General Purpose IO (Memory mapped I/O <most common in today's CPU>) (Time mapped I/O <use special CPU instructions>)
Assembly <help understand how processor work> <runs faster than high level language> <Hardware specific code> <Cost sensitive applications>
Memory <values stored at each location can represent either program data or program instructions>
Computer architecture <Von Neumann – Instructions and data are stored in same memory – cheaper – can't access read/write same time>
<Harvard – more costly – can access read/write same time – Data and instructions are stored into separate memories>
Level of code< C program compile to assembly program<Textual instructions> then assemble to machine program<binary bits>>
Processor Registers <fastest way to read/write, R0 – R12: general purpose, R13: MSP or PSP, R14: Link Register, R15: program counter. 16 total>
Program counter <place to store memory address of next instruction fetched by the memory>
Pipelining <allows hardware resources to be fully utilized, one 32 bit or two 16 bit can be fetched> <4 bytes are fetched from instruction memory>
Dynamically allocated data <Heap memory, grows up>; Program instructions<Instruction memory>; Local variables <Stack memory, grows down>
Initialized global variables<Initialized data segment>; Uninitialized global variables <zero-initialized data segment>
Instructions are executed in ALU; Instruction's operands and results are collected from register memory; Decoded in CPU; Instructions are flash.
Register memory has the fastest access time of any type of memory. 0x2 is the starting address of data memory;0x08 is start of instruction memory.
8 ports in general; 16 pins for each port; Four functions <Digital input, Digital output, Analog functions (D2A, A2D), Alternate functions(PWM)>
GPIO Input <Pull up: pull resistor to ground> <Pull down: pull resistor to ground> <Inputs can be set, high, low, high impedance>
GPIO Output <Push-Pull> <Push: Output goes high, connect to Vcc> <Pull: output goes low, connect to ground> <low power, stable>
<Open-Drain> <Open: Output goes high, high impedance to external circuit> <Drain: Outputs goes low, drain current from external circuit>
No resistor for Open drain mode <high power consumption, slower, but can switch higher or lower voltage>
Output speed: Higher speed increase EMI noise and power consumption, can be configured based on peripheral speed
Slew rate = $\max(\Delta V \div \Delta t)$; Schmitt Trigger: Takes an analog signal and cleans up the digital signal that it produces
ODR<Drives pin output>; IDR<Stores and receives input>; PUPDR<set input pin mode>; OTYPER<set output pin mode>.
MODER<sets pin digital input or output>; OSPEED<Sets output slew rate>; Peripheral memory are memory mapped.
I/O Debouncing <hardware: use a simple low pass filter> <software: read signal after a delay, reading periodically and use a counter as a filter >

Timers <Input capture; Output compares; PWM; One-pulse mode output> external clock is more preferred because high accuracy
Prescaler <A constant that can slow down the time counter> <larger prescaler reduces timer's resolution, decrease the chance of overflow and underflow which improves the energy efficiency> $f_{clock\_cnt} = \frac{f_{clock\_psc}}{Presclar+1}$
Multi-Channel Outputs <Based one 1 signal, it is possible for 4 registers to do different functions at the same time>
Edge-aligned mode <Up counting(increase from smallest value): overflow> <Down counting: underflow> Period = (1+ARR)*Clock Period
<Center aligned(increase then decrease): overflow & underflow> Period = 2*ARR*Clock Period; Duty Cycle =1 - CCR/(ARR)
PWM Mode <Low count true; high if counter < CCR; low if counter >= CCR> Duty Cycle = CCR/(ARR+1)
<high count true; vice versa of low count> Duty Cycle = 1 - CCR/(ARR+1)
Input capture <monitor both rising and falling edge>→ <External signal + edge detector>+<clock signal + timer counter> = captured value
Ultrasonic Distance Sensor: Distance(cm) = Pulse Width/58; Distance(inch) = Pulse Width/148.

Interrupts <Polling & Interrupt> <Polling wastes CPU cycles and occurs periodically> <Interrupt doesn't waste much CPU and occurs at any time>
NVIC <Nested Vectored Interrupt Controller> <Enables and disable a specific interrupt>
Process of the interrupt <Stop current code> <Service the interrupt request> <Resume previous code>
Automatic stacking & unstacking <stacking: hardware automatically pushes eight register into stack before interrupt handler starts>
<unstacking: Hardware automictically pops these eight registers off the stack>
MSP & PSP <MSP → main stack pointer, reverse for kernel type operations and other operating system>
<PSP → Process stack pointer, used for user programs, designed for use by kernel processes>
Link Register (LR) <When software calls a subroutine, LR holds the returning address>

I2C: Inter-Integrated Circuit <Design for low cost, medium data rate applications> <7,10,16 bits>
Characteristics <Serial, byte oriented, multi master and slave, two bidirectional open-drain lines, plus ground> <SDA for data, SCL for clock>
Start condition <High to low transition on SDA when SCL is high> <Address and data bytes are sent most significant bit first>
Stop condition <Low to high transition on SDA when SCL is low>
Master generates the clock signal and sends it to slave during data transfer. <Data on SDA can be changed only when SCL is low>
Master device sends out a start signal → sends a header packet <contains the address of slave device> <7,8,10 bit I2C slave addressing>
Working modes <Master-sender> <Master -receiver> <Slave sender> <Slave receiver>
Data packet <First 7 bits contain the address of the thing that it's trying to communicate to> <8th address indicates if it's read(1) or write(0)>
<9th bit is the acknowledgement from slave device, 1 for no acknowledgment, 0 for yes>
Shift register <Individual flip flops, data can be loaded in parallel> <Two ways in, two ways out> <Used for queuing data, send data in sequence>

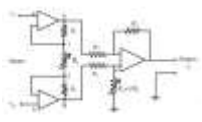
Real Time Clock <Lower power consumption> <Separate power source> <Accurate> <Run independently from processor>
Unix Epoch Time: 00:00:00 UTC, Thursday, 1 January 1970. <correct to seconds>
Crystal Inaccuracy: PPM = (10^{-6}) PPM = 10^6*(Actual frequency – theoretical frequency)/(theoretical frequency)
Trade-off <Low frequency crystals → Large physical size → low current drain> <High frequency crystals → large current drain → smaller size>
Frequency setting $f_{1HZ} = f_{RTC}$/((Asynch_prescaler + 1)*(Synch_prescaler + 1))

Design Consideration <Optimizing design metrics> <Design goal: Construct an implementation with desired functionality>
Common characteristics of embedded systems <Single functioned> <Tightly constrained> <Reactive and real time> <Trade-off happens>
Common metrics: <Unit cost><NRE cost><Size><Performance> <Power> <Flexibility> <Time to prototype> <Time to market> <Maintainability>
Losses due to delayed market entry <The difference between the on-time and delayed triangle areas> %loss = (D*(3W-D)/(2*W^2)*100%
NRE and unit cost Total cost = NRE cost + unit cost*# of units Per product cost = total cost/# of units = (NRE cost/#of units)+unit cost
The performance design metric <widely used measure of system> <Latency> <Throughput> <speedup B over A = B's performance/A's>
Three key technologies <Processor technology> <IC technology> <Design technology>
Processor technology<general purpose→program memory, general data path with large register, low time-to-market and NRE cost, high flexibility>
<Single purpose processor → contains only the component needed to execute a single program, no program memory, fast, low power, small size>
<Application specific processors→Program memory, optimized data path, special functional units, some flexibility, good performance, size, power>

Digital and analog conversion <digital(discrete) are either on or off> <Analog(continuous) may have any voltage level>
Power dissipation problem <use standard, low power digital electronics to read potentiometer> <have a controller to decide voltage>
Signal processing <ADC is one to many; DAC is many to one> <Sample and hold ADC system acquires the current voltage level at some specific frequency → that value is held until the next sample → smaller sampling period, the more closely the resulting digitized output signal resembles the input signal> <Nyquist Theorem unless sample at at least twice the highest frequency of an analog signal, you can't completely reconstruct it>
Quantization <process of mapping continuous infinite values to a smaller set of discrete finite values> $V_Q = V_{ref}/2^n$
ADC technologies <Sigma-delta, reads signal change, adds it to a running total, high resolution, low frequency, used in cellphones for voice, small> <SAR, performs binary search over possible voltage range, low power, medium frequency, ideal for device with tiny batteries>
<Flash ADC, uses an array of comparators, large size, large power, low resolution, fastest, used in oscilloscopes>
DAC technologies <PWM & Filtering: doesn't require DAC module, not suitable for high frequency signals>
<Resistor ladder: Digital bits drive voltage directly through a network of resistors, requires fewer resistors, 2n resisters>
<Resistor string: Digital bits drive transistors, which pick a voltage from a resistor string, require 2^n resistors, simplest technique>

Stepper motors <used for rotary motion, many kinds: AC, DC, servo, stepper<DC motors whose position can be changed in discrete steps>>
Brushless DC Motor with windings: By controlling voltage across poles, we control the magnetic flux and the orientation of rotor
More windings, more stable. Full step control <pros: double the torque> <Cons: power consumption>
Improving resolution <Increase number of phases on stator> <Increase number of magnets on rotor> <Half-stepping> <Micro stepping>
Angular resolution = 360/(# of steps) Angular resolution = 360/2*(# of poles) for 2 phase motors
Half stepping <Double the resolution with no change to control wiring> <Less torque on the motor> <# of wires increase>
Magnetic reluctance <resistance in an electrical circuit> <smaller air gap, lower the reluctance> Angular res = 360/(# of magnets*#of teeth)
Control circuit <motor nee more current than microprocessor><output from the driver is connected to the motor>
H-bridge circuit <protection circuit> <capacitor should be placed in parallel>
Micro stepping <Two phases are used with a variable amount of power>

OpAmps <Operational amplifiers> <Impedance matching: maximum power problem<same resistance>>
Impedance matching amplifiers <high input impedance> <low output impedance> <Unity gain> <piezoelectric sensor>
Operational Amplifiers <External power supply> <high gain $10^5 – 10^9$> <input impedance > 1MΩ> <output impedance < 100 Ω>
Principle <Voltage at input leads is equal> <Current through either input lead is zero>
Saturation <Op amp is linear only in small region of input voltage> <Outside linear is saturated at a value>
Stability <Bandwidth is high > 10KHz> <K changes over time due to temperature, humidity, age> <assumed frequency independent>
Feedback <mitigate effects of stability> <utility gain> <$V_{in} = V_+ = V_- = V_{out}$>



Voltage amplifier: $V_o = (1+R_f/R)*V_i$ <satiable for piezoelectric > Current amplifier: $I_o = (1+R_f/R)*I$
Single ended <One input is grounded> <suffers from ground loop noise> <common mode noise>
Differential or double ended <neither input is grounded> <eliminate all noise or noise cancel out> $V_o = R_f/R *(V_{i2} - V_{i1})$
CMRR = $20*\log_{10}(K/|Kcm|)$ or CMRR = $K*V_{CM}/V_{OCM}$ Vcm is common mode voltage and Vocm is common mode output voltage <K = 0 ideally>
Basic instrumentation amplifier $V_0 = (V_{i2} - V_{i1}) * \left(1 + \frac{2*R_1}{R_2}\right) * \frac{R_4}{R_3}$
Floating signal sources <not referenced to ground> <Digital multimeters, batteries, thermocouples, transformers> <can be referenced to ground by connecting one terminal to ground> <ground loop results from the potential differences between ground at A and B>
Differential mode <input signal low level < 1V> <Leads are greater than 3 m/10ft> <Leads travel through noisy environment> <require ground reference point> <requires 2 channels per> Referenced Single ended <input signal high level > 1V> <Input signal is floating> <Leads are less than 3m/10ft> non-referenced single ended <Input signal high level> <Leads less than 3m/10ft> <Input signal is referenced to ground>

Digital Design <Logic functions, variables, logic networks, logic circuit, sequential circuit, FSM>
NMOS <Use n type(-) MOSFET to implement logic gates and other digital circuits> <nMOS transistor receives a non-negligible voltage, connection from the source to the drain acts as a wire; When receives 0 volt, connection will be broken (open circuit)> PMOS acts exact opposite
<L(x) = x> → <a logic function and x is an input variable> <AND, OR, NOT are basic building blocks of the logic system>
Process to design a logic circuit <Express the requirement as truth table> <Obtain Boolean expression from table (POS or SOP)> <Minimize or optimize the expression> <Implement the minimized expression using suitable gates>
Multiplexer <Select one of the several inputs and give one output> <output of a combinational circuit>
Sequential circuits <Implemented by flip flops> <N bit registers, counters, shifters>
FSM <An abstract machine that can be in exactly one of a finite number of states in any given time>
Moore output <Output derives from pure state elements> <reduce the length of critical path> <use more registers in general>
Mealey output <Deriving things from state elements & results of circuitry> <Requires more combinational circuitry> <Less size> <Preferred>
Sequential logic design <problem statement> <state diagram> <Implementation model> <State table>
Control unit <which memory gets loaded> <which operations are performed in what order>
Creating the data path <Create a register for any declared variable> <Create functional unit for each arithmetic operation> <Connect the ports, registers, and functional units> <Create unique identifier>
Optimizing data path <sharing of functional units> <multi-functional units> Optimize FSM <state encoding> <state minimization>

General Purpose Processors <refer design consideration><Control Unit and data path><Data path is general><Control unit doesn't store algorithm>
Data path operations <Load, ALU operation, store> Control Unit <Configures the data path operations>
Control unit sub operations <Fetch> <Decode> <Fetch operands → move data from memory to data path register> <Execute → move data to ALU> <Store results → Write data from register to memory>
Architectural Considerations <N-bit processor> <PC size determines address space> <Clock frequency, memory access is often the longest>
Superscalar and VLIW Architectures <Performance can be improved by: Faster clock, pipelining, multiple ALU>
Two memory architectures <Princeton: Fewer memory wires> <Harvard: Simultaneous program and data memory access>
Cache memory <memory access may be slow> <Cache is small but fast memory close to processor> <holds copy of part of memory>