Anglo American Cataloguing Rules University Libraries Manual

Anglo American Cataloguing Rules University Libraries are essential frameworks that guide the organization, description, and management of library materials in academic institutions worldwide. These rules ensure consistency, accuracy, and efficiency in cataloging processes, thereby enhancing users? ability to locate and access resources effectively. This article explores the principles of Anglo American Cataloguing Rules (AACR), their application in university libraries, and the benefits they offer to academic communities.

Introduction to Anglo American Cataloguing Rules (AACR)

What Are AACR?

The Anglo American Cataloguing Rules (AACR) are a set of internationally recognized standards for cataloging library materials. Developed initially in the 1960s and regularly revised, AACR provides catalogers with comprehensive guidelines on describing various formats of resources, including books, serials, audio-visual materials, and electronic resources. The primary goal is to create consistent bibliographic records that facilitate resource discovery and management.

Historical Development of AACR

- Origins: AACR was developed as a collaboration between American and British cataloging standards organizations.
- Revisions: The rules have undergone multiple revisions, with notable updates in AACR2 (second edition) and more recently AACR3, reflecting evolving cataloging needs.
- Transition to RDA: Currently, many libraries are transitioning toward Resource Description and Access (RDA), which builds upon AACR principles but emphasizes digital resource cataloging.

The Role of AACR in University Libraries

Standardization and Consistency

University libraries typically house vast collections of diverse materials. Implementing AACR ensures that all catalog records adhere to a standardized format, making it easier for students and researchers to locate resources regardless of the specific library within an institution.

Enhanced Resource Discovery

Accurate and detailed bibliographic records enable users to find materials efficiently through OPACs (Online Public Access Catalogs). AACR emphasizes precise descriptions, author details, publication data, and subject headings, which improve searchability.

Facilitating Resource Management

Proper cataloging under AACR simplifies collection management, including acquisitions, weeding, and preservation efforts. It also supports interlibrary loan services by providing consistent bibliographic data.

Core Principles of AACR in University Libraries

Descriptive Cataloging

Catalogers create detailed records that include:

- Title and statement of responsibility
- Edition statement
- Publisher, publication date, and physical description
- Series information
- Notes on content or special features

Subject Cataloging

AACR encourages the use of standardized subject headings and classification schemes like Dewey Decimal Classification or Library of Congress Classification to organize materials thematically.

Authority Control

Maintaining consistent names for authors, publishers, and subjects is vital for accurate retrieval. AACR emphasizes authority control to prevent duplication and confusion.

Application of AACR in Different Types of University Library Materials

Books and Monographs

Cataloging books involves recording detailed bibliographic information, including edition statements, series data, and author names, following AACR guidelines to ensure clarity and consistency.

Serials and Periodicals

Serial publications require special treatment to account for title changes, numbering, and frequency. AACR provides rules for cataloging continuing resources, ensuring updates are accurately reflected.

Electronic and Digital Resources

With the rise of e-books, online journals, and multimedia, AACR has adapted to include guidelines for digital resource description, emphasizing access points, URLs, and Digital Object Identifiers (DOIs).

Audio-Visual and Non-Print Materials

Movies, music recordings, maps, and photographs are cataloged following AACR?s provisions for non-book materials, with attention to physical characteristics and content description.

Benefits of Using AACR in University Libraries

Improved User Experience

Accurate and consistent records enable students, faculty, and researchers to locate resources quickly, fostering effective teaching and learning.

Interoperability and Data Sharing

Standardized cataloging supports data exchange between libraries worldwide, facilitating resource sharing and collaborative collection development.

Efficiency in Cataloging Processes

Having a clear set of rules reduces cataloging time and minimizes errors, allowing library staff to manage large collections more effectively.

Challenges and Future of AACR in Academic Libraries

Digital Transformation

The shift towards digital resources presents challenges in applying traditional cataloging rules, prompting adaptations and updates to existing standards.

Transition to RDA

Many university libraries are transitioning from AACR to Resource Description and Access (RDA), which offers a more flexible framework suitable for digital environments, but this transition requires retraining and system upgrades.

Integration with Metadata Standards

AACR-based catalog records often integrate with MARC (Machine-Readable Cataloging) formats and other metadata standards, necessitating ongoing alignment and interoperability.

Implementing AACR in University Libraries

Training and Professional Development

Ensuring cataloging staff are well-versed in AACR standards involves regular training, workshops, and participation in professional communities.

Cataloging Software and Tools

Utilizing integrated library systems (ILS) that support AACR-compliant cataloging simplifies record creation and management.

Quality Control and Review

Regular audits and peer reviews help maintain high standards of bibliographic data quality, ensuring continued adherence to AACR principles.

Conclusion

The Anglo American Cataloguing Rules have played a pivotal role in shaping the cataloging practices of university libraries, fostering consistency, accuracy, and resource discoverability. While evolving digital environments pose new challenges, the core principles of AACR continue to underpin effective bibliographic description. As libraries transition towards newer standards like RDA, understanding and applying AACR remains fundamental for library professionals committed to maintaining high-quality, accessible collections that support academic excellence.

References

- American Library Association. (2002). Anglo-American Cataloguing Rules, Second Edition (AACR2).
- Library of Congress. (2020). Introduction to RDA: Resource Description & Access.

- International Federation of Library Associations and Institutions (IFLA). (2010). Functional Requirements for Bibliographic Records (FRBR).
- Smith, J. (2021). Cataloging in Academic Libraries: Principles and Practice. Academic Publishing.

Keywords: Anglo American Cataloguing Rules, AACR, university libraries, bibliographic standards, cataloging, resource description, digital resources, library collections, cataloging guidelines, RDA

Anglo-American Cataloguing Rules (AACR) in University Libraries: A Comprehensive Review

Introduction

In the realm of academic libraries, cataloguing standards serve as the backbone for organizing, describing, and accessing vast collections of resources. Among these standards, the Anglo-American Cataloguing Rules (AACR) have historically played a pivotal role, especially in university libraries across the globe. This detailed review delves into the origins, principles, implementation, and evolution of AACR within the context of university libraries, highlighting its significance in shaping modern cataloguing practices.

Origins and Development of AACR

Historical Background

- Origins: AACR was first published in 1967 as a joint effort by the American Library Association (ALA) and the Chartered Institute of Library and Information Professionals (CILIP) in the UK.
- Purpose: Designed to establish a standard set of rules for cataloguing materials across library types, with a focus on facilitating resource discovery and retrieval.
- Evolution: Over time, AACR underwent multiple revisions to accommodate new formats, technologies, and cataloguing needs, culminating in the latest edition, AACR2, published in 1978, with subsequent updates.

Transition to RDA

- Recognizing the limitations of AACR2 in the digital age, the library community gradually transitioned towards Resource Description and Access (RDA) starting around 2010.
- Despite this, AACR remains influential, especially in legacy catalogues and institutions resistant to immediate change.

Core Principles and Structure of AACR

Fundamental Principles

- Consistency: Ensures uniformity in cataloguing entries across libraries.
- Utility: Focuses on providing information that facilitates resource discovery.
- Flexibility: Adapts to various formats and resource types.
- User-Centric Approach: Prioritizes the needs of users for efficient retrieval.

Structure and Key Elements

- Rules and Instructions: Organized into clearly defined rules governing different resource types.
- Descriptive Cataloguing: Details how to describe titles, authors, publication details, and physical characteristics.
- Subject and Access Points: Provides guidelines for establishing access points, including author names, titles, and subject headings.
- Standardized Formats: Employs consistent data entry formats, such as the use of abbreviations and punctuation.

Implementation of AACR in University Libraries

Cataloguing Workflow

- Resource Identification: Determining the type and scope of the material.
- Descriptive Cataloguing: Applying AACR rules to record bibliographic details.
- Subject Cataloguing: Assigning subject headings and classification numbers.
- Authority Control: Ensuring consistent use of names and subjects across records.
- Record Creation: Generating catalog entries for online public access catalogs (OPACs).

Training and Staff Competence

- University libraries invest heavily in training cataloguers to ensure adherence to AACR standards.
- Continuous professional development ensures familiarity with updates and best practices.

Integration with Library Management Systems

- AACR-compliant records are integrated into integrated library systems (ILS) for seamless

access.
- The cataloguing process is often supported by automated tools that assist in data entry and validation.

Challenges and Criticisms of AACR in University Settings

Limitations in the Digital Environment

- Inadequate for Electronic Resources: AACR was primarily designed for physical materials and struggles to accommodate digital resources effectively.
- Complexity and Rigid Rules: The detailed rules can be cumbersome, especially with the vast diversity of university collections.

Transitioning to RDA

- Many university libraries faced challenges during the shift from AACR2 to RDA, including retraining staff and re-cataloguing collections.
- The transition period led to inconsistencies in bibliographic records across institutions.

International and Multilingual Considerations

- AACR's Anglo-American focus sometimes limits its applicability in multilingual or international collections, necessitating adaptations.

Advantages of Using AACR in University Libraries

Despite criticisms, AACR offers several benefits:

- Standardization: Ensures consistent cataloguing practices across departments and institutions.
- Facilitates Resource Sharing: Uniform records enable easier interlibrary loan and resource discovery.
- Historical Data: A vast legacy of AACR records provides a valuable foundation for ongoing cataloguing efforts.
- Comprehensive Guidelines: Covers a wide range of resource types, from monographs to serials and audiovisual materials.

Impact on Academic and Research Communities

Enhancing Discoverability

- Accurate and detailed bibliographic records improve search precision, aiding researchers and students in locating pertinent resources.

Supporting Academic Integrity and Provenance

- Proper cataloguing ensures clear attribution and resource authenticity, vital in scholarly work.

Enabling Interoperability

- AACR records facilitate data exchange and integration with other library systems and external databases.

The Future of AACR in University Libraries

Transition to RDA and Beyond

- While AACR has historically been dominant, the shift towards RDA is gradually redefining cataloguing standards.
- University libraries are increasingly adopting RDA for its flexibility and better alignment with digital resources.

Digital and Linked Data Initiatives

- Future cataloguing practices are moving towards linked data, semantic web technologies, and automation, which may diminish reliance on traditional AACR rules.
- Nonetheless, understanding AACR remains crucial for managing legacy records and ensuring continuity.

Preservation of Legacy Data

- Many university libraries continue to maintain AACR records for older collections, necessitating ongoing expertise in AACR standards.

Conclusion

The Anglo-American Cataloguing Rules (AACR) have historically served as a cornerstone for cataloguing practices in university libraries, fostering consistency, discoverability, and resource sharing. Despite the advent of newer standards like RDA and technological advancements, AACR's influence remains significant, especially in managing legacy collections and maintaining bibliographic control. As university libraries navigate the digital transformation, a nuanced understanding of AACR?its principles, strengths, and limitations?continues to be vital for effective collection management and scholarly communication. Moving forward, integrating AACR knowledge with emerging cataloguing paradigms will ensure that university libraries remain resilient and adaptable in serving their academic communities.

QuestionAnswer What are the key features of Anglo-American Cataloguing Rules (AACR) used in university libraries? AACR provides standardized guidelines for cataloging library materials, emphasizing consistency in author, title, and subject entries, and facilitating easy retrieval of resources across university libraries. How have the Anglo-American Cataloguing Rules evolved to meet modern university library needs? AACR has evolved into AACR2 and later into RDA (Resource Description and Access), incorporating digital resources, electronic formats, and more flexible cataloging standards to align with technological advancements in university libraries. What is the role of AACR in ensuring cataloging consistency across university libraries? AACR provides comprehensive rules and standards that promote uniformity in catalog records, enabling university libraries to share resources effectively and support scholarly research with reliable, standardized bibliographic data. How do Anglo-American Cataloguing Rules impact the cataloging of electronic and digital resources in university libraries? AACR guides catalogers in applying consistent metadata standards to electronic and digital resources, ensuring accurate description, discoverability, and integration of diverse formats within university library catalogs. What are the challenges faced by university libraries in implementing Anglo-American Cataloguing Rules? Challenges include adapting to new digital formats, maintaining consistency amidst evolving standards like RDA, managing large volumes of diverse resources, and training cataloging staff to stay updated with the latest cataloging practices.

Related keywords: Anglo American Cataloguing Rules, AACR, library cataloging standards, university library cataloging, AACR2, bibliographic description, MARC records, cataloging rules, academic libraries, library metadata

LIBRARIES FOR CODEVISIONAVR

libraries for codevisionavr are essential tools that significantly enhance the development process

when working with AVR microcontrollers using the CodeVisionAVR compiler. These libraries provide pre-written functions and modules that simplify complex tasks, improve code efficiency, and promote code reusability. Whether you are a beginner or an experienced embedded systems developer, understanding how to utilize and implement libraries in CodeVisionAVR can accelerate your project development and ensure more reliable, maintainable code.

Understanding Libraries in CodeVisionAVR

What Are Libraries?

Libraries in programming are collections of precompiled routines that programmers can include in their projects to perform specific functions without writing the code from scratch. In the context of CodeVisionAVR, libraries can be standard or custom, providing functionalities such as handling peripherals, communication protocols, data processing, and more.

Types of Libraries in CodeVisionAVR

- Standard Libraries: These come bundled with the compiler and include functions for I/O, math, string handling, and peripheral control.
- User-Defined Libraries: Created by developers to encapsulate specific functionalities tailored to their projects.
- Third-Party Libraries: Open-source or commercially available libraries created by the community or vendors to extend the capabilities of the compiler.

Popular Libraries for CodeVisionAVR

Many libraries are available to streamline development with CodeVisionAVR. Here are some of the most commonly used:

1. AVR Standard Peripheral Libraries

These libraries provide functions to control microcontroller peripherals such as timers, ADC, UART, SPI, I2C, and GPIOs.

2. LCD and Display Libraries

Libraries like `lcd.h` facilitate easy interfacing with character LCDs, graphical displays, and other visual output devices.

3. Communication Protocol Libraries

Including support for UART, I2C, SPI, CAN, and USB, these libraries simplify serial communication setup and data transfer.

4. Data Handling and Storage Libraries

Libraries for EEPROM, external memory, and data encoding/decoding (e.g., base64) help manage data persistence and processing.

5. Real-Time Operating System (RTOS) Libraries

For complex applications, RTOS libraries provide task scheduling, synchronization, and resource management.

How to Use Libraries in CodeVisionAVR

Including Libraries in Your Project

Most libraries are included by adding their header files at the beginning of your source code:

```c
include
```

Ensure that the corresponding library files are accessible within your project directory or compiler include paths.

Configuring Libraries

Some libraries require configuration before use, such as setting pin modes, baud rates, or peripheral options. Consult library documentation for specific setup procedures.

Linking Libraries

When compiling, ensure that the library object files (`.lib` or `.a`) are linked correctly with your main project files. CodeVisionAVR typically manages this automatically if the libraries are properly included.

Developing Custom Libraries in CodeVisionAVR

Creating custom libraries promotes code reuse and modular design, making complex projects more manageable.

Steps to Create a Custom Library

- Identify common functionalities that can be encapsulated.
- Write the functions in separate source (`.c`) and header (`.h`) files.
- Declare functions in the header file for external linkage.
- Compile the source files into a static library or object files.
- Include the header in your projects and link against the compiled library.

Best Practices for Custom Libraries

- Use clear and descriptive function names.
- Document functions with comments.
- Keep the interface simple and consistent.
- Avoid global variables; prefer parameter passing.

Examples of Common Libraries in Use

1. Controlling an LCD Display

```c

include

void main() {

lcd_init();

lcd_puts("Hello, World!");

while(1) {

// main loop

}

}
```

```

This example demonstrates initializing an LCD and displaying a message using the `lcd.h` library.

### 2. UART Communication

```c

include

void main() {

uart_init(9600);

uart_puts("Data Transmission Started");

while(1) {

// send or receive data

}

}

```

Using UART libraries simplifies serial communication setup.

Resources for Finding and Developing Libraries

Official Documentation and Forums

- Microchip's AVR documentation provides detailed information on peripheral control and library functions.
- CodeVisionAVR forums and user communities are valuable for shared libraries and troubleshooting.

Open-Source Libraries
```

- Platforms like GitHub host numerous AVR-compatible libraries.
- Always verify compatibility and licenses before integrating third-party code.

Creating Reusable Libraries

Developing your own library repository for frequently used functions can save time across multiple projects. Maintain clear documentation, version control, and example usage to maximize benefits.

Optimizing Library Usage for Better Performance

Minimize Library Size

- Include only necessary functions.
- Use compiler optimization settings.
- Avoid unnecessary library features.

Maintain Code Readability

- Comment your code extensively.
- Use consistent naming conventions.
- Modularize code for easier maintenance.

Testing and Validation

- Rigorously test libraries in different scenarios.
- Write unit tests where applicable.
- Keep libraries updated to fix bugs and improve functionality.

Conclusion

Libraries for CodeVisionAVR are invaluable assets that can significantly streamline embedded system development with AVR microcontrollers. Whether leveraging built-in standard libraries or creating custom modules, understanding how to effectively incorporate libraries into your projects will enhance productivity, code quality, and system reliability. As the AVR ecosystem continues to grow, so does the availability of robust libraries, making it easier than ever to develop complex applications with minimal effort.

By mastering the use of libraries, exploring community resources, and developing reusable

modules, developers can harness the full potential of CodeVisionAVR and produce efficient, maintainable, and scalable embedded solutions.

Libraries for CodeVisionAVR are fundamental tools that significantly enhance the development experience when working with AVR microcontrollers using the CodeVisionAVR compiler. These libraries abstract complex hardware functionalities, providing developers with easy-to-use interfaces to perform tasks such as communication, timing, analog-to-digital conversion, and more. By leveraging well-designed libraries, developers can accelerate their development process, improve code reliability, and focus more on application logic rather than low-level hardware management.

Introduction to Libraries in CodeVisionAVR

Libraries in CodeVisionAVR serve as collections of pre-written code modules that implement specific functionalities for AVR microcontrollers. They are designed to simplify programming by providing ready-to-use functions and routines, thus reducing development time and minimizing errors. Libraries can be built-in (supplied with the compiler) or third-party, and they often cover a broad spectrum of hardware peripherals and system features.

The core benefit of utilizing libraries is that they facilitate portability, scalability, and ease of maintenance. For embedded developers, especially those new to AVR microcontrollers, libraries act as a bridge, allowing them to harness complex hardware features without delving into intricate register-level programming.

Built-in Libraries in CodeVisionAVR

CodeVisionAVR comes with a comprehensive suite of built-in libraries that cater to common microcontroller functionalities. These include libraries for handling I/O, timers, USART, SPI, I2C, ADC, PWM, and more. Below is an overview of some of the most frequently used built-in libraries:

Standard I/O Library

Provides routines for general input/output operations, including setting pin directions and reading/writing port values.

Timer/Counter Libraries

Enable precise timing operations, delays, and PWM generation.

Serial Communication Libraries (USART, SPI, I2C)

Facilitate serial data exchange, crucial for interfacing with sensors, modules, or other

microcontrollers.

Analog-to-Digital Converter (ADC) Library

Simplifies reading analog signals from sensors.

Pulse Width Modulation (PWM) Library

Allows for control of motor speed, LED brightness, and other applications requiring analog-like control.

Popular Third-Party Libraries and Extensions

Beyond the standard library suite, the CodeVisionAVR community and third-party developers have created numerous libraries to extend functionality:

RTOS and Multitasking Libraries

Enable real-time operation and multitasking on AVR microcontrollers with limited resources.

USB Libraries

Support for USB device development, including HID, CDC, and mass storage classes.

Sensor and Communication Protocol Libraries

Implement protocols like Modbus, CAN, or custom sensor protocols for applications in industrial automation or robotics.

Graphics and LCD Libraries

Simplify driving graphical displays, LCDs, and OLED screens.

Features and Benefits of Using Libraries in CodeVisionAVR

Utilizing libraries offers several advantages:

- Simplification of Complex Hardware Operations: Libraries abstract low-level register manipulations.
- Code Reusability: Once written or acquired, libraries can be reused across multiple projects.

- Faster Development Cycle: Developers spend less time coding hardware interfaces.
- Enhanced Reliability: Tested libraries reduce the likelihood of bugs in hardware control.
- Portability: Libraries often facilitate porting code between different AVR models.

How to Integrate Libraries in CodeVisionAVR

Integrating libraries into your project typically involves:

- Including the appropriate header files (`include` directives).
- Ensuring the corresponding source files are linked during compilation.
- Configuring any necessary parameters or settings within the library functions.

For built-in libraries, inclusion is straightforward. For third-party libraries, you may need to download or clone the source code, then add it to your project directory.

Case Study: Using the ADC Library in CodeVisionAVR

The ADC library in CodeVisionAVR simplifies reading analog signals:

- Features:
- Multiple channels support.
- Adjustable sampling speed.
- Automatic or manual trigger options.

- Sample Usage:

```c
include

int main() {

ADC_init(); // Initialize ADC

while(1) {

int sensor_value = ADC_read(0); // Read from channel 0
```

```
// Process sensor_value as needed

}

return 0;

}
```

- Pros:
- Easy to implement.
- Provides calibration and reference voltage options.

- Cons:
- Limited customization for advanced timing or filtering.
- Some overhead compared to direct register access.

Challenges and Limitations of Libraries in CodeVisionAVR

While libraries are powerful, they come with certain limitations:

- Overhead and Size: Libraries may add code size, which can be critical for memory-constrained MCUs.
- Reduced Flexibility: High-level abstractions might limit fine control over hardware.
- Learning Curve: Understanding how libraries work internally is necessary for debugging.
- Compatibility Issues: Some third-party libraries may not be fully compatible with all AVR models or CodeVisionAVR versions.

Best Practices for Using Libraries Effectively

To maximize the benefits and minimize issues:

- Read Documentation Carefully: Understand library functions and limitations.
- Keep Libraries Up-to-Date: Use the latest versions to benefit from bug fixes and improvements.
- Modular Design: Use libraries selectively; avoid including unnecessary ones.
- Test Thoroughly: Validate library functions within your application context.

- Optimize for Size: When working with limited memory, choose lightweight libraries or optimize your code.

Conclusion

Libraries for CodeVisionAVR are indispensable tools that streamline embedded development with AVR microcontrollers. They enable rapid prototyping, reliable hardware interfacing, and scalable project development. Whether utilizing the built-in libraries for common peripherals or integrating third-party extensions for specialized needs, understanding their features, advantages, and limitations is crucial for effective embedded system design. As the ecosystem continues to grow, mastering the use of libraries will remain a key skill for developers working within the AVR world, helping to deliver robust and efficient embedded solutions.

In summary, libraries in CodeVisionAVR empower developers to harness the full potential of AVR microcontrollers with minimal complexity. They serve as a bridge between hardware intricacies and application logic, making embedded development more accessible, efficient, and maintainable.

QuestionAnswer What are some popular libraries available for CodeVisionAVR? Popular libraries for CodeVisionAVR include AVR libc for standard C functions, LCD libraries for display control, UART libraries for serial communication, and EEPROM libraries for non-volatile memory management. How can I integrate an LCD library into my CodeVisionAVR project? You can integrate an LCD library by including the relevant header files in your project, configuring the pins according to your hardware setup, and using the provided functions to initialize and control the LCD display. Are there any open-source libraries compatible with CodeVisionAVR? Yes, several open-source libraries such as AVR libc and third-party LCD or sensor libraries are compatible with CodeVisionAVR, often requiring minimal adaptation for your specific project. Can I use custom libraries with CodeVisionAVR for specific peripherals? Absolutely, you can develop or incorporate custom libraries to interface with specific peripherals, ensuring modularity and reusability in your CodeVisionAVR projects. What is the best way to manage dependencies of libraries in CodeVisionAVR? Managing dependencies involves organizing your include paths, keeping libraries updated, and ensuring compatibility with your compiler version, often by maintaining a well-structured project directory. Are there any libraries for real-time clock (RTC) modules in CodeVisionAVR? Yes, there are libraries and code snippets available for interfacing with RTC modules like DS1307 or DS3231, which can be integrated into CodeVisionAVR projects for timekeeping functionalities. How do I troubleshoot library compatibility issues in CodeVisionAVR? Troubleshoot by verifying library compatibility with your compiler version, checking for correct include paths, reviewing documentation, and testing libraries with simple example projects. Is it possible to create custom libraries in CodeVisionAVR for reusable code modules? Yes, you can create your own custom libraries by writing modular code, compiling them into static or dynamic libraries, and including them in your projects for reusability. Are there any community forums or resources for libraries specific to CodeVisionAVR? While dedicated forums are limited, communities

like AVR Freaks, Stack Overflow, and embedded systems forums often share libraries, code snippets, and advice relevant to CodeVisionAVR development. What are the key considerations when choosing libraries for embedded projects in CodeVisionAVR? Consider compatibility with your microcontroller, library stability, community support, documentation quality, and whether the library meets your project?s specific hardware and performance requirements.

Related keywords: CodeVisionAVR, AVR libraries, embedded libraries, microcontroller libraries, AVR C libraries, CodeVision libraries, AVR SDK, code libraries for AVR, software libraries for AVR, programming libraries for CodeVision

Read the full article online: LIBRARIES FOR CODEVISIONAVR