Guidelines for the Use of Function Block Diagram in Reactor Protection Systems

Dong-Ah Lee*, Junbeom Yoo
Konkuk University

Jang-Soo Lee
Korea Atomic Energy Research Institute
Table of Contents

• Introduction

• Background

• Dependable Cases for FBD Programs

• Case Study

• Conclusion
A Safe Programming Guidance of Function Block Diagram for Reactor Protection Systems

INTRODUCTION
Introduction

• **Software in safety critical systems**
  - Nuclear power plant
    • RPS (Reactor Protection System)
    • ESF-CCS (Engineering Safety Feature Component Control System)
  - Automotive systems
  - Medical systems
  - Etc.

• **Failure of the software**
  - injuries to people, damages to the environment, or extensive economic losses

• **Dependability of the software is important for SAFETY and PERMISSION.**
Software development in the nuclear power plant domain

- Software requirement specification (natural language)
- Software design written in FBD/LD
- Software implementation by a software engineering tool

- pSET (POSAFE-Q Software Engineering Tool)
  - KNICS ARP-1400 (PLC)
A Safe Programming Guidance of Function Block Diagram for Reactor Protection Systems

BACKGROUND
Related Work – Function Block Diagram

- International Standard IEC 61131-3
  - Programmable controllers – Part 3: Programming languages
  - IL, ST, LD, FBD, SFC
  - A graphical language
Introduction (cont’d)

• POSAFE-Q Software Engineering Tool (pSET)
  – Korea Nuclear Instrumentation & Control System R&D Center (KNICS)

• FBD and Ladder Diagram (LD) to design a software of POSAFE-Q Programmable Logic Controller (PLC)

• ANSI-C language to implement the design
Related Work – Dependable Programming

- A.k.a.
  - programming guidelines
  - safe programming

- MISRA-C: Automotive industry
- DO-178B: Airborne systems
- NuREG/CR-6463: Nuclear domain
  - IEC 61131-3 programming language, c/c++, Ada, Pascal, PL/M
DEPENDABLE CASES
FOR FBD PROGRAMS
(1/5) Execution control except EN and ENO signals

- Boolean “EN” (Enable) input and “ENO” (Enable Out) output: Optional ports
- Control flows in data flow based language: NOT SUITABLE
- The output ‘E’ is not clear
- SOLUTION: Using selection blocks (SEL, MUX)

If \((A \& B) = \text{FALSE}\) then \(E := C + D\)
If \((A \& B) = \text{TRUE}\) then \(E := C - D\)

If \(G = \text{FALSE}\) then \(\text{OUT} := \text{IN1}\) which \(\text{IN1} = A + B\)
If \(G = \text{TRUE}\) then \(\text{OUT} := \text{IN2}\) which \(\text{IN2} = A - B\)
(2/5) Usage of Output Variables

- One evaluation per one cycle
- Ambiguous connections: [ ①C to ②C ] [ ③C - ②C ]
- Overwriting problems: different values of ‘C’ in a cycle
- SOLUTION: clear connections using connector/continuation or feedback variables
(3/5) Consensus of Data Type

- **OVERLOADING** is not a problem in FBD.
- Automatic type casting of target systems is a **PROBLEM**.

**SOLUTION**
- eliminate implicit type casting → use type conversion blocks
- clarify a type of blocks
(4/5) Initialization of Feedback Variables

- Explicit initiation of all variables are not mandatory.
- Storable variables (i.e., feedback) MUST be initiated.
- No values in feedback variables at the beginning without initialization.
- 3 mechanisms of initialization
  - the default initial value(s) of the underlying elementary data types as defined in IEC 61131-3;
  - NULL, if the variable is a reference;
  - or the user-defined value(s) of the variable; this value is optionally specified in the variable declaration.
(5/5) Explicit Order of Evaluation

- Labeled or displayed order ≠ EVALUATION ORDER (*mislabeled*)

- Two solutions to eliminate the mislabeling:
  - programming without labeling order;
  - programming with labeling order in the same order of evaluation
CASE STUDY

A Safe Programming Guidance of Function Block Diagram for Reactor Protection Systems
Case study

• **FIXED RISING**
  - one of logics in RPS
  - the engineering tool: pSET

• Overwriting output ports

• Implicit execution order
Case study

- **Eliminate overwriting**
  - use connections/continuations
  - use feedback variables
- **Clarify explicit execution order**
A Safe Programming Guidance of Function Block Diagram for Reactor Protection Systems

CURRENT STATUS
FBDChecker

- CASE tool: FBDChecker
  - Automation tool for checking FBD programs about guidelines
  - the *de facto* standard format of FBD files (PLCopen TC6)
  - checks FBD programs
FBDChecker - Guidelines

• Reliability
  – Execution order
  – Eliminating incorrect move block
  – Implicit/explicit type conversion
  – Variable initialization
  – Etc.

• Maintainability
  – Naming convention
    • Length – too short, too long
  – Eliminating crossed connections
  – Eliminating overlapped blocks
  – Etc.
FBDChecker - Results

- Filtering by POU, types of guideline

<table>
<thead>
<tr>
<th>POU</th>
<th>GRADE</th>
<th>KIND</th>
<th>SHOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_R_O_clean</td>
<td>All</td>
<td>name: PTSP</td>
<td>localId: null</td>
</tr>
<tr>
<td>POU_RISING</td>
<td>All</td>
<td>name: PTSP_T19</td>
<td>localId: null</td>
</tr>
<tr>
<td>MANUAL_RATE_FALLING</td>
<td>All</td>
<td>name: TRIP</td>
<td>localId: null</td>
</tr>
<tr>
<td>V_R_O_clean</td>
<td>All</td>
<td>name: TSP</td>
<td>localId: null</td>
</tr>
<tr>
<td>FLY_FALLING</td>
<td>All</td>
<td>name: PTRIP_CNT</td>
<td>localId: null</td>
</tr>
<tr>
<td>V_R_O_clean</td>
<td>All</td>
<td>name: TRIP_CNT</td>
<td>localId: null</td>
</tr>
<tr>
<td>V_R_O_clean</td>
<td>All</td>
<td>name: PTRIP_LOGIC</td>
<td>localId: null</td>
</tr>
<tr>
<td>V_R_O_clean</td>
<td>All</td>
<td>name: TRIP_LOGIC</td>
<td>localId: null</td>
</tr>
<tr>
<td>name: V_R_O_PTRIP_LOGIC</td>
<td>All</td>
<td>name: V_R_O_TSP</td>
<td>localId: null</td>
</tr>
<tr>
<td>name: V_R_O_PTRIP_LOGIC</td>
<td>All</td>
<td>name: V_R_O_PTSP</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: MDL_E</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: ALE</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: PTRIP</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: 1</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: 0</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: TRUE</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: FALSE</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: HYS</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: 60</td>
<td>localId: null</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: PTRIP_LOGIC</td>
<td>localId: 116</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: TRIP_LOGIC</td>
<td>localId: 143</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: AND_BOOL_2</td>
<td>localId: 25</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: AND_BOOL_2</td>
<td>localId: 45</td>
</tr>
<tr>
<td>pouName: V_R_O_clean</td>
<td>All</td>
<td>name: AND_BOOL_2</td>
<td>localId: 48</td>
</tr>
</tbody>
</table>
FBDChecker - Results

- An example of a part of diagram in a logic
  - Too far block
  - Crossed line
  - Type conversion

```
if selection func INT type and input type BOOL didn't matching, using type casting function is needed
SEL_INT_2 and SEL_INT_2 is too far
SEL_INT_2 cross line not good at seen
```
CONCLUSION
Conclusion

- **FBD programming guidelines**
  - 5 specific cases
  - more guidelines in current status
  - an automatic guideline checker (FBDChecker)

- **Future work**
  - improving quality and quantity of guidelines
THANK YOU

Q & A

Dong-Ah Lee
Dependable Software Lab.
Konkuk University
ldalove@konkuk.ac.kr