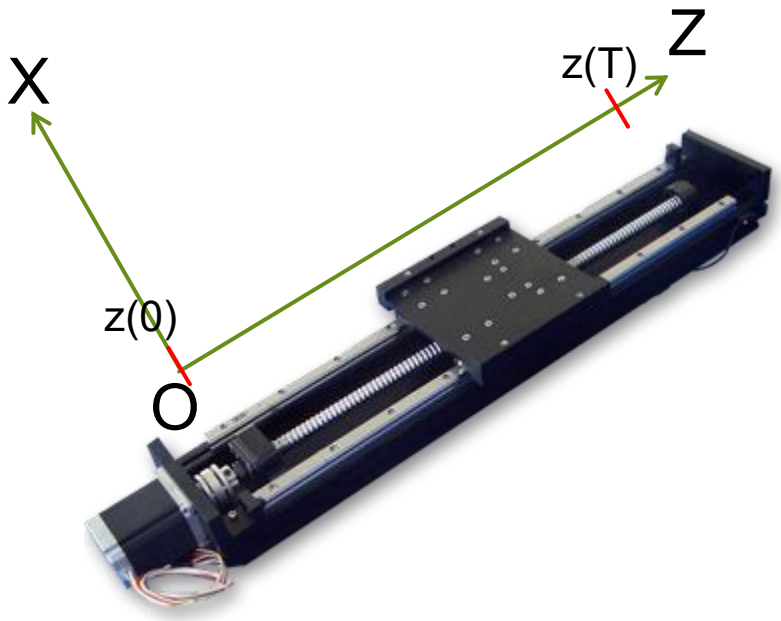
- Step 3: Trajectory for slow-down motion: [T/2, T]

$$z(t) - z(T/2) = v(T/2)(t - T/2) - \frac{1}{2} a_{\max} (t - T/2)^2$$

$$z(T/2) = \frac{5}{8} T^2$$

$$v(T/2) = a_{\max} T/2 = \frac{5}{2} T$$

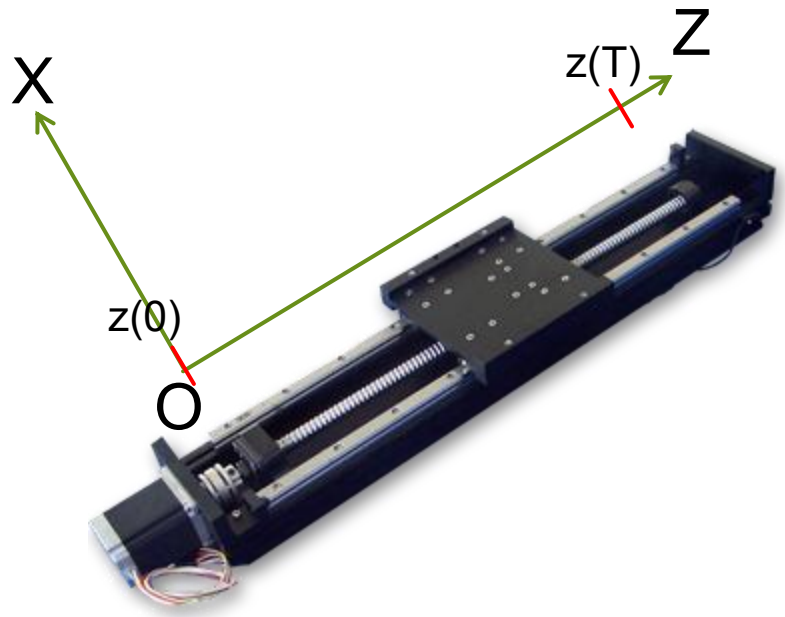




$$z(t)-z(T/2)=v(T/2)(t-T/2)-\frac{1}{2}a_{\max}(t-T/2)^2$$

$$z(T/2)=\frac{5}{8}T^2$$

$$v(T/2)=a_{\max}T/2=\frac{5}{2}T$$



$$z(t)-\frac{5}{8}T^2=\frac{5T}{2}t-\frac{5}{4}T^2-\frac{5}{2}(t-T/2)^2$$

Solution

- Step 4: Shortest Response Time

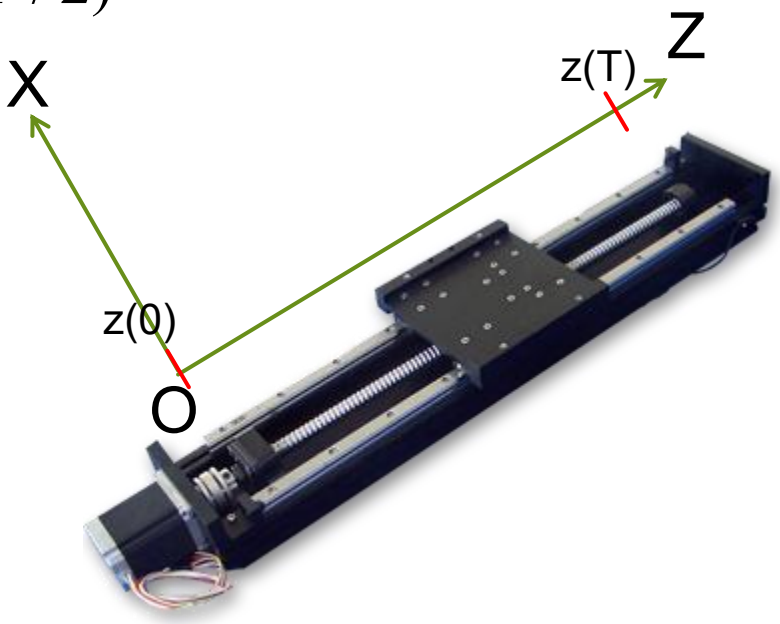
$$z(t)-\frac{5}{8}T^2=\frac{5T}{2}t-\frac{5}{4}T^2-\frac{5}{2}(t-T/2)^2$$

$$z(T)-\frac{5}{8}T^2=\frac{5}{2}T^2-\frac{5}{4}T^2-\frac{5}{2}(T-T/2)^2$$

$$20-\frac{5}{8}T^2=\frac{5}{4}T^2-\frac{5}{8}T^2$$

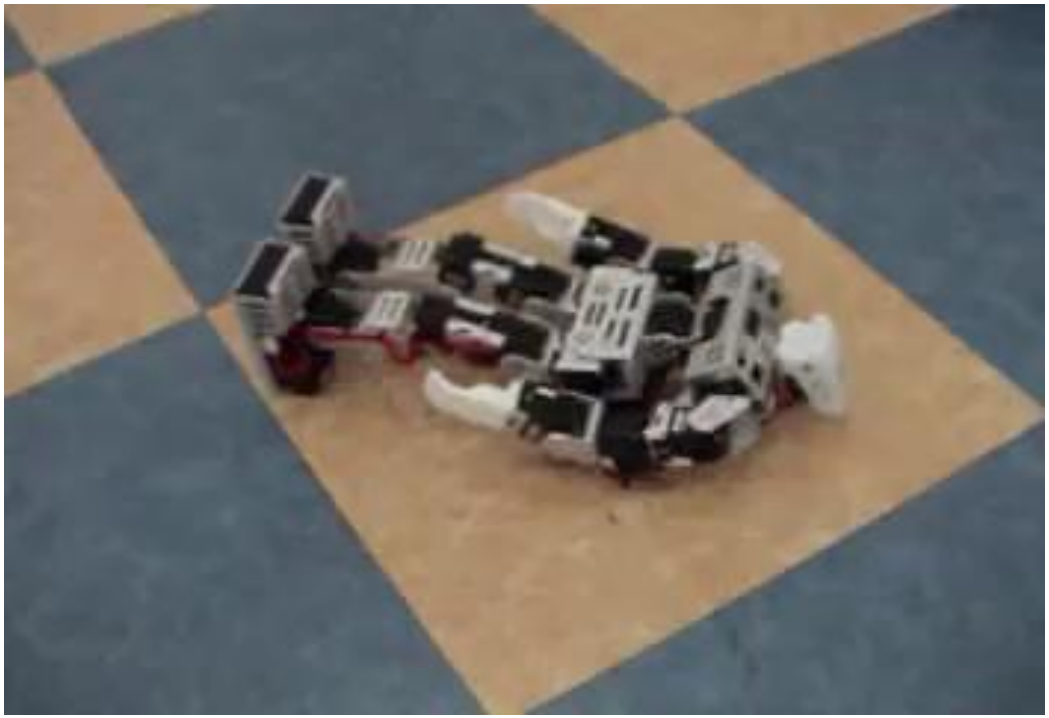
$$T^2=16$$

$$T=4\ \mathrm{s}$$



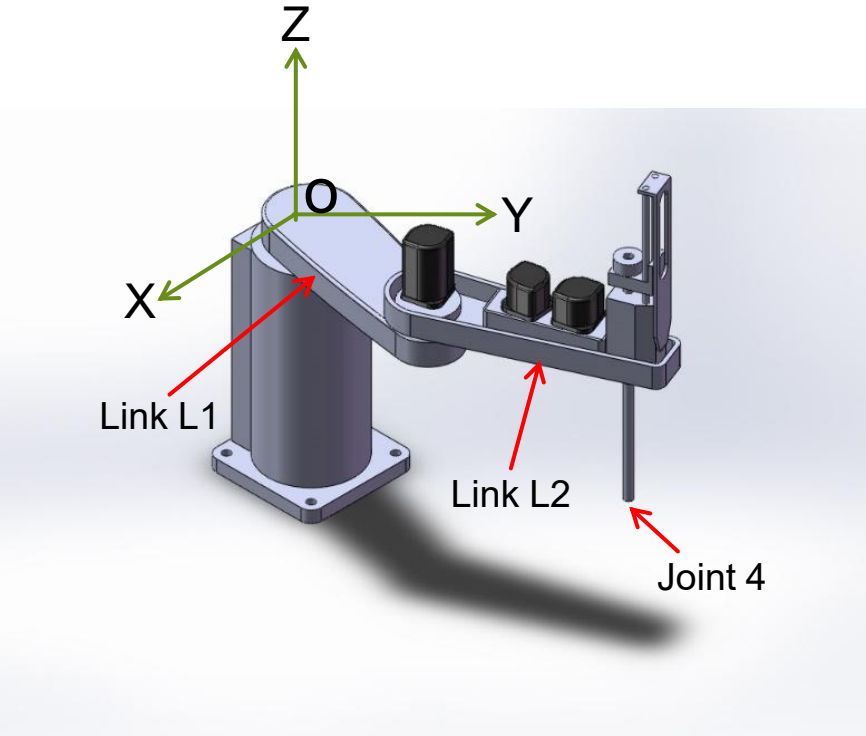
3. Response Accuracy of Robot System

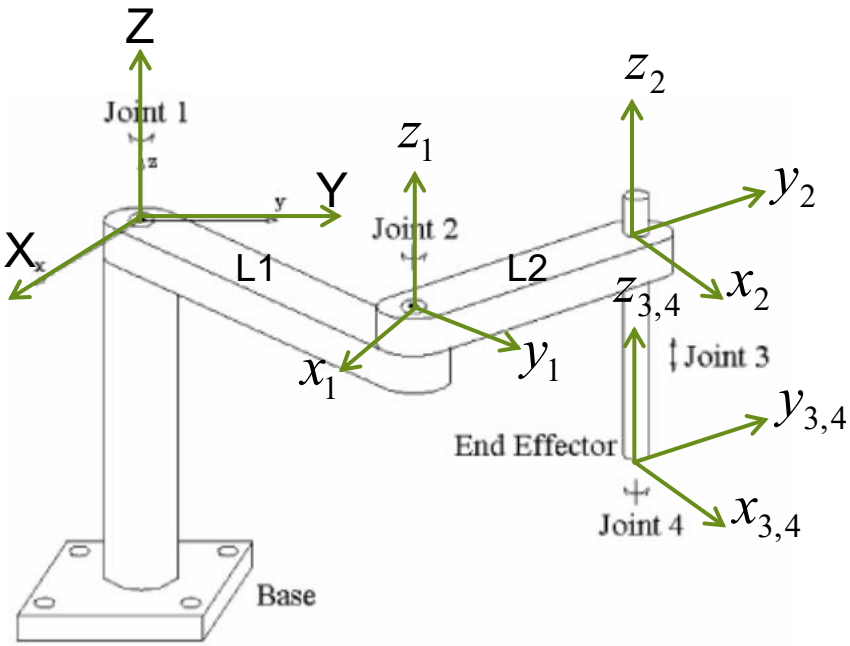
- It is the maximum error when a robot repeatedly performs a same motion.



Example

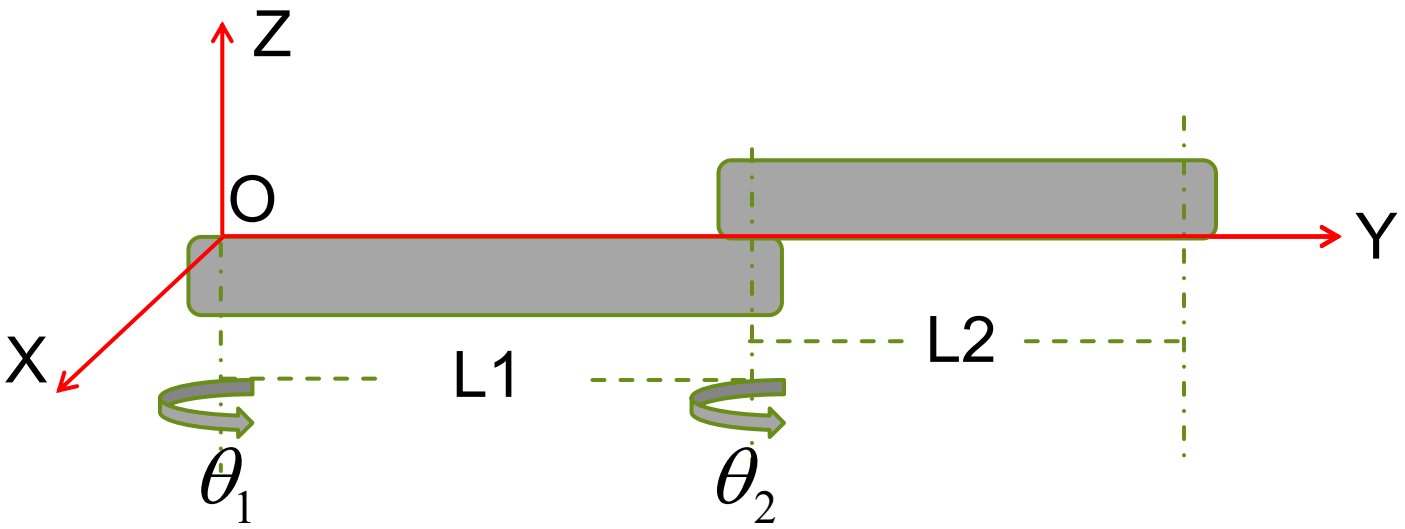
- A SCARA robot has the following geometrical parameters: L1 = 30.0 (cm) and L2 = 25.0 (cm). The angular accuracy of joint 1 is $\pm$ 0.01 degrees, and the accuracy of joint 2 is $\pm$ 0.02 degrees. What is the accuracy of Y coordinate of joint 4 with respect to the robot's base?





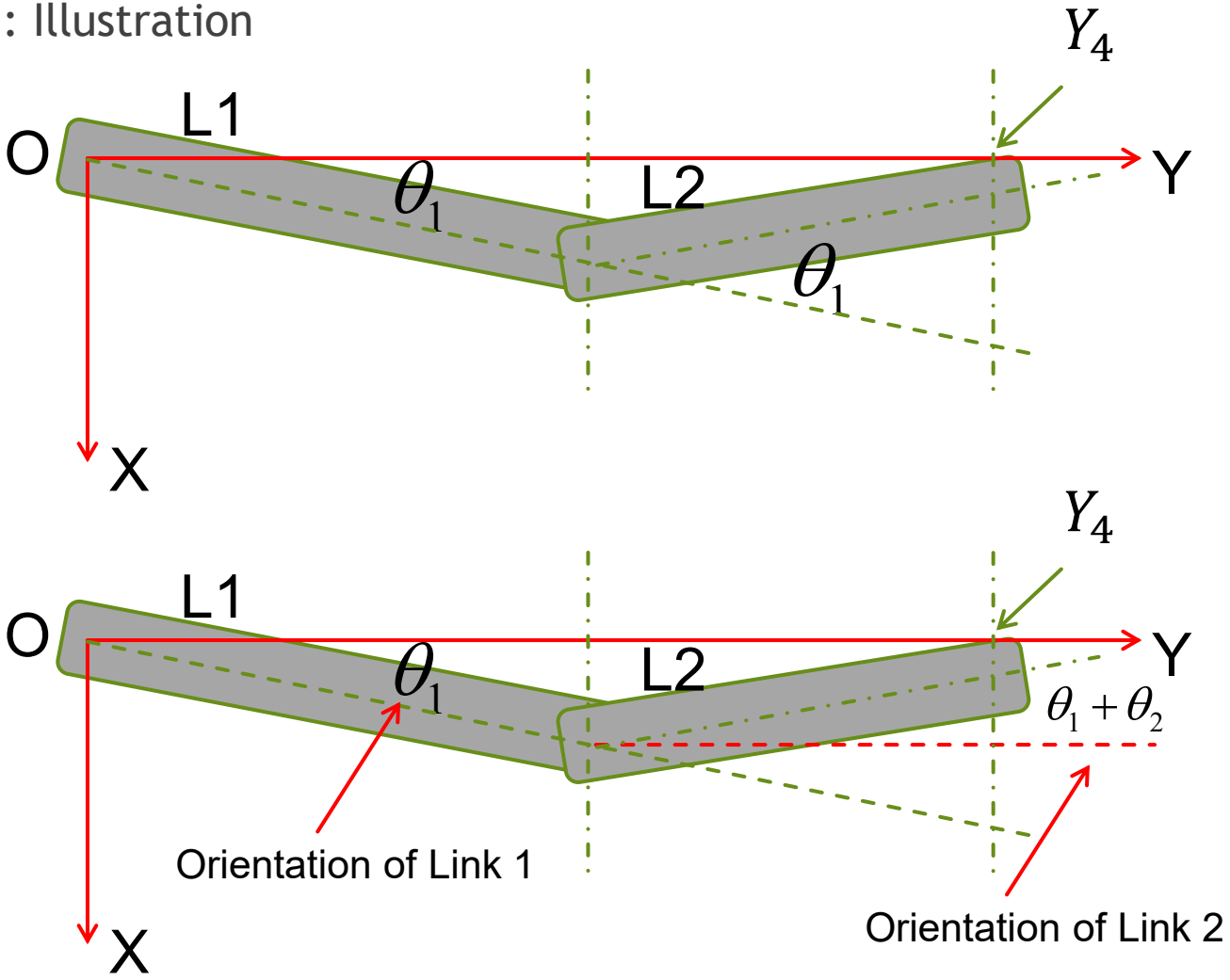
Solution

- Analysis
  - The error of joint 4's Y coordinate will be the largest one when joint 1's angle is 90 degrees while joint 2's angle is zero degree.
  - The error of joint 4's Y coordinate is a function of angles of joint 1 and joint 2.



Solution (continued)

- Step 1: Illustration

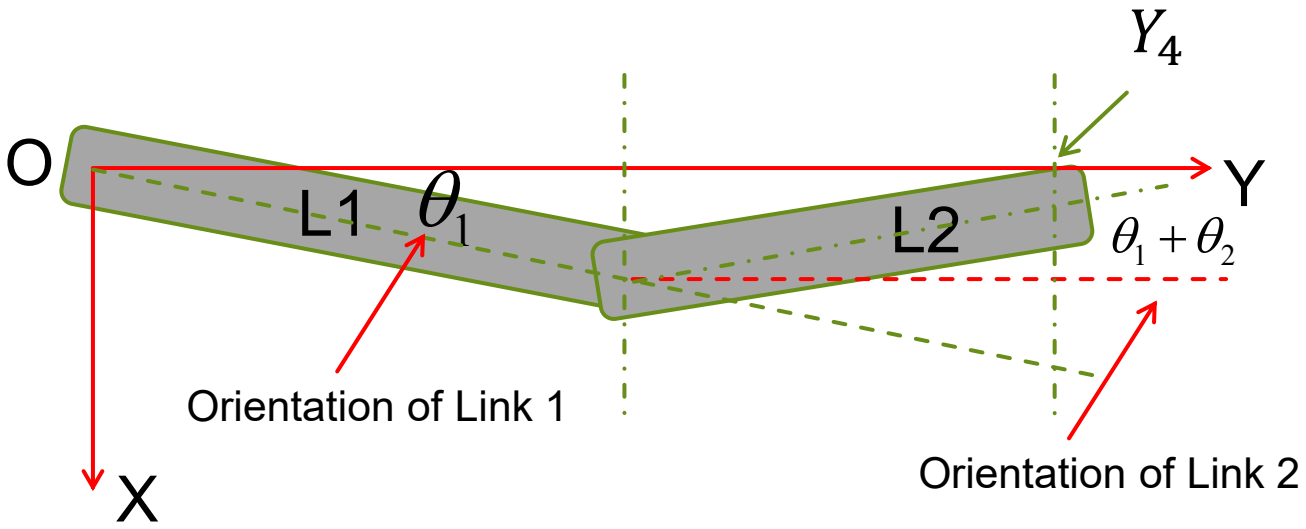


Solution (continued)

- Step 2: Equation of Coordinate

$$y_4 = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$\frac{dy_4}{dt} = -L_1 \sin(\theta_1)\frac{d\theta_1}{dt} - L_2 \sin(\theta_1 + \theta_2)\left(\frac{d\theta_1}{dt} + \frac{d\theta_2}{dt}\right)$$

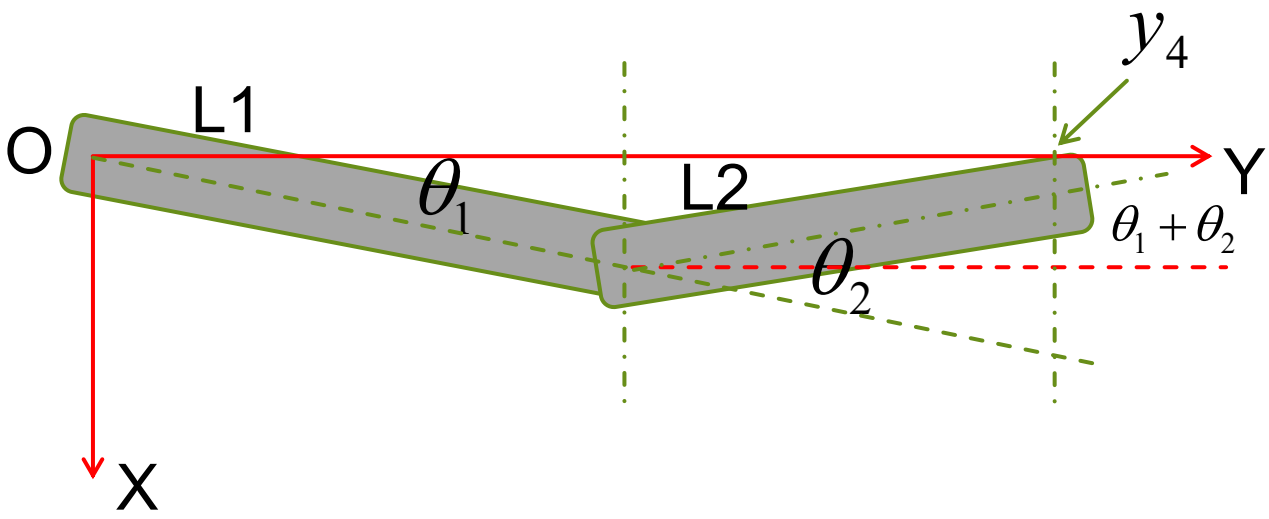


Solution (continued)

- Step 3: Equation of Errors

$$\frac{dy_4}{dt} = -L_1 \sin(\theta_1)\frac{d\theta_1}{dt} - L_2 \sin(\theta_1 + \theta_2)(\frac{d\theta_1}{dt} + \frac{d\theta_2}{dt})$$

$$\Delta y_4 = -L_1 \sin(\theta_1)\Delta\theta_1 - L_2 \sin(\theta_1 + \theta_2)(\Delta\theta_1 + \Delta\theta_2)$$



Solution

- Step 2: Accuracy or Maximum Error of Joint 4's Y Coordinate

$$\Delta y_4 = -L_1 \sin(\theta_1)\Delta\theta_1 - L_2 \sin(\theta_1 + \theta_2)(\Delta\theta_1 + \Delta\theta_2)$$

Worse Case: joint 1's angle is ± 90 degrees while joint 2's angle is zero degree

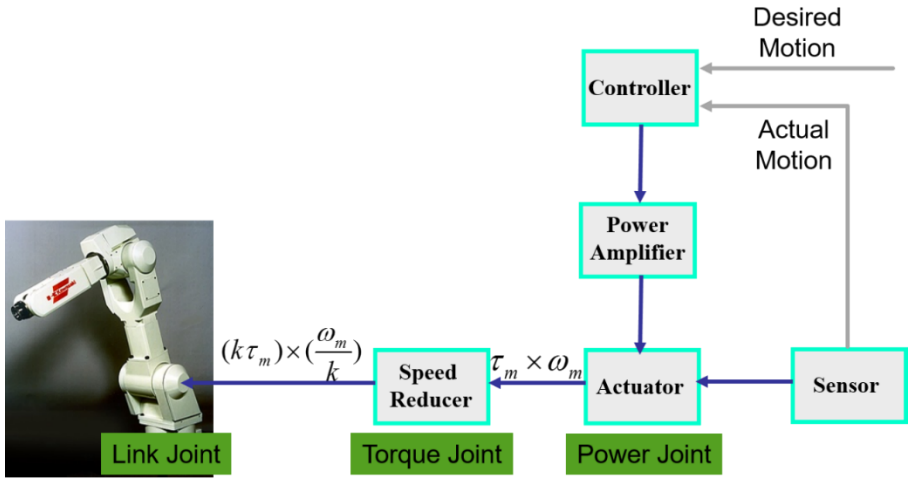
$$\Delta y_4 = \pm L_1 \Delta\theta_1 \pm L_2 (\Delta\theta_1 + \Delta\theta_2)$$

$$\Delta y_4 = \pm 30.0 \times 0.01 \frac{\pi}{180} \pm 25.0 \times 0.03 \frac{\pi}{180}$$

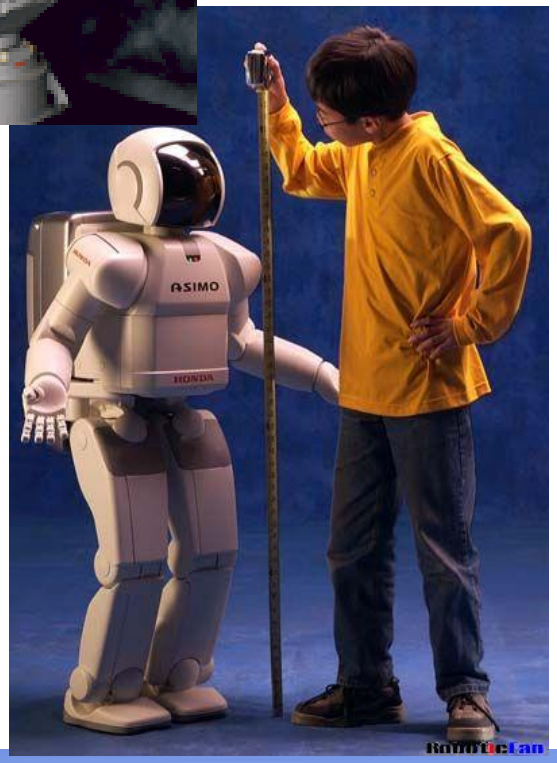
$$\Delta y_4 = \pm 0.0018\ (cm)$$

Outline of Lecture 1

- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems



Design Procedure



Users or Needs

↓

Product Functionalities

↓

Design Specifications

↓

Conceptual Design

↓

Selection of Materials/Components/Devices

↓

Embodiment Design

↓

Prototyping

↓

Optimizing

↓

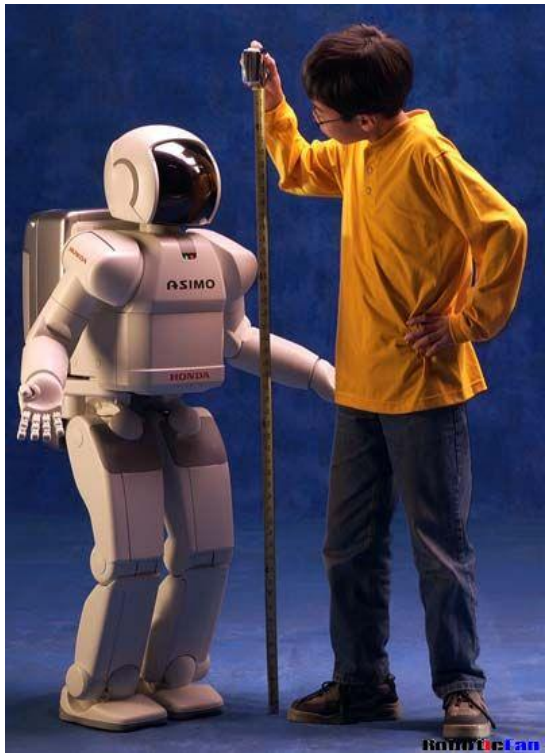
Production

↓

Marketing

Design of Robot's Mechanical System

- Design of Appearance
- Design of Mechanism
- Design of Structure



Design of Robot's Control System

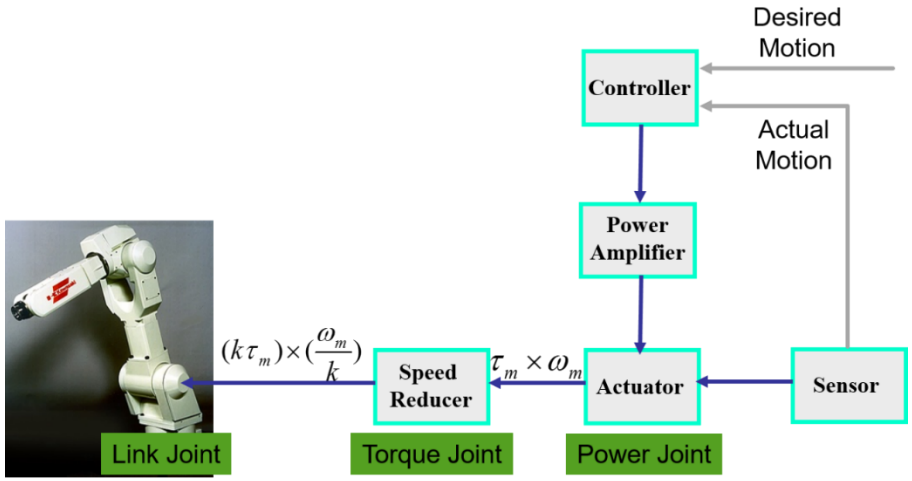
- Design of Input Devices
- Design of Output Devices
- Design of Control Devices

Design of Robot's Programming System

- Design of Programmable Brain
- Design of Software Tools
- Design of Teachable Mind

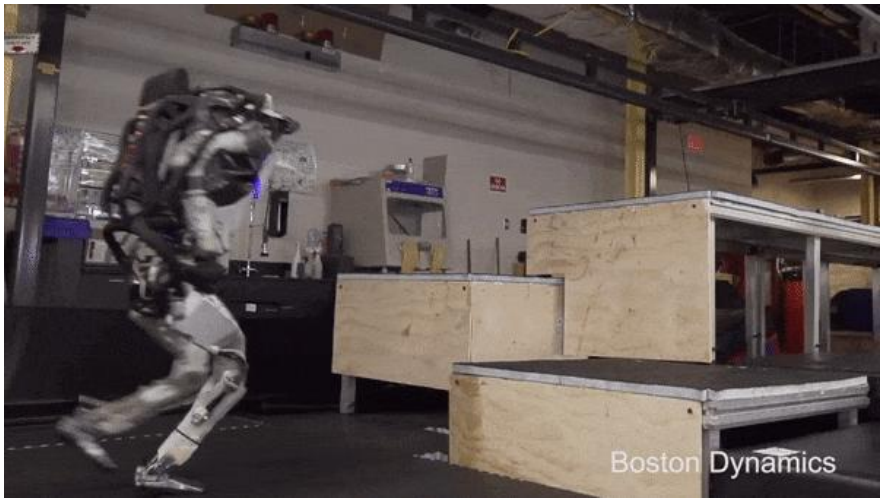
Summary of Lecture 1

- Systems
- Robot Systems
- Performance of Robot Systems
- Design of Robot Systems



Outline of Module 1

- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

Lecture 2

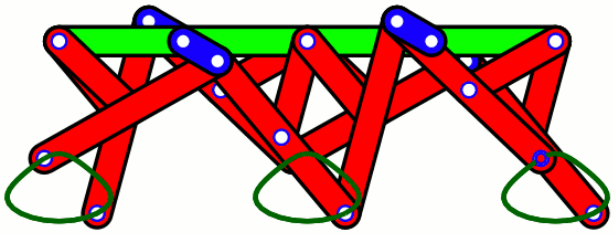
Robot's Mechanical Systems

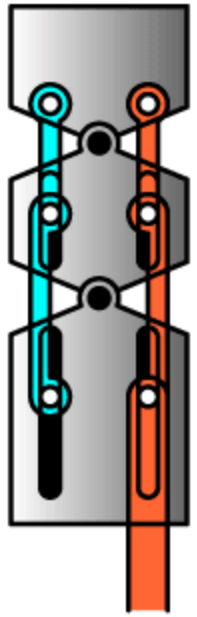
Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

Outline of Lecture 2

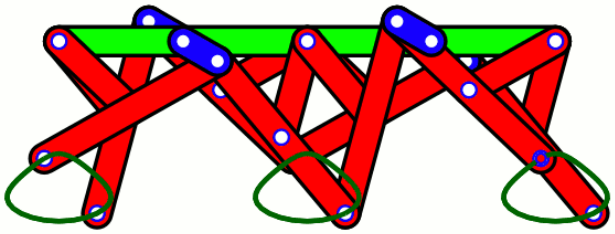
- Design of Link
- Design of Joint
- Design of Mechanism

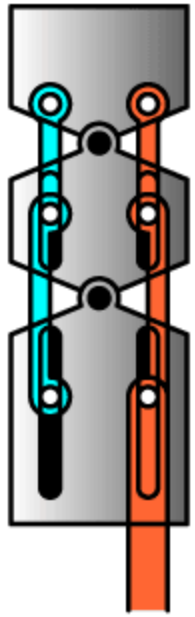




Outline of Lecture 2

- Design of Link
- Design of Joint
- Design of Mechanism



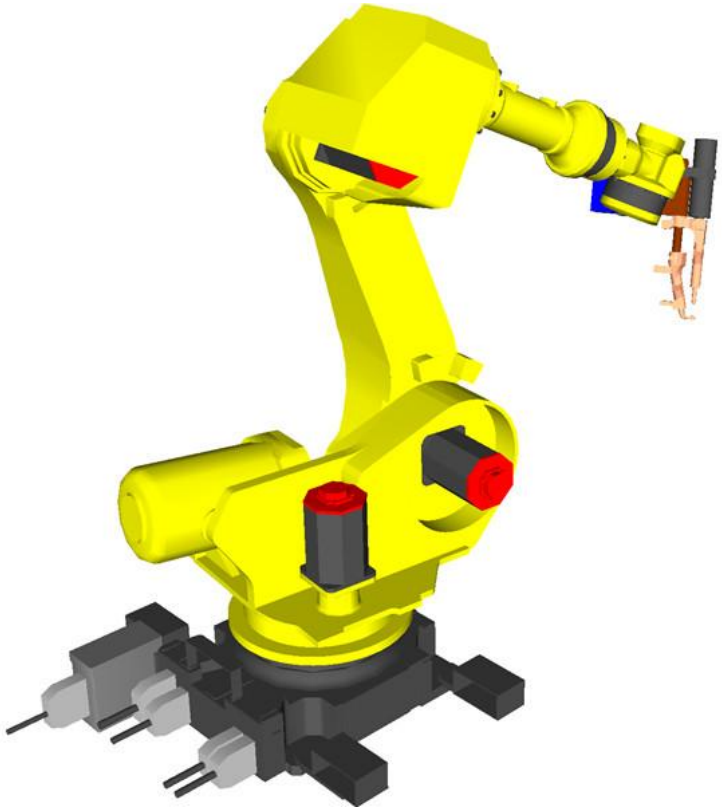


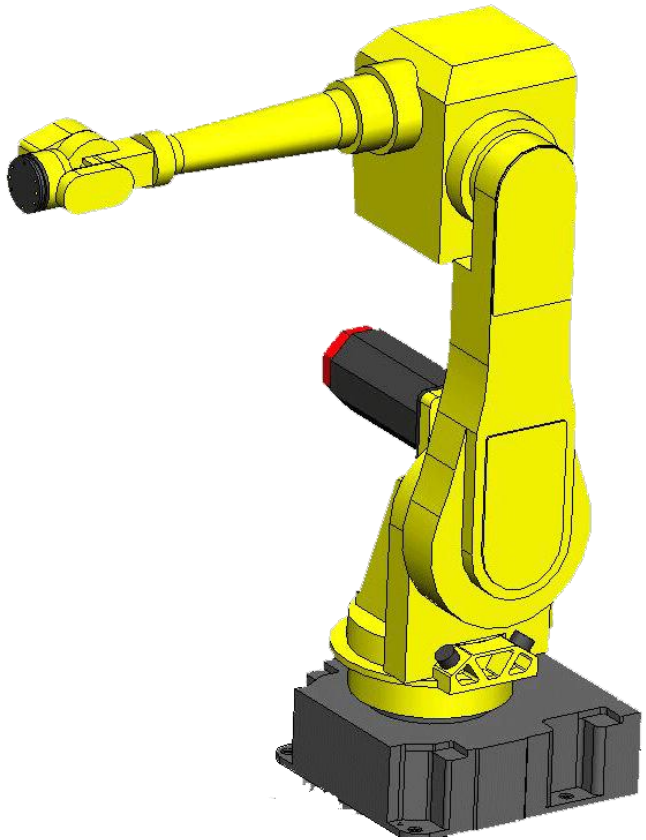
Definition of Links

- A link is a rigid body which has two extremities and is for the purpose of:
  - Connecting to other link at each extremity.
  - Housing components such as actuators, sensors, speed-reducers, motion transmission devices, cables, and appearance panels, etc.
  - Connecting to tools or end-effectors.
  - Performing motions such as translational motions or rotational motions.

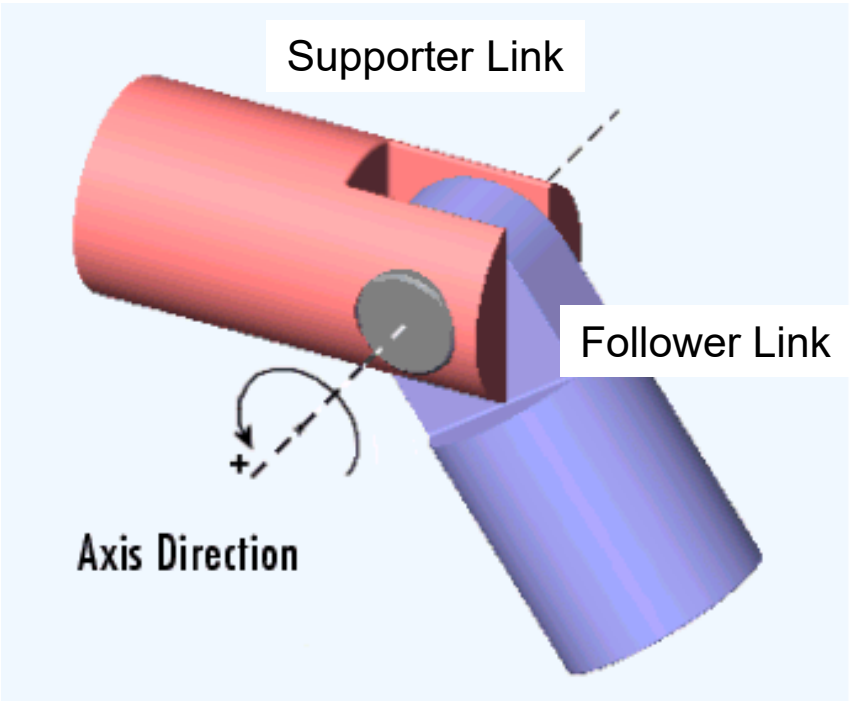
Design Example of Links

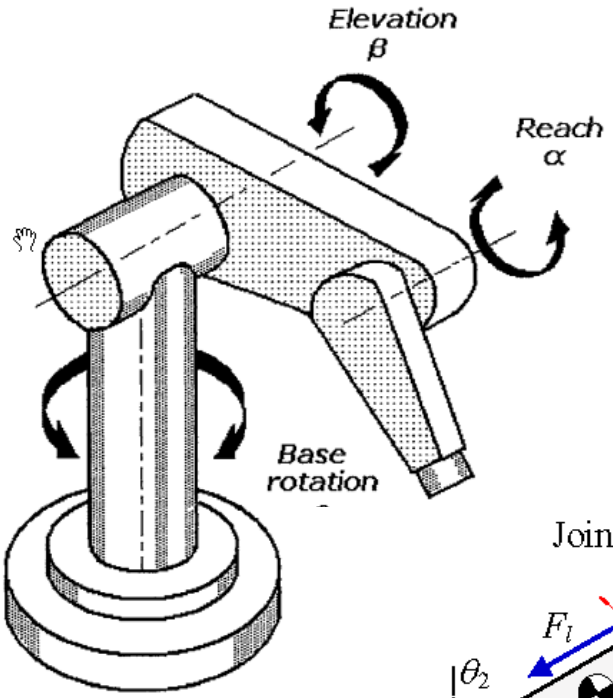
- There is no specific guideline.

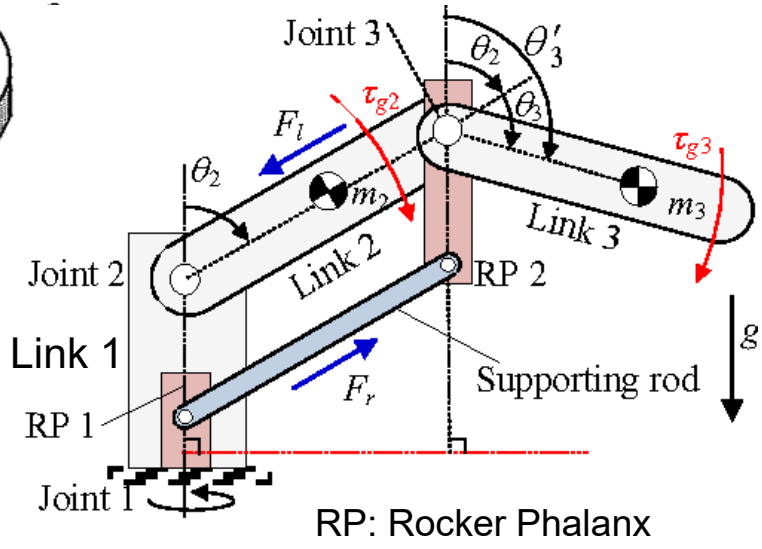




Design Example of Rotational Links

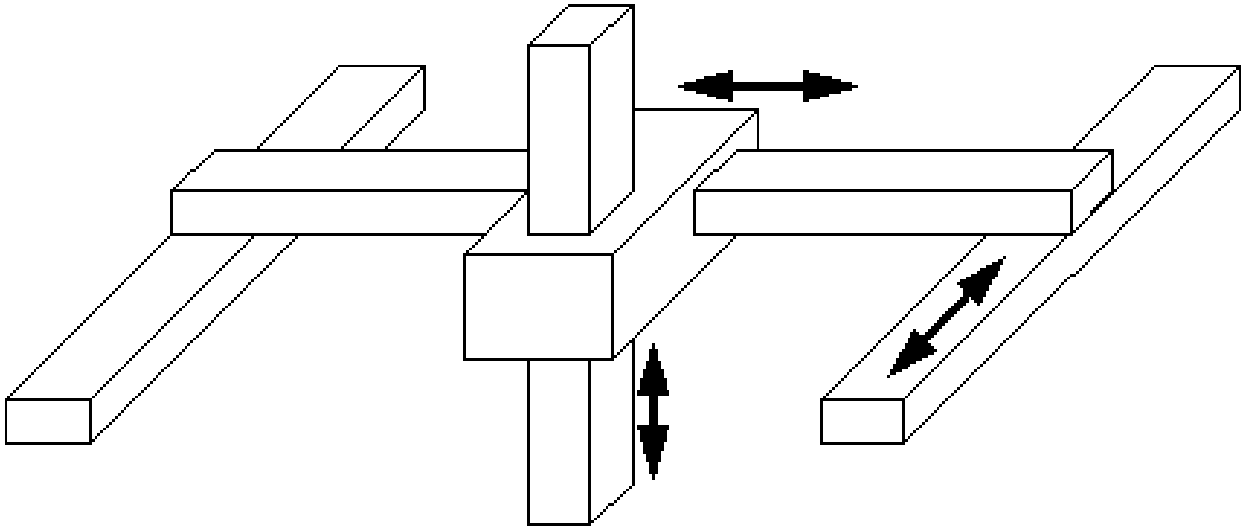




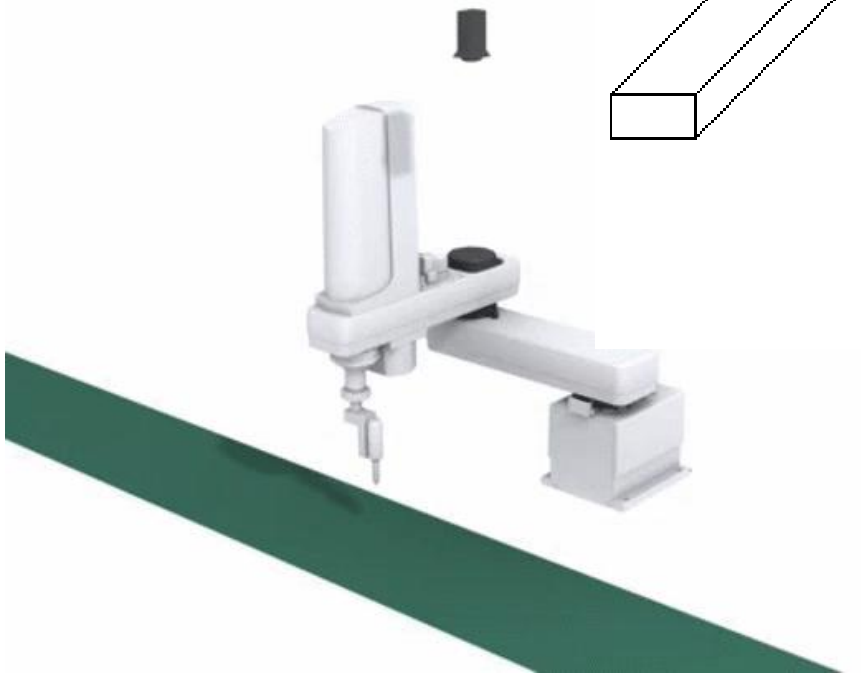


RP: Rocker Phalanx

Design Example of Translational Links

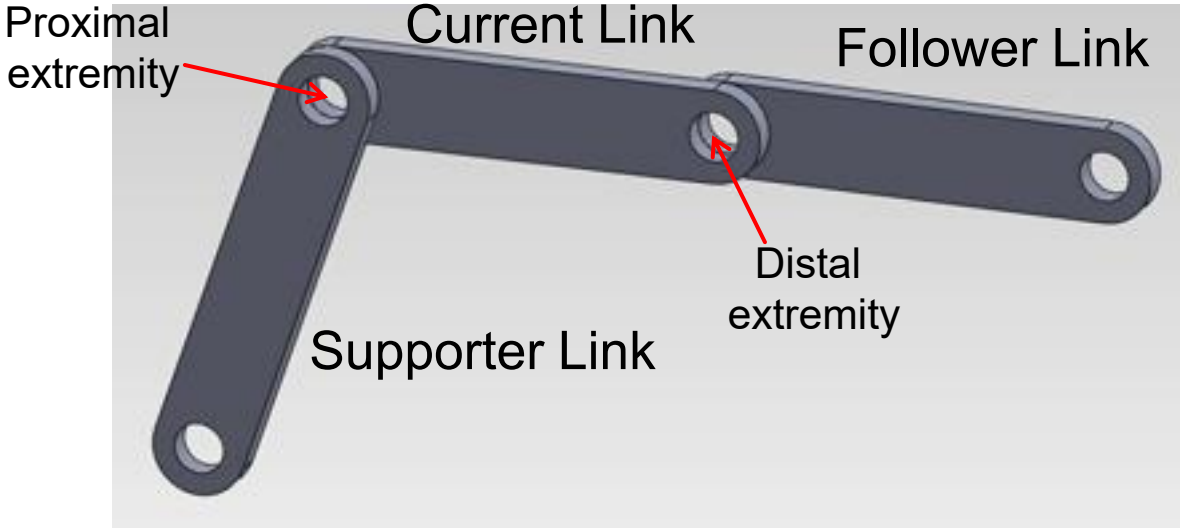


CARTESIAN/RECTILINEAR/GANTRY



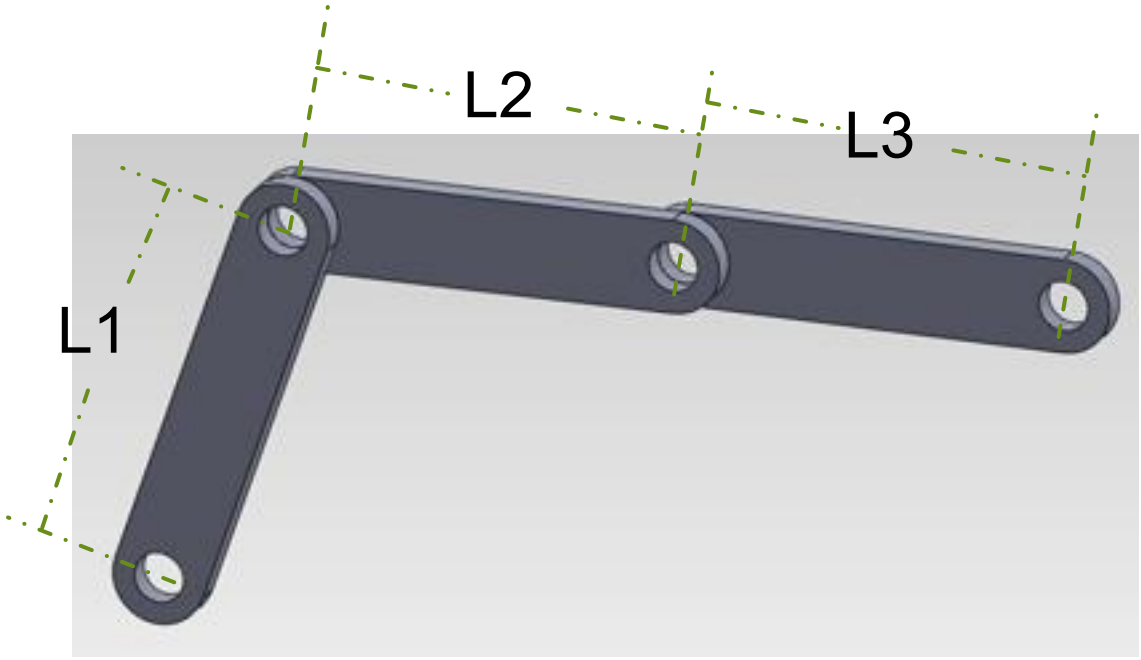
Link's Components: Extremities

- A link has two extremities.
- The one which is to be connected to a supporter link is called the proximal extremity.
- The one which is to be connected to a follower link is call the distal extremity.



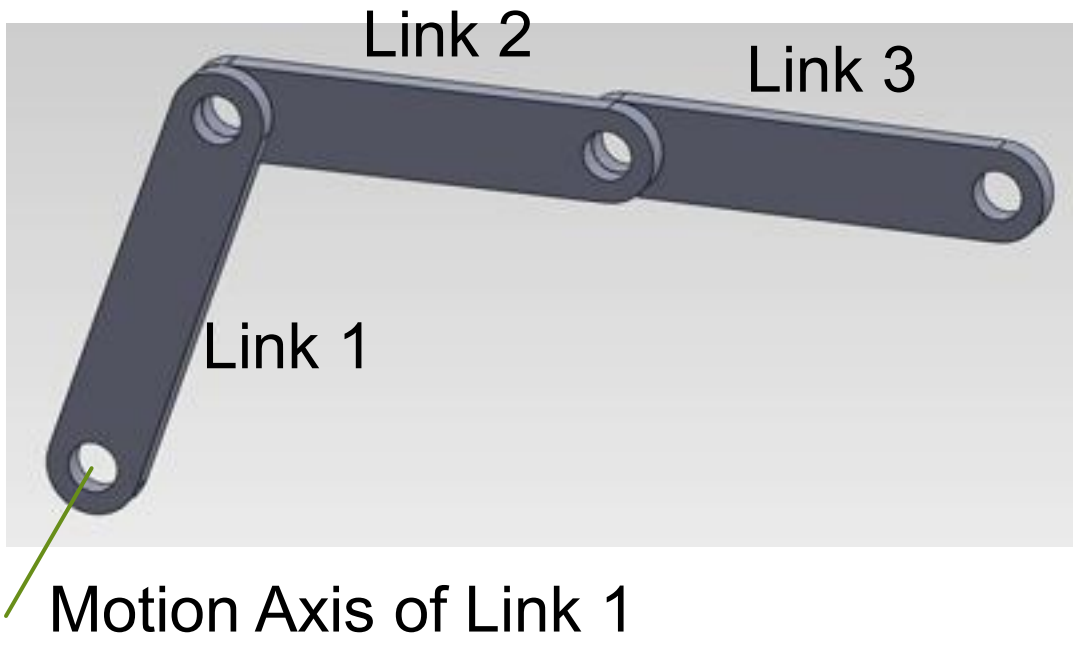
Link's Length

- The length of a link is the distance between its two extremities.

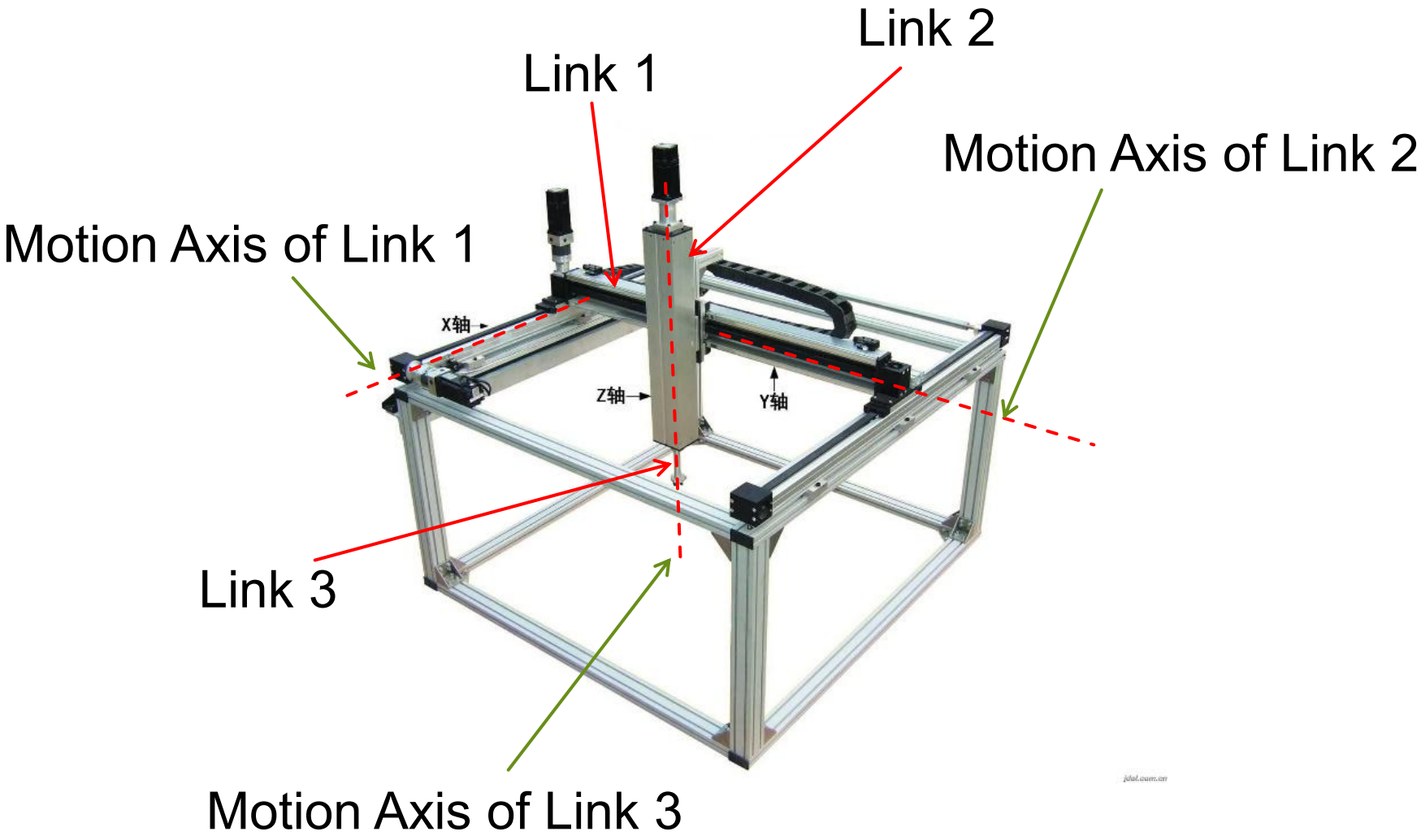


Link's Motion Axis

- A link is a rigid body which is able to undergo either translational motion or rotational motion.
- The axis, which constraints a link's motion, is called the motion axis of a link.

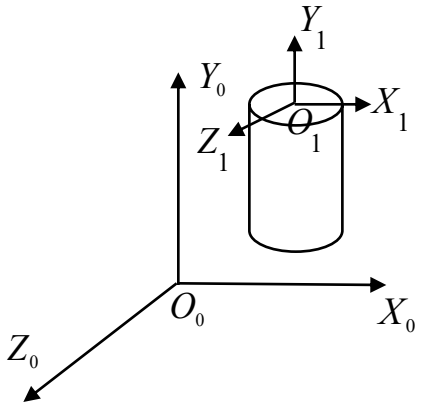


Example of Link's Motion Axes

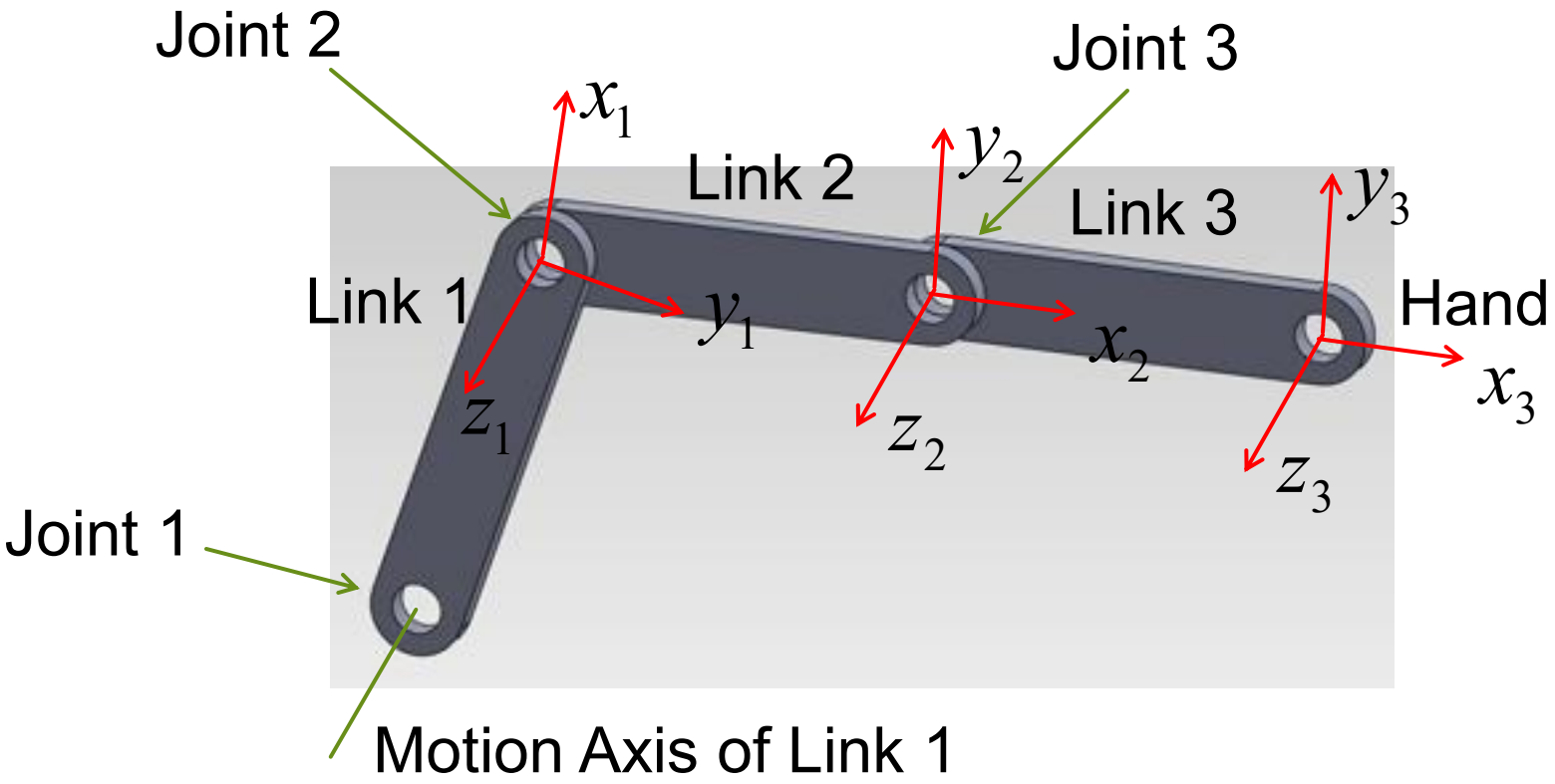


Link's Coordinate System

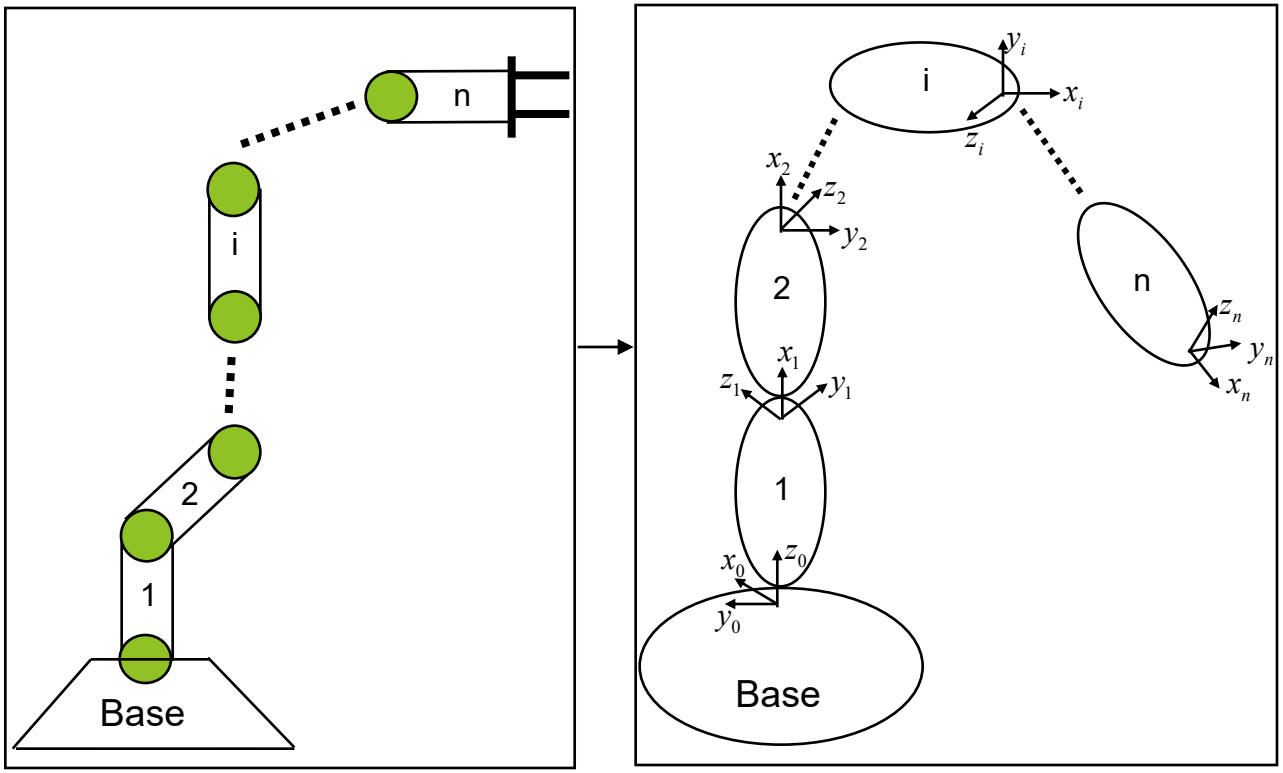
- A link is a rigid body which must be assigned with a coordinate system in order to represent the link's position and orientation. Such a coordinate system is called the coordinate system of a link.
- In robotics, a link's coordinate system is placed at the link's distal extremity, instead of proximal extremity.



Example of Links' Coordinate Systems



Example of Links' Coordinate Systems



What are the transformations between two adjacent coordinate systems?

Link's Kinematic Parameters

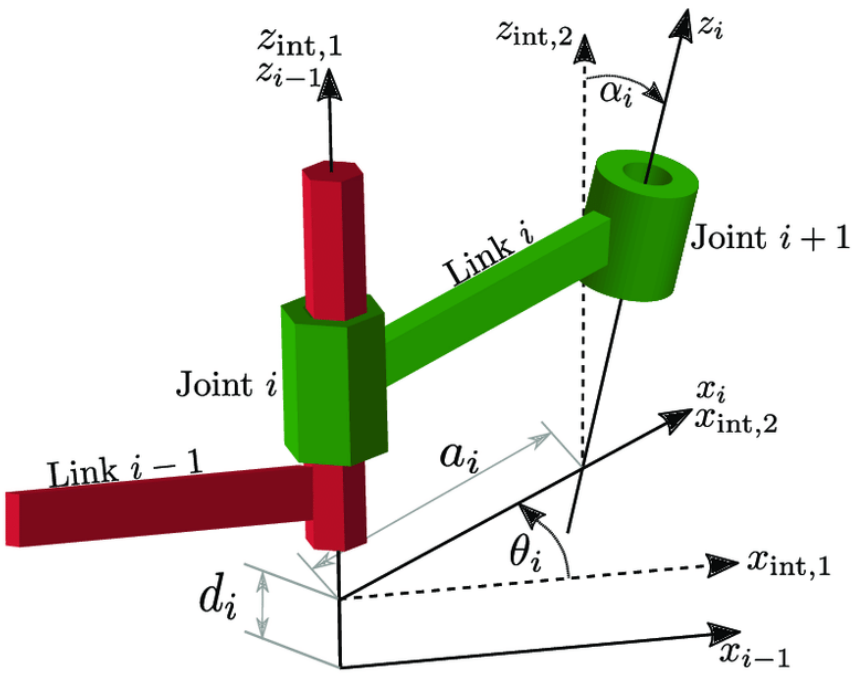


Table of Denavit-Hartenburg Parameters

Link	a_i	α_i	d_i	θ_i
1				
2				
3				

Four Motions to Align Two Coordinate Systems Together

- Rotation about Z axis ⟵ θ_i
- Translation along Z axis ⟵ d_i
- Rotation about X axis ⟵ α_i
- Translation along X axis ⟵ a_i

Link's Transformation Matrix

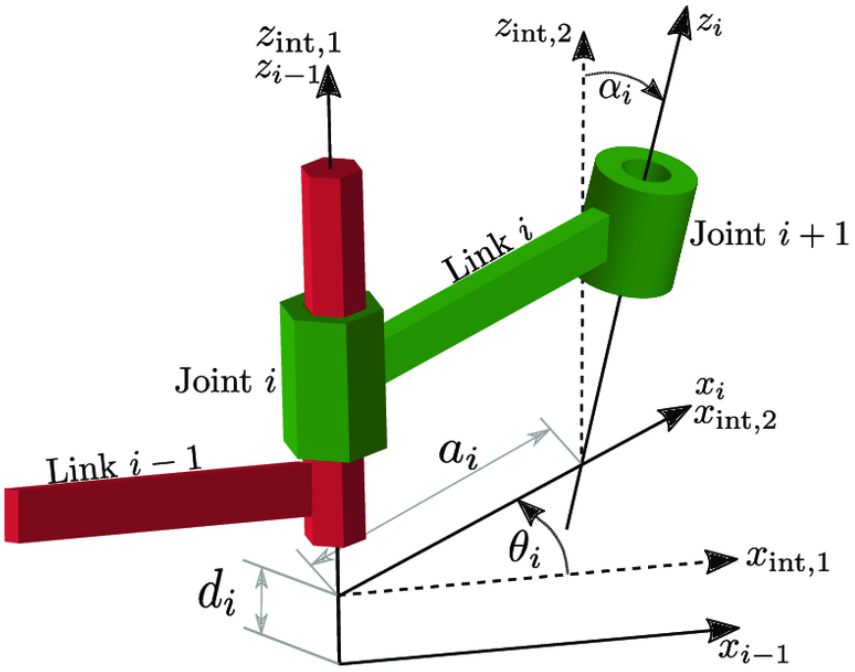


Table of Denavit-Hartenburg Parameters

Link	a_i	α_i	d_i	θ_i
1				
2				
3				

- Rotation about Z axis ⟵ θ_i
- Translation along Z axis ⟵ d_i
- Rotation about X axis ⟵ α_i
- Translation along X axis ⟵ a_i

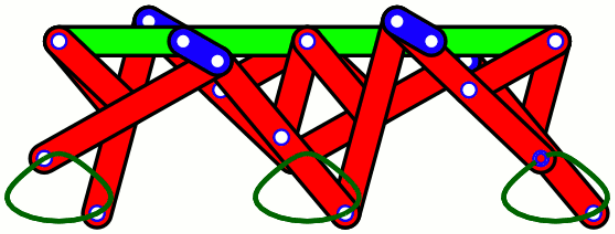
Pose of Link i with respect to Link i - 1

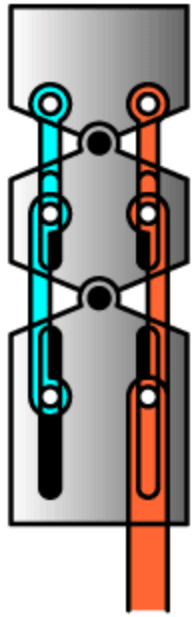
$$L_{i-1,i}(\theta_i) = T_{i-1}(x, a_i)R_{i-1}(x, \alpha_i)T_{i-1}(z, d_i)R_{i-1}(z, \theta_i)$$

(Note: All are 4x4 homogeneous transformation matrices)

Outline of Lecture 2

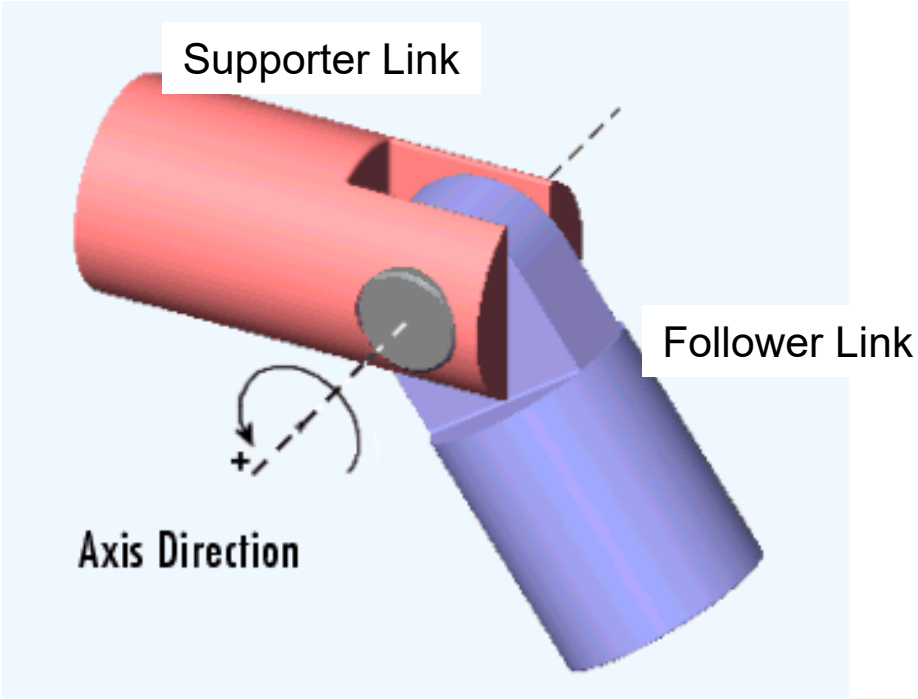
- Design of Link
- Design of Joint
- Design of Mechanism

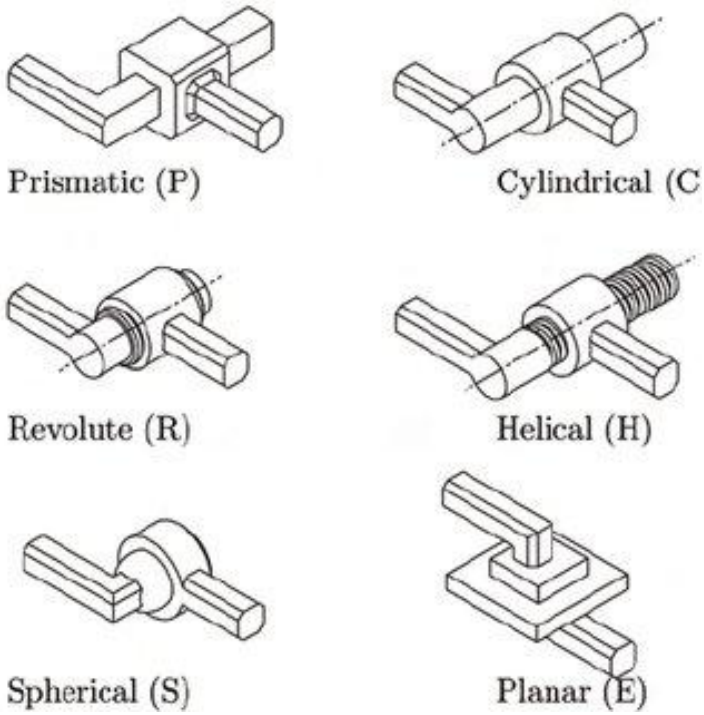




Definition of Joints

- A joint consists of two rigid bodies which can undergo relative motions such as translational motions or rotational motions.





Three types of joints:
- Link Joints
- Torque Joints
- Power Joints

Classification of Joints According to Motion Types

- Prismatic Joints
  - The two rigid bodies undergo relative linear motions.
- Revolute Joints
  - The two rigid bodies undergo relative angular motions.

What are the output from Joints?

Answer:

- All motions involve mechanical energy.
- The output from a joint is mechanical energy, which can be described by mechanical power. Such power is called the power of a joint.

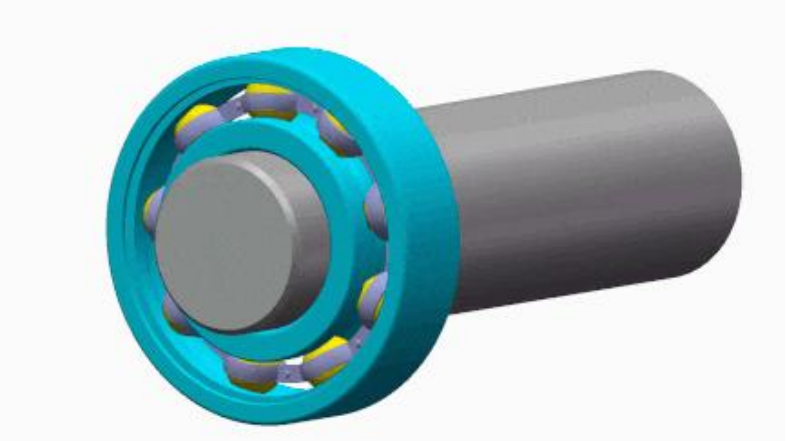
$$P_{linear} = F \bullet v$$

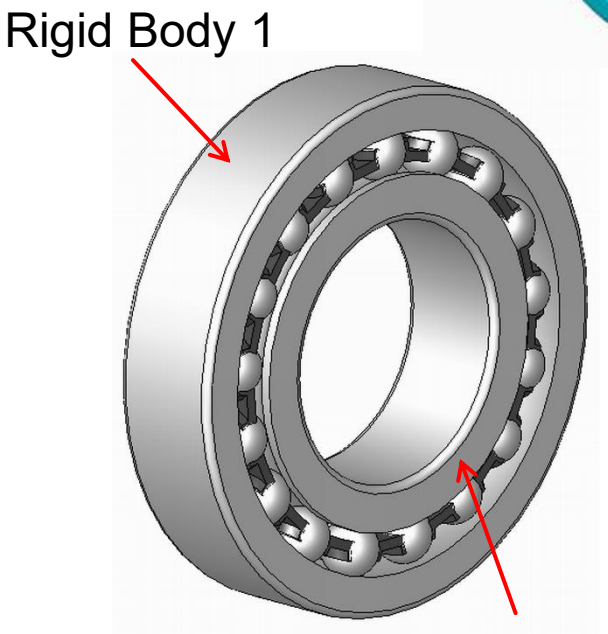
$$P_{rotation} = \tau \bullet \omega$$

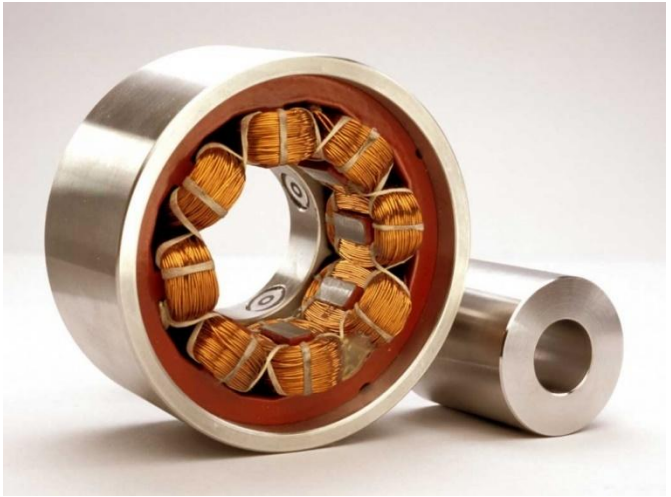
Classification of Joints According to Energy Flow

- Power Joints
  - Joint devices which are the sources of mechanical power.
- Torque Joints
  - Joint devices which amplify the output torques.
- Link Joints
  - Joint devices which constrain the motion between two adjacent links.

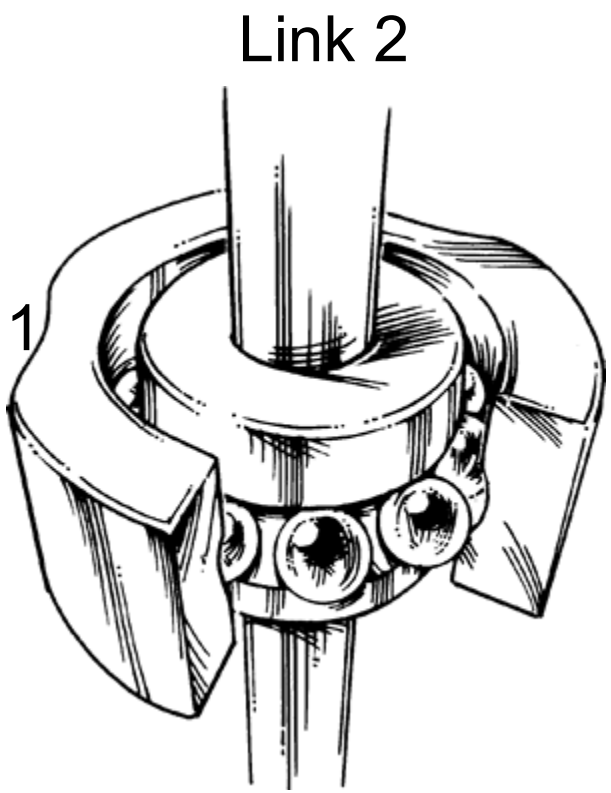
Design Example of Link Joint (1)

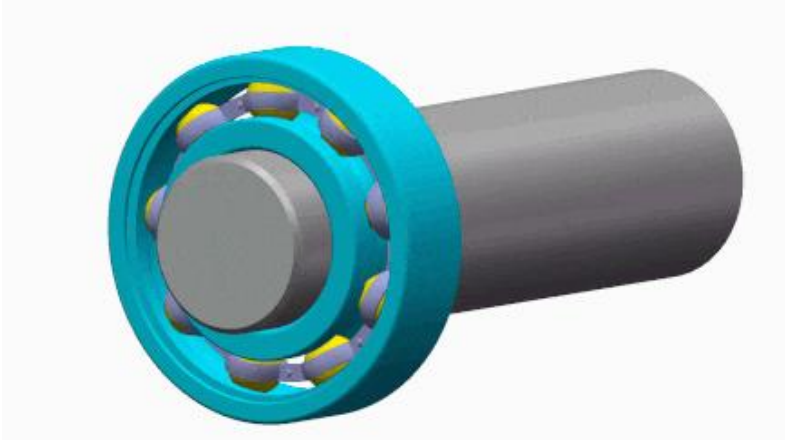




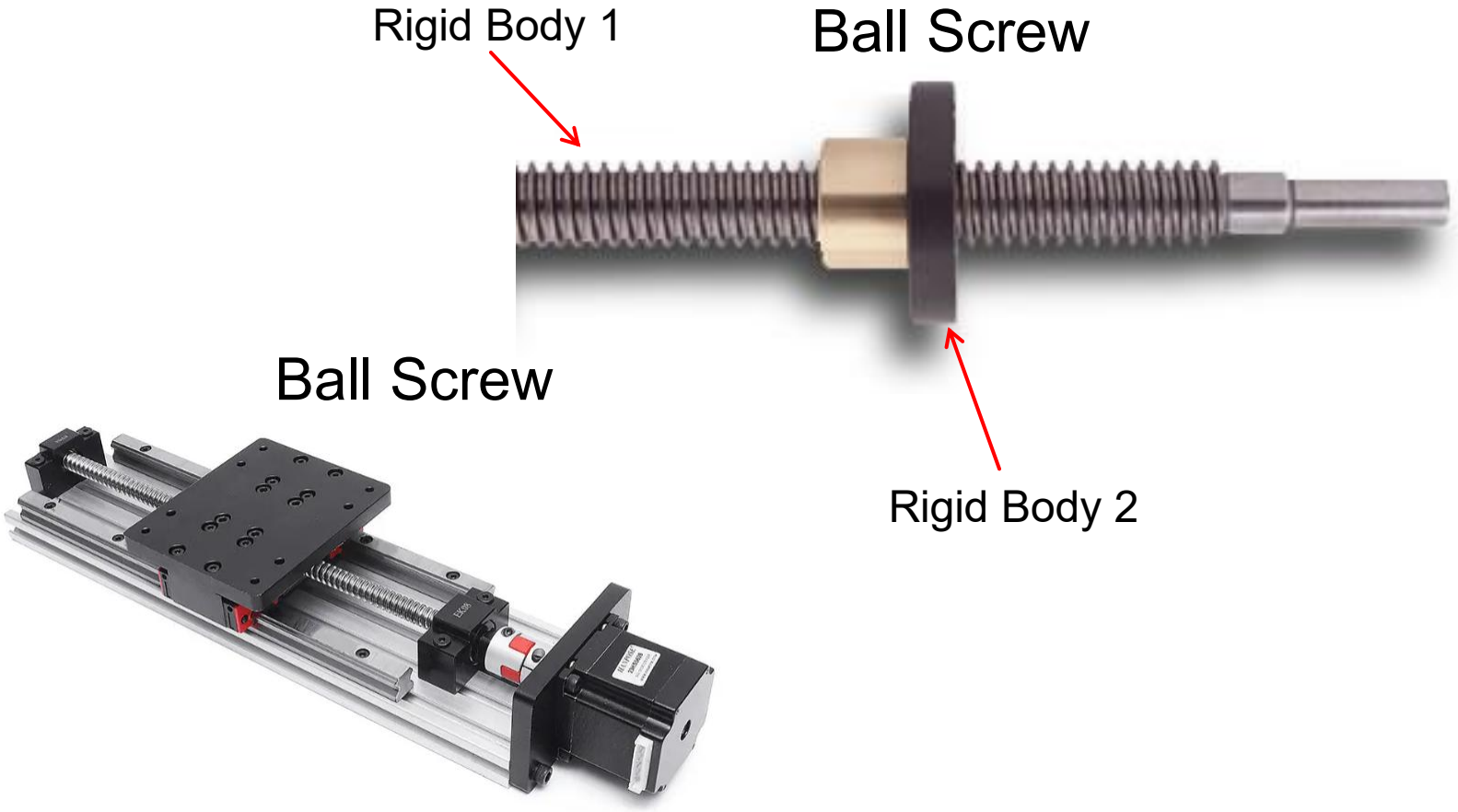


Design Example of Link Joint (2)





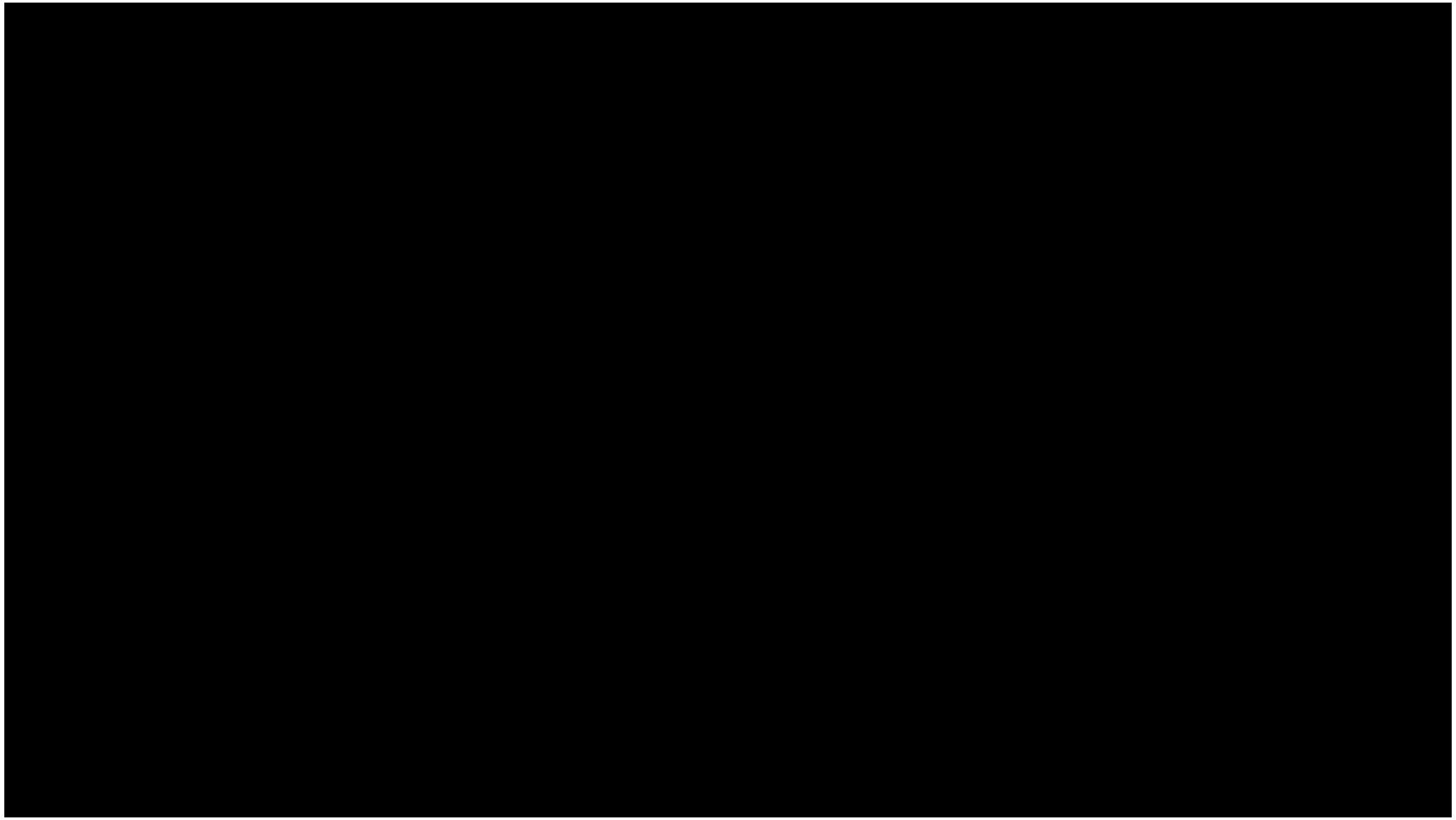
Design Example of Link Joint (3)



Design Example of Link Joint (4)

Design Example of Torque Joint (1)

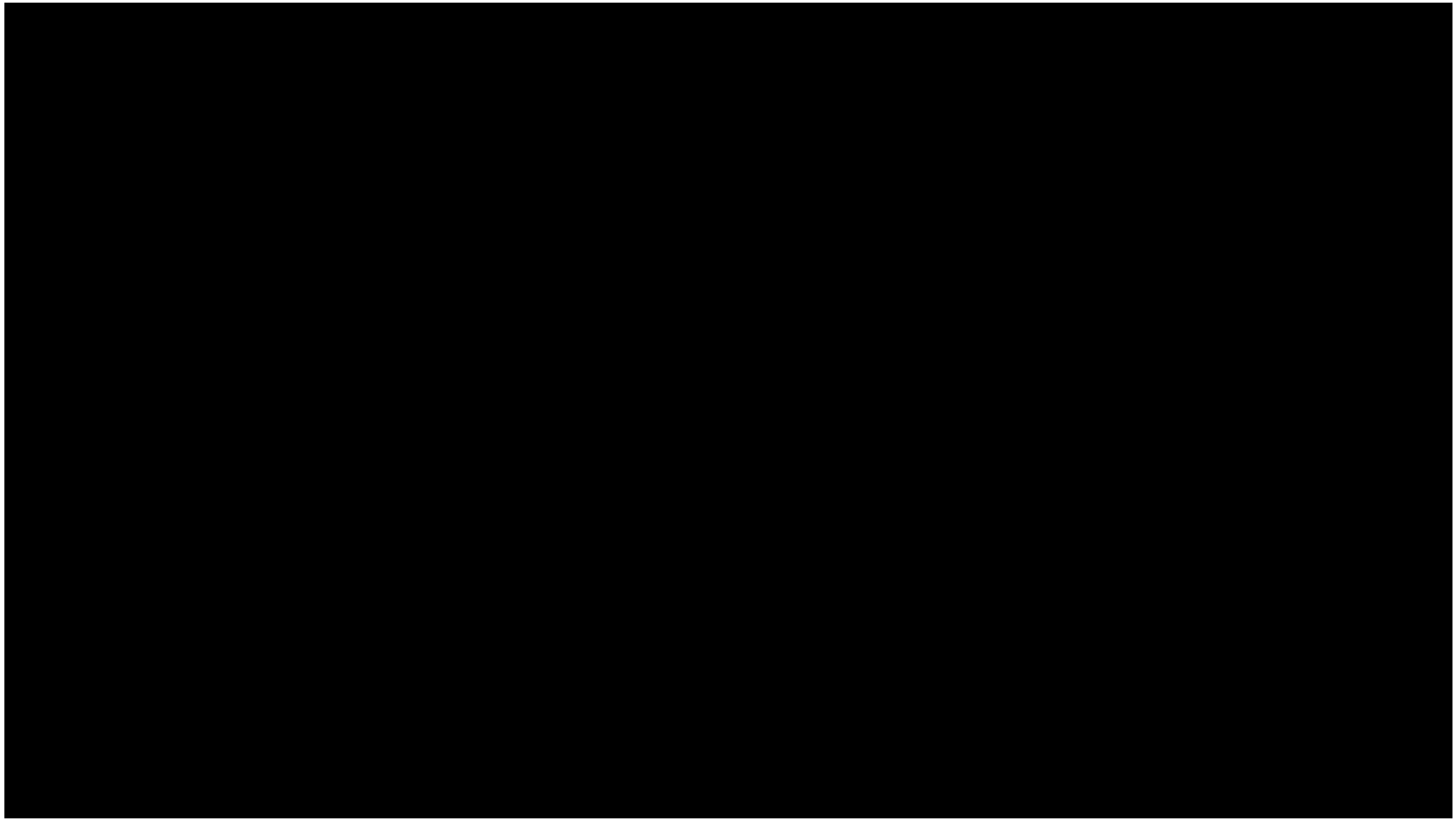
Design Example of Torque Joint (2)



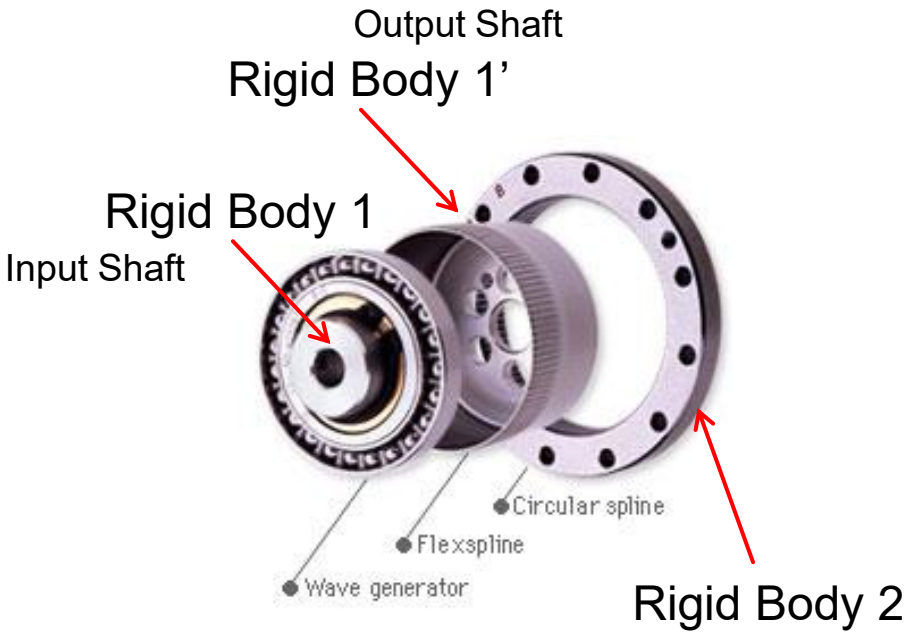
Design Example of Torque Joint (3)

Design Example of Torque Joint (4)

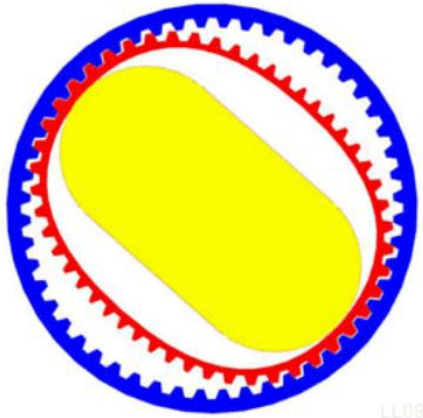
Design Example of Torque Joint (5)



Example of Torque Joint: Harmonic Drive



What is the speed reduction ratio?

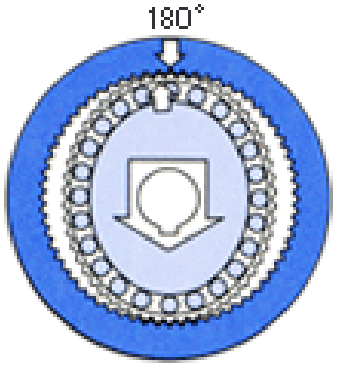


The difference between the teeth of circular spline and the teeth of flexible spline is two.

One full rotation of Rigid Body 1 will make Rigid Body 1' to rorate 2 teeth.

Working Principle of Harmonic Drive

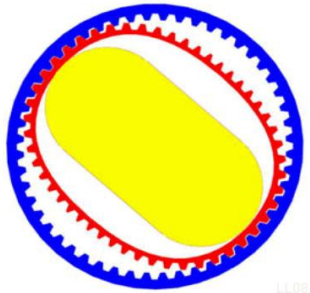
$$\tau_{in}\omega_{in} = k\tau_{in}\frac{\omega_{in}}{k} \qquad \tau_{out} = k\tau_{in}$$



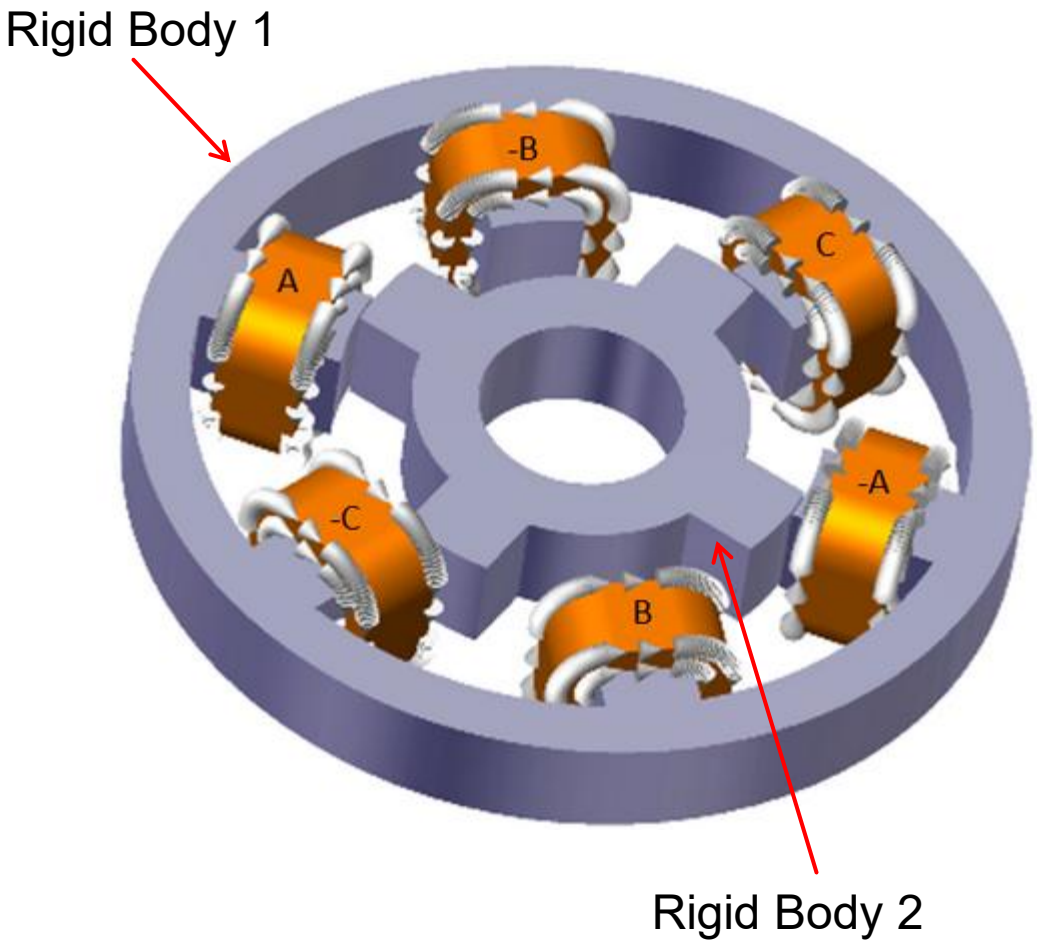
Animation of movements

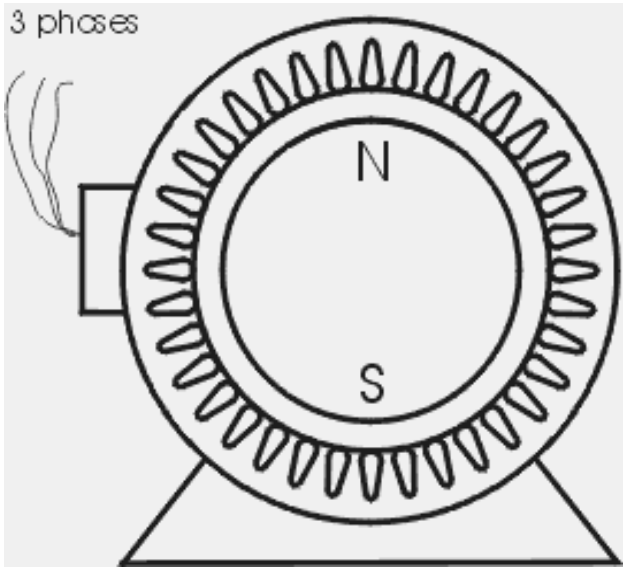
When the wave generator rotates 180 degrees clockwise, the flexspline moves counterclockwise by one tooth relative to the circular spline.

When the wave generator rotates one revolution clockwise (360 degrees), the flexspline moves counterclockwise by two teeth relative to the circular spline because the flexspline has two fewer teeth than the circular spline. In general terms, this movement is treated as output power.



Design Example of Power Joint (Lecture 4)



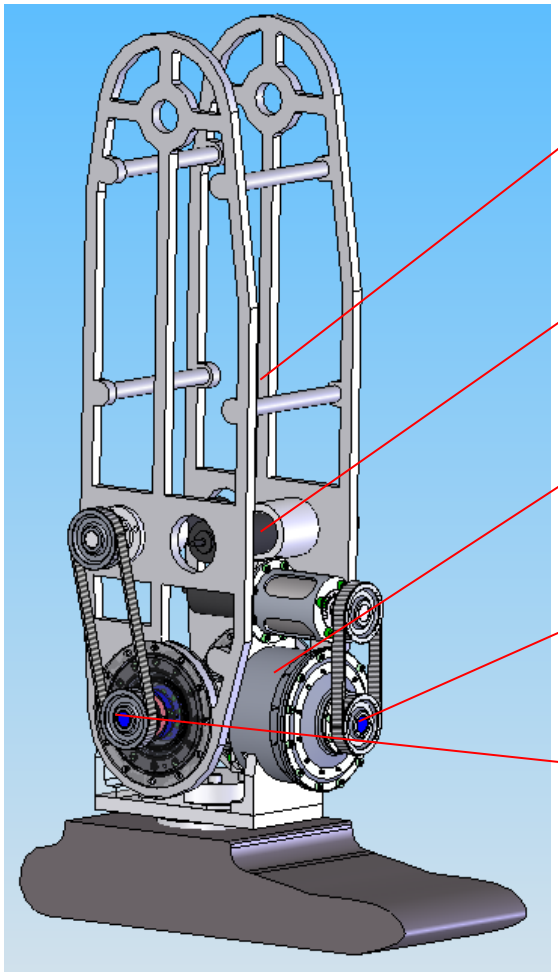


$$P = \tau\omega$$

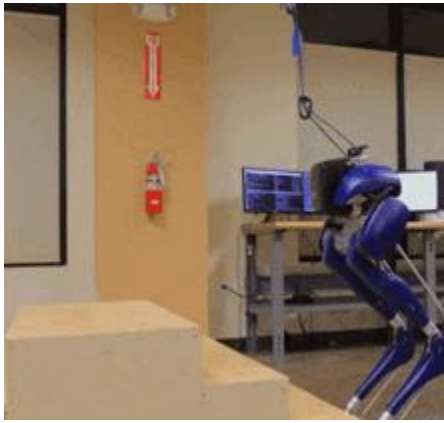
Design Schemes of Robotic Joints

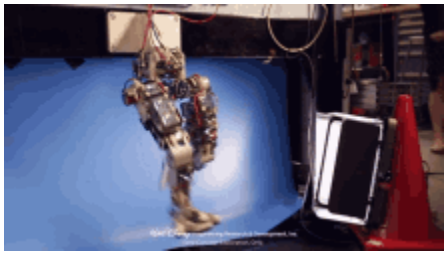
- Scheme 1:
  - Power Joint + Link Joint
- Scheme 2:
  - Power Joint + Torque Joint + Link Joint
- Scheme 3:
  - Power Joint + Transmission + Torque Joint + Link Joint

Design Example of Robotic Joint (1)

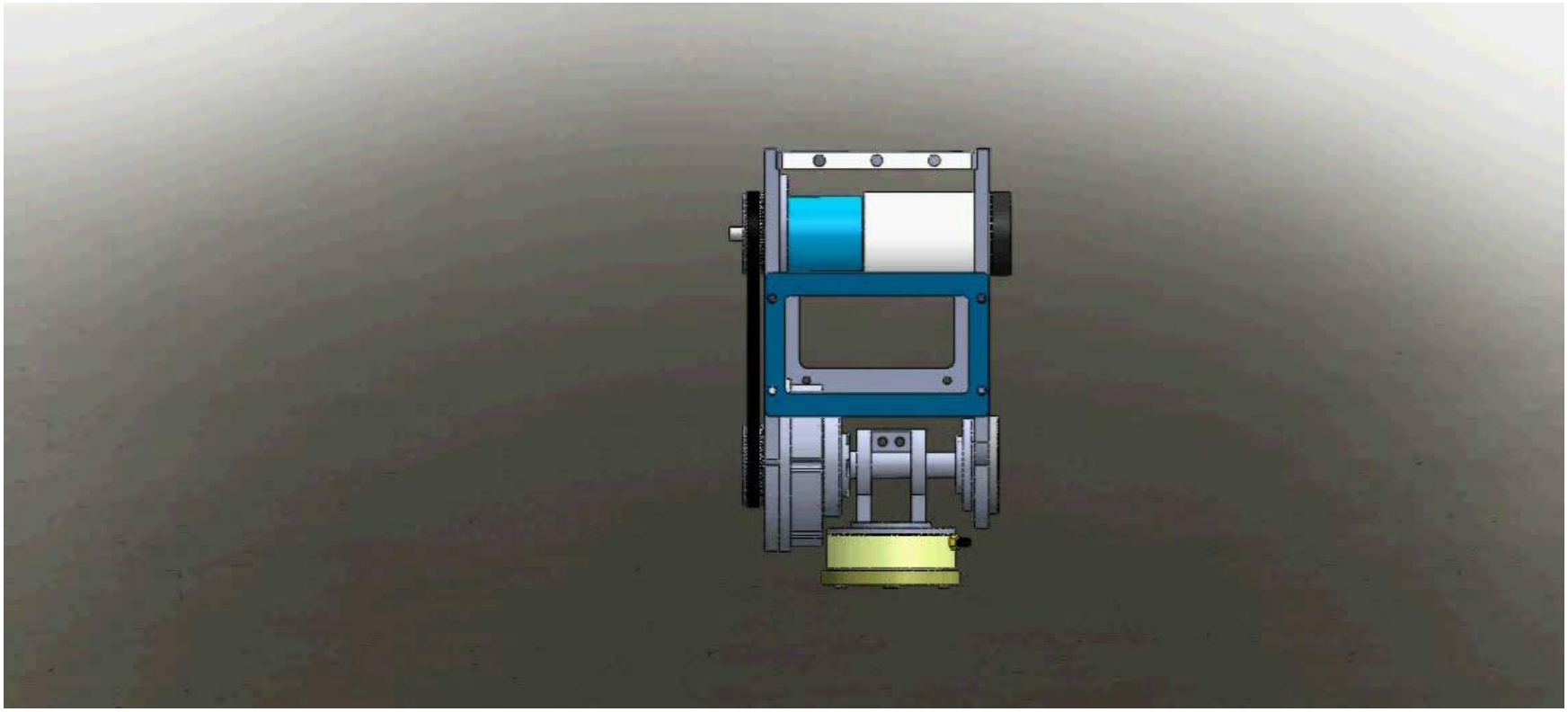


- Link
- Power Joint
- Torque Joint
- Link Joint of Ankle Roll
- Link Joint of Ankle Pitch





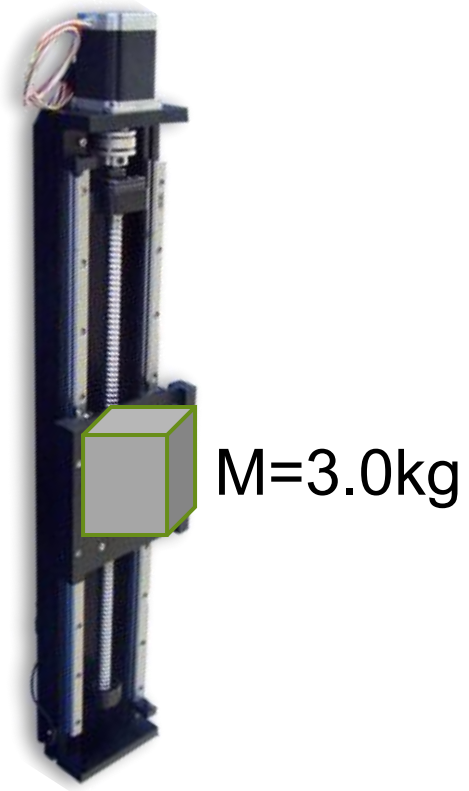
Design Example of Robotic Joint (2)



Design Example of Robotic Joint (3)

Example of Motor Sizing

- A prismatic link joint carries a payload of 3.0 kg in a vertical direction. The kinetic frictional force of the joint is 0.5 N. If the payload moves up at a speed of 3.5 cm/s, what is the output power from the power joint?



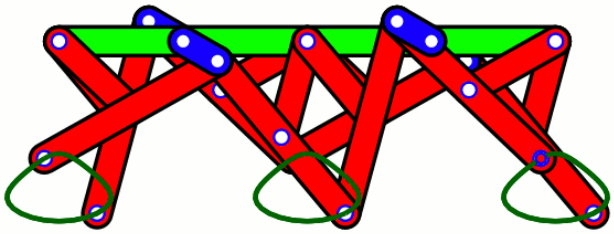
- Answer:

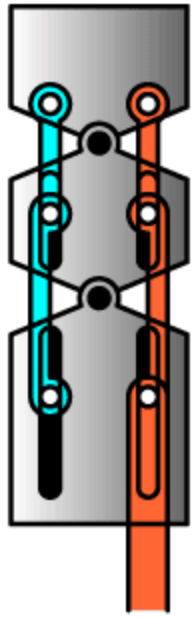
$$F = F_k + mg = 0.5N + 3.0 \times 9.8 = 29.9N$$

$$P_{linear} = F \times v = 29.9 \times 0.035 = 1.029 \text{ W}$$

Outline of Lecture 2

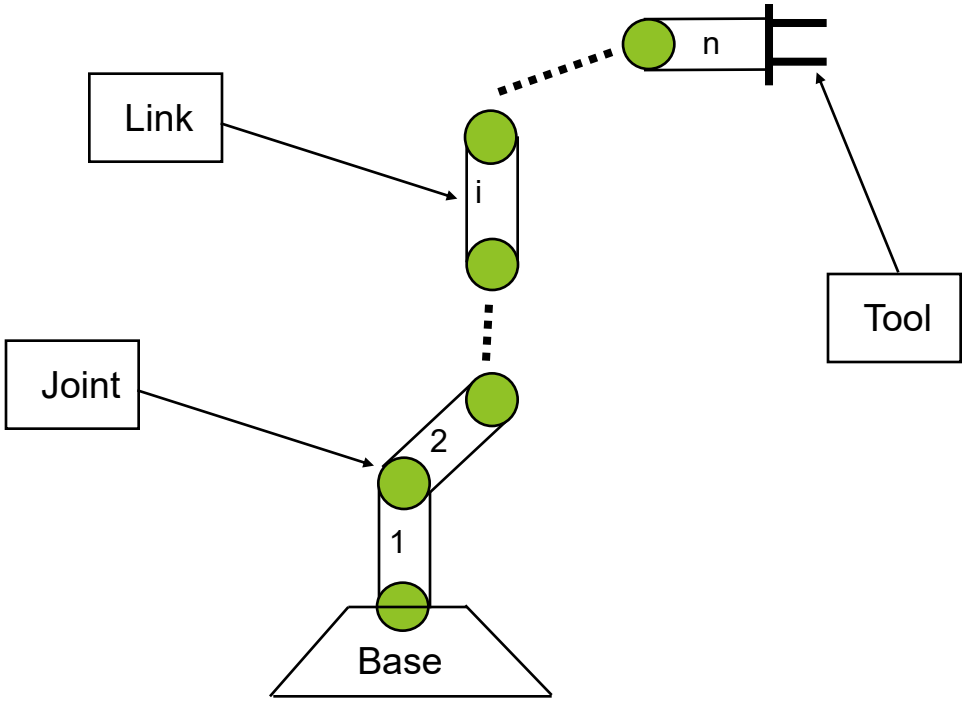
- Design of Link
- Design of Joint
- Design of Mechanism



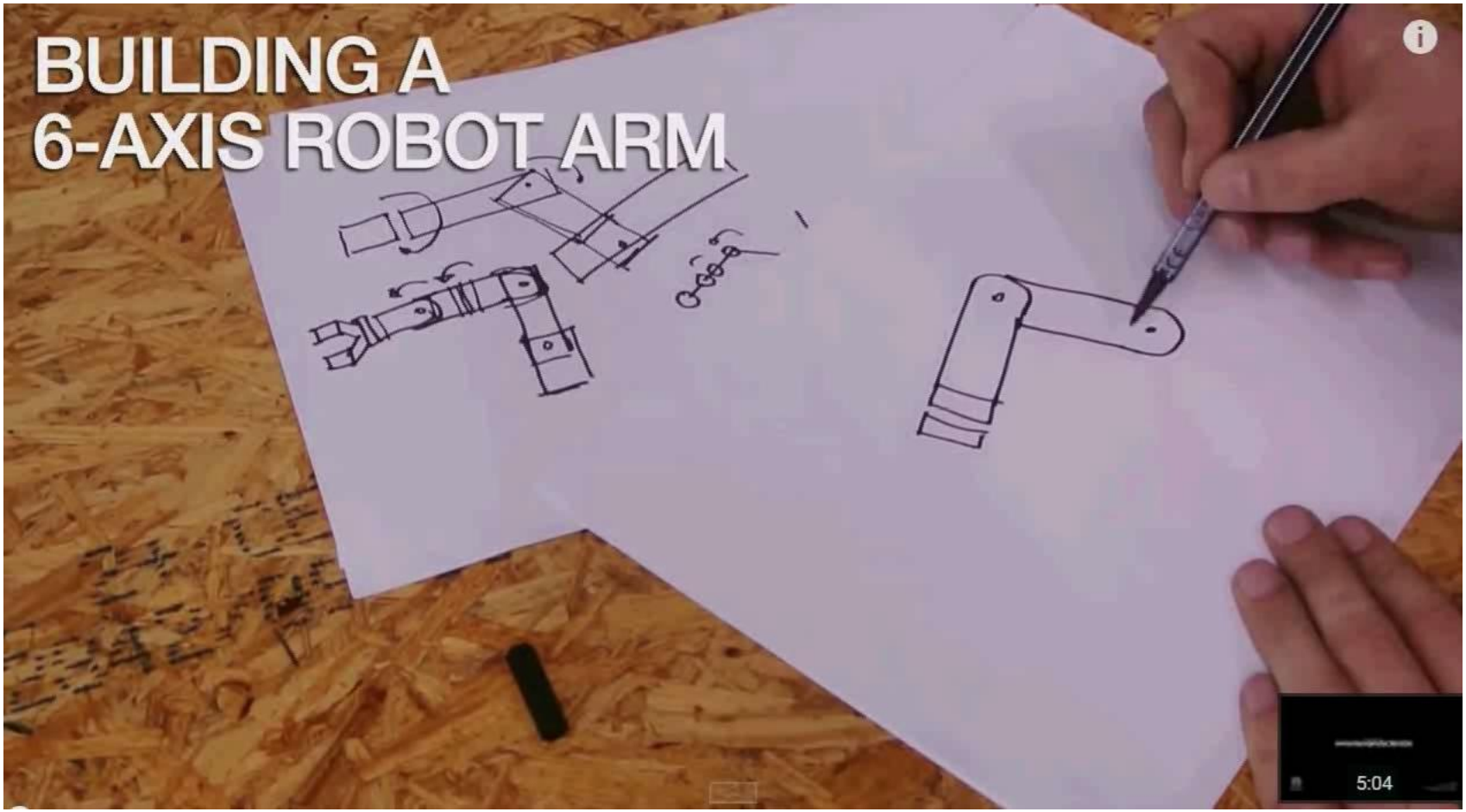


Definition of Mechanism

- A mechanism is a set of links which are interconnected by joints, and could undergo relative motions.

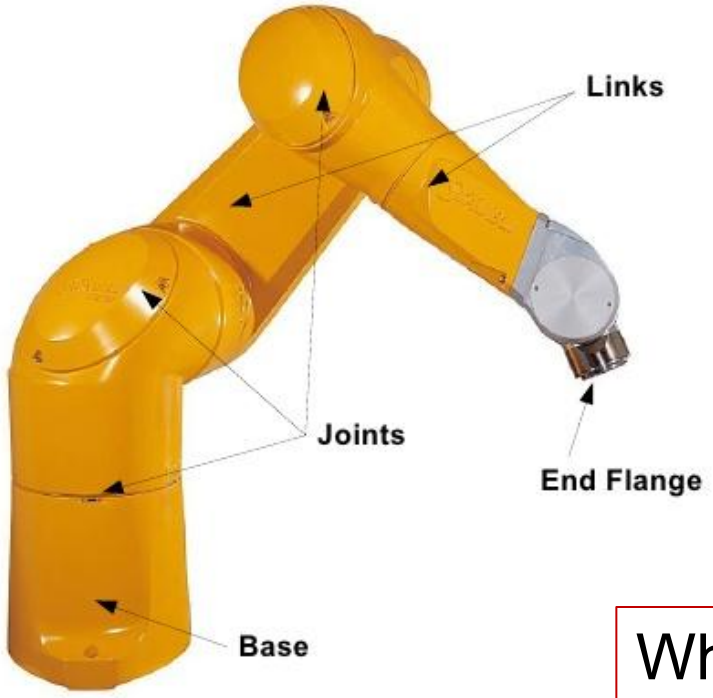


Example of Designing Robot Mechanism



Robot arm's mechanism is also called as kinematic chain which has input and output

Mechanism of A Robot Manipulator



A Robot Arm is a

KINEMATIC CHAIN

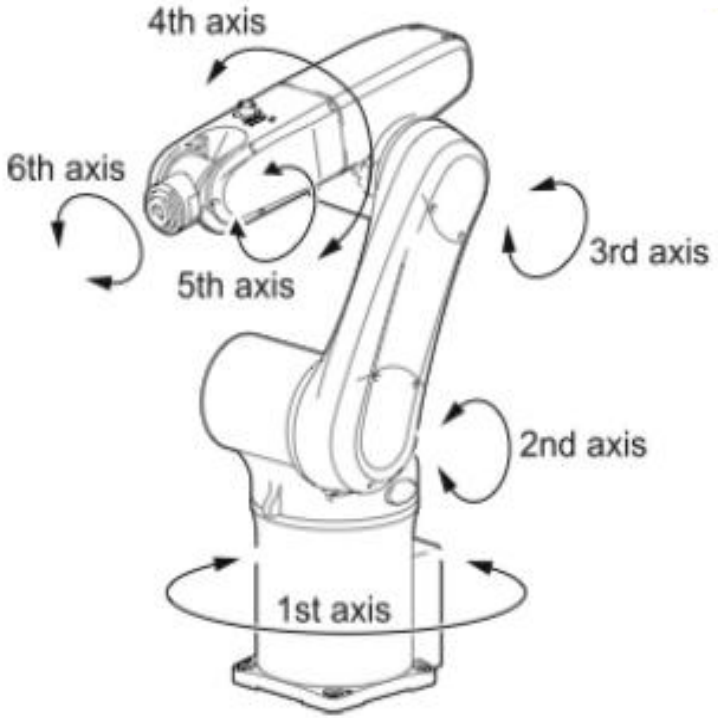
which is comprised of

a. Links
b. Joints

What are the input motions?
What are the output motions?

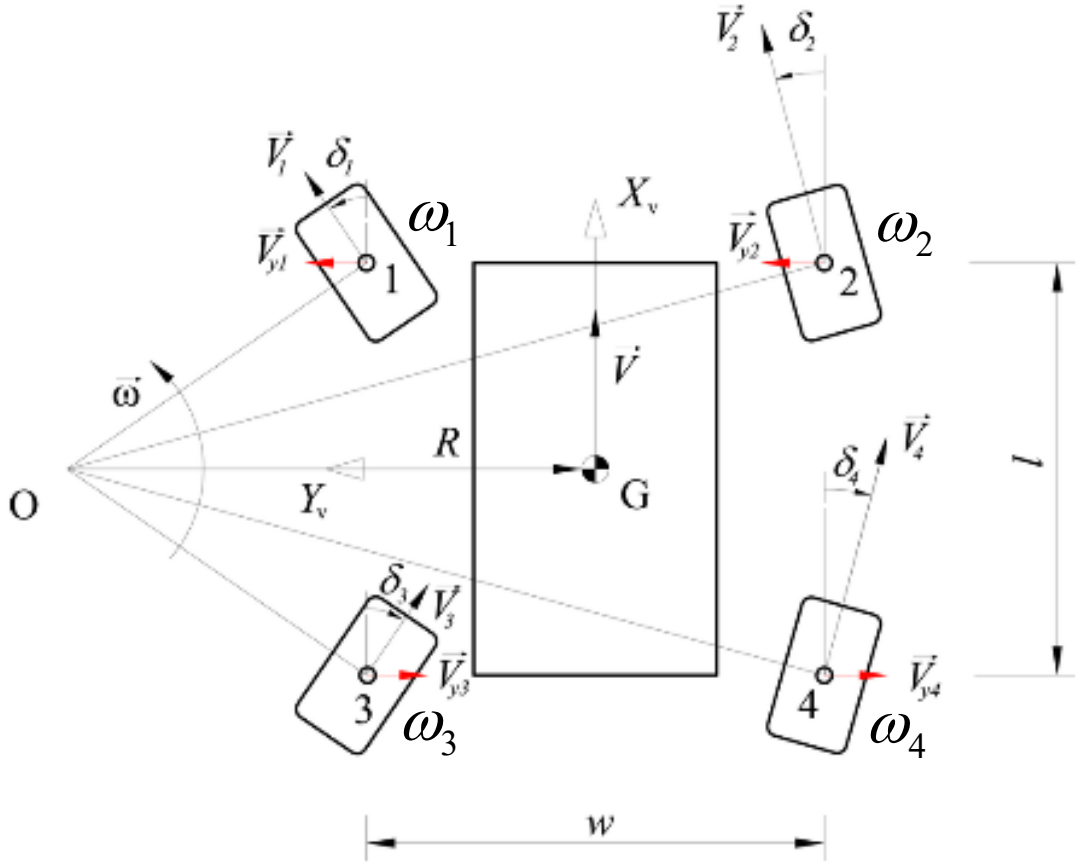
Input Motion of Robot Mechanism

- The motions of the joints inside a mechanism are the input motion of the mechanism.



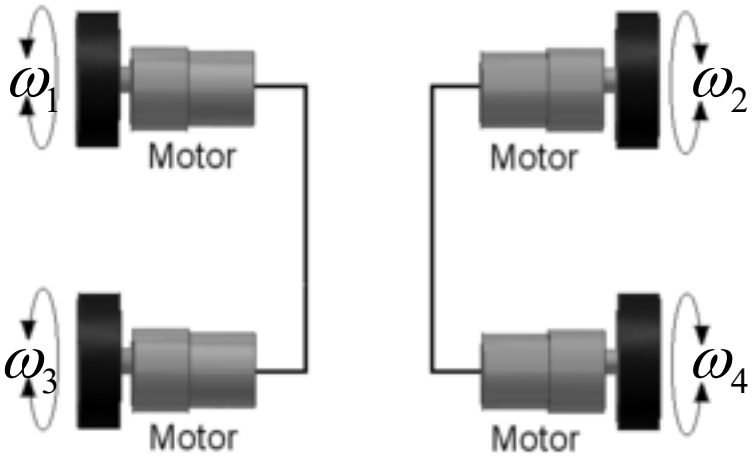
Example of Mobile Base

- What are the input motions which drive the mobile robot below?



The input motion vector is :

$$(\omega_1, \omega_2, \omega_3, \omega_4)$$



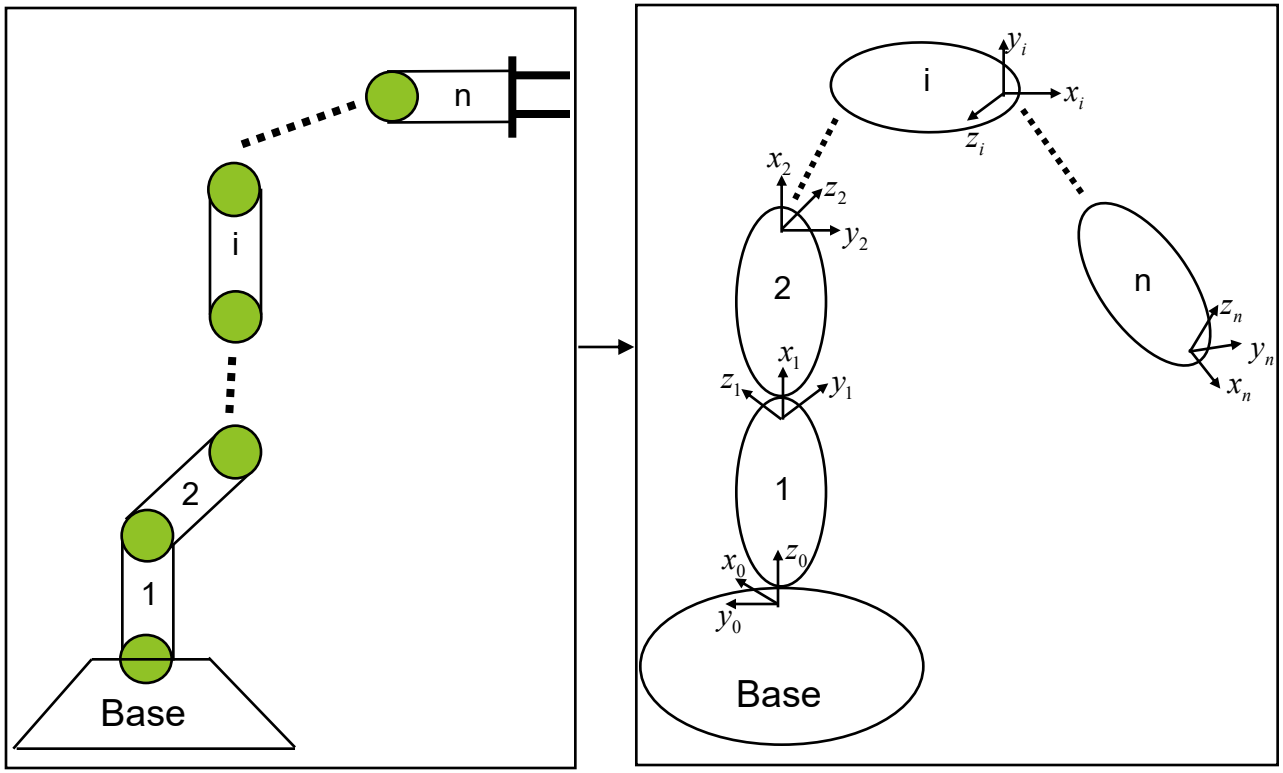
Output Motion of Robot Mechanism

- The motions of a tool attached to the mechanism are the output motions.

What is Kinematics of Robot Mechanism?

- It is the study of mechanisms' motions without consideration of force and torque.
- The study of output motion as functions of input motion in a mechanism is called forward kinematics.
- The study of input motion as functions of output motion in a mechanism is called inverse kinematics.

Robot's Forward Kinematics



Vector of Generalized Coordinates

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_{n-1} \\ \theta_n \end{bmatrix}$$

Pose of Link n with respect to Link 0 or Base Link

$$L_{0,n}(\mathrm{q}) = L_{0,1}(\theta_1)L_{1,2}(\theta_2)\ldots L_{n-1,n}(\theta_n)$$

Example of Forward Kinematics

- A planar arm manipulator has three joints and three links. The first joint angle is between axes X0 and X1. The second joint angle is between axes X1 and X2. The third joint angle is between axes X2 and X3. If the three joint angles are (110.0, -80.0, -45.0) degrees, what are the X and Y coordinate of O3?

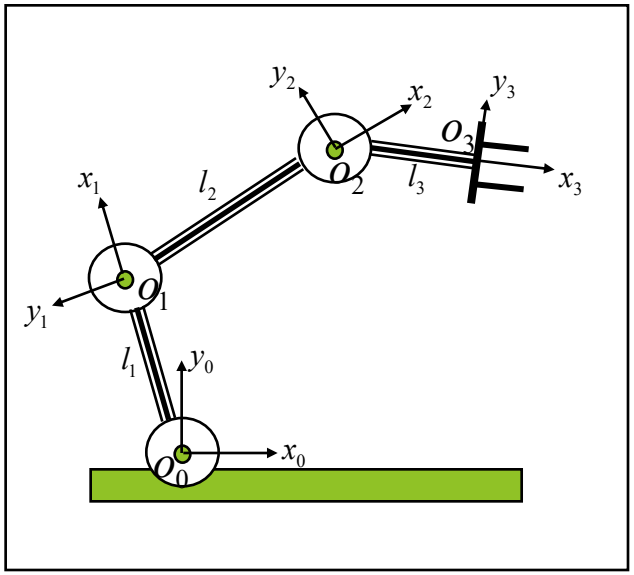
- Answer:

$$x_{o3} = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

$$x_{o3} = l_1 \cos(110^0) + l_2 \cos(30^0) + l_3 \cos(-15^0)$$

$$y_{o3} = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

$$y_{o3} = l_1 \sin(110^0) + l_2 \sin(30^0) + l_3 \sin(-15^0)$$



Robot's Jacobian Matrix

Pose of Link n with respect to Link 0 or Base Link

$$L_{0,n}(\mathrm{q}) = L_{0,1}(\theta_1)L_{1,2}(\theta_2)\dots L_{n-1,n}(\theta_n)$$

Velocity of Link n's Pose = **J** x Velocity of Generalized Coordinates

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{bmatrix}_{6\times 1} = \begin{bmatrix}J_{11} & J_{12} & \cdot & \cdot & \cdot & J_{1n}\\ J_{21} & J_{22} & \cdot & \cdot & \cdot & J_{2n}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \cdot & \cdots & \cdot & \cdot & \cdot & \cdot\\ J_{61} & \cdots & \cdot & \cdot & \cdot & J_{6n}\end{bmatrix}_{6\times n} * \begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\\ \cdot\\ \vdots\\ \dot{q}_n\end{bmatrix}_{n\times 1}$$

What is Dynamics of Robot Mechanism?

- It is the study of mechanisms' motions in relationship with force and torque.
- The study of output motion as functions of applied force and torque in a mechanism is called direct dynamics.
- The study of force and torque as functions of output motion in a mechanism is called inverse dynamics.

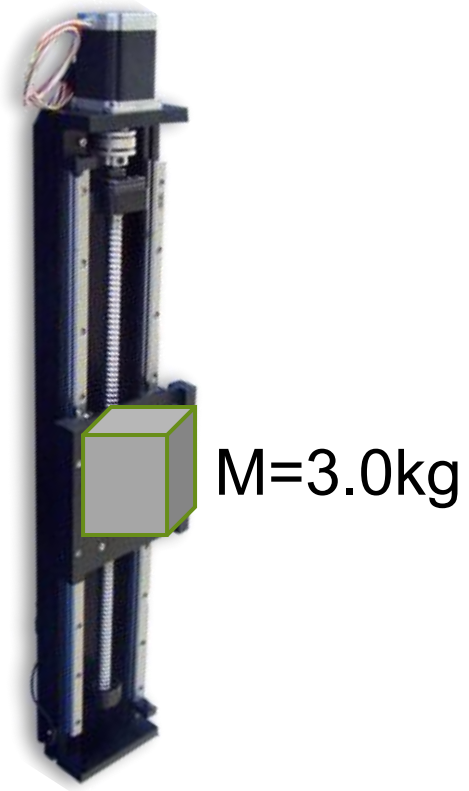
Example of Forward Dynamics

- A prismatic link joint carries a payload of 3.0 kg in a vertical direction. The kinetic frictional force of the joint is 0.5 N. If the acting force from the link joint is 35.0 N, what is the acceleration of the payload?

- Answer:

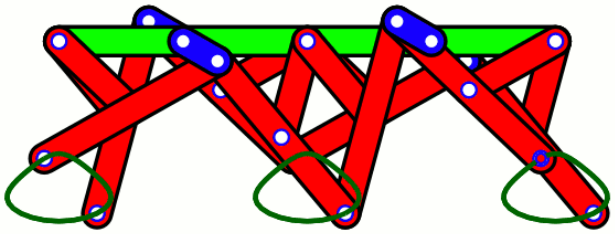
$$F = F_{acting} - F_k - mg = 35.0N - 0.5N - 3.0 \times 9.8 = 5.1N$$

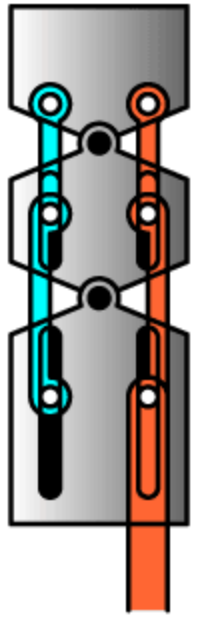
$$a = \frac{F}{m} = \frac{5.1}{3.0} = 1.7\ m/s^2$$



Summary of Lecture 2

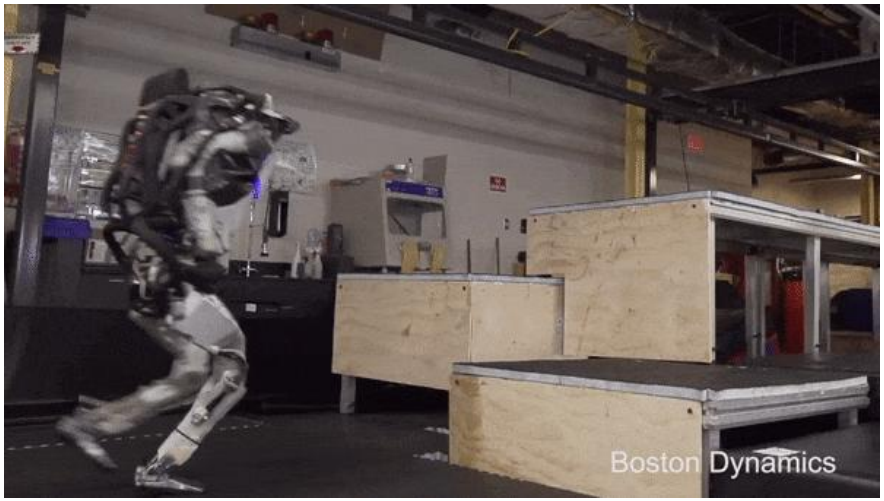
- Design of Link
- Design of Joint
- Design of Mechanism





Outline of Module 1

- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

Lecture 3

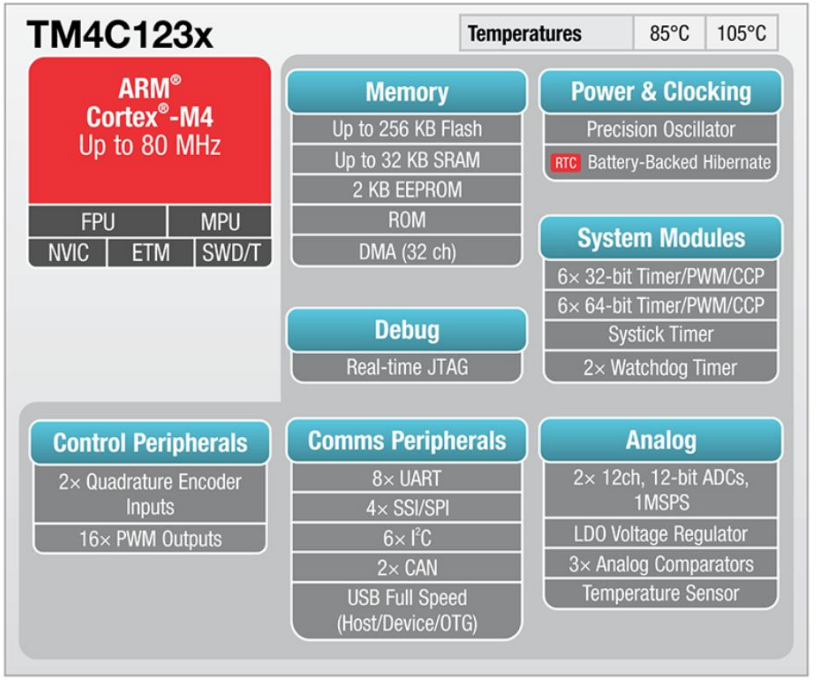
Robot Controllers

Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

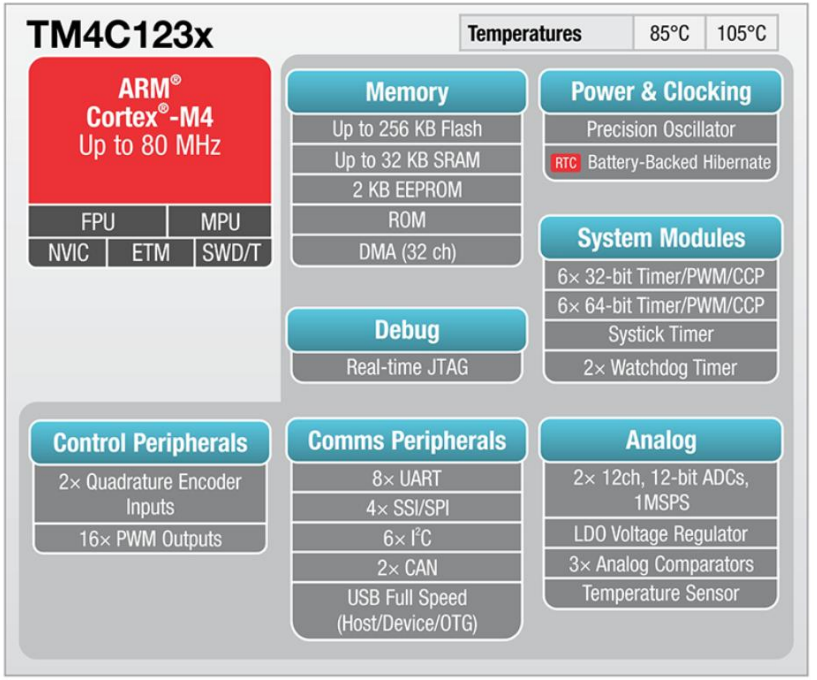
Outline of Lecture 3

- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO



Outline of Lecture 3

- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO

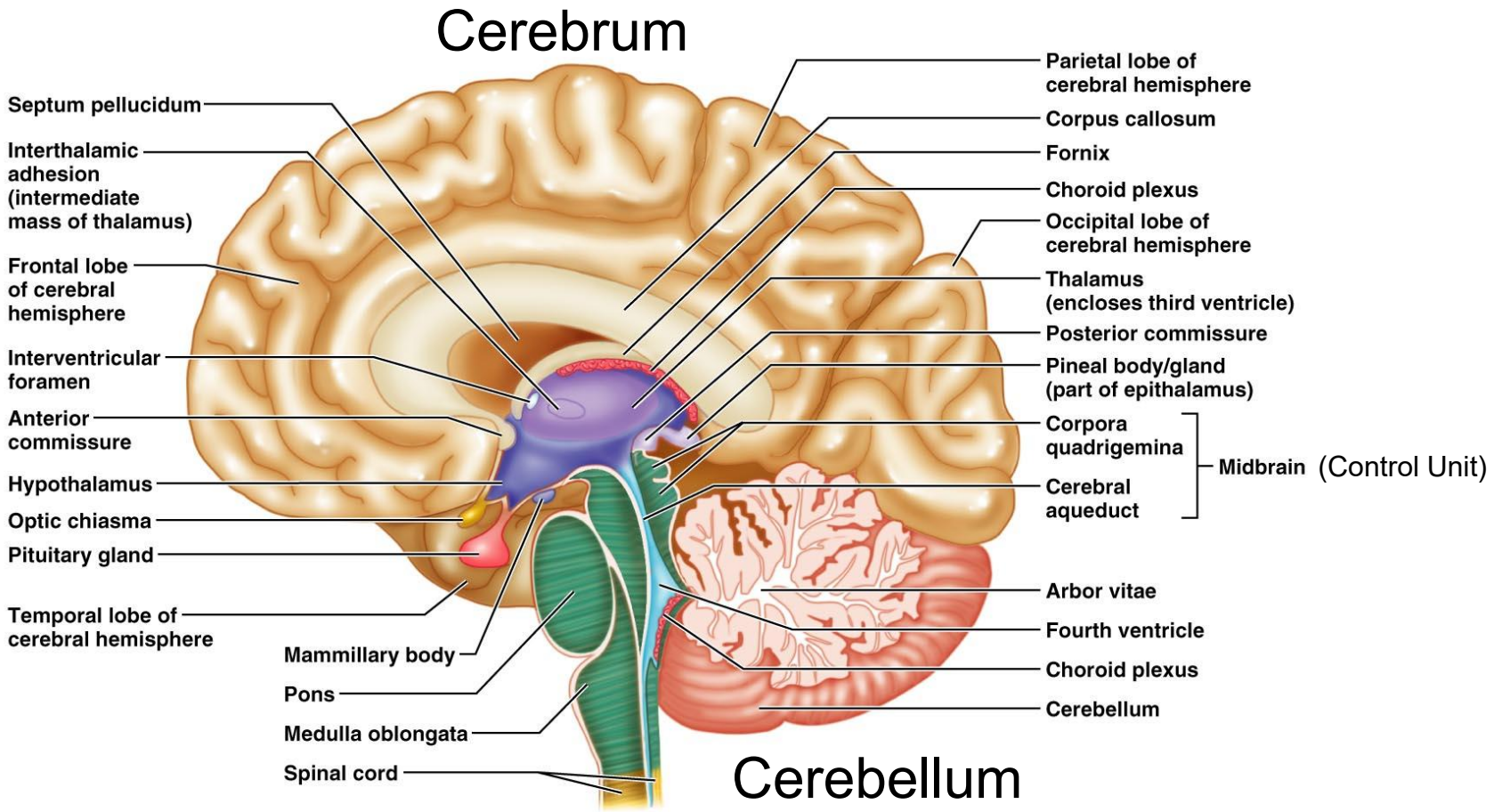


Example of Use of Human Brain/Mind

- Person A tells person B: "Please read this question carefully and give me the answer: what is the sum of 6 + 9?". Explain the mental activities of person B when she or he reads the question and produce the answer.
- Answer:

1. Read and store the question
2. Interpret the meaning of instruction "+"
3. Read the operand 6
4. Read the operand 9
5. Do arithmetic operation of "add"
6. Store the result "15"
7. Synthesize the answer "The sum is 15"
8. Pronounce the answer
9. Wait and listen to the reply from person A

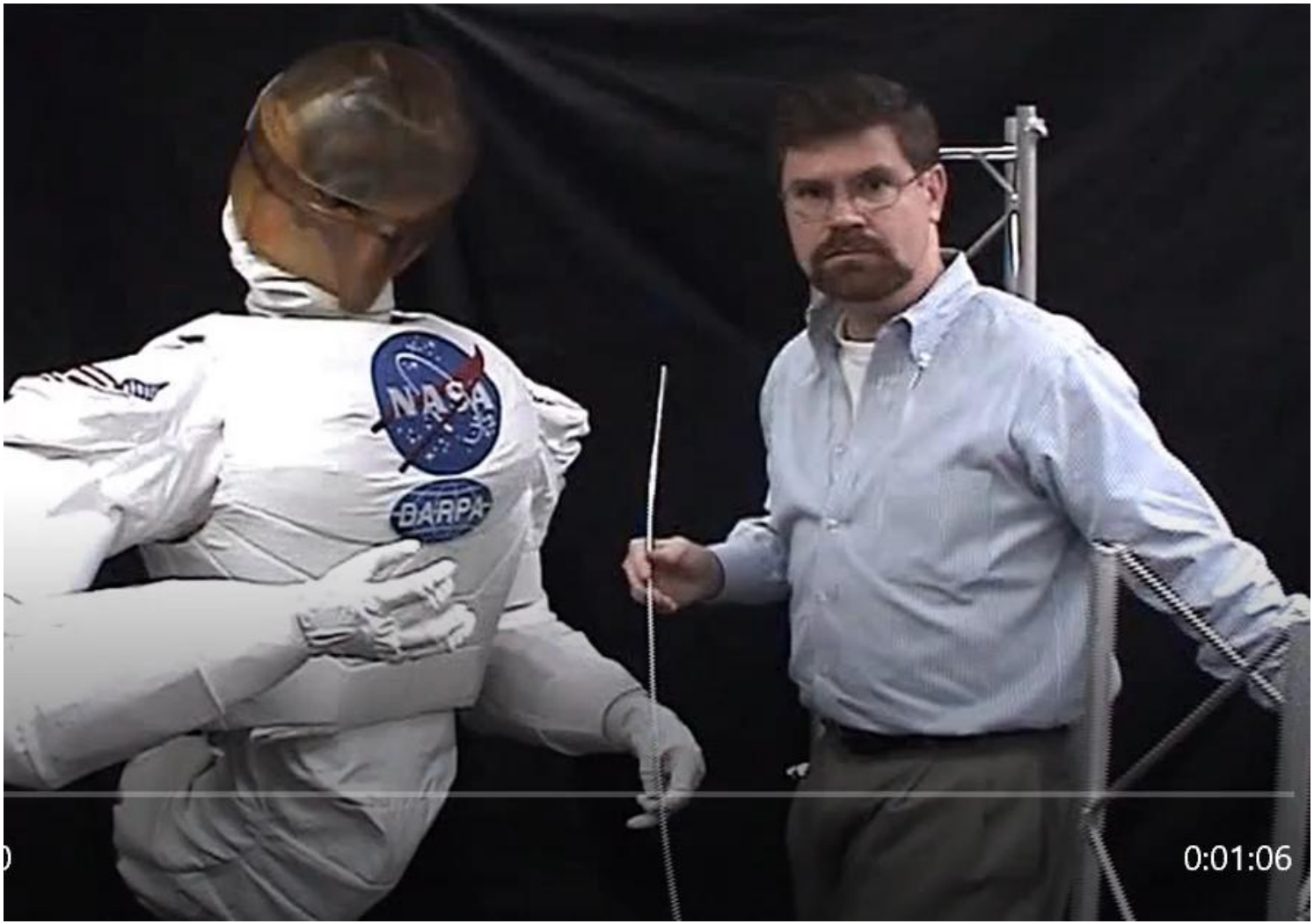
Blueprint of Human Brain



Characteristics of Human Brain and Mind

- Is able to memorize past, present and future
- Is able to capture input data
- Is able to undertake operations with numbers
- Is able to undertake operations with numbers/symbols
- Is able to produce output data
- Is able to control mental operations
- Is able to control physical actions
- Is able to communicate
- Is able to learn human languages
- Is able to learn machine languages
- Is able to learn new knowledge and new skills, etc.

Does a robot need to have a brain/Mind?

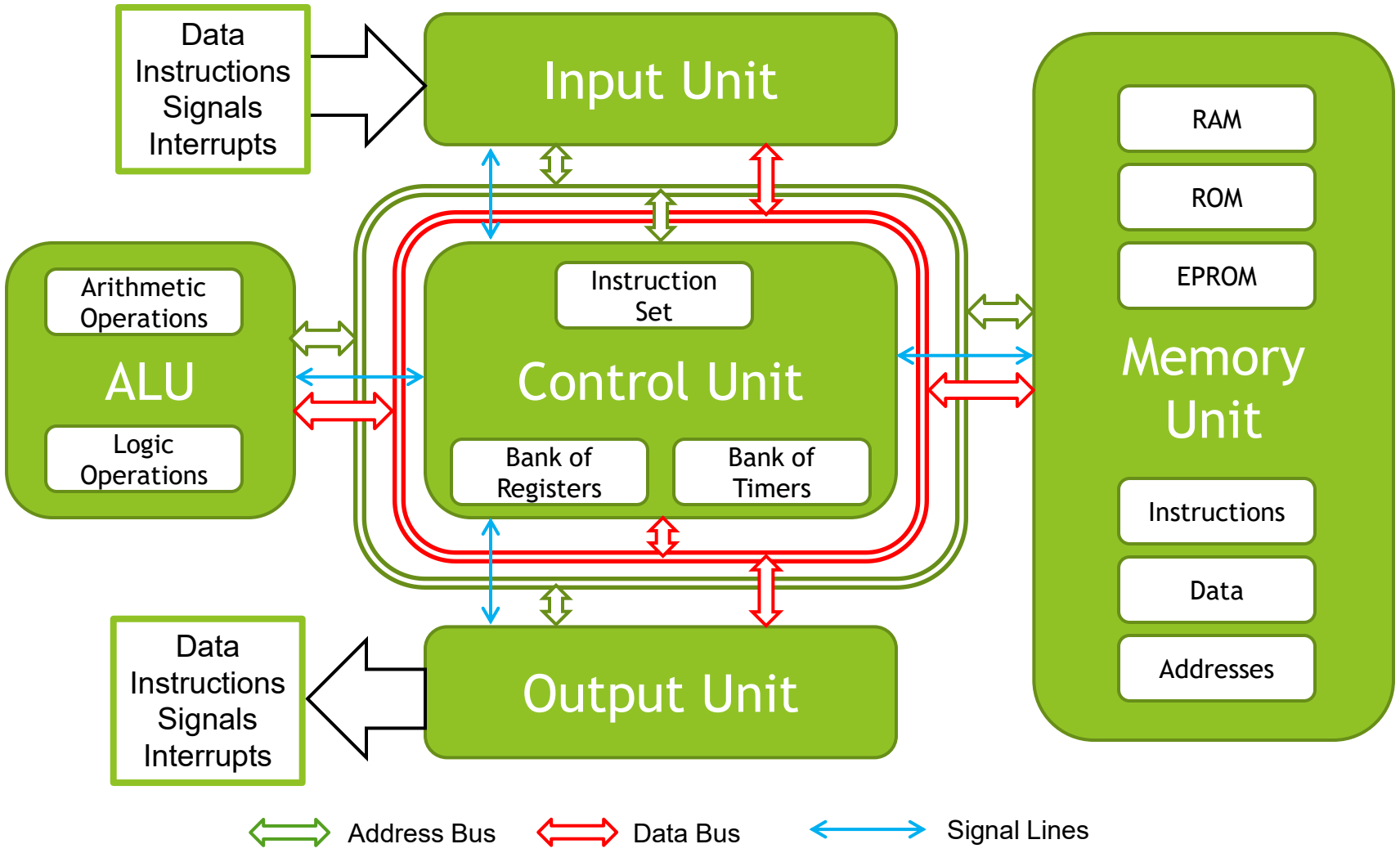


Example of Use of Robot Brain and Mind

- Person A asks robot B the following question: “Is it lunch time now?”. Explain the mental activities of robot B when it prepares the reply.
- Answer:

1. Read the current time T
2. Compare it with 12h00
3. If T > 12h00, compare it with 14h00
4. If 14h00 > T > 12h00, reply “yes”
5. Otherwise, reply “not yet”

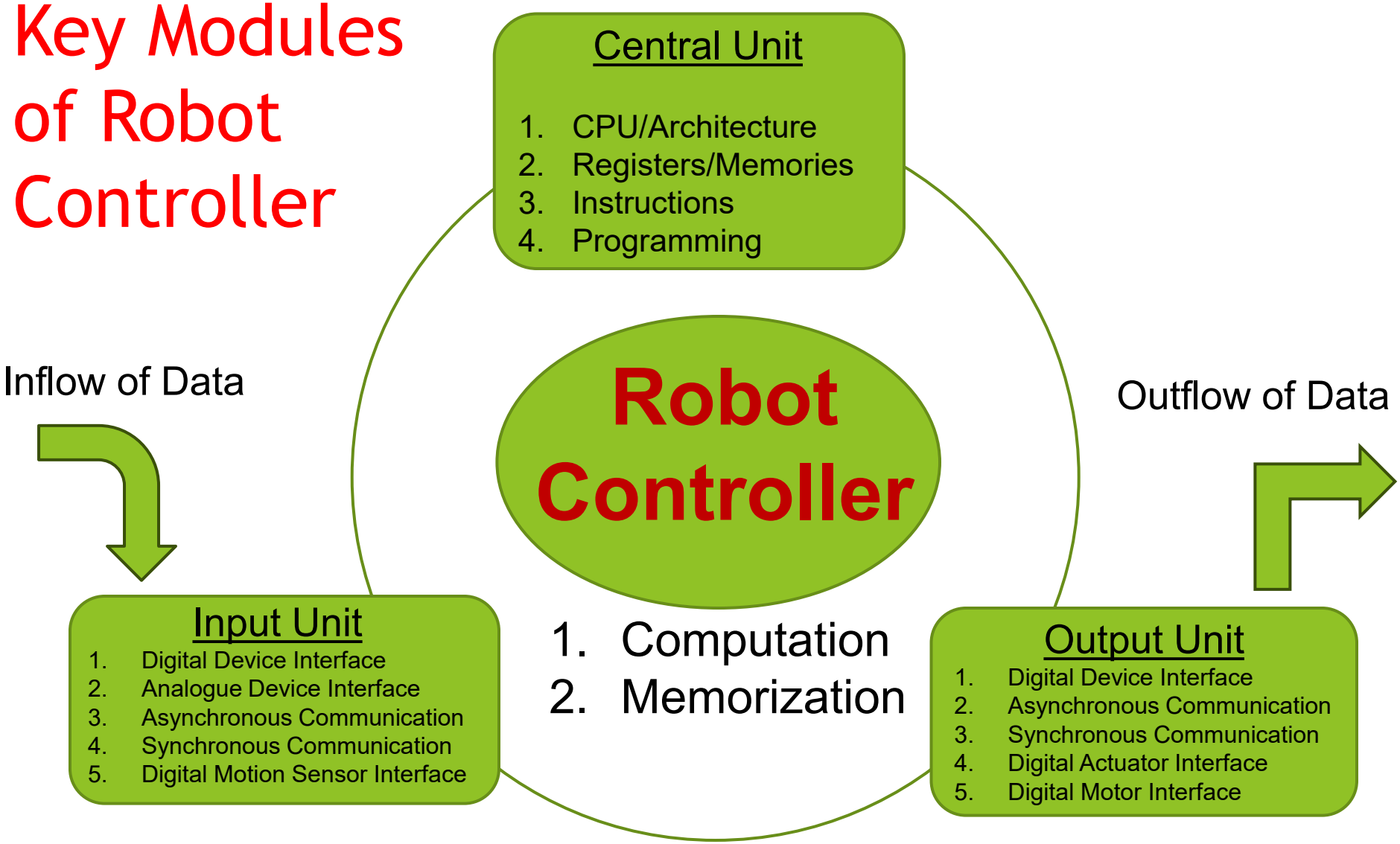
Blueprint of Robot Brain



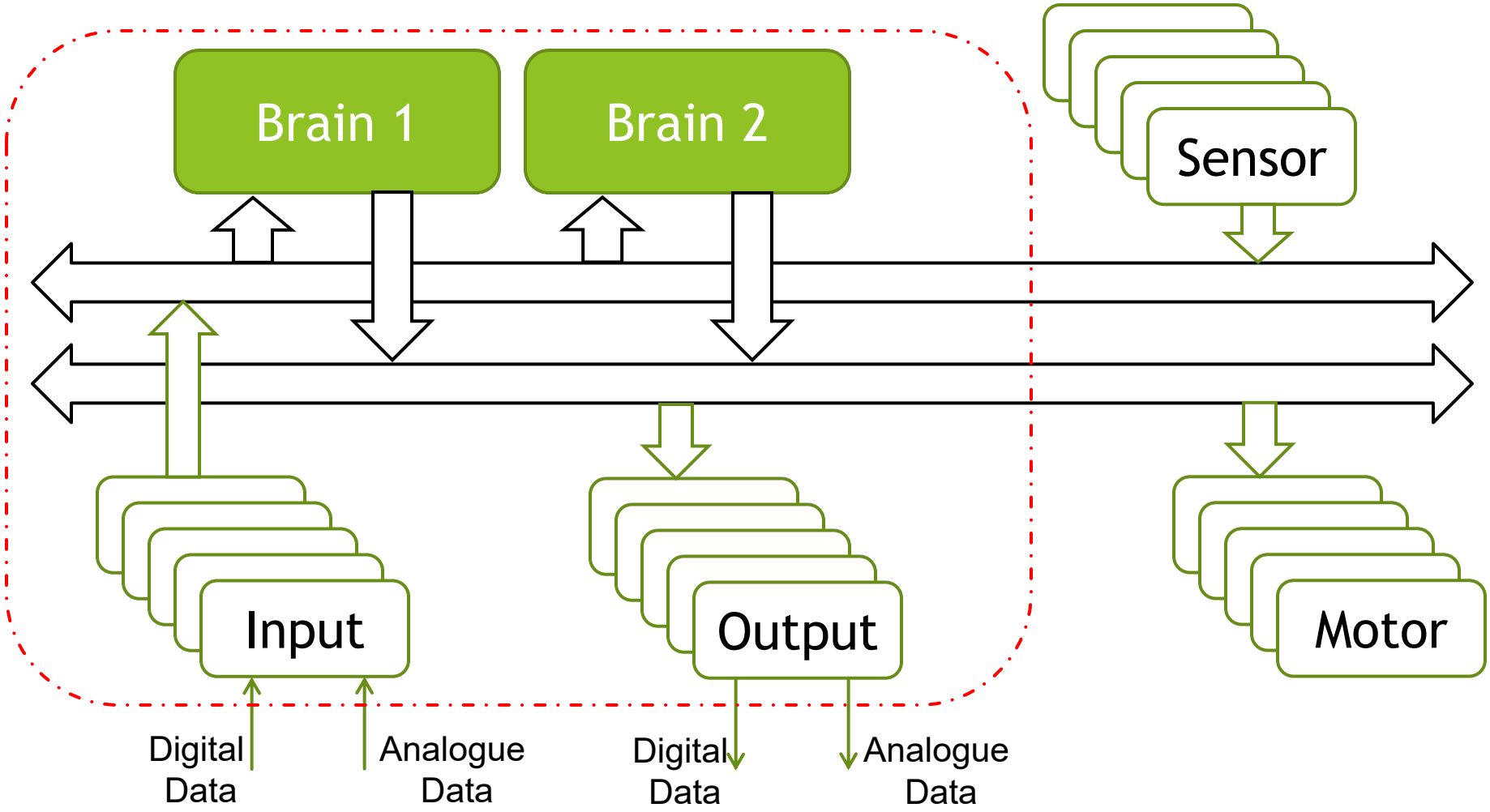
Characteristics of Robot Brain and Mind

- Is able to memorize data (to use) and instructions (to do)
- Is able to capture input data
- Is able to undertake arithmetic operations with numbers
- Is able to undertake logic operations with numbers/symbols
- Is able to produce output data
- Is able to control mental operations
- Is able to control physical actions
- Is able to communicate
- Is able to learn human languages (future)
- Is able to learn machine languages (future)
- Is able to learn new knowledge and new skills, etc. (future)

Key Modules of Robot Controller

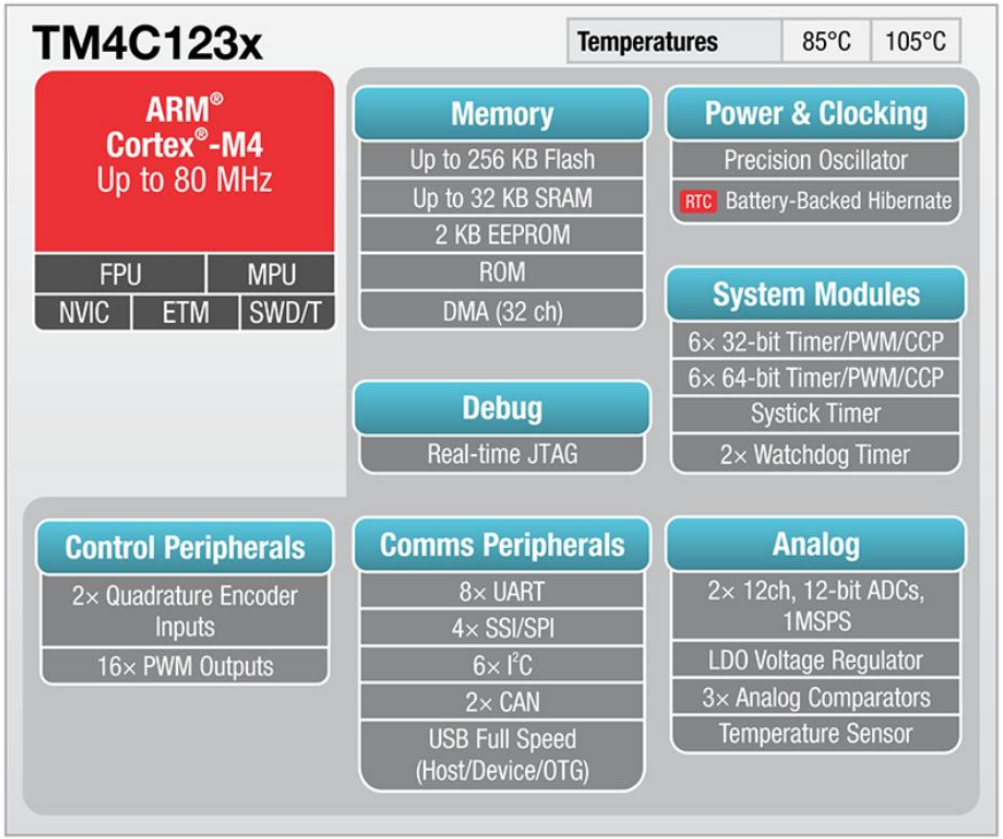


Extended Version of Robot Controller



How to use robot controller?

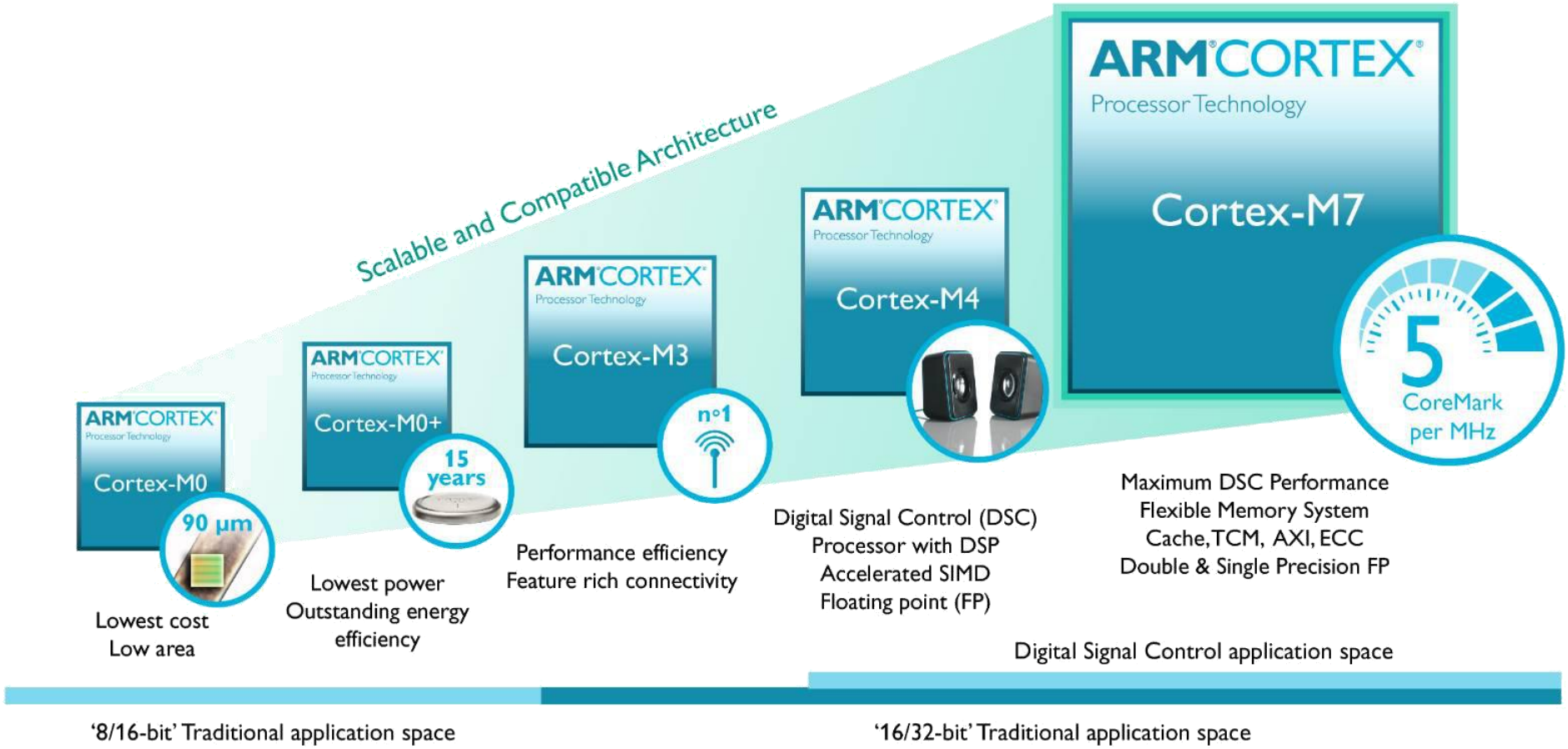
- Configure Control Registers
- Clear/Monitor Status Registers
- Read/Write Data Registers
- Instructions:
  - MOV <address of destination>, <source of value>
  - LDR <address of destination>, <source of value>
  - STR <source of value>, <address of destination>




```
MOV   R0, #0x11
MOV   R1, #2560
MVN   R2, #4
MOVW  R3, #0xC0DE
MOVT  R3, #0xFEED
MOV   R4, R1
```

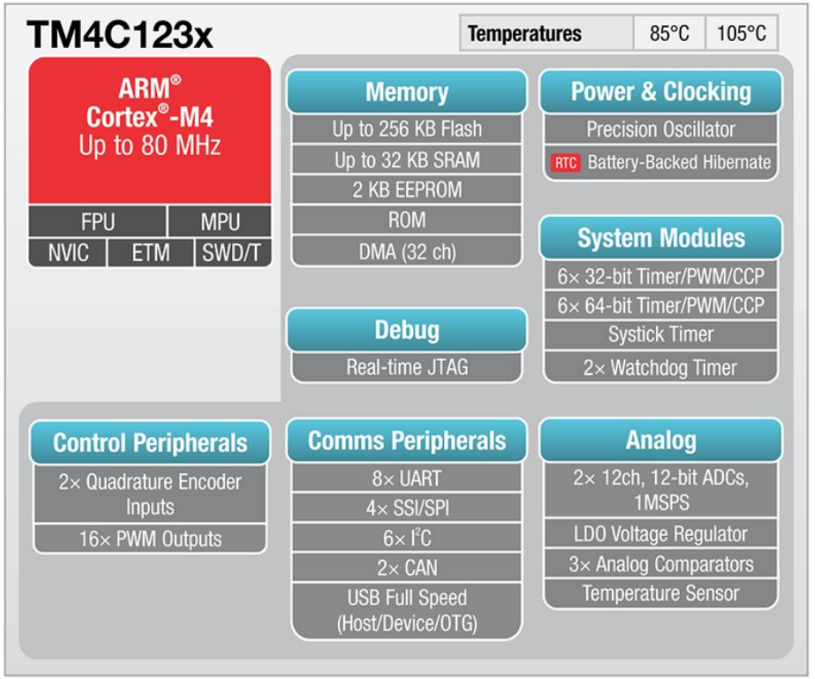
Word = 2 Bytes

Standard Devices to Serve as Robot Controller: Cortex M Series



Outline of Lecture 3

- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO



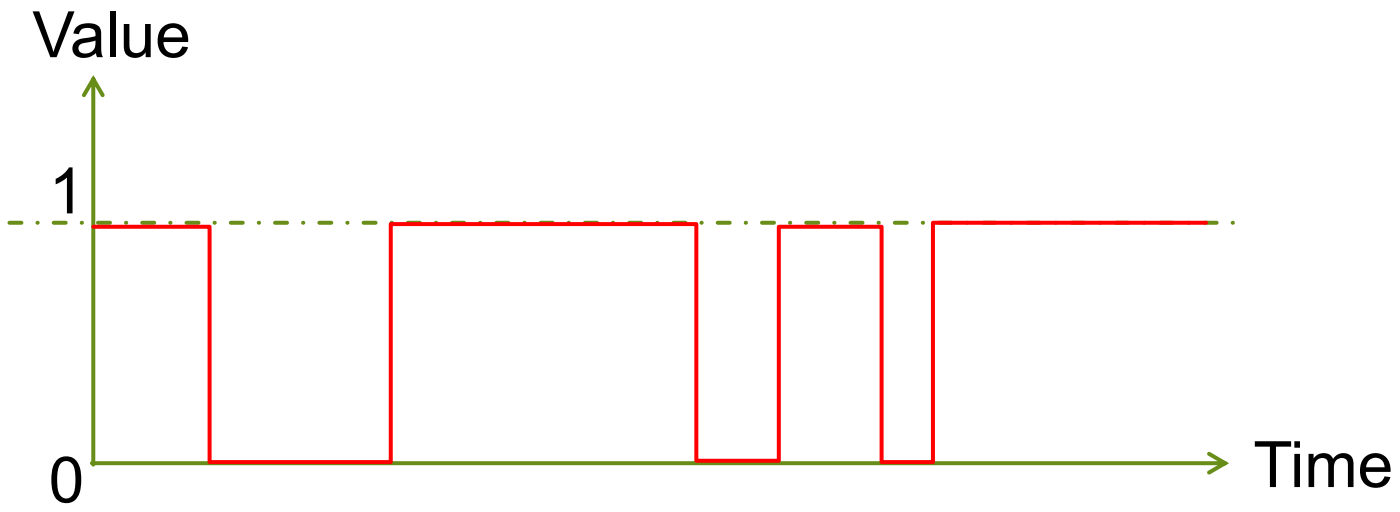
Binary Number Systems

Binary Values

- 1 (Logic High)
- 0 (Logic Low)

Implementation

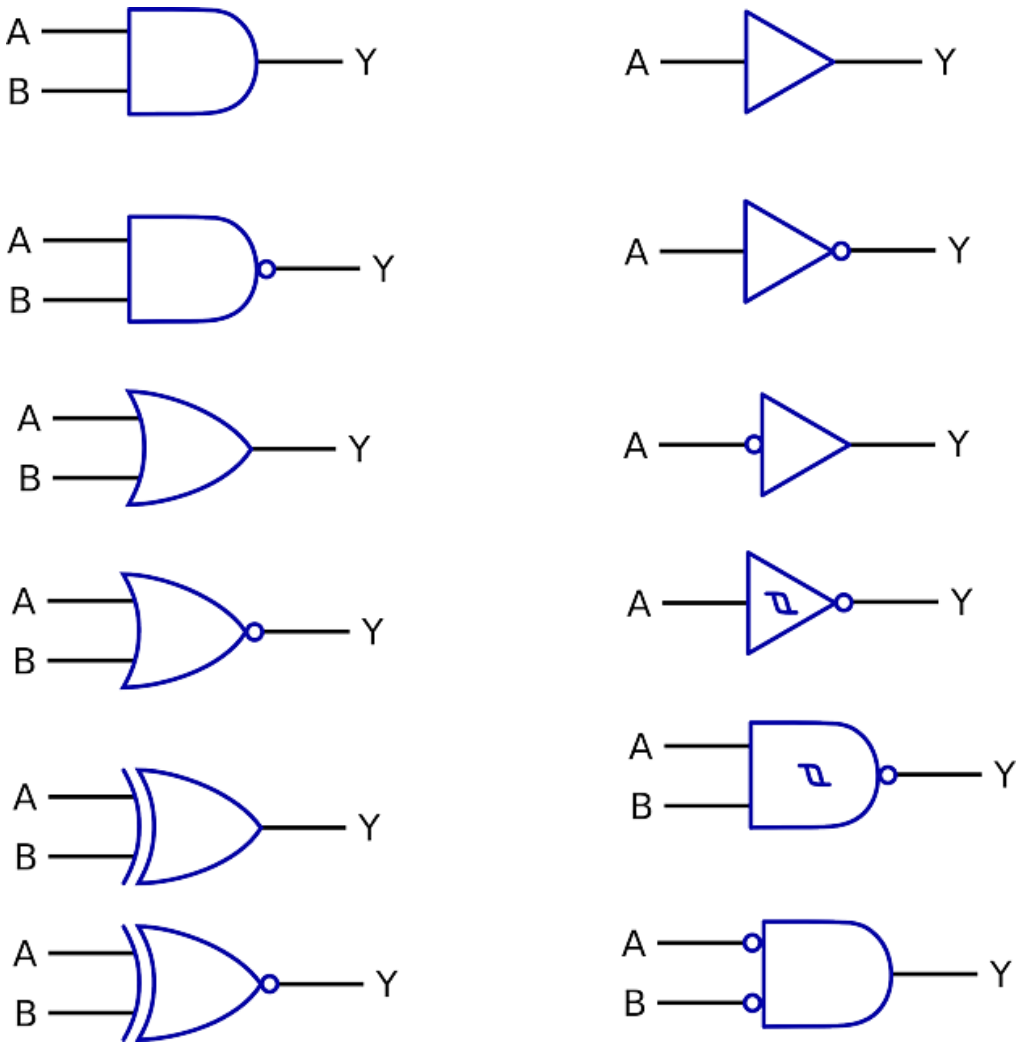
- +5 V (Logic High)
- 0 V (Logic Low)



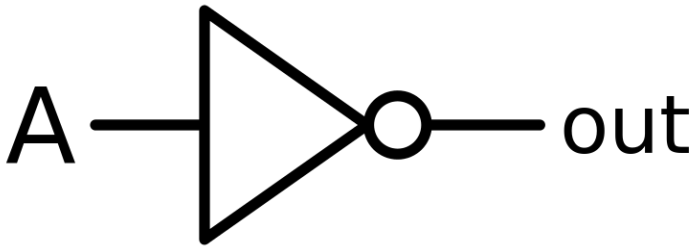
Binary Logic Operations

NOT	A' $\neg A$ $\overline{A}$ $\neg A$
AND	AB $A*B$ $A \bullet B$ $A \wedge B$ $A \cap B$
OR	$A+B$ $A \vee B$ $A \cup B$
NAND	$(AB)'$ $\overline{AB}$
NOR	$(A+B)'$ $\overline{A+B}$
XOR	$A \oplus B$ $A \mathbin{@} B$
XNOR	$(A \oplus B)'$ $\overline{A \oplus B}$ $(A \mathbin{@} B)'$ $\overline{A \mathbin{@} B}$

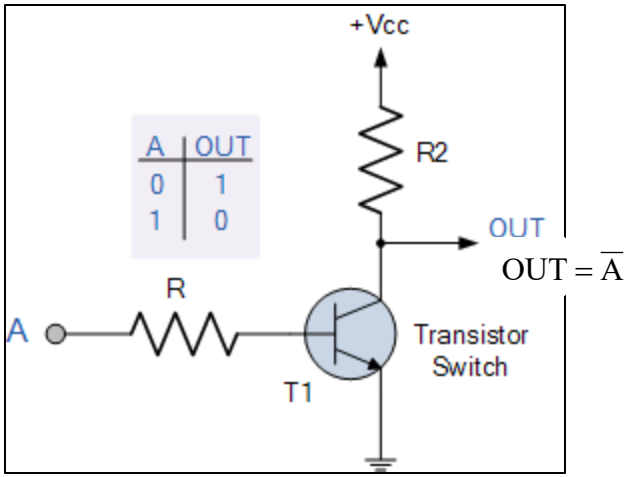
Binary Logic Devices



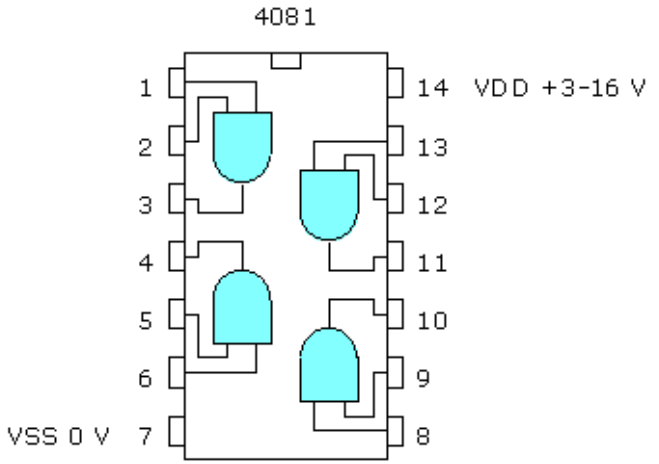
Example: NOT



A	$\overline{A}$
1	0
0	1

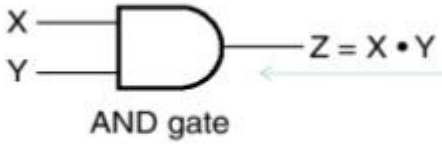


Example: AND



AND Gate

- Logic Symbol, Truth Table And Logic Expression

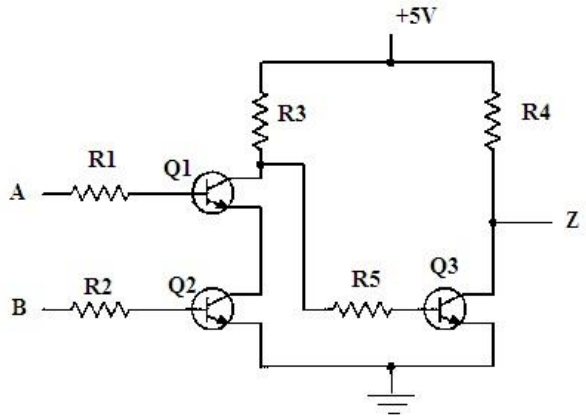


Logic Symbol

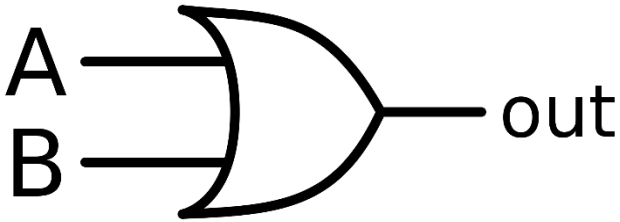
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

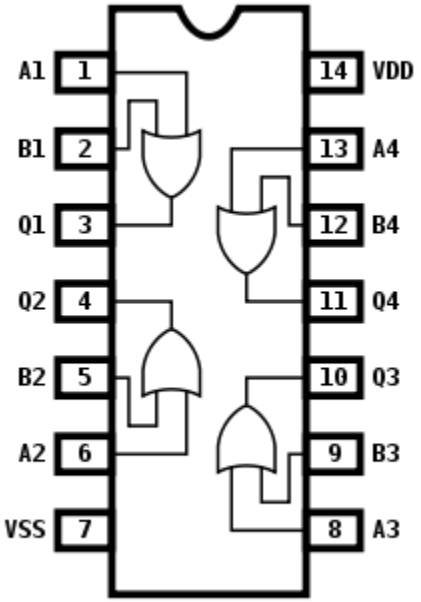
Logic Expression

Truth Table

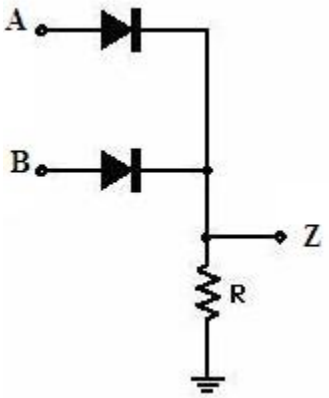


Example: OR



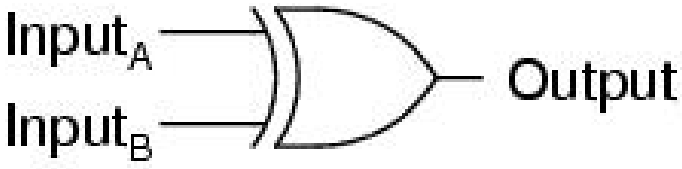


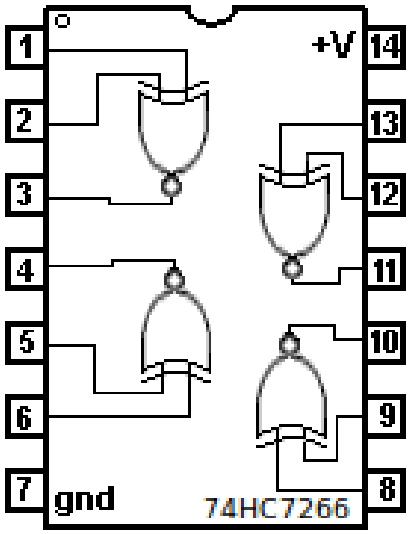
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



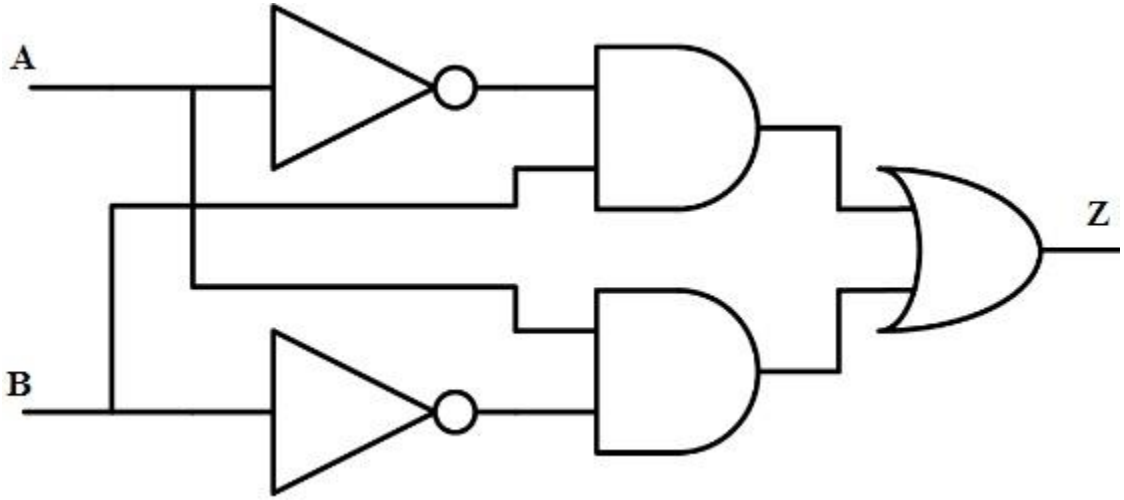
Example: XOR

Exclusive-OR gate





A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

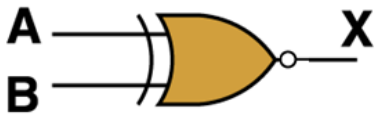


Example: XNOR

Boolean Expression

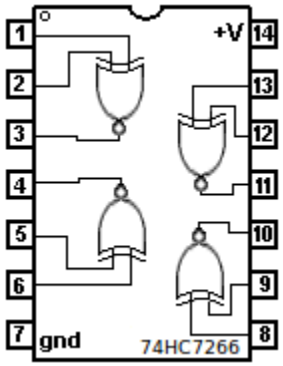
$$X = \overline{A \oplus B}$$

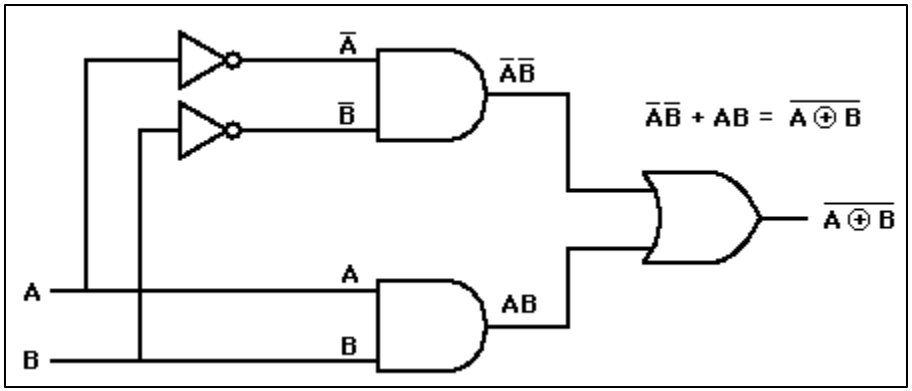
Logic Diagram Symbol



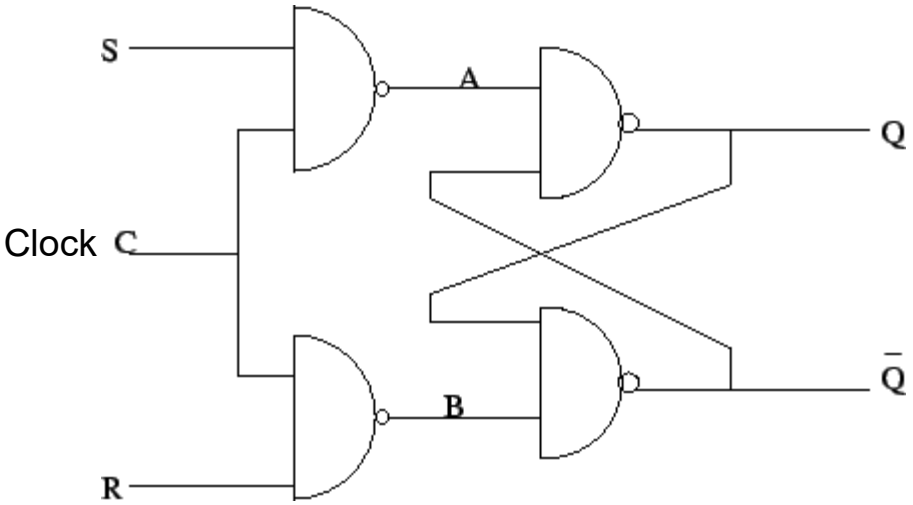
Truth Table

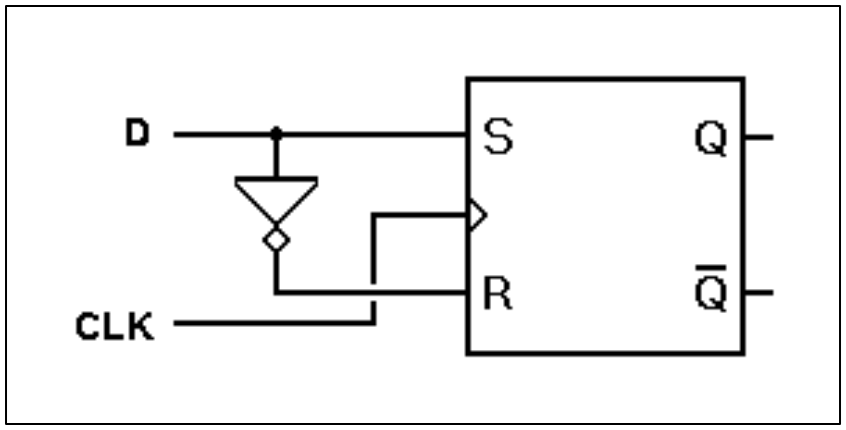
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1





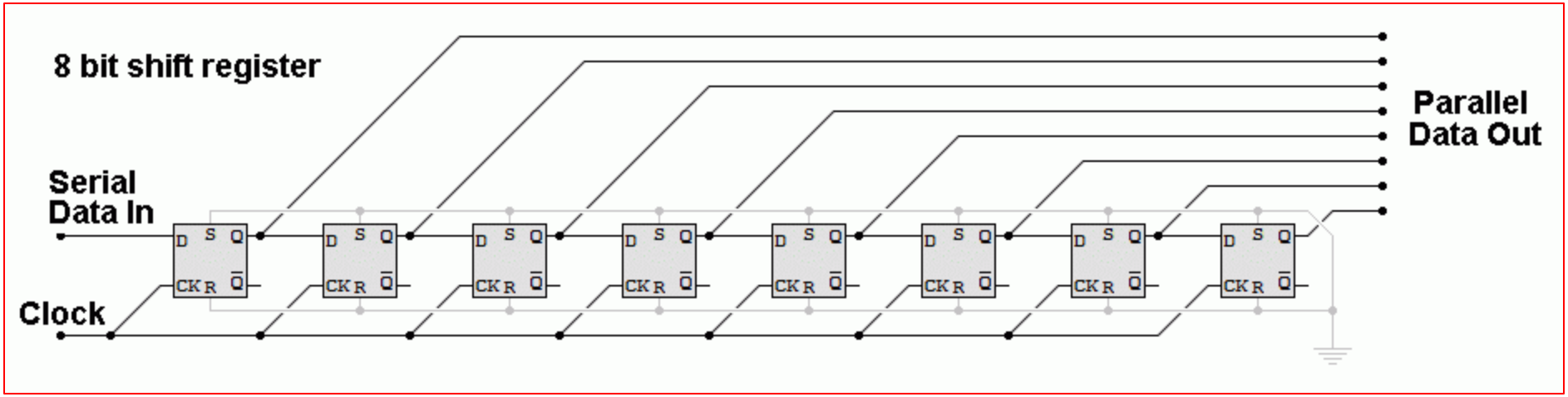
Register: Single Bit Device

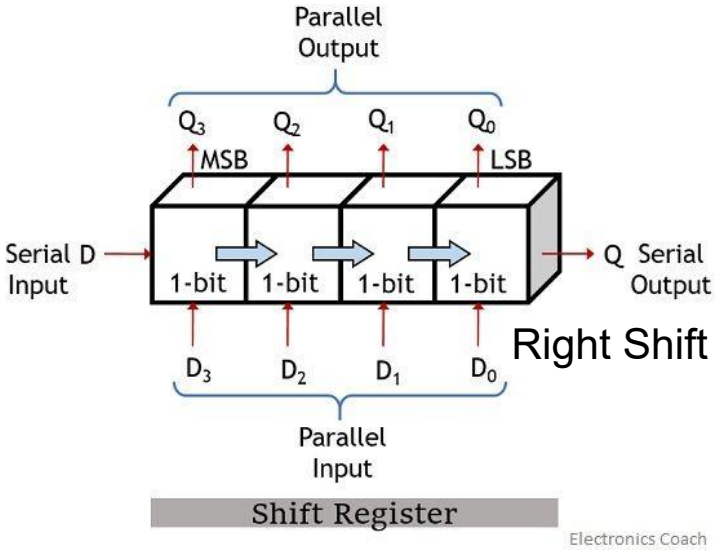


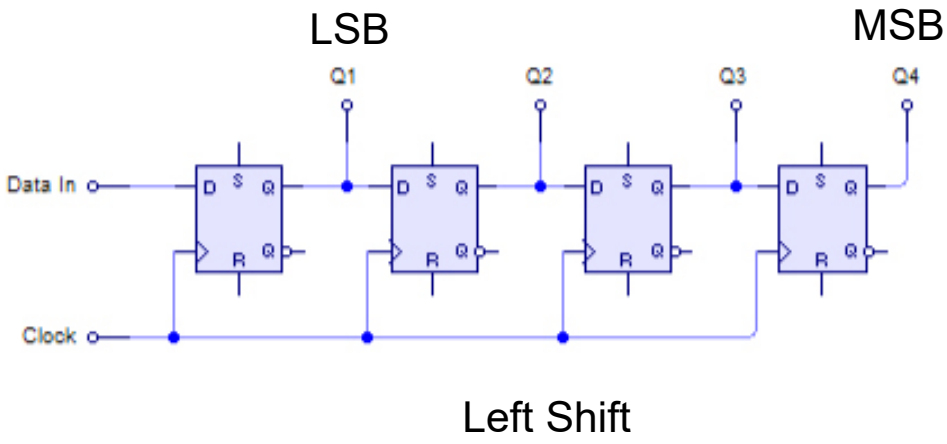


S	R	CLK	Q(t+1)	Comments
0	0	X	Q(t)	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	?	Invalid

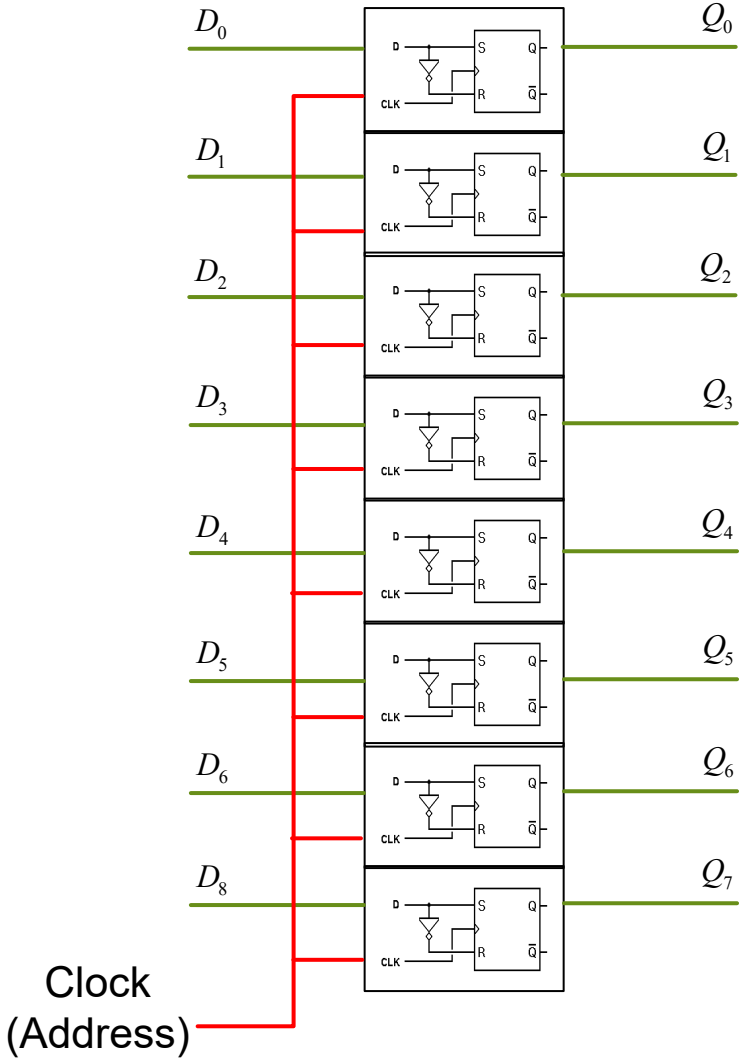
Register: 8-Bit Shift Register



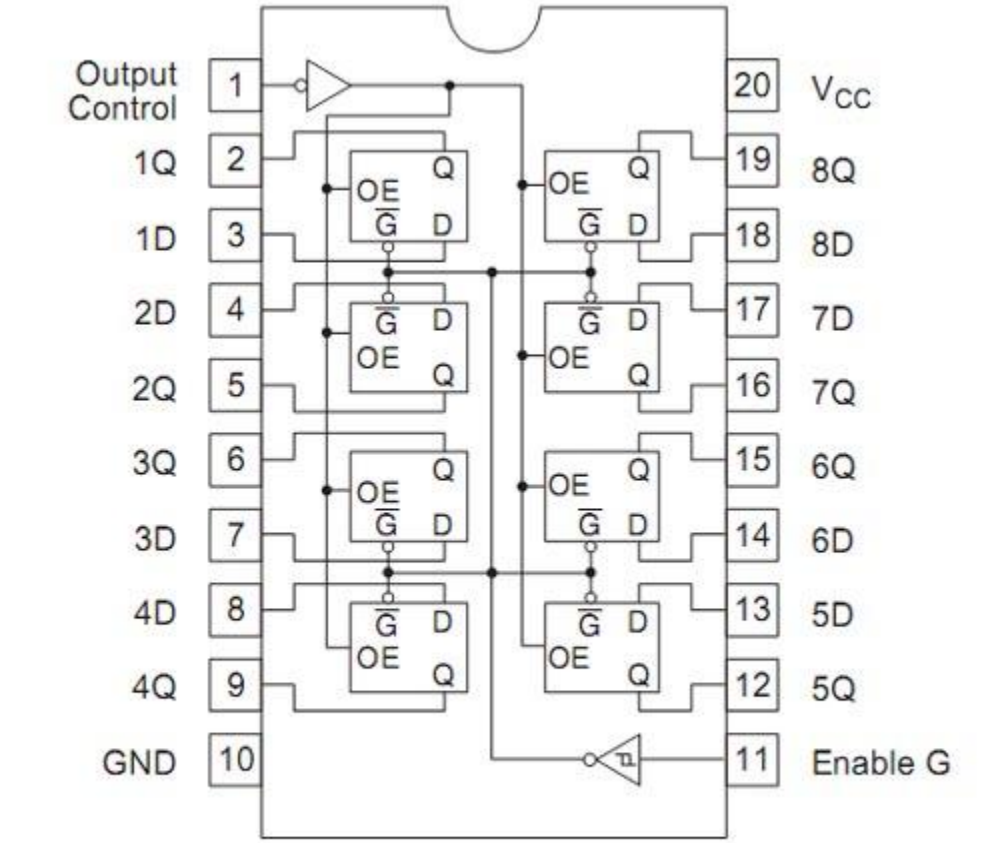




Register: 8-Bit Register (One Byte)

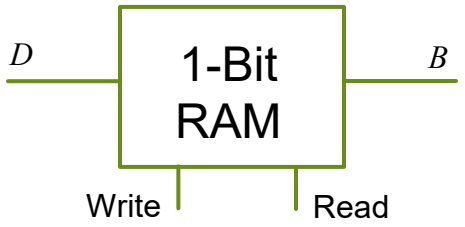


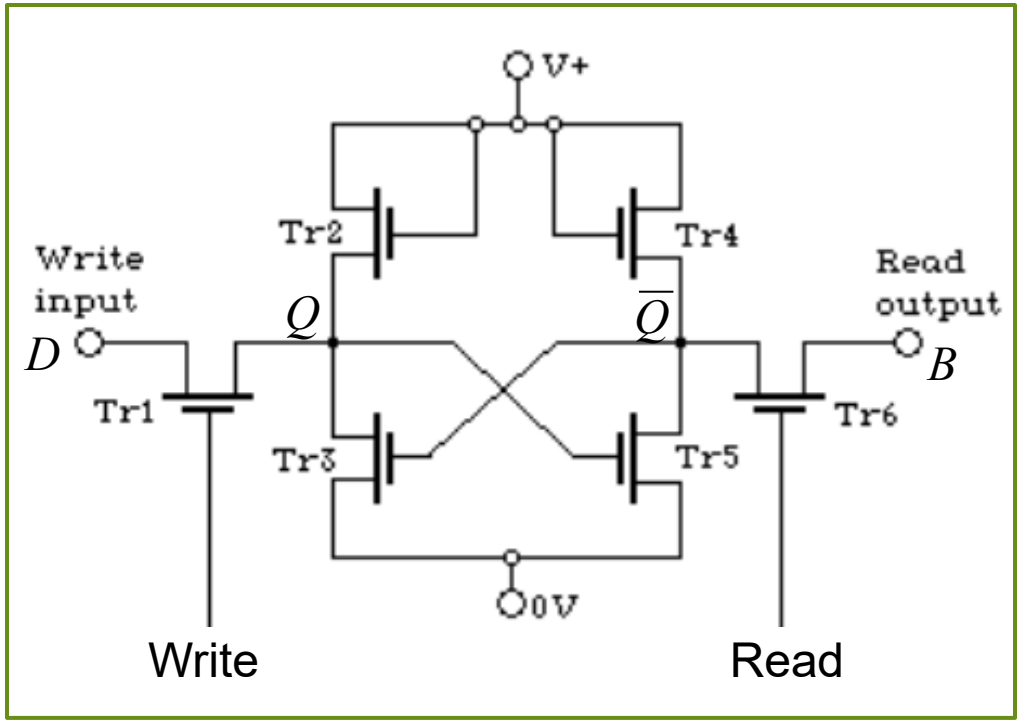
Example of 8-Bit Register (One Byte)



RAM: Single Bit Static RAM

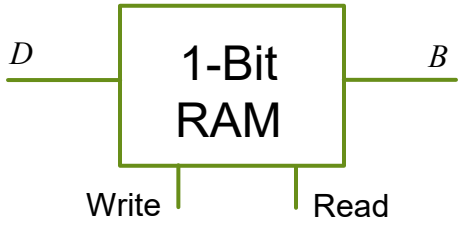
- Write Operation
  - "Write" line is in logic high.
  - Transistor Tr1 is on.
  - Q = D
- Read Operation
  - "Read" line is in logic high.
  - Transistor Tr5 is on.
  - B = 1 - Q



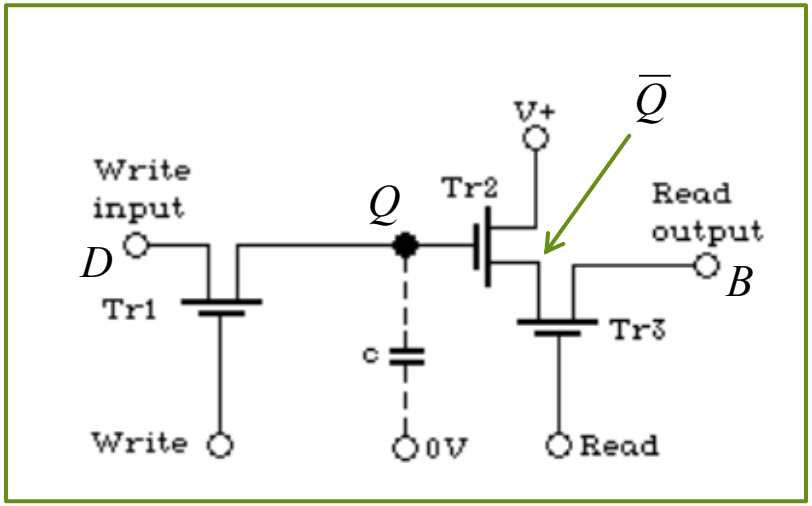


RAM: Single Bit Dynamic RAM

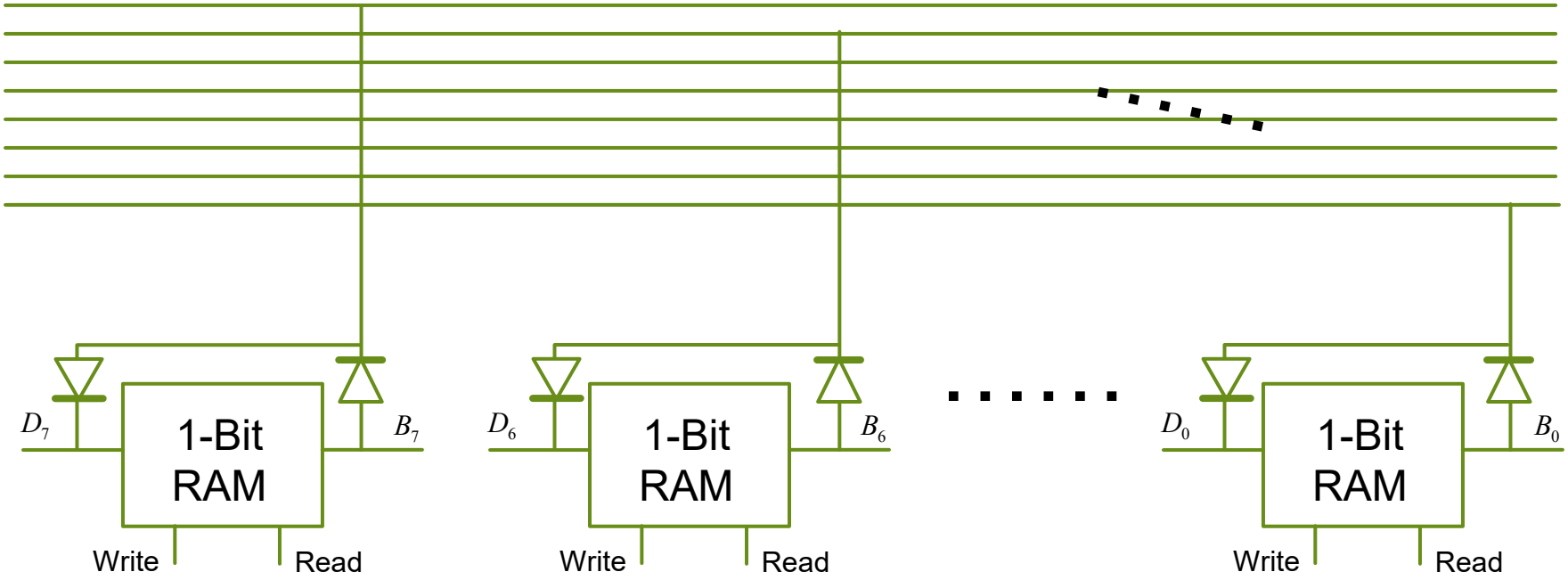
- Write Operation
  - "Write" line is in logic high.
  - Transistor Tr1 is on.
  - Q = D



- Read Operation
  - "Read" line is in logic high.
  - Transistor Tr3 is on.
  - B = 1 - Q

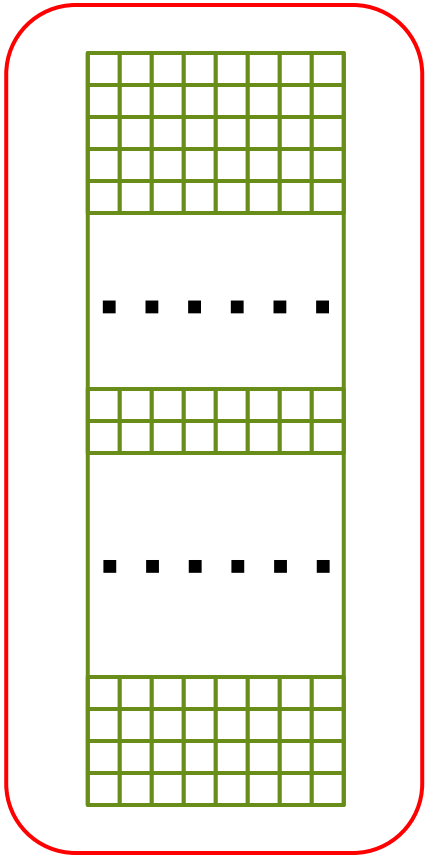


RAM: 8-Bit RAM (One Byte)



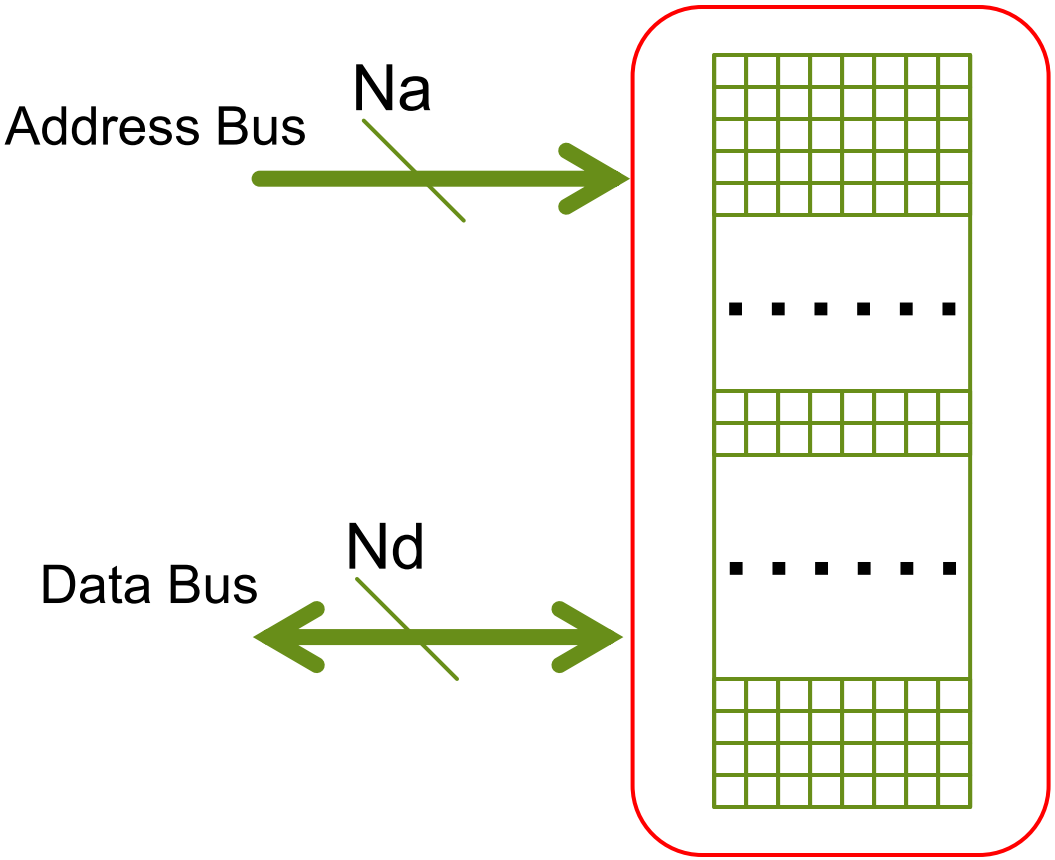
Blueprint of Memory Unit

- A memory unit consists of a set of bytes. Each byte has eight bits. Each bit has one memory cell for read and write operations.



Input to Memory Unit

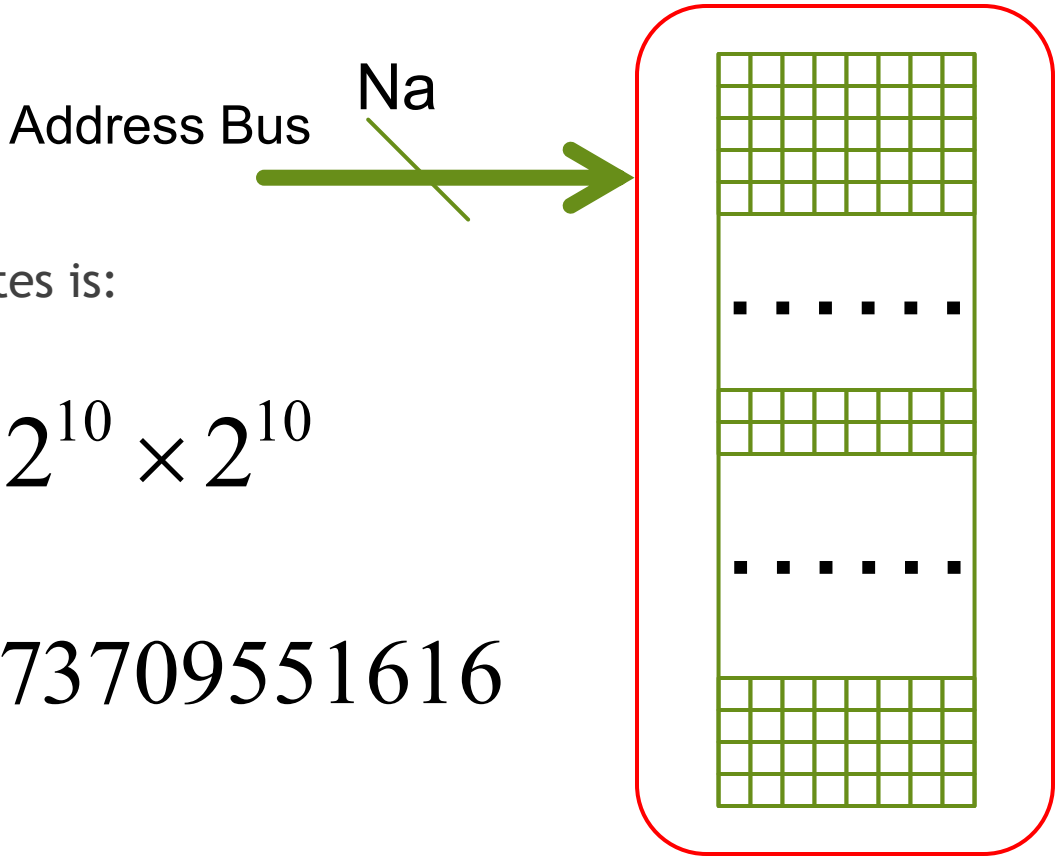
- Input of memory unit includes: a) address of a byte, and b) data of one or more bytes.



Example

- A memory unit's address bus has 64 bits. Hence, Na = 64. What is the maximum number of bytes that the memory unit could have?

- Answer:



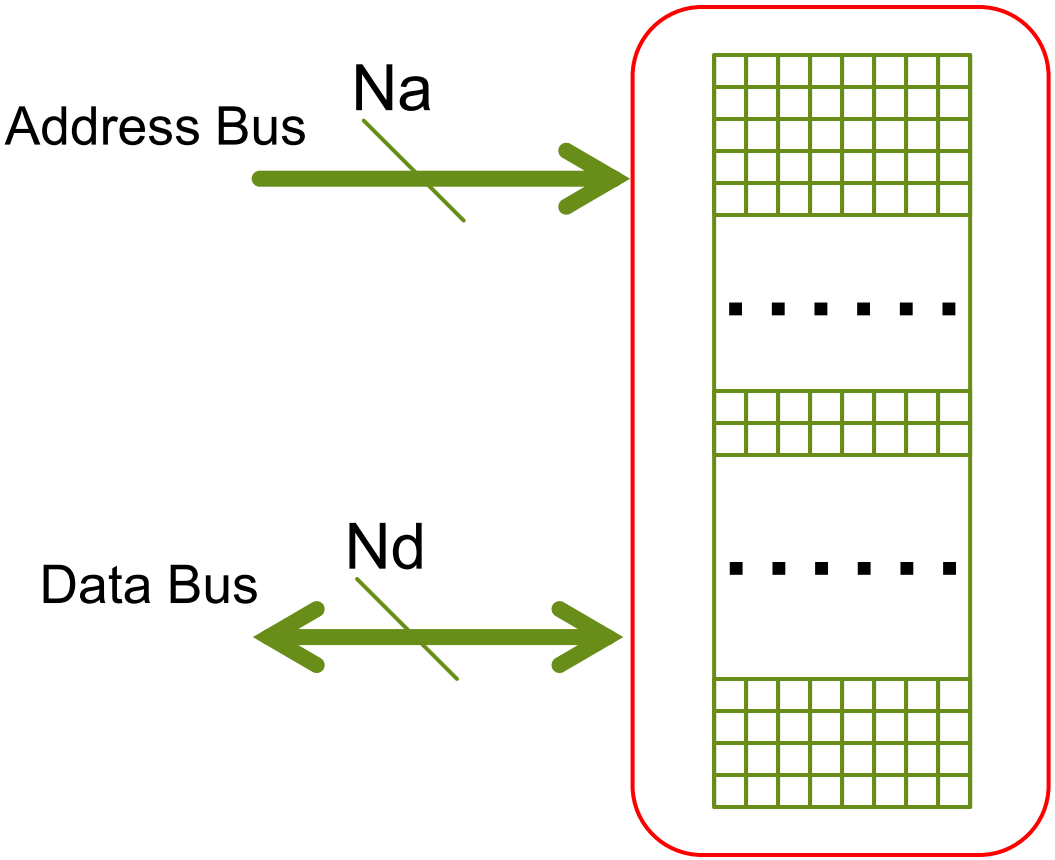
- The maximum number of bytes is:

$$2^{64} = 2^{34} \times 2^{10} \times 2^{10} \times 2^{10}$$

$$2^{64} = 18446744073709551616$$

Output from Memory Unit

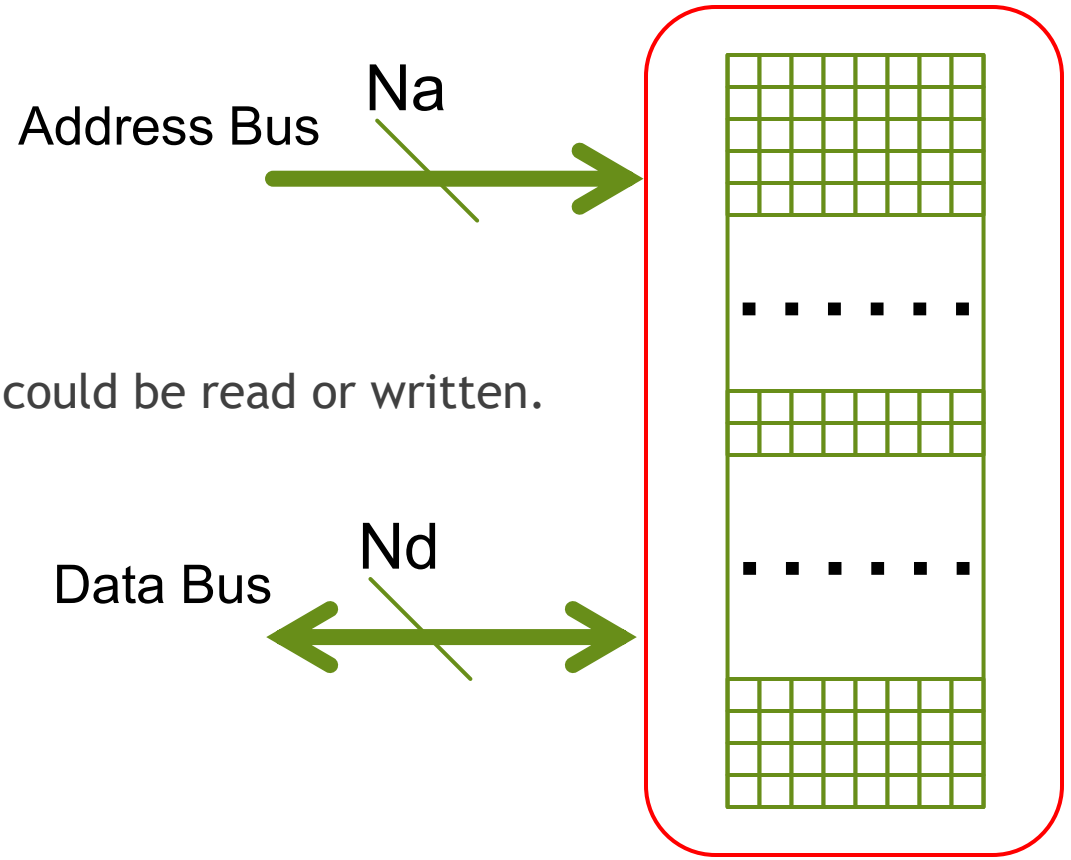
- The output from memory unit is data of one or more bytes.



Example

- A memory unit's address bus has 64 lines. And, its data bus has 32 lines. Hence, Na = 64 and Nd = 32. At one time, how many bytes could the memory unit read or write?

- Answer:



- At one time, 32/8 = 4 bytes could be read or written.

Activities of Writing Programs Include "Management of Memory?

Values of Data

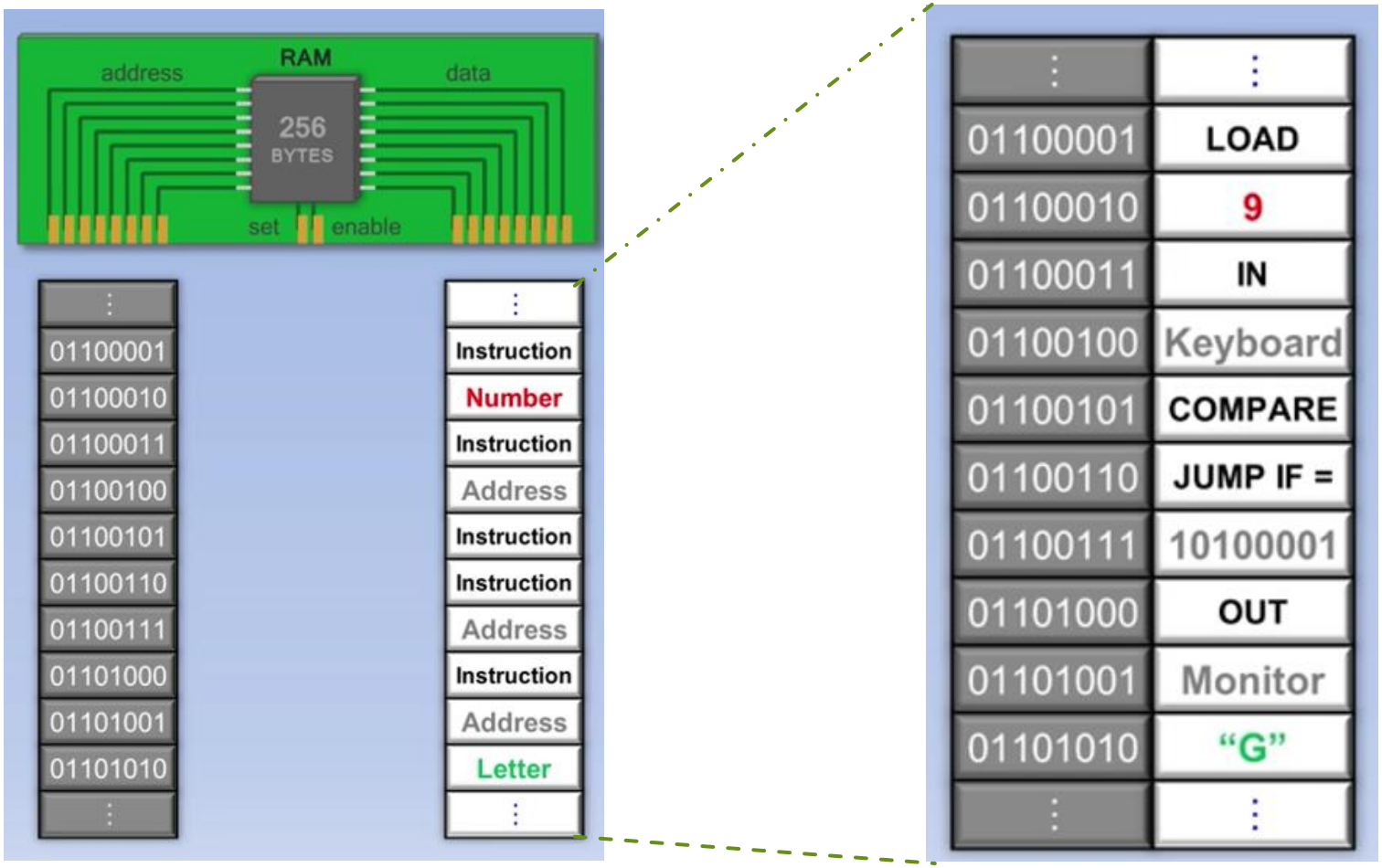
- int x, y, z ;
- x = 10.0 ;
- y = 5.0 ;
- z = x + y ;

Values of Addresses

- int *x, *y, *z, a, b, c ;
- x = &a; *x = a;
- y = &b; *y = b;
- z = &c ; *z = c;

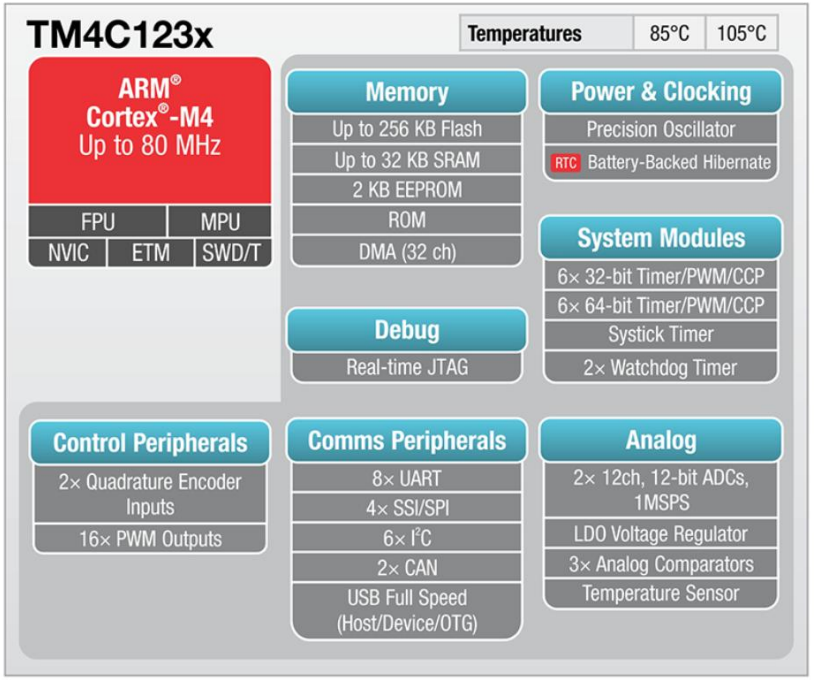
An *ampersand* is a logogram "&" representing the conjunction word "and". In C, it means the address of a variable.

Activities of Writing Programs Include "Management of Computation?

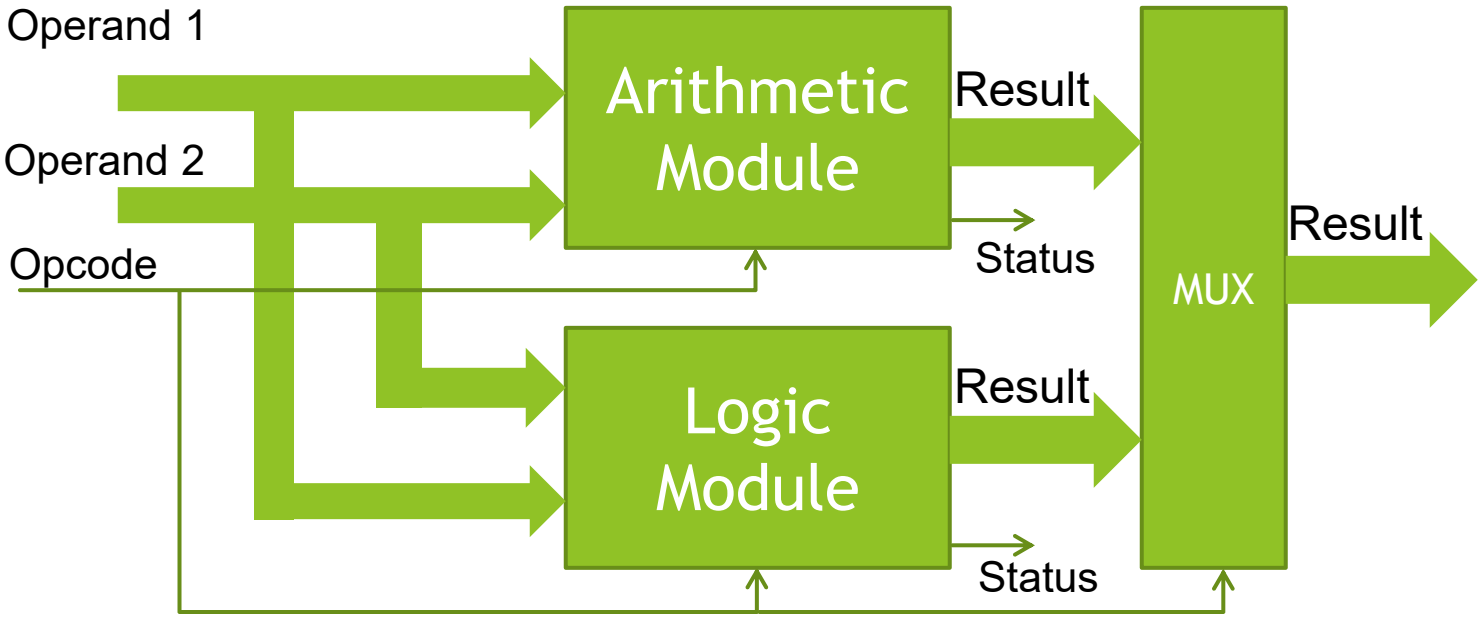


Outline of Lecture 3

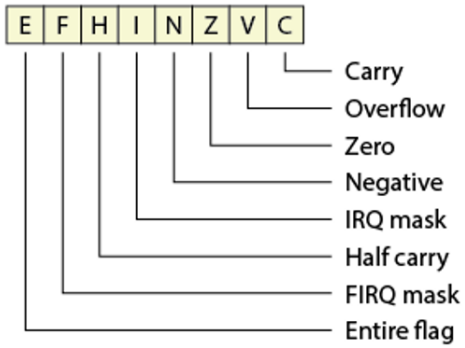
- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO



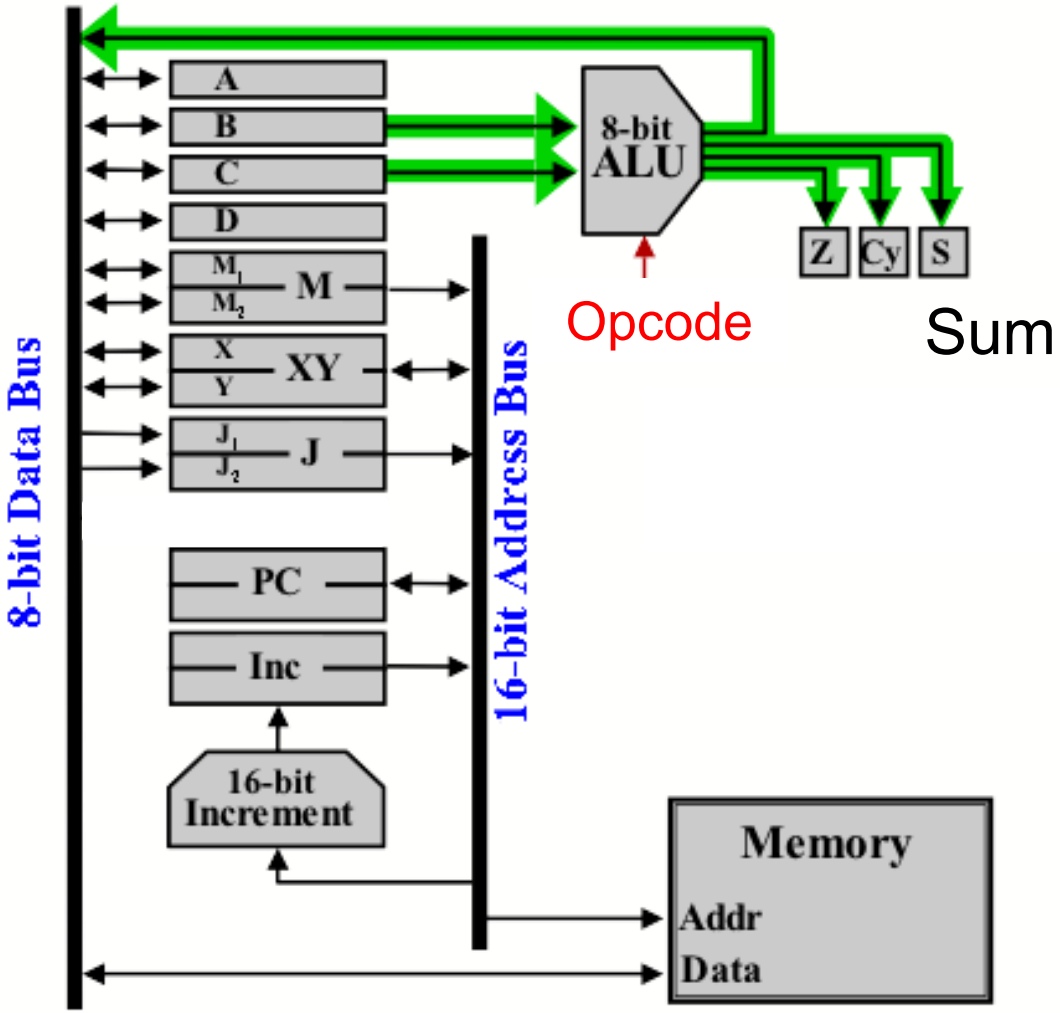
Blueprint of ALU



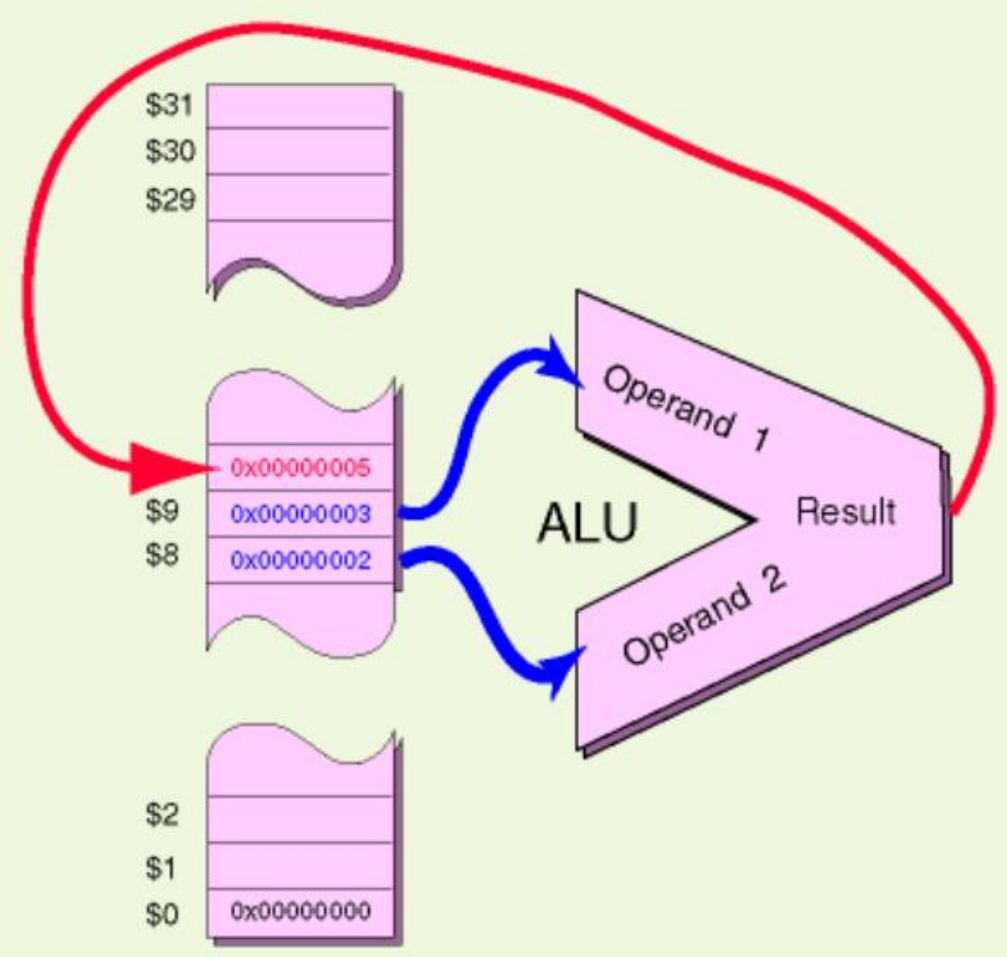
Status Register



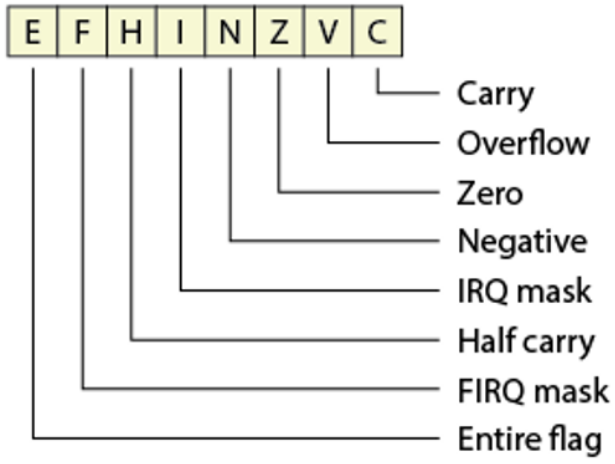
Design Example of 8-Bit ALU



Working Principle of ALU

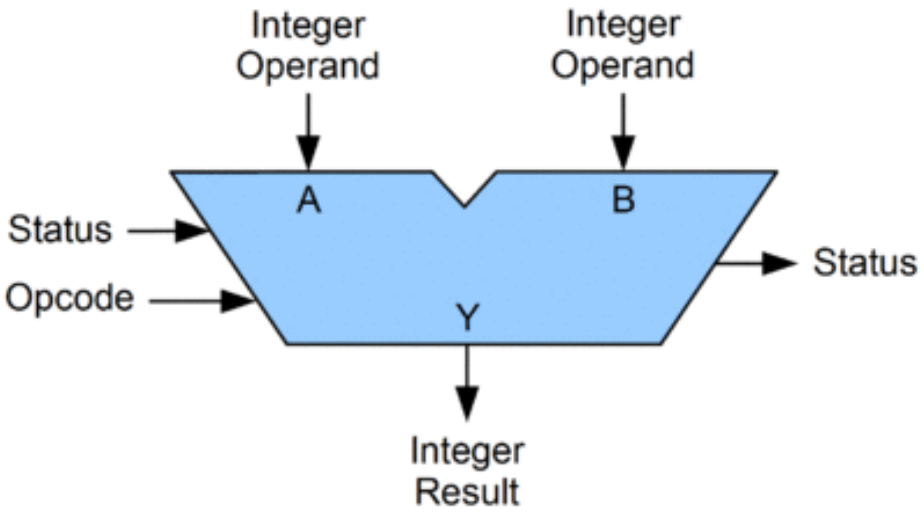


Status Register



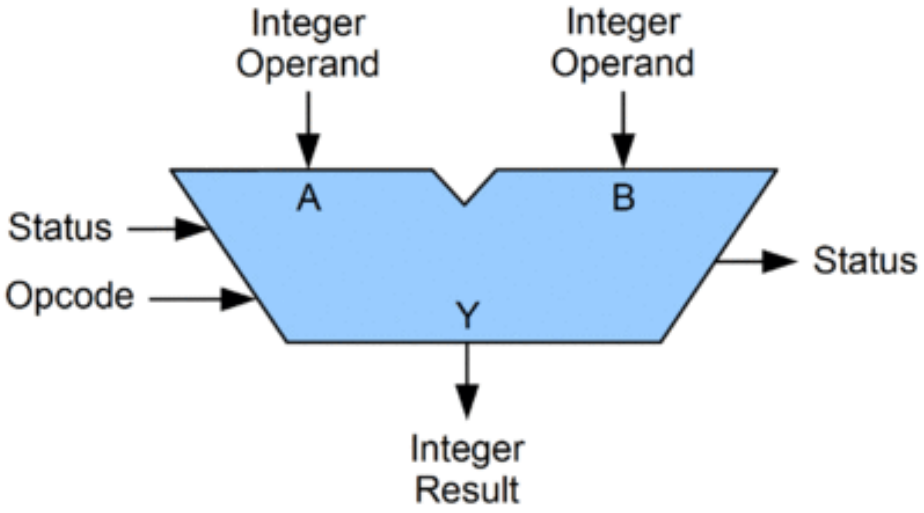
Input of ALU

- Operand A
- Operand B
- Opcode
  - Add
  - Subtraction
  - Multiplication
  - Division
  - Pass, Not, And, Or, Xor
- Status (e.g. carry bit)



Output of ALU

- Result Y
- Status flag bits
  - Overflow
  - Carry
  - Negative
  - Zero



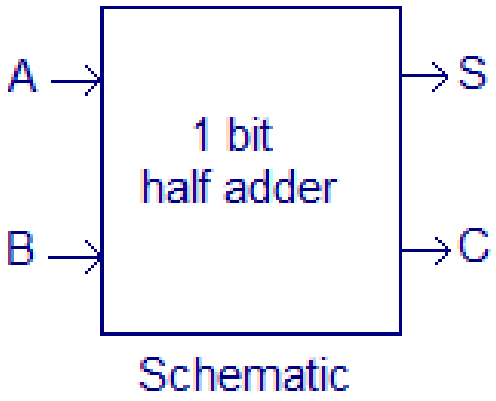
Arithmetic Operations + Logic Operations

Example of Arithmetic Operation:

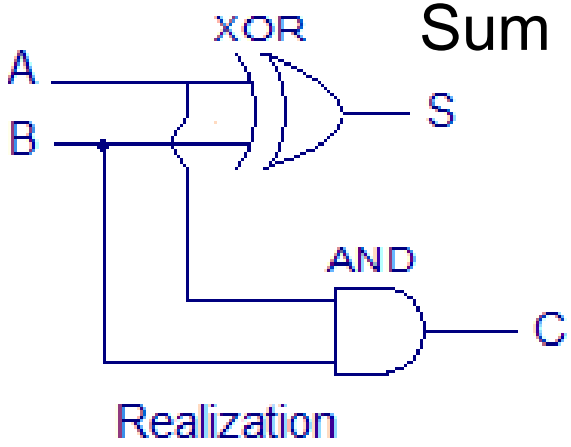
1-bit Half Adder

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



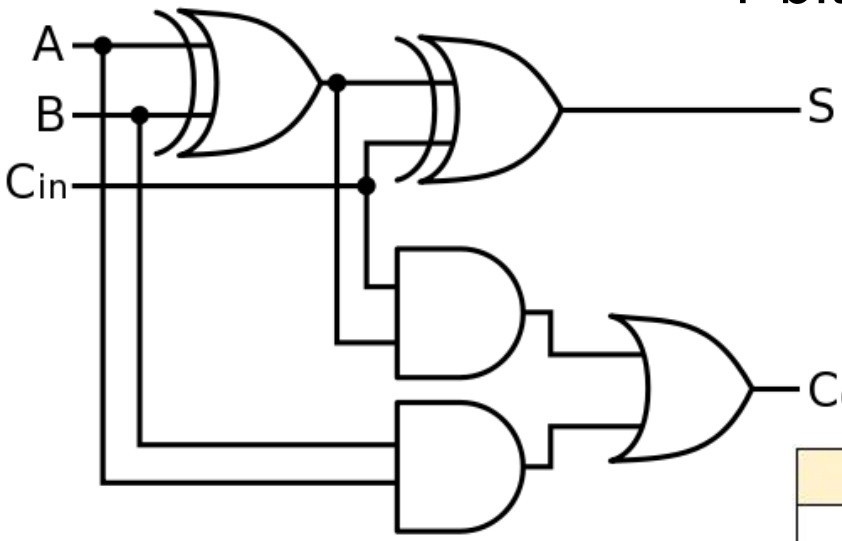
Schematic



Realization

Design of Arithmetic Operation:

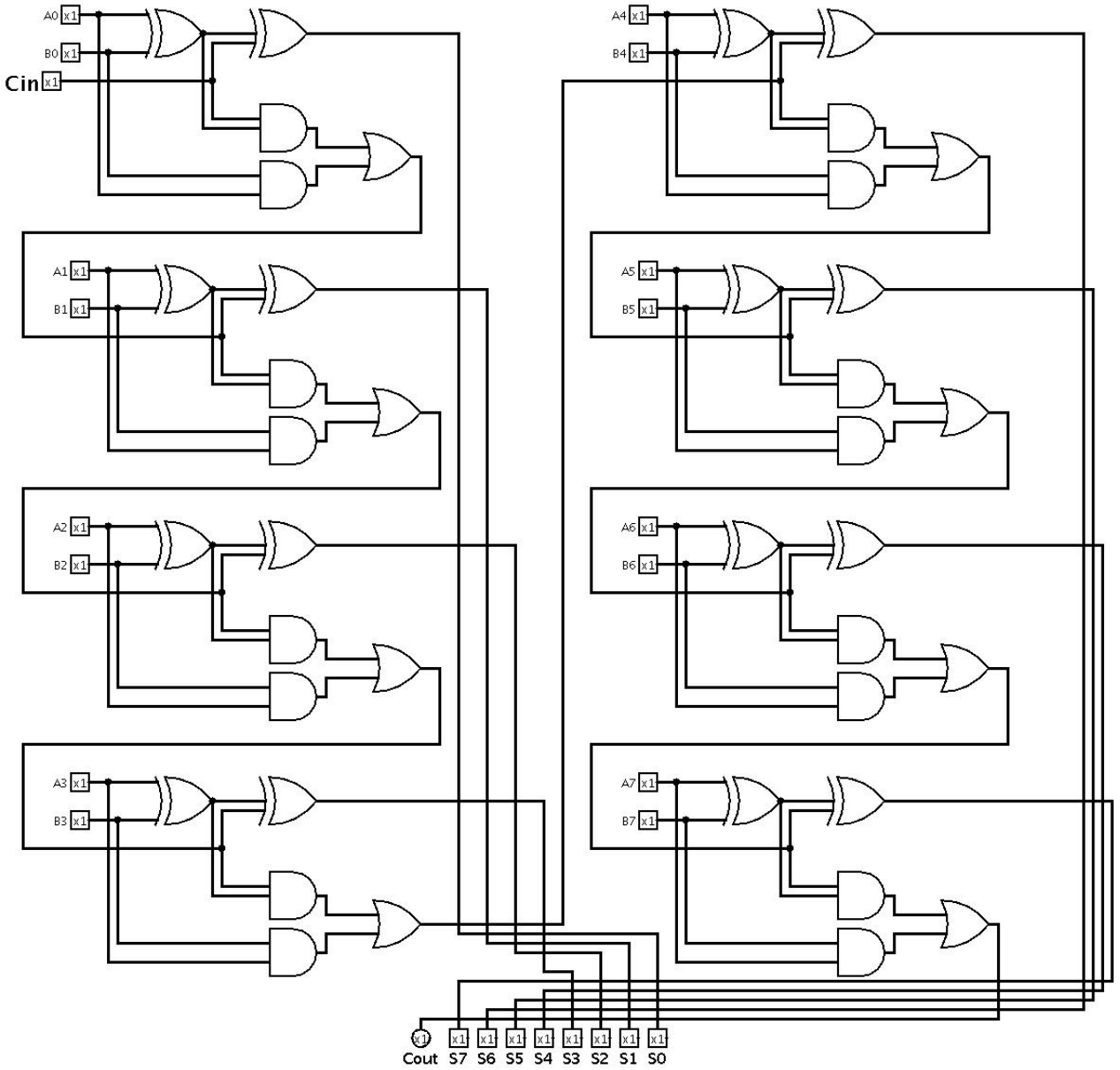
1-bit Full Adder



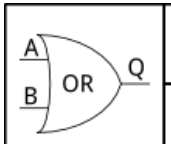
A	B	Carry-In	Sum	Carry-Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Example of Arithmetic Operation:

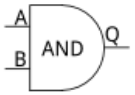
8-bit Full Adder



Examples of Logic Operations



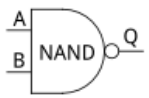
B \ A	0	1
0	0	1
1	1	1



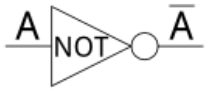
B \ A	0	1
0	0	0
1	0	1

B \ A	0	1
0	0	1
1	1	0

B \ A	0	1
0	1	0
1	0	0



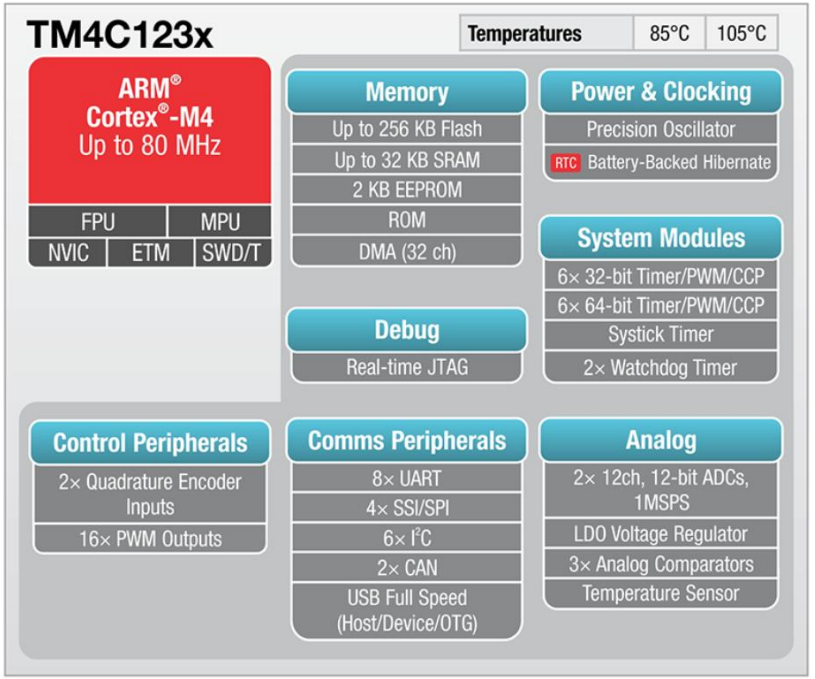
B \ A	0	1
0	1	1
1	1	0



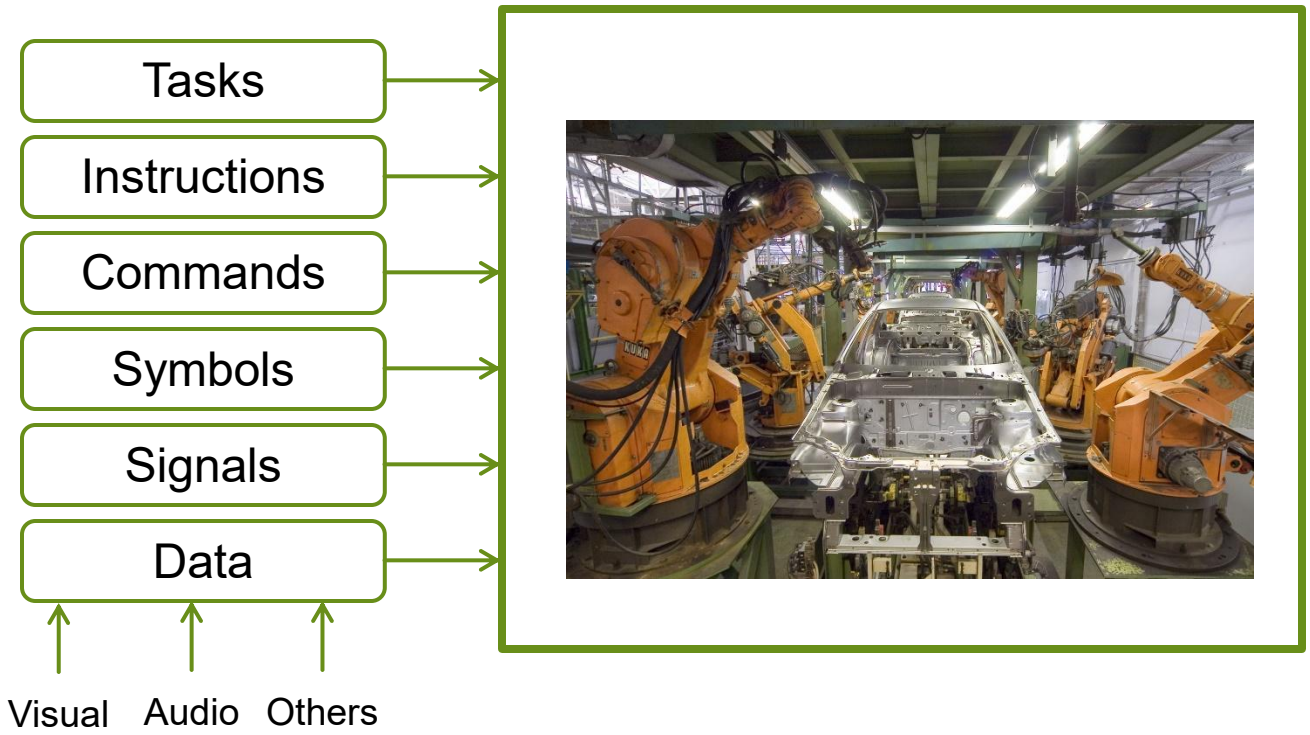
A	
0	1
1	0

Outline of Lecture 3

- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO



Input to Robots

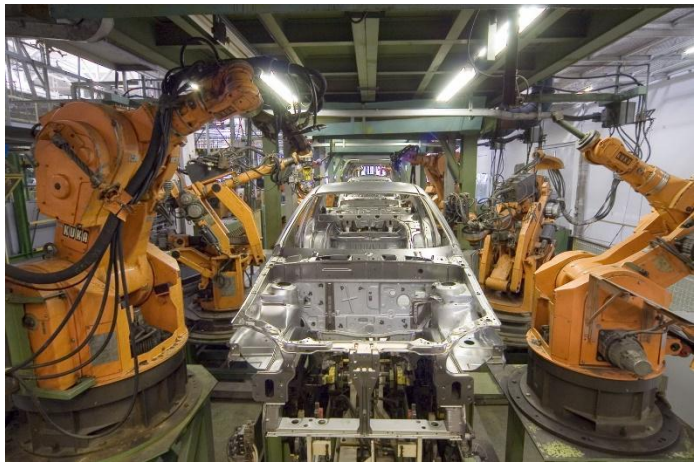


Output from Robots

Robots are machines which perform tasks through motion execution

Tasks
Instructions
Commands
Symbols
Signals
Data

Visual Audio Others



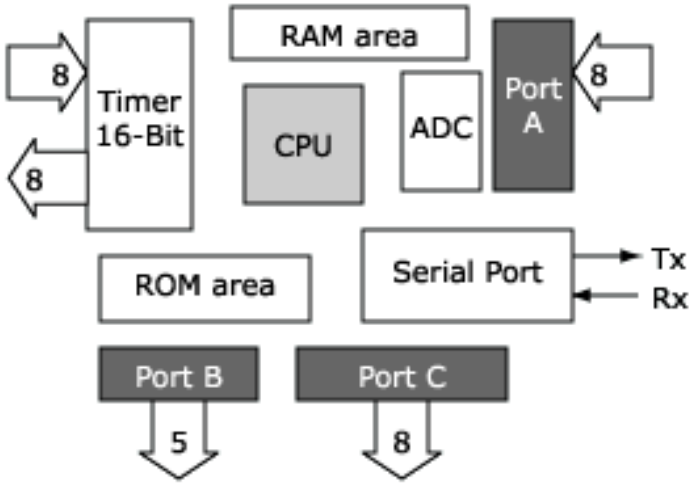
Motions
Symbols
Signals
Data

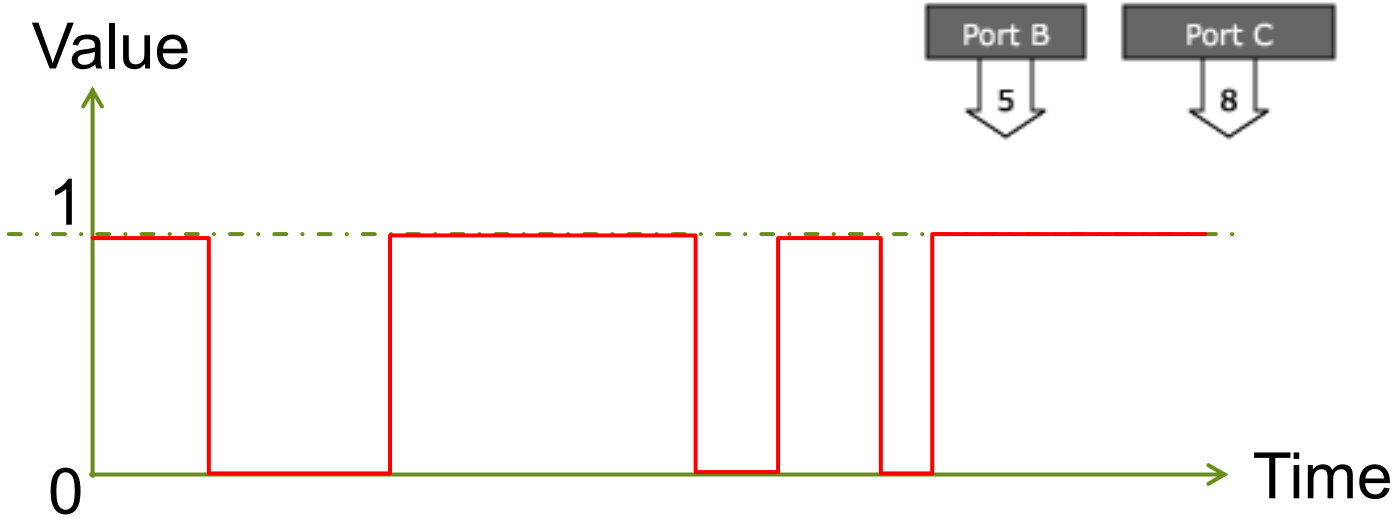
Visual Audio Others

Input and Output (IO) Unit

Representation of Digital Signals

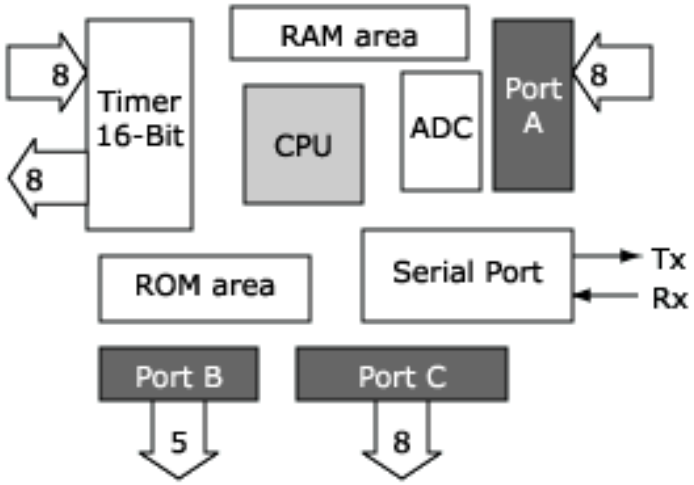
- Time Series of 1s and 0s
- Time Series of Squared Waves

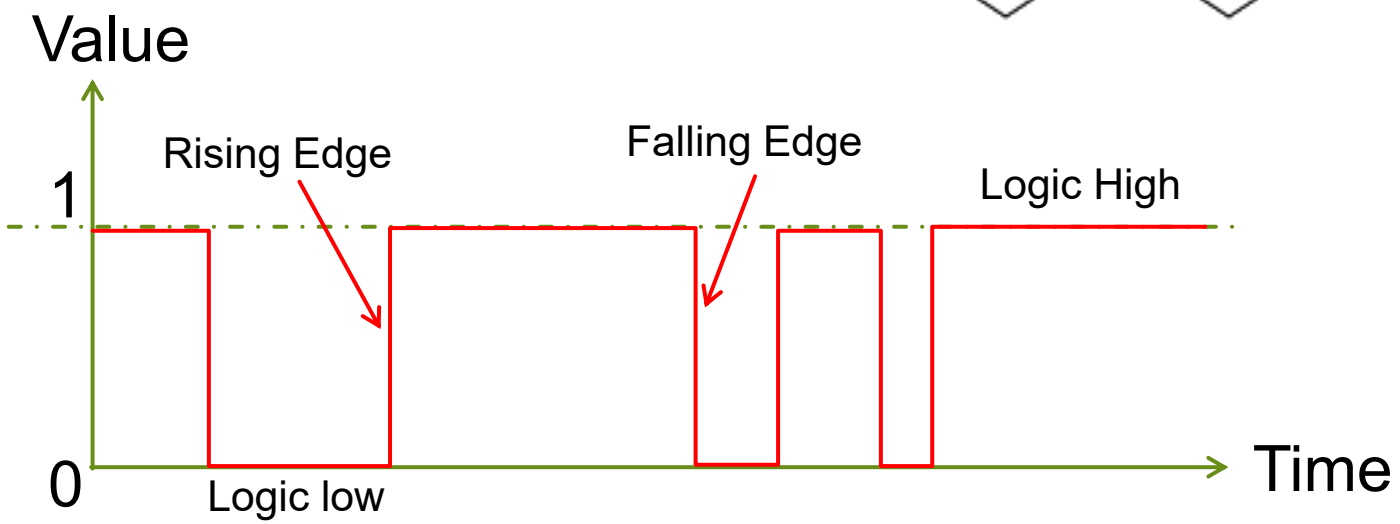




Events of Digital Signals

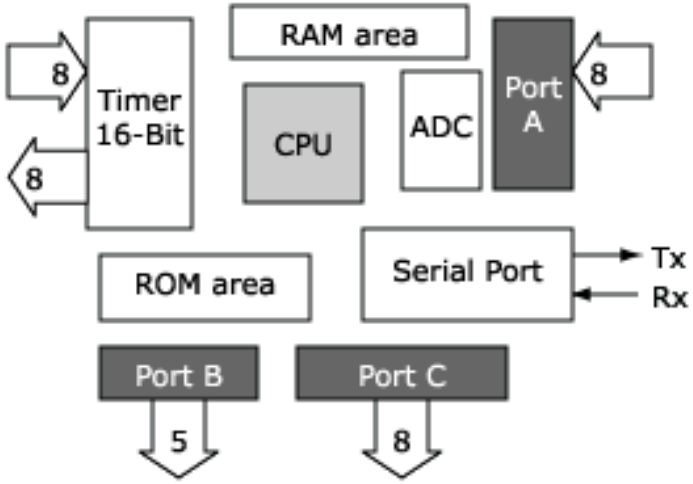
- Rising Edge
- Falling Edge
- Level of Logic High
- Level of Logic Low





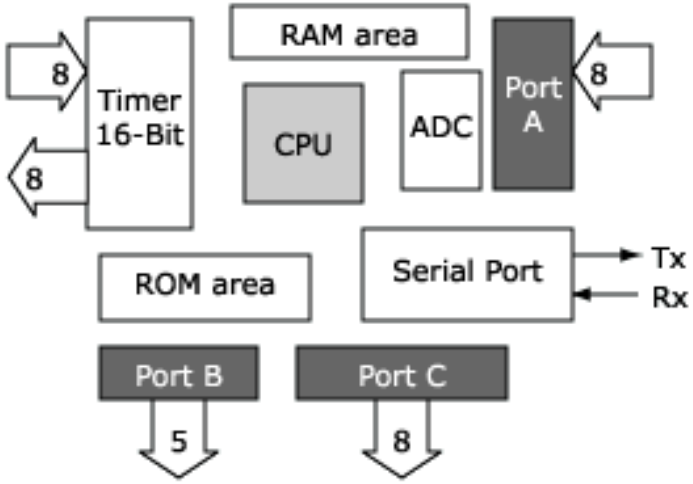
Signal Lines of Digital Signals

- They are physical wires which transmit digital signals from one end to another end.



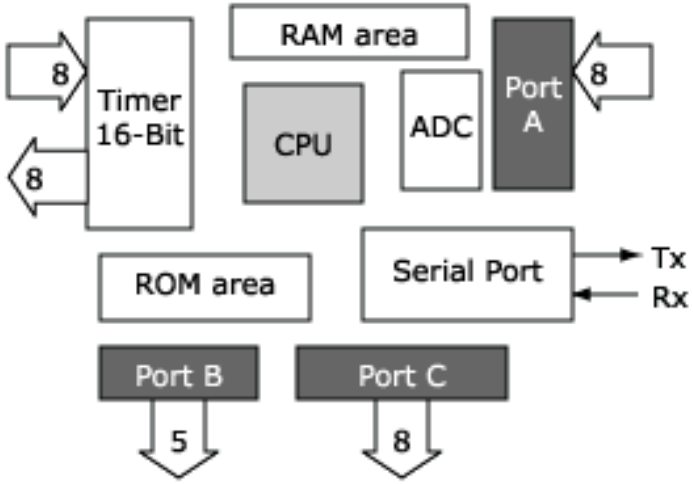
Definition of Digital Output

- When a device sends out a series of digital signals through signal lines, such operation is called Digital Output.



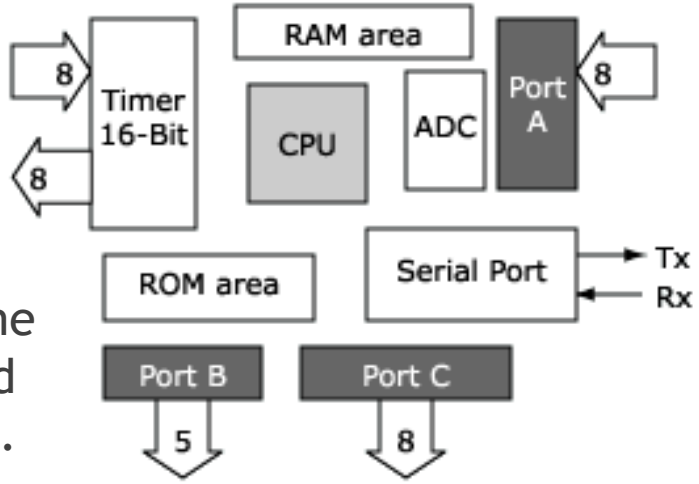
Definition of Digital Input

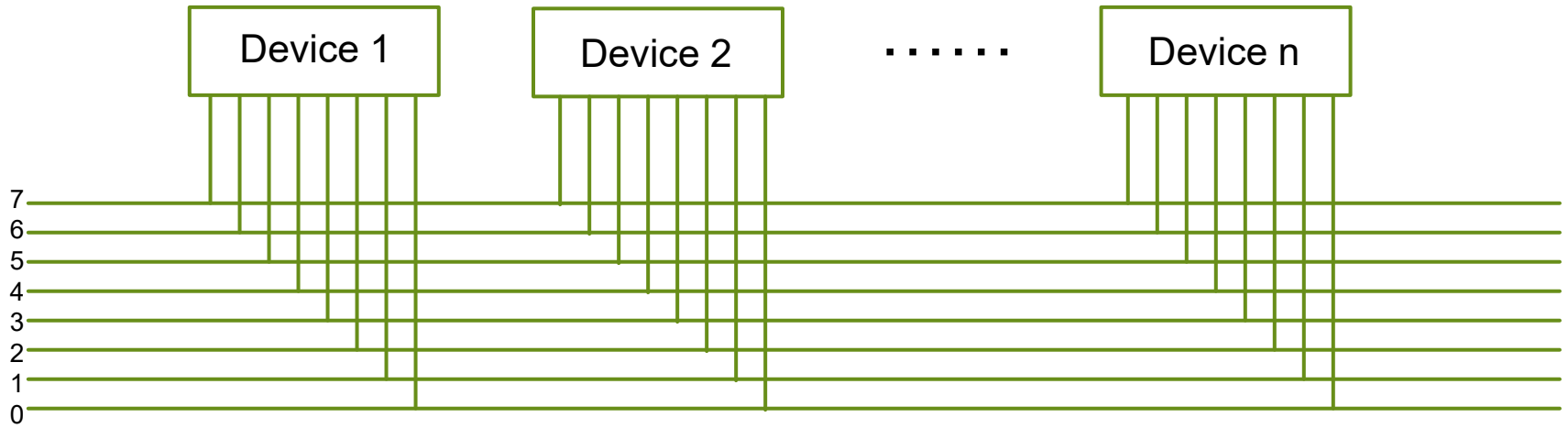
- When a device receives a series of digital signals from signal lines, such operation is called Digital Output.



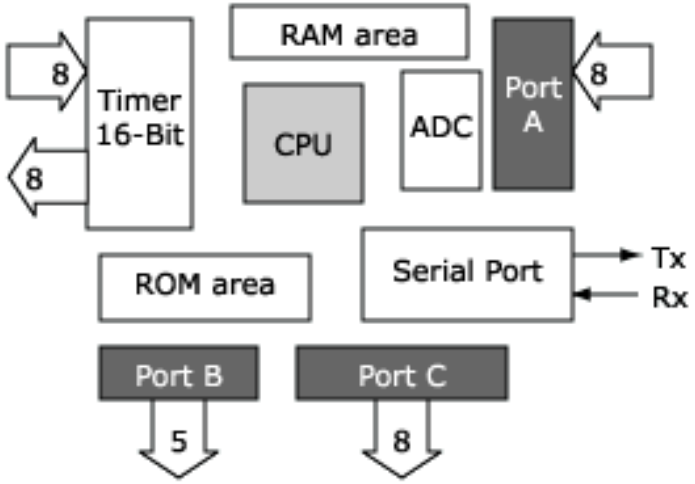
Tristate Devices

- A signal line is only at one voltage at any given time.
- If more than two devices are sharing a same signal line, these devices must have a third state which is the state of high impedance.
- A device, which can output 1, 0 or high impedance to signal lines, is called a tristate device.

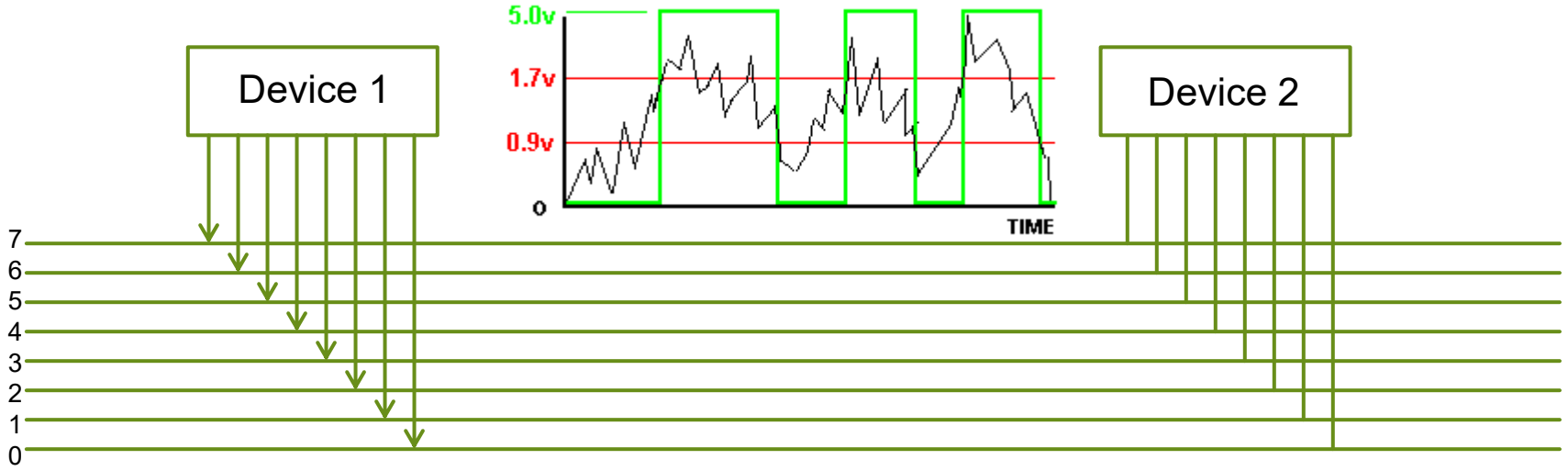




Schmitt Filters or Triggers



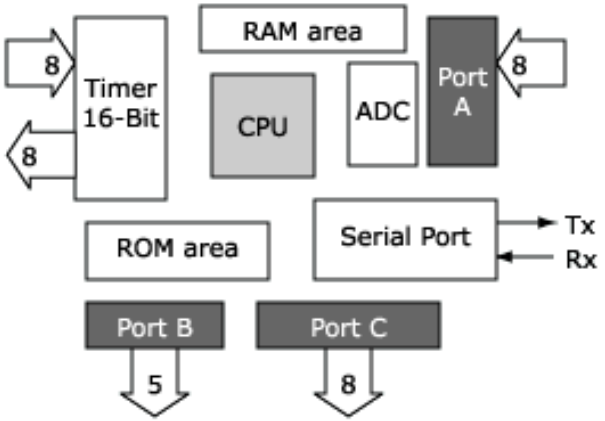
- Digital signals can be contaminated by noises due to various reasons.
- The input of digital signals at the receiver must have the ability to filter out the noises. One type of devices for such purpose is called Schmitt Filter or Trigger.

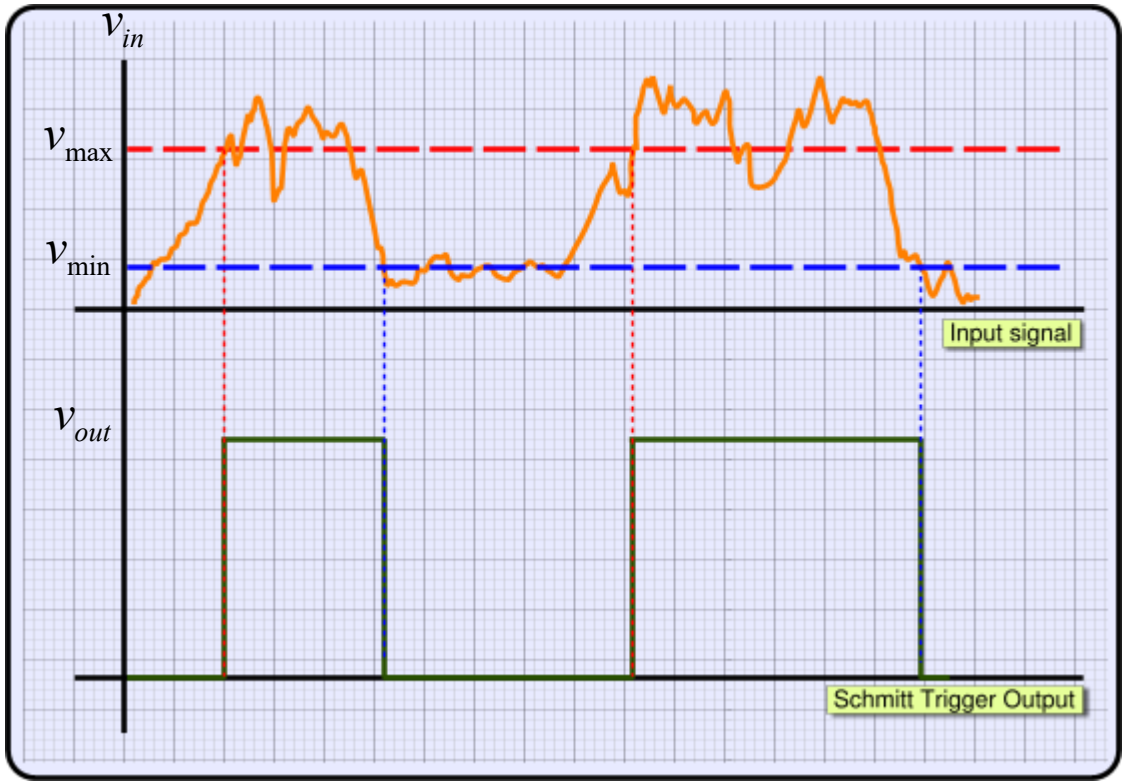


Principle of Schmitt Filter

If $v_{in} > v_{\max}$, then the output is logic high.

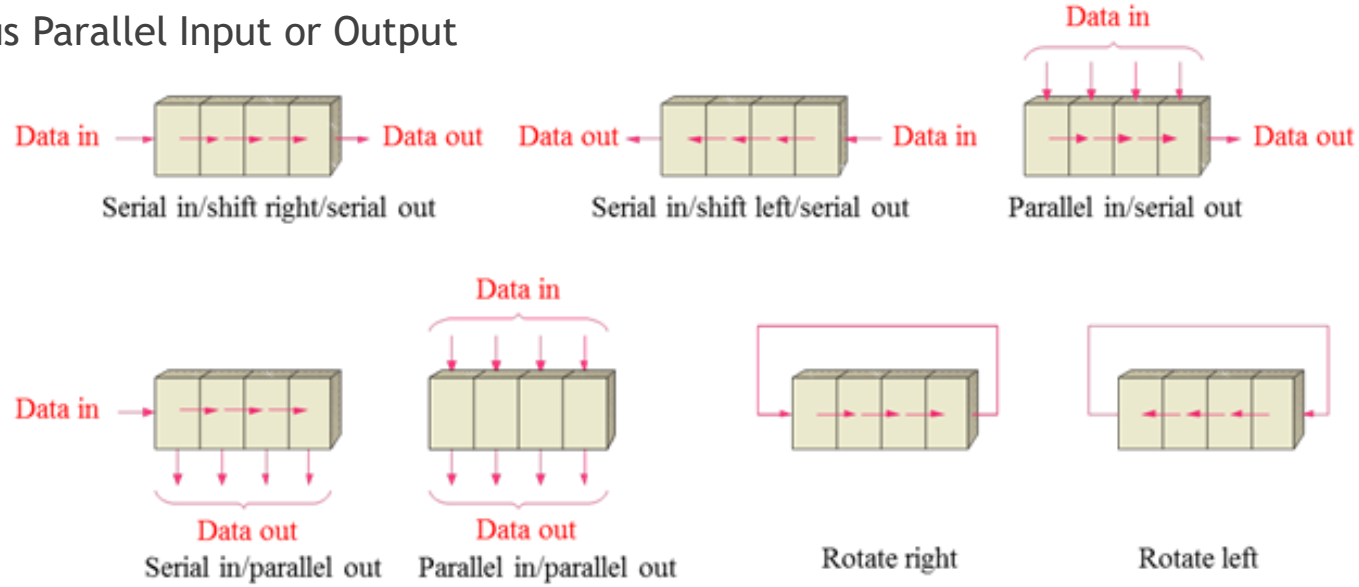
If $v_{in} < v_{\min}$, then the output is logic low.





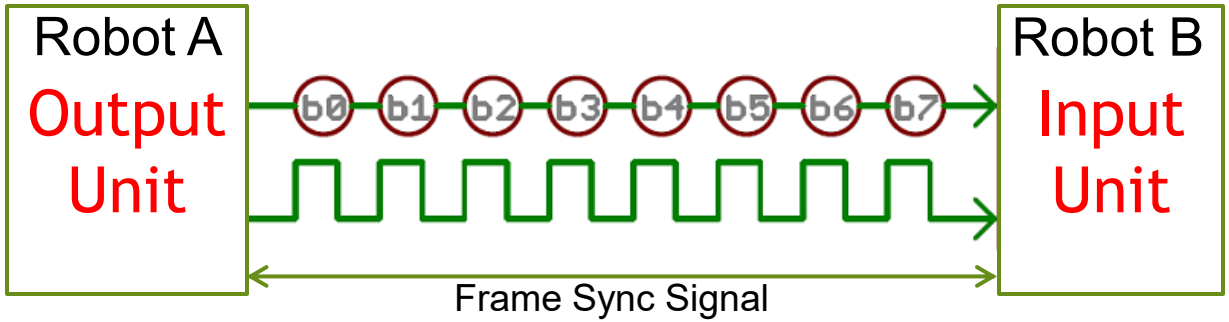
Design Solutions of Digital IO Unit

- Input or output of series of bits with synchronization clock
  - Synchronous Serial Input or Output
- Input or Output of series of bits without synchronization clock
  - Asynchronous Serial Input or Output
- Input or output of series of bytes with synchronization Pulse
  - Synchronous Parallel Input or Output
- Input or output of series of bytes without synchronization Pulse
  - Asynchronous Parallel Input or Output



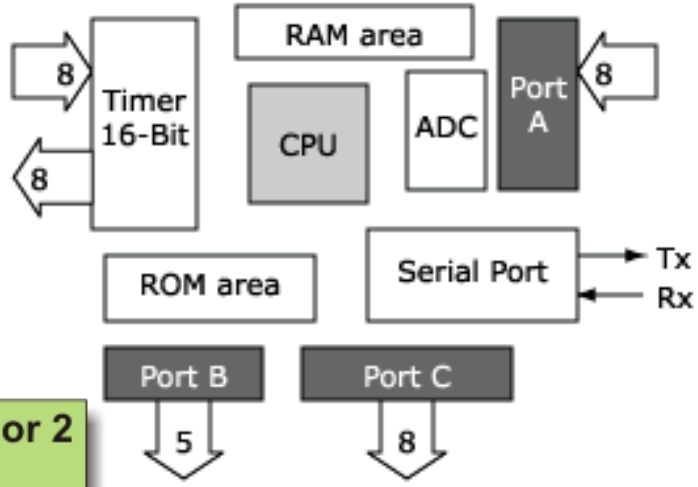
Working Principle of Synchronous Serial Input (or Output)

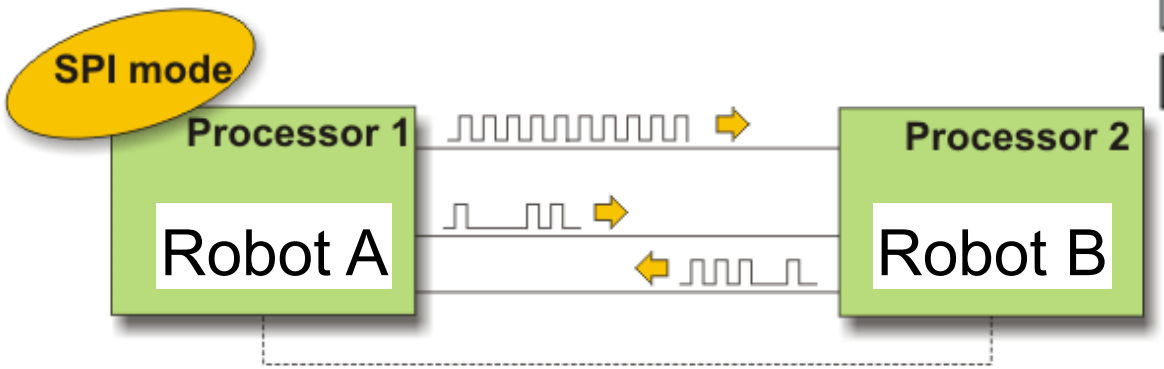
- The input unit consists of
  - Shift Register
  - Data Register
  - Control Registers
  - Status Registers
  - Data Line
  - Clock Line

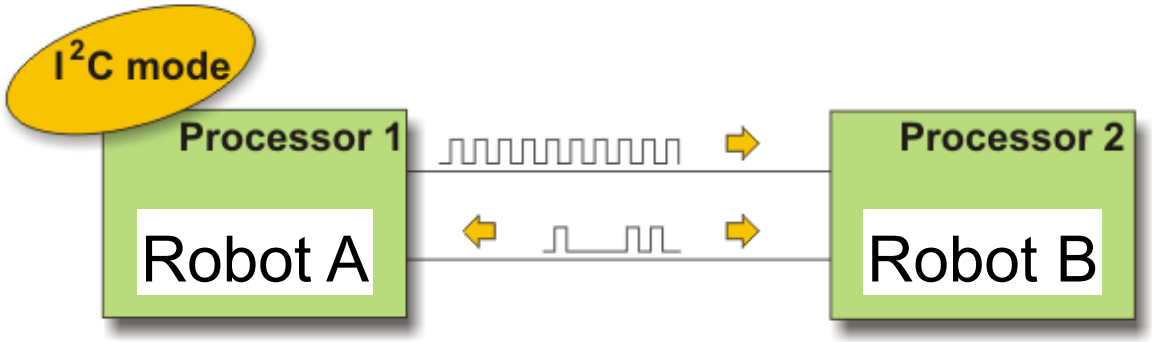


- Frame sync signal sends start-pulse which tells the clock line to produce eight clock pulses.
- Each clock pulse causes one bit to be shifted into the receiver (e.g. Robot B).
- Frame sync signal sends stop-pulse, the content of the shift register is copied to data register.

Application of SPI and I2C

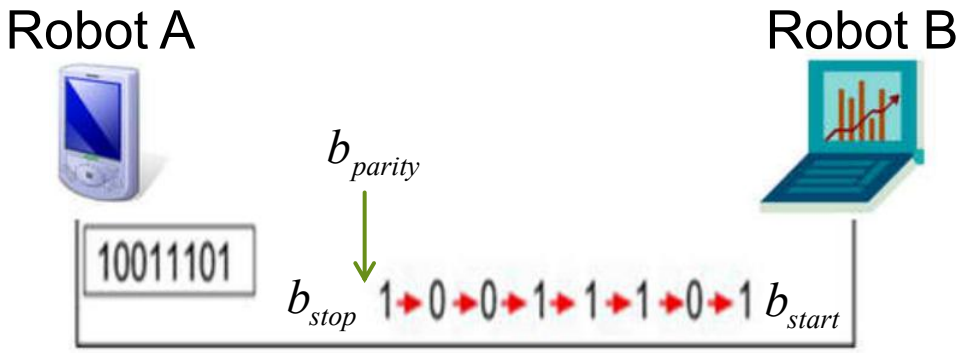






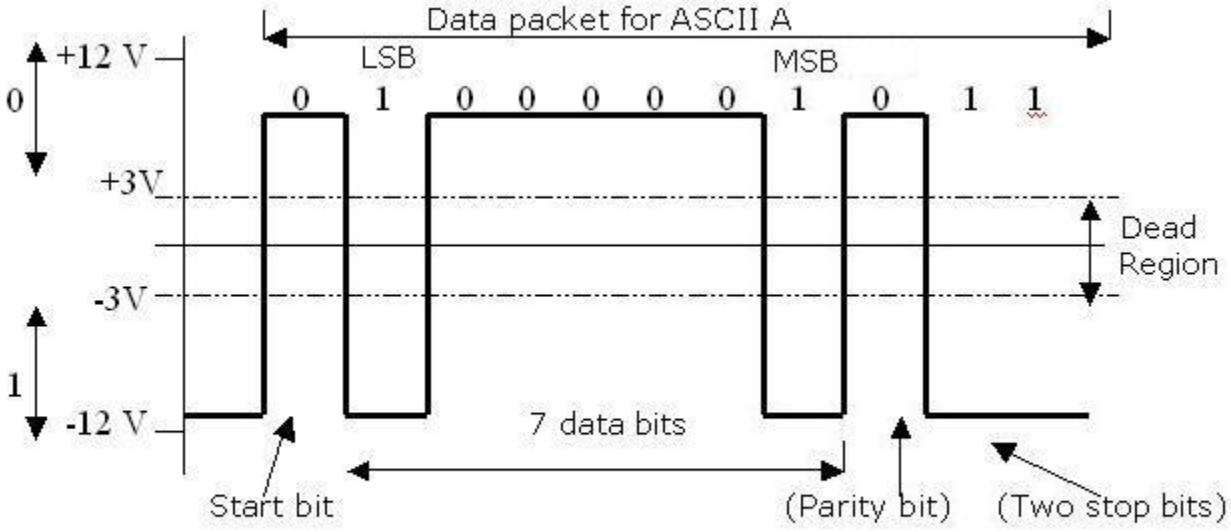
The **Serial Peripheral Interface** (**SPI**) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems.

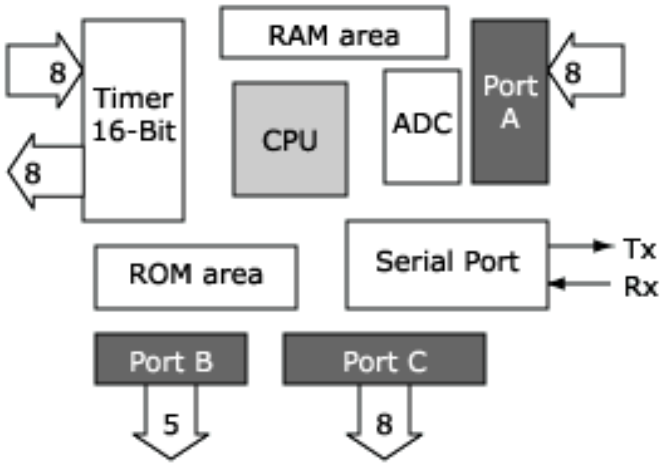
Working Principle of Asynchronous Serial Input (or Output)

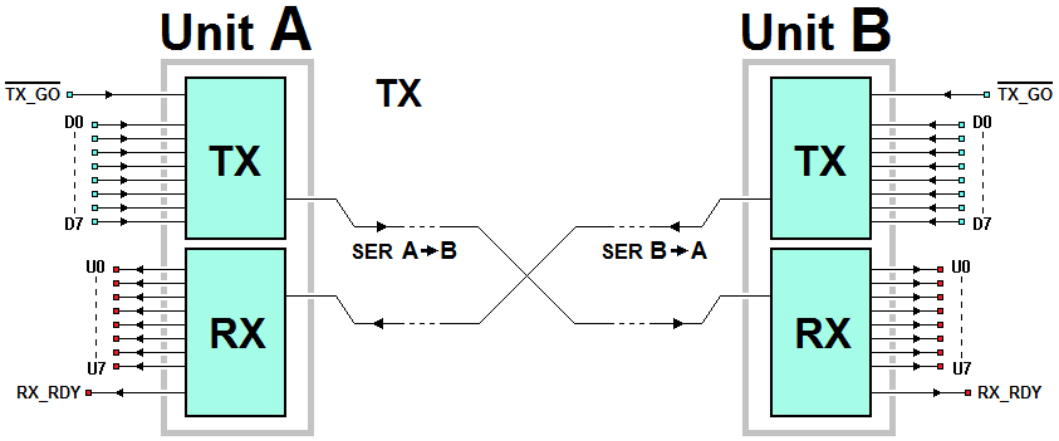


- The input unit consists of
  - Shift Register
  - Data Register
  - Control Registers
  - Status Registers
  - Data Line
- One byte is formatted with additional bits such as start bit, stop bits and parity bit.
- Transmitter and receiver are configured to be running at a same frequency of clock.
- The communication is initiated by transmitter and is triggered by the start bit.
- The start bit wakes up the receiver which will receive the series of bits until the last stop bit. The data is then copied into the data register.

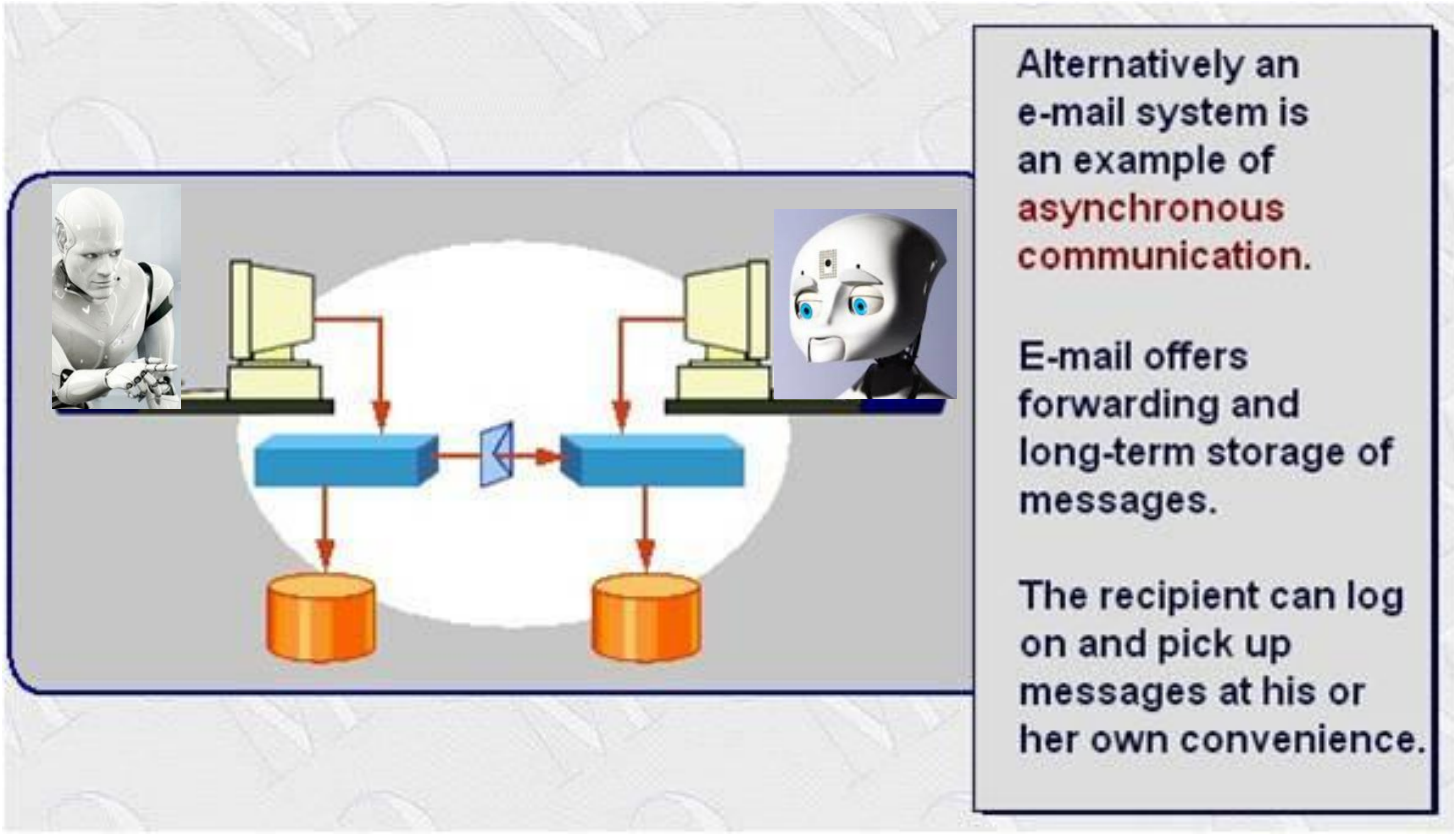
Example of Using ASCII Code





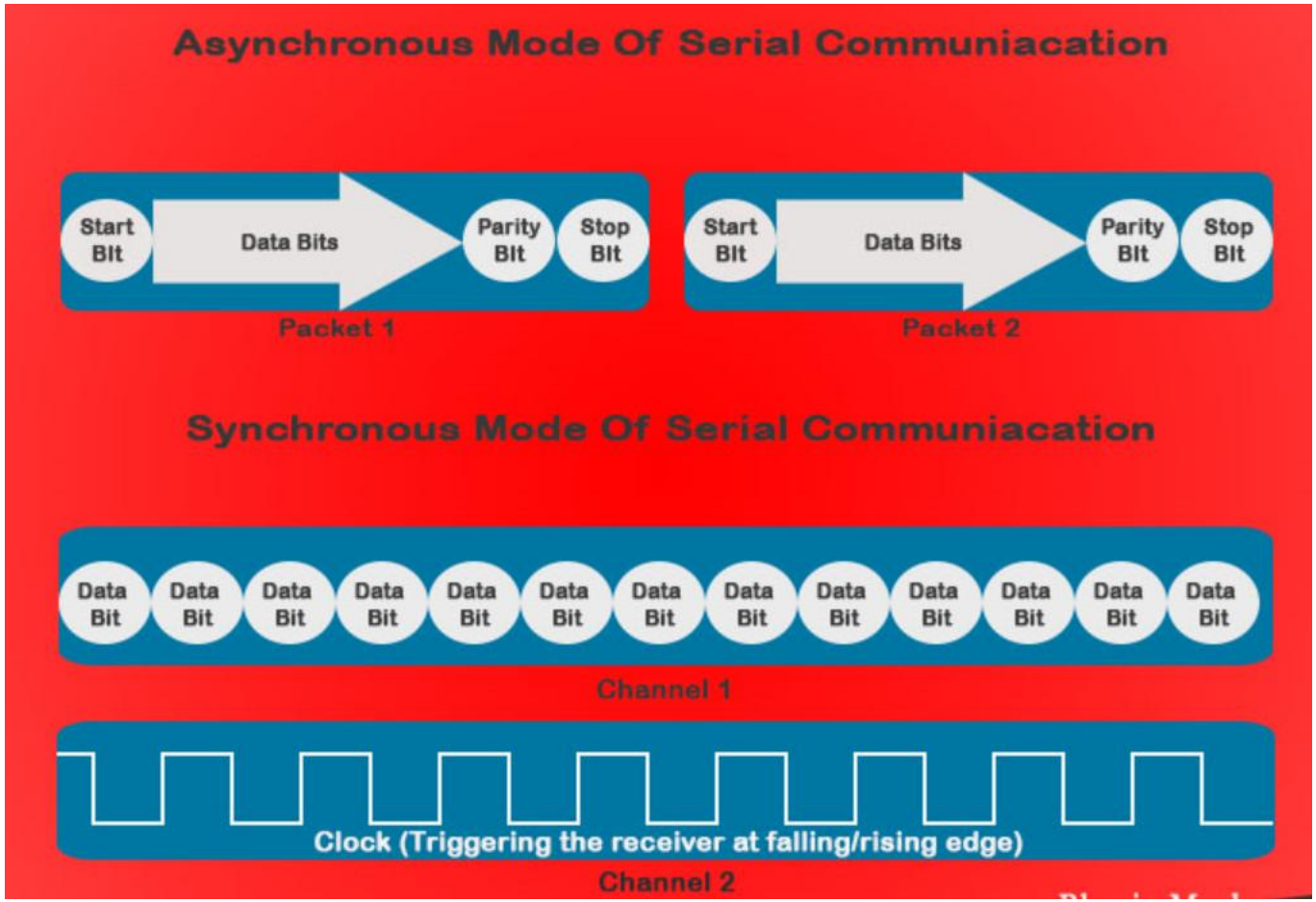


Application for Long-distance Communication



A serial communications interface (SCI) is a device that enables the serial (one bit at a time) exchange of data between a microprocessor and remote peripherals such as printers, external drives, scanners, or mice.

Comparison between SCI and SPI

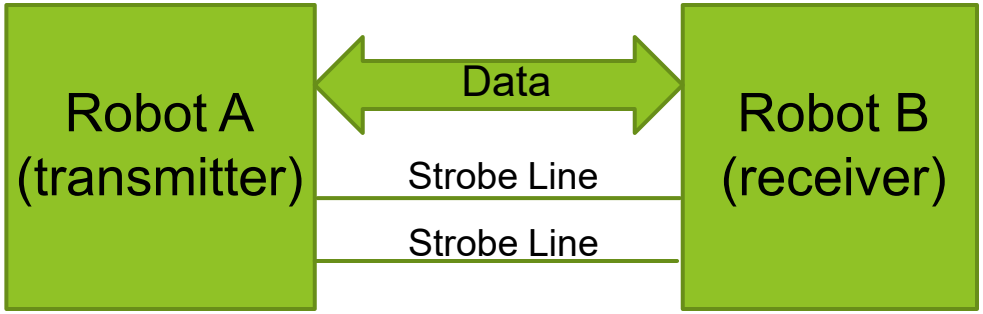


SCI

SPI

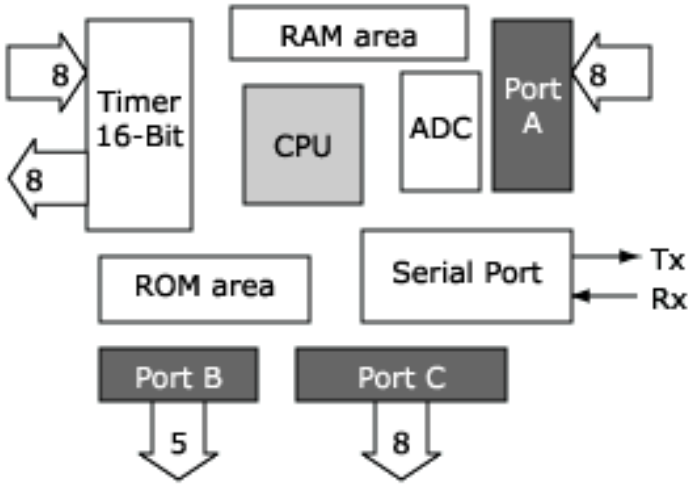
Working Principle of Synchronous Parallel Input (or Output)

- The input unit consists of
  - Data Register
  - Control Registers
  - Status Registers
  - Input Port (8 pins, 1 byte)
  - Strobe Lines



- If Receiver initiates the communication, it first indicates that Receiver is ready to input one byte. Transmitter then sends one byte to data bus and indicates that one byte is ready to be read. Receiver inputs one byte from data bus. (Suitable for reading sensory input)
- If Transmitter initiates the communication, it sends one byte to data bus and indicates that one byte is ready to be read. Receiver inputs one byte from the data bus. (Suitable for sending commands to motors)

Working Principle of Asynchronous Parallel Input (or Output)



- The input unit consists of
  - Data Register
  - Control Registers
  - Status Registers
  - Input Port (8 pins, 1 byte)
- CPU uses the control registers to configure the input port.
- TRIS (i.e. tristate register) controls the directions of pins in the input or output port.
- CPU reads one byte from input port.
- CPU decodes the content.
- CPU acts according to the content.

	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	Features
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

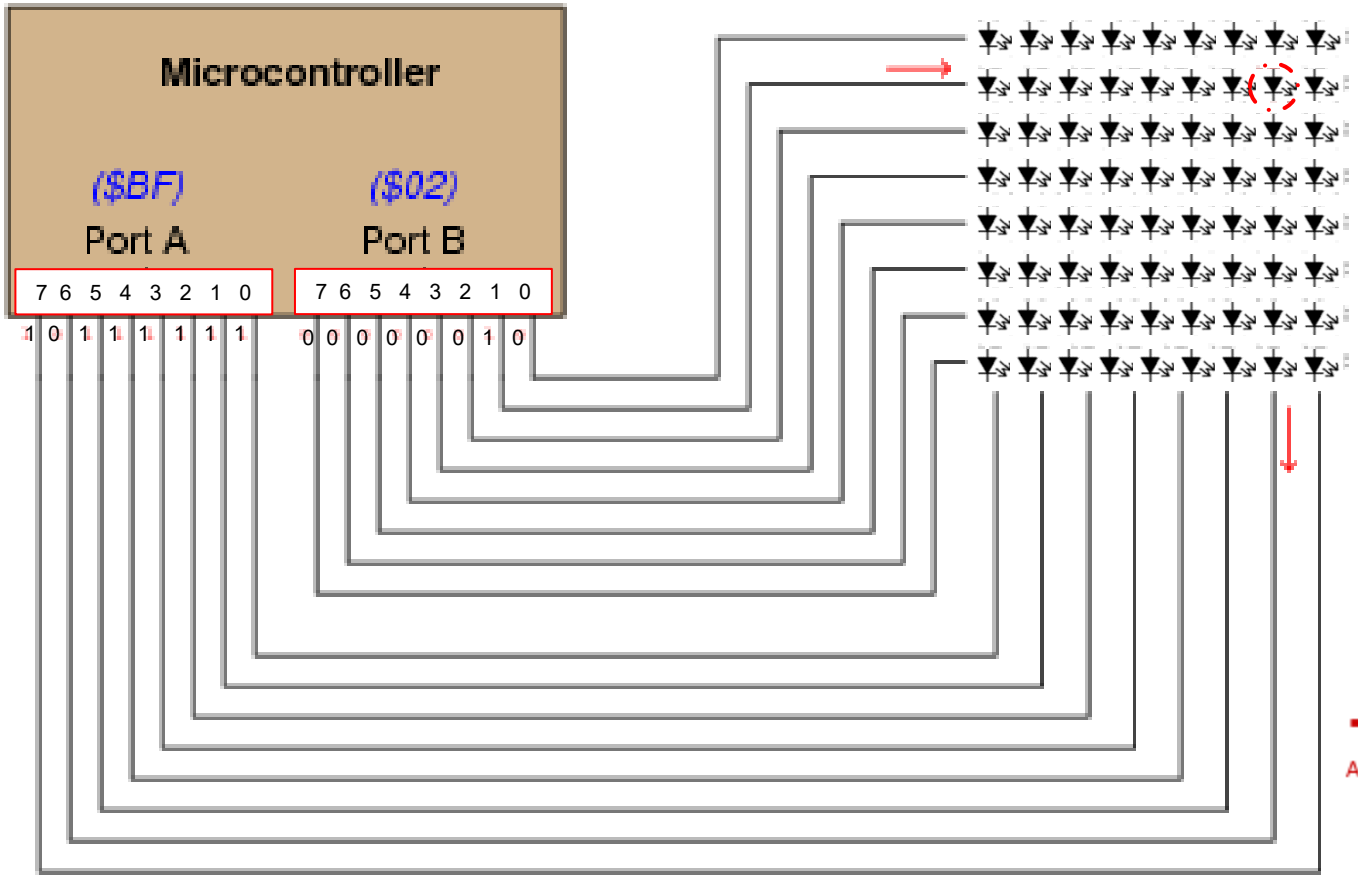
	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	Features
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

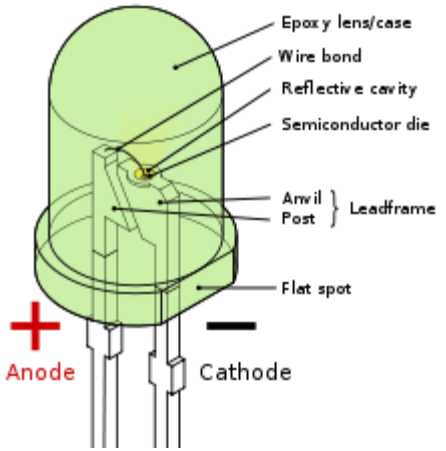
Legend

R/W	Readable/Writable bit
(x)	After reset, bit is unknown
(1)	After reset, bit is set

Example of Interfacing with Dot Matrix of LED

- CPU outputs $BF to port A and $02 to port B.
- Which LED will light up?
- Answer: the one at row 1 and column 6

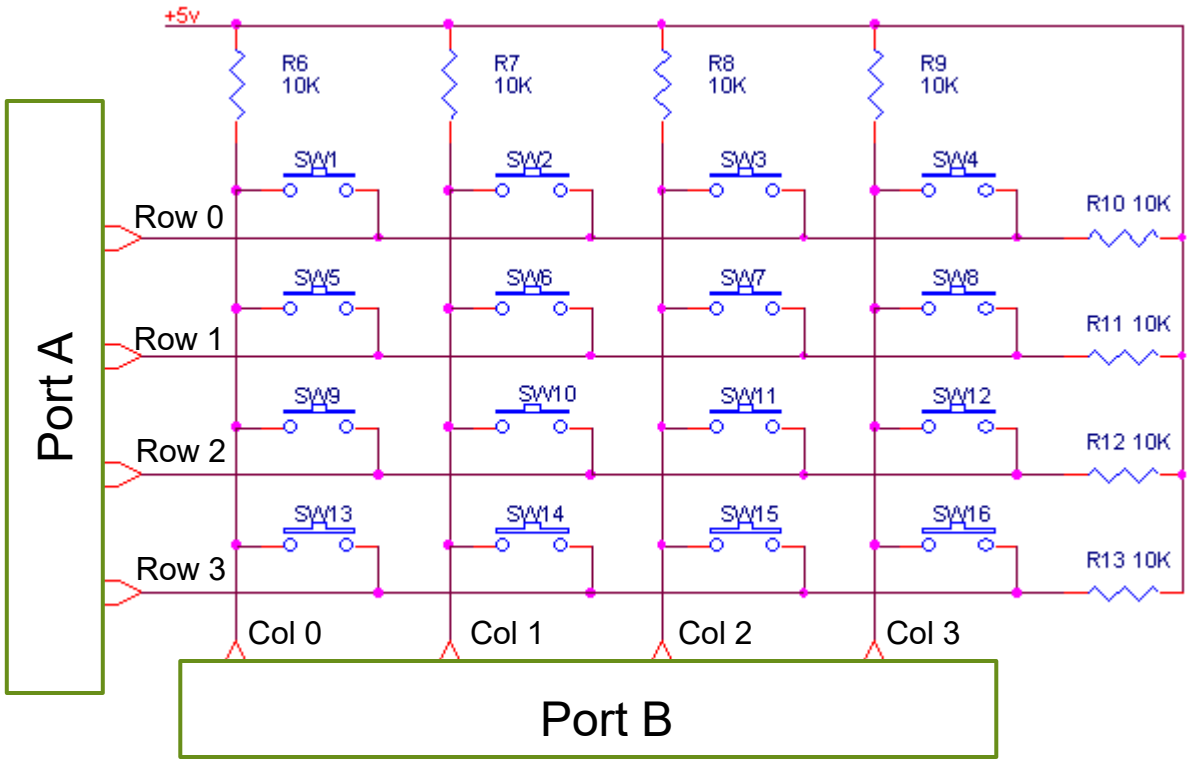


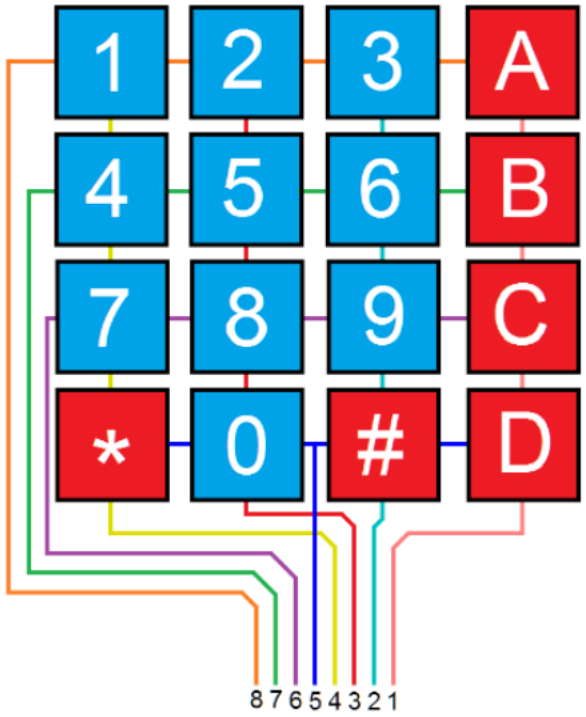


Example of Interfacing with Keypad

Port A = 1101
Port B = 1101
Which key?
Answer: 5

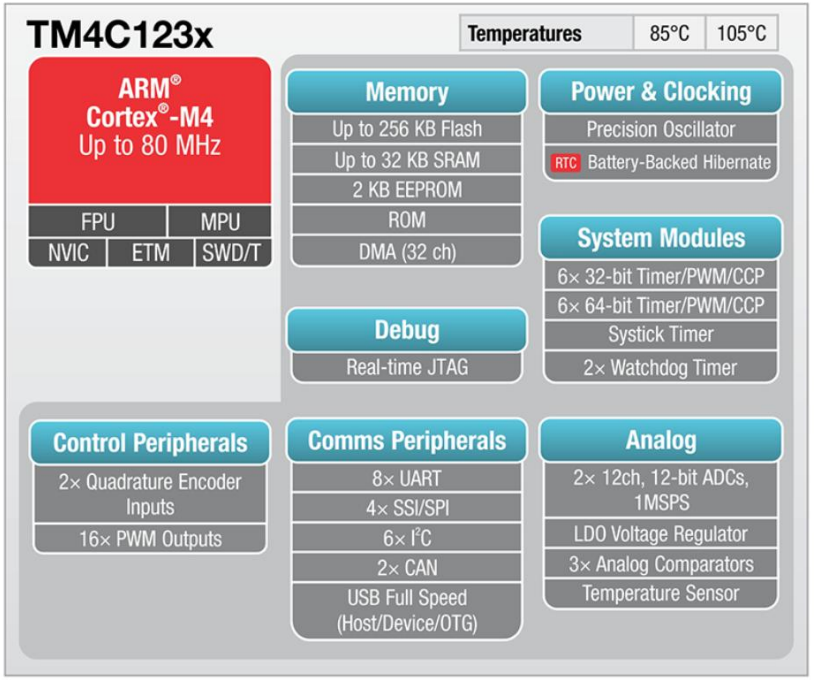
- CPU sequentially outputs 1110, 1101, 1011, 0111 to port A
- CPU inputs data from port B
- By default, all the keys are not pressed.
- When a key is pressed, the corresponding column line will be in logic low (i.e. “0”).





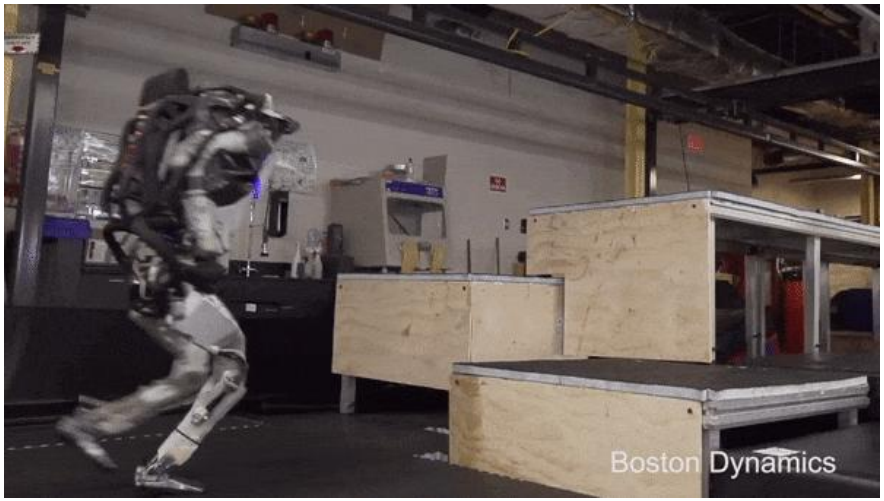
Summary of Lecture 3

- Basics of Controller
- Design of Memory
- Design of ALU
- Design of Digital IO



Outline of Module 1

- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

Lecture 4

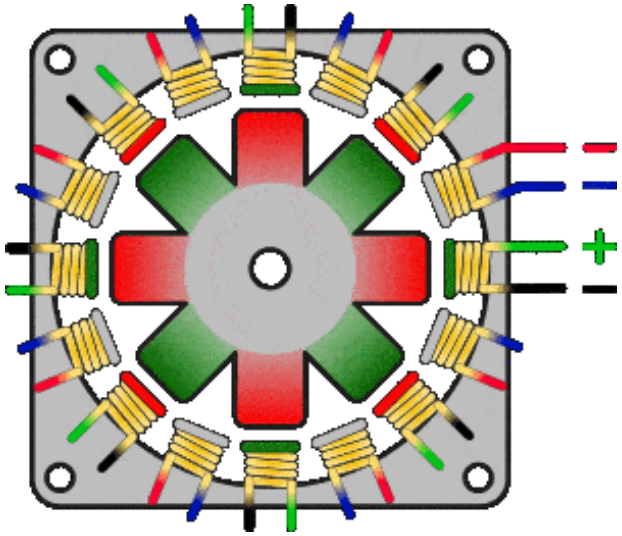
Robot Actuators

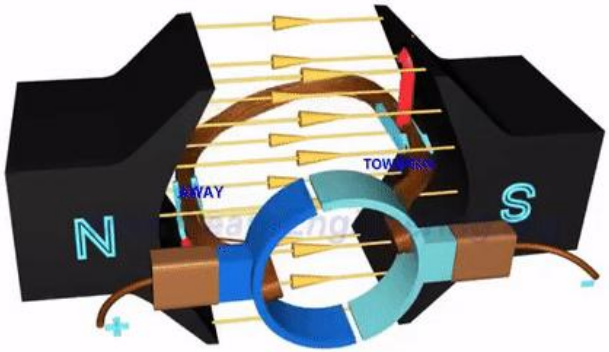
Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

Outline of Lecture 4

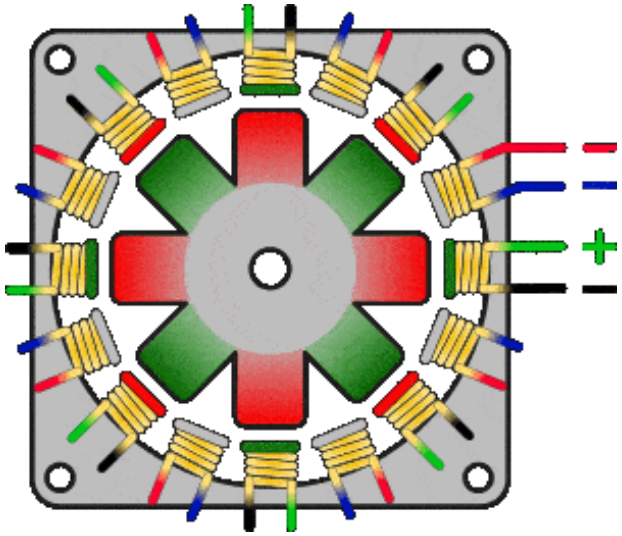
- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor

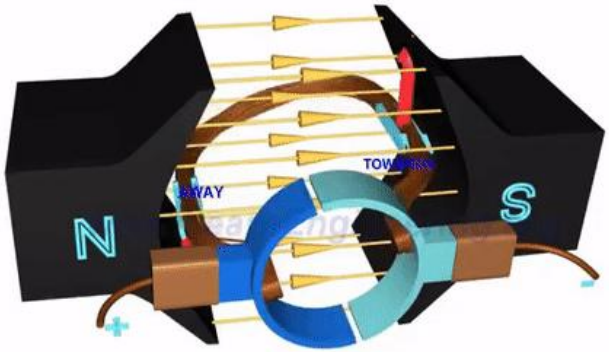




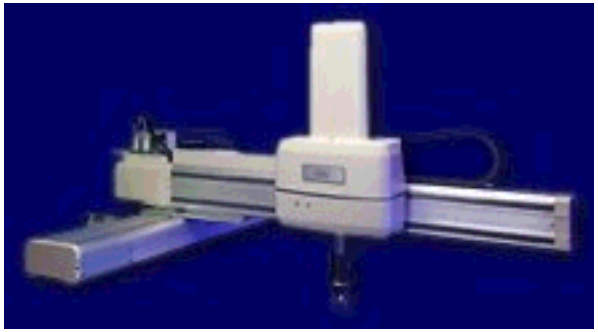
Outline of Lecture 4

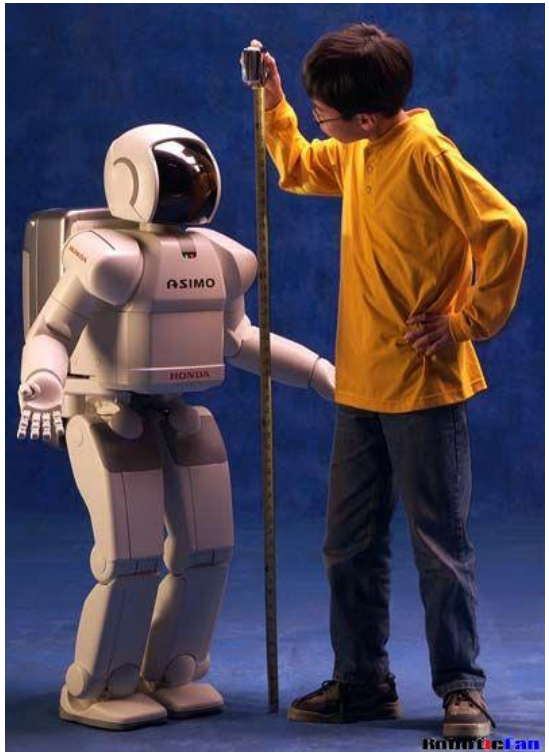
- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor





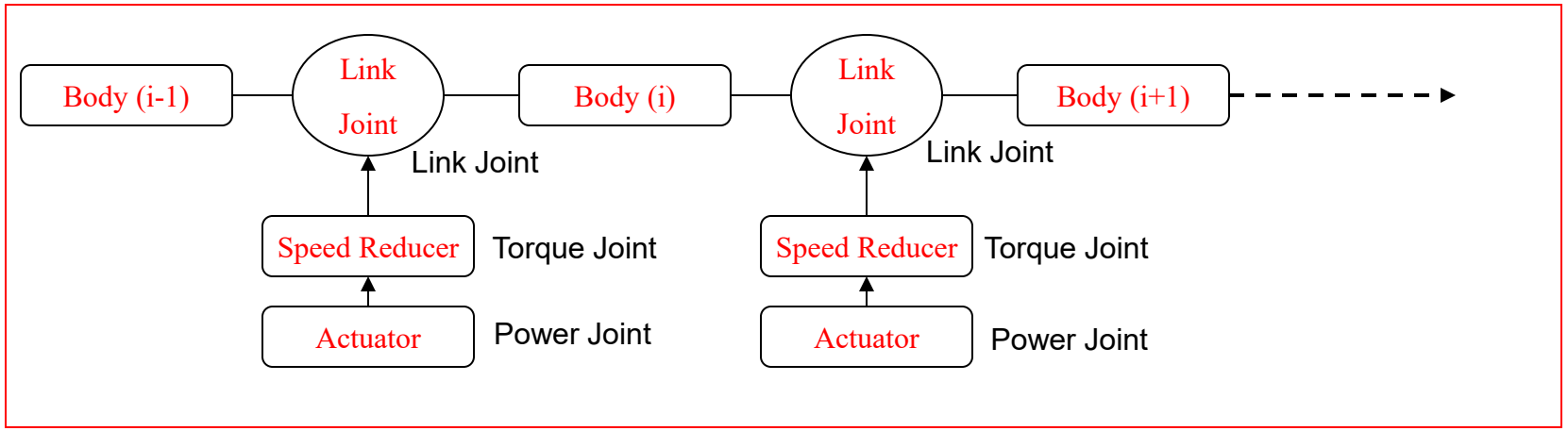
Examples of robots' mechanisms

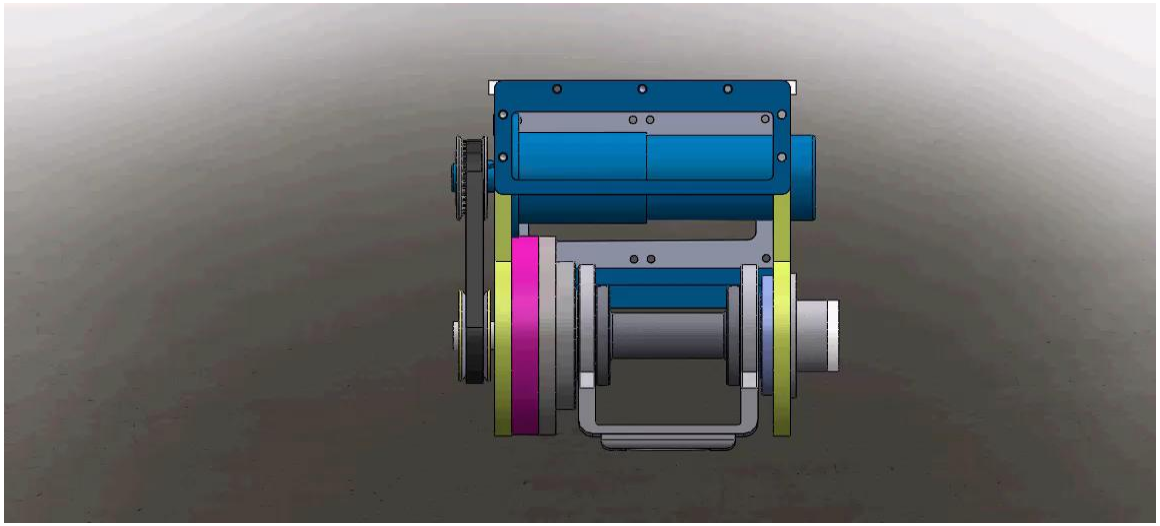




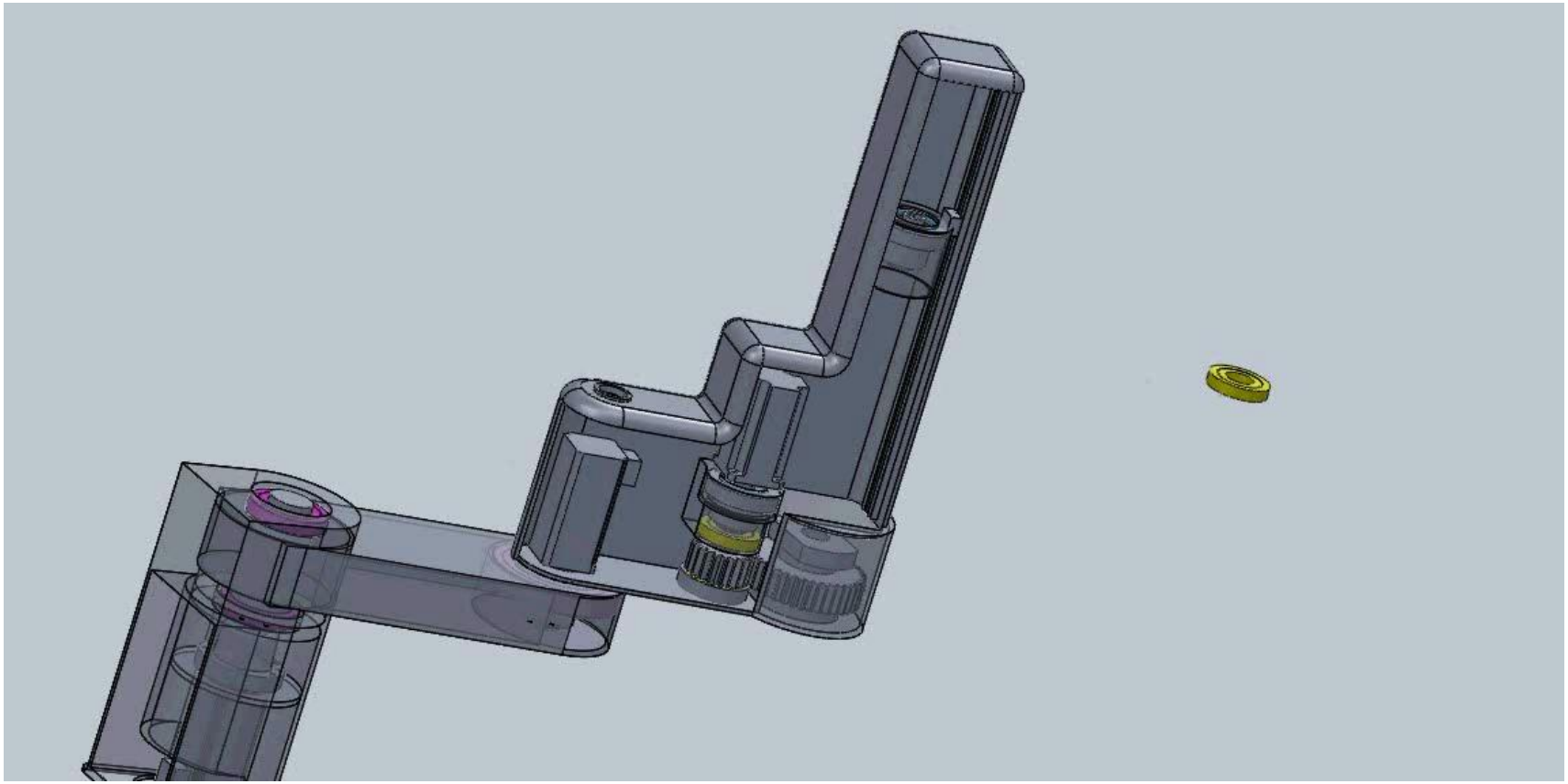
How to actuate a robot's mechanism?

Conventional Method of Actuation





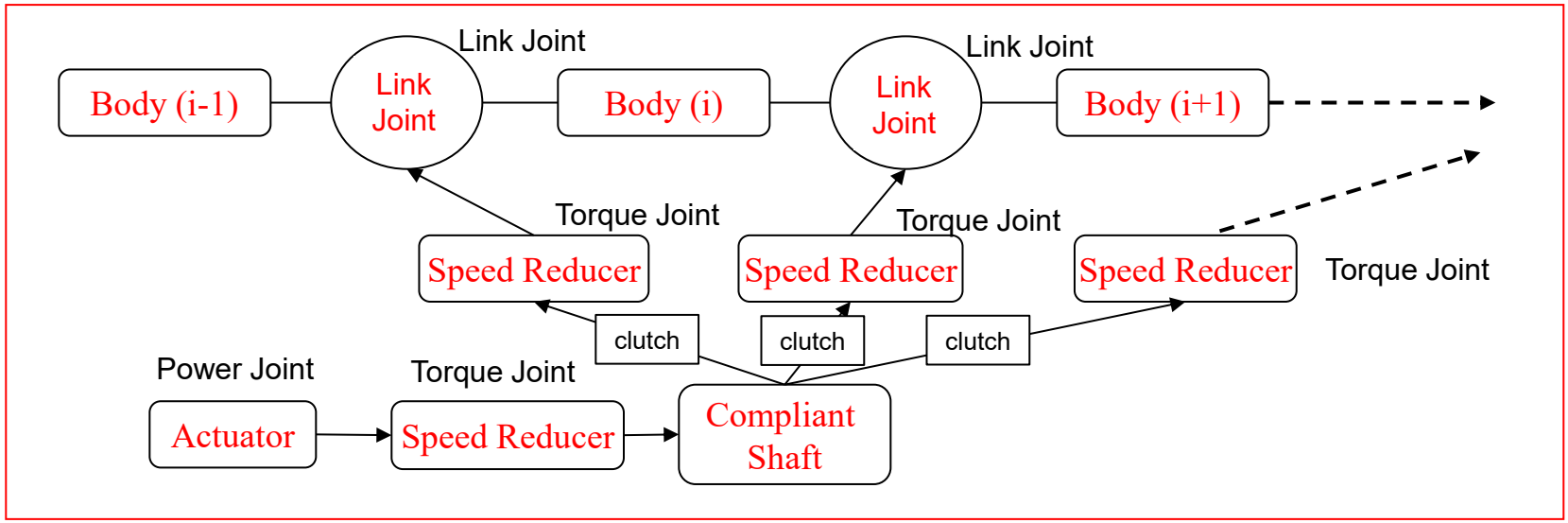
Example of actuating a robot mechanism

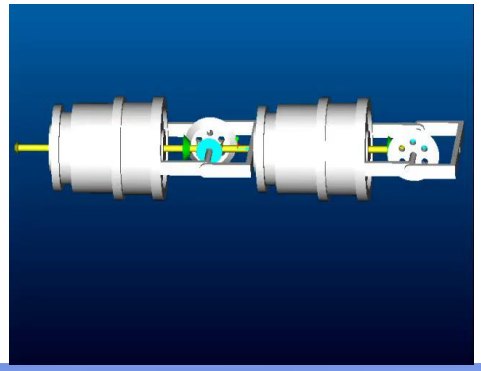


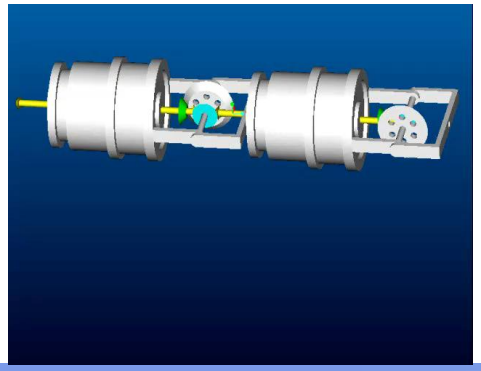
What can we learn from this example?

Innovation Done by Our Team in Year 2000

- One joint will be actuated by multiple actuators.
- One actuator will actuate multiple joints.



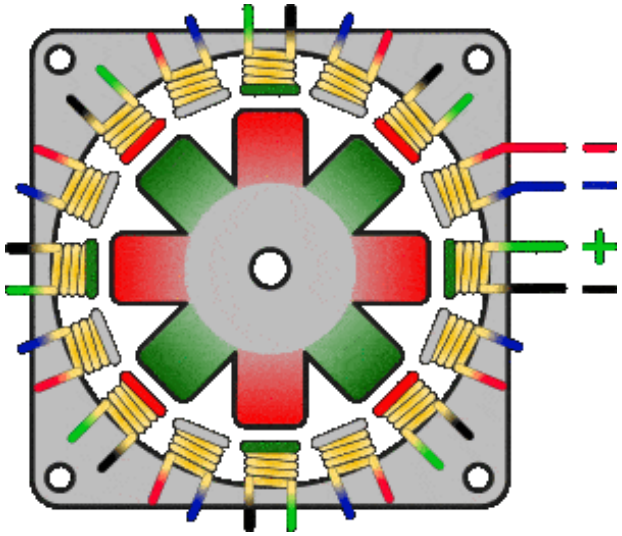


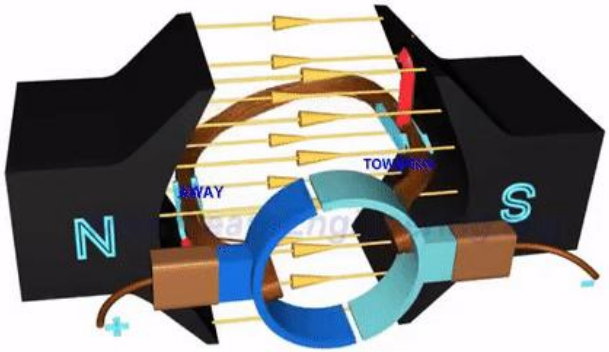


Example of Single-Motor-Driven Robots

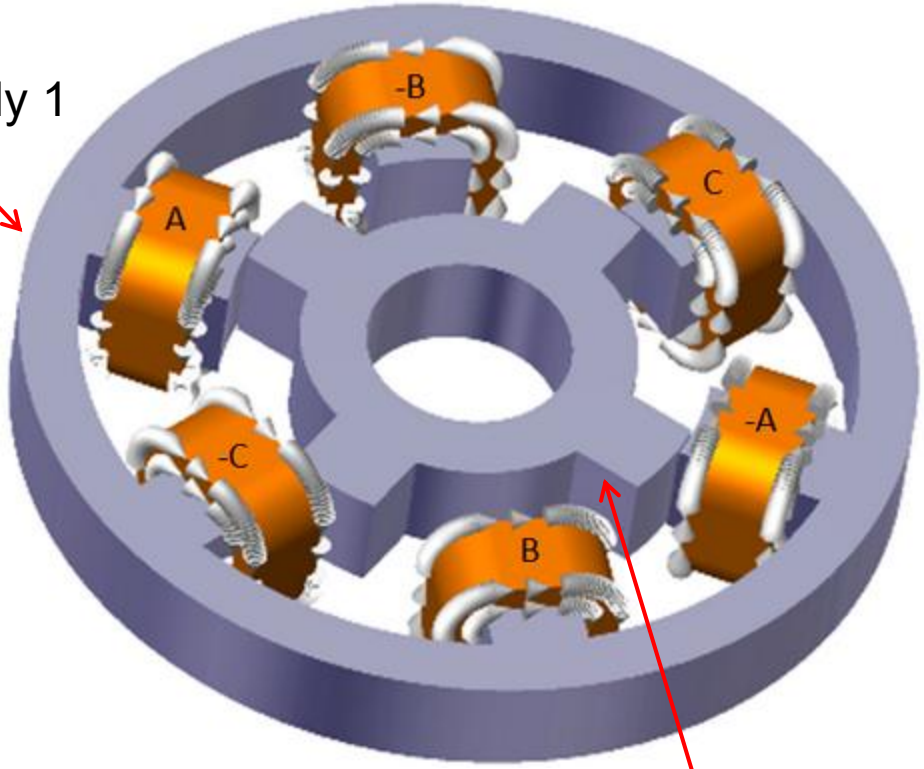
Outline of Lecture 4

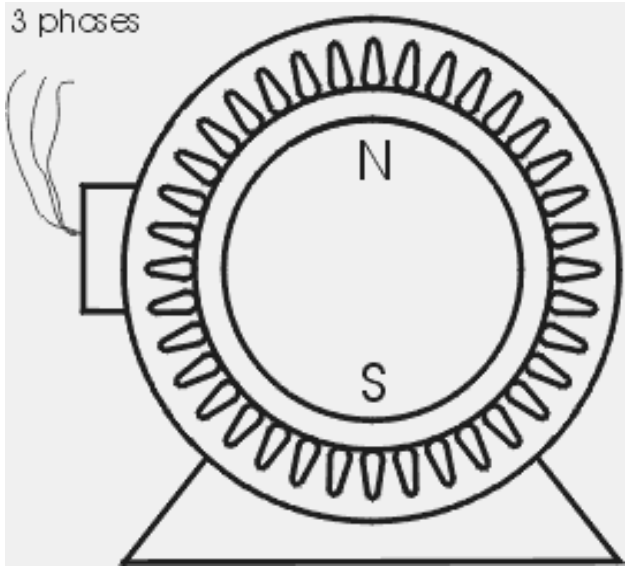
- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor





The design principle of a power joint is to make two rigid bodies to have controllable interacting forces



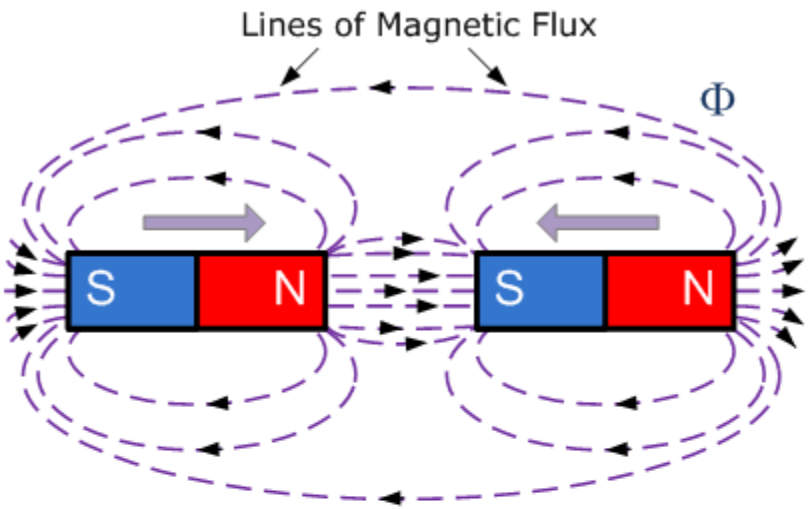


How to have controllable forces? There are several types of controllable forces:

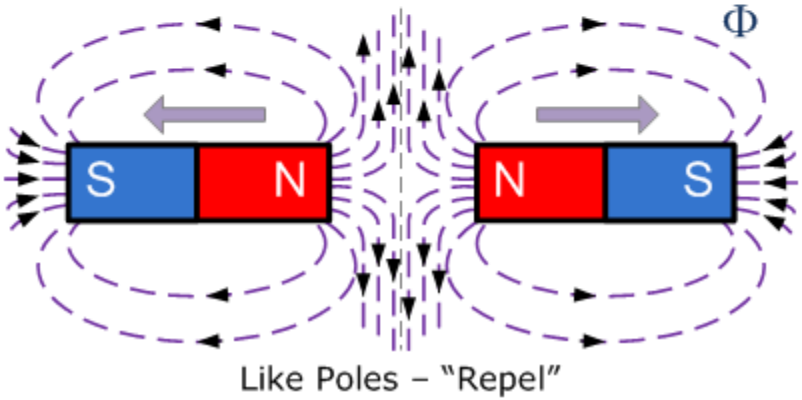
- Controllable Contact Force:
  - Forces produced by compressed airs → Pneumatic Actuators
  - Forces produced by compressed oil → Hydraulic Actuators
  - Forces produced by compressed water

- Controllable Field Force
  - Magnetic forces → Electric Actuators
  - Electromagnetic forces

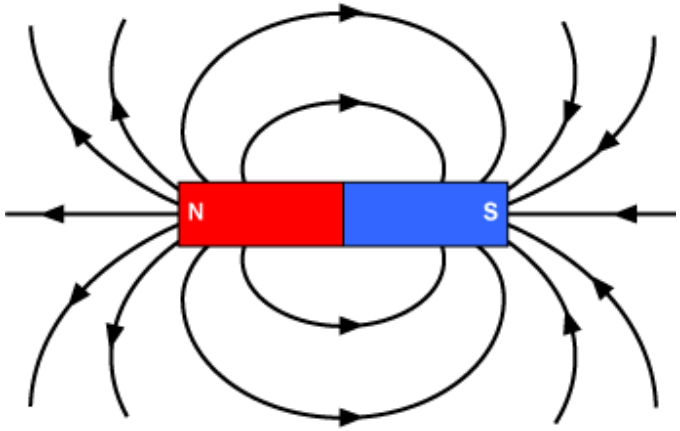
Example of Magnetic Forces



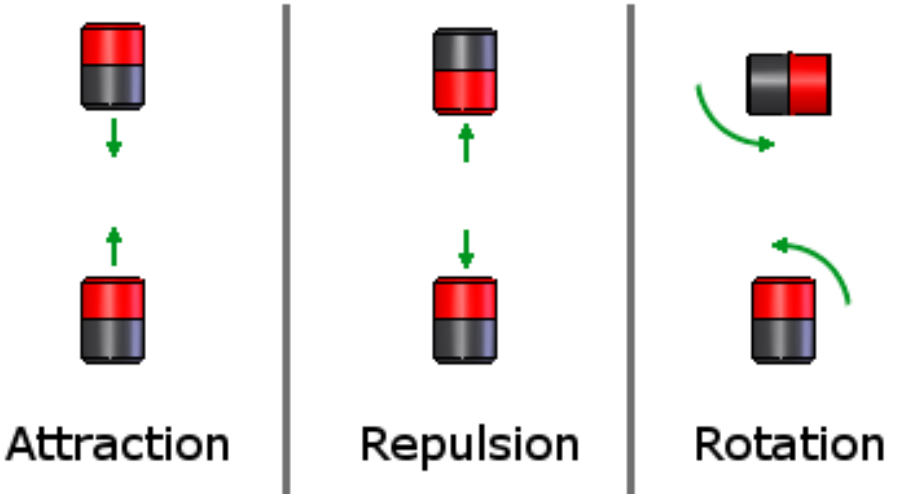
Unlike Poles – "Attract"



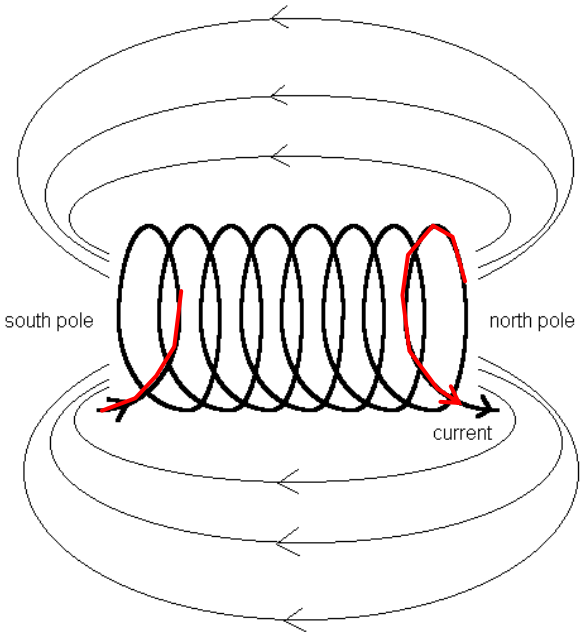
Like Poles – "Repel"

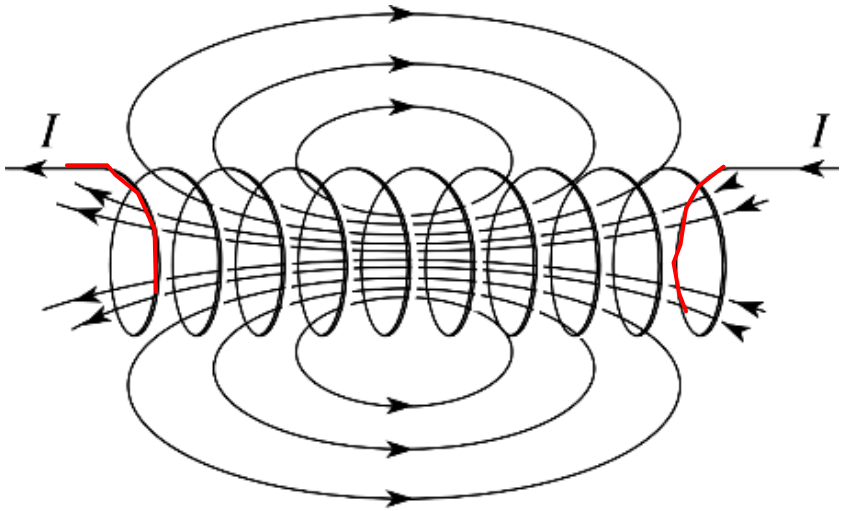


Green Arrows Indicate Magnetic Forces



Example of Electromagnetic Forces

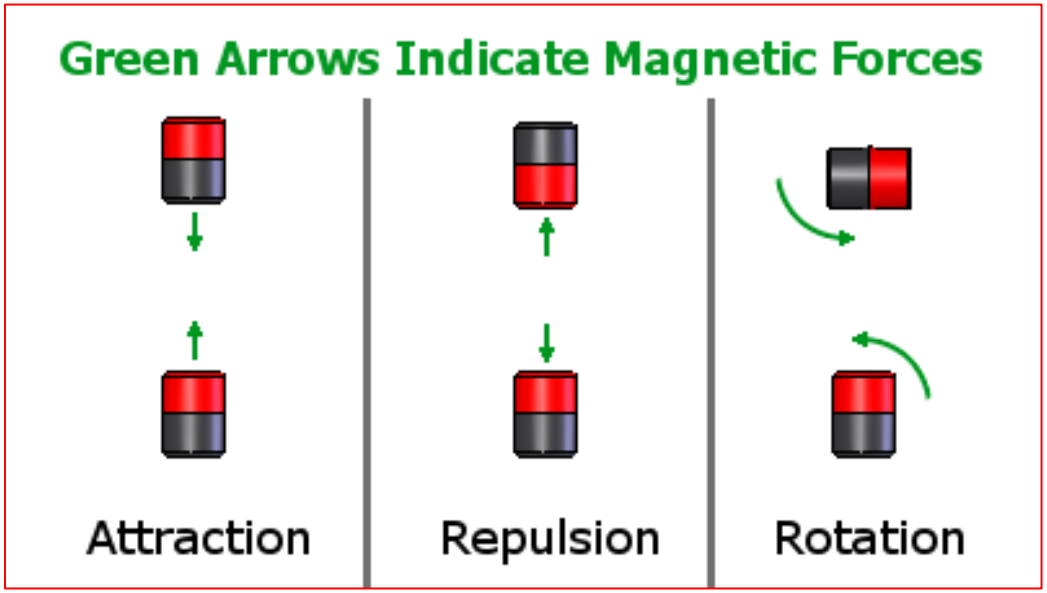




Ways of Producing Interactive Forces

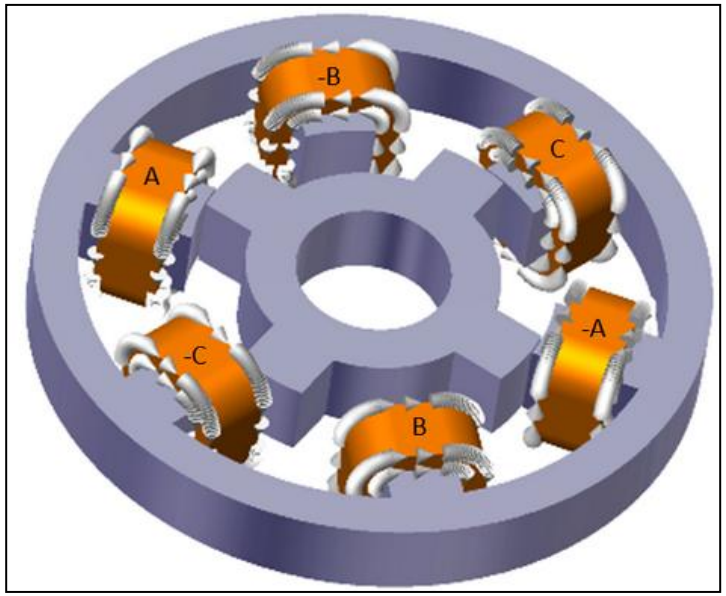
- Magnetic Field + Magnetic Field ⇦ Not controllable
- Magnetic Field + Electromagnetic Field
- Electromagnetic Field + Magnetic Field
- Electromagnetic Field + Electromagnetic Field

Examples

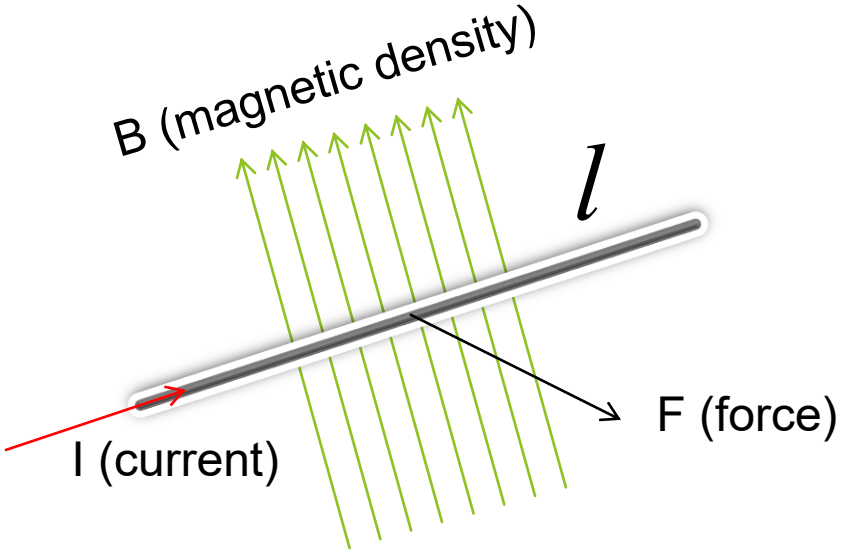


Magnetic Field + Magnetic Field

Electromagnetic Field + Magnetic Field



Understanding of Magnetic Force (1)



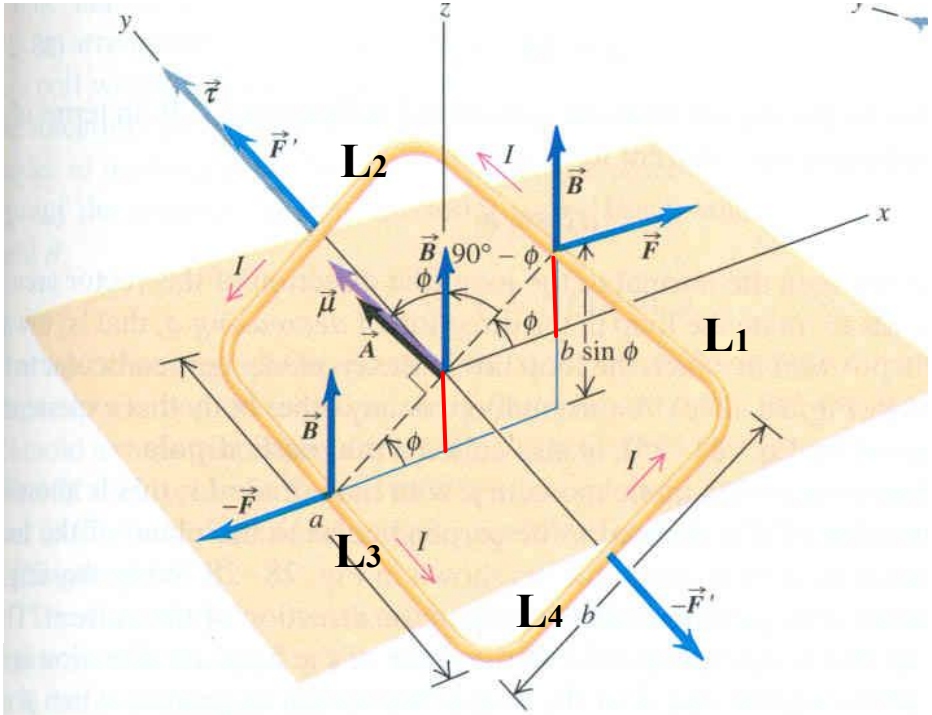
$$F = l \bullet B \bullet I$$

Length of conductor

Density of Magnetic field

Current

Understanding of Magnetic Force (2)



Electromagnetic force on L1:

$$F_1 = L_1 \bullet I \bullet B$$

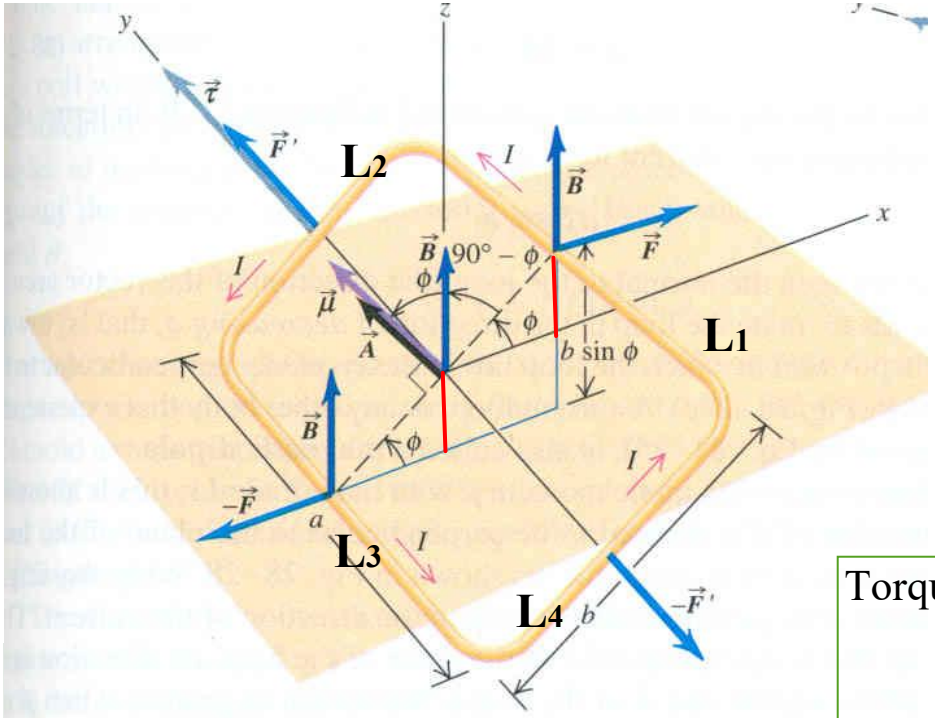
Electromagnetic force on L3:

$$F_3 = -L_3 \bullet I \bullet B$$

The distances from F2 and F4 to XY plane are always zero.

The distances from F1 and F3 to XY plane are not zero in general.

Understanding of Magnetic Force (2)



Distance from F_1 to XY plane is :

$$r_1 = \frac{b}{2}\sin(\phi)$$

Distance from F_3 to XY plane is :

$$r_3 = -\frac{b}{2}\sin(\phi)$$

Torque produced by F_1 about Y axis :

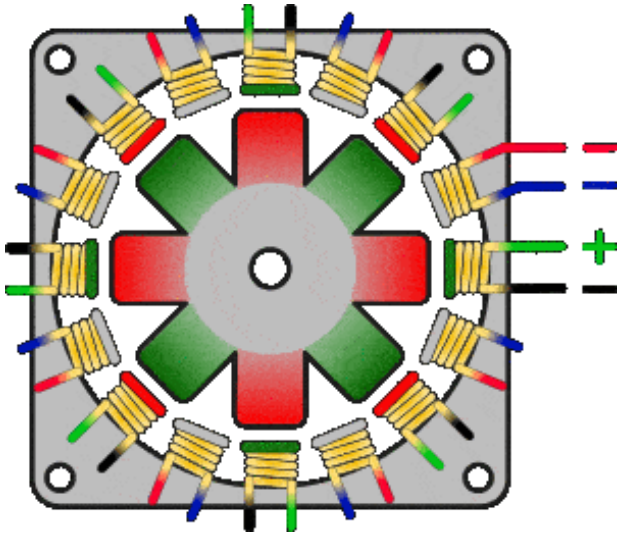
$$\tau_1 = (\frac{b}{2}\sin(\phi)) \bullet F_1 = (\frac{b}{2}\sin(\phi)) \bullet a \bullet I \bullet B$$

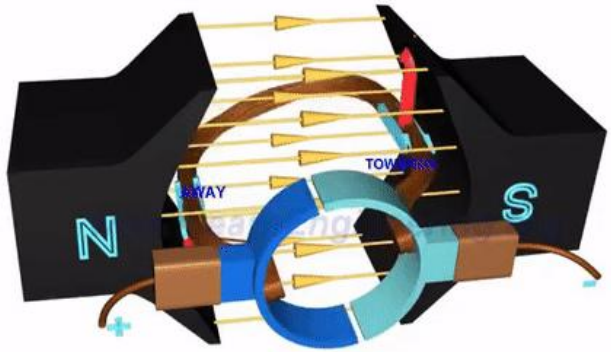
Torque produced by F_3 about Y axis ::

$$\tau_3 = (-\frac{b}{2}\sin(\phi)) \bullet F_3 = (\frac{b}{2}\sin(\phi)) \bullet a \bullet I \bullet B$$

Outline of Lecture 4

- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor



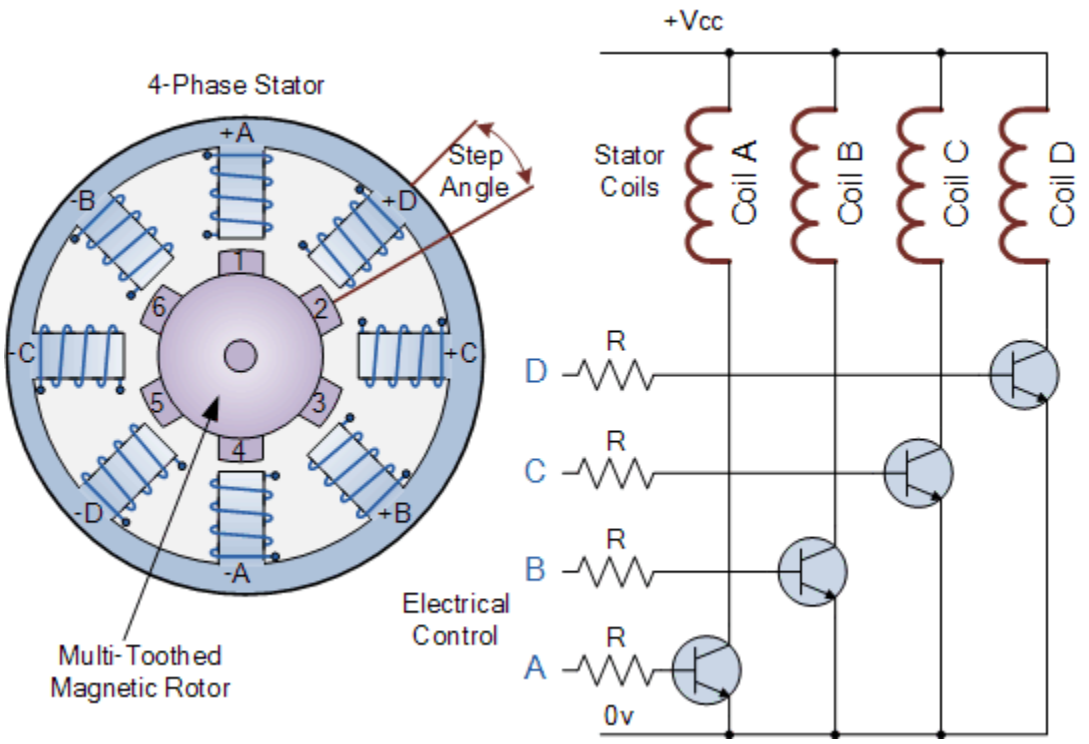


Video Showing Design and Working Principle

How to achieve the angular step of 1.8º ?

Summary of Design and Working Principle

- The rotating body is called rotor.
- The outer body is called stator.
- The rotor is made of permanent magnet.
- The stator has a set of electromagnets.
- When one pair of electromagnet is on, the rotor will make one step motion order to align with the stator's magnetic field.
- By changing the sequence of energizing the stator's electromagnet, we can make the rotator to make stepwise motions.



- Pitch-angle of rotor's teeth: 360/6
- Pitch-angle of stator's teeth: 360/8

Example

- A general stepper motor consists of rotor and stator. The number of teeth on the rotor is 16 while the number of teeth on the stator is 18. What is the rotated angle of each step by the motor?

- Answer:

What is drawback?

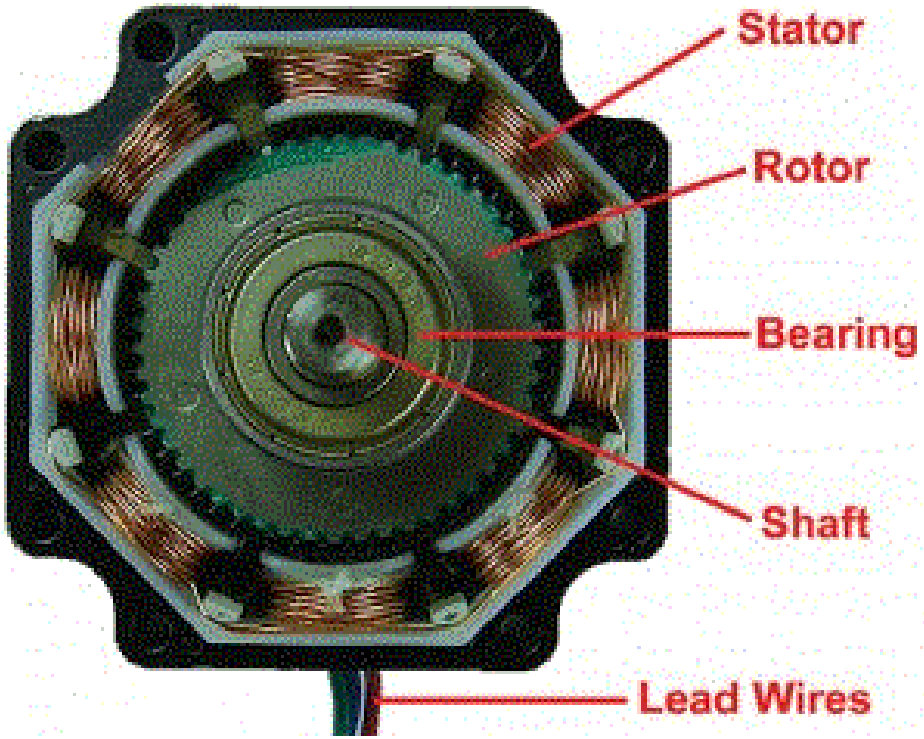
Pitch angle of the rotor is: $\frac{360}{16} = 22.5$

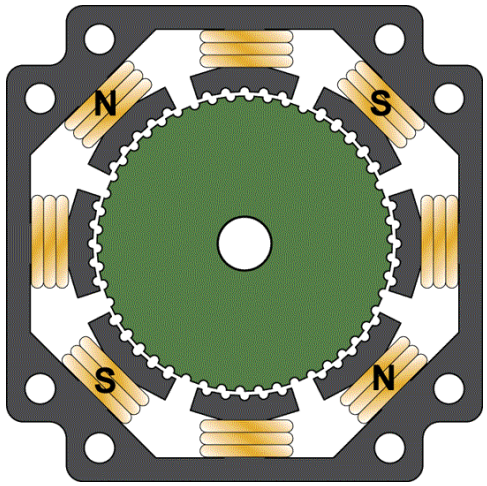
(too many phases)

Pitch angle of the stator is: $\frac{360}{18} = 20.0$

Rotated angle per step is: 22.5 – 20.0 = 2.5 degrees

Design Example of 2-Phase Stepper Motor





2-Phase Motor

$$1.8^o = \frac{360}{50} \times \frac{1}{4}$$

- The pitch-angles of rotor and stator are the same: 360/50
- The teeth of stator are divided into four groups after taking out two teeth.
- The angular displacement of two adjacent groups is: pitch-angle/4

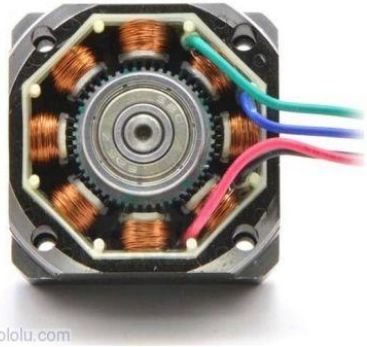
Design Example of 2-Phase Stepper Motor

CUSTOM HOUSING

We can design and manufacture motors specific to your environmental, mechanical, and dimensional requirements.

ENCODERS, DAMPERS, GEARBOXES, & MECHATRONICS

Added value assemblies



MULTIPLE MOUNTING CONFIGURATIONS

NEMA Size
8, 11, 14, 17, 23 & 34

BEARINGS & LUBRICANTS

Ball Bearings, Stainless Steel Bearings, Seals, Special Lubricants for high temperature/ humid operation

MULTIPLE SHAFT OPTIONS

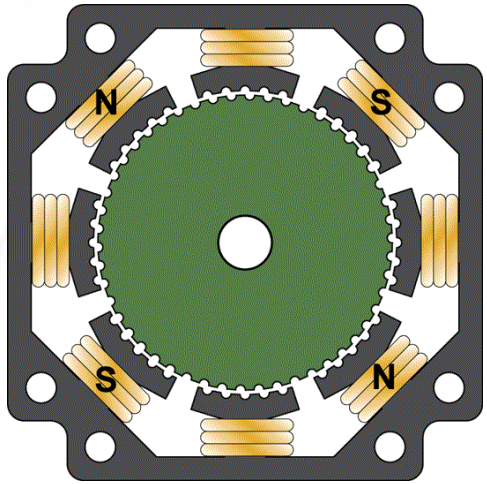
Slotted, cross drilled, oversized, hollow, extended, helical cut, press fit ger and pulley...

CUSTOM WINDING

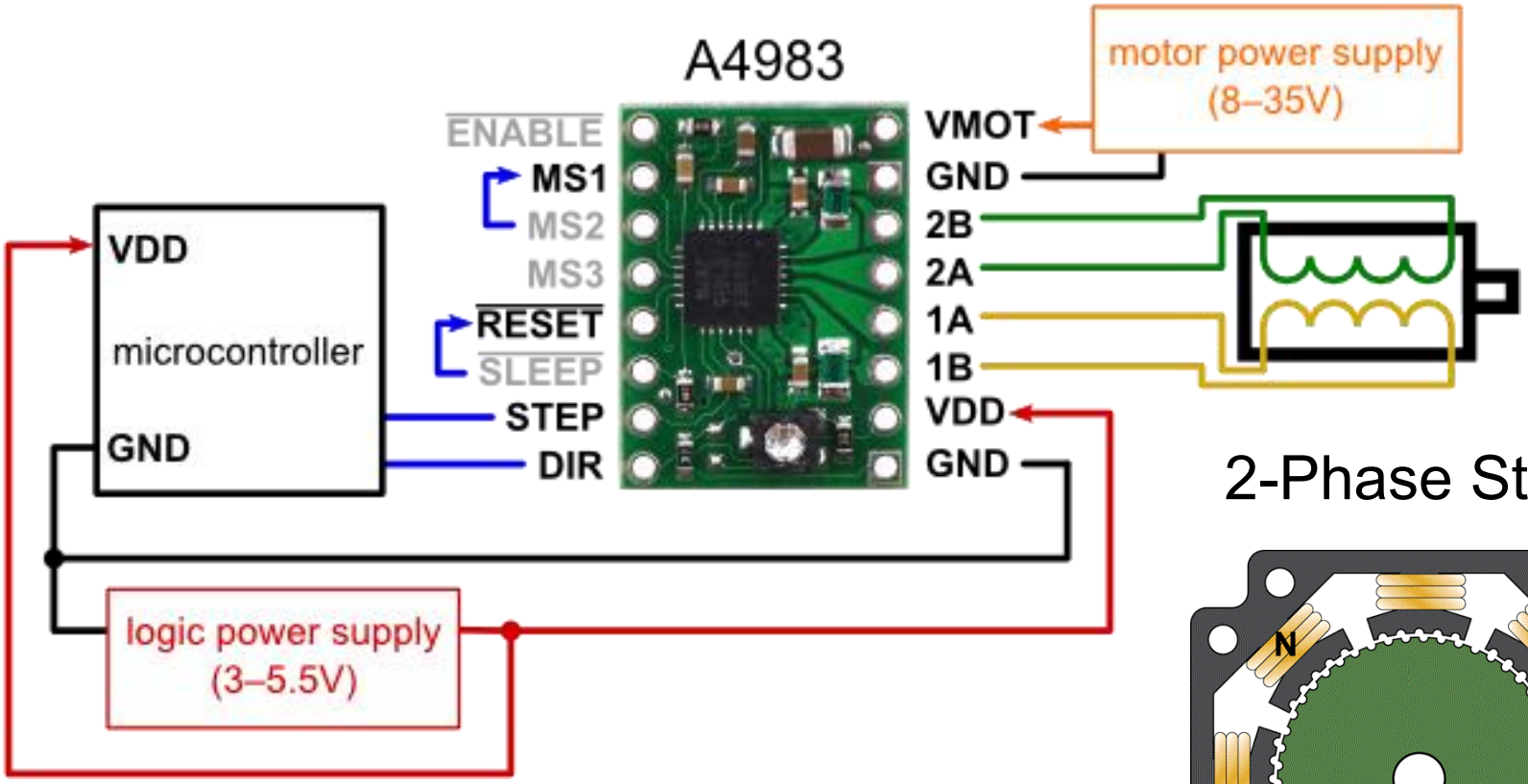
We can provide custom windings (at no extra cost) based on your application needs.

LEAD WIRES & CABLES

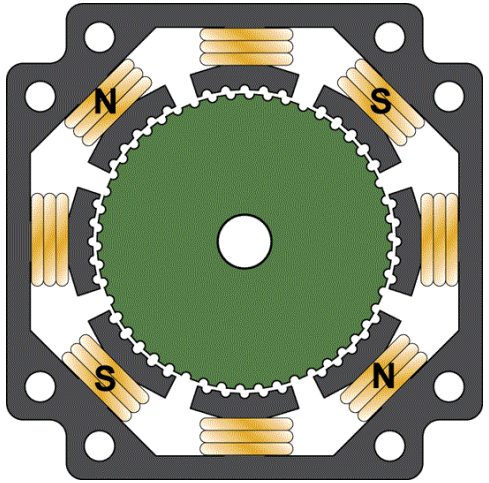
Special lengths, heat shrink tube, custom color code, teflon leads, insulated wire, pins, connectors, ...



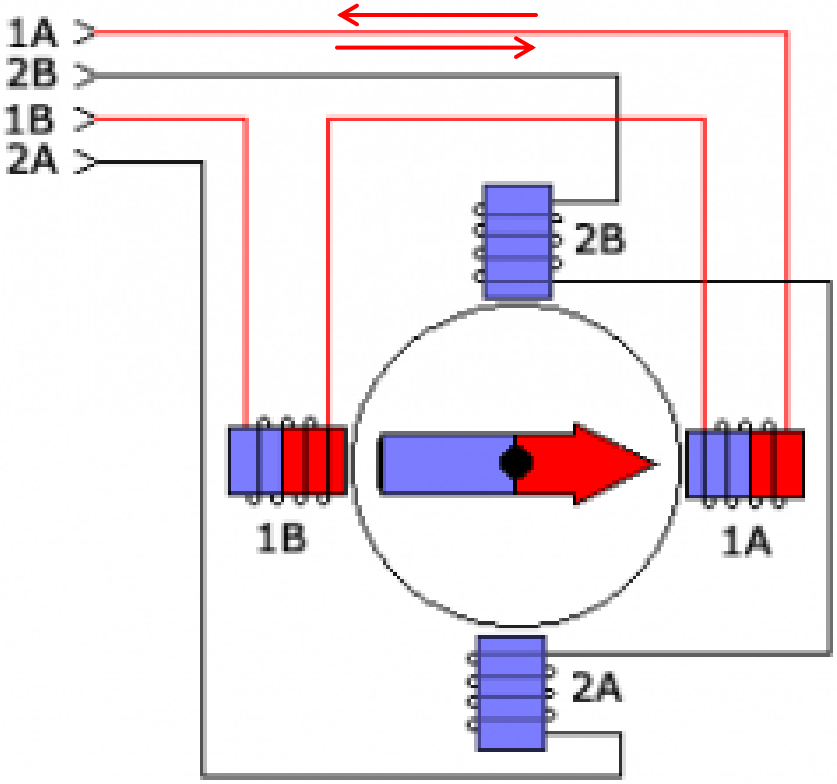
Design Example of Power Supply



2-Phase Stator

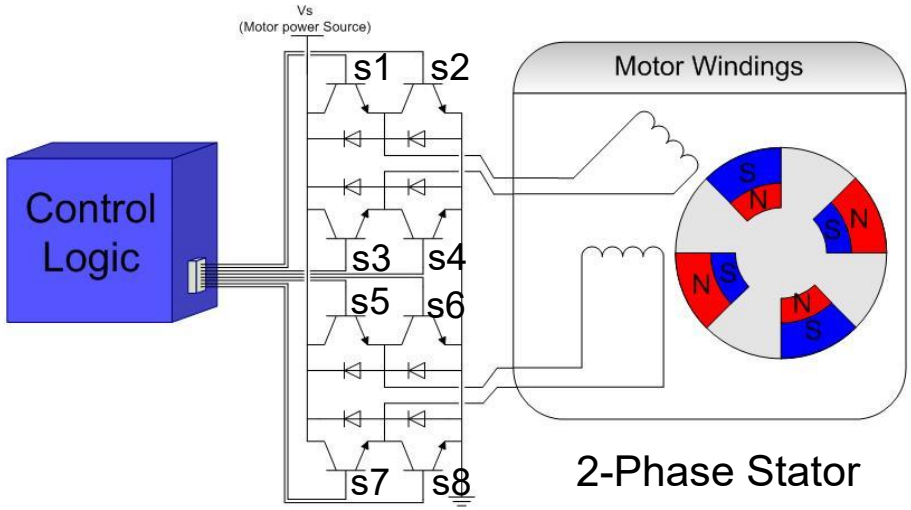


Design Example of Output Control



Clockwise: 1A->2A->1B->2B

Counterclockwise: 1A->2B->1B->2A



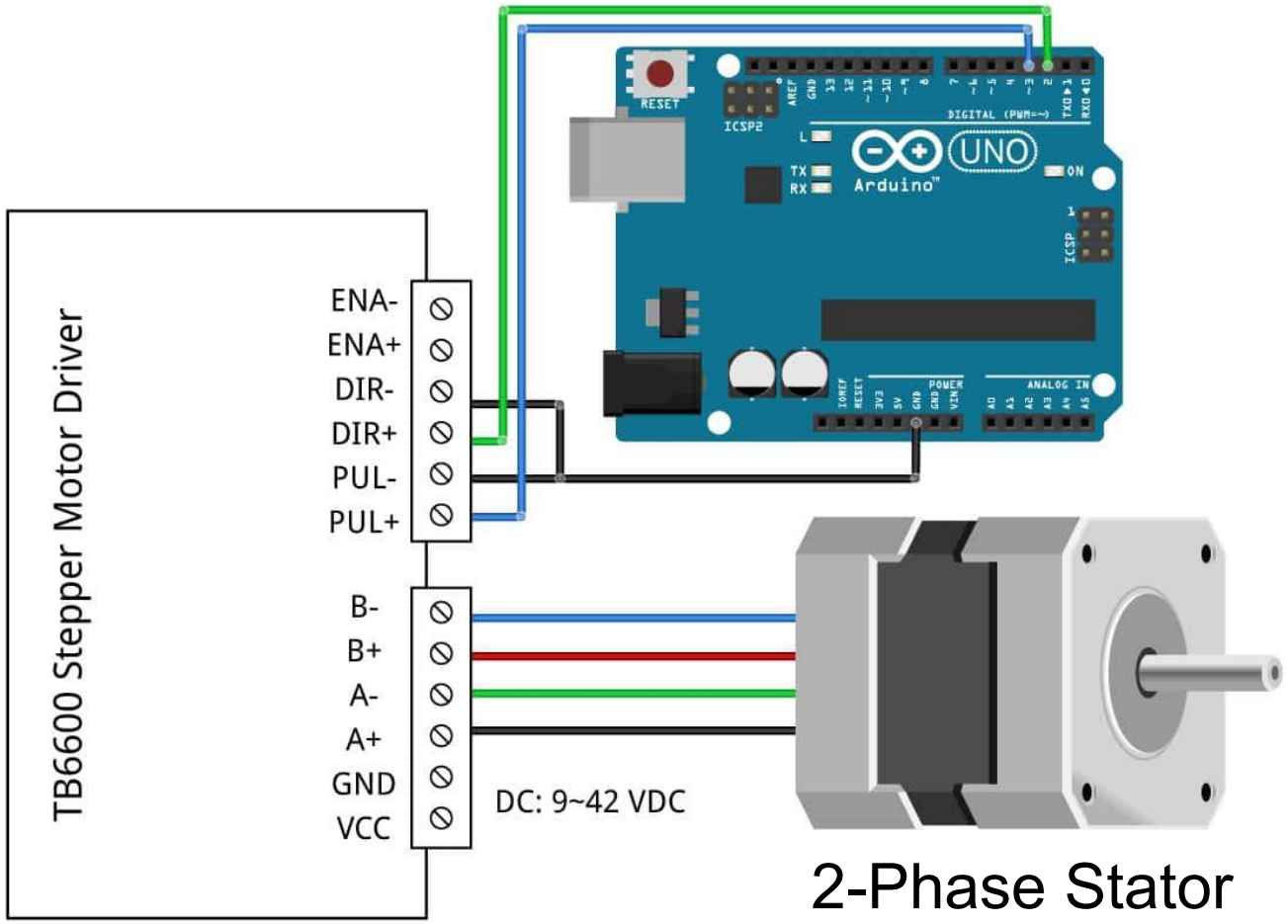
Coils 1A-1B:

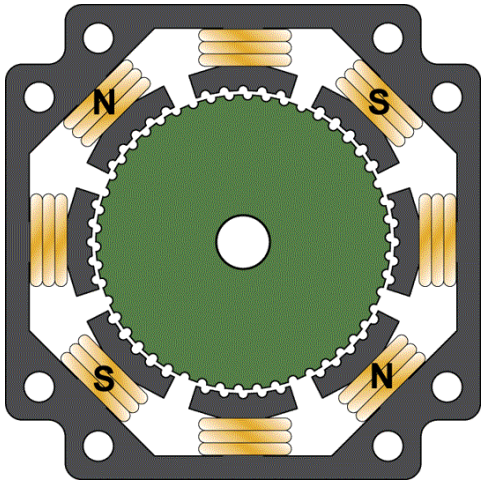
1. South-North: s1=on, s4=on
2. North-South: s2=on, s3=on

Coils 2A-2B:

1. South-North: s5=on, s8=on
2. North-South: s6=on, s7=on

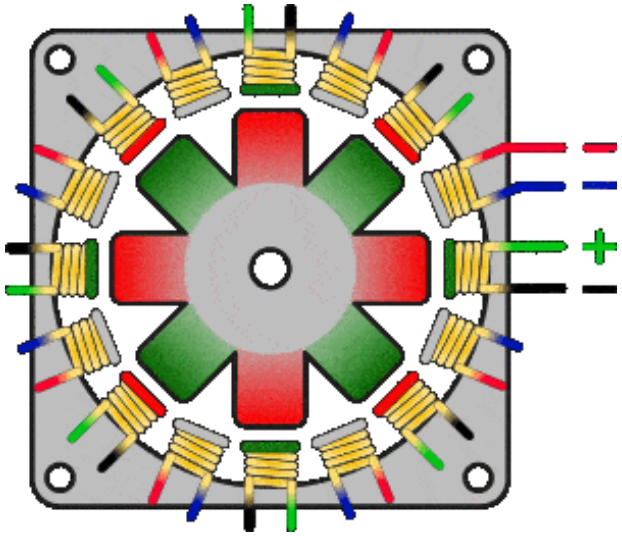
Example of Interfacing with Robot Brain

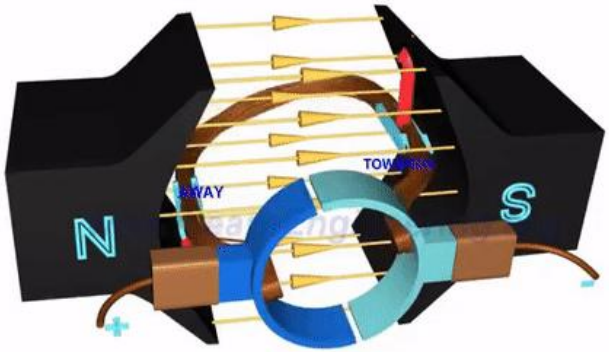




Outline of Lecture 4

- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor

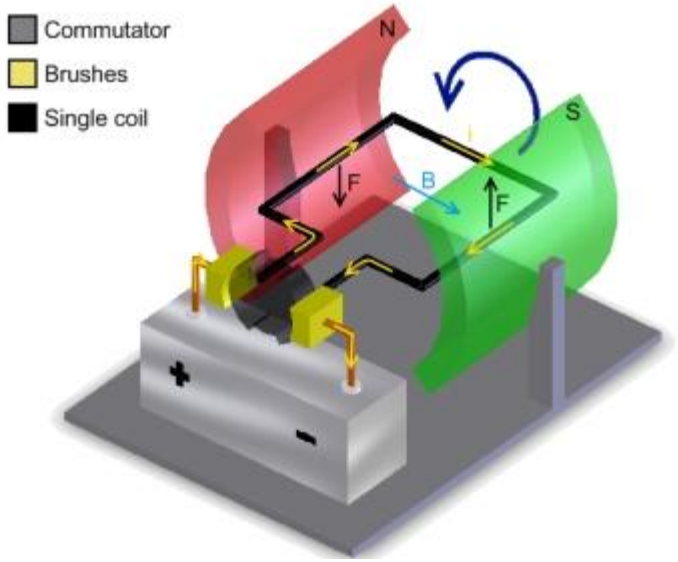


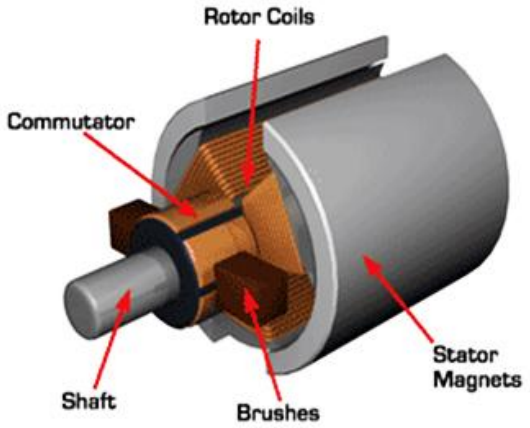


Video Showing Design and Working Principle

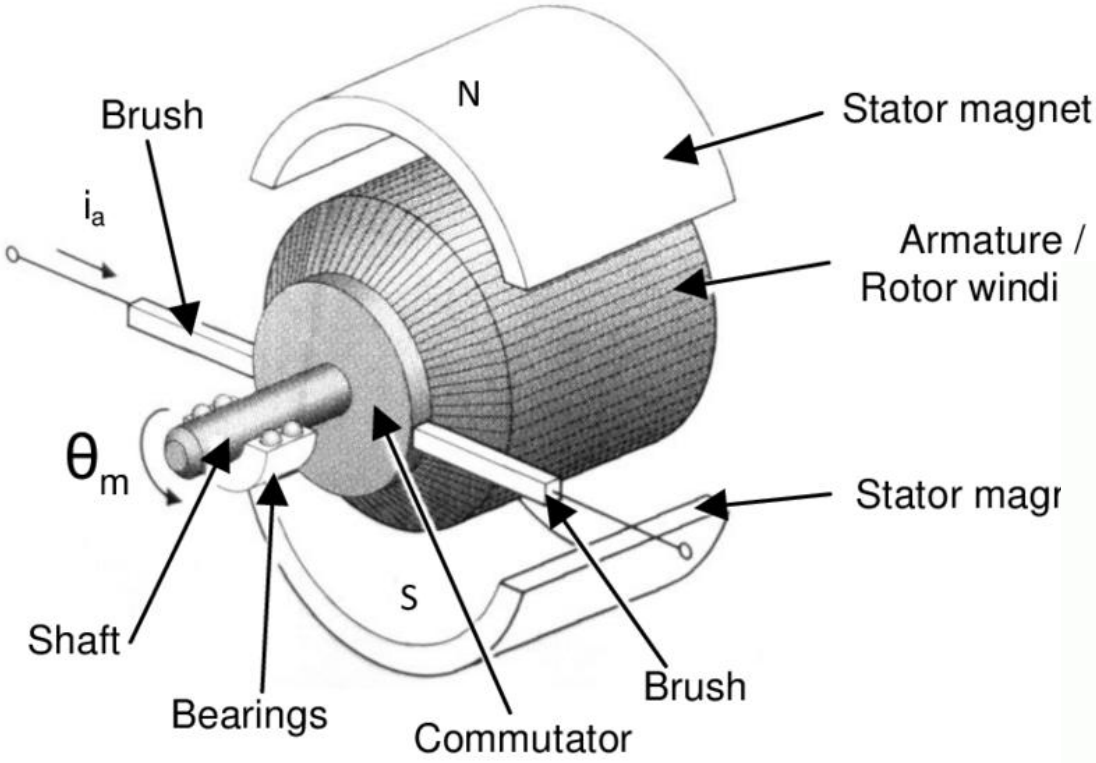
Summary of Design and Working Principle

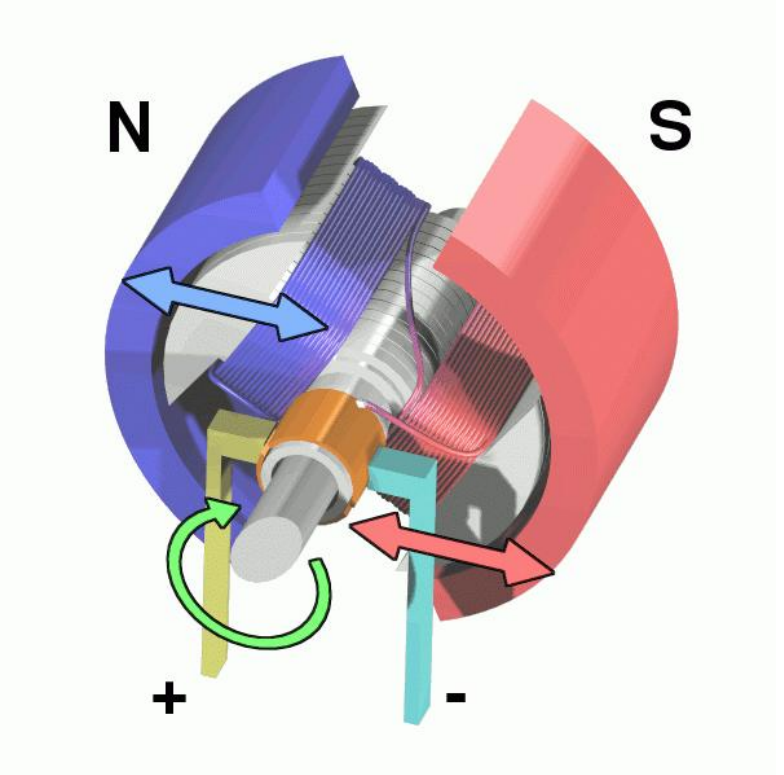
- The stator is made of permanent magnet.
- The rotor has a set of rectangular coils.
- When the rectangular coils are parallel to the magnetic field, current will pass through the coils. And, the edges perpendicular to the magnetic flux will produce magnetic forces, which make the coils to rotate in one direction.
- The motion of rotor will bring another rectangular coils to be parallel to the magnetic field. The process repeats.



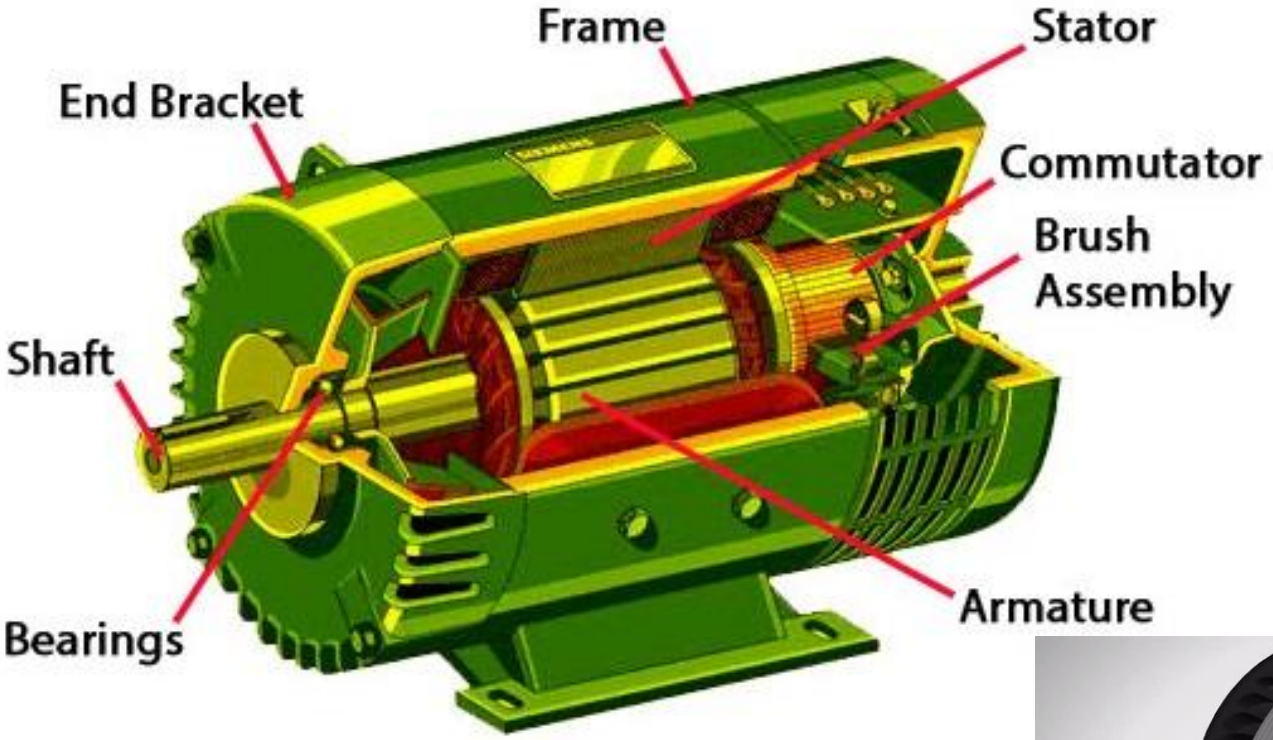


Design Example of Brushed DC Motor

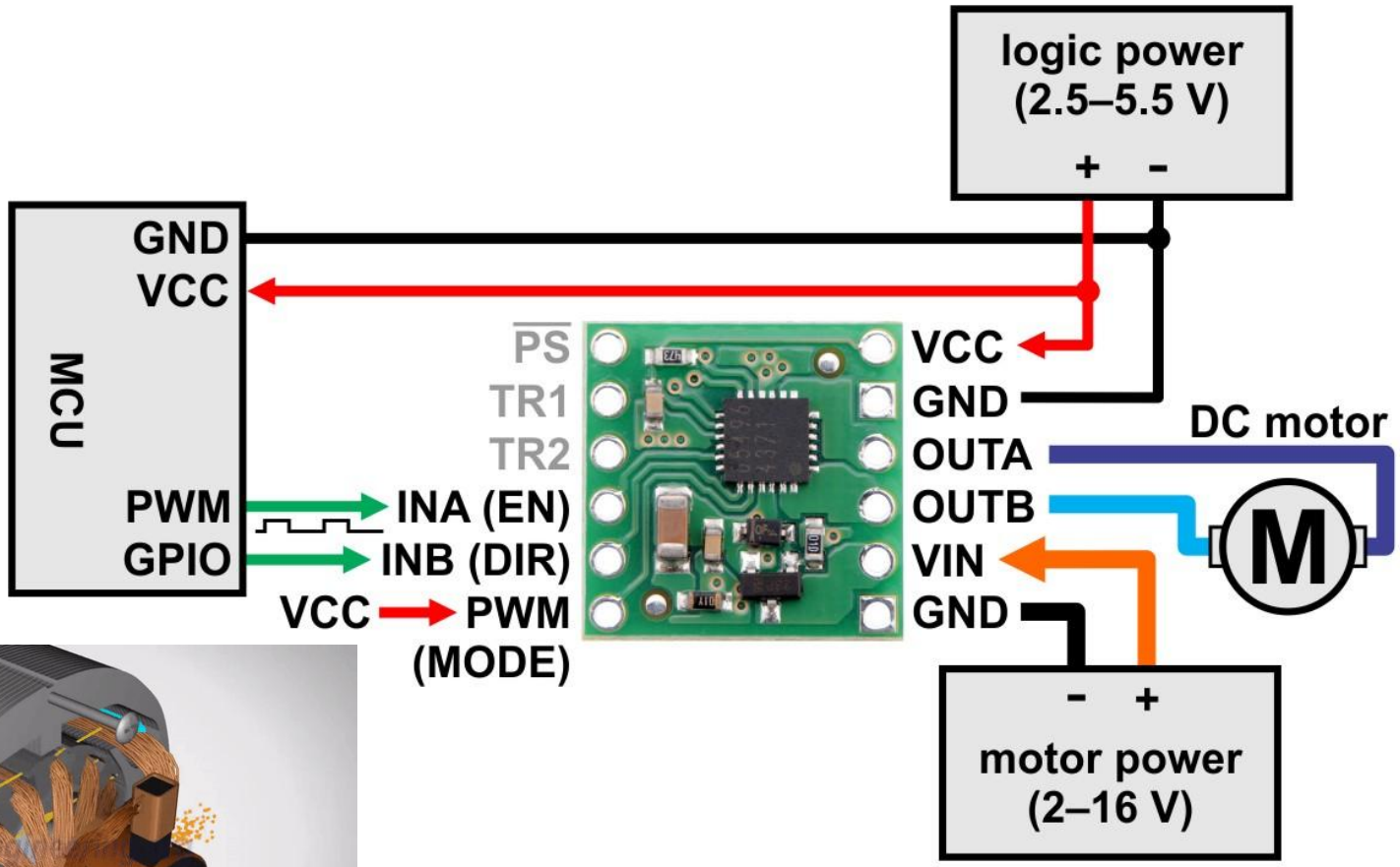


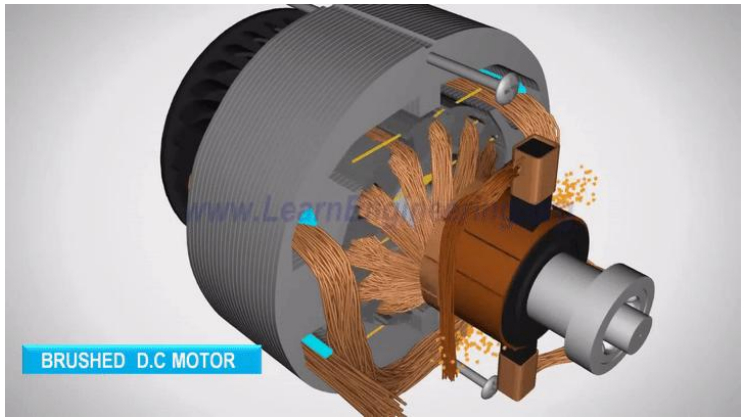


Design Example of Brushed DC Motor

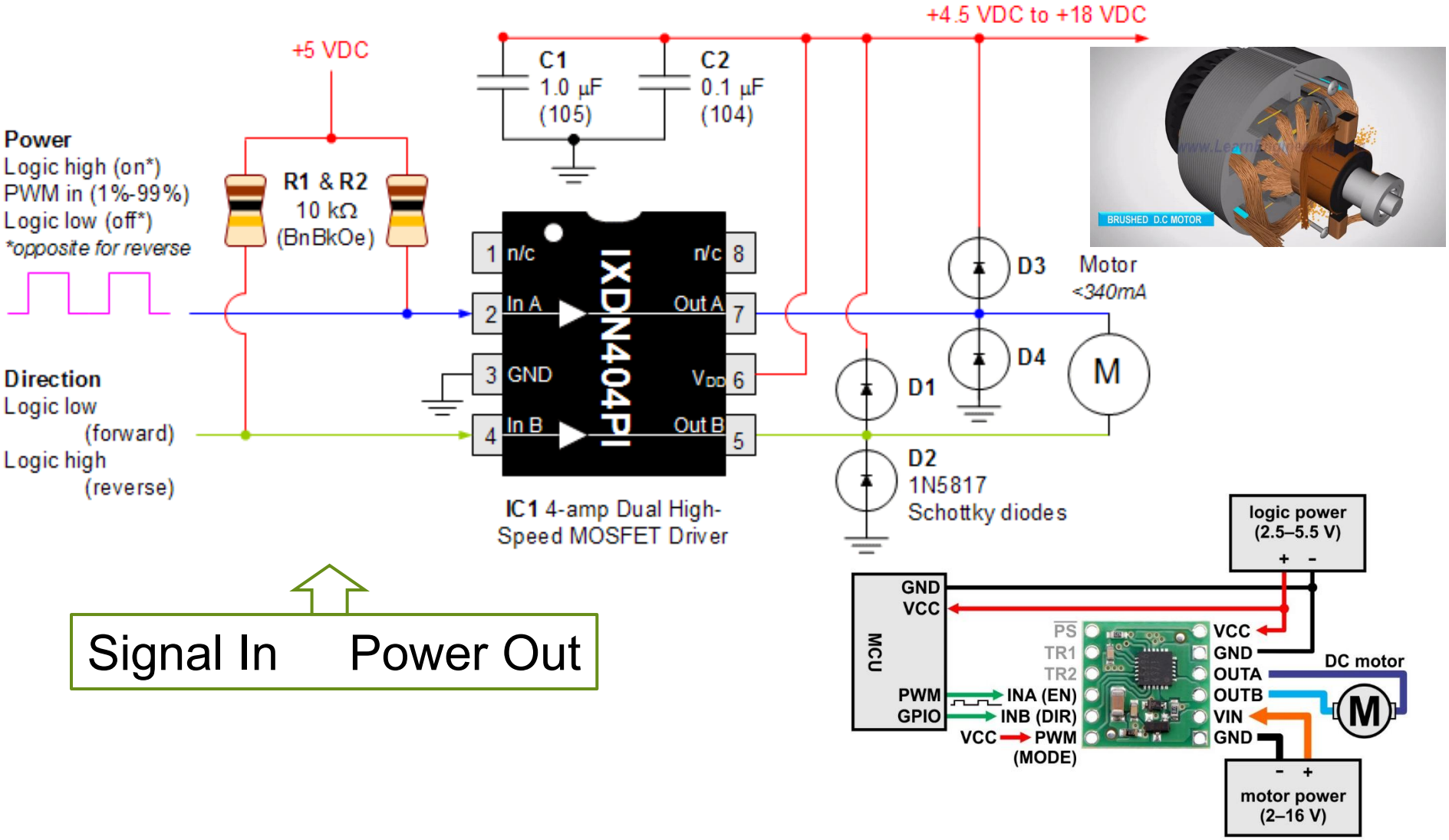


Design Example of Power Supply (1)



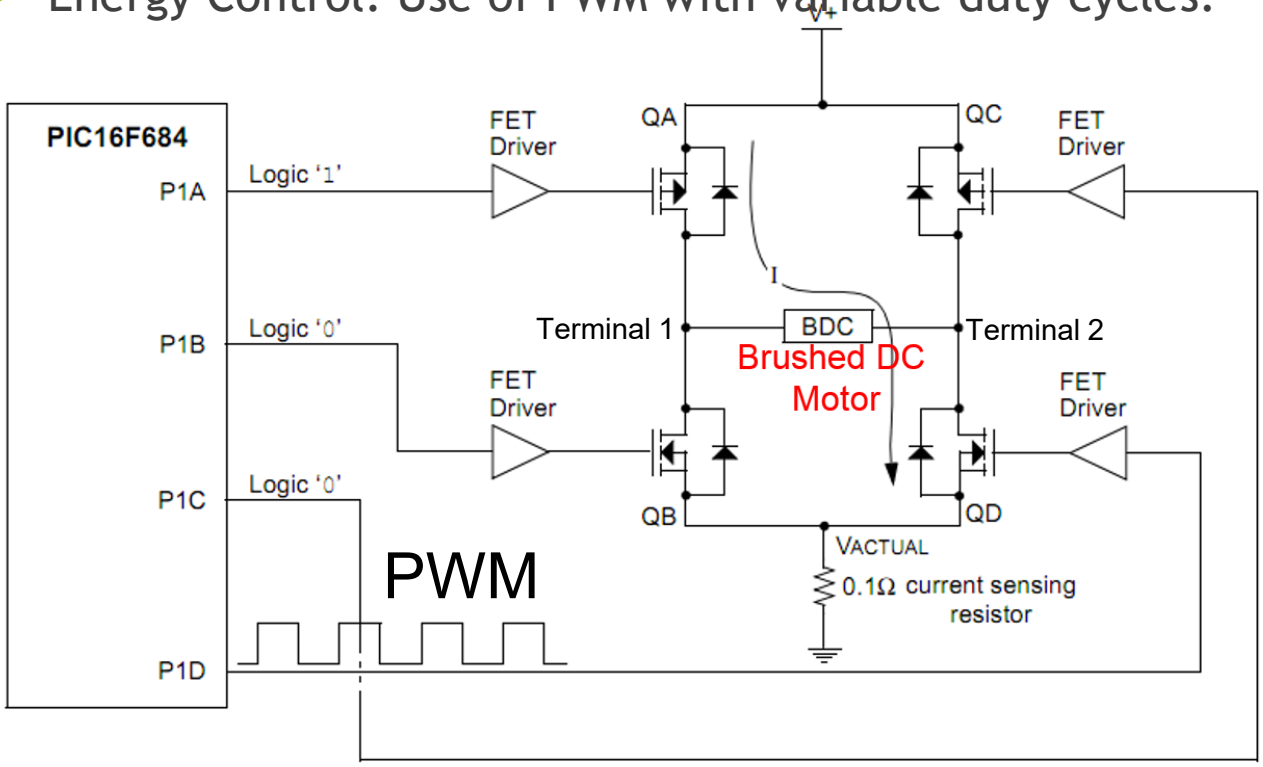


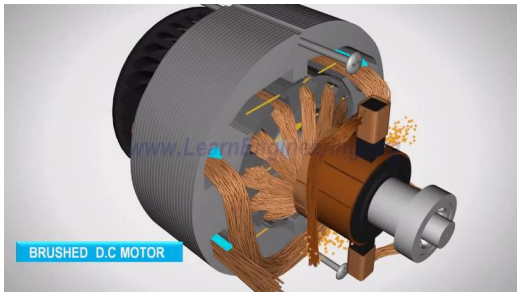
Design Example of Power Supply (2)



Design Example of Output Control

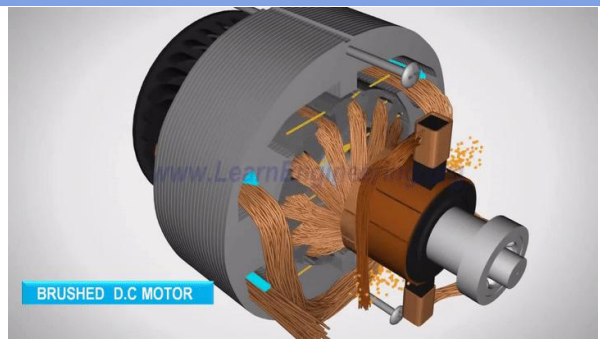
- Direction Control:
  - Clockwise (CW) Rotation: QA is in logic high, QD supplies modulated pulses.
  - Counter-clockwise (CCW) Rotation: QC is in logic high, QB supplies modulated pulses.
- Energy Control: Use of PWM with variable duty cycles.

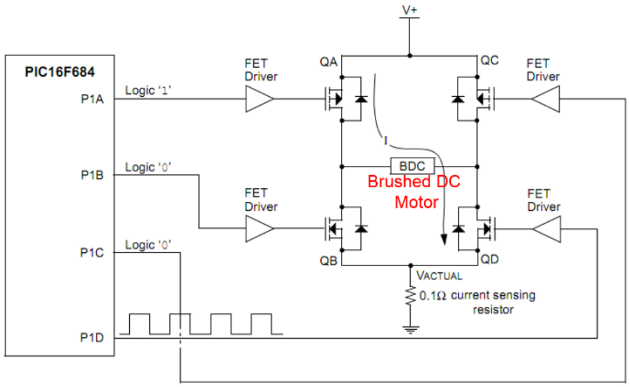




Example of Interfacing with Robot Brain

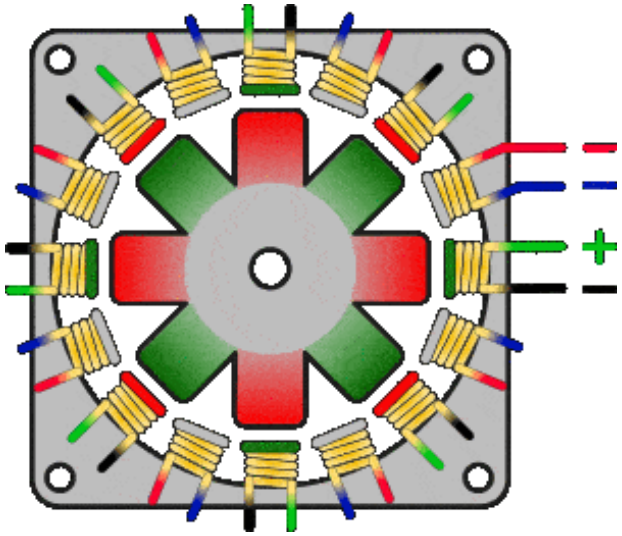
- Direction Control
- Energy Control

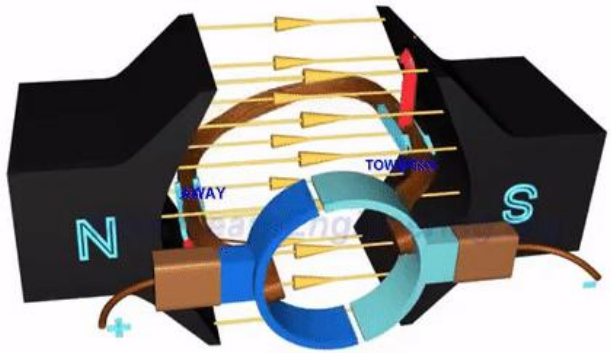




Outline of Lecture 4

- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor

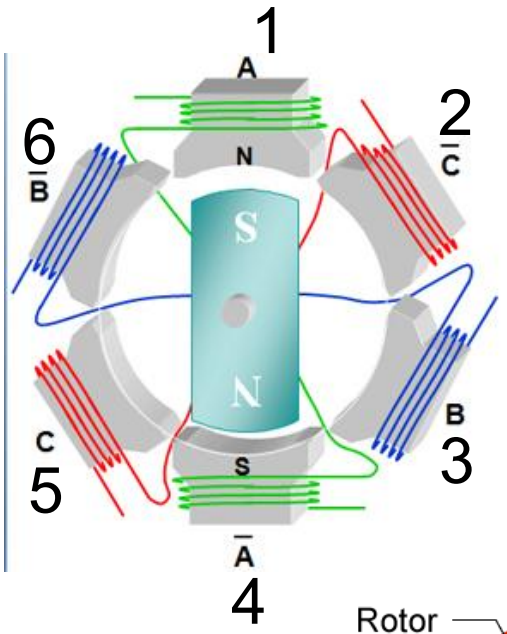


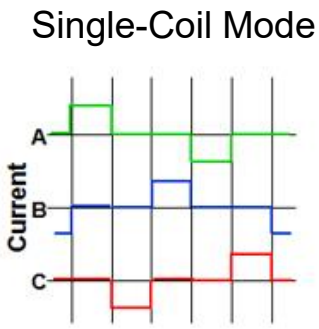


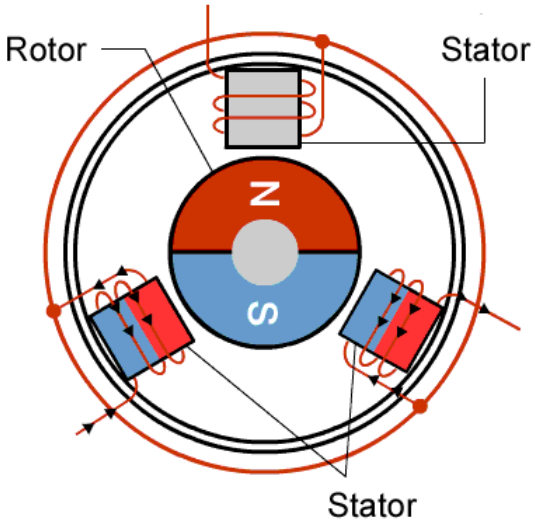
Video Showing Design and Working Principle

Summary of Design and Working Principle

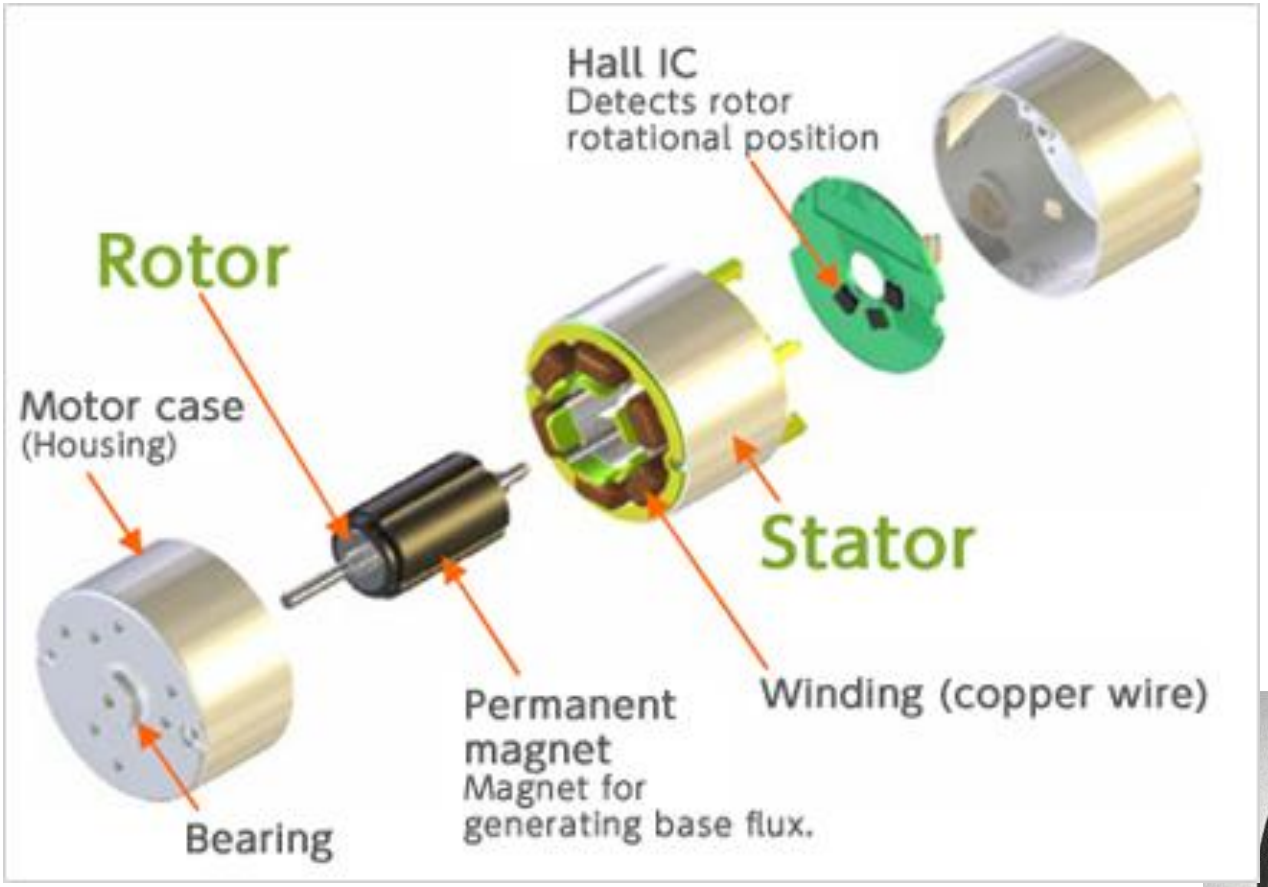
- The rotor is made of permanent magnet.
- The stator has a set of electromagnets.
- The stator's electromagnets are sequentially excited in order to create a rotating magnetic field.
- The stator's rotating magnetic field will cause the rotator to follow the rotation.
- The hall sensors are employed to provide the position feedback so as to synchronize the commutation of excitations.

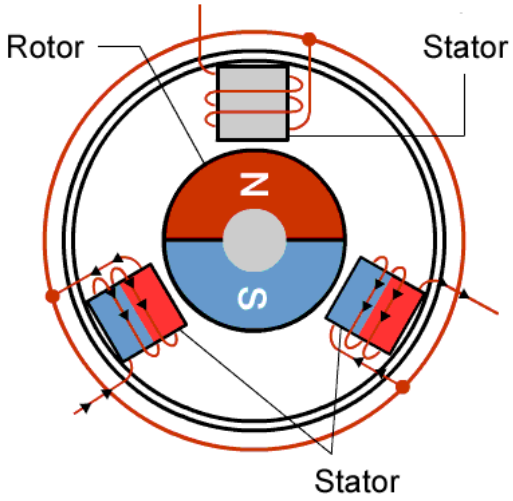


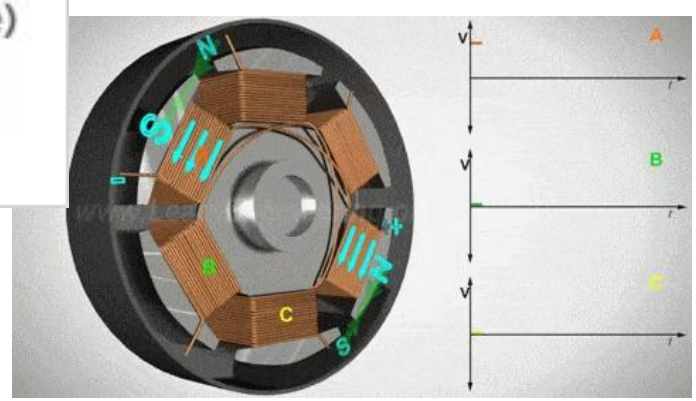




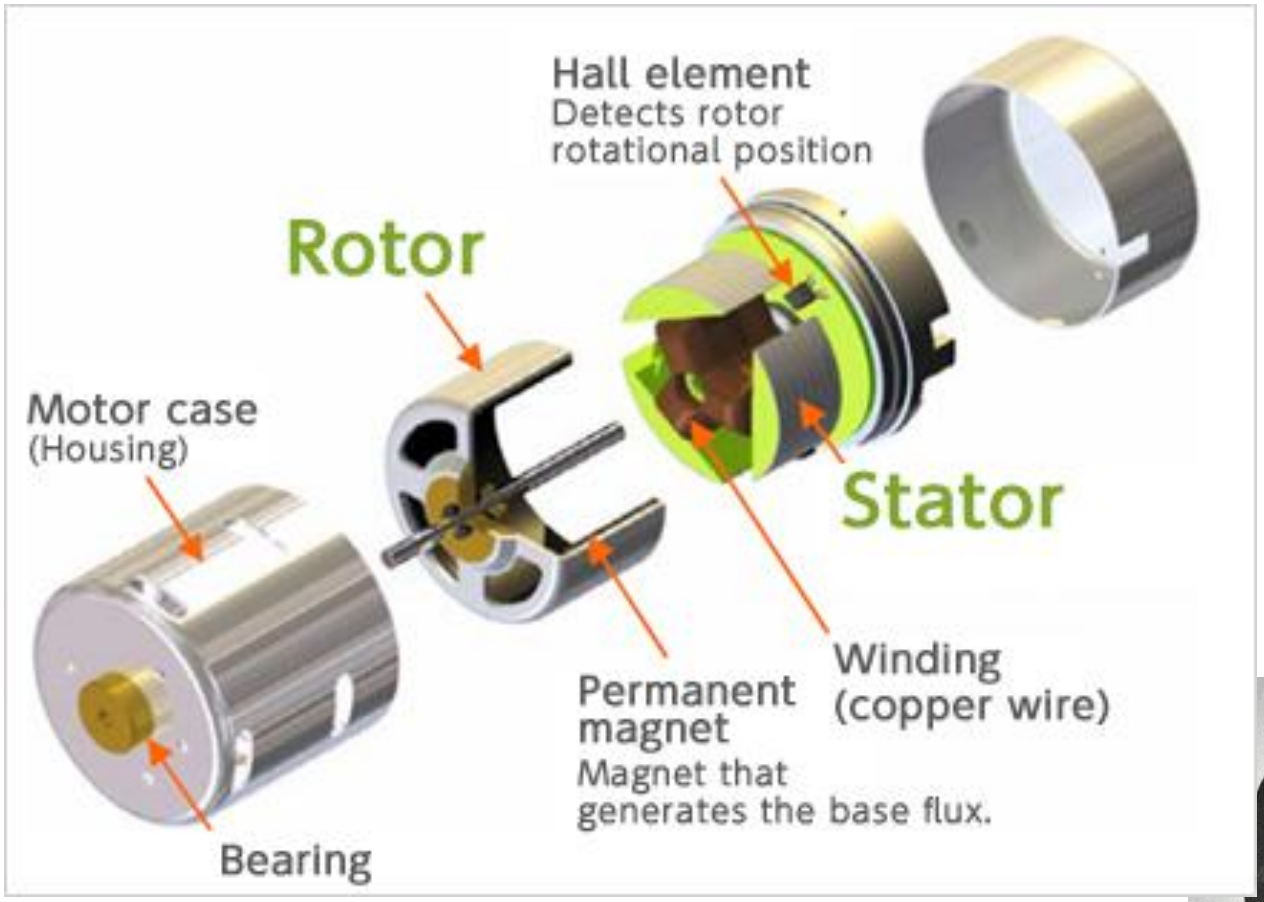
Design Example of BLDC Motor

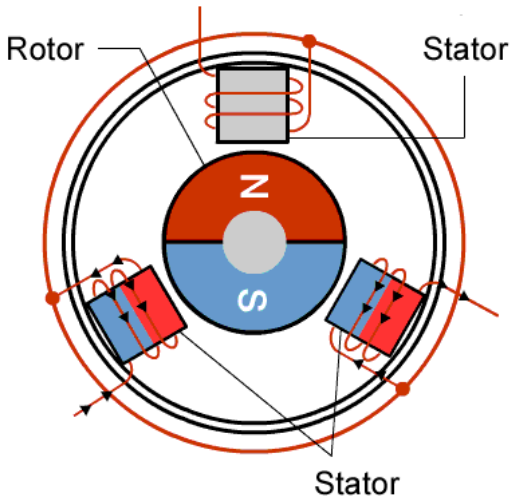


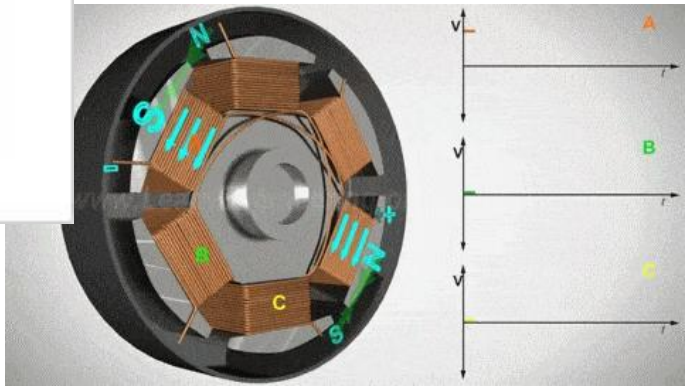




Design Example of BLDC Motor

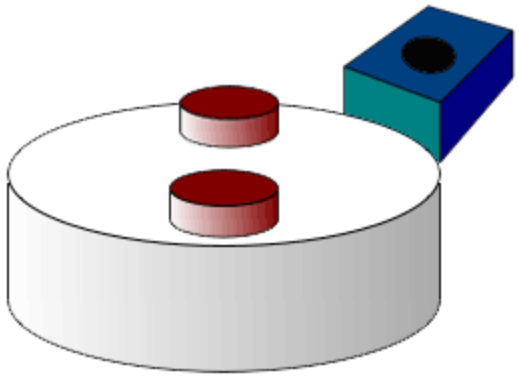




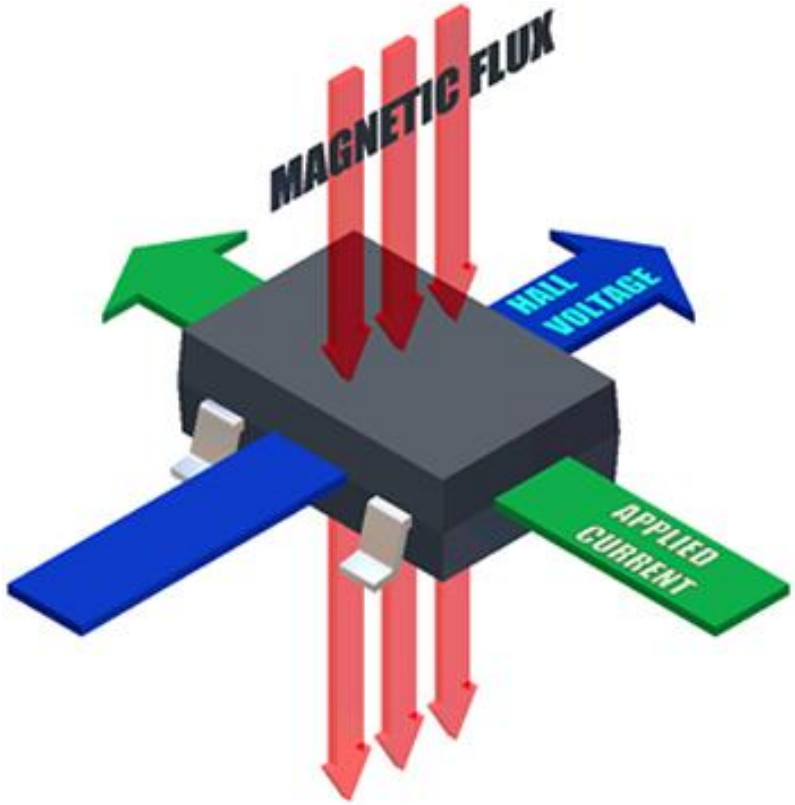


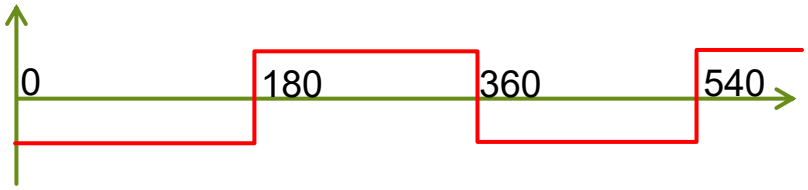
Working Principle of Hall Sensor

- When magnetic flux passes through the sensor, a voltage is produced as output.

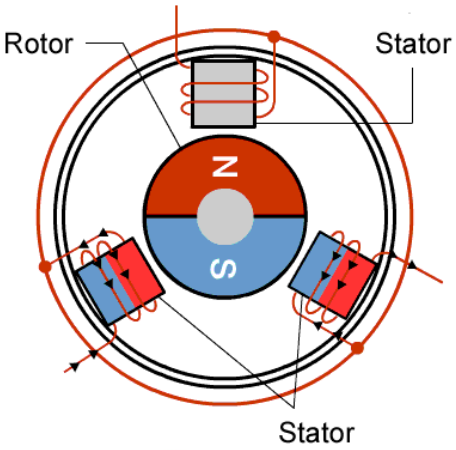


Hall Effect Sensor

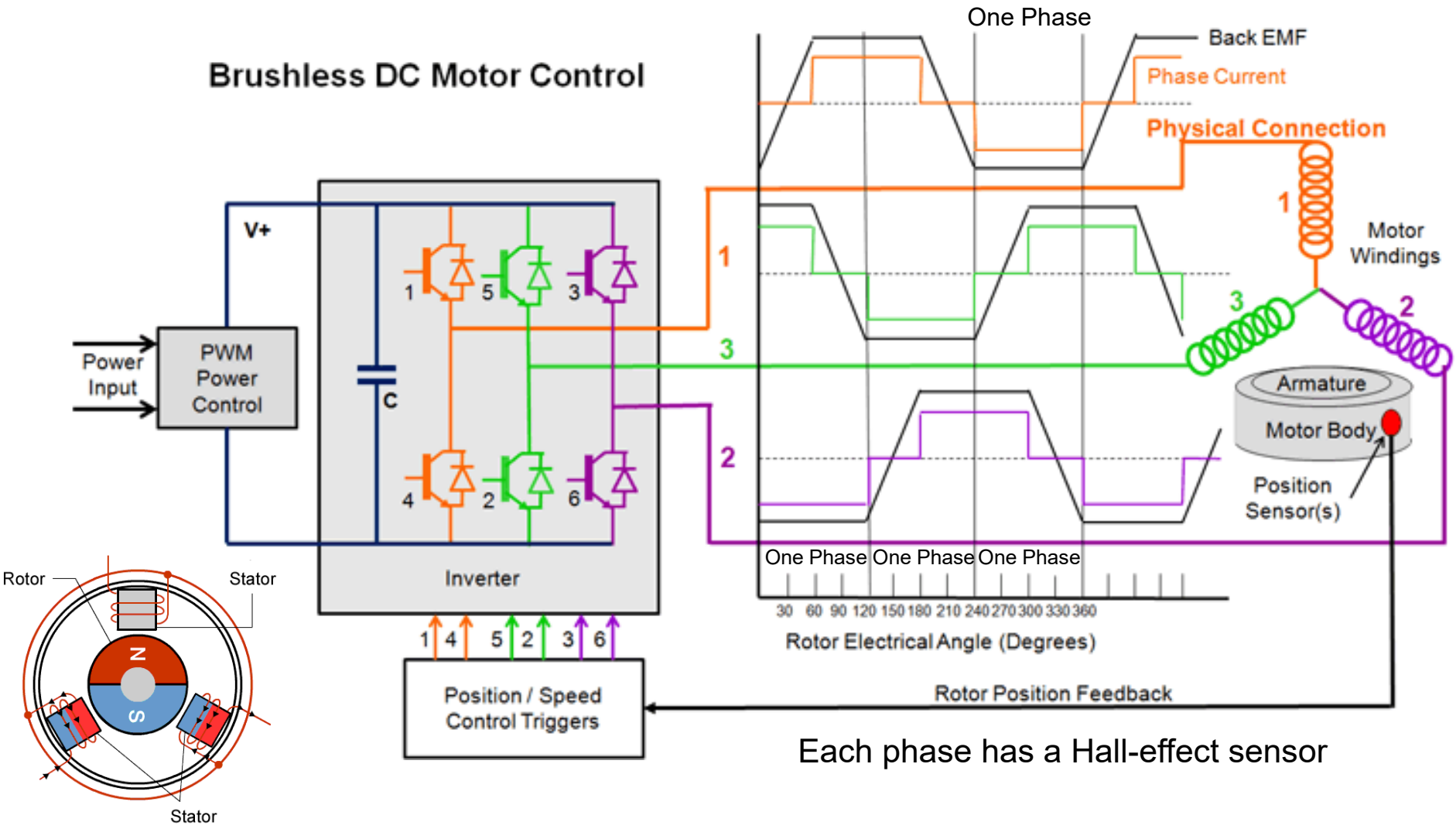




Design Example of Power Supply



Design Example of Output Control

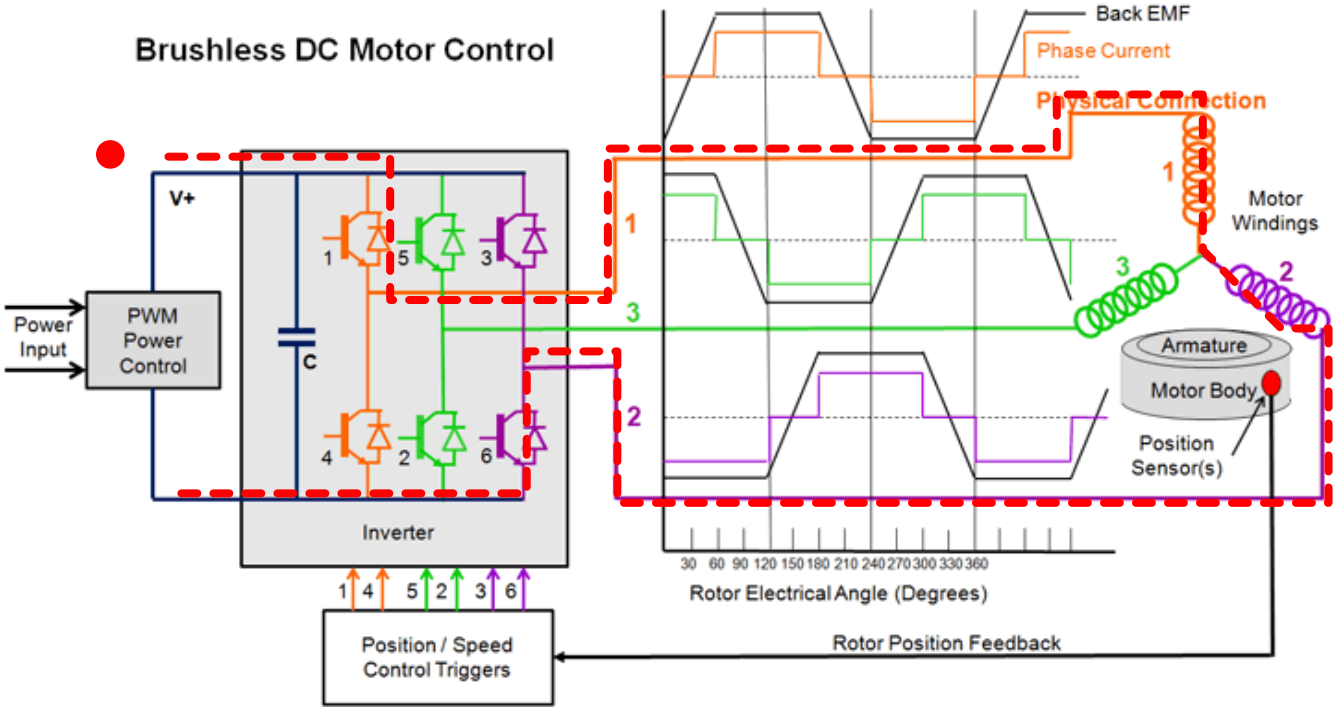


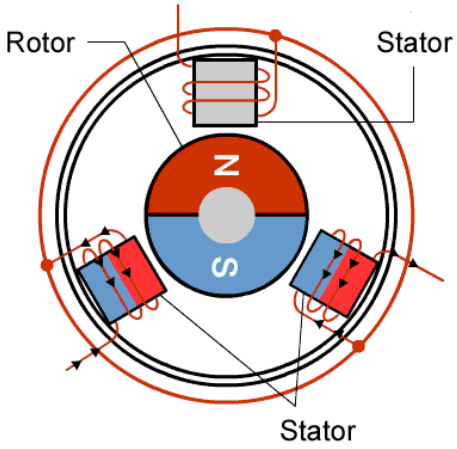
Each phase has a Hall-effect sensor

Example of Controlling Direction

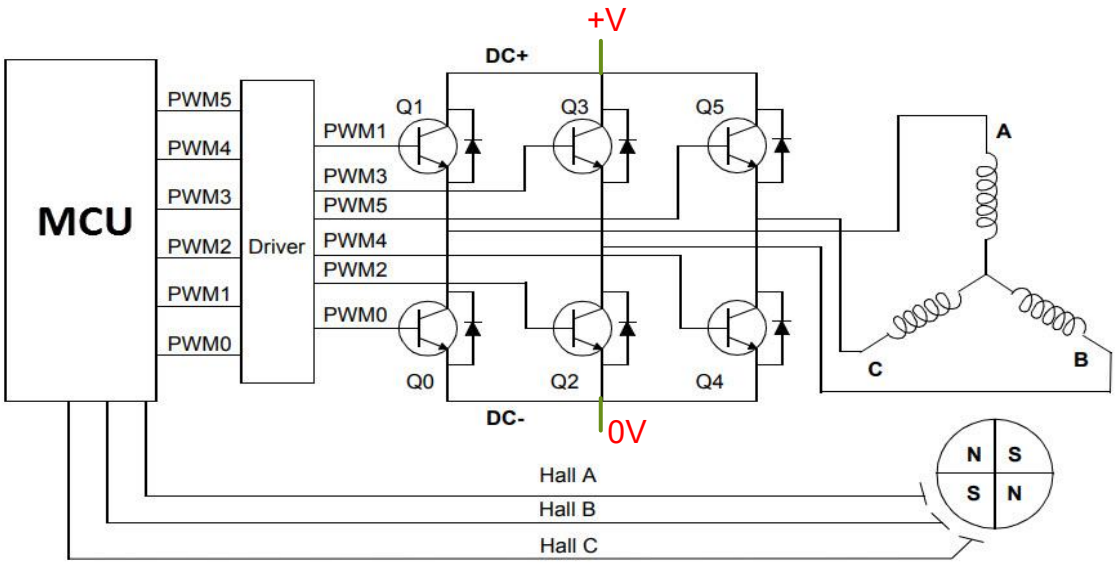
- Draw the flow of current when the switches 1 and 6 are on.

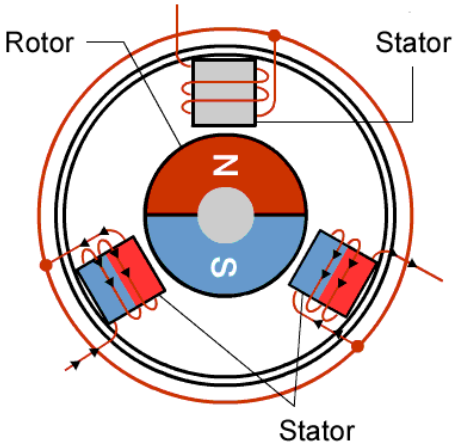
- Answer:

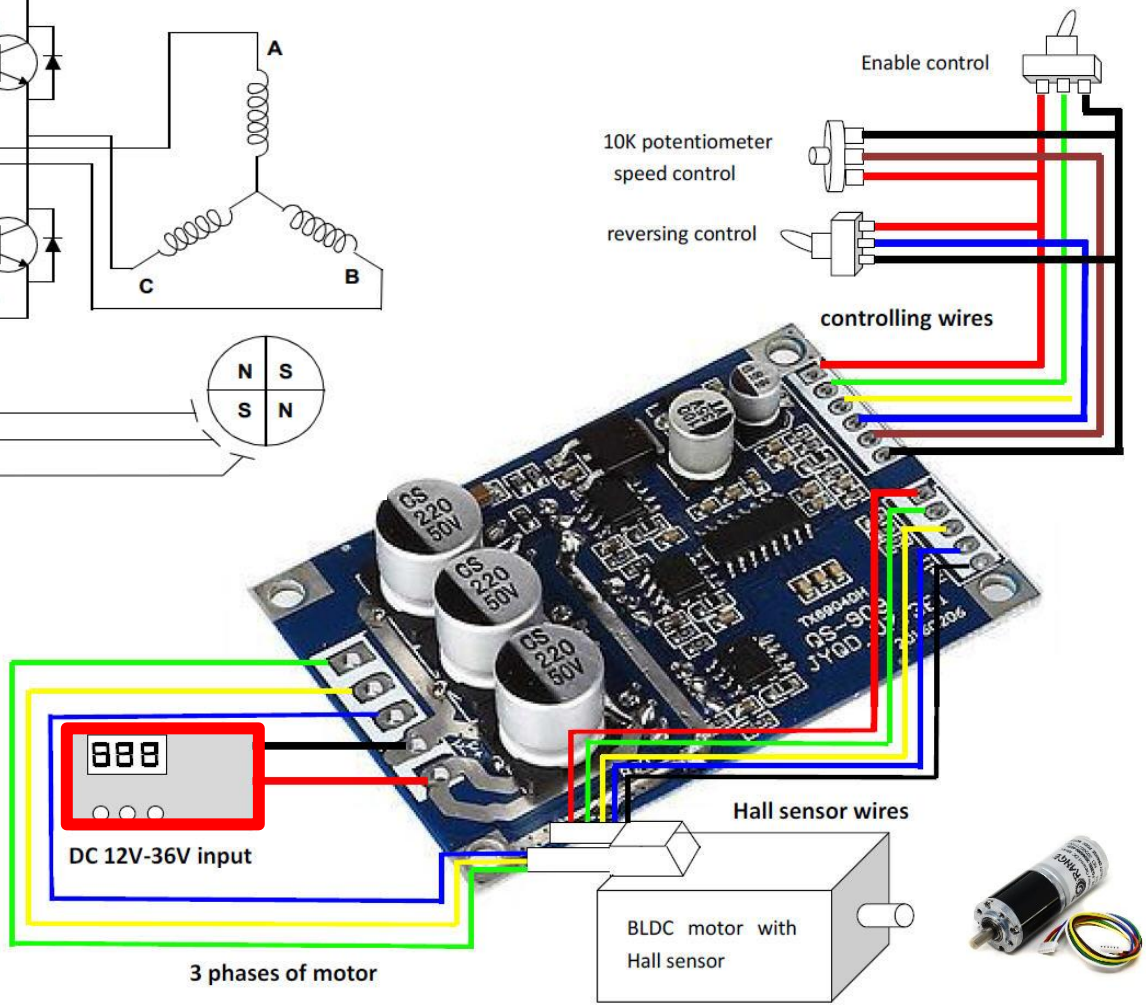




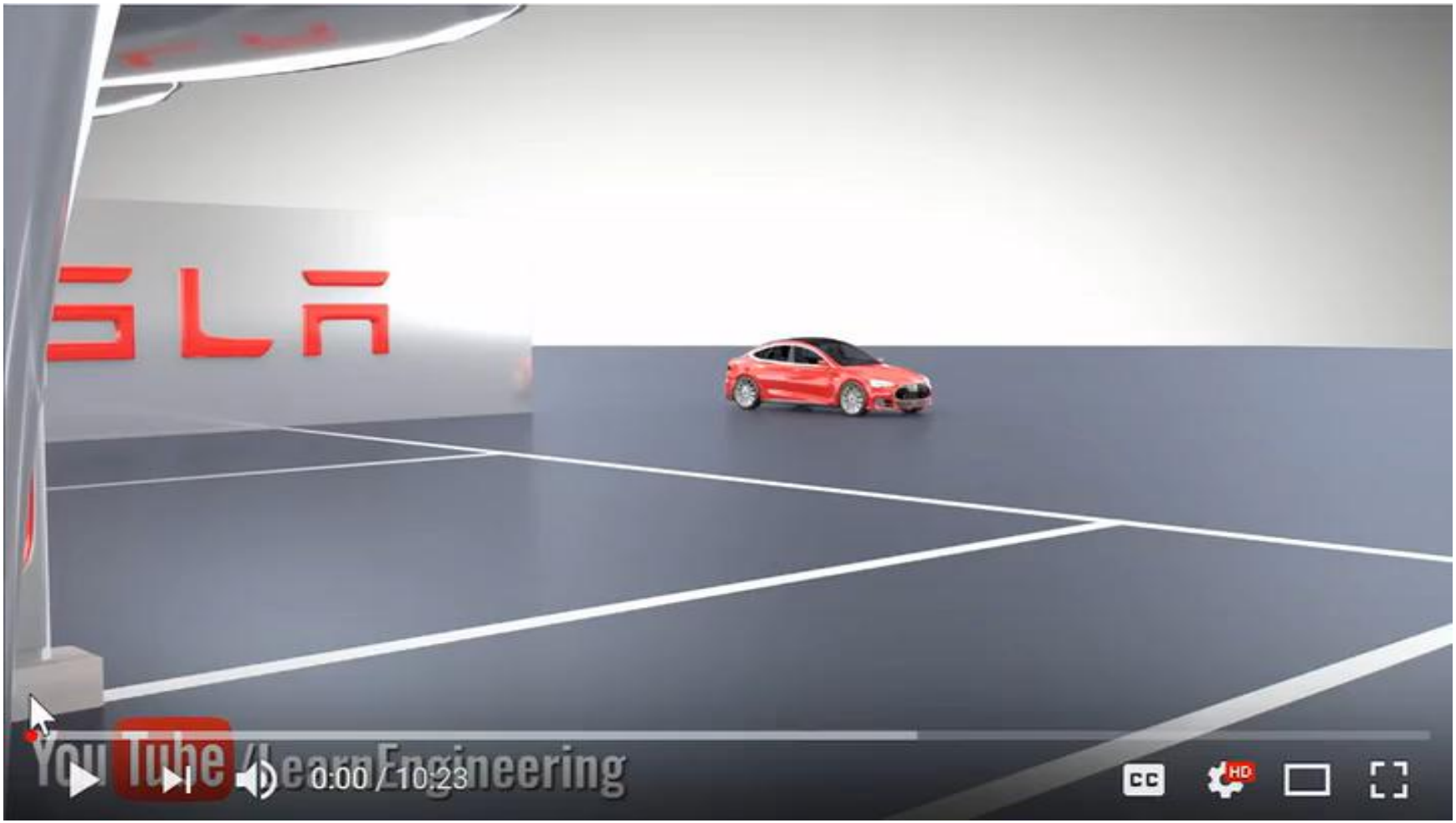
Example of Interfacing with Robot Brain





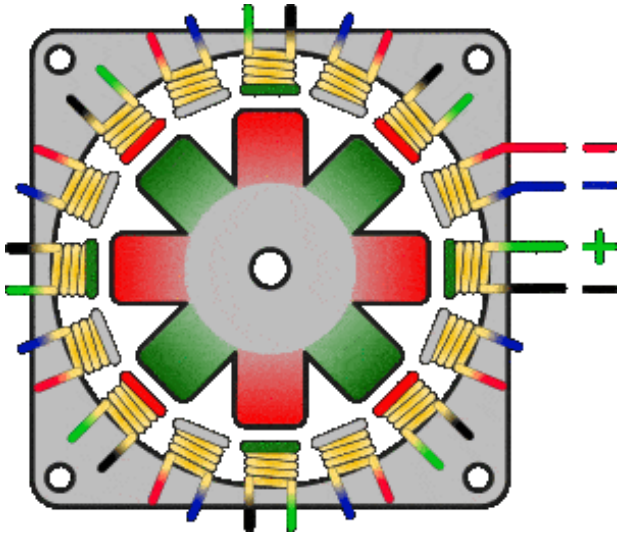


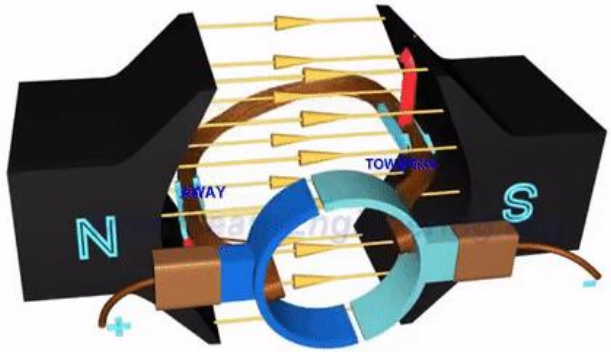
Application of BLDC Motors to Electric Cars



Summary of Lecture 4

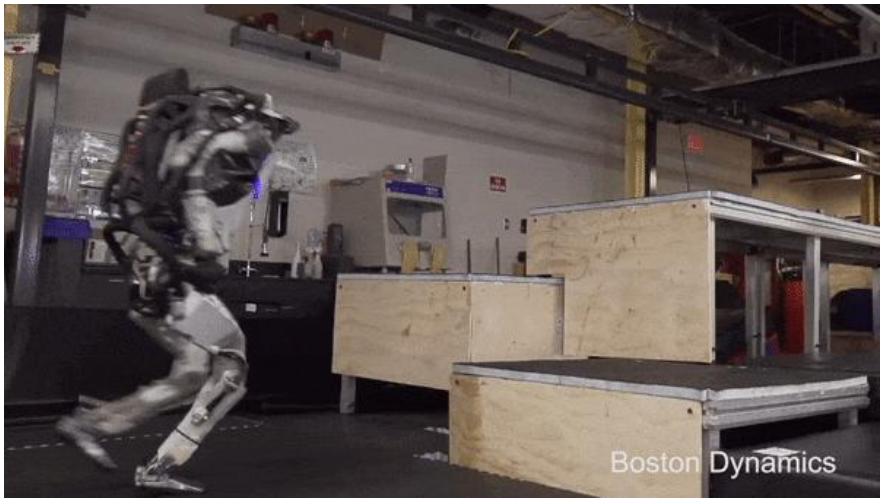
- Actuation of Robot Mechanism
- Design Principle of Power Joints
- Design of Stepper Motor
- Design of Brushed DC Motor
- Design of Brushless DC Motor





Outline of Module 1

- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

Lecture 5

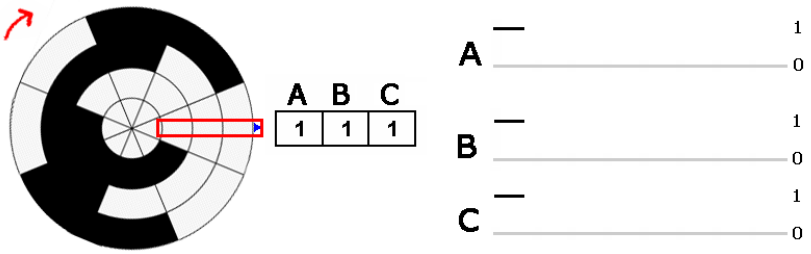
Robot Sensors

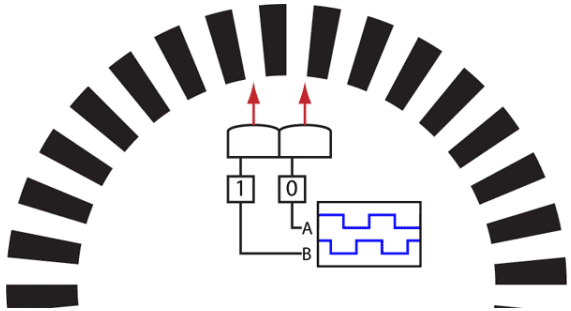
Xie Ming, PhD (France)

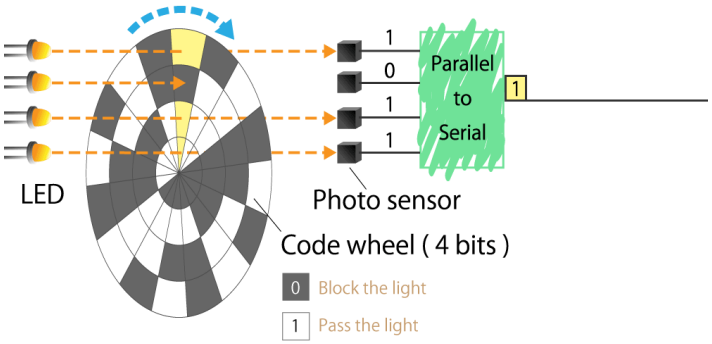
http://personal.ntu.edu.sg/mmxie

Outline of Lecture 5

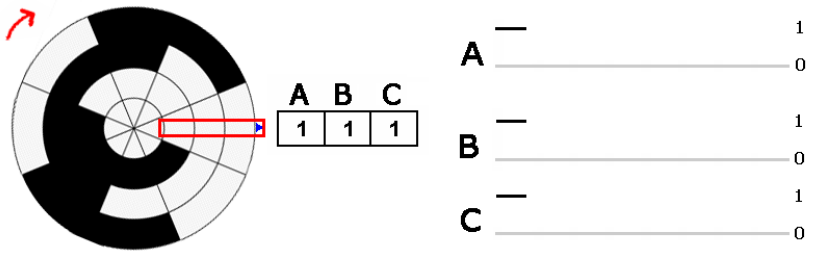
- Sensory-Motor Interaction
- Design of Position Sensor
- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor



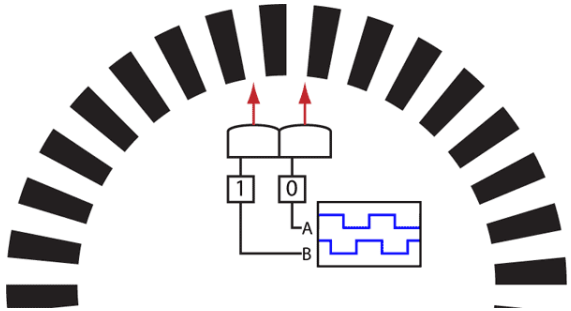


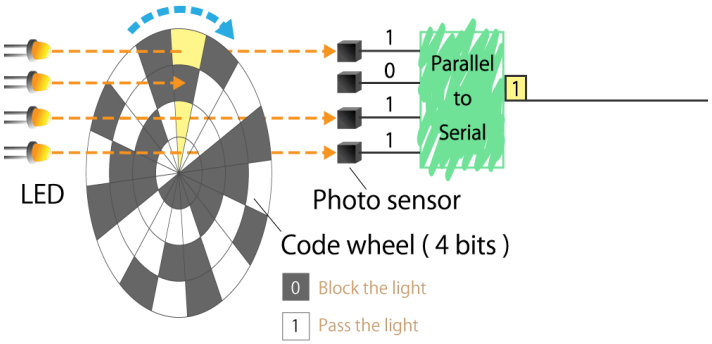


Outline of Lecture 5



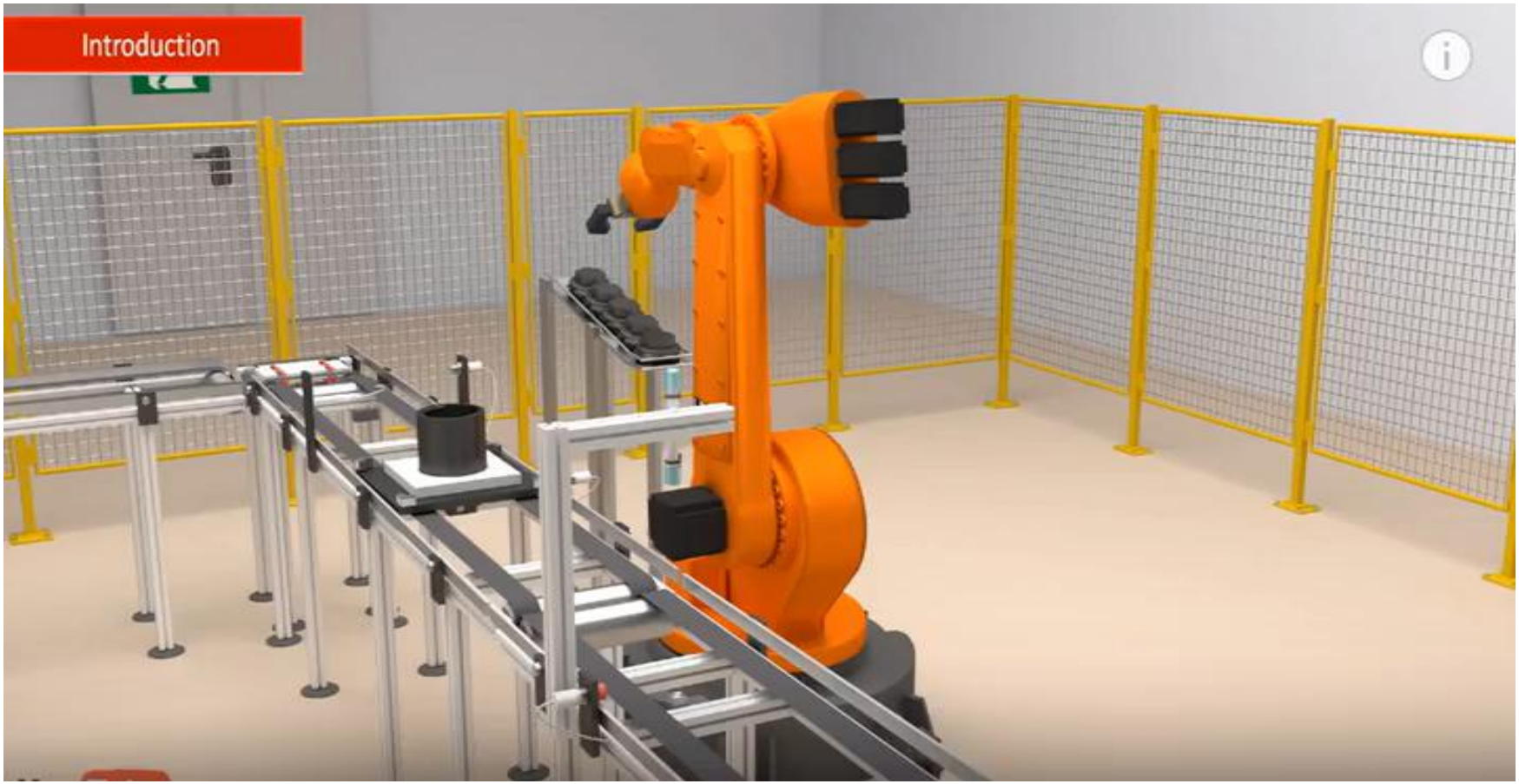
- Sensory-Motor Interaction
- Design of Position Sensor
- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor





How to precisely control a robot?

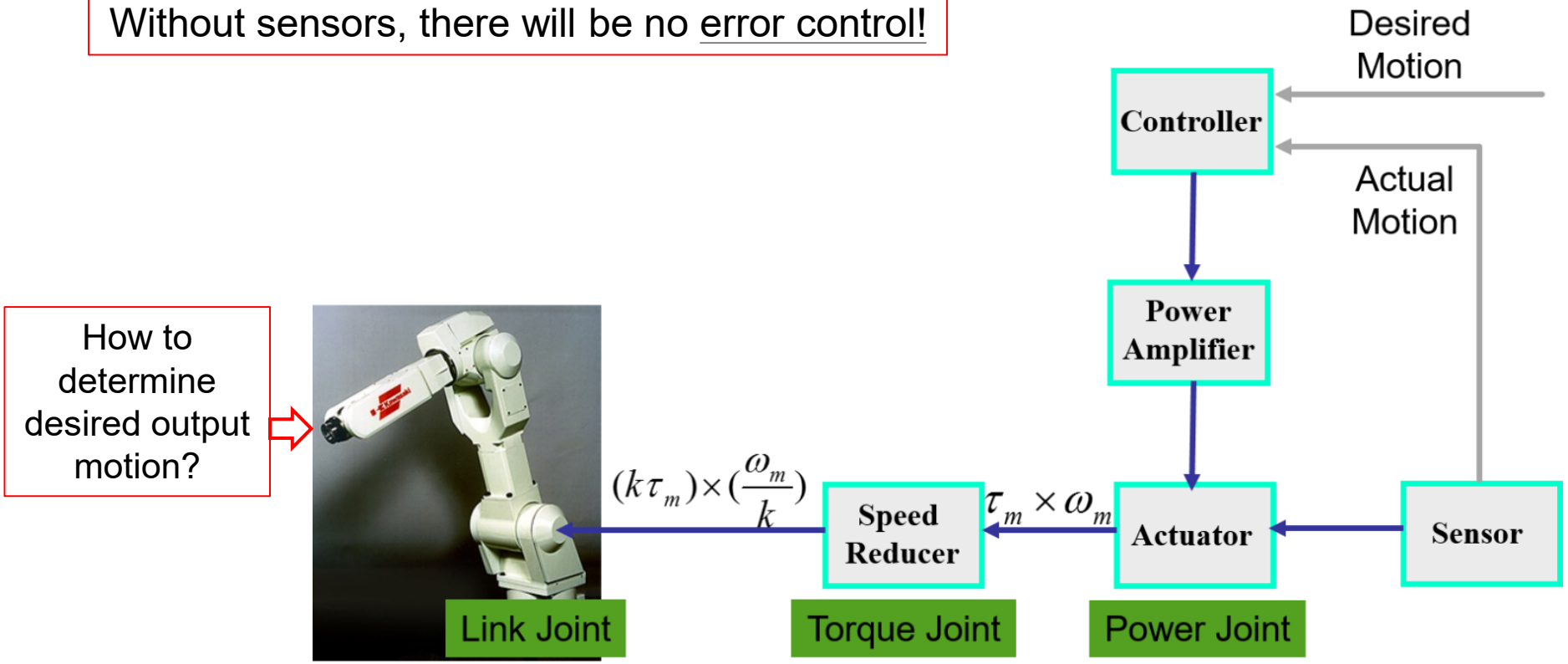
How to precisely control each joint?



Why are sensors so important?

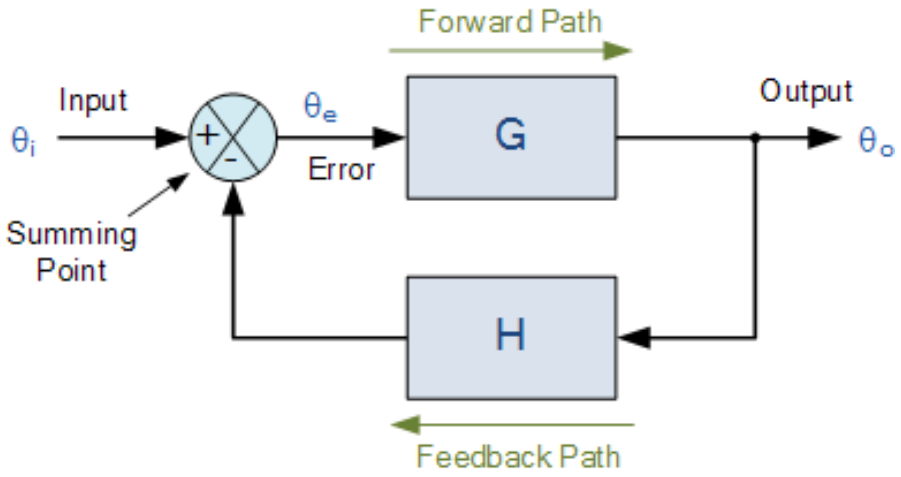
- Answer: Provide sensory feedback to controllers.

Without sensors, there will be no error control!



Why is error control so important?

- The best systems in the universe are static systems.
- Error control is the only way to make dynamic systems to be closer to static systems.
- Attention: Errors from sensors will 100% appear in the output of any system with error control.

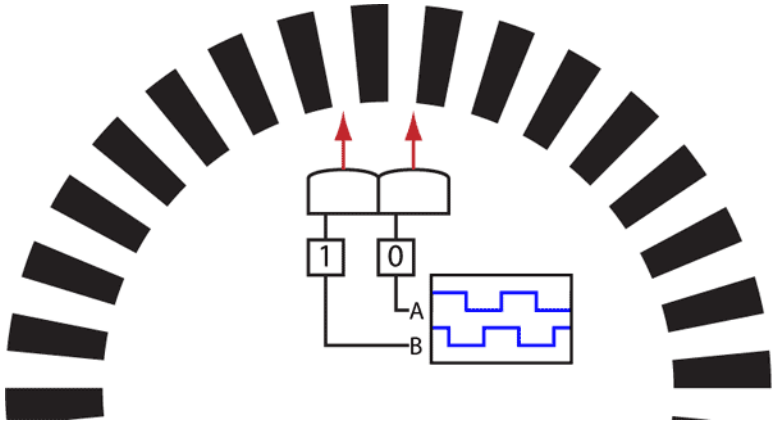


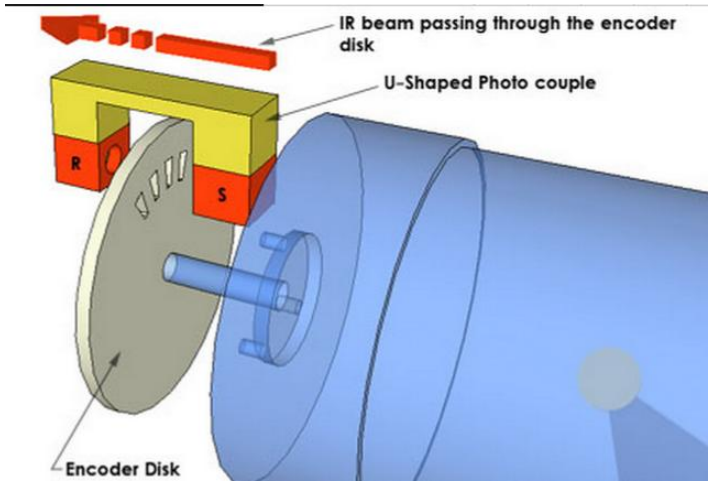
$$\theta_e = \theta_i - H \times \theta_o = 0$$

$$\theta_o = \frac{\theta_i}{H}$$ (without sensory error)

$$\theta_o = \frac{\theta_i}{H + \Delta H}$$ (with sensory error)

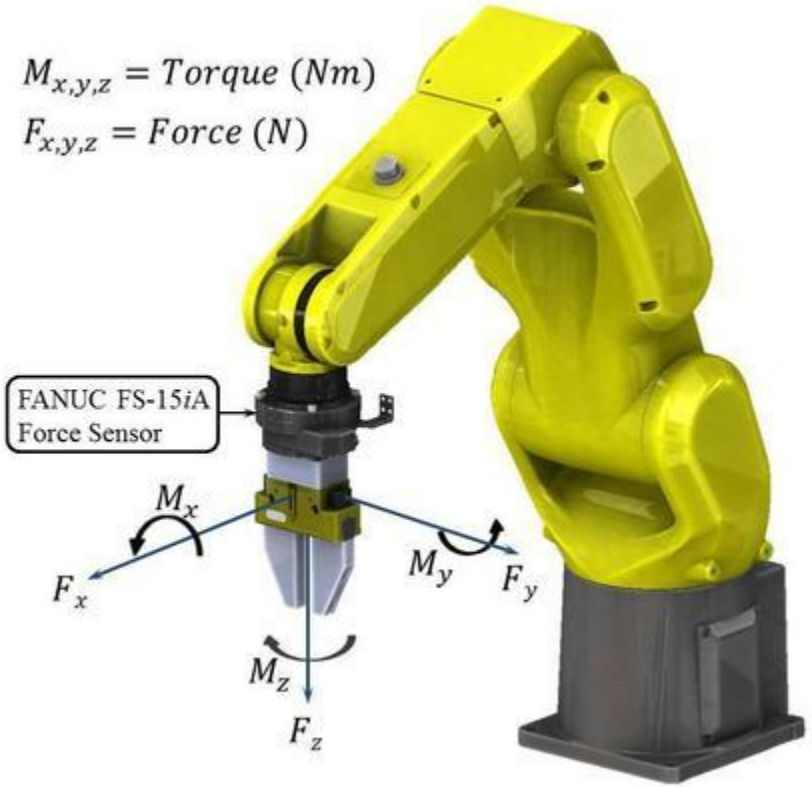
Sensors enable robots to perform controllable motions.



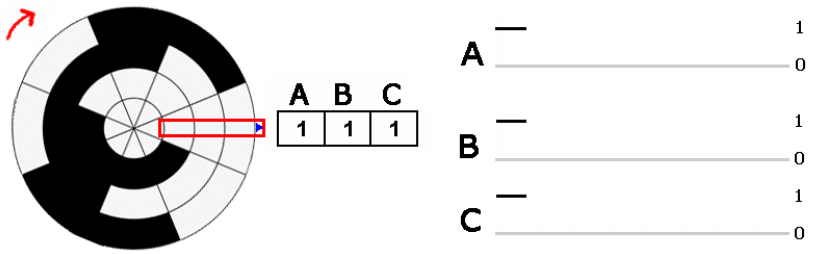


Sensors enable robots to perform constrained motions.

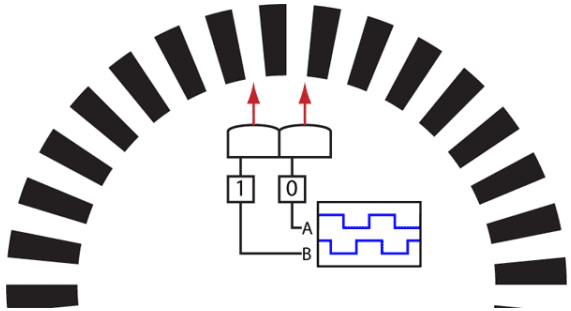
KUKA CAD Driven Milling

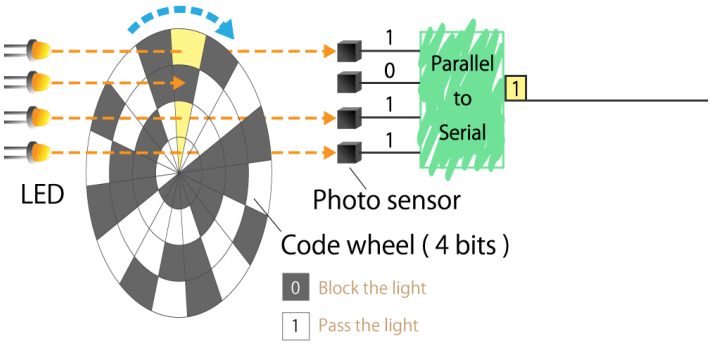


Outline of Lecture 5

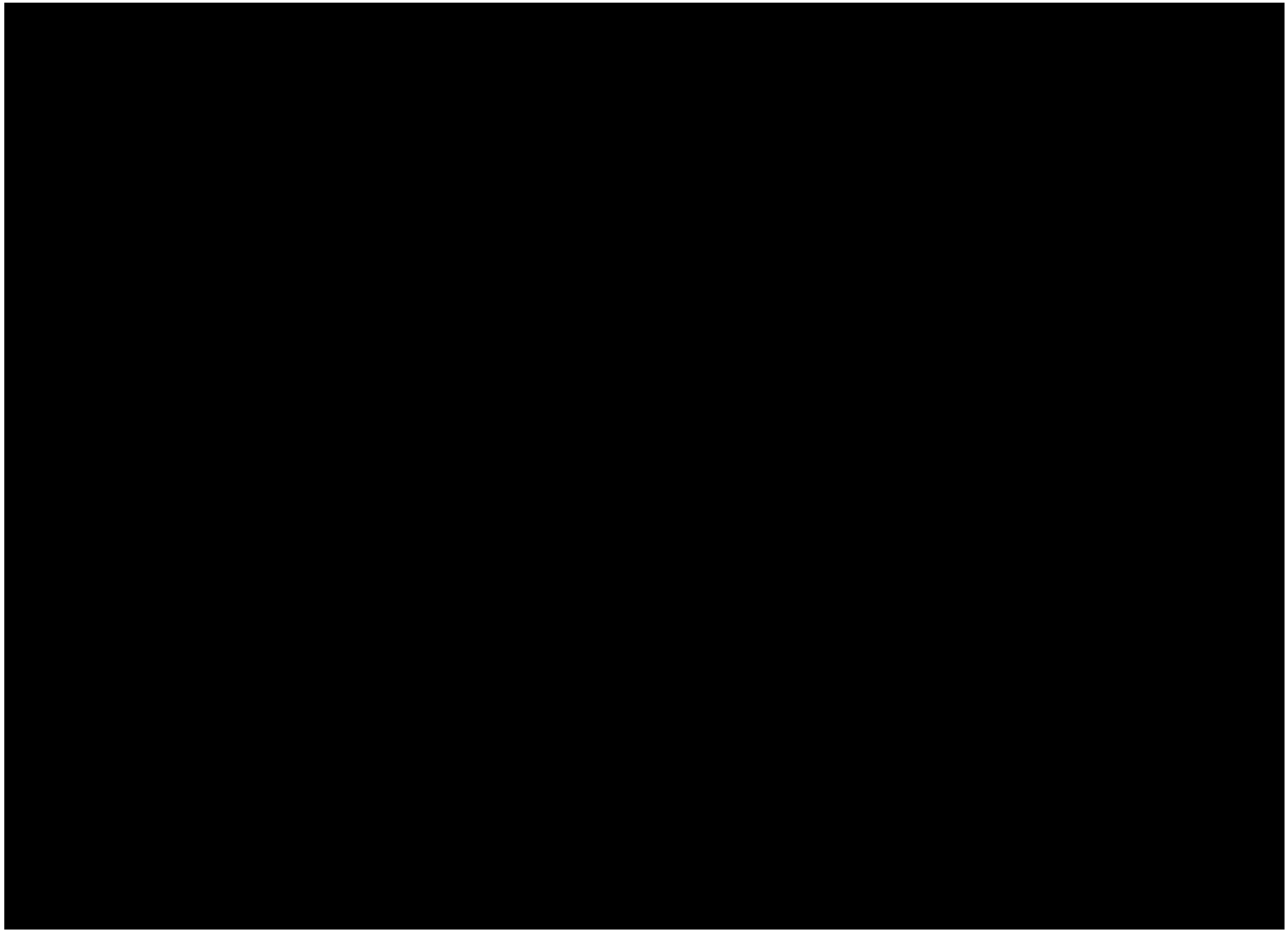


- Sensory-Motor Interaction
- Design of Position Sensor
- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor



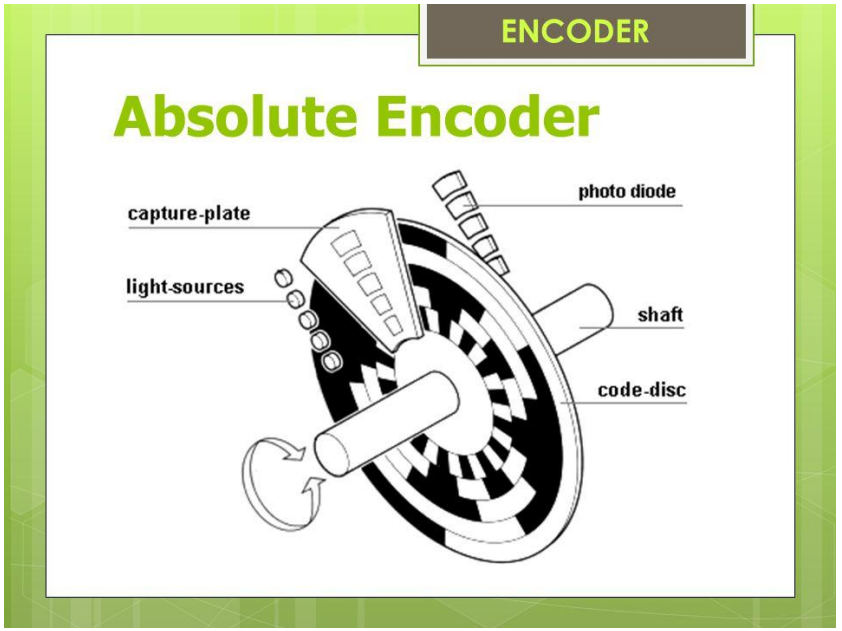


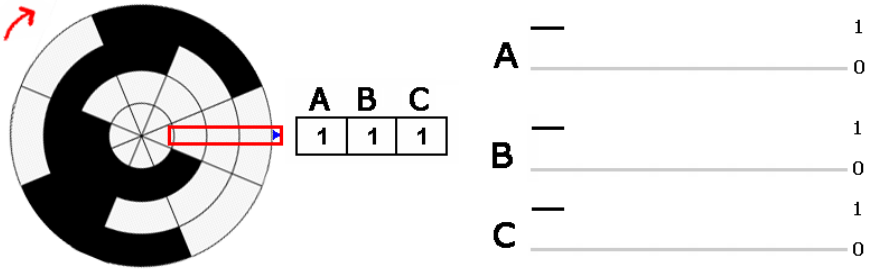
Video Showing Design and Working Principle



Summary of Design and Working Principle

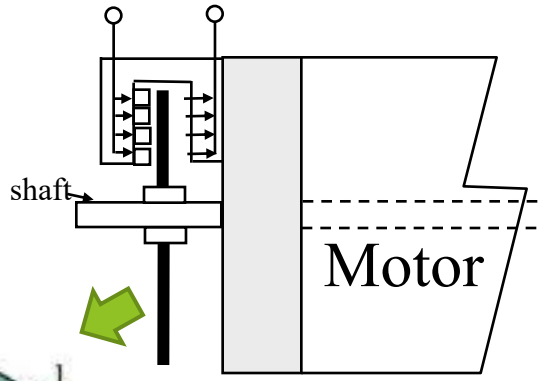
- A code disk has a set of slots or tracks.
- Each slot or track represents one bit.
- Each slot or track has a number of segments of holes (1) and solid zones (0).
- The combination of holes and solid zones in the radial direction is a unique pattern, which represents one angular or linear position.
- The pattern of holes and solid zones are read by a set of photo cells or photo diodes.



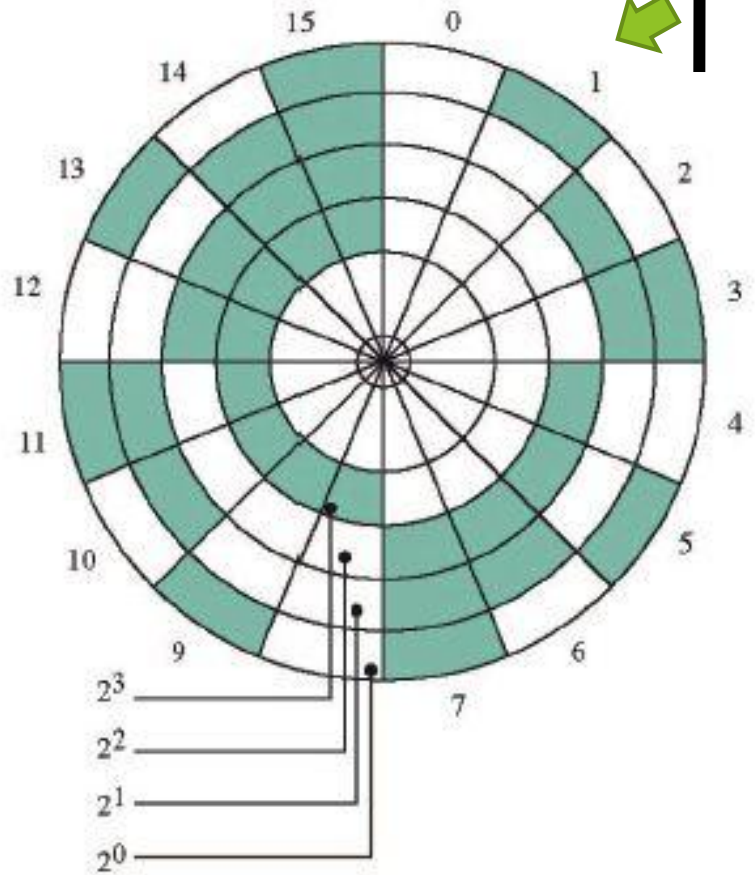


Design Example: 3-bit Sensor

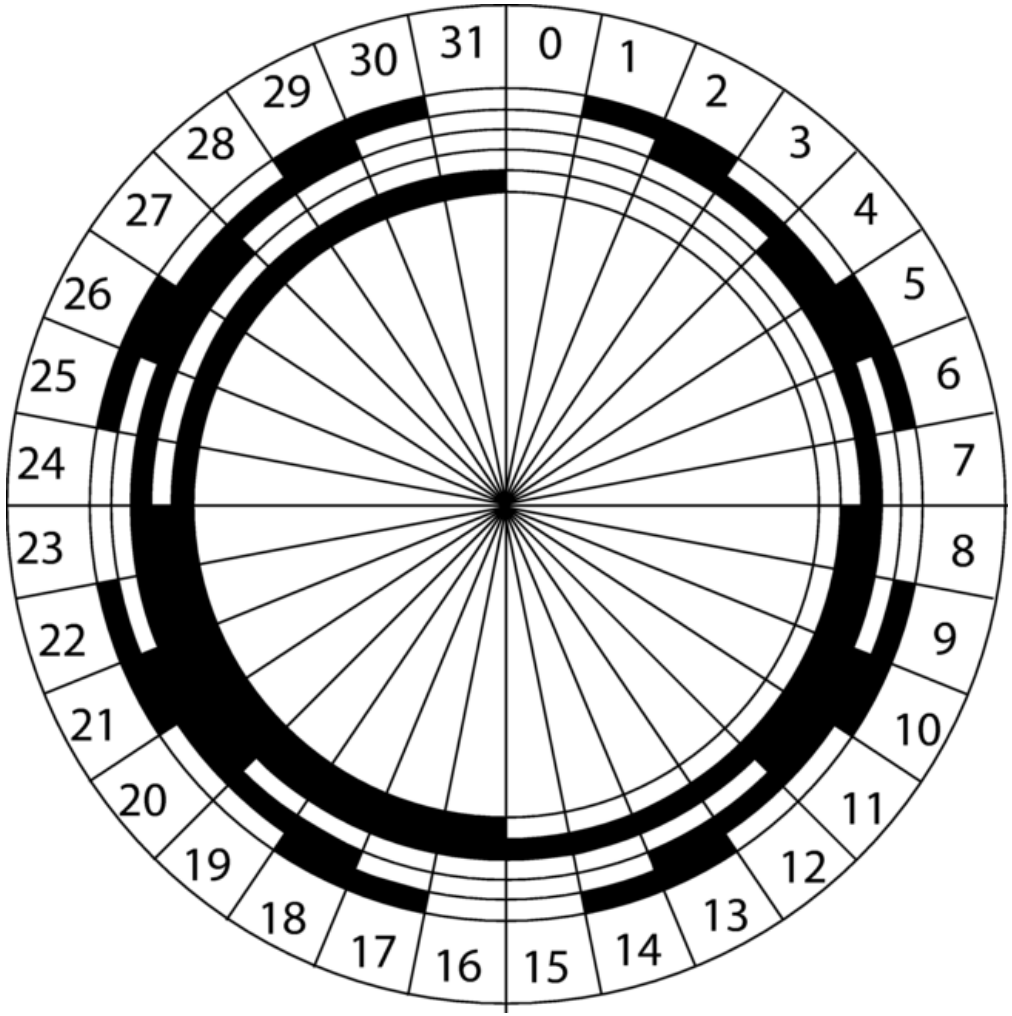
Design Example: 4-bit Sensor



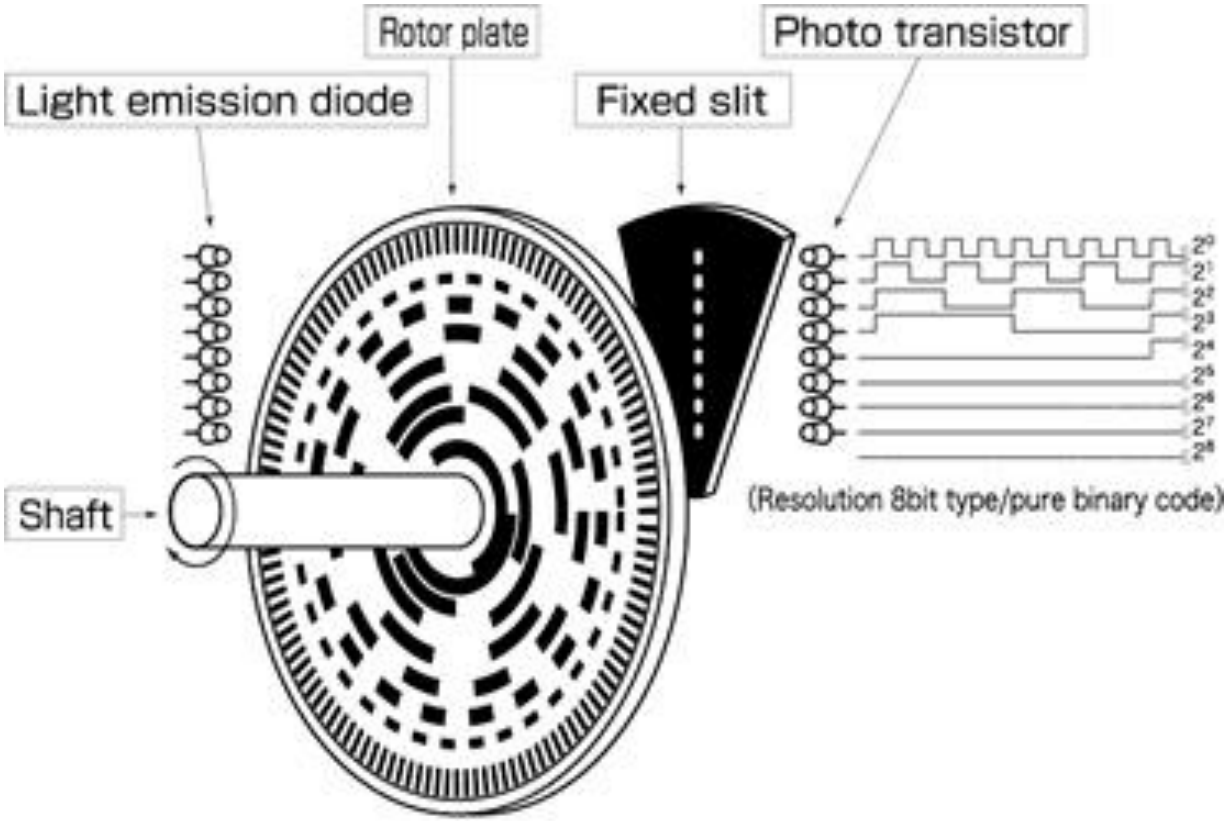
- Bit 0 is controlled by the outer slot or track.
- There are four slots or tracks.
- Each slot or track has sixteen segments.
- Each segment is either a hole (1) or a solid zone (0).
- Lights can only pass through holes in order to reach the photo cells which produce pulses as readings.



Design Example: 5-bit Sensor



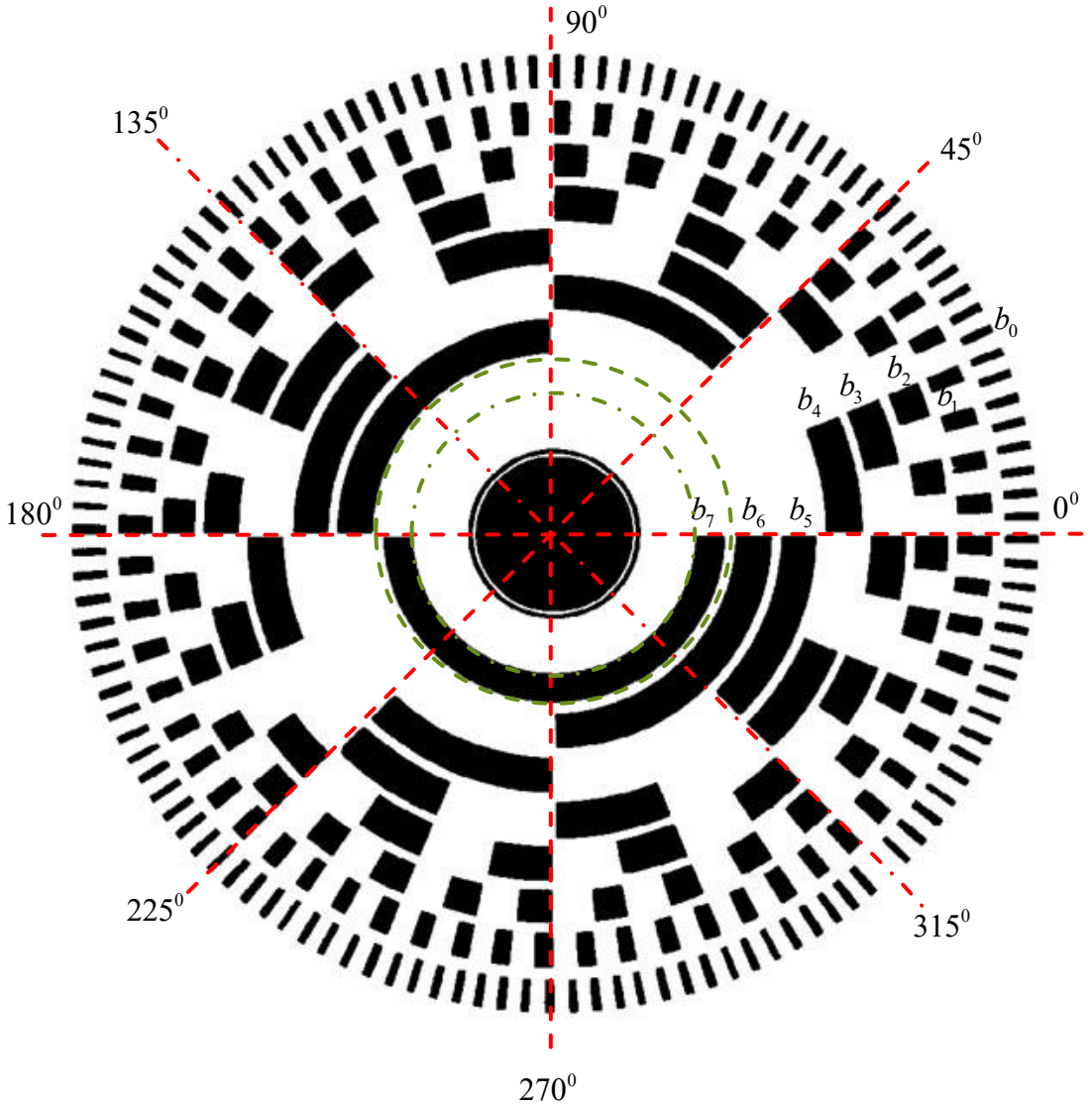
Design Example: 8-bit Sensor



Absolute Encoder Simplified Structure

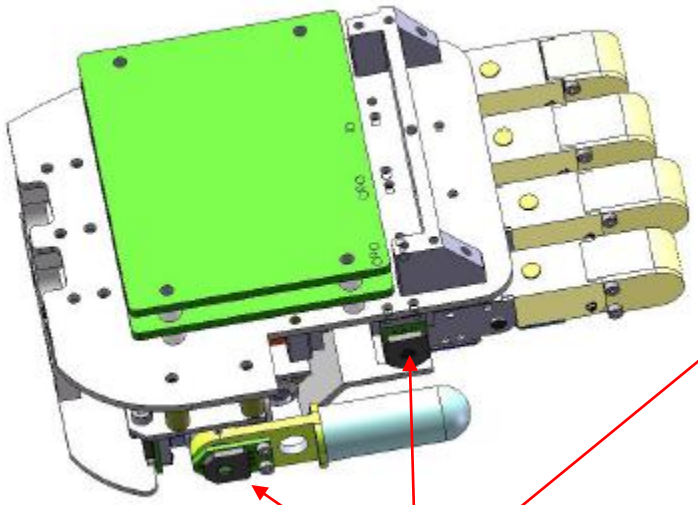
Exercise

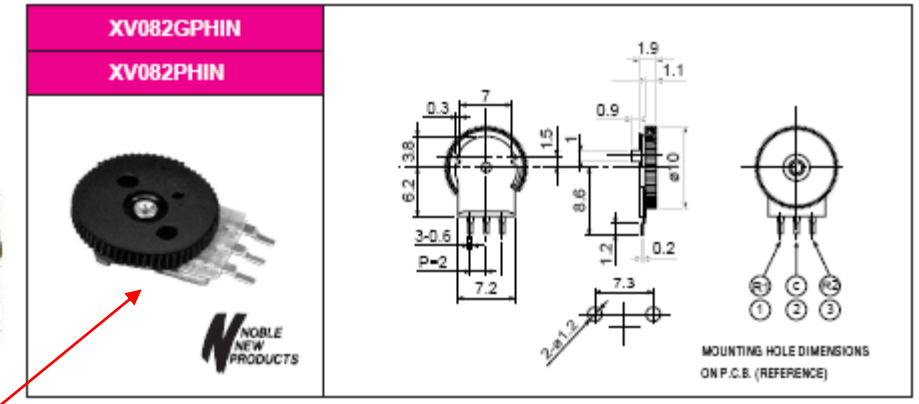
- An absolute position sensor has 8 output bits to represent the discrete positions.
- What is the total number of measurable positions?
- Answer: $2^8 = 256$
- The reading of zero position is 0000 0000.
- The reading of 45 degrees is 0010 0000 (=32=45*256/360).
- The reading of 135 degrees is 0110 0000 (=96=135*256/360).



More Design Example of Position Sensor

Other Type of Position Sensor

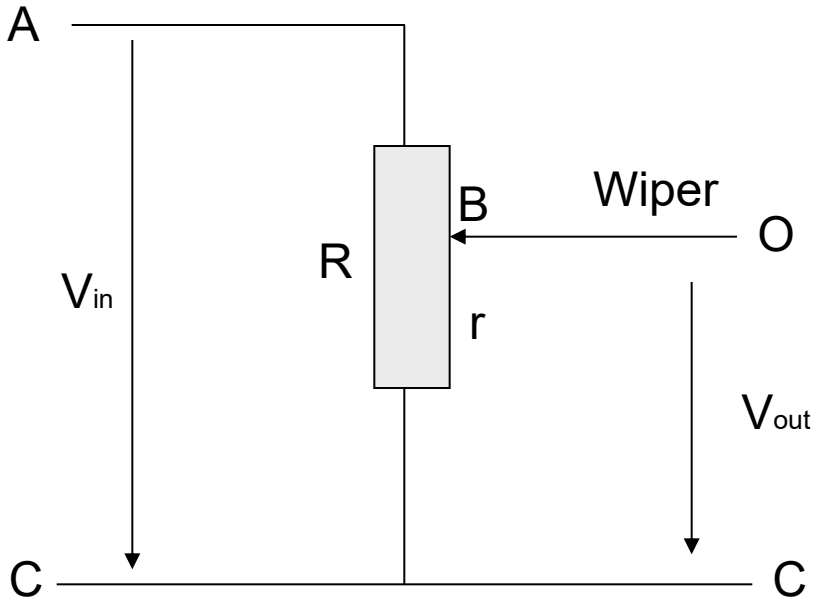




Potentiometer

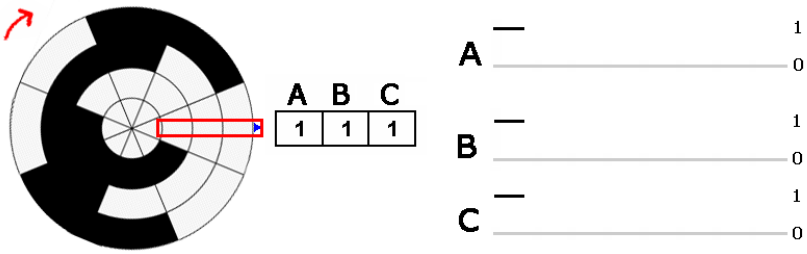
$$V_{out} = \frac{V_{in}}{R} \bullet r$$

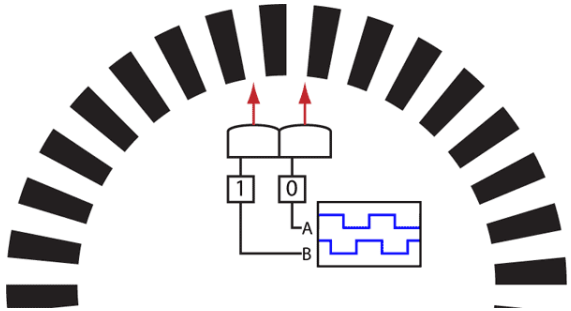
$$= \frac{V_{in}}{2\pi L} \bullet (\theta L) = \frac{V_{in}}{2\pi} \bullet \theta$$

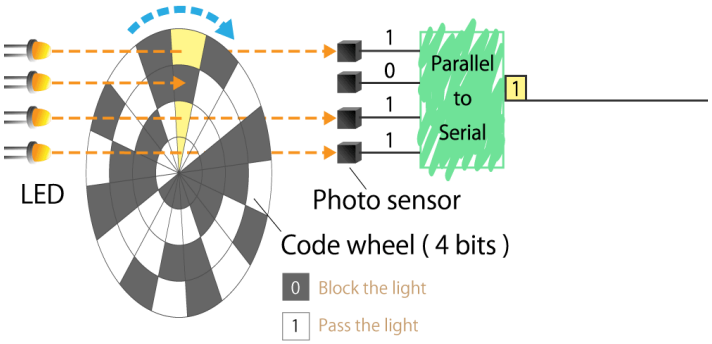


Outline of Lecture 5

- Sensory-Motor Interaction
- Design of Position Sensor
- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor



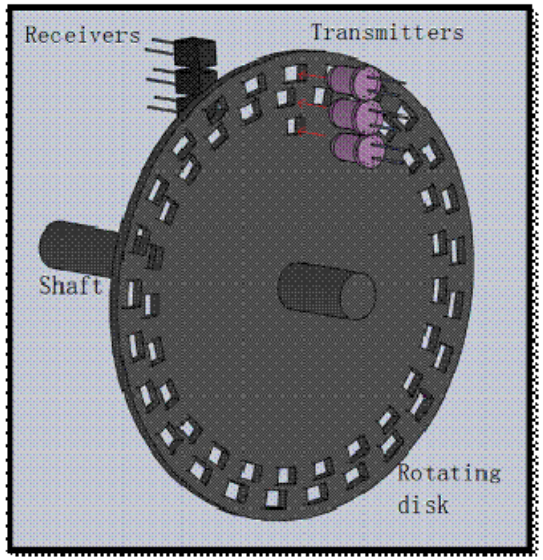


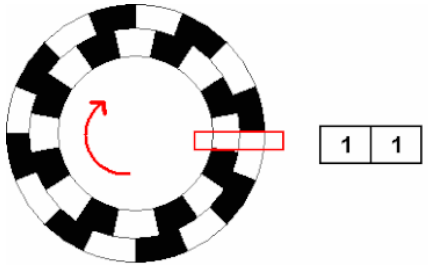


Video Showing Design and Working Principle

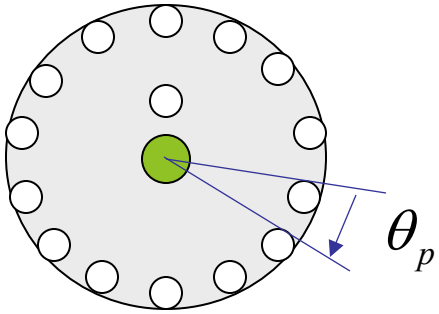
Summary of Design and Working Principle

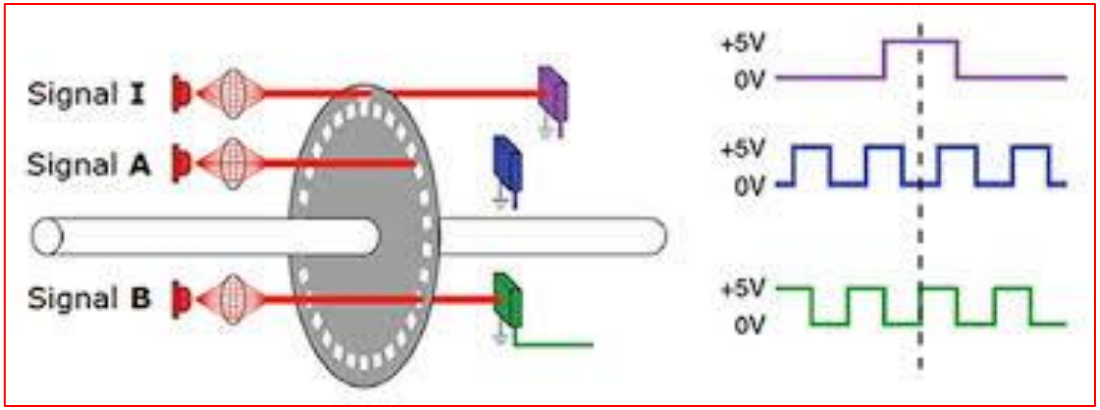
- A code disk has a ring of holes.
- A pair of light and photo cell is to convert a displacement into a pulse of logic 1 and 0.
- The pitch angle between two adjacent holes is equal to an angular displacement.
- The counting of the pulses within a time interval allows to determine the speed.





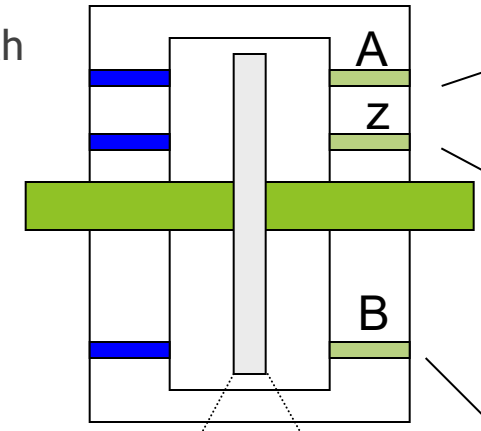
$$\overline{\omega} = \frac{C \times \theta_p}{\Delta t}$$

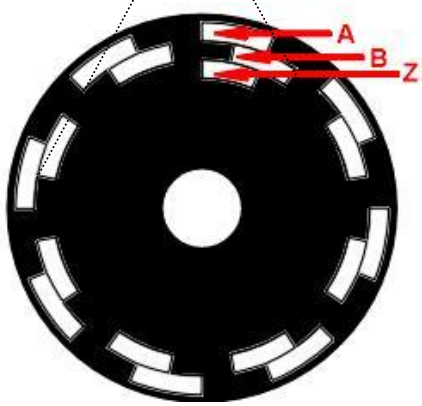


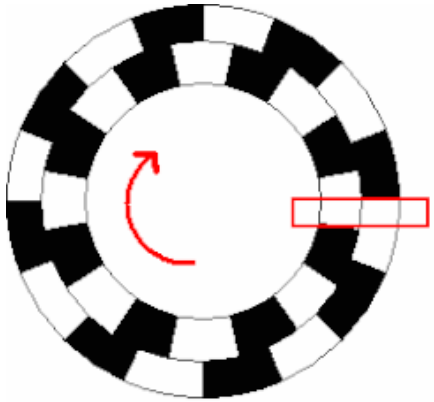


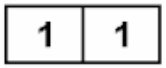
Design Requirements

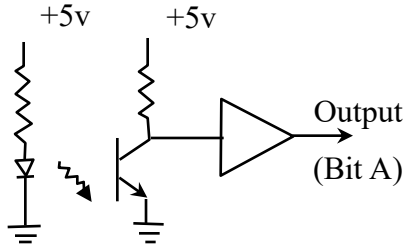
- Measurement of velocity
- Measurement of direction of motion
- Measurement of position with respect to a reference (i.e. zero position)

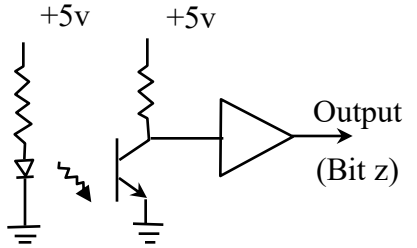


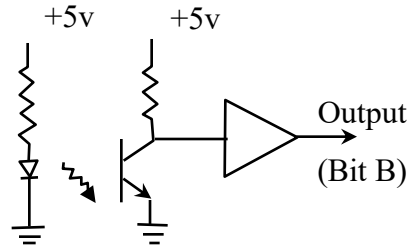




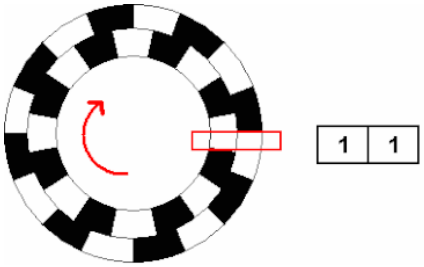


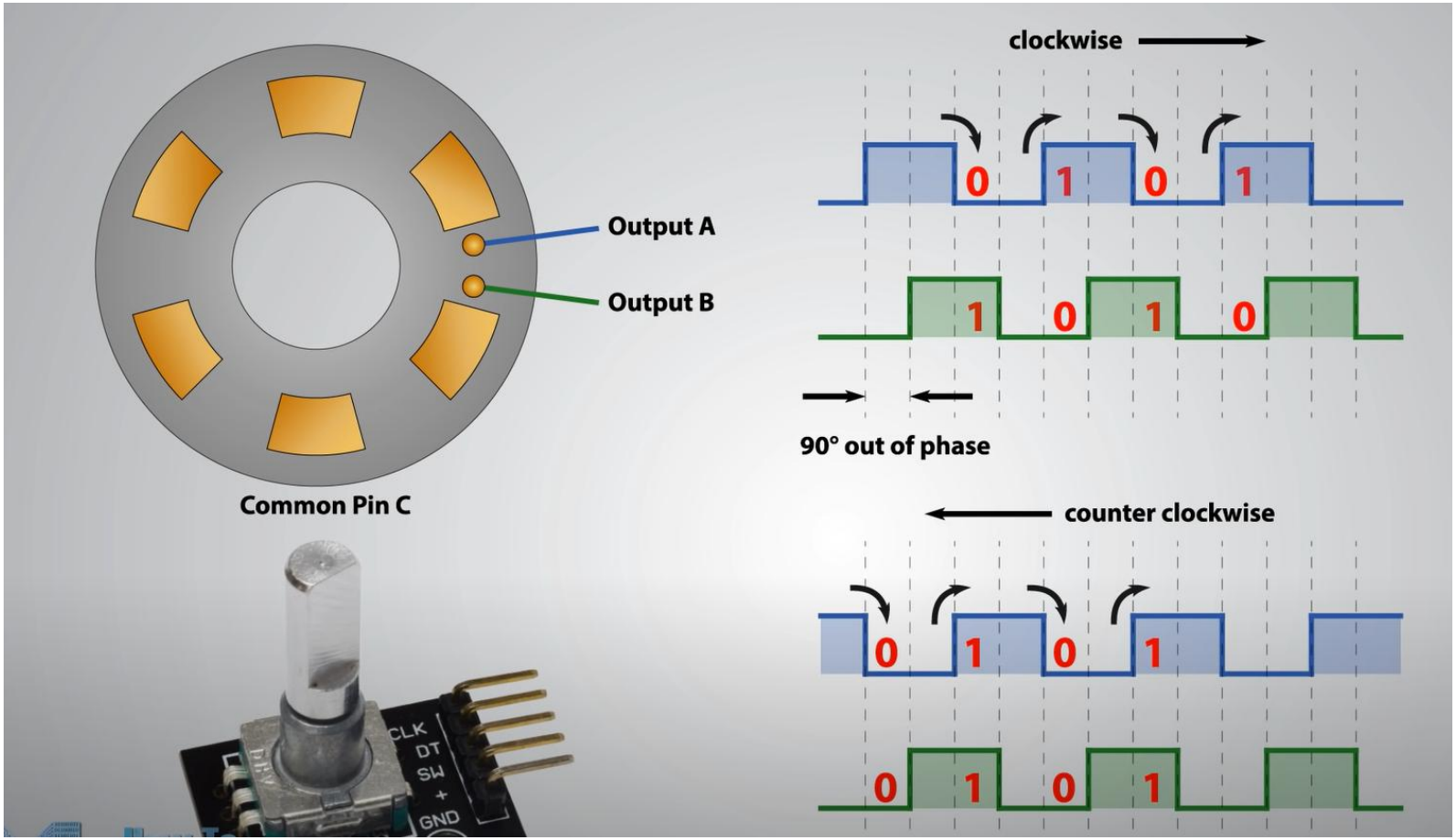




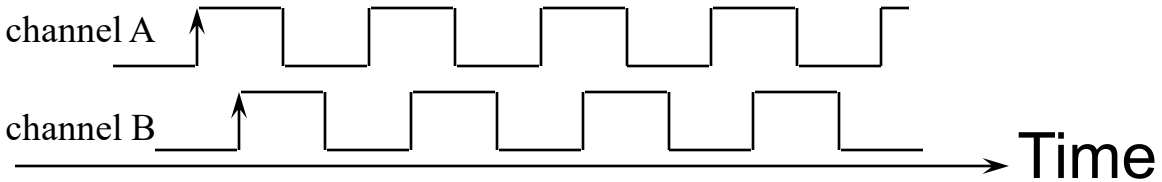


One more illustration

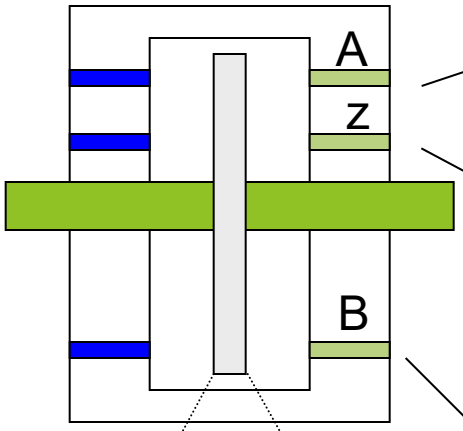


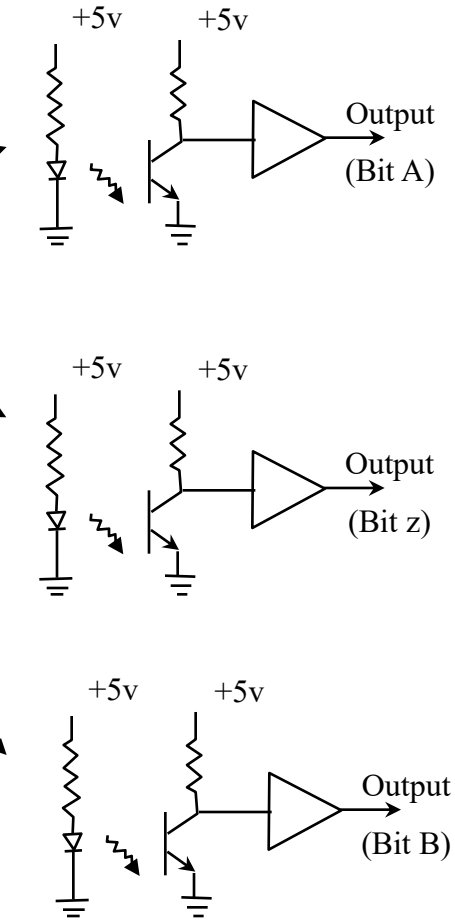


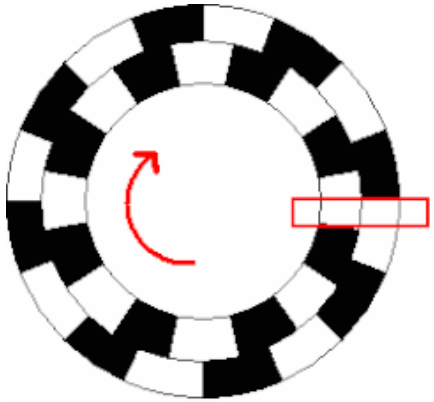
Detecting Counterclockwise Rotation (CCW)



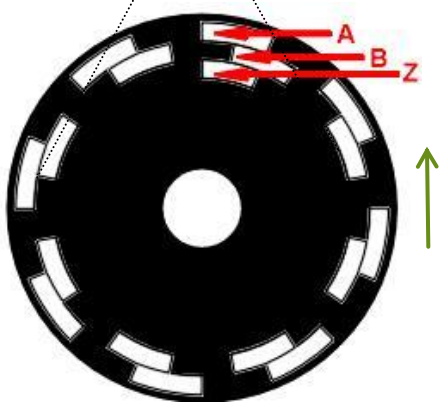
Active signal of A leads active signal of B



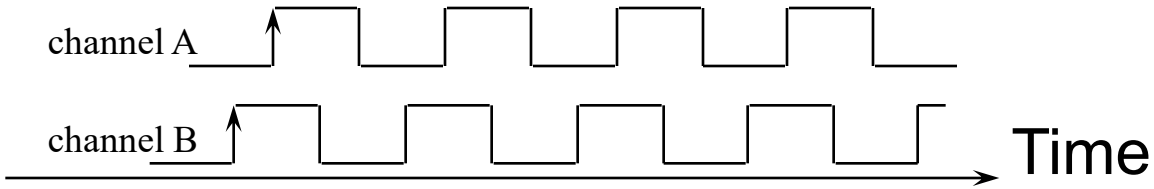




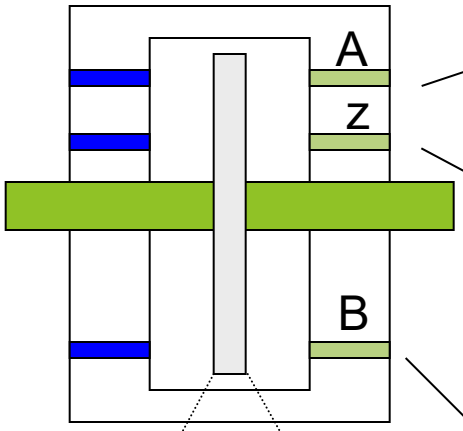
1	1

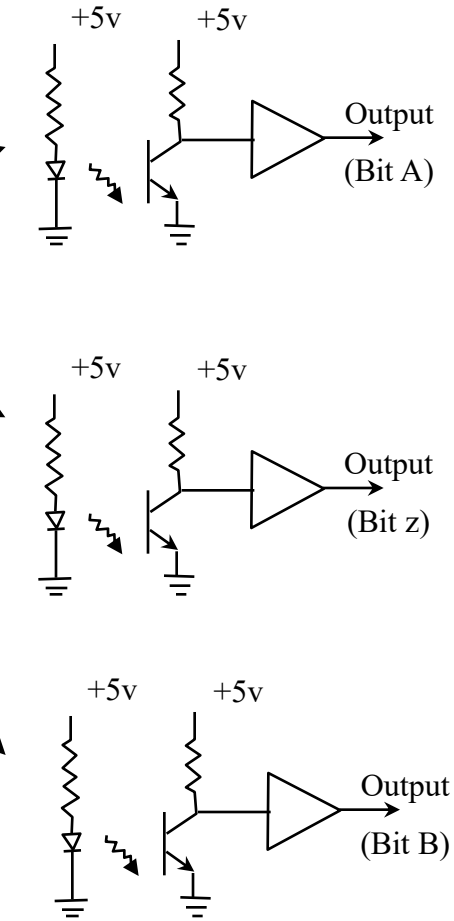


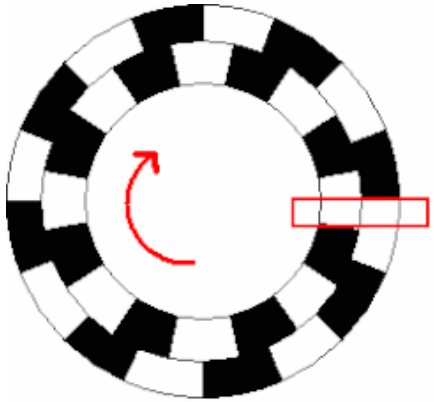
Detecting Clockwise Rotation (CW)



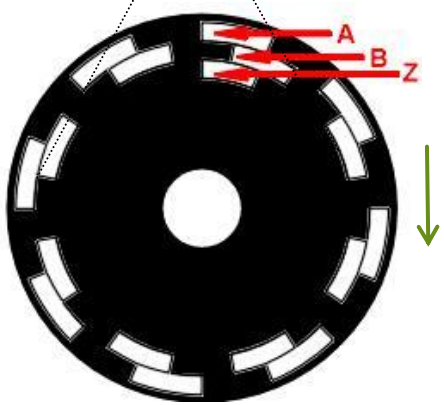
Active signal of A lags active signal of B



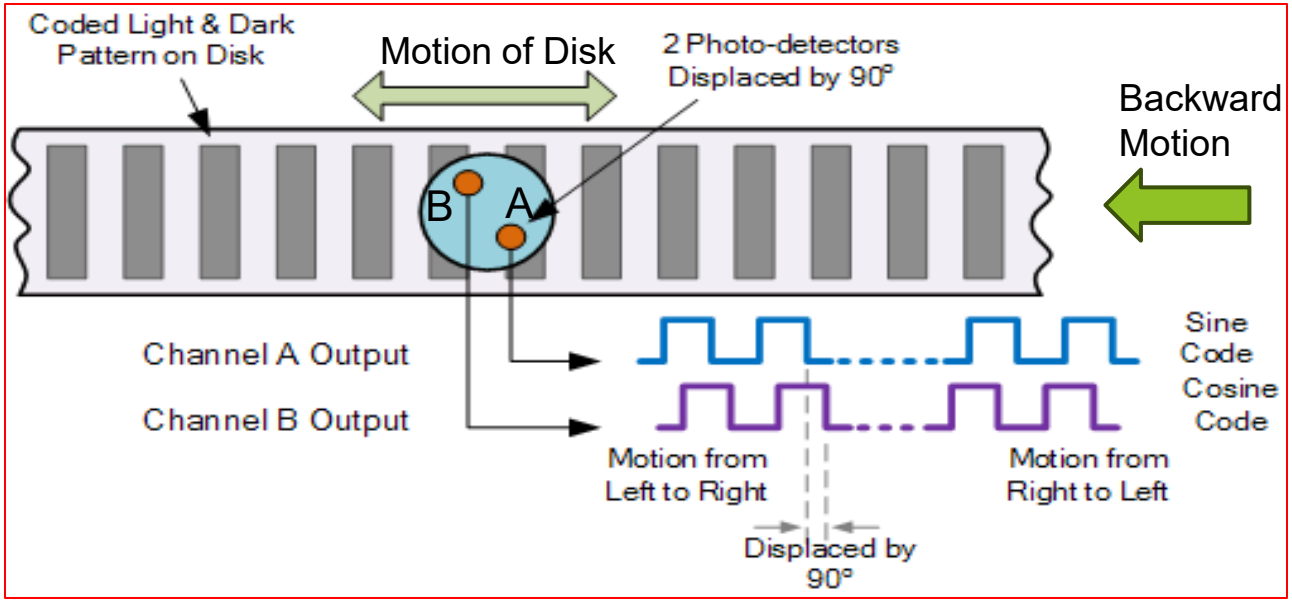


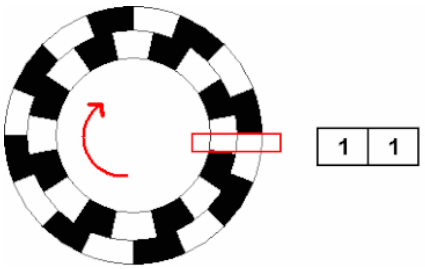


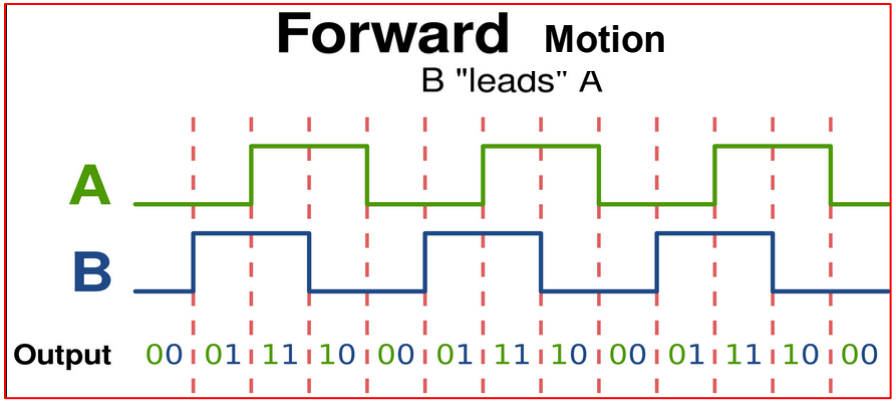
1	1

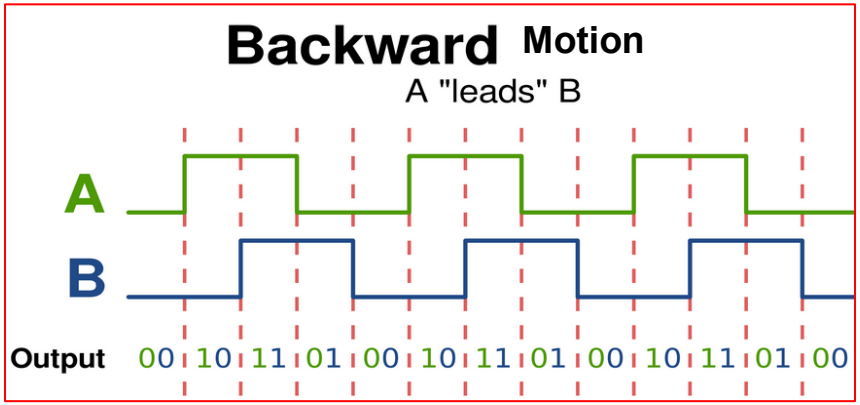


Detecting Linear Motion's Directions



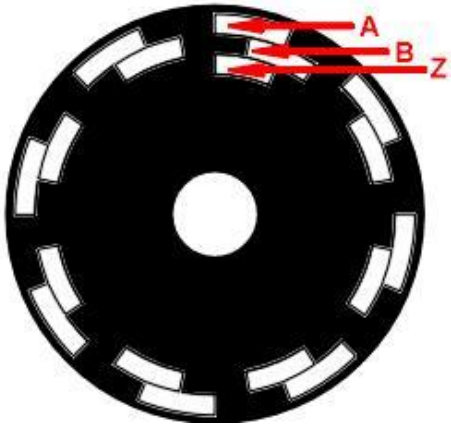




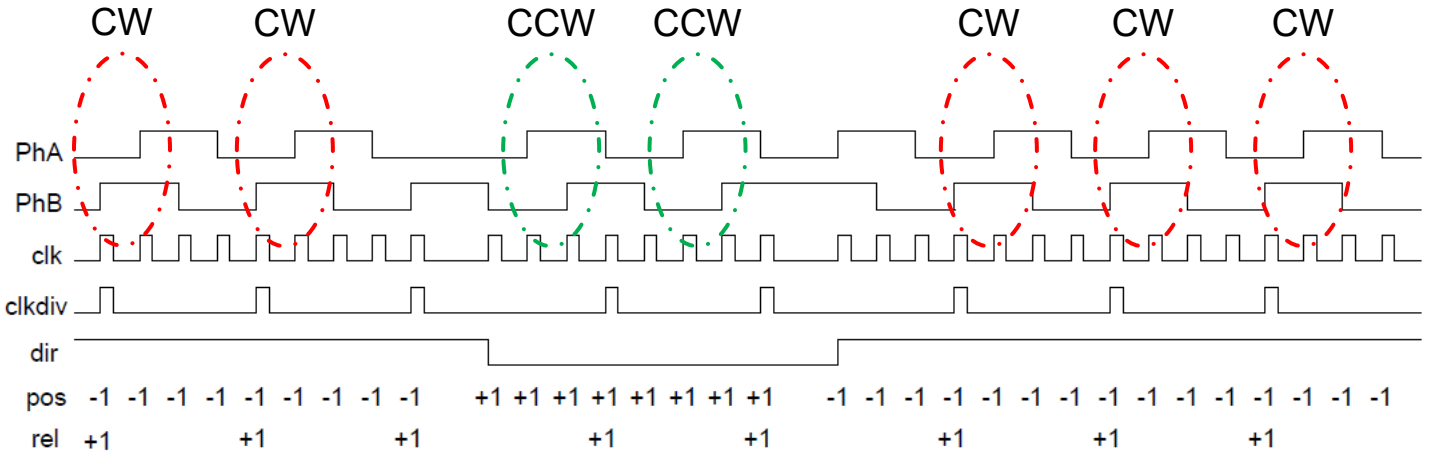


Time

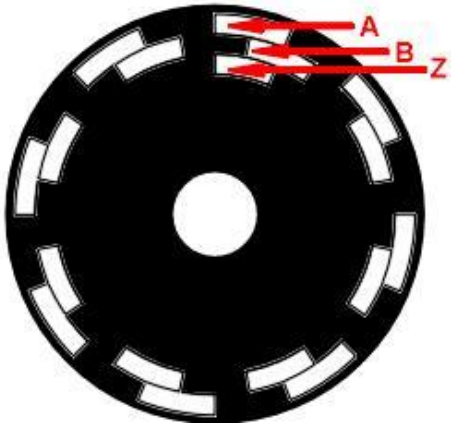
Rule 1: Direction of Motion



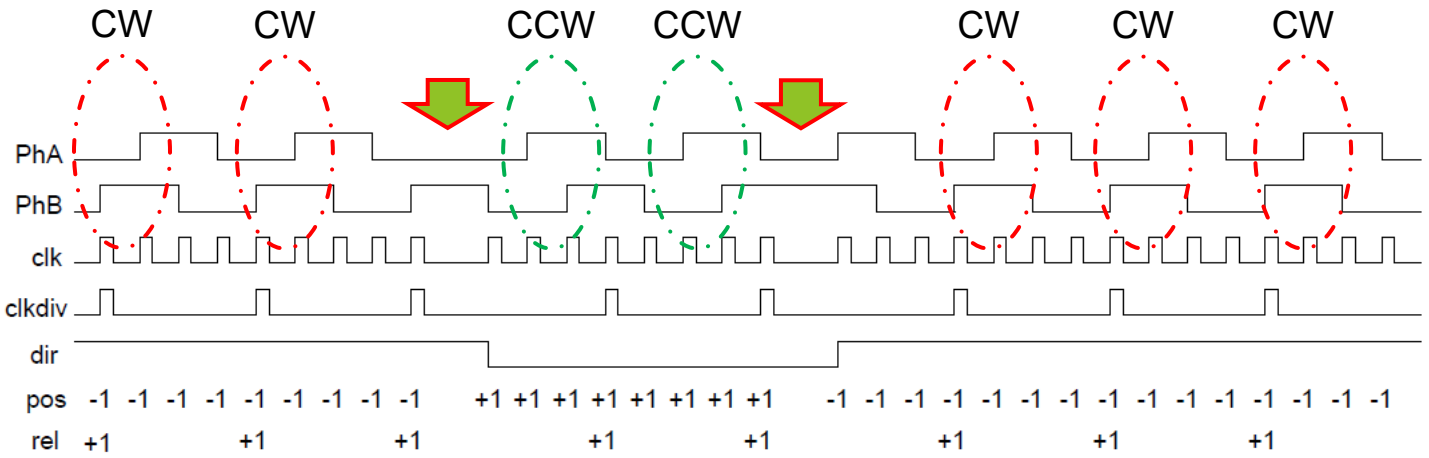
- If pulse B leads pulse A by 90 degrees, the direction is clockwise.
- If pulse A leads pulse B by 90 degrees, the direction is counter-clockwise.



Rule 2: Change of Direction of Motion



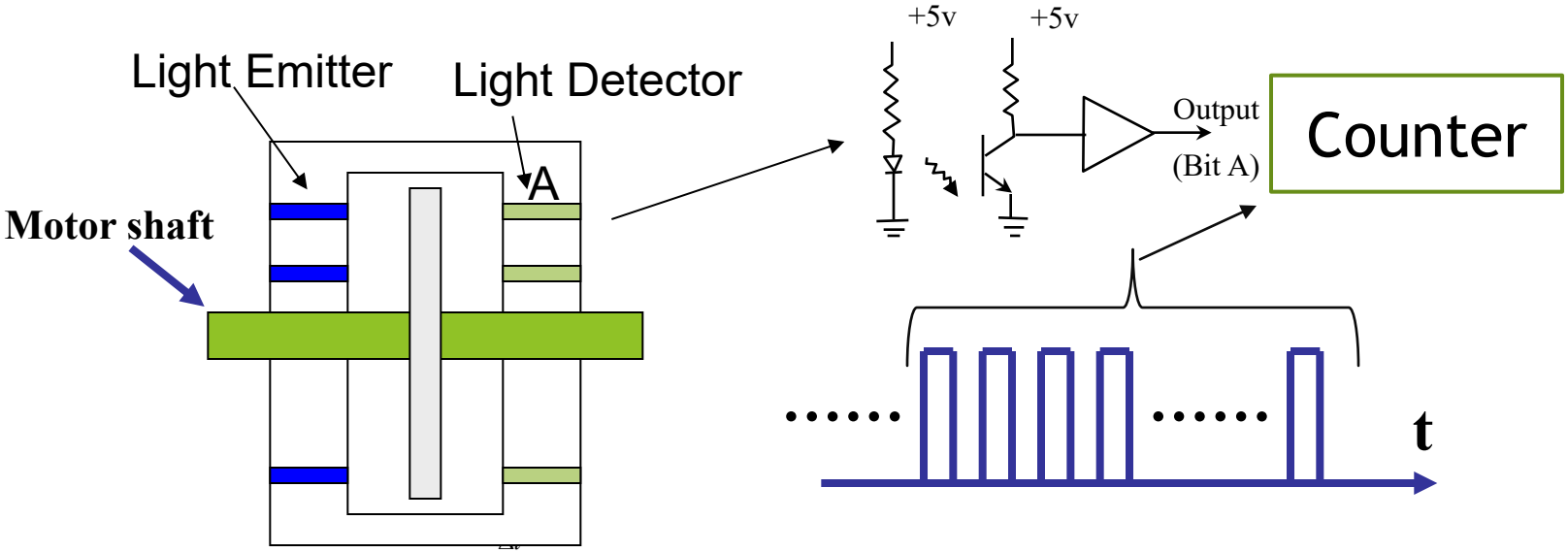
- When pulse B makes two transitions while pulse A remains unchanged, CW is changed to CCW.
- When pulse A makes two transitions while pulse B remains unchanged, CCW is changed to CW.



Exercise

- A velocity sensor can produce 200 pulses when the code disk of the sensor makes a full rotation.
- What is the angular displacement corresponding to one pulse?
- Answer:

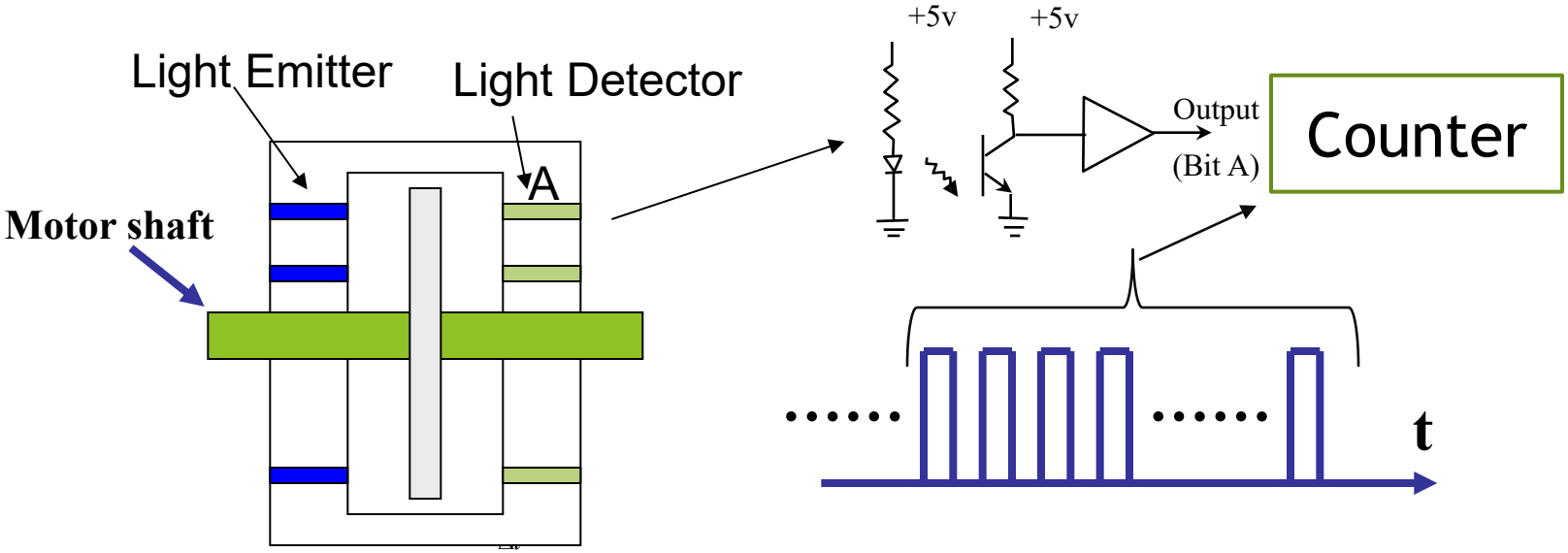
$$\theta_p = \frac{360^0}{200} = 1.8^0$$



Exercise

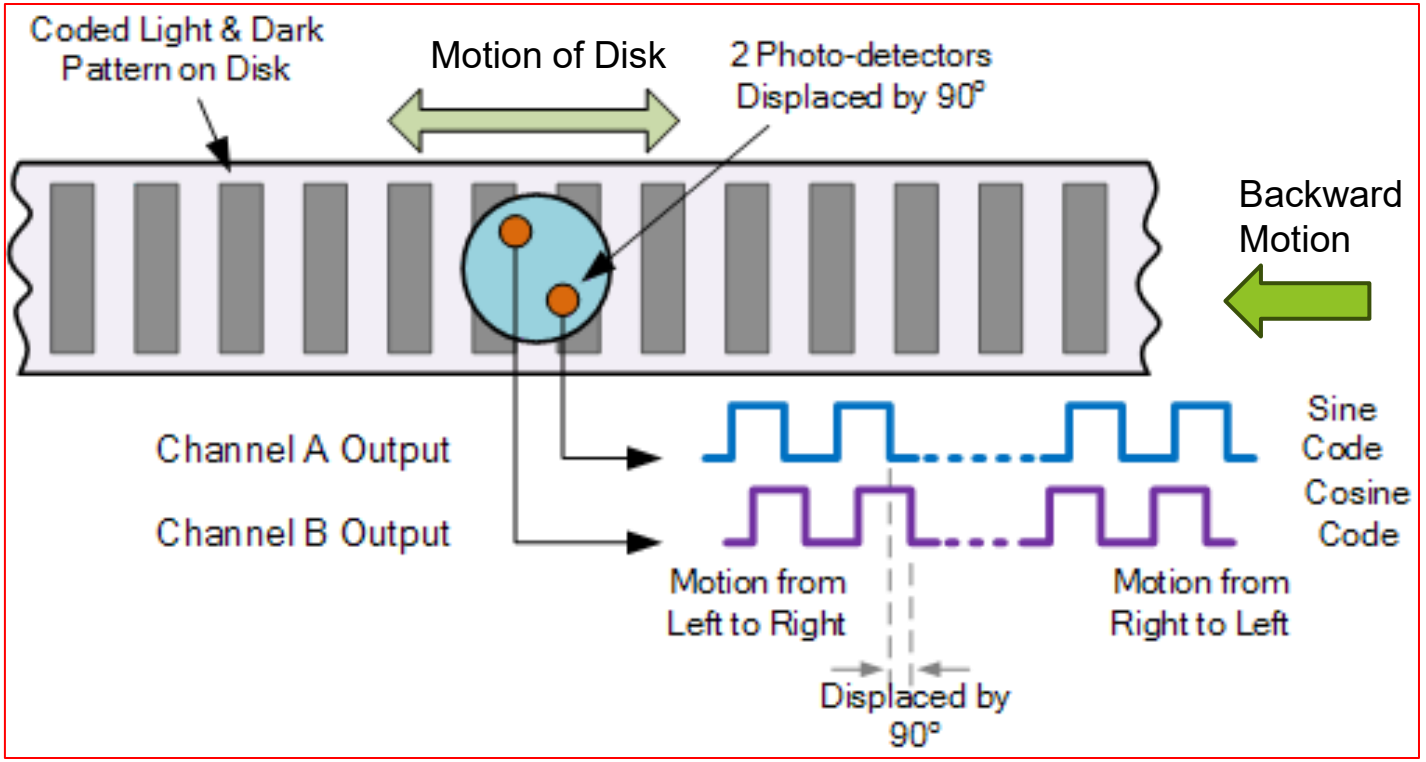
- A velocity sensor's angular displacement is 1.8 degrees. It outputs 500 pulses within 10.0 milliseconds.
- What is the average angular velocity of the motor's shaft?
- Answer:

$$\overline{\omega} = \frac{C \times \theta_p}{\Delta t} = \frac{500 \times 1.8^0}{0.01} \times \frac{\pi}{180^0} = 1570.0 \text{ rad/s}$$



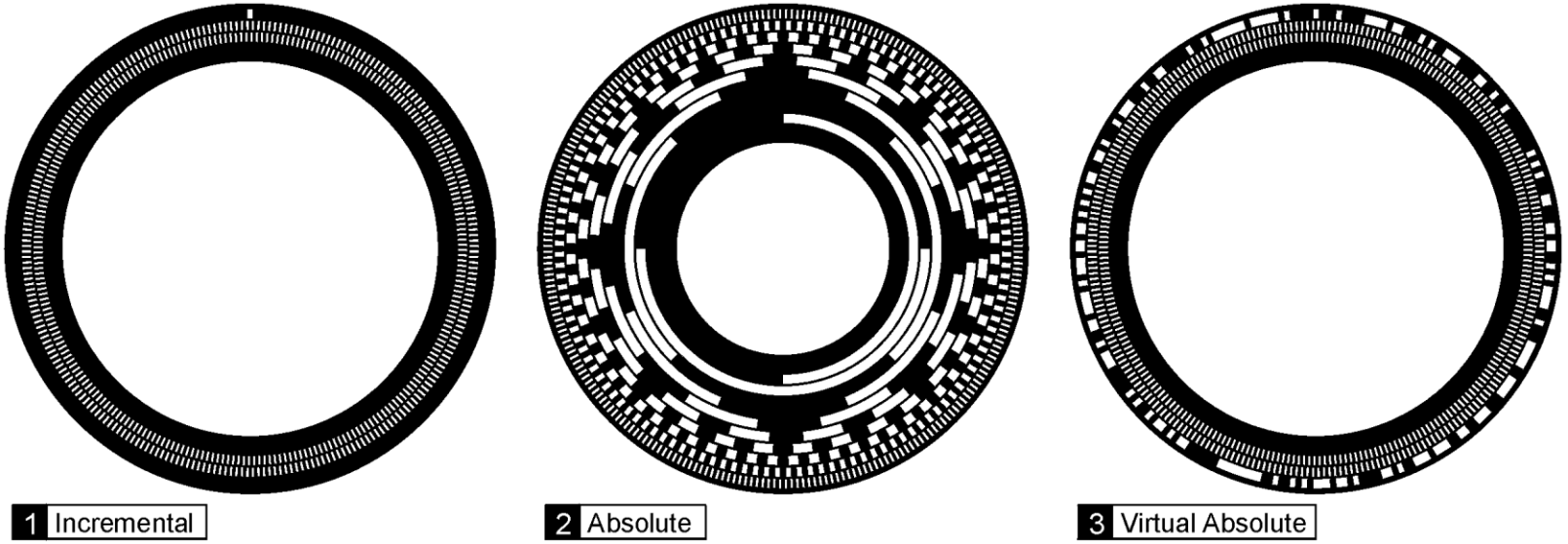
Exercise

- In a linear incremental encoder, the distance between two adjacent empty grids is 0.1mm. If the counted number of pulses is 7000counts/s, what is the linear speed?
- Answer: $v = 0.1 \times 10^{-3} \times 7000 = 0.7 m/s$



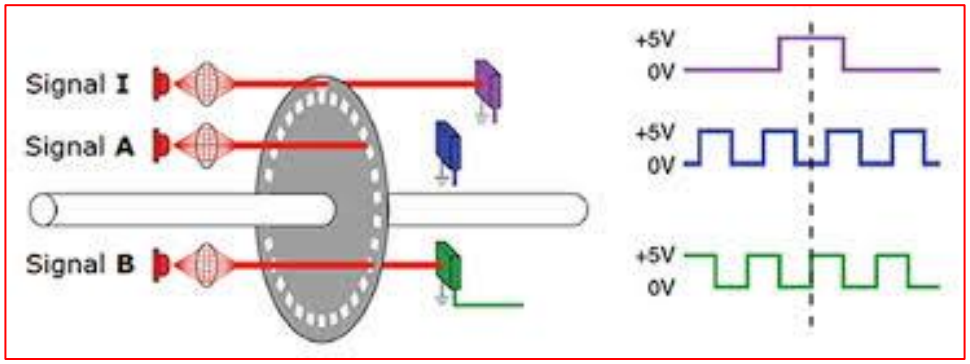
Discussion:

How to use incremental encoders to measure positions?

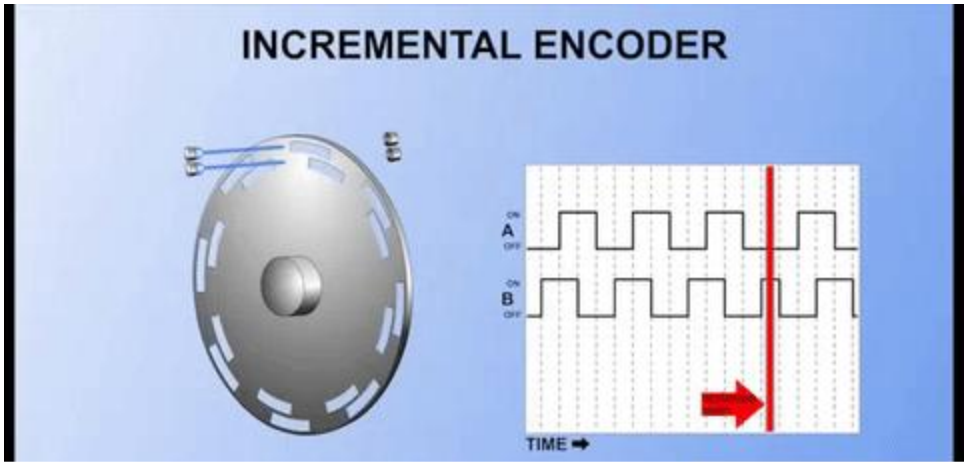


A Virtual Absolute encoder uses just cyclic and index tracks, like an incremental encoder. However, you can see the index track is something like a bar code, instead of just a single line (Figure 1). Absolute position is encoded serially along this one track, rather than being dispersed over multiple parallel tracks. You don't know position immediately upon start-up, as you do when using a conventional absolute device, but after a very short travel, in either direction and starting from anywhere, you know exactly where you are. In a rotary encoder, this initialization angle is typically 1–2º, depending on the encoder's line count (non-binary resolutions are easily accommodated); in a linear encoder, less than 1 mm motion is needed.

How to get the measurement of positions from velocity sensor?



- The solution is to include an additional track which is called I track or Z track. I stands for index of zero point.



Equations of Measuring Positions

- Equations for determining the angular position from the readings of velocity sensor.

$$\overline{\omega} = \frac{\theta - \theta_i}{\Delta t}$$

$$\overline{\omega} = \frac{C \times \theta_p}{\Delta t}$$

$$\frac{\theta - \theta_i}{\Delta t} = \frac{C \times \theta_p}{\Delta t}$$

$$\theta = \theta_i + C \times \theta_p$$

when I signal appears

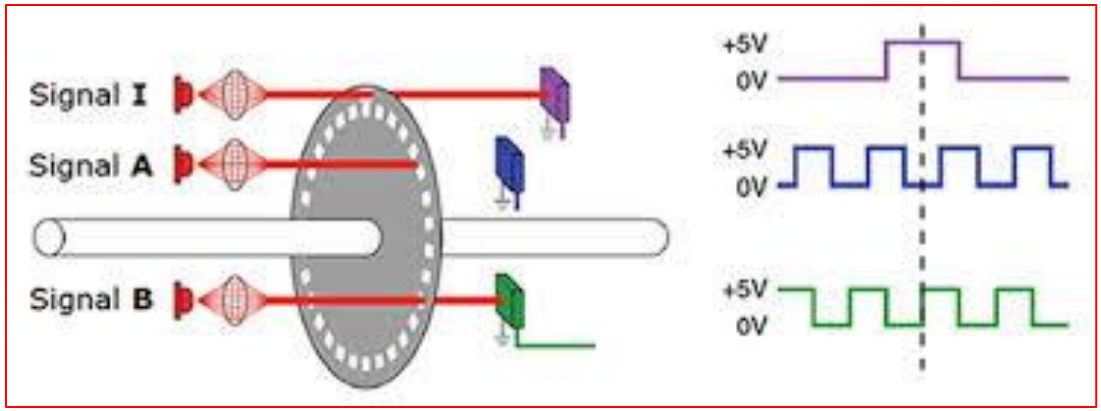
$$\theta_r = \theta_i + C_r \times \theta_p$$

$$\theta - \theta_r = (C - C_r) \times \theta_p$$

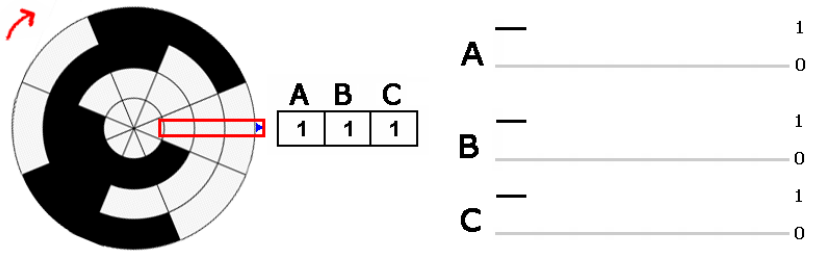
Exercise

- A velocity sensor's pitch angle is 1.8 degrees. When the velocity sensor is powered on, we rotate the motor's shaft until the pulse from the photo cell of the reference (i.e. Signal I) appears. And, the reading of channel A is +650 counts. Now, we let the motor to make a movement. If the reading from channel A is -1190 counts, what is the angular position of the motor's shaft?
- Answer:

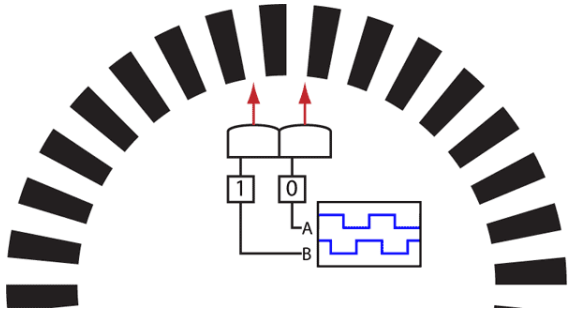
$$\theta - \theta_r = (C - C_r) \times \theta_p = (-1190 - 650) \times 1.8^0 = -3312^0$$



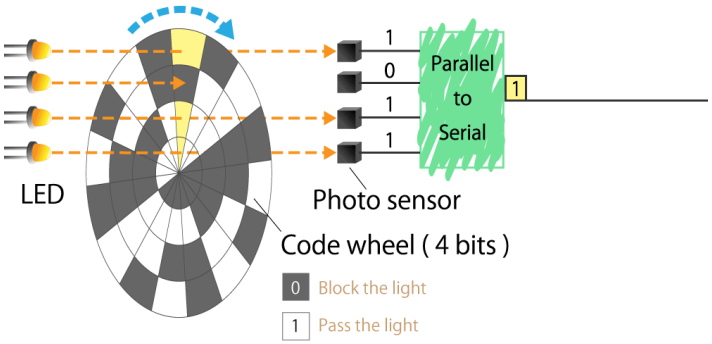
Outline of Lecture 5



- Sensory-Motor Interaction
- Design of Position Sensor

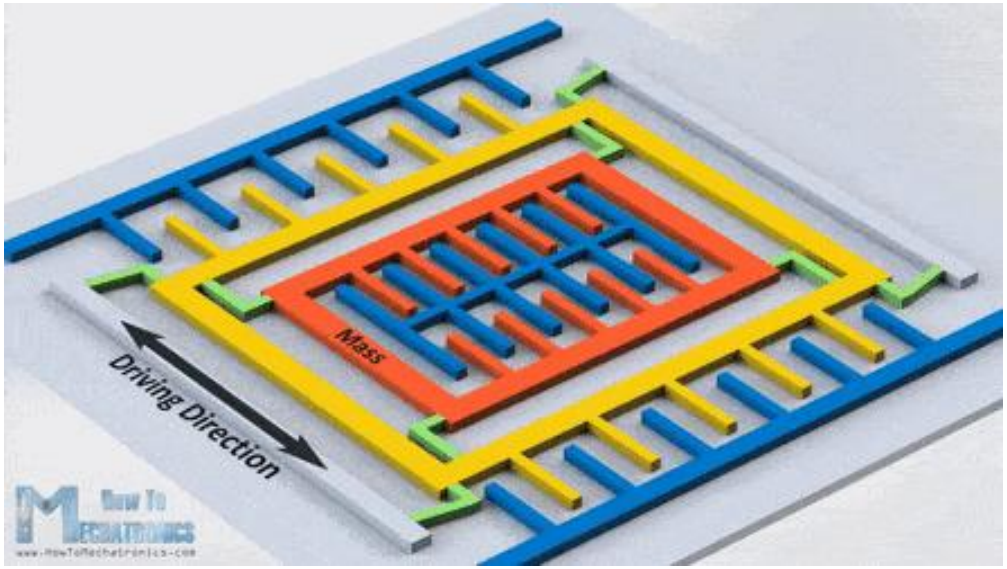


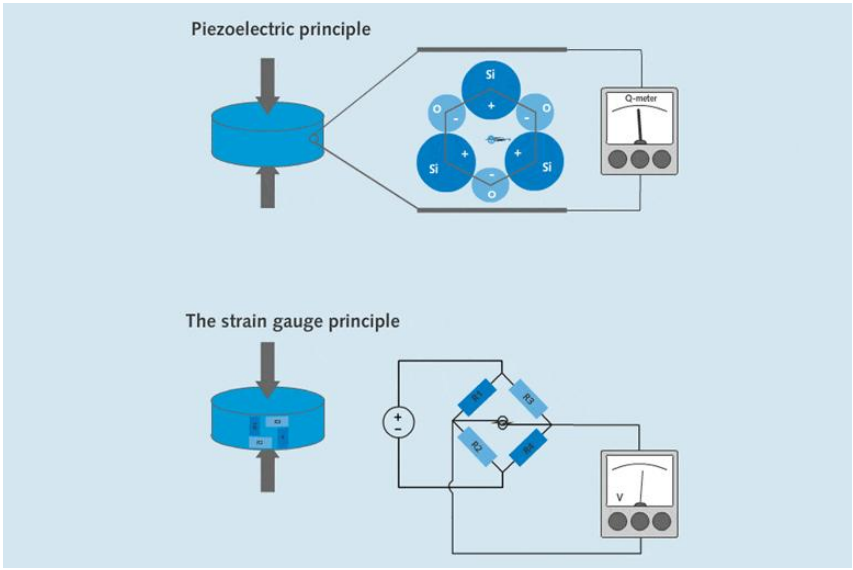
- Design of Velocity Sensor
- Design of Acceleration Sensor



- Design of Force/Torque Sensor

Illustration

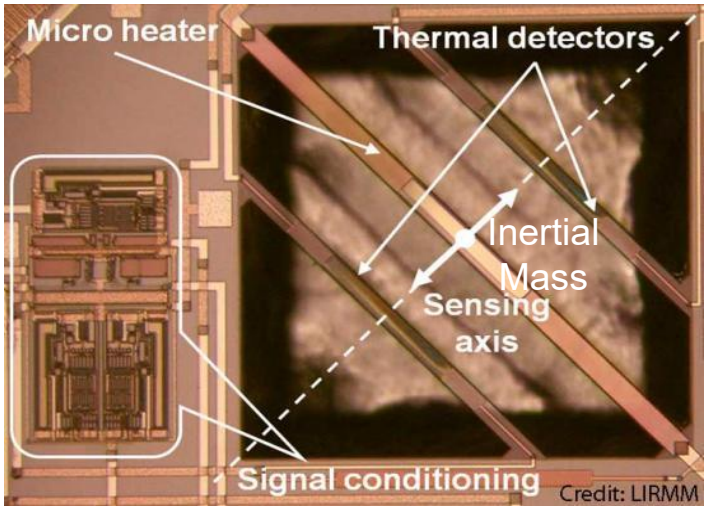


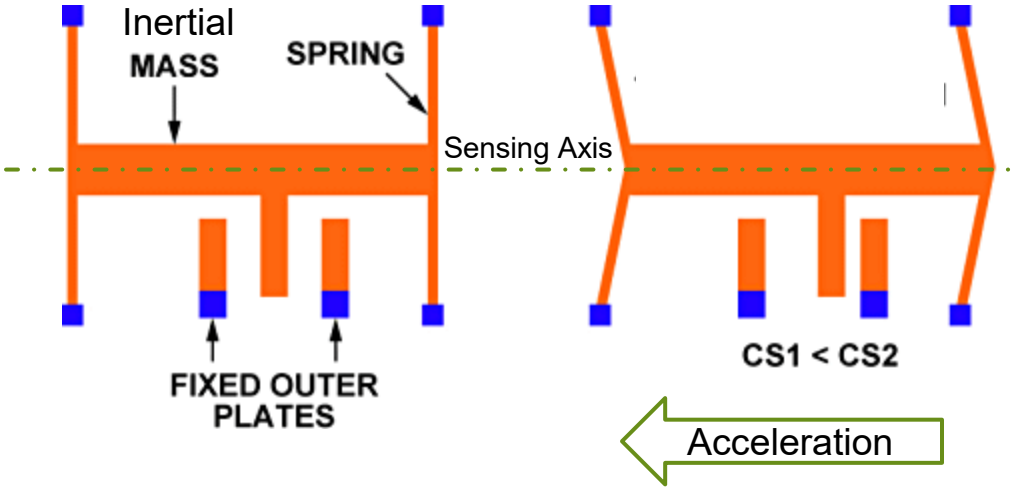


Video Showing Design and Working Principle

Summary of Design and Working Principle

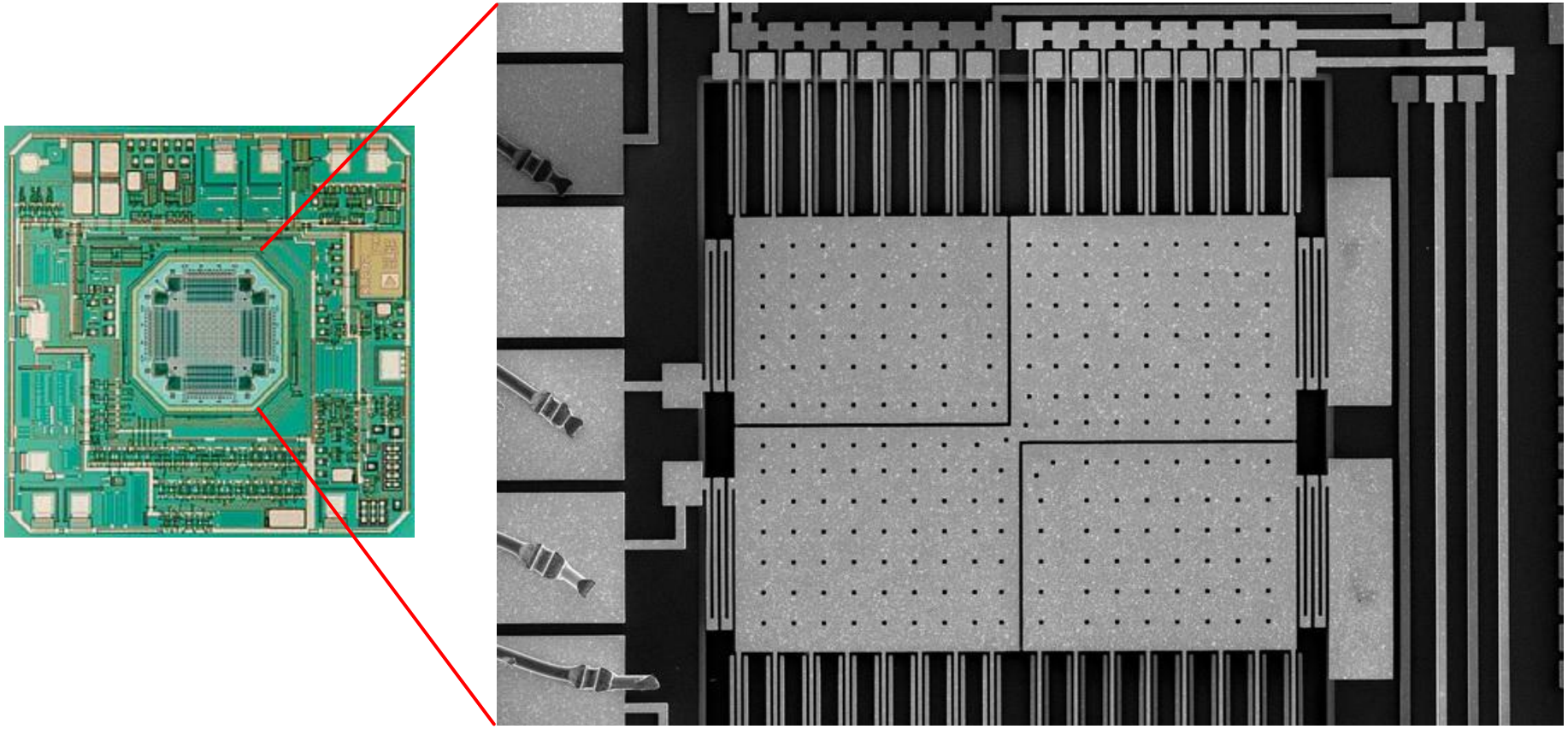
- Inertial mass is connected to sensor body by springs.
- When acceleration occurs, the inertial moves in the opposite direction due to inertial force.
- The displacement of inertial mass can be converted to: a) resistance, b) capacitance, c) temperature, etc.
- These quantities are related to accelerations.
- The measurements of these quantities result in the measurements of accelerations.



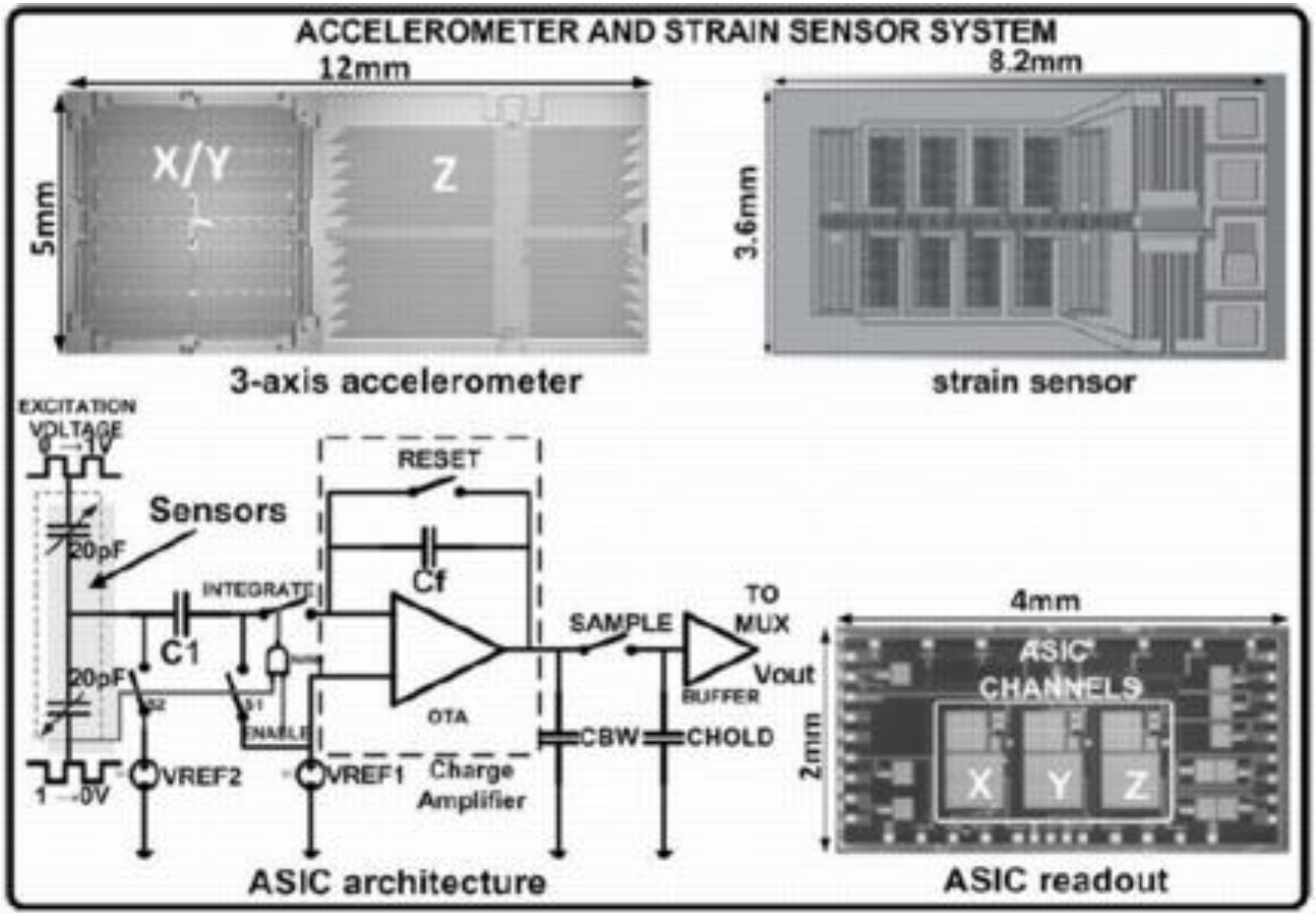


Design Example (1)

- Inertial Mass + Springs + Sensor Body + Electronic Circuits

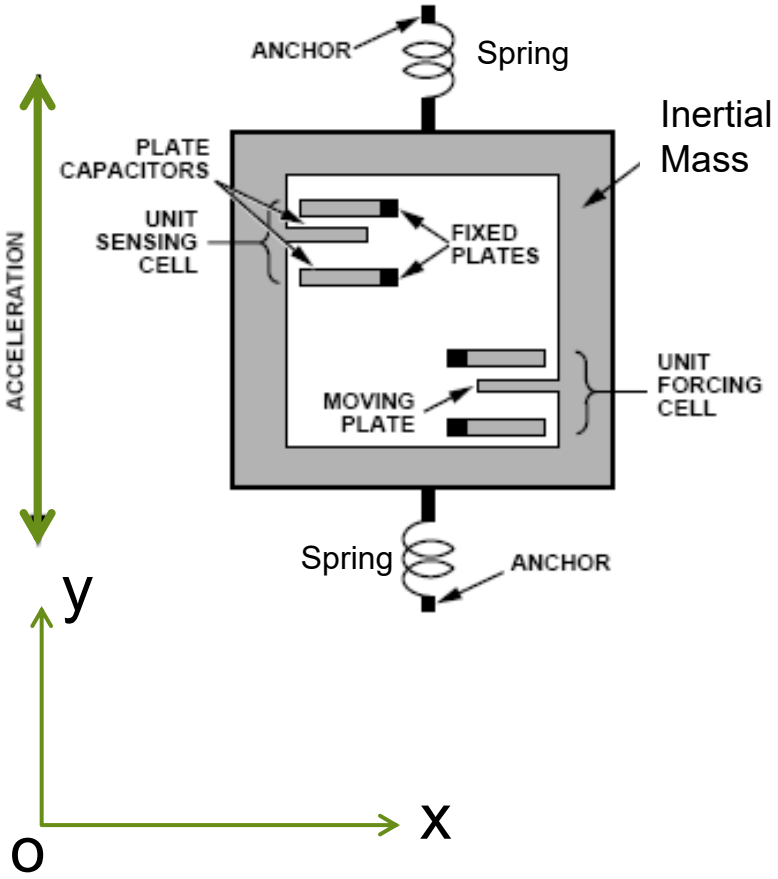


Design Example (2)



Exercise

- An inertial mass of 0.1 g is connected to sensor body by springs which have the spring constant of 1.5 N/m. If the sensor body undergoes an acceleration of 9.8 m/s2, what is the displacement of the inertial mass?



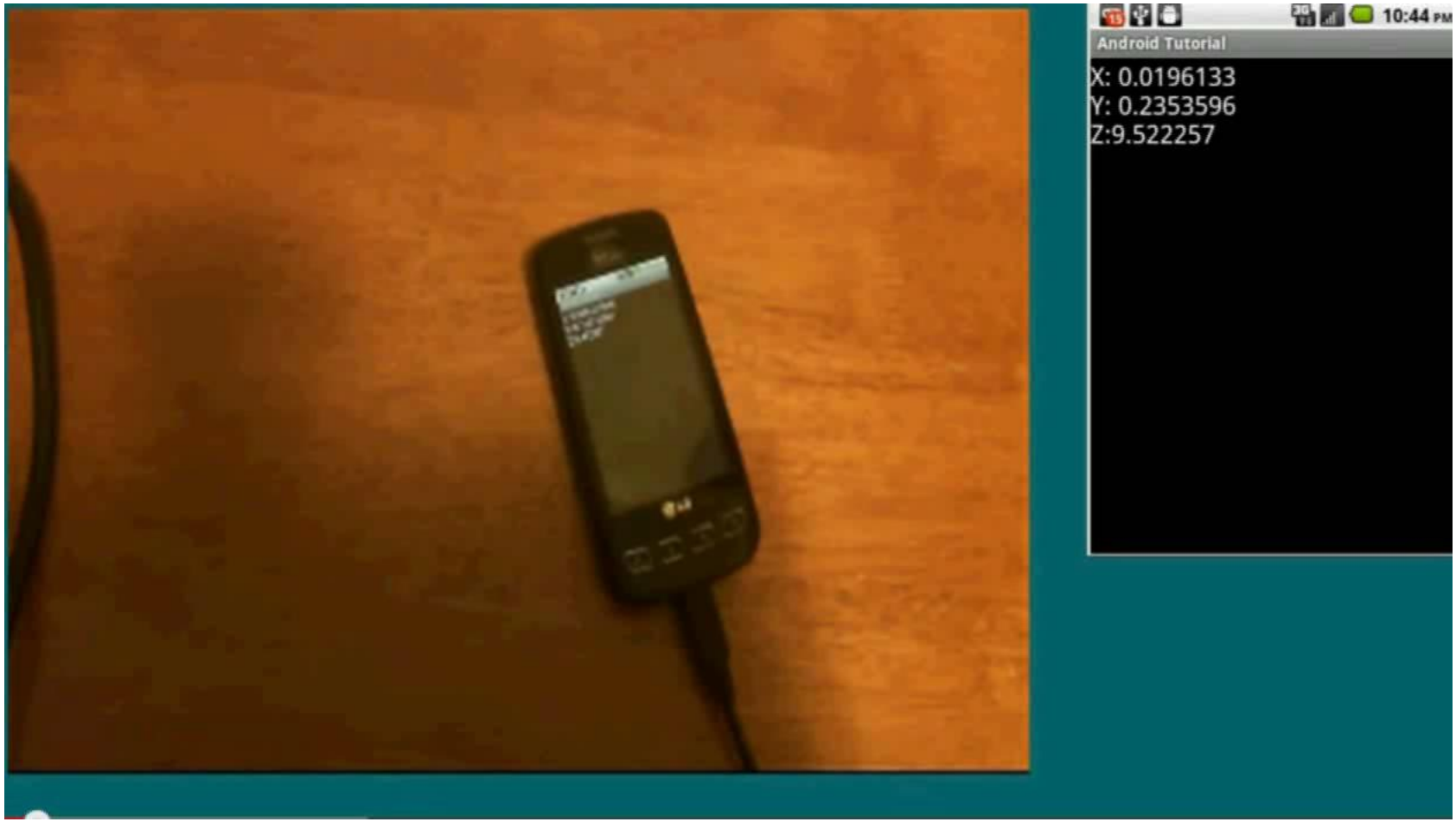
- Answer:

$$k\Delta y = ma_y$$

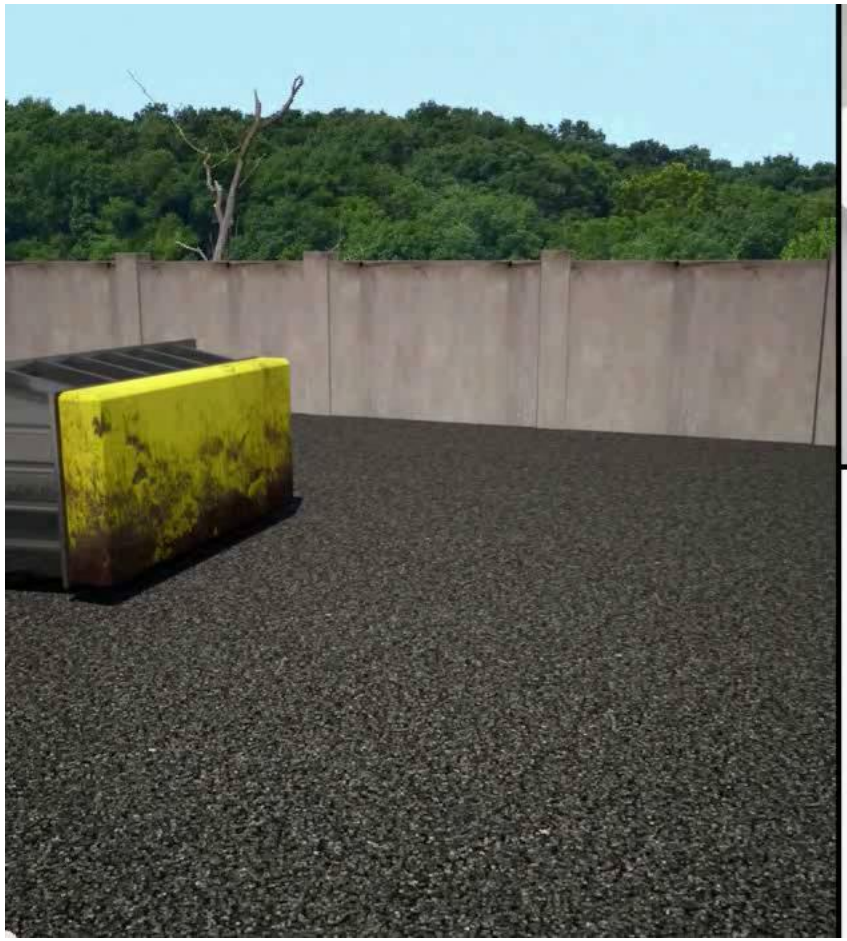
$$1.5\Delta y = 0.1\times10^{-3}\times9.8$$

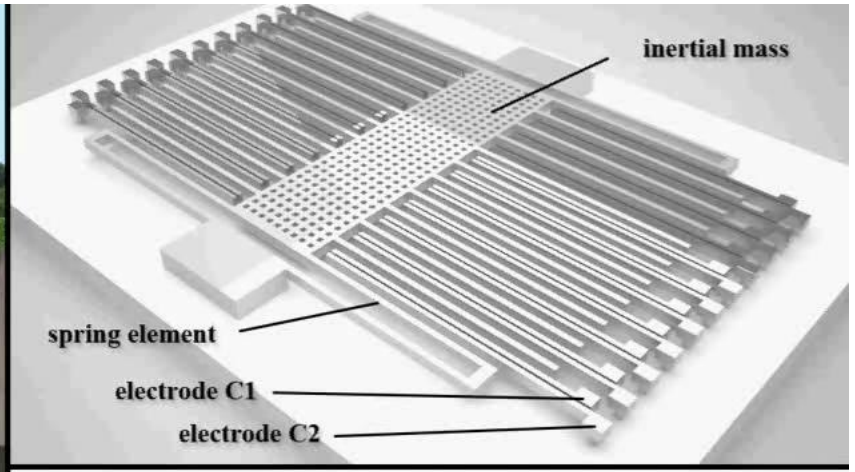
$$\Delta y = 0.65mm$$

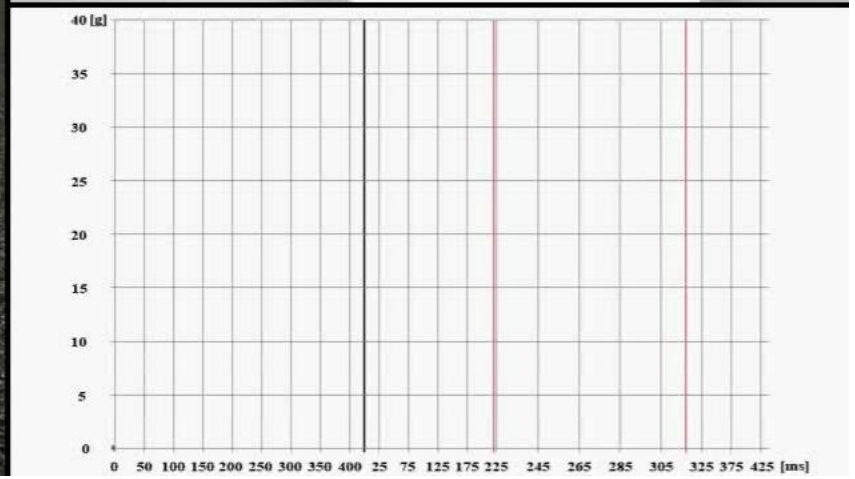
Example of Experiment



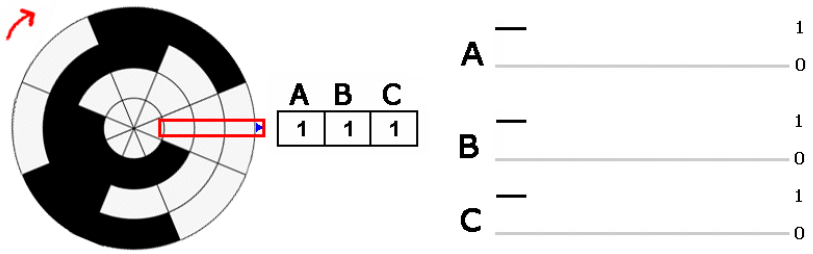
Example of Application



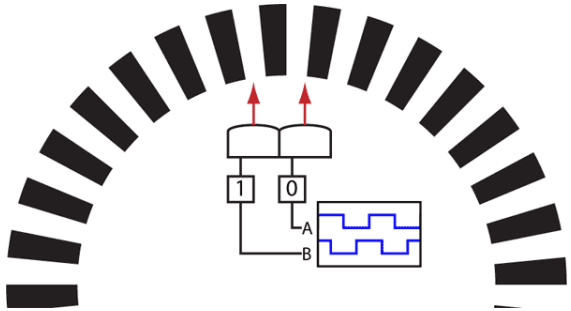




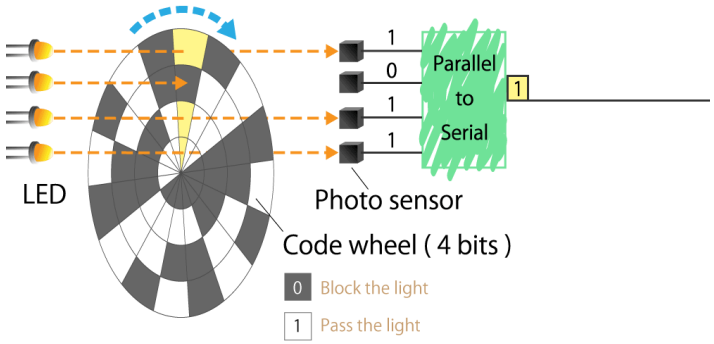
Outline of Lecture 5



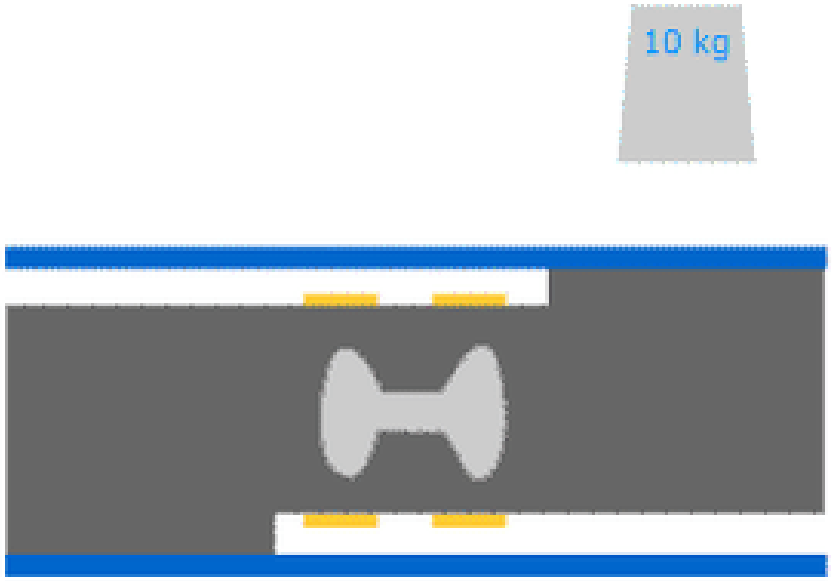
- Sensory-Motor Interaction
- Design of Position Sensor

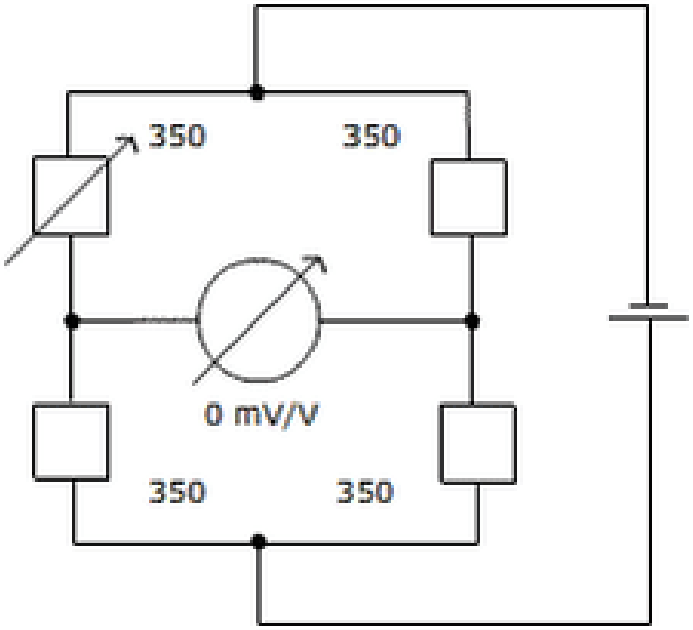


- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor

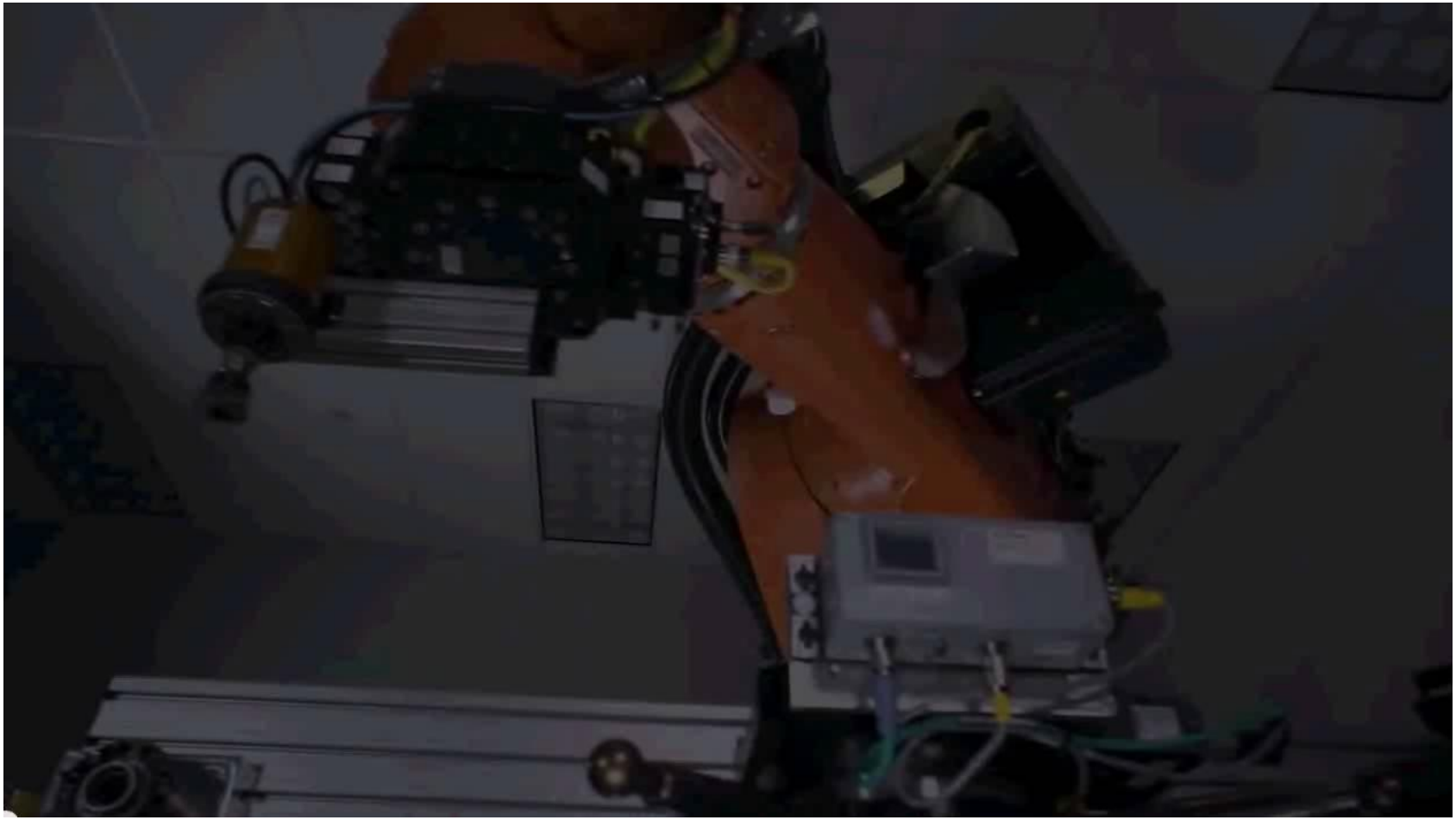


Illustration



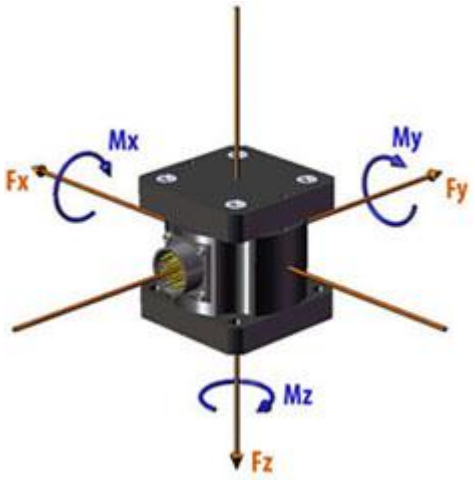


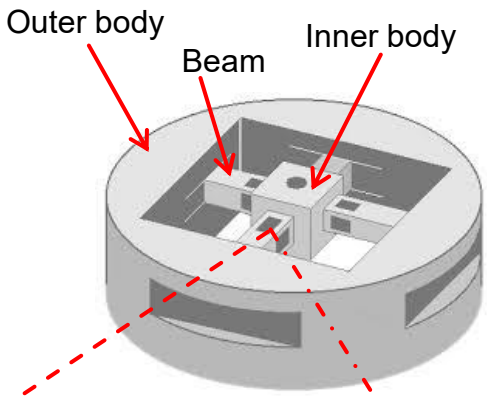
Video Showing Design and Working Principle

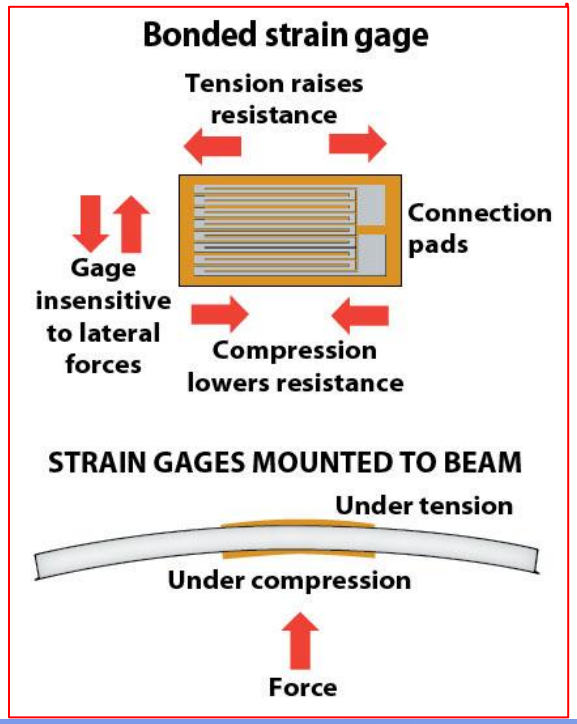


Summary of Design and Working Principle

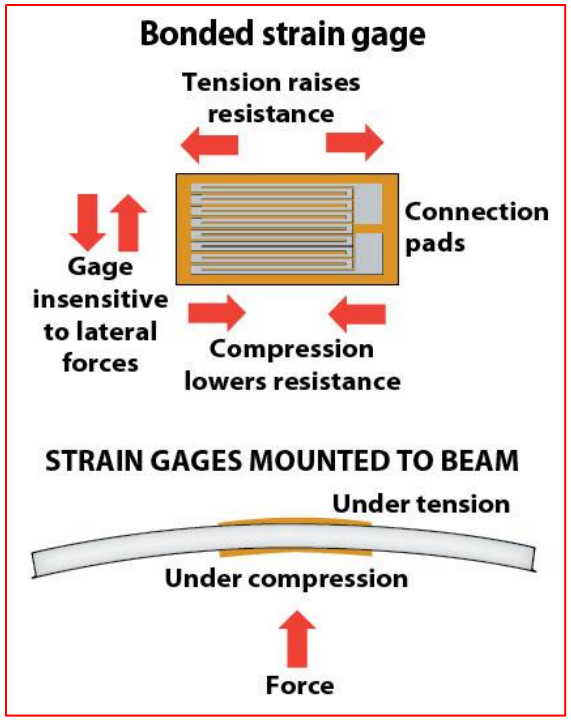
- Outer body is connected to inner body by a set of deformable beams.
- Each beam has two pairs of strain gage sensors.
- The acting forces and torques between outer body and inner body will cause the strain gages to be stretched or compressed.
- Tension and compression will cause the changes of resistances.
- The measurements of changes of resistances allow to determine the three forces and three torques.

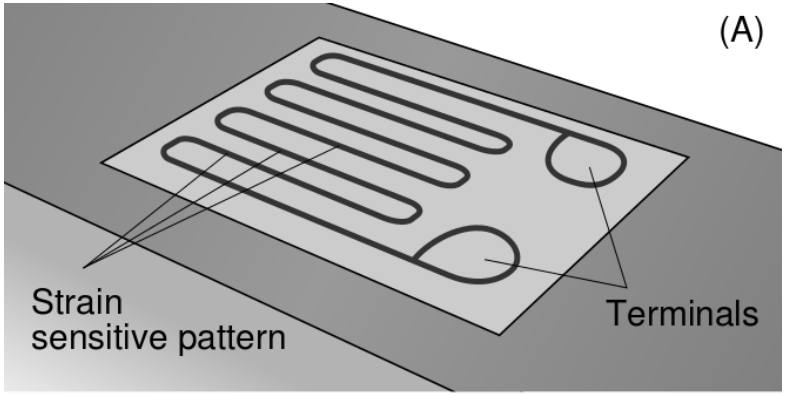


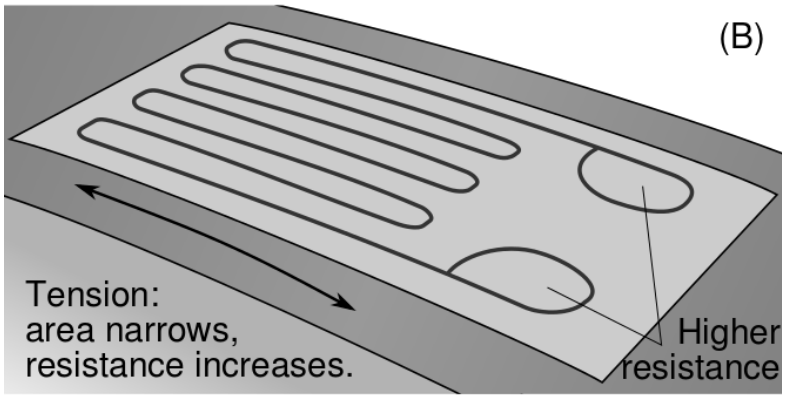


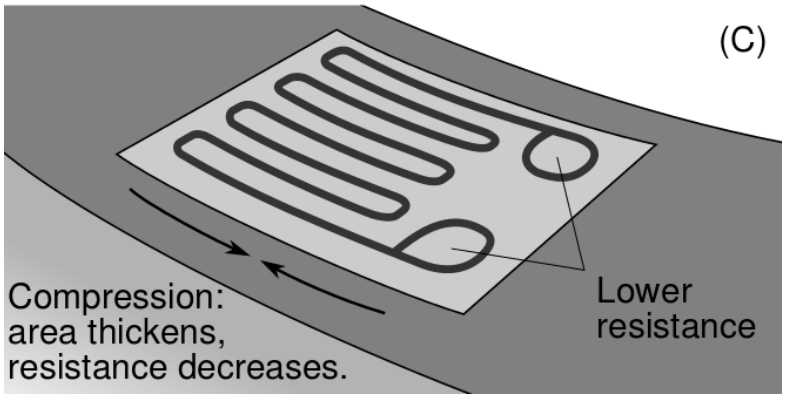


Properties of Strain Gage

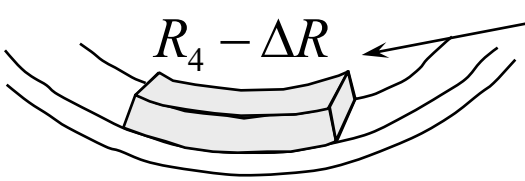




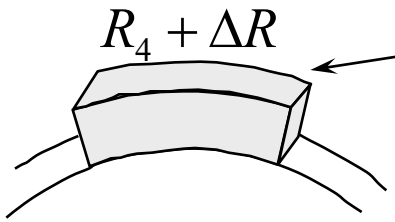




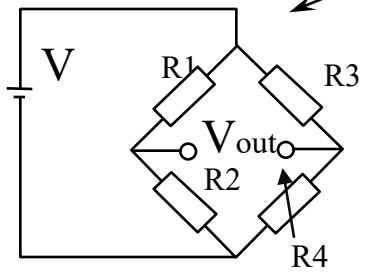
Measurement of A Single Strain Gage



Case 1: The applied force or torque causes the contraction of the strain gage. This reduces its resistance.



Case 2: The applied force or torque causes the expansion of the strain gage. This increases its resistance.



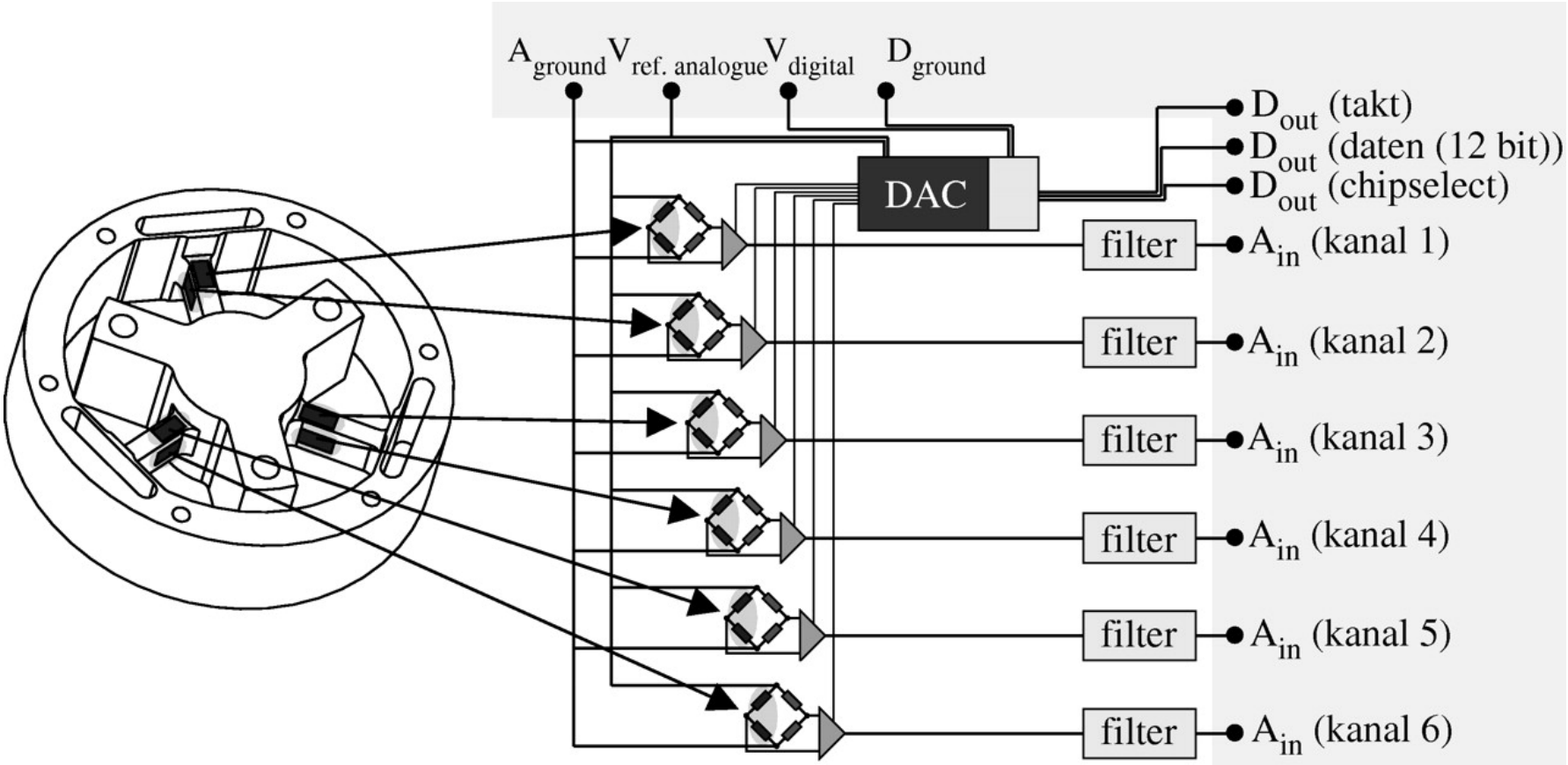
Input/Output Relation:

"Wheatstone Bridge Circuit"

$$V_{out} = \left(\frac{R_2}{R_1 + R_2} - \frac{R_4 + \Delta R_4}{R_3 + R_4 + \Delta R_4}\right) \bullet V$$

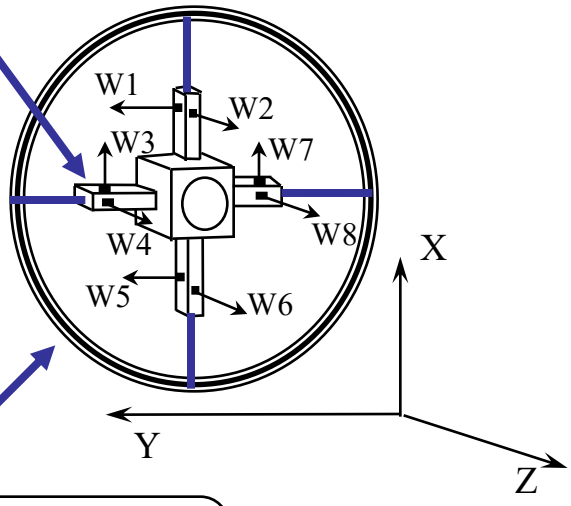
Measurement of Six Strain Gages

- Outer Ring + 3 Beams + Inner Ring + Wheatstone Bridge Circuits



Example of Doing Sensor Calibration

1. Each beam has two pairs of strain gage sensors which are placed at two adjacent facets.



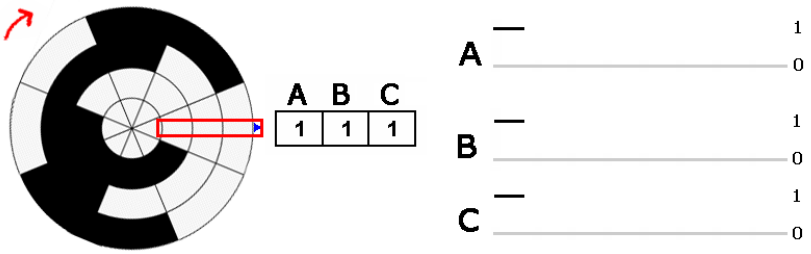
2. Four beams have eight pairs of strain gage sensors. So, eight values of resistances can be measured.

3. The eight values of measurements can be mapped to the output of three forces and three torques by a matrix:

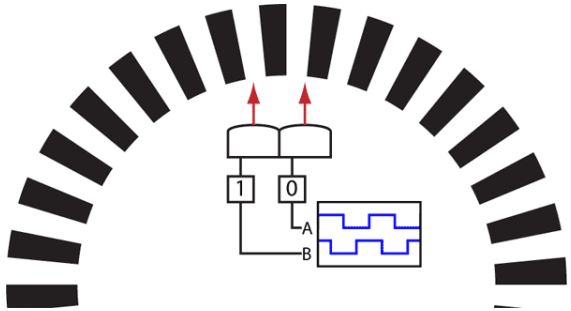
$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{pmatrix} 0 & 0 & K_{13} & 0 & 0 & 0 & K_{17} & 0 \\ K_{21} & 0 & 0 & 0 & K_{25} & 0 & 0 & 0 \\ 0 & K_{32} & 0 & K_{34} & 0 & K_{36} & 0 & K_{38} \\ 0 & 0 & 0 & K_{44} & 0 & 0 & 0 & K_{48} \\ 0 & K_{52} & 0 & 0 & 0 & K_{56} & 0 & 0 \\ K_{61} & 0 & K_{63} & 0 & K_{65} & 0 & K_{67} & 0 \end{pmatrix} \bullet \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \end{bmatrix}$$

Example of Testing and Application

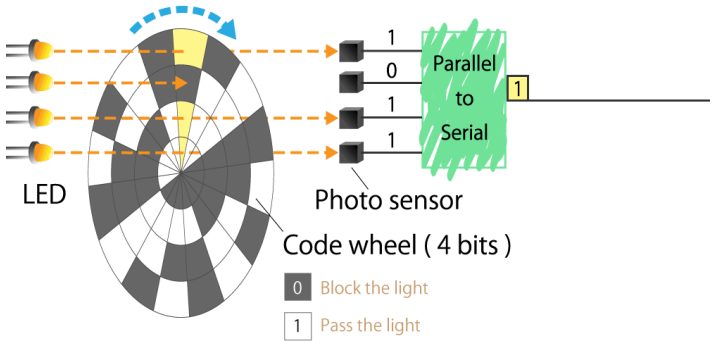
Summary of Lecture 5



- Sensory-Motor Interaction
- Design of Position Sensor

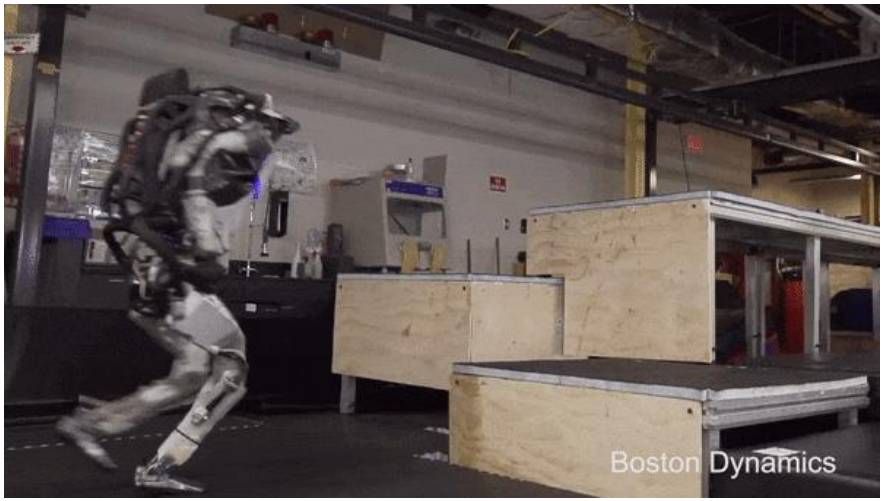


- Design of Velocity Sensor
- Design of Acceleration Sensor
- Design of Force/Torque Sensor



Outline of Module 1

- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems



What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 1

MA4825 Robotics

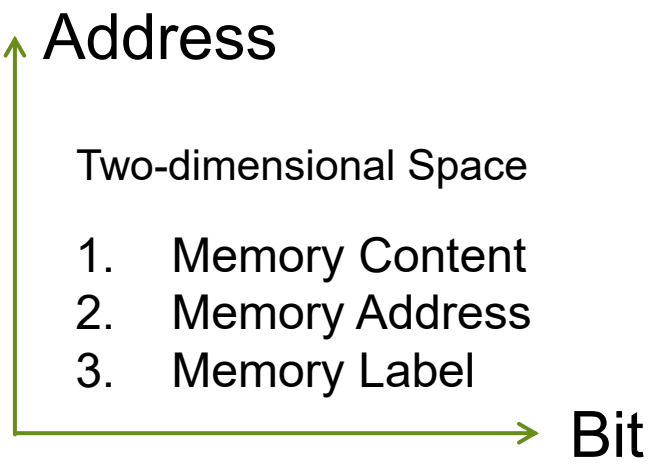
Lecture 6

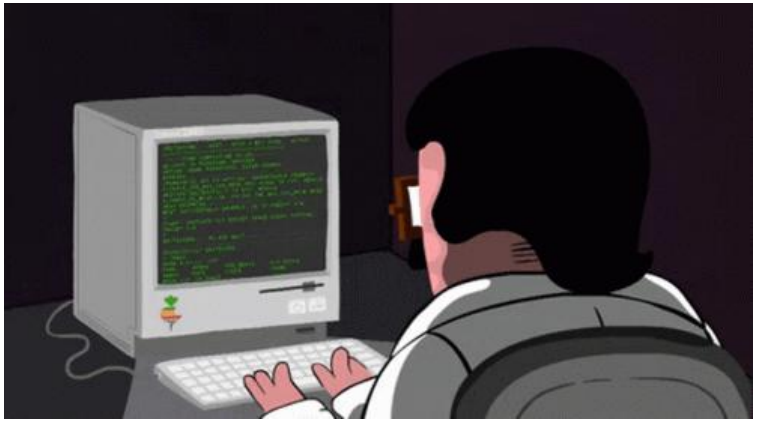
Robot's Programming Systems

Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

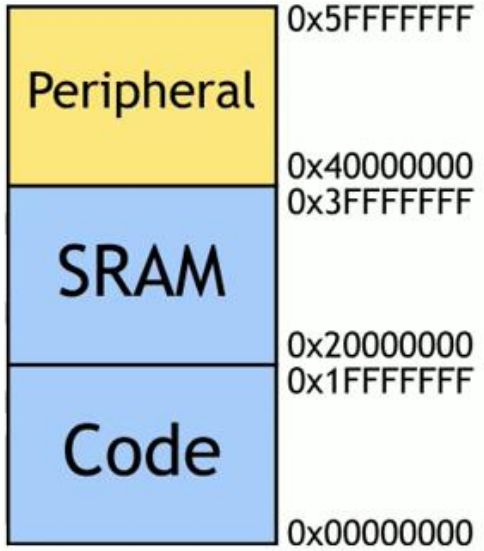
Outline



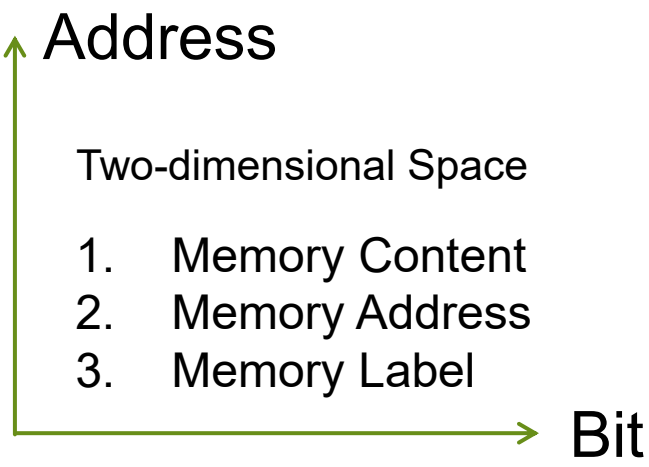


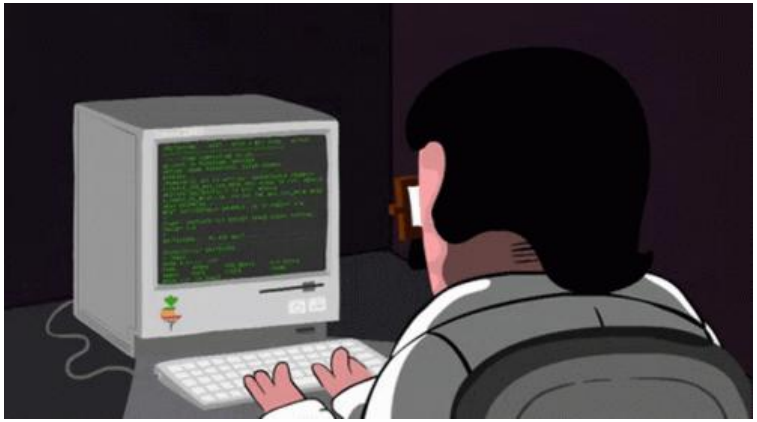
You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



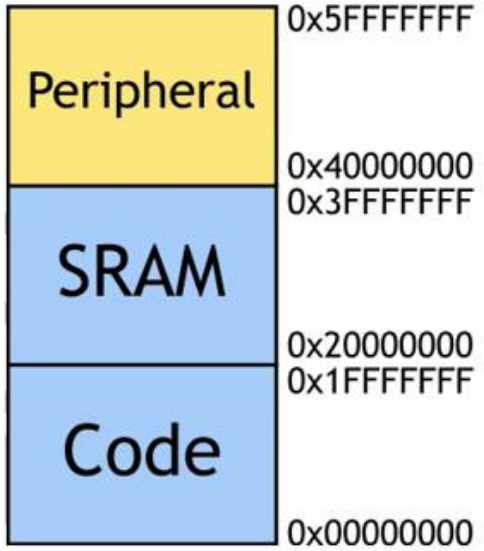
Outline





You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



Programming versus Writing

Programming

Use of Programming Languages

- You have a solution in mind
- You organize your solution
- You compose your program
- You test your program
- You deploy your program

Use of Keil

Writing

Use of Natural Languages

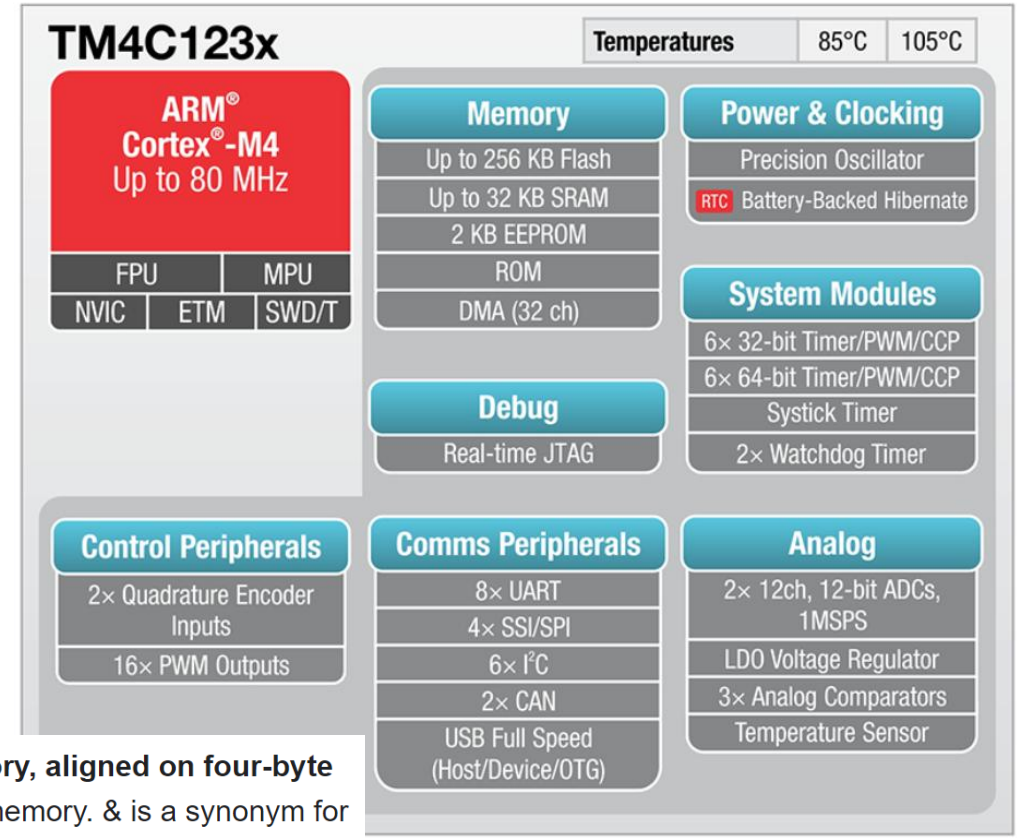
- You have a story in mind
- You organize your story
- You compose your texts
- You proof-read your texts
- You publish your texts

Use of LaTeX, Word

How to organize your solution?

FPU: Floating-Point Unit
MPU: Multi-core Processing Unit

- Step 1: To plan arithmetic and/or logical computations underlying your solution
- Step 2: To allocate memory resources which are indispensable for the implementation of the planned computations.



The DCD **directive allocates one or more words of memory, aligned on four-byte boundaries**, and defines the initial runtime contents of the memory. & is a synonym for DCD . DCDU is the same, except that the memory alignment is arbitrary.

Example of doing x = (a + b) - c

```
                EXPORT Start
                AREA    progA, CODE, READONLY
var_a   DCD     5       ; a = 5    4 byte words
var_b   DCD     6       ; b = 6
var_c   DCD     7       ; c = 7
var_x   DCD     0       ; x = 4

                        ; x = (a + b) – c
Start           ADR r4,var_a    ; get address for a
                LDR r0,[r4]     ; get value of a
                ADR r4,var_b    ; get address for b, reusing r4
                LDR r1,[r4]     ; get value of b
                ADD r3,r0,r1    ; compute a + b
                ADR r4,var_c    ; get address for c
                LDR r2,[r4]     ; get value of c
                SUB r3,r3,r2    ; complete computation of x
                ADR r4,var_x    ; get address for x
                STR r3,[r4]     ; store value of x
stop    B       stop
                END
```

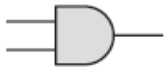
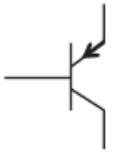
Allocation of Memory

Planning of Computations

ADR loads address to a register

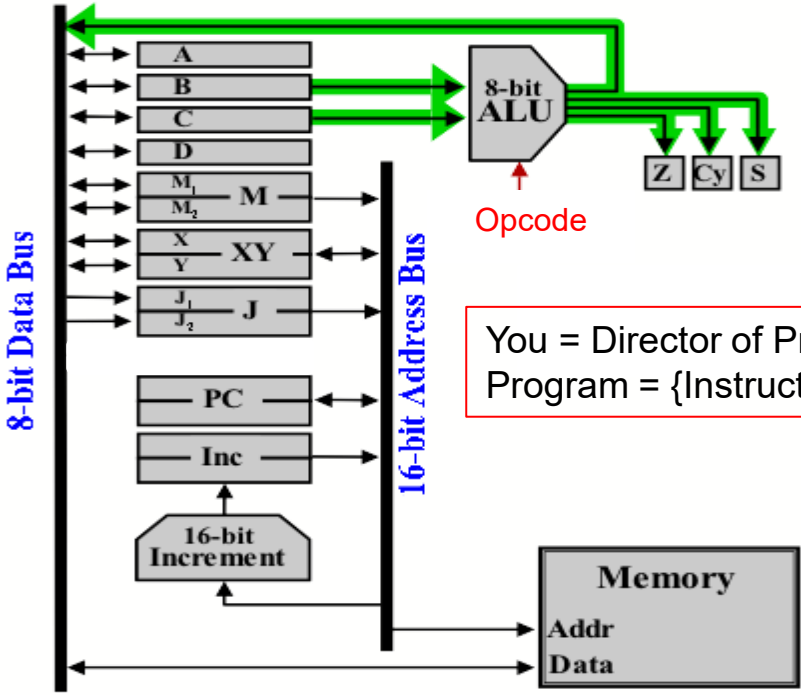
Role of Programmers: Planning and Allocation

- Data = {Values, Symbols, Addresses, Instructions}
- Instructions = {Op Code + Addresses + Value/Symbol}

Applications/OS	
Languages	C++, Java
ISA	EOR r3,r2,r1 BEQ Table
Microarchitecture	
Gates	
Transistors	

Algorithms or Solutions

Problems to be solved

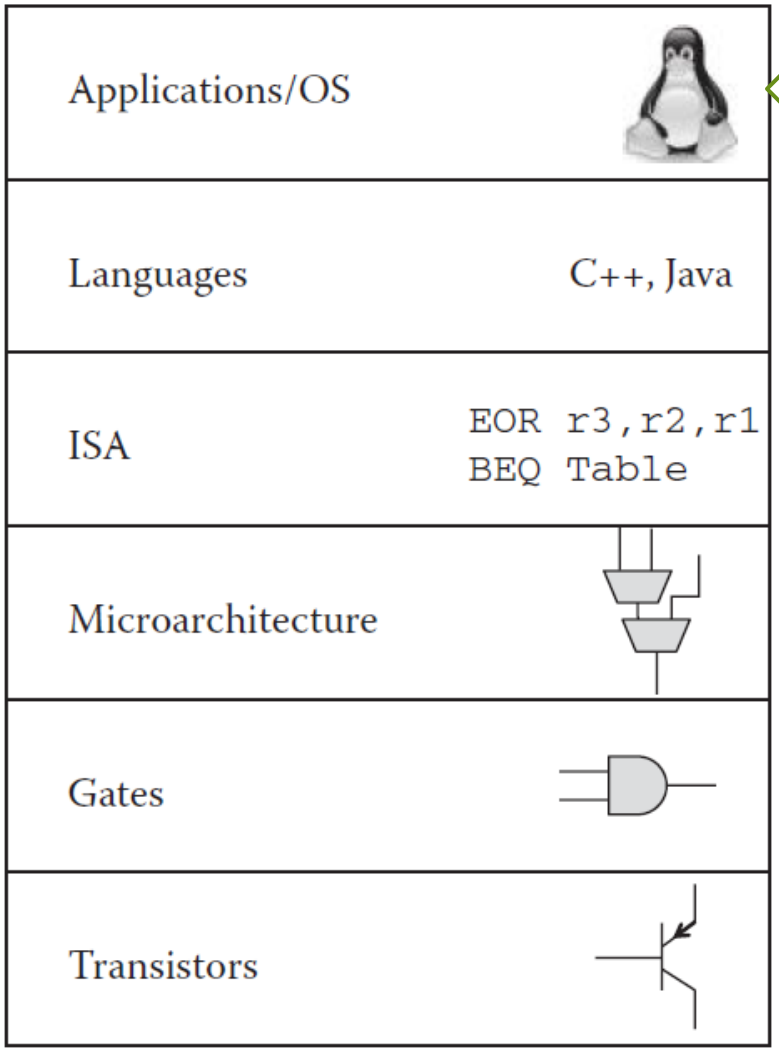


You = Director of Program
Program = {Instructions}

Memory = {Address + Data/Instruction}

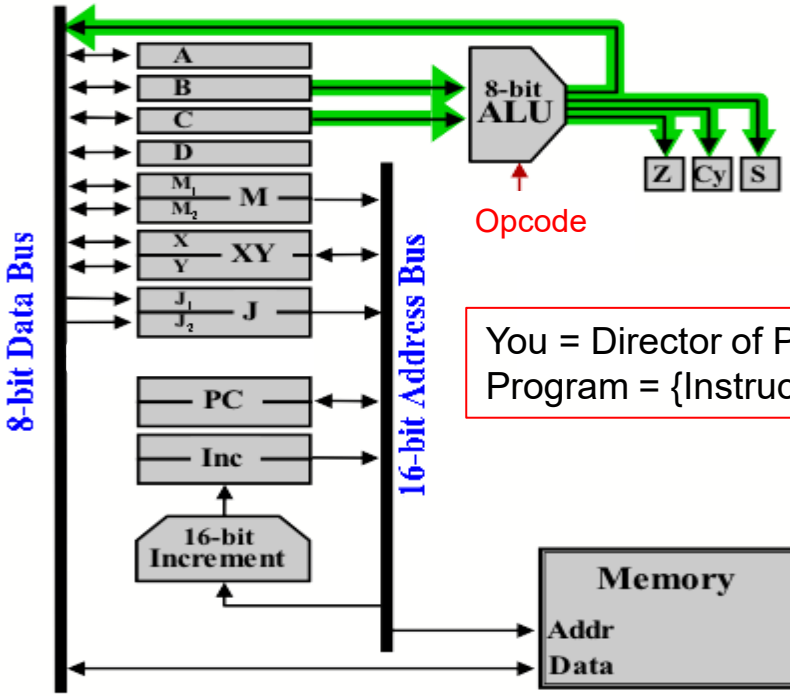
Role of Microprocessors: Do Cycle-by-Cycle Executions

- Data = {Values, Symbols, Addresses, Instructions}
- Instructions = {Op Code + Addresses + Value/Symbol}



Algorithms or Solutions

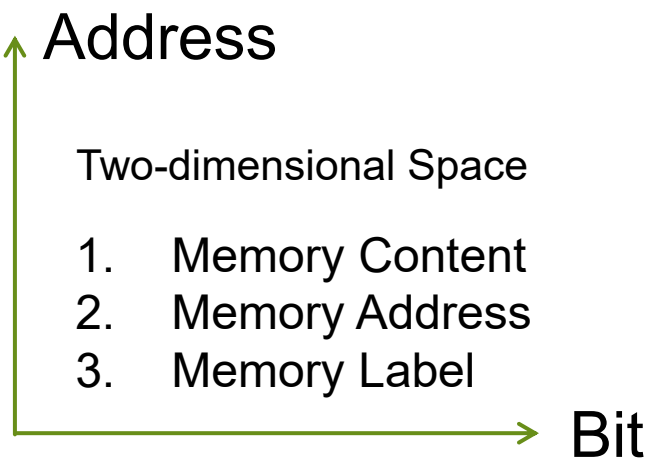
Problems to be solved

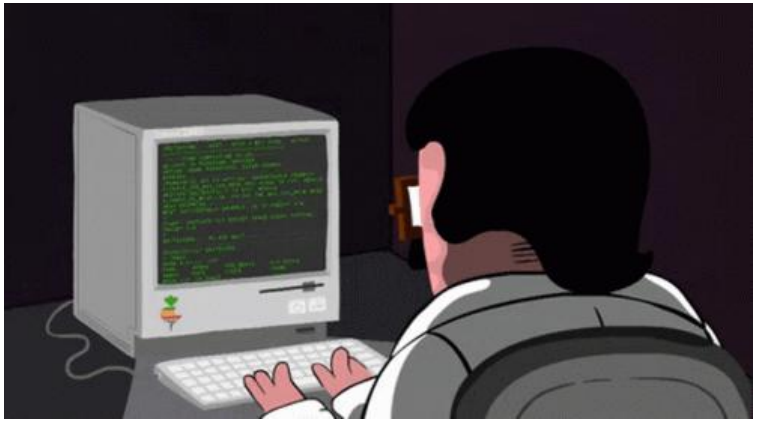


You = Director of Program
Program = {Instructions}

Memory = {Address + Data/Instruction}

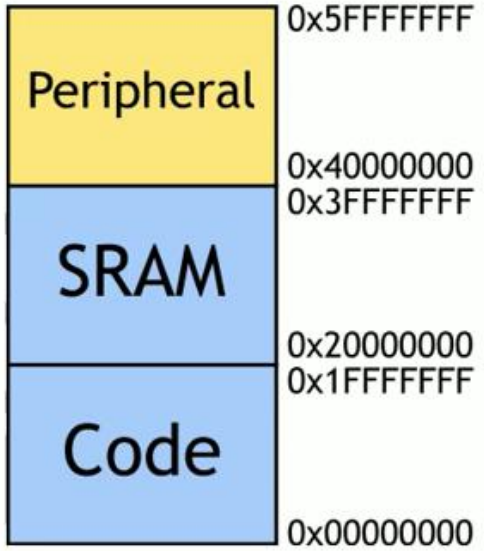
Outline





You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



Procedure of Planning Computations

- Step 1: Describe your solution in the form of mathematics or logics.
- Step 2: Transform your solution into algorithms which consist of series of arithmetic and/or logical computations.
- Step 3: Draw flowcharts which interconnect arithmetic and/or logical computations in terms of their inputs and outputs.
- Step 4: Describe flowcharts with the use of a programming language.

Example of Transforming Solution into Algorithm

Solution:

$$ax^2 + bx + c = 0$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Algorithm:

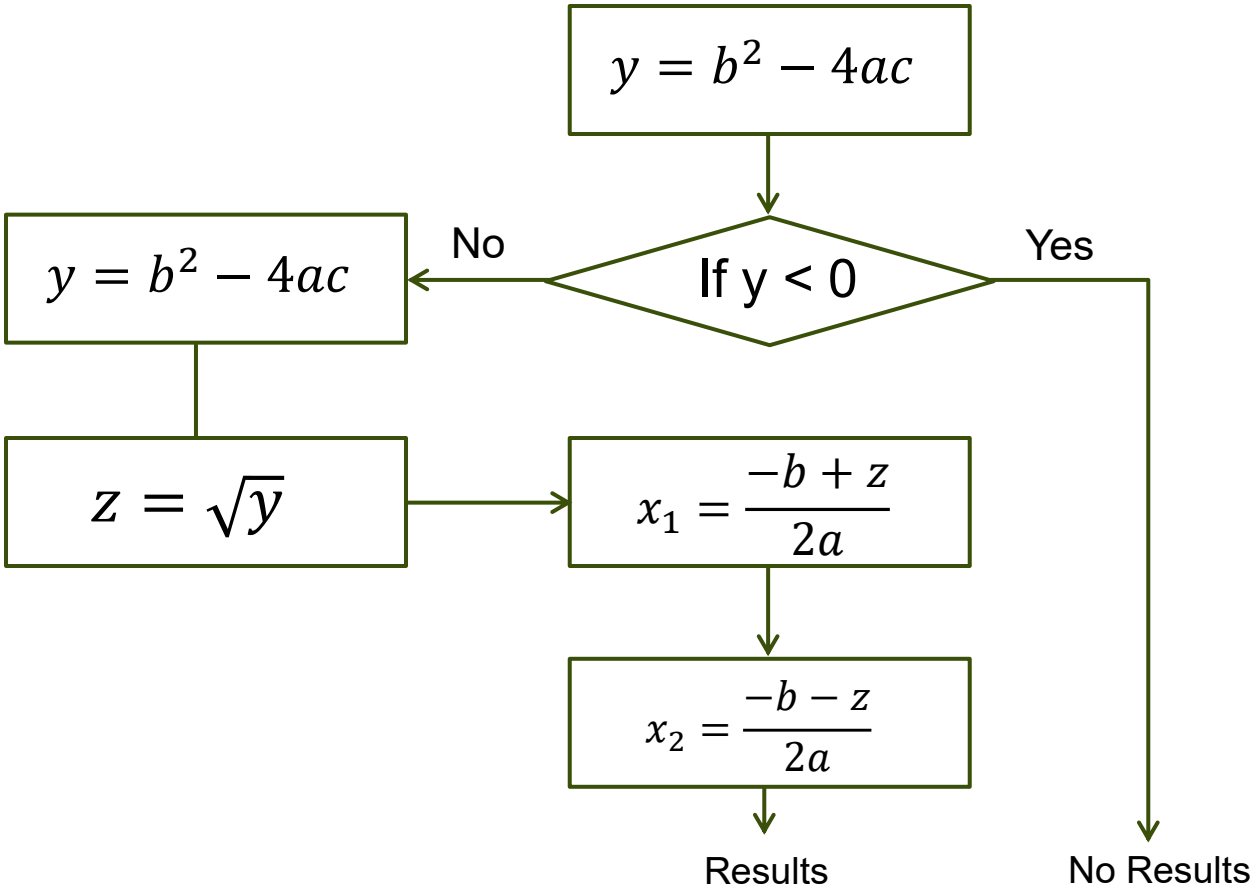
First do: $y = b^2 - 4ac$

Then do: $z = \sqrt{y}$

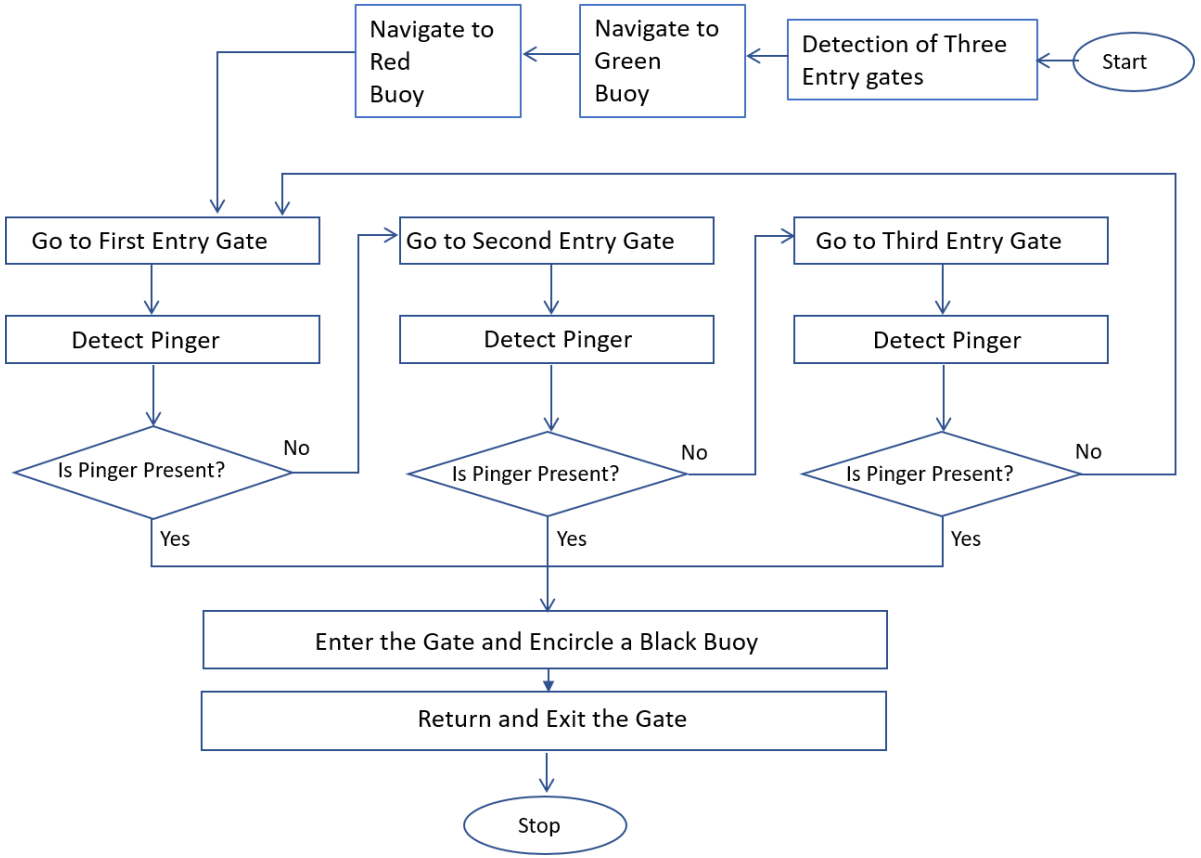
Then do: $x_1 = \frac{-b + z}{2a}$

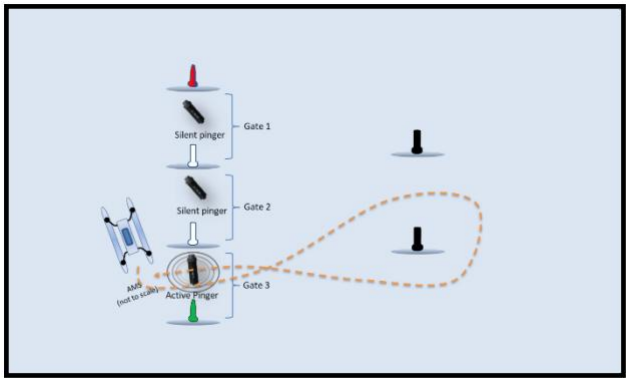
Then do: $x_2 = \frac{-b - z}{2a}$

Example of Flowchart Corresponding to Algorithm

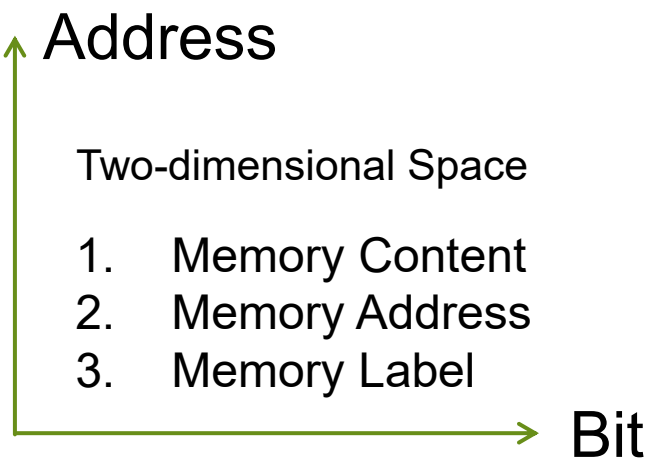


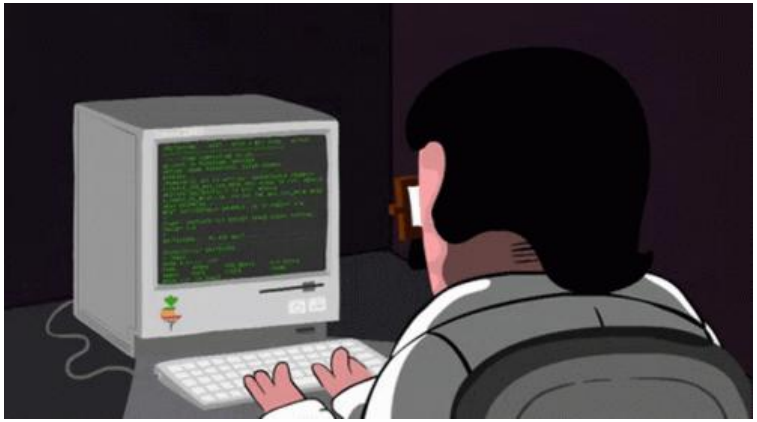
Example of Flowchart Corresponding to Algorithm





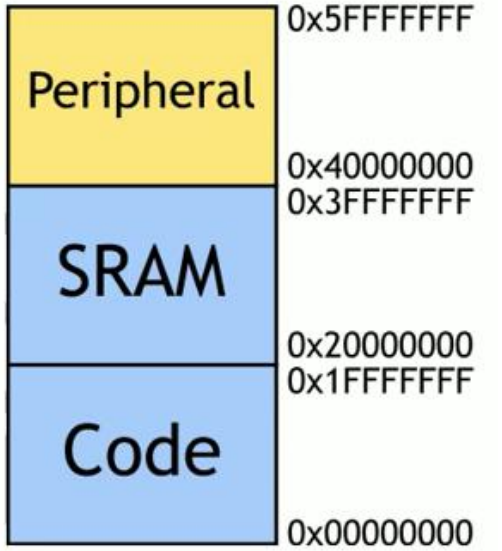
Outline





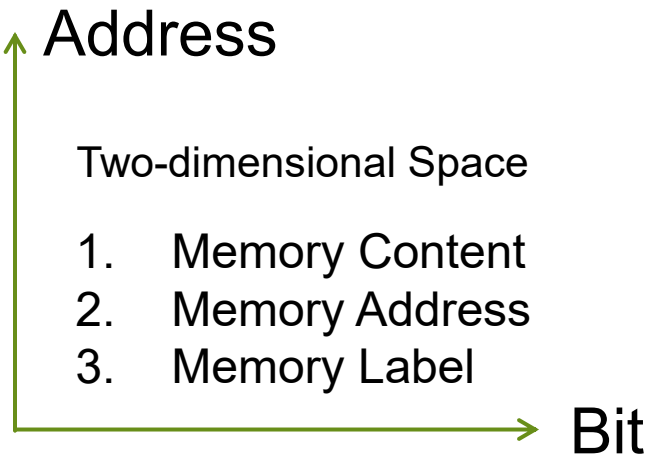
You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



Motivation

- The hardware could support the concurrent running of multiple application programs.
- The ALU is being shared among these programs. This means that an application program has the full access to the ALU during the time when ALU is allocated to it.
- However, memory units are not shared in general.
- Hence, each application program must take care of memory allocation explicitly.



ALU is shared while memory is to be reserved

Scenario of Memory Allocation

- Scenario 1: Allocation of memory for housing a program itself.
- Scenario 2: Allocation of memory for housing constants, parameters and variables of a program.

Address

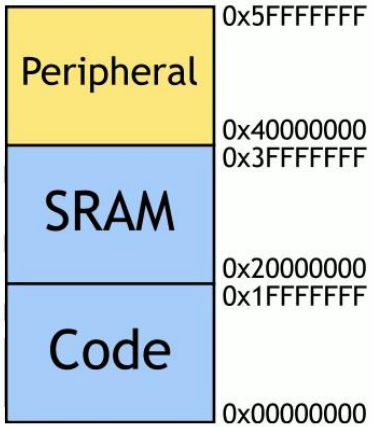
Two-dimensional Space

1. Memory Content
2. Memory Address
3. Memory Label

Bit

Cortex-M Memory

On-chip Memory Space



- On-chip code, data, and I/O are located in the first 1.5 GiB of memory space
- Each is allocated 0.5 GiB
- May use physically separate buses for each space

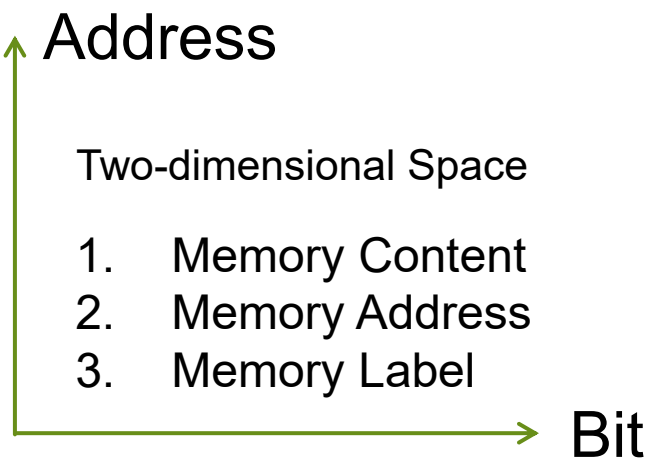
GiB = Giga Byte

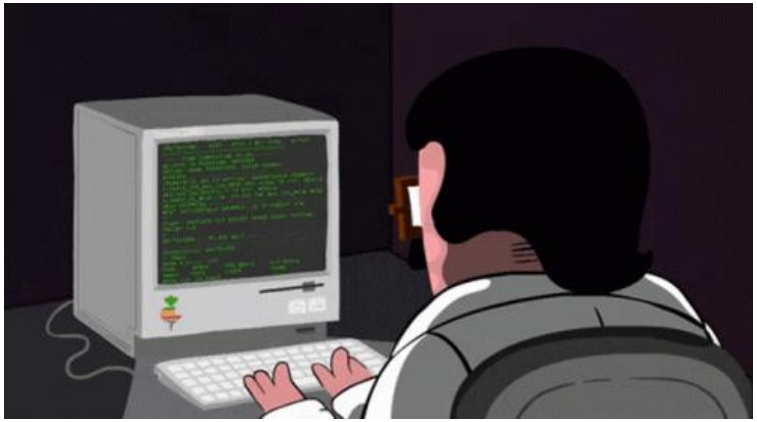
Example of Memory Allocation

```
                EXPORT Start
                AREA   progB, CODE, READONLY
var_a   DCD     1       ; a = 1    (DCD creates 4 byte variables)
var_b   DCD     15      ; b = 15
var_z   DCD     0       ; z = 5

Start                   ; z = (a << 2) | (b & 15)
        ADR r4,var_a    ; get address for a
        LDR r0,[r4]     ; get value of a
        MOV r0, r0, LSL #2   ; perform shift  ---> r0 = (a<<2)
        ADR r4, var_b   ; get address for b
        LDR r1, [r4]    ; get value of b
        AND r1, r1, #15 ; perform AND  ---> r1 = (b & 152)
        ORR r1, r0, r1  ; perform OR   ---> r1 = (a<<2) | (b & 152)
        ADR r4, var_z   ; get address for z
        STR r1, [r4]    ; store value for z
stop    B       stop
        END
```

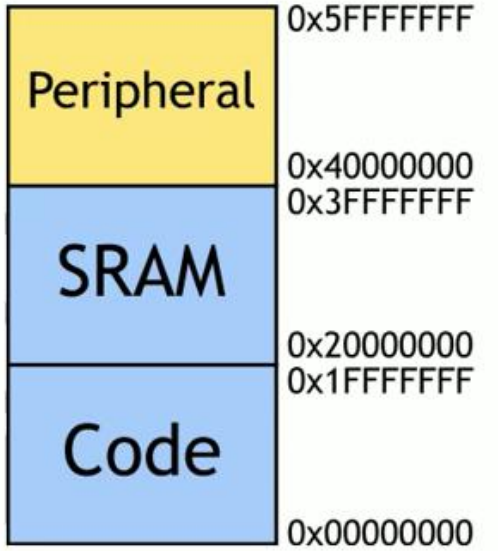
Outline





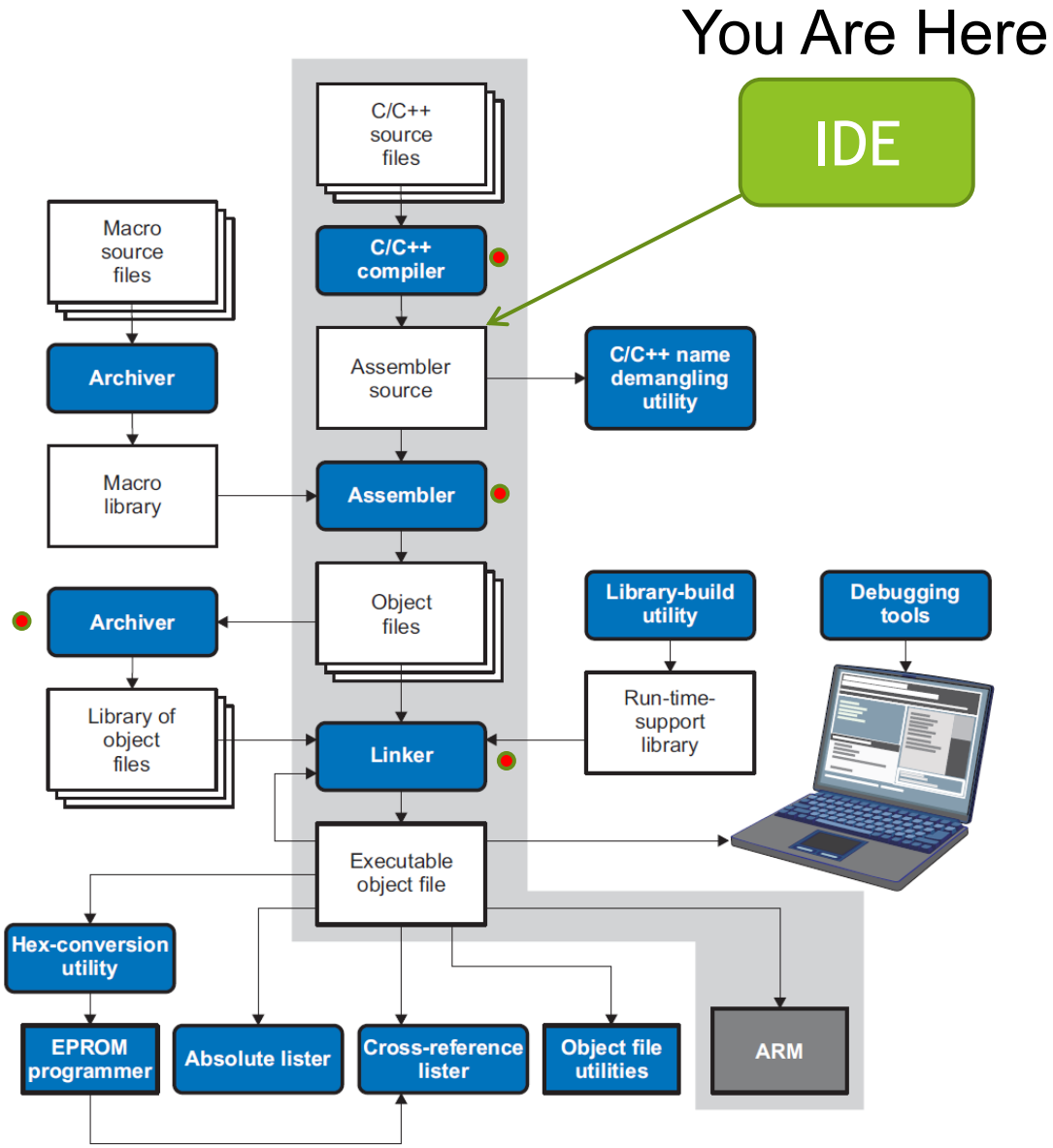
You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



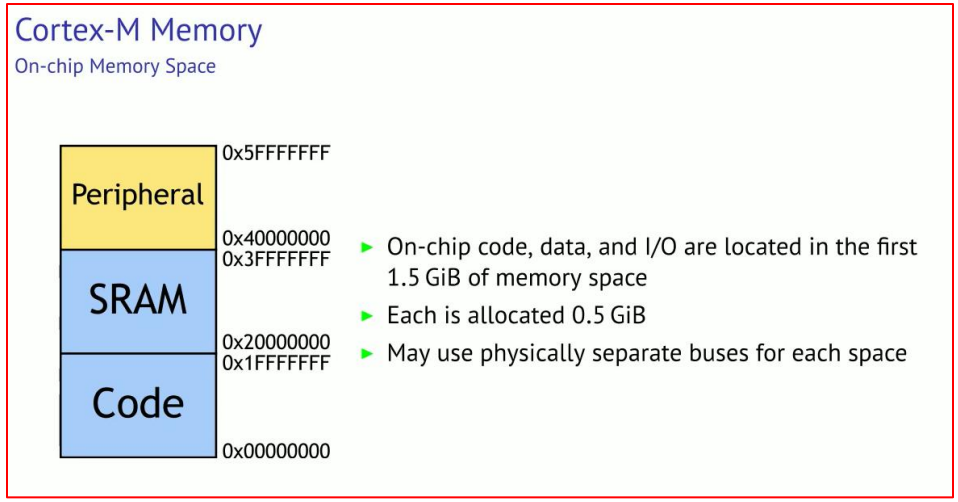
Overview

- Compiler:
  - From .c files to .s files
- Assembler:
  - From .s files to .o files
- Linker:
  - From .o files to .exe files
  - May be linked with .a files



Assembler Directives for Memory Destination

Mnemonic and Syntax	Description
.bss *symbol, size in bytes*[, *alignment* [, *bank offset*]]	Reserves *size* bytes in the .bss (uninitialized data) section
.data	Assembles into the .data (initialized data) section
.sect "*section name*"	Assembles into a named (initialized) section
.text	Assembles into the .text (executable code) section
symbol **.usect** "*section name*", *size in bytes* [, *alignment*[, *bank offset*]]	Reserves *size* bytes in a named (uninitialized) section

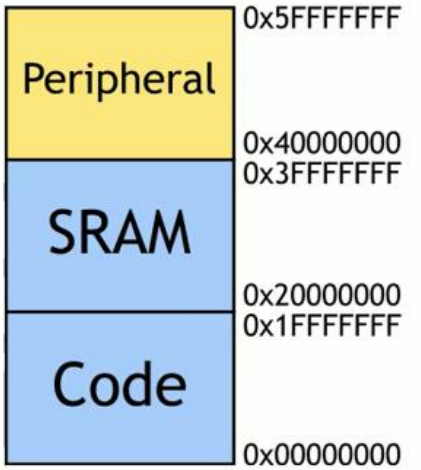


SRAM: Data Only
CODE: Instructions/Data

- Short-term memory
- Working memory

Examples

.space reserves a block of bytes



In this example, code is assembled into the .data and .text sections.

Machine Readable

Human Readable

```
 1                                  ***********************************************
 2                                  **          Reserve space in .data.          **
 3                                  ***********************************************
 4 00000000                                .data
 5 00000000                                .space  0CCh
 6
 7                                  ***********************************************
 8                                  **          Assemble into .text.             **
 9                                  ***********************************************
10 00000000                                .text           ; Constant into .data
11           00000000              INDEX    .set    0
12 00000000 E3A00000                       MOV     R0, #INDEX
13
14                                  ***********************************************
15                                  **          Assemble into .data.             **
16                                  ***********************************************
17 000000cc                        Table:   .data
18 000000cc FFFFFFFF                       .word   -1      ; Assemble 32-bit
19                                                         ; constant into .data.
20
21 000000d0 FF                             .byte   0FFh    ; Assemble 8-bit
22                                                         ; constant into .data.
23
24                                  ***********************************************
25                                  **          Assemble into .text.             **
26                                  ***********************************************
27 00000004                                .text
28 00000004 000000CC               con:     .field  Table, 32
29 00000008 E51F100C                       LDR     R1, con
30 0000000c E5912000                       LDR     R2, [R1]
31 00000010 E0802002                       ADD     R2, R0, R2
```

Assembler Directives for Data's Memory Allocation

Mnemonic and Syntax	Description
.cstring {*expr*$_1$|"*string*$_1$"}[,... , {*expr*$_n$|"*string*$_n$"}]	Initializes one or more text strings
.double *value*$_1$[, ... , *value*$_n$]	Initializes one or more 64-bit, IEEE double-precision, floating-point constants
.field *value*[, *size*]	Initializes a field of *size* bits (1-32) with *value*
.float *value*$_1$[, ... , *value*$_n$]	Initializes one or more 32-bit, IEEE single-precision, floating-point constants
.half *value*$_1$[, ... , *value*$_n$]	Initializes one or more 16-bit integers (halfword)
.int *value*$_1$[, ... , *value*$_n$]	Initializes one or more 32-bit integers
.long *value*$_1$[, ... , *value*$_n$]	Initializes one or more 32-bit integers
.short *value*$_1$[, ... , *value*$_n$]	Initializes one or more 16-bit integers (halfword)
.string {*expr*$_1$|"*string*$_1$"}[,... , {*expr*$_n$|"*string*$_n$"}]	Initializes one or more text strings
.ubyte *value*$_1$[, ... , *value*$_n$]	Initializes one or more successive unsigned bytes in the current section
.uchar *value*$_1$[, ... , *value*$_n$]	Initializes one or more successive unsigned bytes in the current section
.uhalf *value*$_1$[, ... , *value*$_n$]	Initializes one or more unsigned 16-bit integers (halfword)
.uint *value*$_1$[, ... , *value*$_n$]	Initializes one or more unsigned 32-bit integers
.ulong *value*$_1$[, ... , *value*$_n$]	Initializes one or more unsigned 32-bit integers
.ushort *value*$_1$[, ... , *value*$_n$]	Initializes one or more unsigned 16-bit integers (halfword)
.uword *value*$_1$[, ... , *value*$_n$]	Initializes one or more unsigned 32-bit integers
.word *value*$_1$[, ... , *value*$_n$]	Initializes one or more 32-bit integers

Example

Example 1 This example uses the .int directive to initialize words.

```
1 00000000                      .space 73h
2 00000000                      .bss   PAGE, 128    ; Reserve 128 bytes for pointer PAGE
3 00000080                      .bss   SYMPTR, 4    ; Reserve 4 bytes for pointer SYMPTR
4 00000074 E3A00056  INST:  MOV    R0, #056h
5 00000078 0000000A         .int   10, SYMPTR, -1, 35 + 'a', INST, "abc"
  0000007c 00000080-
  00000080 FFFFFFFF
  00000084 00000084
  00000088 00000074'
  0000008c 00000061
  00000090 00000062
  00000094 00000063
```

Example 2 This example shows how the .long directive initializes words. The symbol DAT1 points to the first word that is reserved.

```
1 00000000 0000ABCD  DAT1:  .long   0ABCDh, 'A' + 100h, 'g', 'o'
  00000004 00000141
  00000008 00000067
  0000000c 0000006F
2 00000010 00000000'        .long   DAT1, 0AABBCCDDh
  00000014 AABBCCDD
3 00000018           DAT2:
```

Example 3 In this example, the .word directive is used to initialize words. The symbol WORDX points to the first word that is reserved.

```
1 00000000 00000C80  WORDX:  .word   3200, 1 + 'AB', -0AFh, 'X'
  00000004 00004242
  00000008 FFFFFF51
  0000000c 00000058
```

Assembler Directives for Controlling Conditional Computations

Mnemonic and Syntax	Description
.if *condition*	Assembles code block if the *condition* is true
.else	Assembles code block if the .if *condition* is false. When using the .if construct, the .else construct is optional.
.elseif *condition*	Assembles code block if the .if *condition* is false and the .elseif condition is true. When using the .if construct, the .elseif construct is optional.
.endif	Ends .if code block
.loop [*count*]	Begins repeatable assembly of a code block; the loop count is determined by the *count*.
.break [*end condition*]	Ends .loop assembly if *end condition* is true. When using the .loop construct, the .break construct is optional.
.endloop	Ends .loop code block

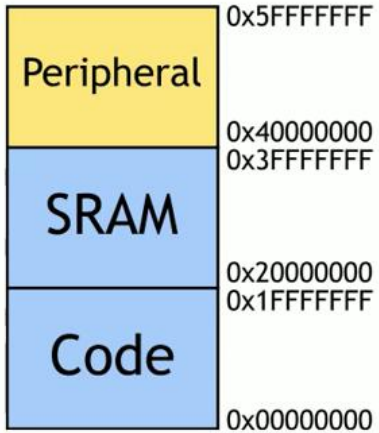
Example

This example shows conditional assembly:

Memory for Data

Memory for Code

```
1                00000001  SYM1    .set    1
2                00000002  SYM2    .set    2
3                00000003  SYM3    .set    3
4                00000004  SYM4    .set    4
5
6                          If_4:   .if     SYM4 = SYM2 * SYM2
7   00000000 04                    .byte   SYM4            ; Equal values
8                                  .else
9                                  .byte   SYM2 * SYM2     ; Unequal values
10                                 .endif
11
12                         If_5:   .if     SYM1 <= 10
13  00000001 0A                    .byte   10              ; Less than / equal
14                                 .else
15                                 .byte   SYM1            ; Greater than
16                                 .endif
17
18                         If_6:   .if     SYM3 * SYM2 != SYM4 + SYM2
19                                 .byte   SYM3 * SYM2     ; Unequal value
20                                 .else
21  00000002 08                    .byte   SYM4 + SYM4     ; Equal values
22                                 .endif
23
24                         If_7:   .if     SYM1 = SYM2
25                                 .byte   SYM1
26                                 .elseif SYM2 + SYM3 = 5
27  00000003 05                    .byte   SYM2 + SYM3
28                                 .endif
```



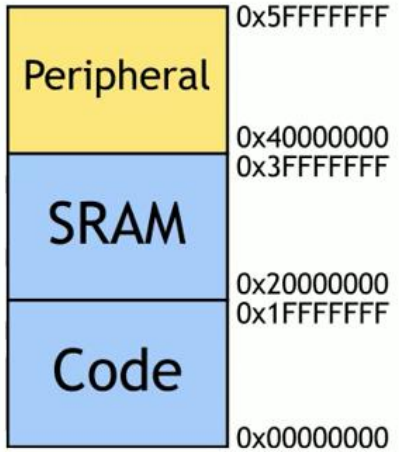
Directive for Creating Reusable Block of Codes (i.e. MACRO)

A macro definition is a series of source statements in the following format:

```
macname   .macro   [parameter1 ] [, ... , parametern ]
          model statements or macro directives
          [.mexit]
          .endm
```

macname	names the macro. You must place the name in the source statement's label field. Only the first 128 characters of a macro name are significant. The assembler places the macro name in the internal opcode table, replacing any instruction or previous macro definition with the same name.
.macro	is the directive that identifies the source statement as the first line of a macro definition. You must place .macro in the opcode field.
parameter $_1$, *parameter* $_n$	are optional substitution symbols that appear as operands for the .macro directive.
.mexit	is a directive that functions as a *goto .endm*. The .mexit directive is useful when error testing confirms that macro expansion fails and completing the rest of the macro is unnecessary.
.endm	is the directive that terminates the macro definition.

Example: x = a + b + c



Macro definition: The following code defines a macro, add3, with four parameters:

```
 1                      *
 2
 3                      *       add3
 4                      *
 5                      *       ADDRP = P1 + P2 + P3
 6
 7                      add3    .macro P1, P2, P3, ADDRP
 8
 9                              ADD     ADDRP, P1, P2
10                              ADD     ADDRP, ADDRP, P3
11                              .endm
```

Create a macro

Macro call: The following code calls the add3 macro with four arguments:

```
12
13 00000000                    add3 R1, R2, R3, R0
```

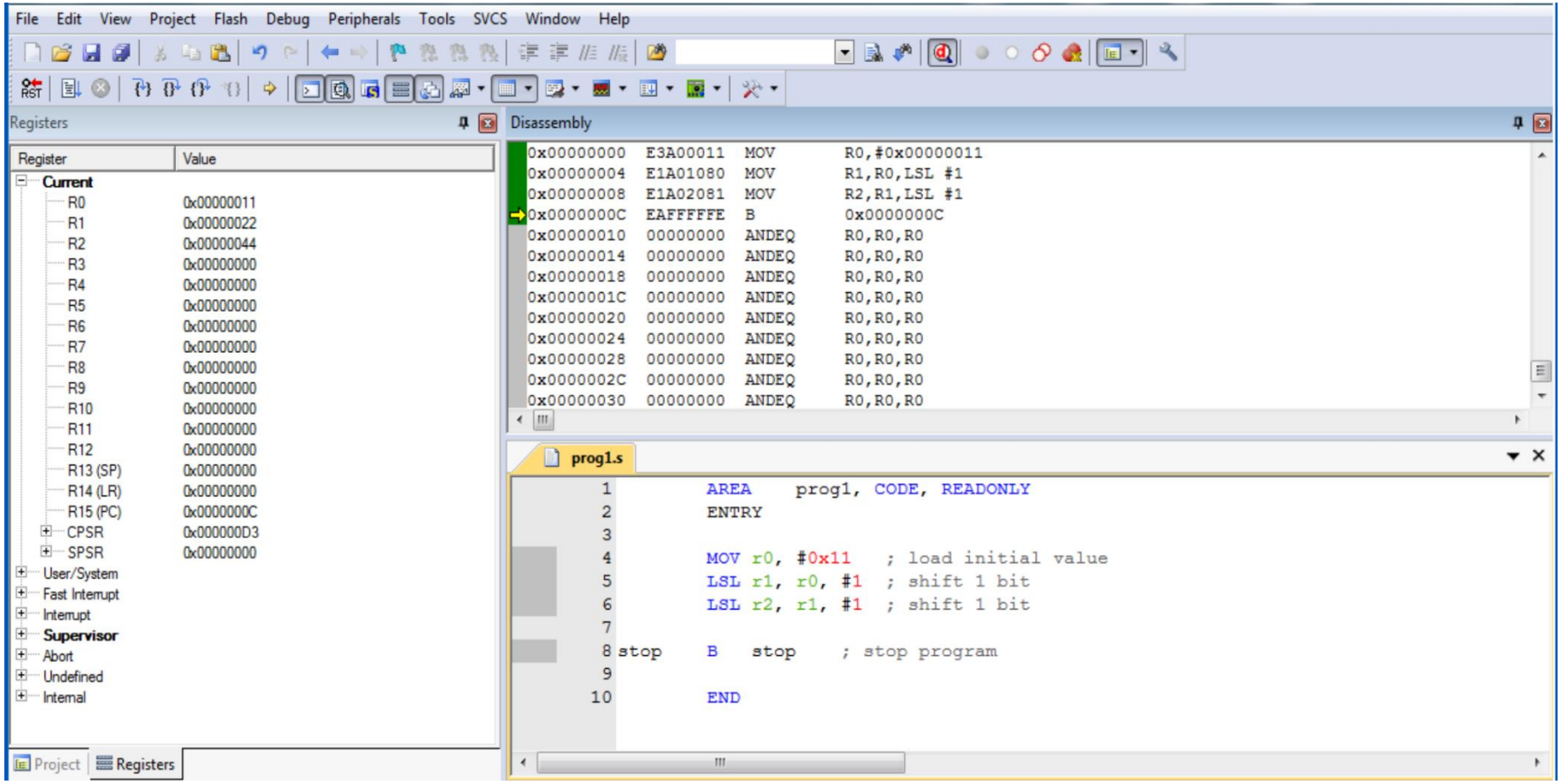
Invoke or call a macro

Macro expansion: The following code shows the substitution of the macro definition for the macro call. The assembler substitutes R1, R2, R3, and R0 for the P1, P2, P3, and ADDRP parameters of add3.

```
1
1       00000000 E0810002              ADD     R0, R1, R2
1       00000004 E0800003              ADD     R0, R0, R3
```

Equivalent instructions

Good News: Integrated Development Environment (Keil)

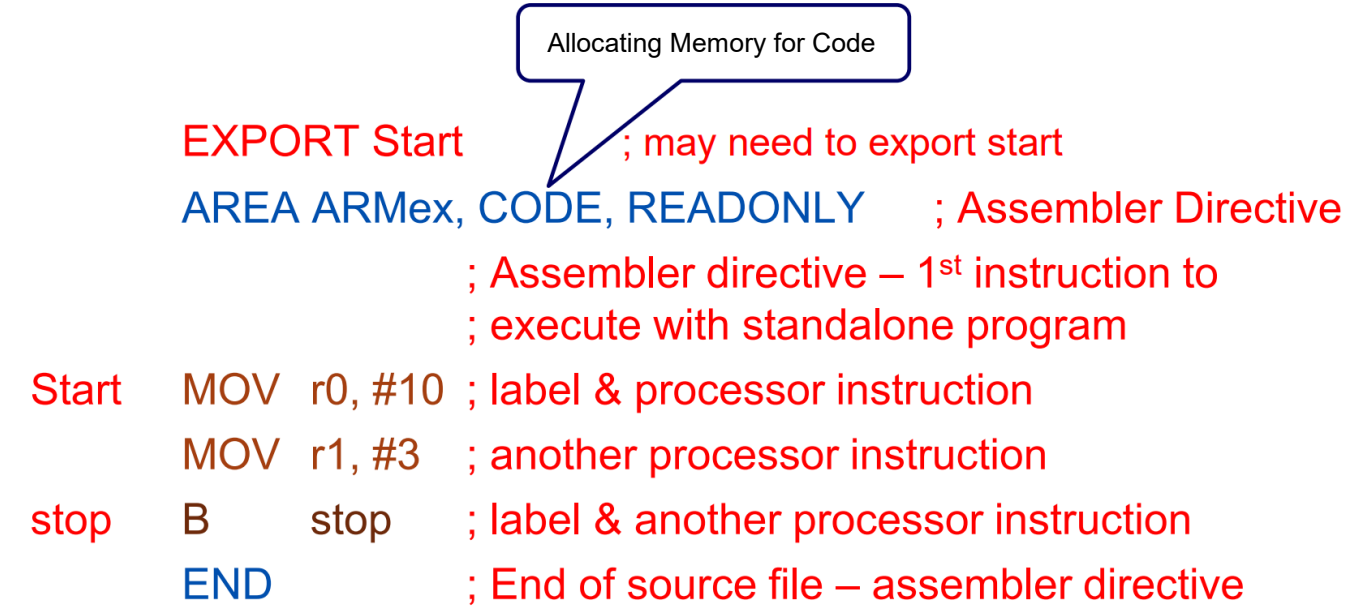


Keil Directives for Allocating Memory to House Instructions/Data

- Format:

  { label } {instruction | directive | pseudo-instruction} { ; comment }

- Example:



Keil Directive for Reusable Instructions (i.e. MACRO and MEND)

- The MACRO directive marks the start of the definition of a macro. Macro expansion terminates at the MEND directive.

- Format:
  - MACRO
  - {$label} macroname{$cond} {$parameter{,$parameter}...}
  - ; code
  - MEND

- Example:

```
MACRO
        ; mcro definition
        ; vara=8 * (varb + varc + 6)

$Label_1 AddMul $vara, $varab, $varac

$Label_1
        Add $vara, $varab, $varac
        Add $vara, $vara  , #6
        LSL  $vara, $vara  , #3
        MEND
```

Invocation of MACRO

```
MACRO
        ; mcro definition
        ; vara=8 * (varb + varc + 6)

$Label_1 AddMul $vara, $varab, $varac

$Label_1
        Add $vara, $varab, $varac
        Add $vara, $vara , #6
        LSL  $vara, $vara , #3
        MEND
```

- Invoke Macro

  AddMul r0, r1, r2

- Resultant code after

  ADD r0, r1, r2

  ADD r0, r1, #6

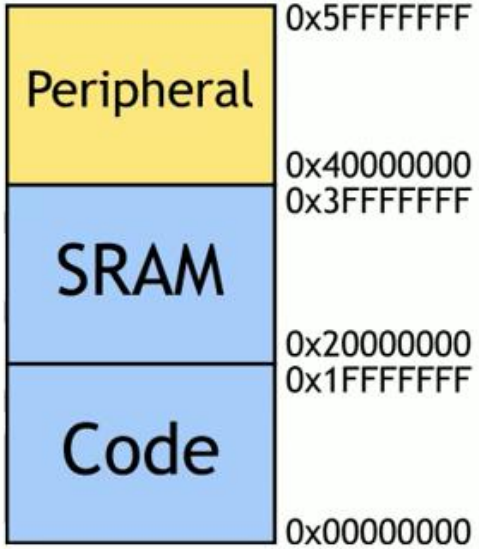
  LSL r0, r1, #3

More Example

```
 ; macro definition
                MACRO                   ; start macro definition
$label          xmac    $p1,$p2
                ; code
$label.loop1    ; code
                ; code
                BGE     $label.loop1
$label.loop2    ; code
                BL      $p1
                BGT     $label.loop2
                ; code
                ADR     $p2
                ; code
                MEND                    ; end macro definition
 ; macro invocation
abc             xmac    subr1,de        ; invoke macro
                ; code                  ; this is what is
abcloop1        ; code                  ; is produced when
                ; code                  ; the xmac macro is
                BGE     abcloop1        ; expanded
abcloop2        ; code
                BL      subr1
                BGT     abcloop2
                ; code
                ADR     de
                ; code
```

Allocation of Constants or Parameters in Keil

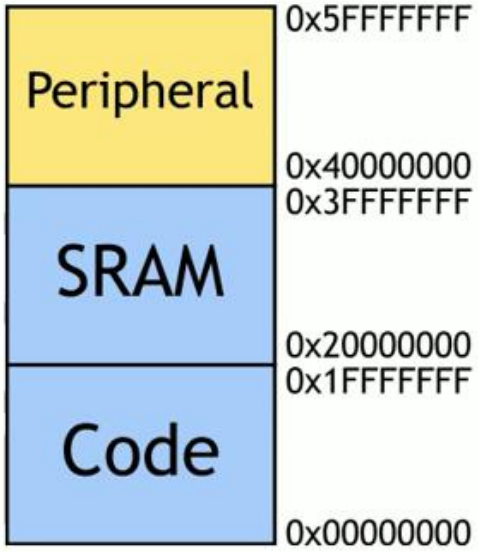
- Format with SETS Directive:
  - {name} SETS value
- Example:
  - mytext SETS “This is my text” ;
- Format with EQU Directive:
  - {name} EQU value {, type}
- Example:
  - temperature EQU 25.6 ;



Formats of Numbers in Keil

Number formats

- 123 : decimal number
- 0x3f : hexadecimal number
- n_xxx n : base (2 to 9) , xxx : number
  - 8_12 : octal number 12
- 'A' : single character constant
- "string" : string constants



Frequently Used Keil's Directives

Assembler User Guide: Directives Reference (keil.com)

Keil Directive	Uses
AREA	Defines a block of code or data
RN	Can be used to associate a register with a name
EQU	Equates a symbol to a numeric constant
ENTRY	Declares an entry point to your program
DCB, DCW, DCD	Allocates memory and specifies initial runtime contents
ALIGN	Aligns data or code to a particular memory boundary
SPACE	Reserves a zeroed block of memory of a particular size
LTORG	Assigns the starting point of a literal pool
END	Designates the end of a source file

The END directive informs the assembler that it has reached the end of a source file.

AREA

- A memory has a physical address
- A memory has a conceptual label

- The `AREA` directive instructs the assembler to assemble a new code or data section.

AREA *sectionname {,attr} {, attr}…*

AREA Name, CODE, READONLY

Valid Section Attributes (Keil Tools)

ALIGN = *expr*	This aligns a section on a 2^{expr}-byte boundary (note that this is different from the ALIGN directive); e.g., if *expr* = 10, then the section is aligned to a 1KB boundary.
CODE	The section is machine code (READONLY is the default)
DATA	The section is data (READWRITE is the default)
READONLY	The section can be placed in read-only memory (default for sections of CODE)
READWRITE	The section can be placed in read-write memory (default for sections of DATA)

RN

- A memory has a physical address
- A memory has a conceptual label

- The `RN` directive defines a name for a specified register.

- Format:

name RN expr

- *name – name to assign to the register*
- *expr – takes values between 0 to 15*

- Example:

```
coeff1      RN      8       ; coefficient 1
coeff2      RN      9       ; coefficient 2
dest        RN      0       ; register 0 holds the pointer to
                            ; destination matrix
```

EQU

- A memory has a physical address
- A memory has a conceptual label

- The `EQU` directive gives a symbolic name to a numeric constant, a register-relative value or a PC-relative value.

- Format: Name EQU expr{, type}
  - *name* is the symbolic name to assign to the value,
  - *expr is an address or integer constant*
  - *Type is optional and can be:* ARM, THUMB, CODE16, CODE32, DATA

- Example:

```
SRAM_BASE   EQU    0x04000000   ; assigns SRAM a base address
abc         EQU    2            ; assigns the value 2 to the symbol abc
xyz         EQU    label+8      ; assigns the address (label+8)
                                ; to the symbol xyz
fiq         EQU    0x1C, CODE32 ; assigns the absolute address
                                ; 0x1C to the symbol fiq, and marks it
                                ; as code
```

ENTRY

- A memory has a physical address
- A memory has a conceptual label

- The `ENTRY` directive declares an entry point to a program.

A program must have an entry point. You can specify an entry point in the following ways:
•Using the `ENTRY` directive in assembly language source code.
•Providing a `main()` function in C or C++ source code.

- Example:

```
        AREA   ARMex, CODE, READONLY
        ENTRY       ; Entry point for the application.
        EXPORT ep1 ; Export the symbol so the linker can find it
ep1                 ; in the object file.
        ; code
        END
```

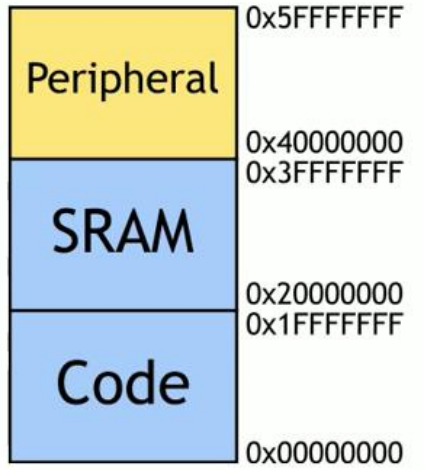
DCB (i.e. Declare 8-bit bytes)

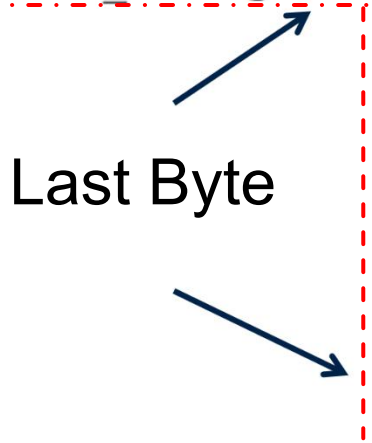
- The `DCB` directive allocates one or more bytes of memory, and defines the initial runtime contents of the memory.

- Format:
  - {label} DCB expression {, expression} …

- Example:

```
C_string DCB "C_string",0
```





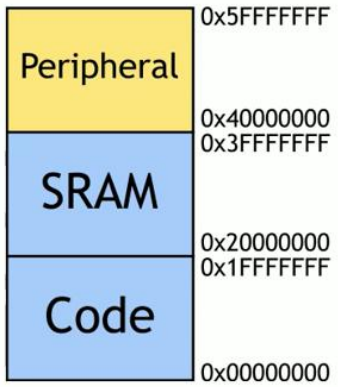
Address		ASCII equivalent
0x4000	43	C
0x4001	5F	_
0x4002	73	s
0x4003	74	t
0x4004	72	r
0x4005	69	i
0x4006	6E	n
0x4007	67	g
0x4008	00	

DCW (i.e. Declare 16-bit words)

- The `DCW` directive allocates one or more halfwords of memory, aligned on two-byte boundaries, and defines the initial runtime contents of the memory. `DCWU` is the same, except that the memory alignment is arbitrary.

- Format:
  - {label} DCW{U} expression {, expression} …

- Example:



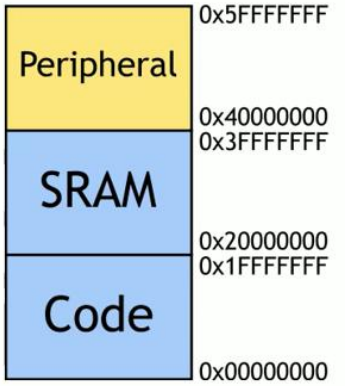

```
data    DCW     -225,2*number   ; number must already be defined
        DCWU    number+4
```

DCD (i.e. Declare 32-bit words)

- The `DCD` directive allocates one or more words of memory, aligned on four-byte boundaries, and defines the initial runtime contents of the memory. `DCDU` is the same, except that the memory alignment is arbitrary.

- Format:
  - {label} DCD{U} expression {, expression} …

- Example:




```
data1   DCD     1,5,20          ; Defines 3 words containing
                                ; decimal values 1, 5, and 20
data2   DCD     mem06 + 4       ; Defines 1 word containing 4 +
                                ; the address of the label mem06
        AREA    MyData, DATA, READWRITE
        DCB     255             ; Now misaligned ...
data3   DCDU    1,5,20          ; Defines 3 words containing
                                ; 1, 5 and 20, not word aligned
```

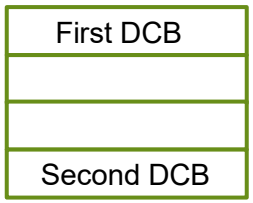
ALIGN (i.e. Change offset of alignment)

- The `ALIGN` directive aligns the current location to a specified boundary.

- Format: ALIGN { expr { , offset } }
  expr – a numeric expression evaluating to any power of 2^0 to 2^{31}
  Current location is defined by: offset + n * *expr. If not defined , location is the next word.*

- Example

```
        AREA    cacheable, CODE, ALIGN=3
rout1   ; code          ; aligned on 8-byte boundary
        ; code
        MOV     pc,lr   ; aligned only on 4-byte boundary
        ALIGN   8       ; now aligned on 8-byte boundary
rout2   ; code
```




```
AREA    OffsetExample, CODE
DCB     1       ; This example places the two bytes in the first
ALIGN   4,3     ; and fourth bytes of the same word.
DCB     1       ; The second DCB is offset by 3 bytes from the
                ; first DCB.
```

ALIGN 4, 3 means: data size = 4 bytes, offset between two consecutive data = 3 bytes

SPACE and FILL (i.e. allocate bytes)

- The `SPACE` directive reserves a zeroed block of memory. The `FILL` directive reserves a block of memory to fill with a given value.

- Format:
  - {label} SPACE expression
  - {label} FILL expression {, value {, value-size in terms of byte}}

- Example:

```
        AREA    MyData, DATA, READWRITE
data1   SPACE   255         ; defines 255 bytes of zeroed store
data2   FILL    50,0xAB,1 ; defines 50 bytes containing 0xAB
```

LTORG (i.e. Lookup-Table Organized)

(Literals mean "fixed and unchanging values")

- The LTORG directive instructs the assembler to assemble the current literal pool immediately. The assembler assembles the current literal pool at the end of every code section. The end of a code section is determined by the AREA directive at the beginning of the following section, or the end of the assembly.

In computer science, and specifically in compiler and assembler design, a literal pool is a lookup table used to hold literals during assembly and execution.

The assembler uses **literal pools** to store some constant data in code sections.

Working memory of a function

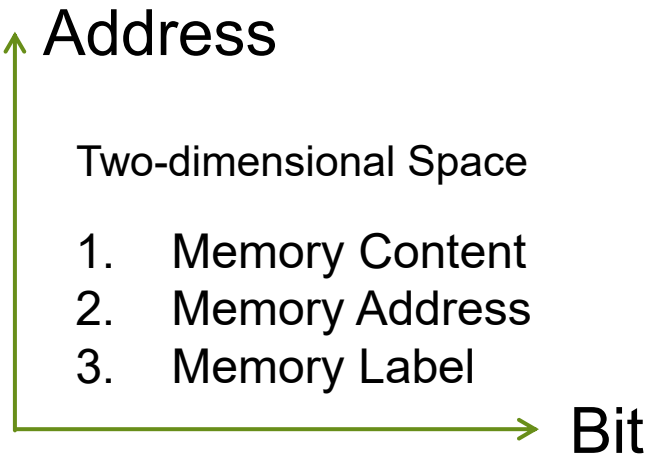
- Example:

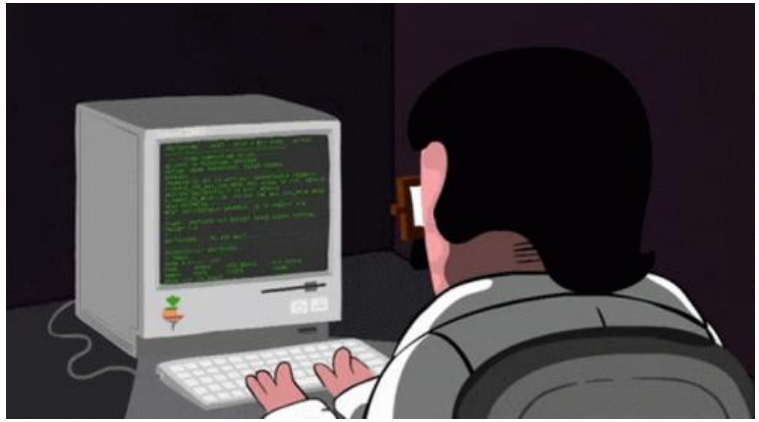
```
        AREA    Example, CODE, READONLY
start   BL      func1
func1                           ; function body
        ; code
        LDR     r1,=0x55555555  ; => LDR R1, [pc, #offset to Literal Pool 1]
        ; code
        MOV     pc,lr           ; end function
        LTORG                   ; Literal Pool 1 contains literal &55555555.
data    SPACE   4200            ; Clears 4200 bytes of memory starting at current location.
        END                     ; Default literal pool is empty.
```

Other Operations Offered by Keil

	Keil Tools
A modulo B	A:MOD:B
Rotate A left by B bits	A:ROL:B
Rotate A right by B bits	A:ROR:B
Shift A left by B bits	A:SHL:B or A << B
Shift A right by B bits	A:SHR:B or A >> B
Add A to B	A + B
Subtract B from A	A – B
Bitwise AND of A and B	A:AND:B
Bitwise Exclusive OR of A and B	A:EOR:B
Bitwise OR of A and B	A:OR:B

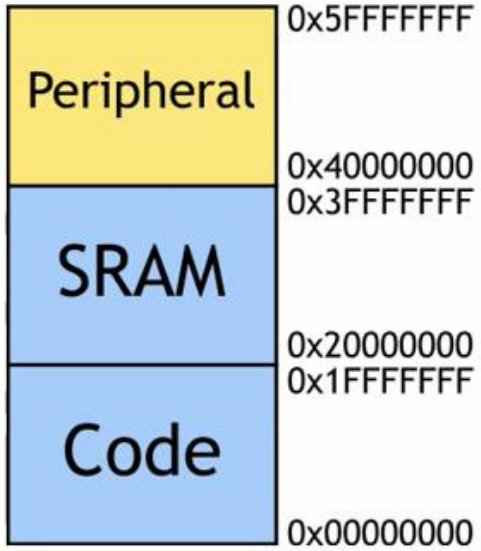
Summary



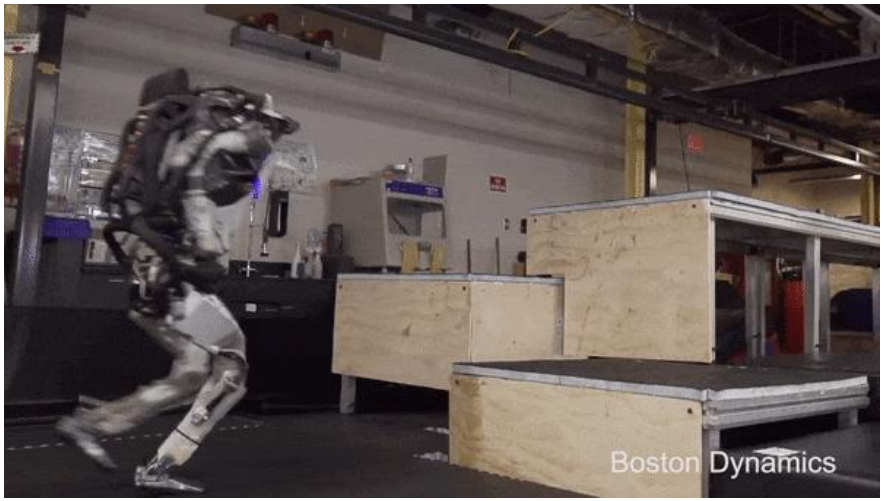


You = Director of Program
Program = {Instructions}

- Nature of Programming
- Planning of Computations in Programming
- Allocation of Memory in Programming
- ARM Programming Tools



Summary of Module 1



- Robot Systems
- Robot's Mechanical Systems
- Robot's Control Systems
  - Controllers
  - Actuators
  - Sensors
- Robot's Programming Systems

What to design?

- The best systems in the universe are static systems
- Most systems in the universe are dynamic systems
- Our goal is to make dynamic systems to be closer to static systems

"Ask not what your country can do for you – ask what you can do for your country," - John F. Kennedy

"Do not think that you are needy – think that you are needed in the world", - Manis Friedman

"Study will make you knowledgeable, resourceful, and hence more needed", - Xie Ming

Thank You for Listening!