ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им.М.В.Келдыша РАН

Бугеря А.Б.

КОНТРОЛЬНЫЕ ТОЧКИ
В ЯЗЫКЕ НОРМА

Москва
2001

Бугеря А.Б.
Контрольные точки в языке Норма.♦

Аннотация

Рассматриваются вопросы определения и реализации контрольных точек языка Норма.

Непроцедурный язык Норма предназначен для автоматизации решения сеточных задач на вычислительных системах с параллельной архитектурой. Этот язык позволяет исключить фазу программирования, которая необходима при переходе от расчетных формул, заданных прикладным специалистом, к программе.

Контрольные точки предназначены для записи во время выполнения программы вычисленных величин и возможности начинать счет не заново, а с последней пройденной контрольной точки.

A.B.Bugerya.
The control points in the NORMA language.

Abstract

The definition and implementation of control points for the Norma language is considered.

The nonprocedural Norma language is a tool aimed for automated solution of the grid-oriented problems on parallel computer systems. This language eliminates the programming phase that is necessary to pass from computational formulas, derived by an application specialist, to a computer program.

The control points may be used for writing the calculated variables during the calculation and then for the ability to start the calculations not from the begin of program, but from the last passed control point.

♦ Работа выполнена при финансовой поддержке Российского Фонда Фундаментальных Исследований (проект № 01-01-00411).

Содержание.

Введение.

Возможность определения в программе на языке Норма [1] контрольных точек предназначена для восстановления вычислительного окружения и продолжения вычислений в случае сбоев оборудования или иных причин, приводящих к снятию программы с счета. Контрольные точки также могут послужить хорошим инструментом при организации счета, требующего много времени, на системе с ограниченным временем выполнения программ. При прохождении вычислительного процесса через контрольную точку все переменные, вычисленные на данный момент и требуемые для дальнейшего счета, сохраняются во внешнем файле. Если в дальнейшем при счете произойдет сбой, начинать вычисления сначала не требуется - все нужные данные для продолжения счета с контрольной точки имеются во внешнем файле. Контрольных точек в программе может быть несколько. Восстановление возможно с последней пройденной из них.

Синтаксис контрольных точек

<Контрольная точка> ::=
CONTROL POINT [(FORMATTED)] <Имя> <Описание>. |
CONTROL POINT [(FORMATTED)] <Где> <Имя раздела или функции>.

<Имя> ::= <Идентификатор>

<Описание> ::=
BEFORE <Список переменных> |
AFTER <Список переменных>

<Список переменных> ::=
<Переменная> {,<Переменная>} [IN ITERATION ON <Индекс с условием> {;<Индекс с условием>}]

<Индекс с условием> ::=
<Индекс итерации>[=<Число>{,<Число>}] |
<Индекс итерации> [EVERY <Число>]

<Индекс итерации > ::= <Идентификатор>

<Переменная> ::= <Идентификатор>

<Где> ::= IN PART | IN FUNCTION

<Имя раздела или функции> ::= <Идентификатор>

Семантика контрольных точек

Все контрольные точки делятся на 2 вида: контрольные точки в текущем разделе или функции (первая альтернатива в синтаксической нотации) и контрольные точки в разделе или функции, вызываемом из текущего (вторая альтернатива). В первом случае определяется собственно контрольная точка, ей дается имя и указывается место и условия ее постановки (см. ниже). Во втором случае только указывается, что вызываемый из текущего раздела другой раздел или функция имеют одну или несколько контрольных точек. В данном случае не указывается имя, и не определяются место и условия постановки контрольной точки. Имя генерируется на основе имени вызываемого раздела и порядкового номера вызова, место постановки – непосредственно перед вызовом раздела или функции, условий нет (ставится всегда).

Если контрольная точка организуется не в главном разделе, то во всех разделах, прямо или косвенно вызывающих данный, должны быть описаны контрольные точки по второй альтернативе. Такая цепочка контрольных точек должна продолжаться вплоть от главного раздела.

Рассмотрим определение места контрольной точки в программе. Так как язык НОРМА является непроцедурным (декларативным) языком, порядок выполнения операторов определяется компилятором. Для того чтобы указать в какой момент времени выполнения программы необходимо создать контрольную точку (обычно это имеет смысл после завершения каких-то достаточно больших вычислений), надо указать список переменных, после вычисления которых (или до) должна быть организована контрольная точка.

После слова BEFORE (или AFTER) идет перечисление имен переменных (величин на сетке, скаляров), которое используется транслятором для определения места контрольной точки. Описание BEFORE <Список переменных> означает, что контрольную точку надо ставить до начала вычисления любой (хотя бы одной) переменной из списка, AFTER - после вычисления всех переменных из списка.

Местом вычисления переменной считается:

1. Оператор ASSUME или скалярный оператор, в котором переменной присваивается какое-то значение.
2. Вызов раздела, в котором переменная стоит после слова RESULT.
3. Итерация или сильно связанная компонента, в которой переменной присваивается какое-то значение.

Если присутствует конструкция IN ITERATION ON и список индексов, то это означает, что контрольную точку надо ставить в теле итерации с индексом из списка индексов, вложенной во все другие итерации из числа итераций с индексами из указанного списка индексов. Контрольная точка в теле этой итерации осуществляется по правилам, изложенным выше.

Контрольная точка будет определена на каждом шаге итерации, если у индекса отсутствуют условия, иначе контрольная точка будет определена

только при заданных значениях индекса. Нулевое или отрицательное значение индекса недопустимо. Если присутствует условие EVERY <Число>, то контрольная точка будет создаваться при значениях индекса, кратному указанному числу (‘на каждом n-ом шаге').

Параметр 'FORMATTED' означает, что файл, в который записываются значения переменных, должен иметь читаемый вид, т.е. можно было бы посмотреть значения сохраненных переменных. Если контрольная точка создается в распределенном разделе, то каждый процессор создает свой собственный файл, в который сохраняет свои переменные, Имя файла строится из имени раздела, в котором производится генерация контрольной точки и, если раздел распределенный, номеров процессоров. Расширение имени файла- '.cp'.

Если контрольная точка попадает после операции вывода или между двумя операциями ввода, то при восстановлении с этой контрольной точки будет пропущена операция вывода, или возможен неправильный ввод данных, о чем будет выдано соответствующее предупреждение.

Особенности организации контрольных точек во внешних разделах.

Если контрольная точка описана как контрольная точка во внешнем разделе или функции, то контрольная точка будет организована перед каждым вызовом этого раздела или функции, и будет создаваться информационный файл, в который записывается, кто вызывает этот внешний раздел и другие параметры вызова для правильного восстановления с контрольной точки во внешнем разделе. Если вызов внешнего раздела или функции попадает в сильно связанную компоненту, то контрольная точка организовываться не будет, о чем будет выдано соответствующее предупреждение. Информационный файл всегда неформатный, имеет имя внешнего раздела или функции, расширение - '.inf'. При заданном режиме 'reliability' (режим повышенной надежности, в этом случае по каждой контрольной точке создаются 2 одинаковых файла, см. ниже) второй информационный файл не нужен, т.к. если сбой произошел при создании информационного файла, при восстановлении он всегда будет создаваться заново.

Если контрольная точка во внешнем разделе или функции, и вызов внешнего раздела или функции производится в цикле, то внутри этого цикла на каждом витке создается файл, в который записываются переменные, вычисляемые в данном операторе и значения переменных цикла. Этот файл имеет имя - имя раздела, который вызывает раздел с контрольной точкой, расширение - '.cp1' (и '.cp2', если был задан режим надежность). При восстановлении с данной контрольной точки уже вычисленные в данном операторе переменные восстанавливаются, и выполненные витки цикла не производятся.

Режим повышенной надежности.

Так как возможен сбой при записи сохраняемых переменных в файл, и при этом теряется его предыдущее содержимое (если были контрольные точки до этого), т.е. восстановиться с предыдущей контрольной точки будет невозможно, то предусмотрено задание режима трансляции 'reliability', при котором используется back-файл. Вначале данные пишутся в основной файл, затем еще раз - в back-файл. Если произойдет сбой при записи в основной файл, предыдущий back-файл испорчен не будет; если при записи в back-файл - значит, основной записан до конца. Back-файл в отличие от основного файла имеет расширение '.cpb'.

Контрольные точки и модули на языке Фортран.

Программа на языке НОРМА может включать в себя отдельные внешние модули, написанные на языке Фортран. Это могут быть подпрограммы чтения данных, подпрограммы, реализующие косвенную адресацию и т.п. вычисления, которые описывать на языке НОРМА нецелесообразно. При организации в такой программе контрольных точек существует следующее ограничение. Если эти внешние модули используют какие-то данные, насчитанные ими же или другими внешними данными ранее, то при организации контрольной точки после того, как эти данные были вычислены, но до того, как использованы, выполнение программы при восстановлении с этой контрольной точки будет неправильным.

Транслятор языка НОРМА ничего не знает об этих данных, о том, что они требуются для дальнейшего счета и что необходимо их сохранять. Поэтому при восстановлении с контрольной точки вызовы, реализующие вычисление данных, выполнены не будут, данные не восстановлены, а модули, использующие эти данные, будут вызваны. Что, соответственно, приведет к неверному выполнению программы.

Для предотвращения описанной выше ситуации необходимо при использовании внешних модулей, написанные на языке Фортран, принять меры для сохранения используемых в них вычисленных ранее величин.

Структура организации контрольных точек в Фортран-программе.

В начале каждого не главного раздела генерируется блок анализа .inf файла. Этот файл должен быть создан вызывающим разделом при вызове данного раздела и содержать информацию о том, какие действия предпринимать: начинать счет или пытаться восстановиться с контрольной точки. Если анализ файла неудачен, или если в нем предписано начинать счет с начала раздела, то непосредственно следующий за этим блоком блок анализа файла с сохраненными величинами обходится, и попытка восстановиться с контрольной точки не производится. Если же в .inf файле предписано пытаться восстановиться с контрольной точки, или в случае главного раздела, начинает выполняться непосредственно следующий за данным блоком блок анализа

файла с сохраненными величинами. И в этом, и во всех остальных блоках, генерируемых в распределенном разделе, успешность выполнения какой-либо операции проверяется при помощи вызова функции синхронизации.

В этом блоке делается попытка найти и прочитать файл с сохраненными величинами. Вначале анализируется основной файл. Из него считывается имя контрольной точки, и управление передается на блок восстановления переменных данной контрольной точки. Из блока восстановления переменных при неудачном восстановлении управление может вернуться обратно и тогда, как и в случае если файл с сохраненными величинами вообще не найден, или если из него не удалось прочитать даже имя контрольной точки, при заданном режиме 'reliability' все то же повторяется с back-файлом. Если режим 'reliability' задан не был, или если и с back-файла восстановиться не получается, счет начинается заново.

Генерация каждой контрольной точки состоит из 2-х блоков. Первым идет блок записи сохраняемых переменных. В начале проверяются условия постановки контрольной точки, если они есть. Если они не выполнены, управление передается на дальнейший счет. Иначе в файл с сохраняемыми величинами записывается имя контрольной точки, параметры вызова (в неглавных разделах) и сохраняемые величины. Если задан режим 'reliability', все то же повторяется с back-файлом. Следующий блок – блок восстановления сохраненных величин – обходится и счет продолжается.

На блок восстановления сохраненных величин возможно передача управления из начала программы – из блока анализа файла с сохраненными величинами. В этом блоке сохраненные величины считываются, и счет продолжается. В случае ошибки при считывании величин управление передается обратно в начало программы для анализа back-файла или счета заново.

Если генерируется контрольная точка во внешнем разделе или функции, то непосредственно после блоков генерации идет блок формирования inf-файла. Соответствующая команда для него (начинать счет заново в вызываемом разделе или пытаться восстановиться) формируется в блоках генерации.

После каждого распределенного раздела с контрольными точками генерируется функция синхронизации. Она служит для определения того, что все подзадачи выполнили какую-либо операцию успешно. Функция вызывается всеми подзадачами с признаком 'успех'/'неуспех' и возвращает 'неуспех', если хотя бы одна подзадача ее вызвала со значением 'неуспех', и 'успех', если все подзадачи вызвали ее со значением 'успех'.

Примеры генерации контрольных точек

Контрольная точка в главном разделе.

```
MAIN PART SUMMA.
!     Вычисление сумм SumV(i) элементов V(i,j) матрицы, распределенной
!    на процессорных элементах (1,1)(1,2);
!   SumV(i) выводится в файл sumv<dat>.
BEGIN
```

```
so:(ts:(t=0..n);ijs:(is:(i=1..n);js:(j=1..n))).
VARIABLE a DEFINED ON ijs.
VARIABLE a1 INTEGER.
VARIABLE b,x,y,xy DEFINED ON is.
DOMAIN PARAMETERS n = 5.
```

Контрольная точка во внешнем разделе.

```
CONTROL POINT IN PART SUMV.
! Вычисление суммы
Oij:(Oi:(i=1..v);Oj:(j=1..w)).
VARIABLE V DEFINED ON Oij DOUBLE.
VARIABLE Vsum,Contr DEFINED ON Oi DOUBLE.
DOMAIN PARAMETERS v=30,w=40.
     FOR Oij  ASSUME V=j+(i-1)*w.
     COMPUTE SUMV(V ON Oij RESULT Vsum ON Oi).
FOR Oi ASSUME
     Contr=SUM((Oj)V).
     OUTPUT Vsum(FILE='sumv',D20.10) ON Oi.
     OUTPUT Contr(FILE='sumv',D20.10) ON Oi.
END PART.

      PROGRAM SUMMA
      INTEGER t,a1,CPCOMM0
      REAL a(5,5),b(5),x(5),y(5),xy(5)
      DOUBLE PRECISION V(30,40),Vsum(30),Contr(30)
      CHARACTER CPFNAME0*15,CPSTR0*9
      DOUBLE PRECISION RF1
```

Блок анализа файла с сохраненными величинами

```
      WRITE(CPFNAME0,40005)
      OPEN(1,FORM='UNFORMATTED',FILE=CPFNAME0,ERR=11,STATUS='OLD')
      READ(1,ERR=11)  CPSTR0
      IF((CPSTR0.EQ.'SUMV0')) GOTO 5
   11 CONTINUE
    2 CONTINUE
      CLOSE(1)

C +++ ОПЕРАТОР 11 +++
      DO     3 j=1,40
      DO     3 i=1,30
      V(i,j)=j+(i-1)*40
    3 CONTINUE
C   КОНТРОЛЬНАЯ ТОЧКА SUMV0
```

Блок записи сохраняемых переменных

```
      OPEN(1,FORM='UNFORMATTED',STATUS='UNKNOWN',FILE=CPFNAME0)
      CPSTR0='SUMV0'
      WRITE(1)  CPSTR0
      WRITE(1)  V
      CLOSE(1)
      GOTO 4
```

Блок восстановления сохраняемых переменных

```
    5 CONTINUE
      READ(1,ERR=2)  V
      CLOSE(1)
      CPCOMM0=0
      GOTO 6
```

Блок формирования inf-файла

```
    4 CONTINUE
      CPCOMM0=1
    6 CONTINUE
C +++ ОПЕРАТОР 12 +++
      OPEN(1,FILE='SUMV.inf',FORM='UNFORMATTED',STATUS='UNKNOWN')
      CPSTR0='SUMMA'
      WRITE(1)  CPSTR0,CPCOMM0,0,0,0,0,0,0,0,0
      CLOSE(1)

      CALL SUMV(V,Vsum)
```

```
C +++ ОПЕРАТОР 13 +++
      DO     8 i=1,30
      RF1 = 0.0
      DO     7 j=1,40
      RF1 = RF1+(V(i,j))
    7 CONTINUE
      Contr(i)=RF1
    8 CONTINUE
C +++ ОПЕРАТОР 14 +++
      OPEN(UNIT=8,FILE='sumv')
      WRITE(8,40001)
      WRITE(8,40002) (Vsum(i),i=1,30)
C +++ ОПЕРАТОР 15 +++
      WRITE(8,40003)
      WRITE(8,40004) (Contr(i),i=1,30)
40001 FORMAT (1X,'Vsum=')
40002 FORMAT ((7(1X,4D20.10/),1X,2D20.10))
40003 FORMAT (1X,'Contr=')
40004 FORMAT ((7(1X,4D20.10/),1X,2D20.10))
40005 FORMAT ('SUMMA.cp')
      CLOSE(8)
      END
```

Контрольная точка в неглавном разделе.

```
PART SUMV.
!     Распределенный раздел:
 !    Вычисление суммы Vsum на двух процессорных элементах.
     V RESULT Vsum
BEGIN
Oij:(Oi:(i=1..v);Oj:(j=1..w)).
VARIABLE V DEFINED ON Oij DOUBLE.
VARIABLE Vsum DEFINED ON Oi DOUBLE.
DOMAIN PARAMETERS v=30,w=40.
DISTRIBUTION INDEX i=1..3,j=1..4.
CONTROL POINT bbb BEFORE Vsum.
     FOR Oi ASSUME
     Vsum=SUM((Oj)V).
END PART.

      SUBROUTINE SUMV(V,Vsum)
      TASK EXTERNAL sumv0
C ----- СТАРТ-ПРОГРАММА SUMV
      DOUBLE PRECISION V(30,40),Vsum(30)
      DOUBLE PRECISION RM1(10,10),RM2(10)
      TASKID ITASKJ(3,4)
      COMMON /ALARM0/IWAIT0
      DATA IWAIT0 /0/
      IF (IWAIT0.EQ.0) THEN
C ----- ОТСЫЛКА МАТРИЦЫ ИМЕН ПРОГРАММ
      DO    22 IPRWWW=1,3
      DO    22 JPRWWW=1,4
      ITASKJ(IPRWWW,JPRWWW)=NEWTASK(sumv0,32*(IPRWWW-1)+JPRWWW)
   22 CONTINUE
      DO    23 IPRWWW=1,3
      DO    23 JPRWWW=1,4
      SEND(ITASKJ(IPRWWW,JPRWWW))ITASKJ
   23 CONTINUE
      END IF
C  ОТСЫЛКА ФАКТИЧЕСКИХ ПАРАМЕТРОВ РАЗДЕЛА
C  Отсылка исходных параметров в матрицу ПЭ по i,j
      DO    26 IPRWWW=1,3
      DO    26 JPRWWW=1,4
      DO    27 IZzWWW=1,10
      DO    27 JZzWWW=1,10
      IIzWWW=IZzWWW+(IPRWWW-1)*10
      JJzWWW=JZzWWW+(JPRWWW-1)*10
      IF ((IIzWWW.LE.30).AND.(JJzWWW.LE.40)) THEN
      RM1(IZzWWW,JZzWWW)=V(IIzWWW,JJzWWW)
      END IF
```

```
   27 CONTINUE
      SEND(ITASKJ(IPRWWW,JPRWWW))RM1
   26 CONTINUE
C  ПРИЕМ РЕЗУЛЬТАТОВ
C  Прием результатов от линейки ПЭ по i (j=1)
      DO    28 IPRWWW=1,3
      RECEIVE(ITASKJ(IPRWWW,1))RM2
      DO    29 IZzWWW=1,10
      IIzWWW=IZzWWW+(IPRWWW-1)*10
      IF (IIzWWW.LE.30) THEN
      Vsum(IIzWWW)=RM2(IZzWWW)
      END IF
   29 CONTINUE
   28 CONTINUE
      IWAIT0=1
      RETURN
      END
      TASK PROGRAM sumv0
C +++++ ПРОГРАММА ДЛЯ ПЭ С НОМЕРАМИ i = 1..3   j = 1..4
      INTEGER CPCOMM0,CPSYNCR0
      DOUBLE PRECISION RFC
      DOUBLE PRECISION V(10,10),Vsum(10),RABM0(10),PRABM0(10)
      CHARACTER CPFNAME0*15,CPSTR0*9,CPSTR1*9
      INTEGER CPVAR0(8),CPVAR1(8)
      TASKID ITASKJ(3,4)
      TASKID IVVVVJ,IWWWWJ
      COMMON /NORMAMESSPASS/ITASKJ,IPRWWW,JPRWWW,IVVVVJ,IWWWWJ
C ----- ПРИЕМ МАТРИЦЫ ИМЕН ПРОГРАММ
      IVVVVJ=MYTASKID()
      IWWWWJ=PARENT()
      RECEIVE(IWWWWJ)ITASKJ
C ----- ОПРЕДЕЛЕНИЕ КООРДИНАТ СВОЕЙ ЗАДАЧИ
      DO    33 IPRWWW=1,3
      DO    33 JPRWWW=1,4
      IF(IVVVVJ.EQ.ITASKJ(IPRWWW,JPRWWW)) GOTO 34
   33 CONTINUE
   34 CONTINUE
   32 CONTINUE
C  ПРИЕМ ФАКТИЧЕСКИХ ПАРАМЕТРОВ РАЗДЕЛА
C  Прием распределенных фактических параметров раздела
      RECEIVE(IWWWWJ)V
      IN00=1
      IK00=10
      JN00=1
      JK00=10
      IN01=1
      IK01=10
      JN03=1
      JK03=10
```

Блок анализа inf-файла

```
      WRITE(CPFNAME0,40001) IPRWWW,JPRWWW
      OPEN(1,FORM='UNFORMATTED',FILE='SUMV.inf',ERR=2,STATUS='OLD')
      READ(1,ERR=2)  CPSTR1,CPCOMM0,CPVAR0
      IF(((IPRWWW.ne.1).OR.(JPRWWW.ne.1))) GOTO 18
      CPSYNCR0=KPSYNC0(1)
      REWIND 1
      WRITE(1)  1
      CLOSE(1)
      GOTO 19
   18 CONTINUE
      CLOSE(1)
      CPSYNCR0=KPSYNC0(1)
   19 CONTINUE
      IF(((CPCOMM0.EQ.1).OR.(CPSYNCR0.EQ.0))) GOTO 3
```

Блок анализа файла с сохраненными величинами

```
      OPEN(1,FORM='UNFORMATTED',FILE=CPFNAME0,ERR=20,STATUS='OLD')
      READ(1,ERR=20)  CPSTR0
      IF((CPSTR0.EQ.'bbb')) GOTO 15
```

```
   20 CONTINUE
    2 CONTINUE
      CPCOMM0=1
      CLOSE(1)
      CALL KPSYNC0(0)
    3 CONTINUE

C +++ ОПЕРАТОР 8 +++
      DO     5 i=IN01,IK01
      RABM0(i)=0.0
      DO     5 j=JN03,JK03
      RFC=V(i,j)
      RABM0(i)=RABM0(i)+RFC
    5 CONTINUE
C       -----ВОЛНА-----
C       --КРАЙНИЕ--
      IF(JPRWWW.NE.1) GOTO 6
      SEND(ITASKJ(IPRWWW,JPRWWW+1),TAG=5)(RABM0(i),i=IN01,IK01)
    6 CONTINUE
      IF(JPRWWW.NE.4) GOTO 7
      SEND(ITASKJ(IPRWWW,JPRWWW-1),TAG=5)(RABM0(i),i=IN01,IK01)
    7 CONTINUE
C       --СРЕДНИЕ--
      IF(JPRWWW.NE.2) GOTO 8
      RECEIVE(ITASKJ(IPRWWW,JPRWWW-1),TAG=5)(PRABM0(i),i=IN01,IK01)
      DO     9 i=IN01,IK01
      RABM0(i)=RABM0(i)+PRABM0(i)
    9 CONTINUE
      SEND(ITASKJ(IPRWWW,JPRWWW+1),TAG=5)(RABM0(i),i=IN01,IK01)
    8 CONTINUE
      IF(JPRWWW.NE.3) GOTO 10
C       --МАСТЕР--
      RECEIVE(ITASKJ(IPRWWW,JPRWWW+1),TAG=5)(PRABM0(i),i=IN01,IK01)
      DO    11 i=IN01,IK01
      RABM0(i)=RABM0(i)+PRABM0(i)
   11 CONTINUE
      RECEIVE(ITASKJ(IPRWWW,JPRWWW-1),TAG=5)(PRABM0(i),i=IN01,IK01)
      DO    12 i=IN01,IK01
      RABM0(i)=RABM0(i)+PRABM0(i)
   12 CONTINUE
      SEND(ITASKJ(IPRWWW,JPRWWW-2),TAG=5)(RABM0(i),i=IN01,IK01)
   10 CONTINUE
C +++ ОПЕРАТОР 7 +++
C       ----ПРИЕМ РЕЗУЛЬТАТА МАТФУНКЦИИ----
      IF(JPRWWW.NE.1) GOTO 13
      RECEIVE(ITASKJ(IPRWWW,JPRWWW+2),TAG=5)(RABM0(i),i=IN01,IK01)
   13 CONTINUE
C   КОНТРОЛЬНАЯ ТОЧКА bbb
```

Блок записи сохраняемых переменных

```
      OPEN(1,FORM='UNFORMATTED',STATUS='UNKNOWN',FILE=CPFNAME0)
      CPSTR0='bbb'
      WRITE(1)  CPSTR0
      WRITE(1)  CPSTR1,CPVAR0
      WRITE(1)  V,RABM0
      CLOSE(1)
      GOTO 14
```

Блок восстановления сохраняемых переменных

```
   15 CONTINUE
      READ(1,ERR=2)  CPSTR0,CPVAR1
      IF(.NOT.((CPSTR0.EQ.CPSTR1).AND.(CPVAR1(1).EQ.CPVAR0(1))
     >.AND.(CPVAR1(2).EQ.CPVAR0(2)).AND.(CPVAR1(3).EQ.CPVAR0(3))
     >.AND.(CPVAR1(4).EQ.CPVAR0(4)).AND.(CPVAR1(5).EQ.CPVAR0(5))
     >.AND.(CPVAR1(6).EQ.CPVAR0(6)).AND.(CPVAR1(7).EQ.CPVAR0(7))
     >.AND.(CPVAR1(8).EQ.CPVAR0(8)))) GOTO 2
      READ(1,ERR=2)  V,RABM0
      CLOSE(1)
      IF(KPSYNC0(1).EQ.0) GOTO 3
   14 CONTINUE
```

```
      IF(JPRWWW.NE.1) GOTO 16
      DO    17 i=IN01,IK01
      Vsum(i)=RABM0(i)
   17 CONTINUE
   16 CONTINUE
C  ОТСЫЛКА ПАРАМЕТРОВ-РЕЗУЛЬТАТОВ РАЗДЕЛА
C  Отсылка распределенных результатов из линейки ПЭ по i
      IF(JPRWWW.NE.1) GOTO 35
      SEND(IWWWWJ)Vsum
   35 CONTINUE
40001 FORMAT ('SUMV',I2.2,I2.2,'.cp')
      GOTO 32
      END
```

Функция синхронизации для распределенного раздела

```
      FUNCTION KPSYNC0(Ival)
      INTEGER i, j
      TASKID ITASKJ(3,4)
      TASKID IVVVVJ,IWWWWJ
      COMMON /NORMAMESSPASS/ITASKJ,IPRWWW,JPRWWW,IVVVVJ,IWWWWJ
      IF(((IPRWWW.NE.1).OR.(JPRWWW.NE.1))) GOTO 1
      Ir=Ival
      DO     2 i=1,3
      DO     2 j=1,4
      IF(((i.EQ.1).AND.(j.EQ.1))) GOTO 2
      RECEIVE(ITASKJ(i,j),TAG=4)Ip
      IF (Ip.LT.Ir) THEN
      Ir=Ip
      END IF
    2 CONTINUE
      GOTO 3
    1 CONTINUE
      SEND(ITASKJ(1,1),TAG=4)Ival
    3 CONTINUE
      IF(((IPRWWW.NE.1).OR.(JPRWWW.NE.1))) GOTO 4
      DO     5 i=1,3
      DO     5 j=1,4
      IF(((i.EQ.1).AND.(j.EQ.1))) GOTO 5
      SEND(ITASKJ(i,j),TAG=4)Ir
    5 CONTINUE
      GOTO 6
    4 CONTINUE
      RECEIVE(ITASKJ(1,1),TAG=4)Ir
    6 CONTINUE
      KPSYNC0=Ir
      END
```

Литература.

[1] А.Н.Андрианов, А.Б.Бугеря, К.Н.Ефимкин, И.Б.Задыхайло. НОРМА. Описание языка. Рабочий стандарт / Препринт ИПМ им.М.В.Келдыша РАН. №120. 1995. 52с.

Бугеря А.Б.

КОНТРОЛЬНЫЕ ТОЧКИ В ЯЗЫКЕ НОРМА