SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECB211 – Microcontroller Programming & Interfacing

II YEAR/ IV SEMESTER

1

UNIT 3 – PIC PROGRAMMING IN C

TOPIC 6 – Programming ROM allocation in C

Programming ROM allocation in C

- **Introduction to ROM (Read-Only Memory)**
  - ROM is a type of memory that retains its contents even when power is turned off.
  - It's commonly used for storing firmware, software that is closely tied to hardware and doesn't need frequent updates.
- **Purpose of Programming ROM Allocation**:
  - In embedded systems and firmware development, managing ROM efficiently is crucial for optimizing space and performance.
  - Allocating ROM effectively involves organizing the code and data to minimize space usage while ensuring accessibility and maintainability.

Programming ROM allocation in C

Strategies for ROM Allocation

a.**Code Optimization**:

•Utilize compiler optimizations to reduce code size without sacrificing functionality.

•Techniques like loop unrolling, inlining, and constant folding can significantly decrease the size of compiled code

b. **Data Compression**:

•Compress data stored in ROM to save space. Use algorithms like Huffman coding or LZ77 to compress data before storing it in ROM.

•Decompression routines are then included in the firmware to unpack data as needed.

c. **Selective Inclusion**:

•Include only the necessary parts of libraries or modules to minimize ROM usage.

•Conditional compilation directives (#ifdef) can be used to include/exclude sections of code depending on the configuration.

Programming ROM allocation in C

Strategies for ROM Allocation

d. **Read-Only Data**:

•Declare read-only data (constants, lookup tables, etc.) with the **const** keyword to instruct the compiler to store them in ROM

•This ensures that data is stored in non-modifiable memory, reducing RAM usage

e. **Memory Banking**:

•Divide ROM into banks and load only the necessary banks into memory as required.

•Useful for systems with limited ROM space where not all code and data need to be loaded simultaneously

f. **Overlay Techniques**:

•Use overlay techniques to manage code that exceeds available ROM space.

•Different parts of the code are loaded into memory as needed, swapping out sections when they are no longer required.

Programming ROM allocation in C

Implementation in C:

•Use compiler-specific directives or linker scripts to specify ROM allocation requirements.

•Utilize **const** keyword to declare read-only data.

•Employ conditional compilation directives to selectively include/exclude code based on requirements.

•Implement compression and decompression routines as necessary.
•Optimize critical sections of code to minimize size while maintaining performance

Programming ROM allocation in C

Conclusion

- Efficient ROM allocation is essential for embedded systems and firmware development
- By employing various strategies such as code optimization, data compression, and selective inclusion, developers can effectively manage ROM usage
- Careful consideration of memory usage and optimization techniques can lead to more compact and efficient firmware designs

THANK YOU