Цена 9 коп.

Ордена Ленина

ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ

имени М.В. Келдыша

Академии наук СССР

Н.И. Вьюкова

РЕАЛИЗАЦИЯ СИСТЕМЫ ВВОДА-ВЫВОДА ДЛЯ ТРАНСЛЯТОРА ФОРЕКС НА ЭВМ БЭСМ-6

Препринт № 138 за 1986 г.

Москва

Описывается реализация системы ввода-вывода в транслятор Форекс с языка Фортран 77 для ЭВМ БЭСМ-6

КЛЮЧЕВЫЕ СЛОВА И ФРАЗЫ: ввод-вывод, форматное редактирование, Фортран 77, Форекс, БЭСМ-6.

ОГЛАВЛЕНИЕ

I. ВВЕДЕНИЕ

В языке Фортран 77 [1,2] по сравнению с Фортраном 66 существенно расширены средства ввода-вывода. В связи с этим возникла необходимость реализации новой системы ввода-вывода для транслятора Форекс, в котором до настоящего времени использовалась система ввода-вывода транслятора Фортран-Дубна [3]. В новой системе добавлены следующие возможности:

1. Дескрипторы преобразования в описателях форматов
   - для вещественных чисел Gш.д,
   - для цепочек типа BIT B, O, Z, Bш, Oш, Zш,
   - для цепочек типа CHARACTER A, Aш.
2. Дескрипторы редактирования T, TL, TR, :, BN, BZ, S, SP, SS.
3. Ввод и вывод в свободном формате.
4. Перехват ошибок и ситуации 'конец файла'.
5. Обмен с терминалом.
6. Бесформатный ввод и вывод цепочек типов BIT и CHARACTER.
7. Форматный обмен с внутренними каналами (вместо инструкций ENCODE и DECODE).
8. Инструкции OPEN, CLOSE, INQUIRE.

В результате трансляции каждая инструкция ввода-вывода (кроме FORMAT) преобразуется в последовательность обращений к служебным подпрограммам. Эти подпрограммы в совокупности с некоторыми вспомогательными подпрограммами и общими блоками образуют систему ввода-вывода транслятора Форекс для языка Фортран 77.

Большая часть подпрограмм системы написана на языке Форекс. Некоторые подпрограммы реализованы на автокоде БЕМШ. Используются также подпрограммы МС ДУБНА для работы с внешними устройствами.

При разработке и реализации системы ставились следующие задачи:

1. Соответствие стандарту языка Фортран 77.
2. Достаточно хорошая совместимость с прежней системой ввода-вывода.
3. Экономия памяти
4. Приемлемое время работы

В последующих разделах описываются особенности реализации

данной системы, показано, что и как удалось сделать для решения поставленных задач.

Работа выполнена под руководством Ю.М. Баяковского. Автор выражает искреннюю признательность также Вик.С. Штаркману, Л.Б. Морозовой, В.А. Галатенко, А.Б. Ходулеву за помощь и полезные обсуждения.

2. ТАБЛИЦА КАНАЛОВ

Каждая инструкция ввода-вывода выполняется с определенным каналом ввода-вывода. Канал либо подсоединен к файлу, либо свободен. Каналы перенумерованы целыми числами от 1 до N, в данной системе значение N=52.

Таблица каналов представляет собой массив, в котором под каждый канал отведено одно слово. Если канал свободен, то соответствующее слово нулевое. В противном случае оно содержит в упакованном виде информацию о файле, к которому подсоединен канал, в частности:

1. тип файла (какому устройству соответствует)
2. длина форматной записи в байтах,
3. признак, разрешающий форматный обмен,
4. признак, разрешающий бесформатный обмен,
5. признак копирования информации на АЦПУ.

Начальное распределение каналов (совместимое с [3]):

1-15 - магнитные ленты с математическими номерами от 41 до 57,
16 - магнитный барабан с математическим номером 16,
21-36 - то же, что 1-16, но при форматном обмене информация копируется на АЦПУ,
37 - видеотон + копия на АЦПУ,
40 - ввод с ПК + копия на АЦПУ,
42 - вывод на ПК + копия на АЦПУ,
47 - видеотон,
50 - ввод с ПК,
51 - АЦПУ
52 - вывод на ПК.

3. ОБРАБОТКА ФОРМАТОВ

Формат в инструкциях READ, WRITE, PRINT, PUNCH может быть задан

1. меткой инструкции FORMAT,
2. целой переменной, которой при помощи инструкции ASSIGN было присвоено значение метки инструкции FORMAT,
3. текстовой константой,
4. текстовым выражением, не являющимся константой,
5. именем текстового массива.

В случаях (1, 2, 3), когда формат известен во время трансляции, он преобразуется во внутреннее представление и в таком виде хранится в оттранслированной программе. Каждому дескриптору присвоен код - целое число от 1 до 33. Формат во внутреннем представлении - это последовательность байтов, в которой под каждый дескриптор отведена группа из 1 или нескольких подряд стоящих байтов. В первом байте группы хранится код дескриптора, в остальных - числовые параметры дескриптора, (если они имеются). Числовыми параметрами являются, например, ширина поля в/в и число цифр после точки в дескрипторах F, E, номер позиции в дескрипторе T и т. п. Параметры со значениями от 1 до 2^8-1 занимают по одному байту. Значения от 2^8 до $2^{16}-1$ имеют представление, занимающее 3 байта. Числа $\geq 2^{16}$ в описателях форматов не допускаются.

Вся работа по синтаксическому анализу форматов переносится на стадию трансляции. Подпрограмма интерпретации форматов, которая работает во время выполнения, имеет дело с удобным внутренним представлением и становится более компактной и быстрой.

Если формат неизвестен во время трансляции, то он преобразуется во внутреннее представление при выполнении инструкции в/в перед началом форматного редактирования элементов списка в/в. Подпрограмма интерпретации формата и в этом случае работает с его внутренним представлением. Для преобразования формата во внутреннее представление на стадии трансляции и во время выполнения используется одна и та же подпрограмма.

4. ТРАНСЛЯЦИЯ ИНСТРУКЦИЙ READ, WRITE, PRINT, PUNCH

Каждая инструкция ввода-вывода транслируется в последовательность обращений к подпрограммам системы в/в. Вначале как правило идут обращения к подпрограммам настройки. Затем следуют обращения к подпрограммам форматного редактирования, по одному на каждый элемент списка в/в. В конце обычно стоит обращение к подпрограмме терминирования. Она служит для приведения общих блоков системы в стандартное состояние.

Пример 1.

```
      CHARACTER STRING*10
      READ 1, K, A, STRING
```

Инструкция READ будет транслирована в последовательность обращений к подпрограммам, которую на Фортране можно записать так:

```
C      *** Н А С Т Р О Й К А ***
C Настройка на формат с меткой 1. A1 - адрес,
C по которому хранится  внутреннее представление формата:
      CALL ЬINFMЙ(/A1/)
C настройка на ввод с ПК:
      CALL ЬREADЙ
C форматный ввод одного целого числа:
      CALL ЬIF2IЙ(K,/1/)
C форматный ввод одного вещественного числа:
      CALL ЬIF2RЙ(A,/1/)
C форматный ввод одной текстовой цепочки длины 10:
      CALL ЬIF2AЙ(STRING,/10/,/1/)
C       *** Т Е Р М И Н И Р О В А Н И Е ***
C возврат системы в/в к стандартному состоянию:
      CALL ЬTRMЙ
```

Пример 2.

```
      CHARACTER F*60
      DOUBLE PRECISION S(100)
       ...
      PRINT F, S
```


В этом примере формат задан текстовой переменной. Поэтому преобразование его во внутреннее представление происходит во время выполнения программы. Инструкция PRINT будет транслирована в последовательность обращений к подпрограммам:

```
C Преобразование формата во внутреннее представление и
C настройка на него:
      CALL ЬTRFMЙ( F, /60/)
C Настройка на АЦПУ:
      CALL ЬPRINЙ
C Форматный вывод массива двойной точности длины 100:
      CALL ЬOF2DЙ( S, /100/ )
C Терминирование вывода по формату и приведение системы в/в
C в стандартное состояние:
      CALL ЬTRMFЙ
```

Пример 3.

```
      PARAMETER (MEC=12, Д=31, IN=37)
      DIMENSION T( MEC, Д )
      READ (UNIT=IN, FMT=*), M, (T(M,J), J=1,Д)
```

В данном случае выполняется ввод в свободном формате, поэтому настройка на формат отсутствует. Неявному циклу в списке ввода в готовой программе соответствует явный цикл по J от 1 до Д.

```
C Настройка на ввод по каналу номер IN:
      CALL ЬINI2Й( /IN/ )
C Ввод одного целого числа в свободном формате:
      CALL ЬIF4IЙ( M, /1/ )
C Ввод строки массива T типа REAL в свободном формате:
      DO (J=1,Д)
        CALL ЬIF4RЙ( T(M,J), /1/ )
      REPEAT
C Приведение системы в/в в стандартное состояние:
      CALL ЬTRMЙ
```

5. ПОДПРОГРАММЫ НАСТРОЙКИ ДЛЯ КАНАЛОВ И УСТРОЙСТВ

Подпрограммы этой группы служат для подготовки общих блоков системы к форматному или бесформатному обмену. Они проверяют, допустима ли данная операция в/в с указанным каналом, инициализируют буфер системы, устанавливают номер текущей позиции в форматной записи равным 1, и т. п. Они также запоминают в системе адрес входа в подпрограмму ввода или вывода одной записи (драйвер) для данного устройства. Подпрограммы настройки для разных типов инструкций в/в оформлены в виде отдельных модулей и содержат ссылки только на те драйверы, с которыми может работать данная инструкция.

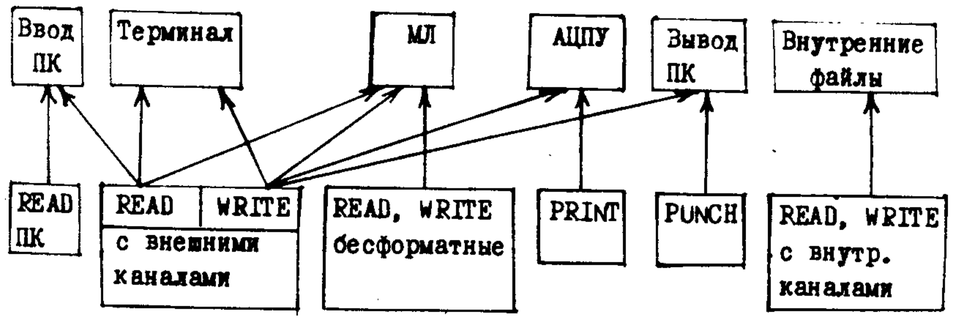


Рис. 1. Использование драйверов в/в в различных инструкциях.

Такая-организация подпрограмм настройки позволяет в некоторых случаях избежать загрузки неиспользуемых драйверов, многие из которых довольно велики по объему. Тем не менее, если в программе пользователя есть инструкции форматного обмена с внешним каналом, (например, WRITE (51,7)A), то во время трансляции невозможно определить, с каким именно устройством она будет работать, даже если номер канала задан константой, поскольку начальное распределение каналов стр. 4 можно изменить при помощи инструкций OPEN, CLOSE. В память будут загружены все возможные

драйверы.

Другое решение задачи о загрузке драйверов на уровне мониторной системы реализовано в работе [4].

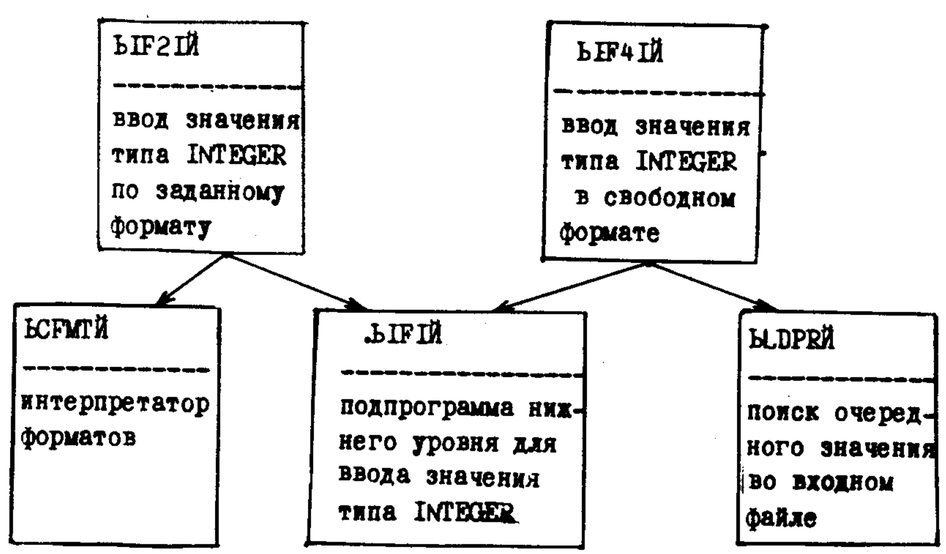
6. СТРУКТУРА ПОДПРОГРАММ ФОРМАТНОГО ВВОДА И ВЫВОДА

Рассмотрим сначала подпрограммы ввода. Для каждого типа значений имеется подпрограмма форматного ввода нижнего уровня. Эта подпрограмма вводит из текущей записи одно значение определенного типа при условии, что в общем блоке системы в/в находятся:

1. номер текущей позиции в форматной записи,
2. размер поля ввода,
3. размер форматной записи,

Для ввода значений числовых типов нужны также:

4. значение д для дескрипторов Iш.д, Fш.д, Eш.д, Gш.д, Dш.д,
5. значение масштабирующего множителя,
6. режим ввода пробелов (в соответствии с дескрипторами BN, BZ),



Подпрограммы нижнего уровня используются при вводе как по заданному формату, так и в свободном формате. На схеме стр. 9 изображена структура подпрограмм форматного ввода для значений типа INTEGER. Аналогичную структуру имеют подпрограммы ввода и для других типов данных.

При вводе по заданному формату используется подпрограмма ЬCFMTЙ - интерпретатор форматов. Она просматривает формат (во внутреннем представлении), начиная с текущей позиции, выполняет дескрипторы редактирования, пока не найдет очередной дескриптор преобразования. Результат своей работы подпрограмма ЬCFMTЙ оставляет в общем блоке. Результатом ее работы является код очередного дескриптора преобразования, размер поля в/в, число десятичных знаков для дескрипторов F, E, G, D, I. Могут измениться также значение масштабирующего множителя, номер текущей позиции в записи, признак ввода пробелов и т. д.

При вводе в свободном формате используется подпрограмма ЬLDPRЙ, которая распознает разделители, коэффициенты повторения, пустые значения. Она находит во входном файле начало и конец очередного элемента данных. По окончании своей работы подпрограмма ЬLDPRЙ оставляет в общем блоке

номер текущей позиции в текущей записи,
размер поля ввода.

Режимы ввода числовых значений (4, 5, 6, стр. 9) устанавливаются в соответствии со стандартом [1].

Аналогичную структуру имеют подпрограммы форматного вывода. Для каждого типа значений имеется подпрограмма форматного вывода нижнего уровня (для типа REAL имеется три таких подпрограммы - по одной для каждого из дескрипторов F, E, G). Подпрограмма вывода нижнего уровня предназначена для вывода одного значения определенного типа, при условии что в общем блоке системы находятся соответствующие параметры вывода (такие же, как параметры ввода на стр. 9, только вместо признака ввода пробелов используется признак вывода знака '+' в соответствии с дескрипторами S, SS, SP).

Подпрограмма вывода по заданному формату обращается сначала к интерпретатору форматов, а затем к соответствующей подпрограмме нижнего уровня.

При выводе в свободном формате в общем блоке системы устанавливаются некоторые фиксированные значения параметров вывода, а затем происходит обращение к подпрограмме нижнего уровня.

Выбранная структура позволяет избежать загрузки в память машины неиспользуемых подпрограмм. Так, если в программе нет ввода или вывода значений типа LOGICAL или BIT, то не будут загружены и соответствующие подпрограммы. Если не используется ввод в свободном формате, то не загрузятся и подпрограммы верхнего уровня для ввода в свободном формате, а также подпрограмма поиска очередного значения ЬLDPRЙ.

7. СОПОСТАВЛЕНИЕ СПИСКА ВВОДА-ВЫВОДА С ФОРМАТОМ НА СТАДИИ ТРАНСЛЯЦИИ

В разд. 3, где рассматривалась обработка форматов, отмечалось, что если формат известен во время трансляции, то он анализируется и преобразуется во внутреннее представление. В результате упрощается процесс интерпретации формата на стадии выполнения. Однако довольно часто можно пойти и дальше - провести на стадии трансляции сопоставление списка ввода-вывода с описателем формата. Пример 4:

```
      PRINT 1, X,У
    1 FORMAT (' X=', F10.2, ', У=', 1P,E14.6)
```

В данном случае имеется достаточно информации для того, чтобы транслировать инструкцию PPINT в последовательность обращений к подпрограммам, выполняющим следующие действия:

1. настройка на АЦПУ,
2. вывод цепочки ' X=' в текущую форматную запись,
3. вывод значения X по формату F10.2,
4. вывод цепочки ', У=',
5. установка значения масштабирующего множителя =1,
6. вывод значения У по формату E14.6,
7. вывод текущей записи,

8. терминирование: приведение общих блоков системы в стандартное состояние.

Действия по настройке и терминированию выполняются при помощи тех же подпрограмм, что и обычно. Вывод элементов списка в/в и вывод текстовых цепочек, содержащихся в формате - при помощи подпрограмм форматного вывода нижнего уровня (разд. 6). Для выполнения действий, соответствующих дескрипторам редактирования нужен специальный набор подпрограмм, которые бы производили необходимые установки в общем блоке системы.

Таким образом, на стадии выполнения производились бы обращения непосредственно к подпрограммам форматного вывода (ввода) нижнего уровня, минуя верхний уровень и интерпретатор форматов. Описанный способ трансляции позволяет:

- экономить время, т. к. на стадии выполнения не нужно интерпретировать формат, проверять соответствие между описателем формата и списком в/в и т. п.,
- экономить память, поскольку не загружаются подпрограммы верхнего уровня и интерпретатор форматов, не нужно хранить описатели форматов,
- обнаруживать еще во время трансляции ошибки типа 'описатель формата не соответствует списку в/в'.

Сопоставление списка в/в с форматом во время трансляции возможно не всегда, даже если формат известен. Пример:

```
      DIMENSION A(N)
      PRINT (F7.2, 5G14.6) A, X
```

если N - переменная, то невозможно во время трансляции установить, по какому дескриптору следует выводить значение X. Отметим, что экономия памяти за счет неиспользования интерпретатора форматов и подпрограмм верхнего уровня возможна лишь при условии, если все инструкции форматного ввода и вывода в данной программе удастся транслировать описанным выше способом.

В существующих версиях транслятора Форекс этот метод не реализован.

9. ПРИМЕРЫ. ВРЕМЯ РАБОТЫ И ЗАНИМАЕМАЯ ПАМЯТЬ

Рассмотрим на нескольких примерах, как меняется время работы и объем памяти, занимаемой системой, в зависимости от вида параметров в инструкциях в/в. Рассмотрим программу, содержащую инструкции READ и PRINT.

Пример 5. Ввод и вывод элементов массива в цикле, формат постоянный.

```
      PROGRAM IO1
      PARAMETER (L=50,
     + FMTRD='(5F12.0)',
     + FMTPR='(F10.0,F16.5,F17.7,E12.3,F15.2)')
      DIMENSION A(L), B(L), C(L), D(L), E(L)
      READ FMTRD, (A(I), B(I), C(I), D(I), E(I), I=1,L)
      PRINT FMTPR, (A(I), B(I), C(I), D(I), E(I), I=1,L)
      END
```

Время работы инструкции READ 0.46 сек. (0.44 сек.),
время работы инструкции PRINT 1.30 сек. (1.38 сек.),
занимаемая память 5314 (6446).

Здесь и ниже в скобках указываются соответствующие данные для прежней системы в/в. В качестве занимаемой памяти приводится адрес 'свободно' в таблице загрузки.

Пример 6. Ввод и вывод элементов массивов в цикле, формат переменный.

```
      PROGRAM IO2
      PARAMETER (L=50)
      DIMENSION A(L), B(L), C(L), D(L), E(L)
      CHARACTER FMTRD*9, FMTPR*36
      DATA FMTRD /'(5F12.0) '/
     + FMTPR /'(F10.0,F16.5,F17.7,E12.3,F15.2)'/
      READ FMTRD, (A(I), B(I), C(I), D(I), E(I), I=1,L)
      PRINT FMTPR, (A(I), B(I), C(I), D(I), E(I), I=1,L)
      END
```

Время работы инструкции READ 0.47 сек. (0.44 сек.),
время работы инструкции PRINT 1.32 сек. (1.39 сек.),
занимаемая память 6635 (6446).

В этом примере в инструкциях READ и PRINT форматы заданы текстовыми переменными, поэтому преобразование форматов происходит во время выполнения. Объем занимаемой памяти по сравнению с примером 5 увеличился за счет подпрограммы преобразования форматов. Время работы существенно не изменилось, т. к. преобразование формата выполняется в обеих инструкциях однократно.

Пример 7. Вывод массива целиком, формат постоянный.

```
      PROGRAM I03
      PARAMETER (K=5, L=50,
     + FMTRD='(5F12.0)',
     + FMTPR='(F10.0,F16.5,F17.7,E12.3,F15.2)')
      DIMENSION F(K,L)
      READ FMTRD, F
      PRINT FMTPR, F
      END
```

Время работы инструкции READ 0.40 сек. (0.44 сек.),
время работы инструкции PRINT 1.20 сек. (1.30 сек.),
занимаемая память 5246 (6411).

Время работы по сравнению с примером 5 сократилось за счет того, что элементы массива вводятся и выводятся не в цикле, а за одно обращение к подпрограмме.

Пример 8. В качестве примера 8 рассмотрим программу I01 из примера 5, транслированную 'вручную' по методу, который описан в разд. 7. Результаты измерений:
время работы инструкции READ 0.35 сек.
Время работы инструкции PRINT 1.08 сек.
Занимаемая память 4515.

По сравнению с примером 5 наблюдается некоторое сокращение времени работы и уменьшение объема памяти.

Пример 9. Бесформатный обмен.

```
( 1)      PROGRAM I05
( 2)      PARAMETER (L=1000,
( 3)     + FMTRD='(5F12.0)',
( 4)     + FMTPR='(10F12.6)',
```



```
( 5)      DIMENSION A(L)
( 6)      READ FMTRD, A
( 7)      WRITE (16) A
( 8)      WRITE (16) (A(I), I=1,L)
( 9)      REWIND 16
(10)      READ (16) A
(11)      READ (16) (A(I), I=1,L)
(12)      PRINT FMTPR, A
(13)      END
```

Время работы инструкций бесформатного вывода (WRITE)
в строке (7) 0.28 сек. (0.48 сек.),
в строке (8) 0.46 сек. (0.50 сек.).
Время работы инструкций бесформатного ввода (READ)
в строке (10) 0.26 сек. (0.48 сек.),
в строке (11) 0.46 сек. (0.50 сек.).
Занимаемая память 10300 (10227).

В строках (7) и (10) обмен выполняется за одно обращение к подпрограмме, а в строках (8) и (11) элементы массивов выводятся и вводятся в цикле, так что обращение к подпрограмме выполняется для каждого элемента массива. Этим объясняется наблюдаемая разница во времени работы. В примерах 5-8 из всего набора драйверов загружались только 'ввод с ПК' и 'вывод на АЦПУ'. В примере 9 загружается еще драйвер обмена с МЛ.

Приведенные примеры, а также другие измерения показывают, что время работы инструкций READ, PRINT, WRITE примерно такое же, как и с прежней системой. В зависимости от характера параметров инструкций могут наблюдаться некоторые отклонения в ту или другую сторону. Объем занимаемой памяти существенно зависит от набора используемых средств.

9. ПЕРЕХВАТ ОШИБОК

В любой инструкции в/в (кроме PRINT, PUNCH, FORMAT) пользователь может задать параметры перехвата ошибок: ERR=м, END=м1, IOSTAT=кот. В соответствии с этими параметрами обеспечивается реакция на ошибки при выполнении инструкции:

1. если есть параметр ERR=м, то выход по метке м,

2. если есть параметр IOSTAT=кот, но нет параметра ERR, то выход на следующую инструкцию,
3. если нет ни одного из параметров ERR, IOSTAT, то печать сообщения об ошибке и STOP.

В случае конца файла при вводе

4. если есть параметр END=м1, то выход по метке м1,
5. если параметр END отсутствует, то конец файла при вводе рассматривается как ошибка с кодом -1 и обрабатывается в соответствии с пп. 1-3.

Если есть параметр IOSTAT=кот, то в переменную (элемент массива, поле записи) кот заносится код ответа.

Реализация. В общем блоке системы имеются три переменные:
ERRS - адрес выхода по ошибке,
ENDS - адрес выхода по концу файла,
IOSTAT - адрес, по которому занести код ответа.
В стандартном состоянии они имеют нулевые значения. Для каждого из параметров ERR, END, IOSTAT, если он задан в инструкции в/в, генерируется обращение к соответствующей подпрограмме настройки. Обращения к подпрограммам настройки для этих параметров выполняются всегда в первую очередь и в строго определенном порядке, как показано на рис. 2.

```
C Для IOSTAT=кот
C (A- адрес следующей инструкции)
      CALL ЫNIOЙ(/A/, КОТ)
C Для ERR=м:
      CALL ЫNERЙ(/M/)
C Для END=м1:
      CALL ЫNENЙ(/M1/)
```

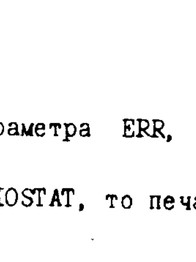


Рис. 2. Схема работы подпрограмм настройки для параметров перехвата ошибок.

Каждая из подпрограмм заносит адрес, передаваемый ей транслятором в качестве аргумента, в переменные ERRS и ENDS согласно схеме на рис. 2. В результате настройки в переменных ERRS и ENDS оказываются либо нулевые значения, либо адреса выхода в соответствии с пп. 1-5 на стр. 15, 16. Кроме того, любая из этих подпрограмм сохраняет в памяти системы содержимое регистров ИР1-ИР7, ИР15.

В системе имеется также подпрограмма обработки ошибки с одним аргументом - кодом ошибки. К ней могут обращаться другие подпрограммы системы в случае обнаружения ошибки при выполнении инструкции в/в. Подпрограмма обработки ошибки анализирует содержимое переменной ERRS. Если оно равно 0, то печатается сообщение об ошибке и выполнение программы пользователя на этом завершается. В противном случае

по адресу, находящемуся в IOSTAT (если он не нулевой) заносится код ответа,

восстанавливается содержимое регистров ИР1-ИР7, ИР15,

восстанавливается стандартное состояние общих блоков системы,

управление передается по адресу, который хранился в ERRS.

Аналогично действует подпрограмма обработки ситуации 'конец файла'.

Описанные в этом разделе подпрограммы реализованы на автокоде. Возможна реализация перехвата ошибок и средствами языка Форекс (Фортран 77). Например, в каждой подпрограмме системы в/в можно было бы иметь дополнительный параметр - альтернативный возврат, и выходить по нему в случае обнаружения ошибки. Но тогда пришлось бы передавать дополнительный параметр, даже если в инструкции в/в в программе пользователя отсутствуют параметры ERR, END, IOSTAT. В результате увеличится объем подпрограмм, замедлится их работа. Поэтому был выбран описанный выше более экономный подход.

10. ЗАКЛЮЧЕНИЕ

В работе рассмотрены лишь некоторые наиболее интересные особенности реализации системы ввода-вывода. Почти ничего не сказано о подпрограммах для инструкций OPEN, CLOSE, INQUIRE,

поскольку они сводятся к несложным манипуляциям с описателем канала.

Вернемся теперь к задачам, которые были сформулированы во введении.

I. Соответствие стандарту языка Фортран 77. Реализованы все возможности языка, за исключением файлов прямого доступа. И рабочих файлов. Реализованы также средства для ввода и вывода цепочек BIT.

2. Совместимость с прежней системой ввода-вывода. Выбрано такое же распределение каналов в/в. Обмен с магнитными носителями выполняется так же, как в прежней системе. Сохранена возможность выводить значения типов INTEGER, REAL, LOGICAL по форматам A, O.

Различия: в новой системе не допускаются инструкции ENCODE и DECODE. Их следует заменить на инструкции WRITE и READ с внутренними файлами. Пользователям придется изменить также форматы для ввода и вывода цепочек типов CHARACTER и BIT, т. к. раньше для значений этих типов в языке Форекс использовались 'нестандартные' форматы.

3. Экономия памяти. Система предоставляет пользователю довольно широкий набор средств, и общий объем входящих в нее подпрограмм велик. Поэтому важно было разбить систему на модули таким образом, чтобы по возможности не загружать те части, которые не нужны в данной программе пользователя.

Объем памяти, занимаемой системой в/в в той или иной программе существенно зависит от набора средств, которые в этой программе используются. Так, если употребляются только инструкции READ (ввод с ПК) и PRINT с постоянными форматами, то программа окажется короче, чем с прежней системой в/в. Если используется весь набор драйверов, то программа займет несколько больший объем памяти. С целью экономии памяти некоторые подпрограммы системы были написаны на автокоде.

4. Приемлемое время работы. В среднем, как показывают измерения, система работает примерно так же, как прежняя, в различных

ситуациях наблюдаются отклонения по времени в ту или иную сторону. Увеличение времени работы может быть связано с тем, что большинство подпрограмм реализовано на языке высокого уровня, а также с тем, что система разбита на многочисленные модули, так что приходится тратить время на входы и выходы из подпрограмм. В целях ускорения работы системы

- формат во время трансляции преобразуется в более удобное внутреннее представление,
- часть подпрограмм реализована на автокоде,
- массивы, встречающиеся в списке в/в, вводятся или выводятся за одно обращение к подпрограмме, а не в цикле. Особенно заметен выигрыш от этого при бесформатном обмене.

Данная система в/в подключается к транслятору Форекс, начиная с версии 5.0. В настоящее время эта версия находится в опытной эксплуатации. Для того, чтобы воспользоваться ей, следует карту *FOREX заменить на *FOR80 .

ЛИТЕРАТУРА

I. Г. Катцан, Фортран 77, Мир, Москва, 1982.

2. Ю. М. Баяковский, Н. И. Вьюкова, В. А. Галатенко, Т. Н. Михайлова, Л. Б. Морозова, А. Б. Ходулев, Вик. С. Штаркман. Расширенный Фортран - Форекс. М., ИПМ АН СССР, 1983.

3. В. Я. Карпов. Алгоритмический язык - ФОРТРАН. М., Наука, 1976.

4. Бардин В. В., Поминов П. В., Руднев А. П. Ввод/вывод в мониторной системе ДУБНА. М., ИАЭ-3682/16, 1982.

5. Белокопытов Ю. А., Каминский А. Г., Клименко С. В., Лебедев А. А., Половников С. А. Символьный вывод в программных системах. Программирование 4, 1981.

6. P. W. Abrahams. L. A. Clarke. Compile Time Analysis of Data List - Format List Correspondense. IEEE Transactions on Software Engineering, Vol. SE-5, No 6, 1979.

Приложение I. Внутреннее представление описателя формата.

Дескриптор	Представление: байты 1 2 3 4 5	Дескриптор	Представление: байты 1 2 3 ... m+2
Bm *	1 m	Im.д	18 д m
Am *	2 m	Im *	19 m
Om *	3 m	Lm *	20 m
Zm *	4 m	mHc ... c	21 m c ... c
коэффициент		TLm *	22 m
повторения к *	5 к-1	TPm *	23 m
(*	6 к-1	mX *	24 m
) *	7 п	/	25
конец формата *	8 п	:	26
Tm *	9 m	SP	27
B	10	S	28
A	11	SS	29
O	12	кP *	30 к
Z	13	-кP *	31 к
Em.дEe	14 'E' e д m	BN	32
Em.дDe	14 'D' e д m	BZ	33
Em.д, Dm.д	15 д m	'большое	
Gm.д	16 д m	число' м	34 $м_1$ $м_2$
Cm.д	17 д m	($м=м_1*256+м_2$)	

Комментарии и примеры.

1. Дескриптор с коэффициентом повторения.

Пример: дескриптор 10B4 имеет представление

5	9	1	4

(5 - код коэффициента повторения)

10 B 4

2. Скобки в описателе формата. При открывающей скобке (код 6) указывается значение к-1, где к - коэффициент повторения, который стоит перед скобкой. Если коэффициент отсутствует, то

считается, что к=1. При закрывающей скобке (код 7) указывается номер байта во внутреннем представлении формата, где начинается соответствующий групповой дескриптор. С концом формата (код 8) задается номер байта во внутреннем представлении формата, откуда начинается повторяемая часть формата. Пример.

```
7   FORMAT ( 3H M=, (T4, 2215))
```

Внутреннее представление:

байт номер	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
содержит	21	3	32	77	61	6	0	9	4	5	21	19	5	7	8	8	6
	3H			M	=	(		T	4		22		I	5	)	)	

3. Цепочка литер в кавычках имеет такое же представление, как цепочка с дескриптором H (код 21).

4. Если дескриптор имеет только один числовой параметр, то этот параметр может принимать и значения $\geq 2^8$ (но $< 2^{16}$). Дескриптор в этом случае имеет другое представление, занимающее 4 байта:

34	$м_1$	$м_2$	код дескриптора

Пример. Дескриптор T257 имеет представление

34	1	1	9

(257=1*256+1, 9 - код дескриптора T)

Дескрипторы, для которых допускается такое представление, отмечены в таблице звездочками.

Приложение 2. Подпрограммы системы в/в.

НАСТРОЙКА ПО ПАРАМЕТРАМ ПЕРЕХВАТА ОШИБОК

ЬINIOЙ(/A/,КОТ) - для IOSTAT=КОТ
ЬINERЙ(/M/) - для ERR=M
ЬINENЙ(/M/) - для END=M

ИНСТРУКЦИИ READ, WRITE, PRINT, PUNCH

2. Настройка на ввод и вывод по формату
ЬINFMЙ(A), A - массив с внутренним представлением формата.
ЬTRFMЙ(F,/L/), F - текст. цепочка с форматом в исходном представлении, L - длина цепочки.

3. Настройка на устройства и каналы
ЬREADЙ - для инструкции READ (ввод с ПК)
ЬINIIЙ(F,/L/,/N/) - ввод по внутр. каналу CHARACTER F(N)*(L)
ЬINI2Й(/K/) - внешний канал K, форматный ввод
ЬINIUЙ(/K/) - внешний канал K, бесформатный ввод
ЬPRINЙ - для инструкции PRINT
ЬPUNCЙ - для инструкции PUNCH
ЬINOIЙ(F,/L/,/N/) - вывод по внутр. каналу CHARACTER F(N)*(L)
ЬINO2Й(/K/) - внешний канал K, форматный вывод
ЬINOUЙ(/K/) - внешний канал K, бесформатный вывод

4. Терминирование
ЬTRMFЙ - для вывода по формату
ЬTRIUЙ - для бесформатного ввода
ЬTROUЙ - для бесформатного вывода
ЬTRMЙ - для всех остальных случаев

6. Список ввода-ввода

Подпрограммы ввода

Тип элемента	по формату	в свободном формате
INTEGER	ЬIF2IЙ(I,/N/)	ЬIF4IЙ(I,/N/)
REAL	ЬIF2RЙ(R,/N/)	ЬIF4RЙ(R,/N/)
DOUBLE PREC.	ЬIF2DЙ(D,/N/)	ЬIF4DЙ(D,/N/)
COMPLEX	ЬIF2RЙ(C,/2*N/)	ЬIF4CЙ(C,/N/)
LOGICAL	ЬIF2LЙ(L,/N/)	ЬIF4LЙ(L,/N/)
CHARACTER	ЬIF2AЙ(A,/L/,/N/)	ЬIF4AЙ(A,/L/,/N/)
BIT	ЬIF2BЙ(B,/L/,/N/)	ЬIF4BЙ(B,/L/,/N/)

Подпрограммы вывода

	по формату	в свободном формате
INTECEP	ЬOF2IЙ(I,/N/)	ЬOF2IЙ(I,/N/)
PEAL	ЬOF2RЙ(R,/N/)	ЬOF2RЙ(R,/N/)
DOUBLE PPEC.	ЬOF2DЙ(D,/N/)	ЬOF2DЙ(D,/N/)
COMPLEX	ЬOF2RЙ(C,/2*N/)	ЬOF4CЙ(C,/N/)
LOCICAL	ЬOF2LЙ(L,/N/)	ЬOF2LЙ(L,/N/)
CHAPACTEP	ЬOF2AЙ(A,/L/,/N/)	ЬOF4AЙ(A,/L/,/N/)
BIT	ЬOF2BЙ(B,/L/,/N/)	ЬOF4BЙ(B,/L/,/N/)

	Подпрограммы бесформатного	
	вывода	ввода
INTEGER, REAL, LOGICAL, DOUBLE PREC.,		
COMPLEX,	ЬOUWЙ(M,/N/)	ЬIUWЙ(M,/N/)
CHARACTER	ЬOUCЙ(A,/N/)	ЬIUCЙ(A,/N/)
BIT	ЬOUBЙ(B,/N/)	ЬIUBЙ(B,/N/)

N - число элементов в массиве, который нужно ввести или вывести, L - длина цепочки типа CHARACTER или BIT. При бесформатном обмене для цепочек указывается общее число байтов или битов, которые нужно ввести или вывести.

ИНСТРУКЦИИ BACKSPACE, REWIND, ENDFILE

ЬBACKЙ(/K/)	BACKSPACE K
ЬRWNDЙ(/K/)	REWIND K
ЬENDFЙ(/K/)	ENDFILE K

ИНСТРУКЦИЯ OPEN

Подпрограммы	соответствуют параметрам:
ЬOSTAЙ(ST,/L/)	- STATUS=ST
ЬOACCЙ(ACC,/L/)	- ACCESS=ACC
ЬOFMЙ(FM,/L/)	- FORM=FM
ЬORCLЙ(IRECL)	- RECL=IRECL
ЬOBLNЙ(BLNK,/L/)	- BLANK=BLNK
ЬOPNDЙ(/K/,FIN,/L/)	- UNIT=K, FILE=FIN

ИНСТРУКЦИЯ CLOSE

ЬCSTAЙ(ST,/L/)	- STATUS=ST
ЬCLOSЙ(/K/)	- UNIT=K

ИНСТРУКЦИЯ INQUIPE

ЬINQUЙ(/K/)	- UNIT=K
ЬINQFЙ(FIN,/L/)	- FILE=FIN
ЬQEXЙ(EX)	- EXIST=EX
ЬQODЙ(OD)	- OPENED=OD
ЬQNUMЙ(NUM)	- NUMBER=NUM
ЬQNMDЙ(NMD)	- NAMED=NMD
ЬQFNЙ(FN,/L/)	- NAME=FN
ЬQACCЙ(ACC,/L/)	- ACCESS=ACC
ЬQSEQЙ(SEQ,/L/)	- SEQUENTIAL=SEQ

ЬQDIRЙ(DIP,/L/)	- DIRECT=DIR
ЬQFMЙ(FM,/L/)	- FORM=FM
ЬQFMTЙ(FMTD,/L/)	- FORMATTED=FMTD
ЬQUNFЙ(UNFM,/L/)	- UNFORMATTED=UNFM
ЬQRCLЙ(IPCL)	- RECL=IRCL
ЬQNRЙ(NP)	- NEXTREC=NR
ЬQBLNЙ(BLNK,/L/)	- BLANK=BLNK

(L - длина текстовой цепочки.)

В приложении приведены имена только тех входных точек, к которым транслятор может сгенерировать обращения в программных компонентах пользователя. Система содержит также ряд вспомогательных подпрограмм. Имена входных точек, относящихся к системе в/в, начинаются с буквы 'Ь' и заканчиваются буквой 'Й'.

Общий объем кода составляет примерно 1800 строк на Автокоде и 2500 строк на языке Форекс.