EET-2242: C AND ASM PROGRAMMING WITH EMBEDDED APPLICATIONS

Cuyahoga Community College

Viewing: EET-2242 : C and ASM Programming with Embedded Applications

Board of Trustees:
March 2021

Academic Term:
Fall 2021

Subject Code
EET - Electrical/Electronic Engineer

Course Number:
2242

Title:
C and ASM Programming with Embedded Applications

Catalog Description:
Introduces microprocessor and microcontroller internal and external hardware components. Assembly language (ASM) programing is introduced to illustrate the internal working of a microcontroller. C programming language is used for embedded and non-embedded environments.

Credit Hour(s):
3

Lecture Hour(s):
2

Lab Hour(s):
2

Requisites

Prerequisite and Corequisite
EET-1241 Digital Fundamentals, or departmental approval.

Outcomes

Course Outcome(s):
Explain the internal architecture of a microprocessor and microcontroller. (OET-004, 1, 3, 4, 5, 6, 9)

Objective(s):
1. Using the manufacturer's data sheet, find and explain the purpose of the Arithmetic and Logic Unit (ALU).
2. Identify internal special functions registers and memory that includes timers, counters, flash memory, writable memory (Random Access Memory), etc., of a microcontroller and the bus structure of a microprocessor that would access the equivalent of external special functions registers and memory.
3. Explain bus timing from the standpoint of having a data source and data sink and also explain why there can be only one bus master.
4. Using a manufacture's data sheet, explain bus timing and correlate the timing diagram to the minimum and maximum timing specifications.
5. Explain how the internal bus structure of a microcontroller and the external bus structure of a microprocessor transfer data.
6. Find (manufacturer's data sheet) and explain the three types of buses; address, data and control.
7. Using a manufacturer's data sheet, locate the bus interface for the Input/Output (I/O) ports (also referred to as General purpose Input/Output GPIO) internally in a microcontroller and externally for a microprocessor.
8. Explain Direct Memory Access (DMA) for the transfer of data and also explain why DMA is faster than interrupt processing.

Course Outcome(s):
Identify and explain the principal components of a microcontroller. (OET-004, 1)

Objective(s):
1. Determine the program address space of a microcontroller or microprocessor.
2. Determine the writable address space of a microcontroller or microprocessor.
3. Explain memory-mapped Input/Output and show its address space.
3. List the special functions registers and explain what they do.
4. List several arithmetic instructions.
5. List several logic instructions.

Course Outcome(s):
Demonstrate the Integrated Development Environment (IDE or a similar tool) using various components. (OET-004: 2)

Objective(s):
1. In the IDE create a new project and specify the family and part that the project will use.
2. Create a source program in C and add it to the project.
3. Set the IDE's communication system to transfer data to/from the target microcontroller.
4. Compile a program.
5. Correct syntax errors that the compiler identifies.
6. Link and download a program to the target microcontroller.
7. Demonstrate and use the IDE's debugger functions that include breakpoints, single step, register contents, variable contents, etc.
8. If present, find and correct run time errors.

Course Outcome(s):
Demonstrate the three principal components of a programming language, in this case the C programming language.

Objective(s):
1. Write and test programs using selector statements that include if, if-else and switch statements.
2. Write and test programs that use iteration statements: while, do-while and for statements.
3. Write and test programs that use process statements and explain how their execution is controlled by selector and loop statements.

Course Outcome(s):
Write a program that has a visible output (scope, LCD, port LEDs, etc.) that performs some useful function. (OET-004, 10)

Objective(s):
1. Indent C statements as required by the concept of structured programming.
2. Comment all principal components of the program.
3. Use debug tools to assist in writing the program and displaying the result(s).
4. Demonstrate the result(s) of the program.
5. Write a lab report using formal lab report guidelines.

Course Outcome(s):
Write simple programs, view and explain key parts of the ASM (Assembly Language) code produced by a C compiler. (OET-004: 2)

Objective(s):
1. Write a small C program with loop, selector, and process statements.
2. View the ASM output file from the C compiler and identify the comments, C statements, and ASM instruction that result.
3. Find conditional statements in a C compiler's ASM output file and the jump instructions that result.
4. Find iterational statements in a C compiler's ASM output file and the jump instructions that result.

Course Outcome(s):

Using embedded C and assembly language, demonstrate the use of GPIO (Input/Output, I/O), read data from ports and write data to ports. (OET-004: OET: 7, 8)

Objective(s):

1. Configure GPIO (General Purpose Input/Output) ports for input and demonstrate how data is transferred from a port to a variable and prove the variable has the correct data.
2. Configure GPIO ports for output and transfer data from a variable to an output port and prove the correct data is on the port.
3. Specify the minimum and maximum permissible voltages for a logic 0 and a logic 1 using positive logic for the microcontroller's family.
4. Using the microcontroller's data specification, determine the maximum source and sink currents.

Course Outcome(s):

Create design documents.

Objective(s):

1. Create a flow chart, state table or pseudo code for a program in the design stage.

Course Outcome(s):

Demonstrate C programming in a non-embedded application.

Objective(s):

1. Write various programs using selector, looping and control statement and generate a printout and verify that the problem statement has been solved.
2. Read data from a disc file.
3. Read data from the keyboard for different types of variables and perform an operation on the data that will solve a problem statement.
4. Solve numerical methods problems that are associated with electronics or math.

Course Outcome(s):

Using embedded C, demonstrate Analog-to-Digital conversion (ADC) and Digital-to-Analog conversion (DAC).

Objective(s):

1. Configure the ADC's special function register and connect a voltage of known value and program to display the result in base 10.
2. Connect a voltage of known value, read the ADC's output and write the code to display the result in base 16.
3. Connect a voltage of known value, read the ADC's output and write the code to display the result in base 2.
4. Write a program that generates a specified periodic waveform using a DAC.

Course Outcome(s):

Write/modify a program that uses interrupts. (OET-004: 7, 8)

Objective(s):

1. Prove that an Interrupt Service Routine (ISR) is not synchronous with a program's instruction or statement execution.
2. Setup the ISR's location (address) in the interrupt vector table.
3. Set the priority level of the interrupt.
4. Enable specific and global interrupts.
5. Demonstrate the ISR is in-fact executed when an interrupt occurs by placing a break point in the ISR's code.
6. Demonstrate the result of the execution of an ISR on input/output (GPIO) port data.

Methods of Evaluation:

1. Examinations
2. Quizzes

3. Lab reports
4. Homework assignments

Course Content Outline:

A. Microcontrollers versus microprocessors
B. Code address space
C. Data address space
D. Memory-mapped I/O
E. Special function registers
F. Arithmetic and logic units
G. Flags
H. Independent Development Environment (IDE)
I. ASM (Assembly Language) and C source files
J. Assembling and compiling
K. Debugging
L. C language constructs
M. Selector statements
N. Process statements
O. Iteration control statements
L. Program design documents
M. ASM output from C compiler
N. Design documents
1. Flow charts
2. Pseudo code
3. State tables

O. General Purpose Input/Output (GPIO)
P. Special functions that may include:
1. Timers
2. Counters
3. Inter Integrate Circuits (I2C)
4. Serial Peripheral Interconnect (SPI)
5. Universal Asynchronous Receiver/Transmitter (UART)
6. Analog-to-Digital (ADC)
7. Digital-to-Analog (DAC)
8. Convertors and Interrupt Service Routines (ISR)

Q. Disk files
R. Using standard I/O in C
.

Resources

Tony Gaddis. *Starting Out with C++*. 9th Ed. Pearson, 2019.

Deiter and Deitel. *C How To Program*. 10th Ed. Prentice-Hall, 2016.

Zed Shaw. *Learn C the Hard Way*. 1st Ed. Pearson, 2016.

Jeganathan Swaminathan. *Modern C++ Programming*. 1st Ed. Packt Publishing, 2017.

Scott Meyers. *Effective Modern C++*. 11th Ed. O'Reilly Media, 2018.

Brain Kernighan. *Programming in C*. 2nd Ed. Prentice Hall, 1988.

Resources Other

1. Lab packet
2. Handouts
    a. Instructor authored notes
    b. Topic summaries
    c. Journal articles
    d. Manufacturer's literature

Top of page
Key: 1665