A 1ST Step Integration of the Restructured MELCOR for the MIDAS Computer Code

S.H.Park a, D.H.Kim a, S.W.Cho b
a T/H Safety Research Division, KAERI, P.O.Box 105, Yusong, Daejon, 305-600 Korea, shpark2@kaeri.re.kr
b Korea Radiation Technology Institute Co. Kusongdong 19, Yusong, Daejon, 305-353 Korea

1. Introduction

KAERI is developing a localized severe accident code, MIDAS, based on MELCOR. MELCOR uses pointer variables for a fixed-size storage management to save the data. It passes data through two depths, its meaning is not understandable by variable itself. So it is needed to understand the methods for data passing. This method deteriorates the readability, maintainability and portability of the code.

As a most important process for a localized severe accident analysis code, it is needed convenient method for data handling. So, it has been used the new features in FORTRAN90 such as a dynamic allocation for the restructuring[1]. The restructuring of the data saving and transferring method of the existing code makes it easy to understand the code. Before an entire restructuring of the code, a restructuring for each package was developed and tested. And then integration of each restructured package was being processed one by one.

In this paper, the integrating scope includes the BUR, CF, CVH, DCH, EDF, ESF, MP, SPR, TF and TP packages. As most of them use data within each package and a few packages share data with other packages. The verification was done through comparing the results before and after the restructuring.

2. Description Existing Data Structure

The MELCOR code is composed of three parts: MELGEN which checks the input data and makes the data file called restart needed for the calculation, MELCOR which calculates with time by using the restart file and makes a log/plot file, and the PLOT-related programs.

At first the data saving and transfer method is analyzed[2]. Through this process, arrays for four data types (real, integer, logical, and character) are read / written for each package.

MELCOR reserved and used four data types for the data storage and transfer effectively. Data are saved in the most effective way within a fixed array and they are transferred through two steps. During these processes, pointer variables are applied differently according to the packages, and they point out a specific location among the database.

The contents that the pointer variables convey can be searched in the subroutines within the packages. Based on these contents, module is constructed for each package. Also some data were found to be referenced in other packages besides their own packages.

3. Restructuring Steps

Before an entire integrating of the code, a restructuring for an simple package was developed and expanded into the entire MELCOR code[3,4]. The new features in FORTRAN90 such as a dynamic allocation were used for code restructuring. The data transfer structures were modularized from a single array into a derived data type for an easy understanding and usage. Also the variables used in the restructured packages were transformed into a modularized type, and their efficiency was increased through a dynamic allocation. And special program named MeltoMid was made and used to minimize manual job, which helps automatic conversion of variables in subroutines[5].

3.1 Construction of the Module

In MELCOR, the subroutines, connect the pointer variables with the next level subroutines. These next level subroutines then apply the actual-meaning variables instead of the pointer variables. In the restructured and integrated packages, these pointer variables were removed by using the module which was constructed based on the analysis of the pointer variable's usage.

3.2 Subroutine Reorganization

In advance of the restructuring of the subroutines, the whole existing MELCOR code written in FORTRAN77 was transferred into FORTRAN90. The results before and after the conversion were found to be the same.

In the next step, a restructuring of each package was done. In the newly constructed module, all the variables were transformed into direct variables which have a meaning related the package.

The direct variables are used instead of the local variables which are calculated with pointer variables and are passed through an argument.

Lastly, the execution files for each package BURMELCOR, CFMELCOR, MPMELCOR, TPMELCOR, etc., were made.

3.3 Integration of Restructured Packages

The integrating process for MIDAS was done from simple packages to complicated packages. At present, BUR, CF, CVH, DCH, EDF, ESF, MP, SPR, TF and TP packages are integrated. Within them, the data for some packages was shared in many other packages. So, some parts of subroutines in their packages were changed according restructured modules. Later, the whole packages will be integrated.

4. Result and Verification

In order to verify the new results, a three-step process was implemented. At first, a simple language conversion

process from FORTRAN77 to FORTRAN90 was checked by comparing major variables in all the packages before and after the conversion. The values of the major variables were the same. Therefore the language conversion was confirmed to be successful.

The next step was to verify the new results against the results in the first step. As each package was restructured using the module and the derived type variables, the interface program was developed for the data communication between the restructured package and the original packages. The execution files such as BURMELCOR, CFMELCOR, MPMELCOR, TPMELCOR were processed by using the data created at the first step. Also, a temporary program was made for the comparison of the values between the restructured variables and the original values. It provided the same values for the restructured variables through a read/write processing.

The final step, which was the main part here, was to verify the results of the execution file named MIDAS and the results created at the second step. The restructured packages were included into MIDAS. From figure 1 to 2 it is shown comparison of subroutine during restructuring process. As shown in Figure 3 to 4, the integrated results were the same as the package-wised results.

5. Conclusions

To restructure the data storage, the data management process was analyzed for the entire MELCOR code. In this procedure, the data structure was restructured to remove the pointer variables through the FORTRAN90 features such as the MODULE and USE statements.

```
!
!! edit of the ICS  model if active
      IF (IVICPL (1) .NE.0) THEN
!       WRITE (NOW,304) ABS(NMIC),    (ICVNUM(IVICPL(I)),I=1,NMIC    )
        WRITE (NOW,304) ABS(ESF_NMIC),(ICVNUM(IVICPL(I)),I=1,ESF_NMIC)
        WRITE (NOW, 504)
        WRITE (NOW,305) ICVNUM(IVICDN),ICVNUM(IVICSO),ICVNUM(IVICVN),  &
                        ESF_ELVICT,ESF_VNTICN,ESF_VNTICD,              &
                        ESF_VOLIC,ESF_PFICMX,IILTMP,IICNCN,IICDPR
```

Figure 1. Subroutine CNDEDT in restructuring of ESF

```
!
!! edit of the ICS  model if active
      IF (IVICPL (1) .NE.0) THEN
!       WRITE (NOW,304) ABS(NMIC),    (ICVNUM(IVICPL(I)),I=1,NMIC    )
        WRITE (NOW,304) ABS(ESF_NMIC),(CVH_VL(IVICPL(I))%ICVNUM,       &
                                                    I=1,ESF_NMIC)
        WRITE (NOW, 504)
!       WRITE (NOW,305) ICVNUM(IVICDN),ICVNUM(IVICSO),ICVNUM(IVICVN),  &
        WRITE (NOW,305) CVH_VL(IVICDN)%ICVNUM, CVH_VL(IVICSO)%ICVNUM,  &
                                               CVH_VL(IVICVN)%ICVNUM,&
                        ESF_ELVICT,ESF_VNTICN,ESF_VNTICD,              &
                        ESF_VOLIC,ESF_PFICMX,IILTMP,IICNCN,IICDPR
```

Figure 2. Subroutine CNDEDTin MIDAS Integration (1st step)

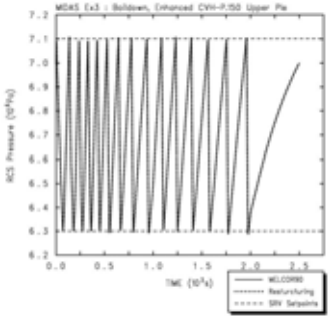

Figure 3. Pressure in Upper Plenum
- MIDAS -

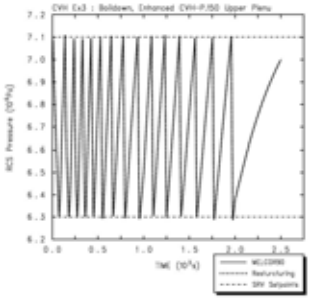

Figure 4. Pressure in Upper Plenum
- Restructuring CVH package only -

Using the reconstructed modules, the subroutines in the target packages were restructured. By comparing the important variables, it was confirmed that the results were the same but differences appeared in the CPU time comparison. They are supposed to be caused by an array transformation in some subroutines. Based on the various results, the output of various executing results, and consulting other experiences, it has been confirmed that the integration of restructuring was done right.

Therefore, the propriety of the integrating was verified. Through the integrating process, the base was constructed for an easy grasp of the variables everywhere in the code and it became easy for a code improvement and for an addition of new models. The integrating process proposed in this paper will be extended to the entire code for the MIDAS development.

REFERENCES

[1] A Multi-Dimensional Thermal-Hydraulic System Analysis Code, MARS 1.3.1, Vol.31, Number 3, pp.344-363, June 1999.

[2] D.H.Kim, S.H.Park, Analysis of MELCOR Code Structure, KAERI/TR-1543/00, June, 2000.

[3] D.H.Kim, et al., Experimental and Analytical Research on Severe Accident Phenomena, KAERI/RR-2216/2001, May, 2002.

[4] S.H.Park, D.H.Kim, S.W.Cho, A Restructuring of the CAV and FDI Package for the MIDAS Computer Code, Proceedings of the Korean Nuclear Spring Meeting, May, 2006.

[5] Y.M.Song, S.H.Park, D.H.Kim, Development of a Computer Program for Automatic Variable Conversion in MELCOR Code, Proceedings of the Korean Nuclear Autumn Meeting, 2000.