2002 춘계학술발표회 논문집
한국원자력학회

SMART 성능해석기 개발

Development of SMART NPA

이규형, 김희경, 이영진, 윤한영

한국원자력연구소
대전시 유성구 덕진동 150

요 약

SMART-NPA(Nuclear Plant Analyzer)는 SMART의 실시간 모의 분석 프로그램인 TASS/SMR에 GUI(Graphic User Interface) 기능을 추가하여 분석자가 SMART의 운전 상태를 그림으로 볼 수 있도록 해주며 마우스나 키보드를 통해서 원하는 명령문을 입력시키거나 여러 가지 제어를 할 수 있도록 개발된 프로그램이다. 이 프로그램의 계산 엔진인 TASS/SMR은 Fortran으로 작성되어 있고 사용자와의 일차적인 Interface 부분은 TASS/Win이라고 하며 Visual C++(MFC)로 되어 있다. SMART-NPA는 Visual Basic을 사용하여 작성된 ActiveX Control 형태로 개발되어 있으며 Interface Function을 통해서 TASS/SMR을 둘러싸고 있는 TASS/Win과 변수값을 주고받도록 되어있다. SMART-NPA는 Overview, Primary, Secondary, PRHRS, Control Panel 의 5개 주 화면이 Tab Control로 분석자가 편리하게 선택하여 볼 수 있으며 주요 변수의 거동을 그래프로 보여주는 화면도 포함하고 있다. 개발된 SMART-NPA 프로그램은 계산 부분인 TASS/SMR 코드의 계산값과 비교하여 일치함을 확인하였다

Abstract

SMART-NPA is the second user interface part of TASS/SMR in order to improve GUI(Graphic User Interface). Using SMART-NPA the analyzer not only can see the running status of SMART but control SMART. TASS/SMR, the calculation part, was written in Visual Fortran whereas the first user interface part, called TASS/Win, was written in Visual C++. For these reason the ActiveX control was the solution of SMART-NPA development. The five ActiveX controls were built in Visual Basic. They were Overview, Primary, Secondary, PRHRS and Control Panel ActiveX controls. They were contained in tab control, and can easily selected by user. They could communicate with TASS/Win using many interface functions. The graph screens were also developed for the display of major variable's trend. The integrity of SMART-NPA was verified through the comparision with TASS/SMR calculation results. The comparison show the same result, which reflects SMART-NPA is verified

1. 서론

일체형 원자로인 SMART (System-integrated Modular Advanced ReacTor)의 성능 및 안전

성을 해석하는데 사용자의 편의성 제공 및 운전원의 교육 등의 목적으로 사용하기 위하여 SMART-NPA (Nuclear Plant Analyzer)를 개발하였다. SMART-NPA 는 계산 모듈, 윈도우 모듈, Active X 모듈로 구성되어 있다. 계산 모듈은 SMART 의 고유한 특성을 분석하기 위하여 새로이 개발된 모델을 장착한 TASS/SMR (Transient And Setpoint Simulator / Small and Medium Reactor) 코드이다. 윈도우 모듈에는 사용자의 입력과 출력을 윈도우 환경에서 처리할 수 있도록 TASS/Win 환경이 구현되어있다. 5 개의 Active X 모듈은 tab 콘트롤 방식으로 각각 독립된 화면에 구성되어 있다. 사용자는 SMART-NPA 화면으로부터 계통의 주요 거동을 볼 수 있으며 각종 주요 밸브의 개폐, 펌프의 정지/기동 및 제어봉의 삽입/인출을 할 수 있도록 구성되어 있다.

Visual Fortran 으로 만들어진 TASS/SMR 코드와 윈도우 환경에서 사용자 입출력을 가능하게 하는 TASS/Win 프로그램간의 데이터 공유를 위하여 공유 메모리 (shared memory)가 설정되었다. 윈도우 프로그램으로부터 값을 받아 동적인 모양을 표현하기 위하여 연계함수 (interface function)가 TASS/Win 과 SMART-NPA 간에 설정되었다.

개발된 SMART-NPA 프로그램의 건전성을 확인하기 위하여 TASS/SMR 코드에서의 계산값과 SMART-NPA에서 계산된 값을 비교하여 일치함을 확인하였다. SMART-NPA 사용자 설명서가 작성되었으며 주요계통의 변수값 변화가 모의되었다.

2. TASS/SMR 코드 설명

TASS/SMR 코드는 SMART 원자로의 성능 및 안전성을 분석하기 위하여 개발된 해석 코드[1]이다. TASS/SMR 코드에는 SMART 원자로의 주요한 거동인 나선형 증기발생기에서의 열전달 모델, 가압기의 중앙공동과 잔열제거계통의 보상탱크에 존재하는 비응축성가스인 질소의 거동, 초기 정상상태를 자동으로 초기화시켜 주는 루틴, 잔열제거계통의 튜브에서의 열전달 모델 등이 있다.

TASS/SMR 코드는 사용자의 입출력 편의성을 제고하기 위하여 다양한 제어문을 사용하여 프로그램 진행을 제어할 수 있다. 사용 가능한 명령문은 표 1 에 나와 있다.

3. TASS/Win 코드 환경

TASS/SMR 코드는 Visual Fortran compiler를 사용하여 생성된 콘솔용 프로그램이다. 사용자의 입력 편의성과 실시간으로 계통거동을 관찰하기 위하여 콘솔용 프로그램을 윈도우 환경으로 전환이 필요하다. 콘솔용 프로그램에서 윈도우 환경에서의 변수값 전달을 위하여 TASS/Win 프로그램이 개발되었다.

TASS/Win 프로그램은 C++ 언어를 사용하여 윈도우 환경을 설정하고 있으며 Fortran 과 변수

값을 공유하기 위하여 공유 메모리 (shared memory) 방식을 채택하고 있다. NPA_WRITE_SHARED_MEMORY() 함수를 이용해서 공유 메모리의 값을 지정한 내부변수(Fortran에서 사용)에 직접 써넣는 방식을 사용한다. 이때 중요하게 사용되는 함수 2가지가 있다. 내부 변수 (Fortran에서 사용)의 값을 공유메모리로 가져오기 위한 SHARED_MEMORY_OVERVIEW() 함수와 공유메모리의 값을 내부 변수로 써넣기 위한 NPA_WRITE_SHARED_MEMORY() 함수이다. 이 두 함수는 shared_memory_overview.f90 코드에 FORTRAN 서브루틴으로 만들어져 있는 것을 C++ 코드 쪽에서 외부 함수로 선언하여 void 형 함수의 형태로 사용되어진다. 대화상자의 값을 변경하여 OK 버튼을 클릭하면 NPA_WRITE_SHARED_MEMORY() 함수를 사용하여 내부 변수에 값을 써넣고 있다.

C++ 프로그램에서 TASS/SMR 코드의 주 프로그램인 TASSEXEC를 부프로그램으로 변경하였다. 공유메모리 선언과 TASSEXEC의 변경은 다음과 같이 이루어졌다.

◎ **Fortran Subroutine 선언**

○ 선언위치 : Drtass1.f90

```
SUBROUTINE TASSEXEC (COMMAND, TASSFLAG)
CHARACTER(*), INTENT(IN) :: COMMAND
INTEGER*4 TASSFLAG [VALUE]
...
END SUBROUTINE TASSEXEC
```

◎ **C++ 함수선언**

○ 선언위치 : MainFrm.cpp

```
extern "C"
{ void __stdcall TASSEXEC ( char *command, unsigned int command_len, int tassflag); }
```

◎ **C++ 함수호출**

○ 호출위치 : void CMainFrame::RunningCommand()

```
void CMainFrame::RunningCommand()
{
        char command[132];
        int command_len;
        ...
        TASSEXEC(command, command_len, Flag);
        Flag = 1;
        ...
}
```

상기의 코드에서 함수 TASSEXEC은 두 개의 인자를 넘겨받도록 되어 있는데, COMMAND 인자에는 사용자가 입력한 명령문이 문자열 형태로 저장되어 있고, TASSFLAG 인자에는 TASSEXEC를 호출하는 목적이 TASS/SMR 코드의 초기화 목적(TASSFLAG = 0 인 경우)인지 아닌지(TASSFLAG = 1인 경우)를 나타내주는 값이 들어 있다.

즉, command 파일로부터의 입력인 경우는 그대로 입력받도록 하고, 표준 입력으로부터의 입력

인 경우는 TASSEXEC 함수의 인자인 COMMAND 인자의 값을 COMMON 변수인 CMD에 저장하여 이를 상기와 같이 parsing에 이용하도록 하였다.

이와 같이 사용자로부터의 명령문을 받은 이후에는 그림 1과 같이 내부적으로 TASSEXEC 함수 내에서 처리되도록 하였다. TASS/Win 프로그램을 실행시키고 나서 TASSEXEC 함수가 주메뉴에 있는 명령문 메뉴의 'Initialize'를 이용해 호출되었을 경우에는 TASSEXEC 함수의 인자인 TASSFLAG는 0의 값을 가지고 각종 초기화만 수행된다. 그리고 만일 작업 디렉토리 내에 autotass._cd 파일이 존재할 경우에는 이 command 파일내의 명령어들도 batch 형태로 처리하도록 되어 있다.

그리고 초기화가 수행된 이후 요청되는 명령어의 경우에는 TASSFLAG는 1의 값을 가진다. 이 때 'GO'를 제외한 다른 명령어의 경우 '명령문 실행'이라는 부분에서 EXECMD 함수를 통해 처리된 후 C++ 코드로 다시 돌아가고, 'GO' 문의 경우 그림 1의 문 번호 20번 이후의 부분에서 정해진 계산 시간에 도달할 때까지 반복적으로 수행된다. 이 때 'DO' 명령을 이용한 수행중이면 문 번호 16으로 다시 돌아가 다음 명령을 수행하고, 그렇지 않은 경우 TASSEXEC을 끝내고 C++ 코드로 다시 돌아오는 제어 구조를 가지고 있다. 기존 도스 버전의 TASS/SMR 코드에서는 상기와 같이 Visual C++로 돌아가는 구조를 가지고 있지 않고, 도스창에 '?' 프롬프트를 표시하고 다음 사용자 명령을 기다리는 형태로 구조가 이루어져 있다.

상기와 같은 제어 구조는 DRTASS 코드의 수정을 최소화한다는 원칙아래 설정된 구조이다. 따라서 추후 DRTASS 코드에 대한 심층 분석을 수행하고 C++ 코드의 특성을 최대한 반영하여 DRTASS 코드를 구현할 경우에는 다음과 같이 제어구조가 수정될 것이다. 'DO' 명령어나 'GO' 명령어 처리시 그림 1과 같이 Fortran 코드에서 모든 처리를 다하고 중간 중간에 나오는 메시지만 C++ 함수(OutputWindow)를 통해 중간 결과를 표시하는 제어구조가 아닌, 실제 C++ 코드가 명령어 처리 루프를 수행하면서 중간 결과도 생성하고, 프로그램 수행 제어권도 가지게 되는 형태로 수정될 것이다.

윈도우 환경에서 TASS/SMR코드의 제어구조와 GUI(Graphic User Interface)는 그림 1과 그림 2에 나와 있다.

4. SMART-NPA 환경

NPA에서 보여줄 모델들은 TASS/SMR에서 계산되어진 값들을 받아 분석자가 쉽게 이해할 수 있도록 현재 상태를 그림으로 나타낸다. 이러한 모델을 구현하기 위하여 Visual Basic을 사용하였고 구현되어진 모델은 Active X Control 형태로 다른 언어로의 이식성과 재 사용성을 높였다.

현재 NPA 의 구조는 5 개의 모델을 담고 있는 NPA_Mother.ocx와 각각의 모델들 (NPA_PRHRS.ocx, NPA_Overview.ocx, NPA_Primary.ocx, NPA_Secondary.ocx, NPA_ControlPanel.ocx) 로 구성되어 있다. TASS/NPA 는 여러 개의 모델을 해석자가 편리하게

선택할 수 있도록 탭 콘트롤을 사용하여 각 모델간의 전환이 이루어지도록 하였다.

Fortran 쪽에서 계산되어지는 값을 구조체 (Labeled Common)에 쓰고 C++쪽에서 그 값을 구조체로부터 읽어서 NPA 모델에 전달해준다.

이와 같은 외부 구조체를 선언하여 Fortran에서 선언한 변수에 접근할 수 있다. 현재 C++ 코드에서는 타이머를 사용하여 주기적 (0.2초) 으로 값을 읽어서 NPA용 Active X control에 그 값을 전달한다.

공유 메모리를 통하여 TASS/SMR 코드의 내부 변수값을 갖고 있는 윈도우 환경에서의 변수값은 SMART-NPA 의 interface 함수를 통하여 값이 전달되며 NPA_Mother.ocx는 이 값을 각각의 모델에 전달한다.

각각의 interface 함수는 보여질 값을 필요로 하는 모든 NPA 콘트롤상의 요소에 대하여 제작되었다. 제작된 interface 함수의 예는 다음과 같다. m_NPA 객체는 NPA_Mother 컨트롤이 부모 클래스이고 그 내부에 각 모델에 각 모델들이 독립된 컨트롤로 존재한다. 멤버함수를 통하여 값을 전달받은 m_NPA 는 내부 모델 컨트롤들에게 그대로 값을 전달한다.

◎ Interface **함수선언**

○ 선언위치 : NpaDlg.cpp

```
extern "C"
{ .
.
.
m_NPA.SetPrimary_T42_Temp(NPA.Temp_Tot42);
m_NPA.SetPrimary_T43_Temp(NPA.Temp_Tot43);
.
.
m_NPA.SetPrimary_TT1_Press(NPA.Press1);
m_NPA.SetPrimary_TT2_Press(NPA.Press2);
.
.
}
```

5 개의 모델들로의 값의 전달은 NPA_Mother.ocx를 통하여 이루어진다. 윈도우 프로그램이 NPA_Mother.ocx를 로드한다. interface 함수를 통하여 NPA_Mother.ocx로 전달되어진 값들은 각 모델들이 갖는 interface 함수를 이용해서 각 모델들로 전달되어지고 이 값을 이용해서 각 모델들은 현재상태를 해석자에게 그림으로 보여준다. SMART-NPA의 구조를 요약하면 다음과 같다.

[Frotran 코드 계산값] → [C++ NPA용 구조체] → [NPA_Mother.ocx] → [각 모델들] (그림 3)

5. SMART-NPA 실행

SMART-NPA 의 화면은 5 개의 주 화면을 가지고 있다. 5 개는 각각 Overview, PRHRS, Primary, Secondary, ControlPanel 화면으로 구성되었다. 그림 4부터 그림 8은 각각의 화면을 보

여 준다.

생성된 SMART-NPA 프로그램의 건전성을 확인하기 위하여 계산 부분인 TASS/SMR 코드의 계산 값과 SMART-NPA 프로그램 구동을 통하여 얻은 값을 비교하였다. 비교한 대상은 SMART 원자로에서의 안전해석과 성능해석이 수행된 여러 가지 경우이며 모두 일치한 결과를 보여 주었다. 그림 9와 그림 10은 급수관 파단 사고에 대한 TASS/SMR 계산 결과와 SMART-NPA 계산 결과를 보여준다.

6. 결론

일체형 원자로인 SMART (System-integrated Modular Advanced ReacTor)의 성능과 안전성을 분석하는데 해석자의 사용 편의성과 운전원 교육등의 목적으로 SMART-NPA (계통성능해석기) 가 개발되었다. 개발에 사용된 방법은 C++ 언어를 사용한 윈도우 환경에서의 사용자 입출력 개선과 Visual Basic을 사용한 ActiveX control이다. 개발된 SMART-NPA 는 그 건전성을 TASS/SMR 계산 결과와 비교하여 입증하였다.

7. 감사의 글

본 연구는 과학기술부의 원자력 연구개발 사업의 일환으로 수행된 것이다.

8. 참고문헌

[1] 윤한영외, “TASS/SMR 열수력 모델 기술서”, KAERI/TR-1835/2001, 2001.
[2] 김희경, “TASS-NPA 개발”, KAERI/TR-1231/99, 한국원자력연구소, 1999.

표 1 TASS/SMR 코드 명령어 목록

BACKUP	변수(들) 값을 화면 또는 파일에 표시
CLEAR	pending action을 삭제함
DO	command 파일 처리
DUMP	변수(들) 값을 일정한 format으로 화면에 표시함
GO	계산을 진행함
MODEL	계산 진행에 사용할 모델을 지정함
name	변수(들) 값을 저장하거나 화면에 표시함
QUIT	계산을 종료함
RAMP	변수 값을 선형적으로 변화시킴
RECORD	계산되는 변수(들) 의 값을 파일에 적어줌
REST	저장되어 있는 모든 변수(들) 값을 읽어들임
SCREEN	계산 도중에 변수(들)의 값을 주기적으로 화면에 표시함
SET	코드 내부 flag 값을 지정함
SNAP	모든 변수값을 지정된 파일에 적음
TRACE	계산 진행중 변화되는 변수 값을 화면에 표시
WHEN	조건을 지정하여 만족되면 특정한 명령을 수행함

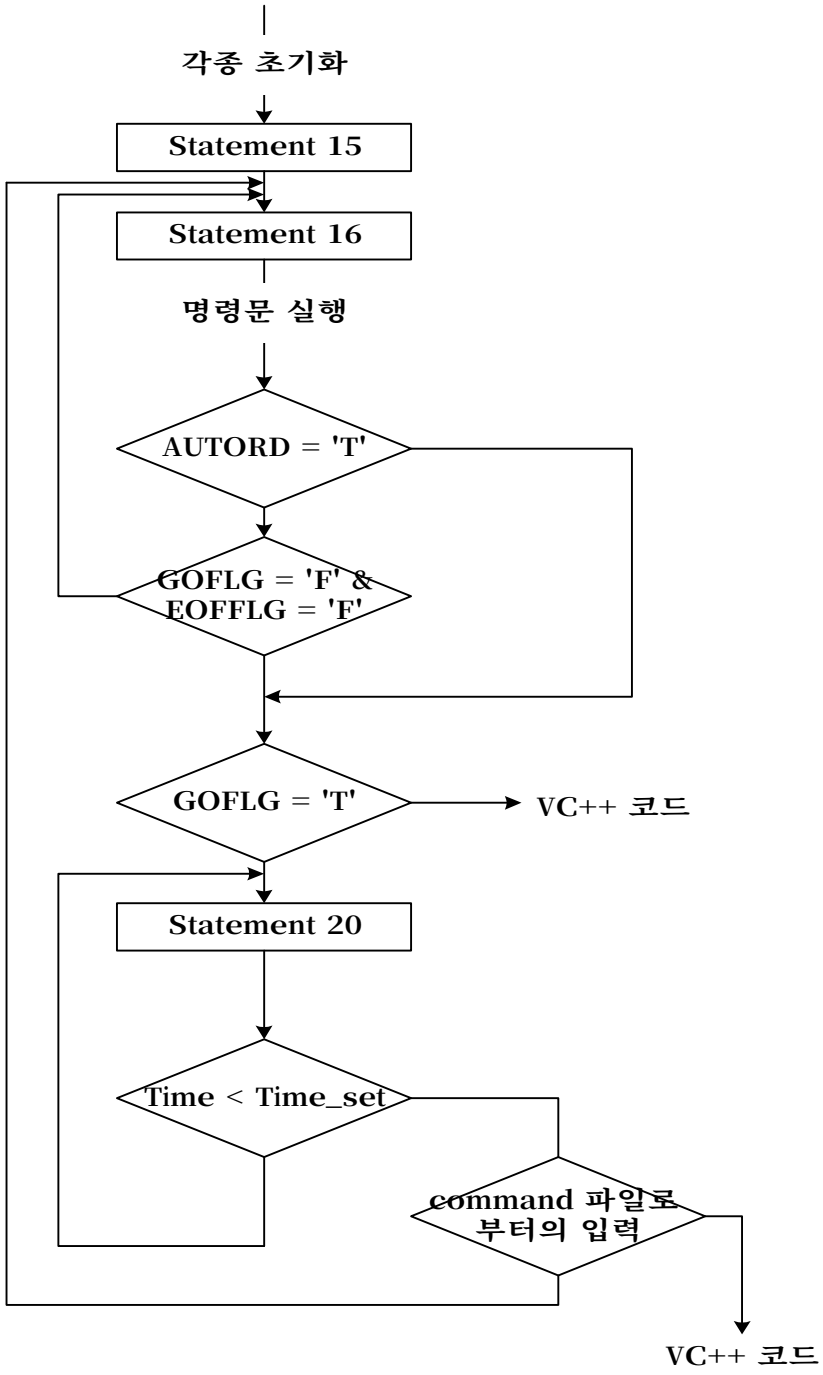

그림 1 윈도우 환경으로 수정된 TASS/SMR 코드의 제어구조

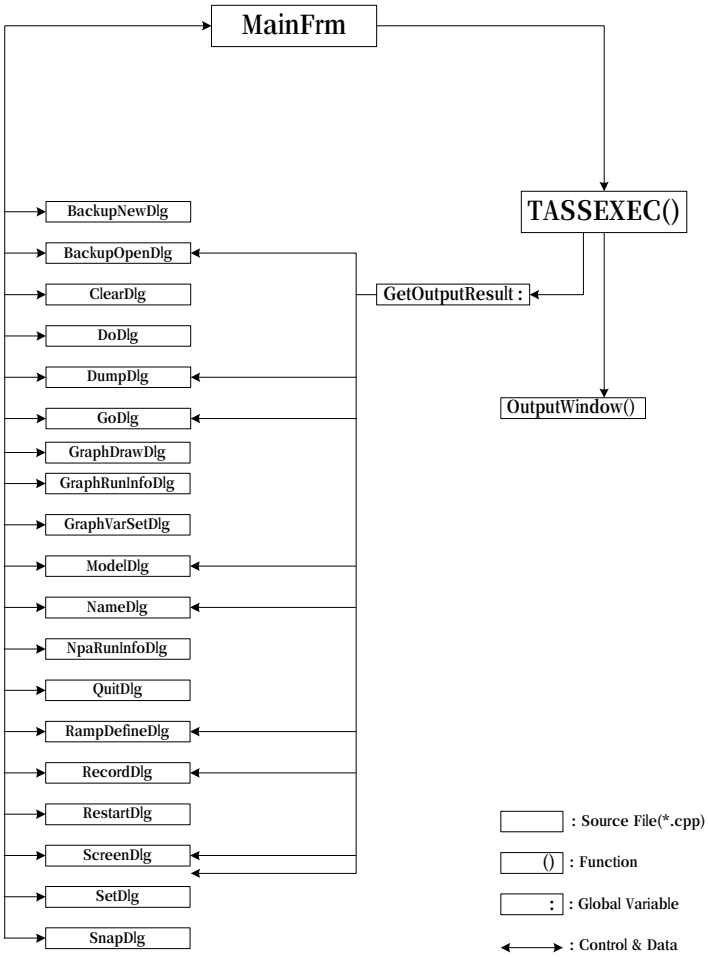

그림 2 윈도우 프로그램의 GUI 구조

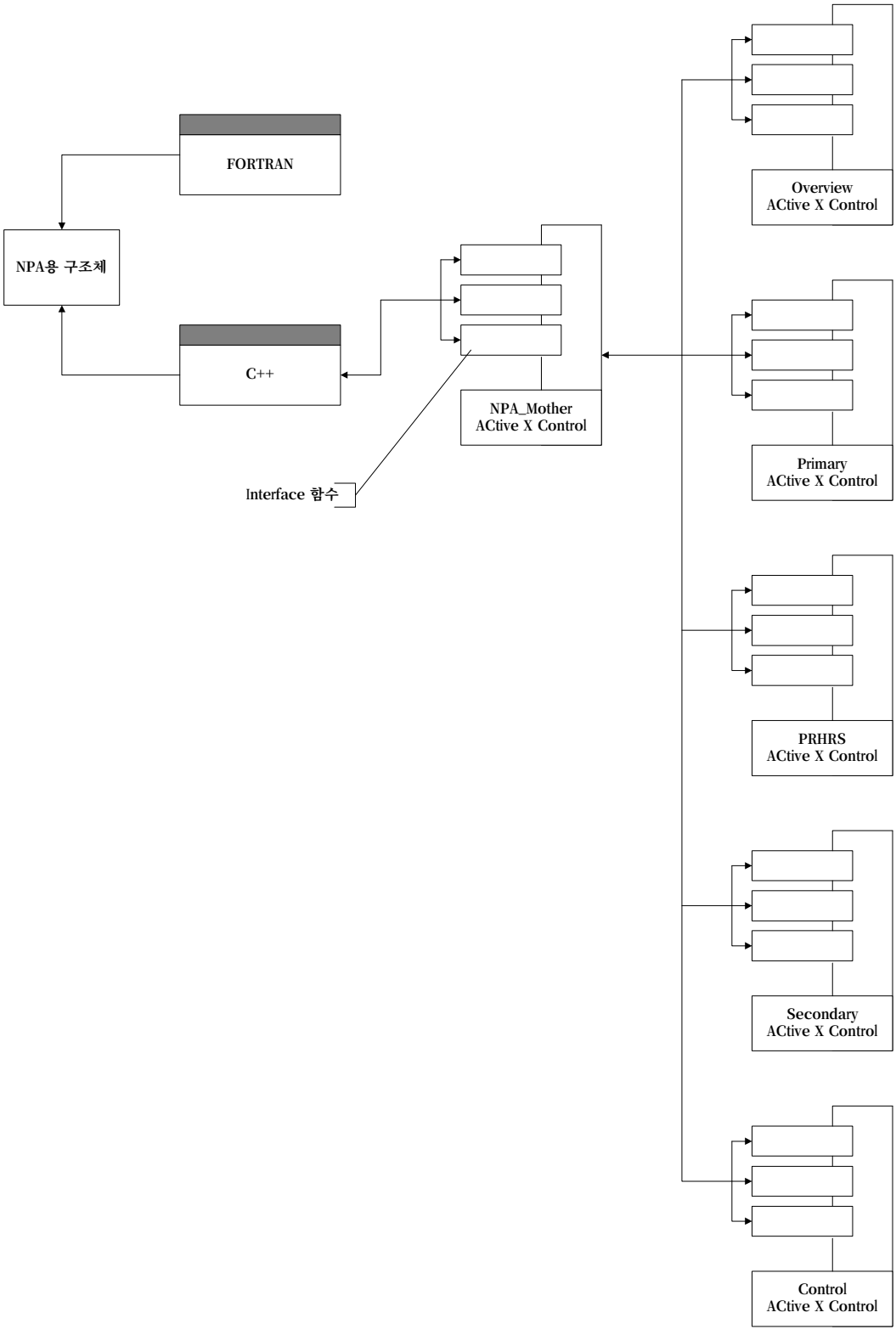

그림 3 SMART-NPA 코드 구조

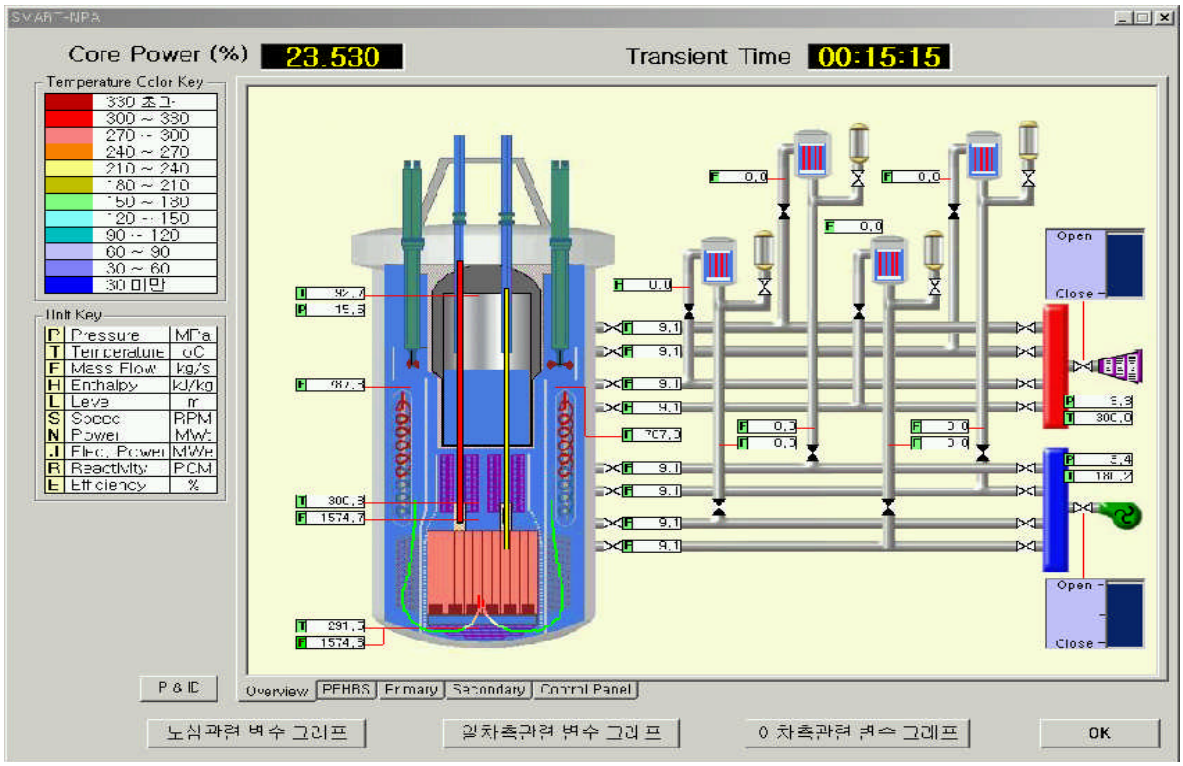

그림 4 SMART-NPA overview 화면 구성

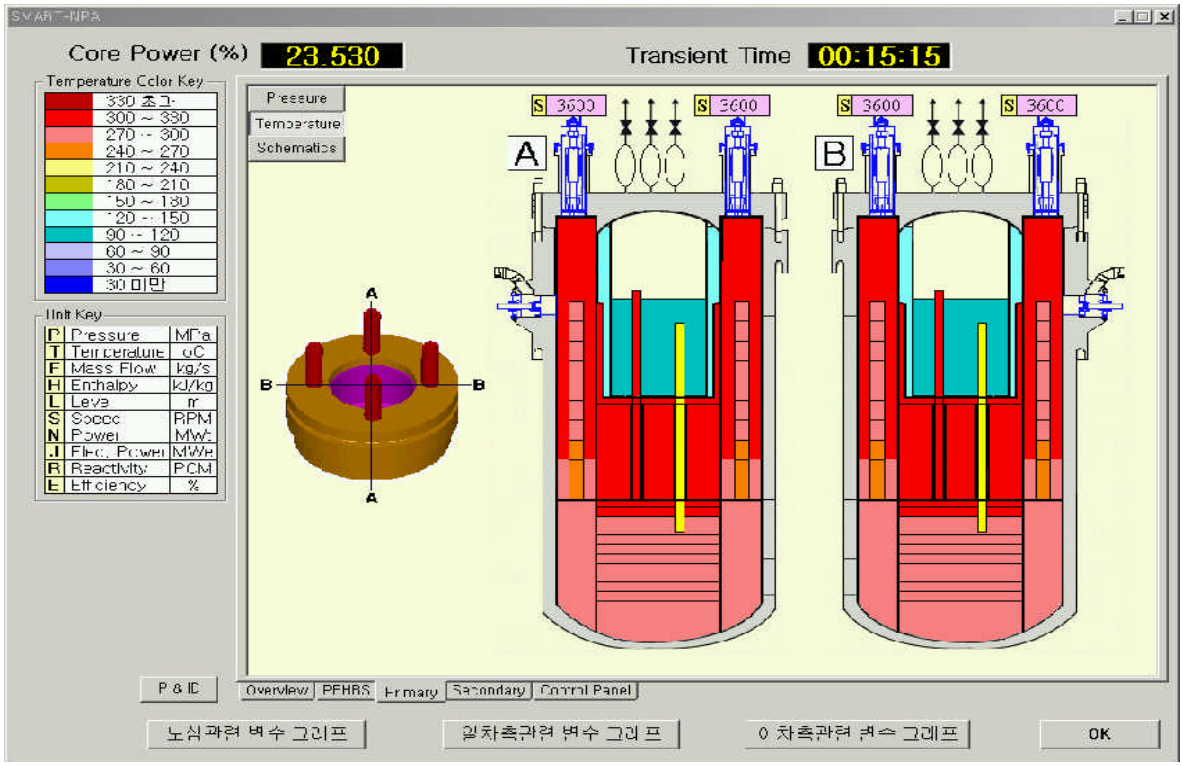

그림 5 SMART-NPA Primary 화면

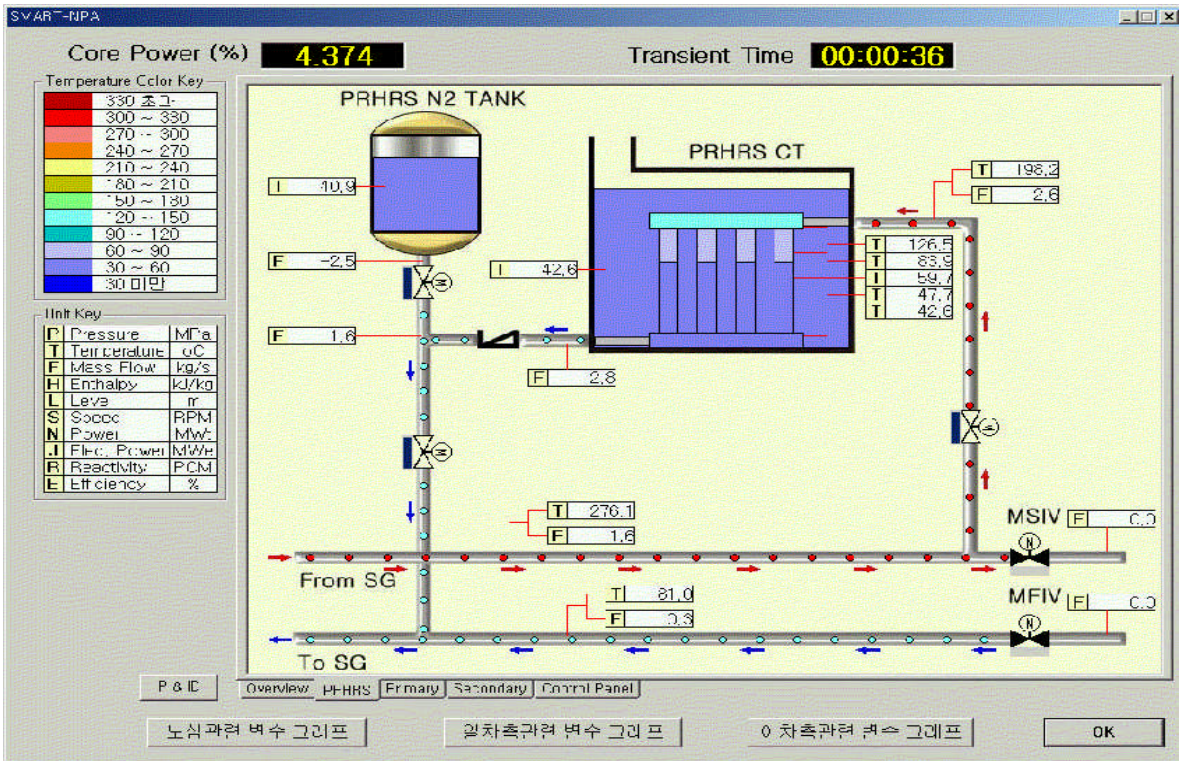

그림 6 SMART-NPA PRHR 화면

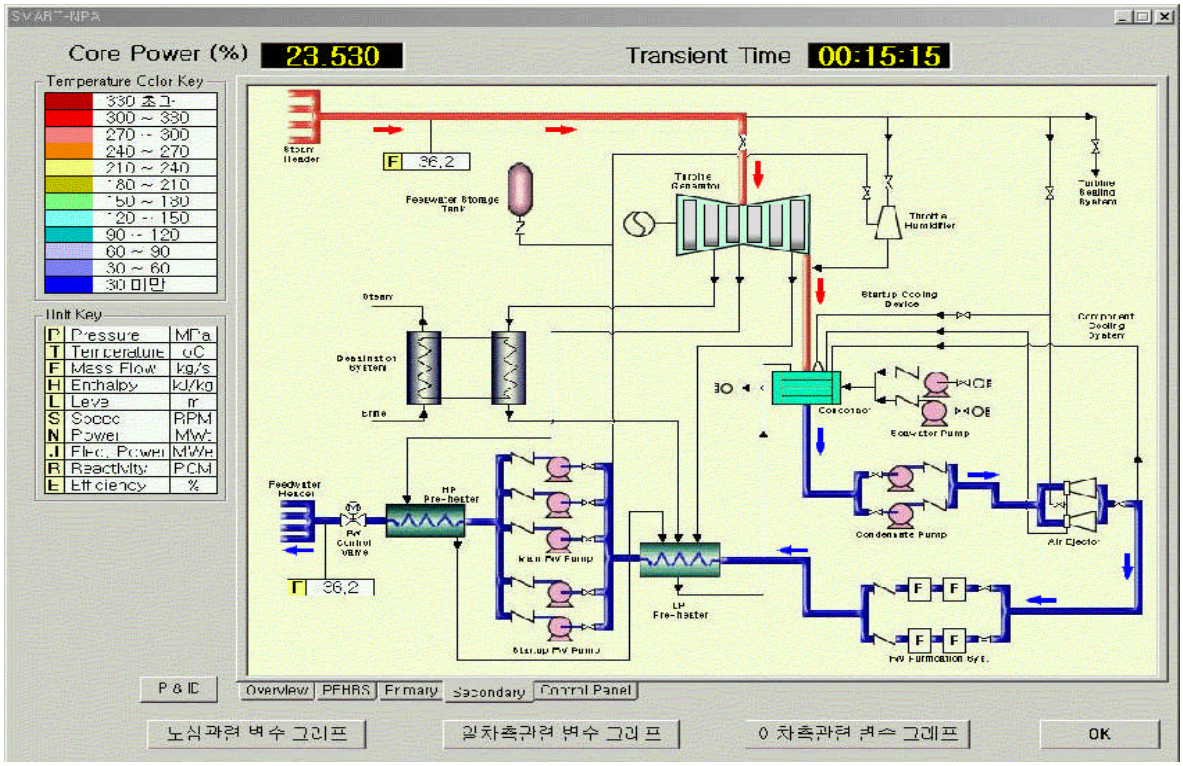

그림 7 SMART-NPA Secondary 화면

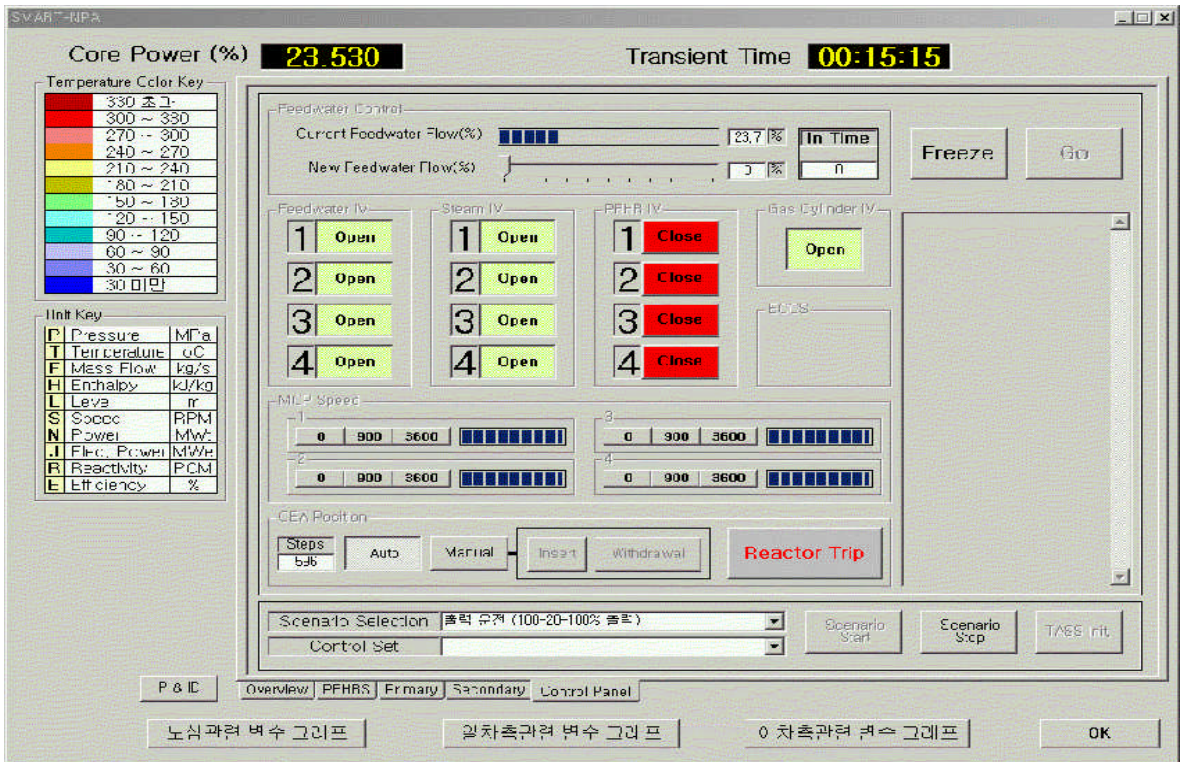

그림 8 SMART-NPA Control Panel 화면 구성

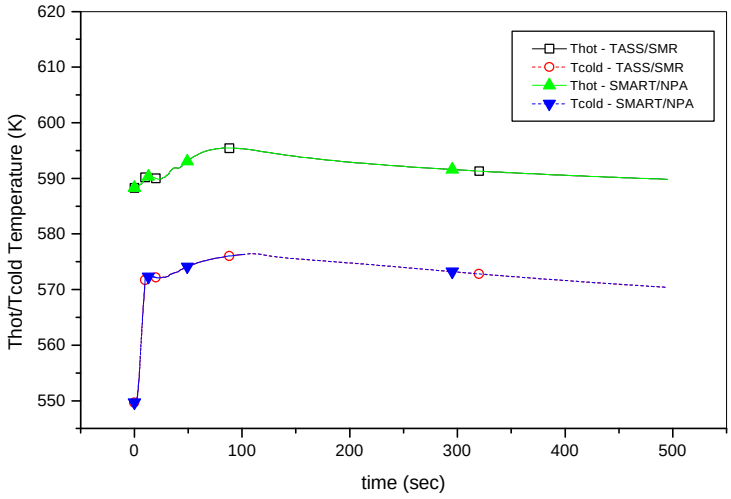

그림 9 급수관 파단사고 냉각재 온도 거동 비교

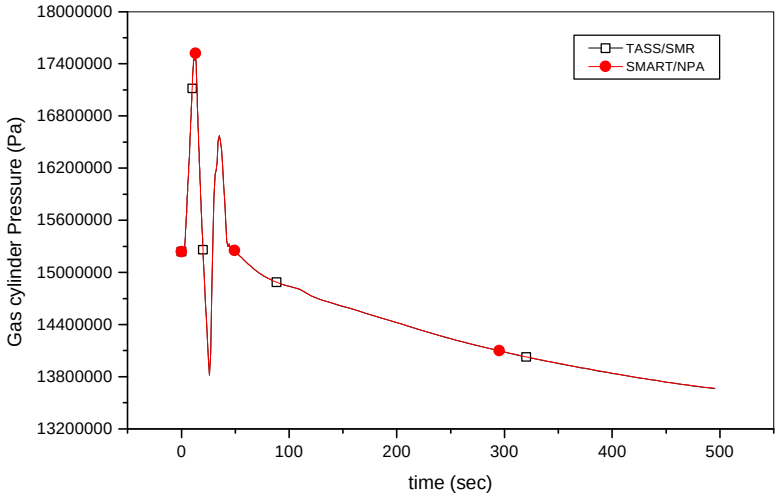

그림 10 급수관 파단사고 가스실린더 압력 거동 비교