**CODE REVIEW PROCESS AND PROCEDURES**

**Purpose**

This document describes what and how ABB Bartlesville code review process shall be accomplished. Code review is a tool used to improve software quality by finding and fixing errors early in the development process. The review checklist will help reviewer to analyze source code. The process supports 2 different types of code reviews: 1) Table Top Code Review and 2) Peer Over-The-Shoulder Code Review.

The Table Top Code review is a formally planned sit-down review. The process that shall be adhered to as defined below.

**Table Top Code Review Process:**

1. Minimum of 2 reviewers.
2. Code shall be place under configuration control using shelving (TFS based code).
3. Source code, Code Review Checklist (Appendix A) and Code Review Form (Appendix B) shall be sent to reviewers at least 48 hours prior to comment due date.
4. Reviewers record comments in Code Review Form. Reviewers provide Code Review Form back to coder at least 4 hours prior to review.
5. Source code review is limited to no more than 100 lines of new/changed code per review (executable, white space and comments), per session. Multiple sessions will be required for large code deliveries.
6. Coder shall respond to each comment. Coder’s response is recorded in summary Code Review Form.
7. Coder holds meeting and all questionable comments shall be discussed and clarified during meeting.
8. Coder summaries any additional comments in summary Code Review Form and stores in document repository and notifies reviewers.
9. Code Review Form includes total lines of code reviewed, time spent reviewing the code and number of comments recorded for Quality Metrics.
10. Reviewers shall evaluate the responses to their comments in the Code Review Form. If they determine that any of the responses are unacceptable to them then they shall get with the Coder within 48 hours. If resolution cannot be reached then they shall work with the Project Manager to resolve the issue.

The Peer Over-The-Shoulder review is an informal code review. The process for this type of review is as defined below:

**Peer Over-The-Shoulder Code Review Process:**

1. Minimum of 2 reviewers.
2. Coder sits down with reviewers and reviews code at desk or other appropriate area using Code Review Checklist (Appendix A).
3. Coder records comments in Code Review Form (Appendix B).
4. Source code review is limited to no more than 100 lines of new/changed source per reviewer, per session. Multiple sessions will be required for large code deliveries.
5. Coder shall respond to each comment. Coder response is recorded Code Review Form.
6. Coder shall store Code Review Form in document repository.
7. Code Review Form includes total lines of code reviewed, time spent reviewing the code and number of comments recorded for Quality Metrics.
8. If reviewer disagrees with coder’s response to a comment then review has 48 hours to resolve the issues with the coder. If resolution cannot be reached then they shall work with the Project Manager to resolve the issue.

## APPENDIX A

**CODE REVIEW CHECKLIST**

## Code Review Check List (In-House)

**Specification**

|  |  |
| --- | --- |
|  | Does the code meet the specification? |

**Structure**

|  |  |
| --- | --- |
|  | Conforms to coding standards (In-house or MicroSoft Windows Coding Standard)? |
|  | Unreachable or “dead” code found? |
|  | Repeated code that could be made into functions? |
|  | Code that could be replaced by existing functions? |
|  | Enumerations and defined constants rather than numeric constants? |
|  | Code complexity minimized? |

**Loops and Branches**

|  |  |
| --- | --- |
|  | Are all loops properly nested? |
|  | Case statements include default? |
|  | Indexes are initialized prior to entering a loop? |
|  | Index variables are not modified within loop? |
|  | Index variables have sufficient range? |

**Documentation**

|  |  |
| --- | --- |
|  | Comments are used where appropriate? |
|  | Comments consistent with code? |
|  | Function definitions include return value, argument and parameter definitions. |

**Variables**

|  |  |
| --- | --- |
|  | Variable names are meaningful and appropriate? |
|  | Variables have proper type consistency or casting? |
|  | Variables are used? |

**Arithmetic Operations**

|  |  |
| --- | --- |
|  | Avoid comparing floating-point numbers for equality? |
|  | Code minimizes rounding errors? |
|  | Avoid additions and subtractions on numbers with greatly different magnitudes? |
|  | Divisors tested for zero? |
|  | Computations done using variables of consistent types? |
|  | Overflows and underflows are checked? |

**Defensive Programming**

|  |  |
| --- | --- |
|  | Array indexes tested against limit? |
|  | Pointers and file handles tested for valid assignments before being used? |
|  | File access tested for valid return cases and record requests within bounds? |
|  | Files checked for existence before attempting to access them? |
|  | Opened files closed appropriately? |
|  | Function input parameters tested for validity and usage? |
|  | Function output variables assigned valid values? |
|  | Memory allocated is properly de-allocated? |
|  | Timeout and exit cases used for external device access? |

**System Related Checks**

|  |  |
| --- | --- |
|  | Applications warm start properly? |
|  | Variables initialized? |
|  | Potential memory leaks? |
|  | Change impacts external interfaces or protocols? |
|  | Register access verified? |
|  | Semaphore and locks do not cause “deadly embrace”? |
|  | Memory used is appropriate size and type? |
|  | Processor usage is appropriate? |
|  | Is follow-up code review necessary? |

**APPENDIX B**

**CODE REVIEW FORM**

**Code Review Form**

**Date: Reviewer:**

**Time Spent Reviewing: Lines of Code Reviewed:**

**File and File Versions Reviewed:**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
4. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
5. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**File and File Versions Containing Fixes:**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
4. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
5. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

| **Comment #** | **File Name & Line Number** | **Action Item Description** | **Originator** | **Status** | **Results/Comments** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |