FBD로부터 C프로그램 코딩 가이드라인을 준수하는 C코드 변환규칙

송승현° ,이동아,유준범

건국대학교 공과대학 컴퓨터공학부

union93@konkuk.ac.kr, ldalove@konkuk.ac.kr, jbyoo@konkuk.ac.kr

C code conversion rules complying with C program coding guidelines from FBD

SeungHyeon Song° , Dong-Ah Lee, Junbeom Yoo

Division of Computer Science and Engineering College of Engineering, KONKUK University

요 약

원자력 발전소와 같은 안전필수 시스템은 안전을 최우선하여야 한다. 따라서 안전시스템에 빈번하게 사용되는 PLC의 소프트웨어 구현은 프로그래밍언어 코딩 가이드라인을 준수해야한다. 본 논문에서는 FBDtoC도구의 변환규칙이 안전시스템 C언어 코딩 가이드라인 NUREG/CR-6463을 준수하여 변환하는 규칙을 제안한다. 이를 위해 기존의 연구에서 사용된 FBDtoC도구의 변환결과물과 NUREG/CR-6463을 분석하여 가이드라인 위반사항을 검출하고 변환규칙이 위반된 가이드라인을 준수하도록 개선하고 이를 적용한 도구를 개발하였다.

1. 서 론

원자력 발전소와 같은 안전필수 시스템은 안전이 최우선 사항이다. 시스템의 안전을 위해서는 시스템에 사용되는 PLC의 소프트웨어는 안전 코딩 가이드라인을 준수하여야 한다. PLC는 제어 시스템 구현에 사용되는 산업용 컴퓨터이다. PLC는 FBD(Function Block Diagram)[1]로 디자인되며, PLC의 소프트웨어 구현은 FBD로 작성된 프로그램을 다시 C언어 프로그램으로 변환하여 구현한다. 일반적으로 FBD로부터 C코드의 변환은 도구를 사용해서 이루어진다. 이전의 연구에서는 FBD로 설계된 프로그램을 C프로그램으로 변환하는 FBDtoC[2]를 개발하였지만, 안전 코딩 가이드라인을 준수하지 않는 문제가 존재한다. 따라서 기존의 FBDtoC도구의 자동변환규칙이 C코딩 가이드라인을 준수하도록 개선할 필요가 있다.

본 논문에서는 FBD로부터 코딩 가이드라인을 준수하는 C 코드를 생성하기 위해 몇몇 변환 규칙을 개선하여 제안한다. 이를 위해서 RPS(Reactor Protection System)에 사용된 FBD를 FBDtoC도구로 변환하여 코딩 가이드라인을 준수하는지 분석하고 이를 준수하도록 하는 규칙을 제안한다. 또한 본 논문에서는 이를 위한 도구를 개선하였다. 본 논문은 다음과 같이 구성된다.

2장에서 가이드라인에 관한 배경지식을 서술한 후 3장에서 샘플이 가이드라인에서 위반한 사항을 분석한다. 4장에서는 FBDtoC도구가 위반된 가이드라인 또한 준수하도록 개선사항을 제시한다.5장에서는 사례연구를 통해 실제 RPS FBD가 어떻게 개선되었는지 확인한다. 마지막으로 6장에서 결론 및 향후 연구에 대해 언급한다.

2. NUREG/CR-6463

NUREG/CR-6463[3]은 원자력 시스템의 구현에 사용되는 프로그램 언어에 대한 가이드라인으로 각 언어별 코딩 가이드라인과 해당 가이드라인을 준수하지 않을 시 발생하는 이슈를 기술하고 있다. 또한 올바르지 못한 코딩 스타일과 가이드라인을 준수한 코딩 스타일을 예시를 통해 보여주고 있다. 코딩 가이드라인은

1. Reliability : 코드의 신뢰성
2. Robustness : 코드의 내구성
3. Traceability : 코드의 추적성
4. Maintainability : 코드의 유지 보수성

카테고리로 가이드라인을 구분하고 있다. Reliablilty 파트는 코드가 주어진 조건에서 올바르게 작동하도록 가이드라인을 제시하며, Robustness 파트는 예상치 못한 상황에 대하여 코드가 대응 할 수 있도록 한다. Traceability 파트는 해당 코드가 어떤 기능, 목적에서 작성되었는지 명확하게 할 것을 요구한다. 마지막으로 Maintainability 파트는 코드 작성이후 유지 보수를 위한 가이드라인을 제시하고 있다.

본 연구에서는 NUREG/CR-6463에서 C프로그래밍에 관한 모든 카테고리에서 코딩 가이드라인을 준수하도록 FBDtoC도구의 변환규칙을 개선한다.

3. FBDtoC 변환결과물의 코딩 가이드라인 위반 사항 분석

본 연구에서는 "FBDtoC"의 결과물(C 프로그램 코드)을 변환결과물과 가이드라인의 샘플을 수동으로 확인하였다. 위반된 가이드라인 중 프로그램의 실행과 컴파일에 영향을 주는 경우 *실행*으로 구분하였으며, 코딩 스타일이 수정되어야 하는 경우 *스타일*로 구분하였다. 이 밖에 FBD to C 변환 규칙과 위반사항이 무관할 경우 *외부*로 분류하였다. 분석 결과 FBD to C를 통해 작성된 C프로그램은 표1 과 같이11개의 가이드라인을 준수하지 않고 생성되는 것을 확인하였다. 변환된 결과물은 11개의 위반 사항을 제외하면 컴파일 혹은 실행에 영향을 주지않거나, RPS 시스템에서 사용되지 않는 명령어에 대한 가이드라인으로 확인되었다.제시된 위반사항 중 4.1.1.3과 4.1.2.6, 4.4.5.1은 실제 컴파일/실행에 영향을 주는 항목으로 3절에서 개선 필요성과 개선방안에 대하여 설명한다. 이외의 항목은 유지, 보수에 필요한 가독성, 명확성에 관한 가이드라인으로서 컴파일,실행에 지장을 주지는 않으나 향후 유지 보수를 위해 가이드라인의 형식으로 작성될 필요가 있다.

표1 NUREG/CR-6463 가이드라인 위반 사항

가이드 번호	내용	위반 유형
4.1.1.3	루틴을 지나는 매개변수 제어	실행
4.1.2.6	데이터 타입 사용 제어	실행
4.1.2.7	부동소수점의 정확도와 정밀도	실행
4.2.2.1	예외의 로컬 처리	실행
4.2.2.2	외부 제어흐름의 보존	실행
4.3.3	버전 관리자 툴의 사용	외부
4.4.1.2	식별자 이름 설명	스타일
4.4.1.3	주석 및 내부 문서	스타일
4.4.1.6	모호하거나 미묘한 프로그래밍 구조 회피	스타일
4.4.1.8	리터럴 사용 최소화	스타일
4.4.5.1	플랫폼 기반의 언어 사용자제	스타일

4. NUREG/CR-6463의 가이드라인을 준수하는 변환 규칙 제안

확인된 11개의 위반사항을 준수하도록 FBD to C도구의 변환 규칙을 개선하였다.

표2 FBD to C NUREG/CR-6463 가이드라인 위반사항 개선 결과

가이드 번호	위반 유형	규칙 개선사항
4.1.1.3	실행	매개변수 31개로 제한
4.1.2.6	실행	Int 타입의 사용을 제한하고 unsigned short 형으로 정수 표현
4.1.2.7	실행	Float 자료형을 Double로 표현
4.2.2.1	실행	데이터 타입에 대한 예외처리 적용
4.2.2.2	실행	외부 제어
4.3.3	외부	버전관리 도구를 사용할 것을 권장하는 지침 작성
4.4.1.2	스타일	변수 및 함수 의 이름 작성시 다른 변수 및 함수와의 차이점부터 작성
4.4.1.3	스타일	주석을 보강하고 내부 문서를 작성하도록 권장 지침 작성
4.4.1.6	스타일	삼항연산자와 같은 유지 보수 시 알아보기 힘든 코딩 스타일을 제거하고 조건문과 같은 식별이 쉬운 형태로 변경
4.4.1.8	스타일	리터럴의 사용을 최소화 하고 가능한 경우 const를 대체하여 사용 리터럴 사용이 강제될 경우 소괄호를 사용하여 구분
4.4.5.1	스타일	Int 타입의 사용을 제한하고 플랫폼에 상관없는 short, long 타입으로 데이터 타입 변경

정수형 데이터 타입을 int 대신 사용하는 데이터 값의 범위에 따라 unsigned short, signed short, unsigned long, signed long, unsigned long long, signed long long 으로 세분화 하도록 개선하였다. 기존의 FBD to C도구는 FBD 프로그램에서 사용되는 모든 정수형 자료를 변환된 C코드에서는 int 데이터 타입으로 사용하였다. int 형 데이터 타입만 사용할 경우 특정 컴파일러 혹은 플렛폼에 의존적이 되어 int 타입 데이터를 지원하지 않는 PLC에서 프로그램이 작동하지 않을 수 있다. 따라서 표 3과 같이 FBD의 정수 데이터타입에 따라 C의 데이터 타입을 INT가 아닌 플렛폼 의존적이지 않은 데이터형으로 변환하도록 규칙을 개선하였다.

표3 FBD 정수,실수 데이터 타입에 따른 C 데이터 타입 변환

FBD 데이터타입		C 데이터 타입
SINT (1 byte)	->	signed short (2 byte)
INT (2 byte)		
DINT (4 byte)	->	signed long (4 byte)
LINT (8 byte)	->	signed long long(8 byte)
USINT (1 byte)	->	unsigned short (2 byte)
UINT (2 byte)		
UDINT (4 byte)	->	unsigned long (4 byte)
ULINT (8 byte)	->	unsigned long long(8 byte)

삼항연산자 대신 if 조건문을 사용하도록 규칙을 개선하였다. 기존의 규칙은 매개변수가 2개인 비교 조건문은 삼항연산자로 표현하도록 설계되었다. 삼항연산자로 비교조건문을 작성할 경우 코딩 길이가 if 조건문 보다 짧아지나 이후 유지보수에서 코드 구조를 파악하는데 착오가 있을 수 있기 때문에 그림1과 같이 if 조건문으로 작성하도록 규칙을 개선하였다.

* CP : Comparison Operator(비교연산자)	
기존의 비교조건함수 작성 규칙	개선된 비교조건함수 작성 규칙

function(A,B){
 	
return (A CP B) A:B ;
 	
}
	function(A,B){
	if(A CP B){
	return A;
	}
	else{
	return B;
	}
	}

그림1 비교조건문 함수 스타일 개선

모든 리터럴은 그 값에 소괄호를 적용하여 모호함이 없도록 개선하였다. 리터럴은 표현식으로 사용될 가능성이 존재하기 때문에 그 값을 일관되게 표현하는 것이 중요하다. 유지보수의 관점에서 소괄호를 통해 값을 분명하게 적시해주는 편이 유리하기 때문에 그림 5와 같이 기존의 리터럴은 소괄호 구분가 없었으나 그림 2와같이 리터럴의 값에 소괄호로 구분하도록 개선하였다.

기존의 리터럴 작성규칙	개선된 리터럴 작성규칙
#define value;	#define (value);

그림2 기존의 리터럴 표현

5. 사례연구

4절의 개선 규칙을 준수하도록 FBDtoC도구를 개선한 결과 RPS FDB 프로그램 FBDtoC변환 결과물이 그림 3,4,5와같이 개선되었다.

<Variable_Rate_Rising> 정수형 데이터 타입 개선	
개선 전	개선 후
int RNG_MAX = 29400; int RNG_MIN = 600;	unsigned short RNG_MAX = 29400; unsigned short RNG_MIN = 600;

그림 3 정수형 데이터 타입 변환이 개선된 사례

<Fix_Falling> 모호한 비교조건문 코드 스타일 개선 & 정수형 데이터 타입 개선	
개선 전	개선 후

bool LT_INT_2(int IN0 , int IN1){
return (IN0 < IN1)? true:false;
}
	bool LT_INT_2(unsigned short IN0 , unsigned short IN1){
	if(IN1 > IN0){
	return true;
	}
	else{
	return false;
	}
	}

그림 4 모호한 비교 조건문과 정수형 데이터 타입의 개선

<Header.h>에 정의된 리터럴 코드의 스타일 개선	
개선 전	개선 후
#define bool int #define true 1 #define false 0	#define bool (int) #define true (1) #define false (0)

그림 5 리터럴 코드의 스타일 개선

해당 사례들을 통해 개선된 변환 규칙이 도구에 반영되었음을 확인 할 수 있다.

6. 결론

이번 연구에서는 기존의 FBD to C 변환도구를 개선함으로써 C 프레임워크 상에서 발생하는 안전 이슈, 특히 데이터 타입과 관련한 이슈들을 해결할 수 있었다. 4.1에서는 INT 데이터타입의 제한을 통해 작성된 PLC 소프트웨어가 플랫폼 의존성을 띄지 않도록 하였다. 4.2와 4.3에서 삼항연산자 혹은 소괄호 없는 리터럴을 개선함으로써 PLC 소프트웨어의 유지보수에 있어 모호하게 인식될 수 있는 코드를 제거하였다. 차후 연구에는 NUREG/CR -6463의 모든 가이드라인을 준수하기 위해 FBD 와 C 언어 간의 변환 규칙 이외에 소프트웨어 메모리 관리와 FBD 단계에서 매개변수 제어와 같은 새로운 규칙 추가를 통해 FBD to C도구의 C프로그램 결과물이 NUREG/CR-6463의 모든 규칙을 연구할 계획이다

Acknowledgement

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업(NRF-2017M3C4A7066479)과 과학기술정보통신부 및 정보통신기획평가원의SW중심대학지원사업(No.2018-0-00213, SW중심대학(건국대학교))의 지원을 받아 수행된 연구임

참 고 문 헌

[1] IEC, International Standard for Programable Controllers: Programming Languages (Part3), 2003

[2] JunBeom Yoo, Eui-Sub Kim, Jang-Soo Lee, A Behavior-Preserving Translation From FBD Design To C Implementation For Reactor Protection system software,Nuclear Engineering And Technology, Vol.45 No.4 August 2013

[3]H. Hecht, M. Hecht, S. Graff, W Green, D. Lin, S. Koch, A. Tai, D. Wendelboe, Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems, June 1996