Утвержден

26.00152-01 34 07-1-ЛУ

Операционная система МОС ЭП

Подсистема разработки программ

Символический отладчик

Руководство оператора

26.00152-01 34 07-2

Листов 219/4К

1К
6К
215К
216К

1987

Перв. Примен.
26.00152-01

Литера О

СОДЕРЖАНИЕ

1. Команды отладчика

1.1. Общие сведения

В данном разделе приводится описание команд отладчика. Для каждой команды указывается следующая информация:

- описание команды;
- формат команды;
- параметры команды;
- квалификаторы команды;
- один или несколько примеров.

В примерах используются допустимые сокращения команд отладчика.

1.1.1. Общий формат команд отладчика

Ключевое слово - это слово, группа символов или другое символическое обозначение, распознаваемое в пределах определенного контекста как имеющее специальное значение или определяющее особые действия. Ключевые слова, распознаваемые отладчиком, называются командами, квалификаторами команд (или просто квалификаторами) или параметрами команд.

Кроме того, среди некоторых ключевых слов и в соответствии с принятым синтаксисом отладчик распознает информацию пользователя (которая также называется параметрами), такую как числа и спецификации файлов.

Отладчик распознает так же следующие символы:

- синтаксические разделители (например, точка с запятой, двоеточие, запятая, пробел и косая черта);
- специальные символы (например, точка, циркумфлекс и обратная косая черта);

- знаки операций (например, знаки плюс, минус и коммерческое "при" (@)).

Команда отладчика может содержать одно ключевое слово или более. Командная строка содержит команду отладчика, а также один из следующих элементов:

- квалификатор;
- параметр;
- последовательность WHEN;
- последовательность DO;
- комментарии.

Пример показывает порядок, в котором можно указывать эти элементы в командной строке (необязательные элементы заключены в квадратные скобки):

команда [/квалификатор] [параметр] [WHEN(выражение)]-
[DO(командная_строка;...)] [!комментарии]

Команда - активная компонента командной строки, отражающая специфичную функцию, которую необходимо выполнить. Например, EXAMINE, CANSEL TRACE и SET EXCEPTION BREAK являются командами.

/квалификатор - ключевое слово, которое модифицирует (квалифицирует) действие команды. Например, /ALL является квалификатором в командной строке SET MODULE/ALL. Перед квалификатором ставится косая черта.

Параметр - величина, используемая командой, или величина для дальнейшей модификации команды. Например, в командной строке SET TRACE LOOP1 параметр LOOP1 указывает место установки точки отслеживания, а в командной строке SET STEP INTO параметр INTO модифицирует команду.

WHEN(выражение) - командная последовательность WHEN, состоящая из ключевого слова WHEN и выражения на текущем

языке программирования, определяющего условие (такое, как TRUE или FALSE).

DO(командная_строка;...) - командная последовательность DO, состоящая из ключевого слова DO и следующих за ним командных строк, заключенных в круглые скобки.

Комментарии - строка символов (текста) пользователя, начинающаяся с восклицательного знака. Комментарии не оказывают влияние на выполнение команд отладчика.

1.1.2. Правила синтаксиса

На одной строке терминала можно размещать более одной командной строки. Командные строки разделяются точкой с запятой.

Командную строку можно продолжить на другой строке терминала. Для этого в конце строки ставится знак переноса (тире) и нажимается клавиша RETURN. В ответ отладчик выводит на следующей строке терминала запрос в виде символа подчеркивания, после чего пользователь может продолжить набор командной строки.

1.1.3. Сокращения

Для упрощения ввода допускается сокращать ключевые слова отладчика. Минимальная аббревиатура ключевого слова содержит столько символов, сколько необходимо для отличия этого слова среди множества других ключевых слов.

Из этого правила существуют исключения:

1) для наиболее часто используемых команд, таких как EXAMINE и DEPOSIT, можно использовать сокращение до одного символа;

2) в некоторых случаях отладчик правильно интерпретирует неоднозначные аббревиатуры, основываясь на контекст.

1.2. Команда ALLOCATE

Команда ALLOCATE позволяет расширить пул памяти.

Формат:

ALLOCATE число_байтов

Параметры:

Число_байтов - указывает число байтов, на которое пользователь хочет расширить пул памяти отладчика. Это число должно быть не менее 1000. Если будет указано меньшее число, то выдается сообщение об ошибке. Сообщение об ошибке будет выдано также и в том случае, когда система не может распределить память, затребованную пользователем.

Квалификаторы:

Отсутствуют.

Описание.

Команда ALLOCATE позволяет расширить пул памяти, используемый отладчиком, что дает возможность пользователю устанавливать большее количесто модулей. С помощью команды SHOW MODULE можно узнать, сколько памяти требуется для каждого модуля и сколько имеется доступной памяти.

При использовании команды ALLOCATE отладчик обращается к подпрограмме LIB$GET_VM для получения дополнительной памяти. Если при этом в программе пользователя имеется вызов LIB$GET_VM, то возможно нарушение распределения памяти в этой программе.

Пример.

```
DBG> ALLOCATE 2000
```

Эта команда увеличивает размер пула памяти дополнительно на 2000 байтов.

1.3. Команда Эспецификация_файла

Команда Эспецификация_файла вызывает выполнение команд отладчика, которые содержатся в указанном файле. (Файл, указанный в этой команде, называется командной процедурой или командным файлом.)

Формат:

Эспецификация_файла[P1[,P2,...,PN]].

Параметры:

Спецификация_файла - определяет командную процедуру, которая должна быть выполнена. Если спецификация не включает тип файла, то используется тип файла по умолчанию COM. Следует отметить, что спецификация файла в данной команде не выделяется ни кавычками, ни апострофами.

P1,P2,...,PN - определяют параметры, заявленные внутри командной процедуры. Более подробно о параметрах, заявленных в командной процедуре, говорится в описании команды DECLARE.

Квалификаторы:

Отсутствуют.

Описание.

Командная процедура может содержать любые команды отладчика, включая другую команду Эспецификация_файла. Ког-

да отладчик выполняет команду EXIT в командной процедуре или достигает конца файла, он возвращает управление потоку команд, из которого была вызвана данная командная процедура. Потоком команд может быть терминал (т.е. ввод команд с терминала), предыдущая командная процедура или последовательность DO в командах SET BREAK, SET TRACE или SET WATCH.

При вводе команды SET OUTPUT VERIFY все команды, которые считываются из командной процедуры, отображаются на терминале.

Пример.

```
DBG>SET OUTPUT VERIFY    !требование отображения команд до
                         !их выполнения
DBG>MAIN                 !вызов на выполнение командной
                         !процедуры MAIN
%DEBUG-I-VERIFYICF, INTERING INDIRECT COMMAND FILE "MAIN"
                  (ввод косвенного командного файла "MAIN")
SET MODU/ALL
SET BR SUB1
GO
ROUTINE START AT MAIN\MAIN !начало подпрограммы с
                           !MAIN/MAIN
ROUTINE BREAK AT SUB1\SUB1 !приостанов на SUB1\SUB1
E/WORD SUB1
SUB1\SUB1:  4800
E/I
SUB1\SUB1+02:  MOVAL  L^SUB1\Y,R11
EXIT
%DEBUG-I-VERIFYICF, EXITING INDIRECT COMMAND FILE "MAIN"
              (выход из косвенного командного файла "MAIN")
```

```
DBG>
```

1.4. Команда ATTACH

Команда ATTACH передает управление терминалом пользователя с текущего процесса другому процессу.

Формат:

```
ATTACH имя_процесса
```

Параметры:

Имя_процесса - указывает процесс, к которому пользователь хочет "прикрепить" свой терминал. Перед этим необходимо знать, что такой процесс уже существует. Если имя процесса содержит не буквенно-цифровые символы или пробелы, то это имя необходимо заключать в двойные кавычки ("").

Квалификаторы:

Отсутствуют.

Описание:

Команда ATTACH позволяет переходить от сеанса отладки к уровню DCL и обратно, а так же переходить от одного сеанса отладки к другому. Для этого сначала надо создать подпроцесс (см. Описание команды SPAWN), к которому затем можно подключаться, когда это потребуется. Для возврата в исходный процесс с минимальными системными затратами используется опять команда ATTACH.

Пример.

```
DBG>SPAWN
$ ATTACH JONES
%DEBUG-I-RETURNED, CONTROL RETURNED TO PROCESS JONES
```

(управление возвращено процессу JONES)

```
DBG> ATTACH JONES_1
$
```

Этот набор команд порождает подпроцесс JONES_1, затем отладчик (работающий с процессом JONES) подключается к этому подпроцессу.

1.5. Команда CALL

Команда CALL обеспечивает вызов указанной процедуры. Пользователь может определить имя процедуры или виртуальный адрес. Если процедура требует одного или нескольких аргументов, то пользователь должен определить их в этой команде.

Формат:

```
CALL имя_подпрограммы [(Аргумент[,аргумент...])]
```

Параметры:

Имя_подпрограммы - определяет имя или виртуальный адрес вызываемой процедуры.

Аргумент - этот параметр определяет один или несколько аргументов, требуемых для процедуры. Аргументы могут передаваться по адресу, по дескриптору, по ссылке и по значению:

%ADDR - передача аргумента по адресу. Формат:
CALL имя_программы (%ADDR адресное_выражение).
Для выполнения этой функции отладчик получает результат адресного выражения и передает его адрес вызываемой подпрограмме. В случае простой переменной (такой как X) подпрограмме передается

ее адрес (X). Этот механизм такой же, как в фортране ROUTINE (X). Другими словами, для поименованных переменных использование %ADDR соответствует вызову по ссылке на фортране. Однако для других выражений для вызова по ссылке необходимо использовать функцию %REF;

%DESCR - передача аргумента по дескриптору (описателю). Формат:

CALL имя_программы (%DESCR языковое_выражение)

для выполнения этой функции отладчик оценивает языковое выражение и создает стандартный описатель (дескриптор) передаваемой величины. Затем дескриптор передается вызываемой подпрограмме. Данная функция должна использоваться при передаче строк подпрограммам на фортране;

%REF - передача аргумента по ссылке. Формат:

CALL имя_программы (%REF адресное_выражение)

для выполнения данной функции отладчик вычисляет адресное выражение и передает указатель значения вызываемой подпрограмме. Этот механизм соответствует передаче результата выражения в фортране;

%VAL - передача аргумента по значению. Формат:

CALL имя_программы (%VAL языковое_выражение).

Для выполнения этой функции отладчик вычисляет выражение и передает значение непосредственно подпрограмме.

Квалификаторы:

Отсутствуют.

Описание.

Когда пользователь выдает команду CALL, отладчик выполняет следующие действия:

1) сохраняет текущие значения общих регистров;

2) создает список аргументов;

3) выполняет вызов подпрограммы, указанной в команде, и передает ей имеющиеся аргументы;

4) выполняет данную подпрограмму;

5) воспроизводит выдаваемое подпрограммой значение в регистре R0;

6) восстанавливает значения общих регистров в соответствии с тем, какими они были непосредственно перед данной командой;

7) выдает свой запрос (DBG>).

Отладчик предполагает, что вызываемая процедура соответствует правилам вызова процедур системы МОС ВП.

Пользователь может вызвать процедуру и отлаживать ее независимо от остальной части программы. Команду CALL можно также использовать для вызова программ пользователя, которые собирают отлаживаемую информацию.

Примеры:

```
1. DBG>CALL SUB1(X)
ROUTINE START AT SUB1 (начало подпрограммы)
VALUE RETURNED IS 7FFF07DC (выдаваемая величина)
```

Эта команда вызывает подпрограмму SUB1, передавая ей требуемый параметр x.

```
2. DBG> CALL SUB(%REF 1)  !эта команда передает указатель
                          !ячейки памяти, содержащей
                          !числовой литерал 1 для подпрог-
```

!раммы SUB.

1.6. Команда CANCEL

Команда CANCEL отменяет точки приостанова, точки отслеживания и точки просмотра, а также восстанавливает диапазон, режимы ввода и воспроизведения, установленные пользователем, основания и типы в соответствии с их значениями по умолчанию. Отменяемый элемент определяется ключевым словом, указанным в команде.

Формат:

CANCEL ключевое_слово [параметры]

Параметры:

Ключевое_слово - определяет отменяемый элемент. С этой командой может быть использовано одно из следующих ключевых слов: ALL, BREAK, EXCEPTION BREAK, DISPLAY, MODE, MODULE, RADIX, SCOPE, SOURCE, TRACE, TYPE, WATCH или WINDOW.

Параметры - зависят от указанного ключевого слова.

Квалификаторы:

Квалификаторы зависят от указанного пользователем ключевого слова.

Пример.

```
DBG>SHOW BREAK
BREAKPOINT AT AVERAGE\%LINE 10
BREAKPOINT AT AVERAGE\%LINE 15
DBG>CANCEL BREAK %LINE 10
DBG>SHOW BREAK
BREAKPOINT AT AVERAGE\%LINE 15
```

```
DBG>
```

Этот пример показывает, как отменена точка приостанова в строке 10.

1.7. Команда CANCEL ALL

Команда CANCEL ALL отменяет все точки приостанова, отслеживания и просмотра, а также восстанавливает диапазон, режимы ввода и воспроизведения, установленные пользователем, и типы в соответствии с их значениями по умолчанию.

Команда CANCEL ALL не оказывает воздействия на текущий язык программирования и на модули, включенные в таблицу символических имен отладки.

Формат:

```
CANCEL ALL
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>CANCEL ALL
```

Эта команда отменяет все ранее установленные пользователем контрольные точки. Она также восстанавливает диапазон, режимы и типы в соответствии с их значениями по умолчанию.

1.8. Команда CANCEL BREAK

Команда CANCEL BREAK отменяет точки приостанова в ячейках, определяемых соответствующими квалификаторами или параметром (параметрами) адресного выражения.

Формат:

CANCEL BREAK [адресное_выражение[,...]]

Параметры:

Адресное_выражение - определяет ячейку, в которой должна быть отменена точка приостанова.

Квалификаторы:

/ALL - обеспечивает отмену всех точек приостанова.

/BRANCH - отменяет действие ранее установленной команды SET BREAK/BRANCH.

/CALL - отменяет действие ранее установленной команды SET BREAK/CALL.

/EXCEPTION - отменяет действие ранее установленной команды SET BREAK/EXCEPTION.

/INSTRUCTION - отменяет действие ранее установленной команды SET BREAK/INSTRUCTION.

/LINE - отменяет действие ранее установленной команды SET BREAK/LINE.

/MODIFY - отменяет действие ранее установленной команды SET BREAK/MODIFY.

Описание.

Пользователь должен определить либо адресное выражение, либо квалификатор (но не оба).

Следует отметить, что команда CANCEL ALL также отме-

няет все точки приостанова.

Примеры:

1. DBG>CAN BRE MAIN\LOOP+10

Эта команда отменяет точку приостанова, указанную адресным выражением MAIN LOOP+10.

2. DBG>CAN BRE/ALL

Эта команда отменяет все точки приостанова, установленные ранее пользователем.

1.9. Команда CANCEL DISPLAY

Команда CANCEL DISPLAY удаляет экранный кадр.

Формат:

CANCEL DISPLAY [имя_кадра,...]

Параметры:

Имя_кадра - указывает имя экранного кадра, которое надо удалить. Если пользователь указывает конкретный кадр, квалификатор /ALL не используется.

Квалификаторы:

/ALL - указывает, что все экранные кадры надо удалить.

Описание.

Команда CANCEL DISPLAY удаляет отдельный экранный кадр или сразу все кадры. Пользователь должен указать либо имя кадра, которое он хочет удалить, либо квалификатор /ALL, но не то и другое одновременно.

После удаления кадра его содержимое полностью теряется, он удаляется из списка кадров, и вся память, которая была ему отведена, освобождается.

Пример.

```
DBG>CANCEL DISPLAY/ALL
```

Эта команда удаляет все экранные кадры.

1.10. Команда CANCEL EXCEPTION BREAK

Команда CANCEL EXCEPTION BREAK отменяет точки приостанова по исключительной ситуации.

Формат:

```
CANCEL EXCEPTION BREAK
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Команда CANCEL EXCEPTION BREAK отменяет действие команды SET EXCEPTION BREAK. В результате отмены условия исключительной ситуации, генерируемые программой пользователя, обрабатываются следующим образом:

1) отладчик фиксирует и сообщает об исключительной ситуации;

2) если пользователь определил обработчик исключительных ситуаций в своей программе, то этот обработчик выполняется;

3) если пользователь не определил обработчик исключительных ситуаций или если этот обработчик повторно сообщает об исключительной ситуации, то выдается диагностическое сообщение и управление возвращается отладчику, который пос-

ле этого воспроизводит свой запрос.

Пример.

```
DBG>CANCEL EXCEPTION BREAK
```

Эта команда отменяет действие ранее установленной команды SET EXCEPTION BREAK.

1.11. Команда CANCEL MODE

Команда CANCEL MODE отменяет режим основания и все установки режимов, заданные командой SET MODE. В результате восстанавливаются значения режимов по умолчанию.

Формат:

```
CANCEL MODE
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>CANCEL MODE
```

Эта команда удаляет все режимы, установленные ранее командой SET MODE и восстанавливает значения режимов по умолчанию.

1.12. Команда CANCEL MODULE

Команда CANCEL MODULE удаляет символические имена, описанные в указанном модуле или модулях (или во всех модулях), из таблицы текущих символических имен (RST).

Формат:

```
CANCEL MODULE [имя_модуля[,имя_модуля...]]
```

Параметры:

Имя_модуля - указывает имя модуля, символические имена которого должны быть выведены из таблицы RST.

Квалификаторы:

/ALL - удаляет символические имена всех модулей из таблицы RST.

Описание.

Команда CANCEL MODULE может быть использована, когда таблица RST заполнена и необходимо стереть символические имена одного или нескольких модулей для того, чтобы обеспечить возможность размещения символических имен одного или нескольких других модулей.

Можно удалять символические имена одного модуля, списка модулей или всех модулей.

Примеры:

1. DBG>CAN MODU SUB1

Эта команда удаляет символические имена модуля SUB1 из таблицы RST.

2. DBG>CAN MODU/ALL

Эта команда удаляет символические имена всех модулей из таблицы RST.

1.13. Команда CANCEL RADIX

Команда CANCEL RADIX отменяет текущее основание, установленное командой SEI RADIX.

Формат:

CANCEL RADIX

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Команда CANCEL RADIX отменяет текущее основание, которое пользователь мог задать командой SET RADIX. После отмены устанавливается значение основания по умолчанию, которое для большинства языков программирования является десятичным.

Пример.

DBG>CANCEL RADIX

Данная команда отменяет все основания, которые пользователь мог установить командой SET RADIX, и восстанавливает основание по умолчанию.

1.14. Команда CANCEL RADIX/OVERRIDE

Команда CANCEL RADIX/OVERRIDE отменяет текущий режим основания, отменяющий другие режимы основания, установленный командой SET RADIX/OVERRIDE.

Формат:

CANCEL RADIX/OVERRIDE .

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Команда CANCEL RADIX/OVERREIDE ликвидирует текущий режим основания, отменяющий другие основания. Она заменяет текущий режим основания на другой, который пользователь может впоследствии задать командой SET RADIX. Если в течении сеанса отладки пользователь не укажет основания, то устанавливается режим основания по умолчанию, который для большинства языков программирования является десятичным.

Пример.

DBG>CANCEL RADIX/OVERRIDE

Эта команда отменяет действие команды SET RADIX/OVERRIDE.

1.15. Команда CANCEL SCOPE

Команда CANCEL SCOPE отменяет список поиска текущего диапазона, установленный командой SET SCOPE.

Формат:

CANCEL SCOPE

Параметры:

Отсутствуют

Квалификаторы:

Отсутствуют.

Описание.

Данная команда по своему действию эквивалентна команде SET SCOPE 0. В результате выполнения данной команды символические имена без префиксов имени пути предполагаются размещенными в модуле, содержащем счетчик программы PC (адрес последней выполненной команды).

Пример.

```
DBG>CAN SCO
```

Данная команда отменяет текущий диапазон.

1.16. Команда CANCEL SOURCE

Команда CANCEL SOURCE отменяет список обследуемых каталогов для поиска исходного файла, организованный предыдущей командой SET SOURCE.

Формат:

```
CANCEL SOURCE[/MODULE=имя_модуля]
```

Параметры:

Отсутствуют.

Квалификаторы:

/MODULE=имя_модуля - определяет имя модуля, для которого должен быть отменен список обследуемых каталогов.

Описание.

Данная команда без квалификатора отменяет действие предыдущей команды SET SOURCE; она не отменяет действие любых предшествующих ей команд SET SOURCE/MODULE=имя_модуля.

Команда CANCEL SOURCE/MODULE=имя_модуля отменяет действие предыдущей команды SET SOURCE/MODULE=имя_модуля, в которой указано такое же имя модуля. Она не отменяет действие предыдущей команды SET SOURCE или команды SET SOURCE/MODULE=имя_модуля, в которой указанно другое имя модуля.

Когда команда SET SOURCE отменяется, отладчик снова предполагает, что каждый исходный файл находится в том же каталоге, в котором он был во время компилирования.

Пример.

```
DBG>SH SOU
SOURCE DIRECTORY SEARCH LIST FOR COBOLTEST:
(список обследуемых каталогов для модуля COBOLTEST:)
       SYSTEM::DEVICE:[PROJD]
       [014,015]
SOURCE DIRECTORY SEARCH LIST FOR ALL OTHER MODULES:
(список обследуемых каталогов для всех остальных
модулей)
          [PROJA]
          [PROJB]
          [PETER.PROJC]
DBG>CAN SOU
DBG>SH SOU
SOURCE DIRECTORY SEARCH LIST FOR COBOLTEST:
```

(список обследуемых каталогов для модуля COBOLTEST

SYSTEM::DEVICE:[PROJD]

[014,015]

```
DBG>CAN SOU/MODU=COBOLTEST
DBG>SH SOU
NO DIRECTORY SEARCH LIST IN EFFECT
```

(никакой список обследуемых каталогов не действует)

Команда CANCEL SOURCE отменяет действие предыдущей команды SET SOURCE. Она не отменяет списка обследуемых каталогов для определенных модулей. Однако команда CANCEL SOURCE/MODULE=имя_модуля (в данном случае имя_модуля=COBOLTEST) отменяет список обследуемых каталогов для поиска исходного файла этого модуля.

1.17. Команда CANCEL TRACE

Команда CANCEL TRACE отменяет точку отслеживания в ячейках, указанных определенными квалификаторами или параметром (параметрами) адресное_выражение.

Формат:

```
CANCEL TRACE [адресное_выражение[,адресное_выражение,...]].
```

Параметры:

Адресное_выражение - определяет ячейку, в которой должна быть отменена точка отслеживания.

Квалификаторы:

/ALL - отменяет все точки отслеживания.

/BRANCH - отменяет действие предыдущей команды SET TRACE/BRANCH.

/CALL - отменяет действие предыдущей команды SET

TRACE/CALL.

/EXCEPTION - отменяет действие предыдущей команды SET TRACE/EXCEPTION.

/INSTRUCTION - отменяет действие предыдущей команды SET TRACE/INSTRUCTION.

/LINE - отменяет действие предыдущей команды SET TRACE/LINE.

/MODIFY - отменяет действие предыдущей команды SET TRACE/MODIFY.

Описание.

Если пользователь определяет адресное выражение в качестве параметра, то точка отслеживания в ячейке, указанной этим адресным выражением, отменяется. Если пользователь указывает квалификаторы, то отменяются точки отслеживания в инструкциях, отмеченных этими квалификаторами. Если пользователь указывает квалификатор /ALL, то отменяются все точки отслеживания.

Следует отметить, что команда CANCEL ALL также отменяет все точки отслеживания.

Примеры:

1. DBG>CAN TRA MAIN\LOOP+10

Эта команда отменяет точку отслеживания в ячейке MAIN\LOOP+10.

2. DBG>CAN TRA/ALL

Эта команда отменяет все установленные точки отслеживания.

1.13. Команда CANCEL TYPE/OVERRIDE.

Команда CANCEL TYPE/OVERRIDE отменяет тип, отвергающий другие типы, установленный командой SET TYPE/OVERRIDE, т.е. устанавливает его на "отсутствие".

Формат:

CANCEL TYPE/OVERRIDE

Параметры:

Отсутствуют.

Квалификаторы:

/OVERRIDE - обязательный квалификатор. Он имеет минимальное сокращение /OVERR.

Описание.

В результате команды CANCEL TYPE/OVERRIDE программные об'екты интерпретируются в сгенерированных компилятором типах или в типе по умолчанию.

Следует отметить, что команда CANCEL TYPE должна указываться с командным квалификатором /OVERRIDE.

Пример.

DBG>CAN TYPE/OVERR

Эта команда отменяет действие предыдущей команды SET TYPE/OVERRIDE.

1.19. Команда CANCEL WATCH

Команда CANCEL WATCH отменяет точку просмотра, установленную в ячейке, определяемой указанным адресным выражением, или все точки просмотра.

Формат:

CANCEL WATCH [адресное_выражение[,адресное_выражение,...]]

Параметры:

Адресное_выражение - определяет ячейку, в которой должна быть отменена точка просмотра.

Квалификатор:

/ALL - вызывает отмену всех точек просмотра.

Описание.

Если пользователь в качестве параметра данной команды определяет адресное выражение, то отменяется точка просмотра в ячейке, указанной этим адресным выражением. Если пользователь указывает квалификатор /ALL, то отменяются все точки просмотра.

Следует отметить, что команда CANCEL ALL также отменяет все точки просмотра.

Примеры:

1. DBG>CAN WAT SUB2\D

Эта команда отменяет точку просмотра в ячейке SUB2\D.

Эта команда отменяет все установленные точки просмотра.

1.20. Команда CANCEL WINDOW

Команда CANCEL WINDOW уничтожает определение экранного окна.

Формат:

CANCEL WINDOW [имя_окна,...]

Параметр:

Имя_окна - определяет имя отменяемого экранного окна. Если указывается какое-то конкретное окно, то квалификатор /ALL не используется.

Квалификатор:

/ALL - указывает на отмену всех определений экранных окон, как предопределенных отладчиком, так и заданных пользователем.

Описание.

Данная команда отменяет определение конкретного экранного окна или определения всех окон. Для этого указывается или имя отменяемого окна, или квалификатор /ALL, но не то и другое вместе. После отмены экранного окна его имя далее использовать в командах DISPLAY или SET DISPLAY нельзя.

Пример.

```
DBG> CANCEL WINDOW COMPARE
```

Эта команда отменяет определение экранного окна COMPARE.

1.21. Команды CTRL/C, CTRL/W, CTRL/Y, и CTRL/Z

Команды управления (клавиши) CTRL/C, CTRL/Y и CTRL/Z прерывают работу отладчика. CTRL/C и CTRL/Y передают управление интерпретатору команд, CTRL/Z приводит к окончанию сеанса отладки.

Команда CTRL/W обновляет экран при работе в экранном режиме отладки.

Формат:

CTRL/C

CTRL/W

CTRL/Y

CTRL/Z

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Если система или программа пользователя не определяет служебную подпрограмму CTRL/C, то действия в результате нажатия клавиш управляющих символов (команд) CTRL/Y или CTRL/C идентичны: образ прерывается, но не заменяется, входной буфер терминала очищается, и управление передается интерпретатору команд.

Прерывание выполнения программы с помощью команд CTRL/Y или CTRL/C удобно, если пользователь запустил свою программу без отладчика, но в процессе работы решил его использовать. Если программа перешла в бесконечный цикл, в

котором нет точки приостанова, то нажатием клавиши CTRL/Y и выдачей команды DEBUG этот цикл прерывается корректным образом. Управление возвращается отладчику.

После прерывания программы с помощью команды CTRL/Y или CTRL/C можно запустить или перезапустить отладчик с помощью команды DEBUG командного языка оператора DCL. Этот прием полезен для случая прерывания команды отладчика, не дожидаясь окончания ее выполнения. В целом нажатие клавиши CTRL/Y и выдача затем команды DEBUG приводят к выдаче запроса отладчика (DBG>).

Выдача команды CTRL/Z вызывает упорядоченное окончание сеанса отладки. Ее действие аналогично действию команды EXIT.

Нажатие клавиши CTRL/W дает тот же результат, что и команда DISPLAY/REFRESH.

Пример.

```
DBG>CTRL/Y  !прерывание сеанса отладки с помощью
^Y          !команды CTRL/Y.
$ DEBUG     !возобновление сеанса отладки с помощью
DBG>CTRL/Z  !команды DEBUG языка DCL, А затем окон-
$           !чание этого сеанса с помощью команды
            !CTRL/Z.
```

1.22. Команда DECLARE

Команда DECLARE позволяет понимать параметры командными процедурами.

Может использоваться только внутри командных процедур.

Формат:

DECLARE имя_параметра:тип[,имя_параметра:тип,...]

Параметры:

Имя_параметра - определяет имя формального параметра.

Тип - указывает тип параметра. Разрешены следующие ключевые слова для определения типа параметра:

ADDRESS - определяет адресное выражение. Действие - аналогично действию команды DEFINE/ADDRESS имя_параметра=...;

VALUE - определяет выражение в терминах текущего языка программирования. Такой же результат дает команда DEFINE/VALUE имя_параметра=...;

COMMAND - определяет строку символов, обрамленную кавычками аналогично команде DEFINE/COMMAND имя_параметра=...

Квалификаторы:

Отсутствуют.

Описание.

Команда DECLARE позволяет передавать параметры командным процедурам. Она связывает реальные параметры (которые стоят в командной строке за символом "D") с формальными параметрами (именами параметров внутри командной процедуры). Каждая заявка параметра действует аналогично команде DEFINE: она связывает имя параметра со значением или другой конструкцией. Сами параметры совместимы с параметрами, воспринимаемыми командой DEFINE и могут быть убраны из таблицы символов командой UNDEFINE. Более подробно можно узнать из описаний команд DEFINE и UNDEFINE.

Отдельной командой DECLARE можно определить несколько пар "имя_параметра:тип", разделяя их запятыми. Параметры воспринимаются в том же порядке, в каком они перечисляются в команде. Команду DECLARE можно использовать в цикле, пока не обработаются все параметры.

Нельзя использовать нуль-параметры (представленные двумя подряд запятыми или запятой в конце команды).

Пример.

```
¤CREATE DISPLAY.COM
DECLARE P1:ADDRESS
EXAMINE P1:P1+100


¤ RUN PROG
DBG>DEFINE/COMMAND DIS="@DISPLAY"
DBG>DIS X
X
X+4
X+8


X+100
```

Данный пример показывает, как команда DECLARE связывает параметр P1 внутри командной процедуры. Далее, во время отладки команда DEFINE/COMMAND присваивает имени DIS команду отладчика (@DISPLAY), вызывающую командную процедуру DISPLAY. После вызова командной процедуры опрашивается

адрес X (как параметр P1), как указано в команде EXAMINE P1:P+100.

1.23. Команда DEFINE

Команда DEFINE определяет одно или несколько символических имен и присваивает им определенные адреса на время соответствующего сеанса отладки.

Формат:

DEFINE символическое_имя=параметр[символическое_имя=-
параметр,...]

Параметры:

Символическое_имя - указывает определяемое символическое имя. При определении символических имен необходимо использовать правила:

1) символические имена могут комбинироваться из алфавитно-цифровых символов (A-Z и 0-9) и символов подчеркивания. Отладчик интерпретирует алфавитные символы нижнего регистра как символы верхнего регистра. Таким образом, слова "LOOP", написанные заглавными и строчными буквами, для отладчика одинаковы;

2) первый символ не может быть цифрой (0-9);

3) символическое имя не может превышать 31 символ.

Параметр - зависит от применяемого квалификатора.

Квалификаторы:

/ADDRESS - указывает, что определяемое имя является аббревиатурой адресного выражения. В этом случае "параметр" является адресным выражением. Вычисляется значение этого адресного выражения и результирующий адрес связывается с

данным символическим именем.

/COMMAND - указывает, что определяемое имя будет рассматриваться как новая команда отладчика. В этом случае "параметр" должен быть строкой символов, заключенной в кавычки. В простых случаях этот квалификатор по существу обеспечивает то же, что и команда языка DCL "символическое_имя:=параметр". Для определения сложных команд нужно использовать косвенные командные файлы с параметрами. За информацией об об'явлении параметров для командных файлов (или процедур) следует обращаться к описанию команды DECLARE.

/LOCAL - указывает, что определение остается локальным для косвенного командного файла, в котором имеется команда DEFINE. Другими словами, указанные символические имена являются неявными на уровне команд отладчика.

/VALUE - определяет имя как аббревиатуру для величины. "параметр" имеет форму выражения на текущем языке программирования. Вычисленное значение этого выражения присваивается указанному символическому имени. Величина может быть любого типа, поддерживаемого отладчиком для данного языка программирования.

Описание.

Определенное данной командой символическое имя можно использовать для обращения к адресу в образе. Можно, например, определить символическое имя для несимволической ячейки программы или для символической программной ячейки, имеющей длинный префикс имени пути. Это позволяет обращаться к этой ячейке программы по заново определенному символическому имени.

Далее символические имена можно использовать и в других случаях. Например, символическое имя можно присвоить значению (или результату оценки языкового выражения, а не адресному выражению).

Более того, можно определять аббревиатуру команд и даже определять новые команды из команд отладчика или из косвенных командных файлов.

При трансляции символических имен отладчик сначала выполняет поиск символических имен, определенных пользователем в течении сеанса отладки. Следовательно, если пользователь определяет символическое имя, которое уже существует в его программе, отладчик транслирует это символическое имя в соответствии с его определением (если только пользователь не указал префикс имени пути).

В одной команде DEFINE можно определить несколько символических имен, разделив их (а также разделив соответствующие выражения) запятыми.

Для просмотра значения, эквивалентного имени, можно применить команду SHOW SYMBOL/DEFINED *.

И, наконец, можно явно отменить использованное определение командой UNDEFINE.

Примеры:

1. DBG>DEF SCHK=MAIN\LOOP+10

Эта команда присваивает адресу MAIN\LOOP+10 имя SCHK.

2. DBG>DEFINE/VALUE COUNTER=0
DBG>SET TRACE/SILENT R DO (DEFINE/VALUE COUNTER=COUNTER+1)

Первая команда присваивает значение 0 имя COUNTER. Вторая команда указывает отладчику увеличивать значение величины COUNTER на единицу всякий раз, когда встретится

адрес R. Другими словами, в примере дано указание отладчику подсчитывать число вызовов R.

3. DBG>DEFINE COMMAND BRE="SET BREAK"

Эта команда присваивает имени BRE команду отладчика SET BREAK.

1.24. Команда DEFINE/KEY

Команда DEFINE/KEY позволяет назначать клавише терминала командную строку.

Формат:

DEFINE/KEY имя_клавиши "строка_эквивалентности"

Параметры:

Имя_клавиши - обозначает требуемую клавишу. Стандартные обозначения клавиш приведены в табл.1.

Строка_эквивалентности - обозначает функцию или команду, которую должна выполнять клавиша. Если состоит более чем из одного слова, то заключить в кавычки.

Таблица 1

Перечень обозначений клавиш терминала

Обозначение клавиши	Наименование клавиши (тип терминала)
PF1	Красная (LK201,VT100,VT52)
PF2	Голубая (LK201,VT100,VT52)
PF3	Черная (LK201,VT100,VT52)
PF4	(LK201,VT100)
KP0,KP1,...,KP9	Цифры от 0 до 9
PERIOD	Точка
COMMA	Запятая

Продолжение табл. 1

Обозначение клавиши	Наименование клавиши (тип терминала)
MINUS	Минус
ENTER	Ввод
UP	Стрелка вверх
DOWN	Стрелка вниз
LEFT	Стрелка влево
RIGHT	Стрелка вправо
E1	Найти (LK201)
E2	Вставить здесь (LK201)
E3	Удалить (LK201)
E4	Выбрать (LK201)
E5	Предыдущий экран (LK201)
E6	Последующий экран (LK201)
HELP	Помочь (LK201)
DO	Выполнить (LK201)
F6,F7,...,F20	Клавиши функций (LK201)

Квалификаторы команды

/[NO]ECHO - определяет необходимость повторения содержания командной строки после считывания ("эхо"). По умолчанию принимается значение /ECHO. Нельзя использовать /NOECHO с квалификатором /NOTERMINATE.

/[NO]IF_STATE = состояние - определяет перечень состояний, при которых должны задействоваться определяемые клавиши. Если квалификатор IF_STATE не указан или задан /NOIF_STATE, используется текущее состояние. Наименование состояния представляет собой буквенно-числовую строку. Состояние устанавливается квалификатором /SET_STATE.

/LOCK_STATE - означает, что состояние, установленное квалификатором /SET_STATE, остается в действии до тех пор, пока не будет полностью изменено.

/NOLOCK_STATE - указывает, что установка состояния с помощью /SET_STATE действует только до следующей нажатой клавиши или до следующего знака окончания считывания, вводимого пользователем. /NOLOCK_STATE действует по умолчанию, пока не будет указан квалификатор /TERMINATE.

/[NO]LOG - управляет выдачей сообщения об успешном выполнении операции определения клавиши. По умолчанию действует /LOG.

/[NO]SET_STATE - указывает требуемое состояние для определения клавиш. Если /SET_STATE опущен или задан /NOSET_STATE, остается задействованным только текущее состояние.

/[NO]TERMINATE - определяет, надо ли заканчивать текущую строку эквивалентности или обрабатывать ее при нажатии клавиши. По умолчанию действует /NOTERMINATE, который позволяет нажимать другие клавиши до обработки строки эквивалентности. Использование /TERMINATE имеет тот же эффект, что и нажатие клавиши RETURN.

Описание.

Команда DEFINE/KEY позволяет присваивать клавишам терминала строку эквивалентности (командную строку). В дальнейшем при нажатии такой клавиши отладчик вводит соответствующую ей строку в командную строку пользователя. Система должна быть установлена в режим работы с клавиатурой.

Определение клавиши действует до тех пор, пока оно не будет определено заново или будет отменено командой

DELETE/KEY; определение клавиши не действует после завершения отладчика. Определение клавиш можно включить в командный файл, например, в файл инициализации отладчика.

Параметр состояния, употребляемый с квалификаторами /IF_STATE, /LOCK_STATE и /SET_STATE, может иметь различные определения в зависимости от выбранного состояния.

Пример.

```
DBG>DEFINE/KEY/TERMINATE PF1 "SET RADIX/OVERRIDE HEX".
```

Пример показывает, как связать команду SET RADIX/OVERRIDE HEX с клавишей PF1.

1.25. Команда DELETE

Команда DELETE удаляет определение символического имени из таблицы глобальных символических имен (GST) или таблицы локальных символических имен. Глобальным символическим именем является то имя, которое определено с помощью команды DEFINE без квалификатора /LOCAL. Локальным - то, которое определено в косвенном командном файле отладчика командой DEFINE/LOCAL, так что его определение ограничивается этим командным файлом. Команда DELETE идентична команде UNDEFINE.

Формат:

```
DELETE [имя[,...]]
```

Параметр:

Имя - указывает одно или несколько символических имен, чьи определения должны быть удалены из таблицы локальных или глобальных символических имен. Если используется /ALL, параметр не указывается. При задании /LOCAL указываемые

имена должны быть ранее определены командой DEFINE/LOCAL. Если /LOCAL не задается, то указываемые имена должны быть ранее определены с помощью команды DEFINE без квалификатора /LOCAL.

Квалификаторы:

/ALL - удаляет все определения глобальных символических имен. Если при этом указывается /LOCAL, то удаляются все определения локальных символических имен, связанные с текущим косвенным командным файлом (но не определения глобальных символических имен).

/LOCAL - удаляет определения указанных имен (локальных) из текущего косвенного командного файла. Данные имена должны быть заранее определены с помощью команды DEFINE/LOCAL.

Примеры:

1. DBG>DEFINE X=INARR, Y=OUTARR

DBG>DELETE X,Y

Команда DEFINE определяет X и Y как глобальные символические имена, соответствующие INARR и OUTARR. Команда DELETE выводит эти два определения из таблицы GST.

2. DBG>DELETE /ALL/LOCAL

Команда DELETE/ALL/LOCAL удаляет все определения локальных символических имен из текущего косвенного командного файла.

1.26. Команда DELETE/KEY

Команда DELETE/KEY стирает определение клавиши, которое было установлено с помощью команды DEFINE/KEY.

Формат:

DELETE/KEY [имя_клавиши]

Параметр:

Имя_клавиши - указывает наименование клавиши, чье определение необходимо стереть. Этот параметр не задается, если используется квалификатор /ALL.

Квалификаторы:

/ALL - указывает, что надо стереть все определения клавиш в обозначенном состоянии. Если используется квалификатор /ALL, имя клавиши не указывается. Если состояние не задается, стираются все определения клавиш для текущего состояния. Для указания одного или более состояний используется квалификатор /STATE.

/[NO]LOG - управляет выдачей сообщения о стирании указанных определений клавиш. /LOG работает по умолчанию и указывает, что сообщение о стирании должно отображаться.

/[NO]STATE=(имя_состояния[,...]) - указывает имя состояния, для которого обозначенные определения клавиш должны стираться. Если задается только одно имя состояния, круглые скобки можно опустить. Именем состояния может быть любая подходящая буквенно-цифровая строка.

Если квалификатор /STATE не указывается или используется /NOSTATE, стираются определения клавиш для текущего состояния.

Описание.

Чтобы стереть определение клавиши, должен быть установлен режим работы с малой клавиатурой. Команда DELETE/KEY обладает таким же действием, что и команда UNDEFINE/KEY.

Пример.

```
DBG>DELETE/KEY KP4
```

Команда DELETE/KEY стирает определение для клавиши KP4 для состояния, установленного последней командой SET KEY (по умолчанию это состояние - DEFAULT).

1.27. Команда DEPOSIT

Команда DEPOZIT заносит выражение, приводимое непосредственно после знака равенства, в ячейку, определяемую адресным выражением.

Формат:

```
DEPOSIT адресное_выражение=выражение[,выражение...]
```

Параметры:

Адресное_выражение - определяет ячейку, в которую должно быть занесено выражение, непосредственно следующее за знаком равенства.

Команда DEPOSIT устанавливает символическое имя текущего об'екта (.) в программную ячейку, определяемую адресным выражением. Логически последующие ячейки вычисляются в этом случае с использованием значения символического имени текущего об'екта.

Выражение - определяет значение, которое должно быть занесено.

При выполнении команды DEPOSIT выражение вычисляется в

соответствии с синтаксисом исходного языка. Значение этого выражения преобразуется в тип, связанный с типом адресного выражения, и помещается в ячейку, определяемую этим адресным выражением.

Если выражение является цепочкой кодов КОИ-8 или команд ассемблера СМ 1700, оно должно быть заключено в кавычки или апострофы.

Квалификаторы:

/ASCIC - заносит подсчитанную строку КОИ-8 в адресное выражение.

/ASCID - интерпретирует адресное выражение как адрес дескриптора строк, указывающий на строку КОИ-8. Поля CLASS и DTYPE дескриптора не проверяются, а поля LENGTH и POINTER обеспечивают длину и адрес строки КОИ-8, после чего строка заносится.

/ASCII:N - связывает тип символов кода КОИ-8 (длиной в N байтов) с ячейкой, определяемой соответствующим адресным выражением. Величина N интерпретируется с основанием, определяемым командным квалификатором режима основания или текущим режимом основания. Если величина N опущена, отладчик использует длину указанной строки.

/ASCIW - интерпретирует адресное выражение как адрес переменной строки (двухбайтовое поле длины, за которым непосредственно следует строка КОИ-8 указанной длины). Этот тип данных встречается в языках Паскаль и Пл/1.

/ASCIZ - заносит оканчивающуюся нулем строку КОИ-8 в адресное выражение.

/BYTE - связывает тип байтового целого (длиной в один байт) с ячейкой, определенной соответствующим адресным

выражением.

/D_FLOAT - связывает тип вещественного числа формата D (длиной 8 байтов) с ячейкой, определяемой соответствующим адресным выражением. Значения чисел этого типа могут быть от .29*10**(-38) до 1.7*10**38 с точностью примерно 16 десятичных цифр.

/DATE_TIME - преобразует строку, представляющую дату и время (например, 31-MAR-1986:08:00), во внутренний формат МОС ВП для даты и времени и заносит это квадрослово в адресное выражение.

/FLOAT - связывает тип вещественного числа формата F (длиной 4 байта), с ячейкой, обозначенной адресным выражением. Значения этого числа могут варьироваться от .29*10**(-38) до 1.7*10**38 с точностью примерно 7 десятичных цифр.

/G_FLOAT - связывает тип вещественного числа формата G (длиной 8 байтов) с ячейкой, определяемой адресным выражением. Величины этого типа могут иметь диапазон от .56*10**(-308) до .9*10**308 с точностью приблизительно 15 десятичных цифр.

/H_FLOAT - связывает тип вещественного числа формата H (длиной 16 байтов) с ячейкой, определяемой адресным выражением. Величины этого типа могут иметь диапазон от .84*10**(-4932) до .59*10**4932 с точностью приблизительно 33 десятичные цифры.

/INSTRUCTION - связывает тип команд ассемблера СМ 1700 (переменной длины) с ячейкой, определяемой адресным выражением. Длина зависит от числа операндов и типа режимов адресации, используемого в команде ассемблера.

/LONGWORD - определяет тип целого двойного слова (дли-

ной в 4 байта), который должен быть связан с ячейкой, определяемой адресным выражением.

/OCTAWORD - связывает тип целого октаслова (длиной 16 байтов) с ячейкой, определенной адресным выражением.

/PACKED:N - заносит упакованное десятичное значение длиной N в ячейку, определенную адресным выражением.

/QUADWORD - определяет тип целого квадрослова (длиной в 8 байтов) в ячейке, указанной адресным выражением.

/WORD - определяет тип целого слова (длиной в 2 байта) в ячейке, указанной адресным выражением.

Описание.

Командные квалификаторы типа могут быть использованы для связи типа с ячейкой, определяемой адресным выражением, а также с логически последующими ячейками (в случае, если выполняется занесение нескольких выражений).

Более того, для воздействия на интерпретацию целых как в адресных выражениях, так и в выражениях можно использовать операторы основания. Дополнительная информация по использованию операторов основания приведена в подразделе 6.1 26.00152-01 34 07-1.

Примеры:

1. DBG>DEPOSIT X=A+2.5

Эта команда заносит выражение A+2.5 в ячейку X.

2. DBG>DEPOSIT/BYTE WORK=%HEX 212

Эта команда заносит выражение %HEX 212 в ячейку WORK и преобразует его в байтовое целое.

3. DBG>DEPOSIT/OCTAWORD BIGINT=111222333444555

Эта команда заносит выражение 111222333444555 в ячейку BIGINT и преобразует его в целое октаслово.

4. DBG>DEPOSIT/FLOAT BIGFLT=1.11949*10**35

Эта команда заносит выражение 1.11949*10**35 в ячейку BIGFLT и преобразует его в число формата F.

5. DBG>DEPOSIT/ASCII:10 WORK+20='ABCDEFGHIJ'

Эта команда заносит выражение 'ABSDEFGHIJ' в ячейку WORK+20.

6. DBG>DEPOSIT/INSTR SUB2+2='MOVL #20A,R0'

Эта команда заносит команду MOVL #20A,R0 в ячейку SUB2+2.

1.28. Команда DISABLE AST

Команда DISABLE AST отключает поступление сигналов AST (асинхронных системных прерываний) в программу пользователя. Это ограждает программу от таких прерываний при отладке. Команда ENABLE AST возобновляет подачу прерываний, в том числе уже ожидающих.

Формат:

DISABLE AST

Параметры:

Отсутствуют

Квалификаторы:

Отсутствуют

Пример.

```
DBG>DISABLE AST
DBG>SHOW AST
ASTS ARE DISABLED
```

Команда DISABLE AST закрывает поступление AST, что

подтверждается командой SHOW AST.

1.29. Команда DISPLAY

Команда DISPLAY отображает одно или несколько изображений (кадров) на экране терминала.

Формат:

DISPLAY [имя-кадра[AT спецификация_окна][тип_кадра][,...]].

Параметры:

Имя_кадра - определяет необходимый экранный кадр. Можно задать какое-либо из следующих имен:

- предопределенное отладчиком: SRC, OUT, INST, REG;

- имя кадра, предварительно созданного командой SET DISPLAY;

- псевдо-имя: %CURDISP, %NEXTSCROLL, %NEXTDISP, %NEXTINST, %NEXTOUTPUT, %NEXTSOURCE.

Этот параметр обязателен, если не используется /CLEAR (параметр необязателен), /GENERATE (параметр необязателен) или /REFRESH (параметр недопустим).

Можно указывать более одного кадра, каждый со своей необязательной спецификацией окна и типом кадра.

Спецификация_окна - определяет экранное окно, в которое должен быть помещен кадр при изменении его положения. Можно указать следующее:

- предопределенное отладчиком окно. Например, FS (весь экран), H1 (верхняя половина), Q3 (третья четверть), T2 (средняя треть), Q23 (средние две четверти);

- окно, предварительно определенное командой SET WINDOW;

- спецификацию окна в виде начальной строки и количества строк. Например, (12,4) указывает окно, начинающееся со строки 12 и занимающее 4 строки.

Если параметр опущен, положение кадра на экране не меняется.

Тип_кадра - указывает новый тип кадра, если требуется его изменить. Ключевые слова для этого параметра включают:

1) DO(список_команд) - обозначает автоматически корректируемый кадр выходной информации. Команды списка выполняются каждый раз, когда управление переходит к отладчику. Их выход образует содержимое кадра;

2) INSTRUCTION - указывает некорректируемый автоматически кадр команд ассемблера. Этот кадр содержит выход команды EXAMINE/INSTRUCTION. В этот кадр ничего не записывается до тех пор, пока он не задан в качестве текущего кадра команд ассемблера командой SELECT/INSTRUCTION;

3) INSTRUCTION(список_команд) - указывает автоматически корректируемый кадр команд ассемблера. В списке команд необходимо определить отдельную команду EXAMINE/INSTRUCTION. Кадр корректируется каждый раз, когда управление переходит к отладчику;

4) NORMAL - указывает обычный кадр выходной информации. В него ничего не записывается, пока он не выбран в качестве текущего кадра выходной информации с помощью команды SELECT/OUTPUT;

5) REGISTER - указывает автоматически корректируемый кадр содержимого регистров. Кадр корректируется каждый раз, когда управление переходит к отладчику;

6) SOURCE - указывает кадр исходного текста, который автоматически не корректируется. Этот кадр содержит выход

команд TYPE или EXAMINE/SOURCE. В него ничего не записывается до тех пор, пока он не выбран в качестве текущего кадра исходного текста с помощью команды SELECT/SOURCE;

7) SOURCE(список_команд) - указывает автоматически корректируемый кадр исходного текста. В списке команд надо указать отдельную команду TYPE или EXAMINE/SOURCE. Кадр корректируется каждый раз, когда управление переходит к отладчику;

Квалификаторы:

/CLEAR - стирает все содержимое указанных экранных кадров. Если ни один кадр не задан, стирается содержимое всех кадров.

/GENERATE - восстанавливает содержимое указанных кадров. Восстанавливаются только автоматически генерированные кадры. Если имя кадра не указано, восстанавливается содержимое всех автоматически генерированных кадров.

/HIDE - скрывает указанный кадр под другими кадрами, которые занимают ту же область на экране терминала. В результате какой-либо другой, который был ранее скрыт под указанным кадром, становится видимым.

/[NO]MARK_CHANGE - управляет отметкой измененных строк на кадрах типа DO(список_команд) каждый раз при автоматической коррекции кадра. При использовании /MARK_CHANGE все строки, содержимое которых изменялось со времени последней корректировки, высвечиваются в обратном изображении. Этот квалификатор особенно удобен, когда желательно, чтобы все переменные в автоматически корректируемом кадре выделялись при своем изменении. /NOMARK_CHANGE - указывает, что нет необходимости выделять изменяющиеся строки кадра типа

DO(список_команд). Этот квалификатор отменяет действие ранее выданного квалификатора /MARK_CHANGE для указанного кадра.

Этот квалификатор неприменим к другим типам кадров.

/REFRESH - обновляет экран терминала. При указании этого квалификатора параметры команды не используются.

/REMOVE - отмечает кадр как удаленный с экрана, поэтому его нельзя будет выдать на экран, пока явно не затребовать этого командой DISPLAY. Хотя удаленный с экрана кадр не виден на экране, он существует, и его содержимое сохраняется.

/SIZE:N - изменяет максимальный размер кадра до N строк. Если в кадр записано больше строк, чем N, самые ранние его строки теряются и на них пишутся новые. Если этот квалификатор опущен, максимальный размер кадра не меняется.

Описание.

Команда DISPLAY выполняет различные функции. Ее основной функцией является показ содержимого требуемого кадра. Кадр помещается над другими кадрами, занимающими то же окно на экране терминала. Заданный кадр становится видимым, а все другие кадры, которые занимают это же окно, скрываются под ним (хотя и продолжают существовать).

Однако данную команду с определенными квалификаторами можно использовать для удаления кадра с экрана терминала или для обновления всего экрана. Можно так же применять эту команду для изменения окна, для изменения максимального размера кадра (в строках), для изменения типа кадра или для изменения списка команд, подлежащих выполнению.

Примеры:

1. DBG>DISPLAY REG

Эта команда выдает содержимое предопределенного кадра REG.

2. DBG>DISPLAY/SIZE:5 SRC

Эта команда показывает 5 строк предопределенного кадра SRC.

3. DBG>DISPLAY NEWDISP AT T2

Эта команда отображает определенный пользователем кадр NEWDISP во второй трети экрана.

1.30. Команда EDIT

Команда EDIT вызывает языково-ориентируемый редактор МОС ЭП. Команду можно использовать только при наличии такого редактора в системе.

Формат:

EDIT[[имя_модуля\]номер_строки]

Параметры:

Имя_модуля - указывает имя модуля, чей исходный файл должен редактироваться. Если имя модуля задается, то нужно также указать и номер строки. Если имя модуля опускается, то для редактирования выбирается файл, содержащий выполняющийся модуль.

Номер_строки - указывает номер строки (положительное целое число), на которой должен центрироваться кадр исходного текста. Если параметр опущен, кадр исходного текста центрируется по текущему значению счетчика pc.

Квалификаторы:

/[NO]EXIT - управляет окончанием процесса отладки перед запросом редактора. Если указывается /EXIT, процесс отладки заканчивается и вызывается редактор. Если указывается /NOEXIT (действует по умолчанию), сеанс редактирования осуществляется как под процесс, из которого происходит возврат к сеансу отладки после завершения редактирования.

Описание.

Команда EDIT вызывает редактор, который должен быть включен в систему. Обычно эта команда используется без параметров. В этом случае редактор центрирует кадр исходного текста, предназначенного для редактирования, по значению счетчика исполняющегося в данный момент модуля. Таким образом, если отладка проводилась в экранном режиме, кадр редактора соответствует текущему кадру исходного текста отладчика.

Можно отменить действие по умолчанию, указывая или номер строки, или одновременно и имя модуля и номер строки. Отладчик выдаст соответствующий файл с исходным текстом.

Пример.

```
DBG>EDIT SWAP\12
```

Команда EDIT запускает редактор для редактирования файла исходного текста, содержащего модуль SWAP. Кадр редактора центрируется по строке 12 исходного текста модуля SWAP.

1.31. Команда ENABLE AST

Команда ENABLE AST разрешает поступление асинхронных системных прерываний (AST) в программу пользователя, включая ожидающих своей очереди AST. С начала работы поступление AST разрешается по умолчанию.

Формат:

```
ENABLE AST
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>ENABLE AST
DBG>SHOW AST
ASTS ARE ENABLED
```

Команда ENABLE AST разрешает поступление AST в программу пользователя, что подтверждается с помощью команды SHOW AST.

1.32. Команда EVALUATE

Команда EVALUATE выполняет оценку выражения. Отладчик интерпретирует параметр в данной команде в качестве выражения на исходном языке, оценивает его в соответствии с синтаксисом и семантикой исходного языка и воспроизводит его величину в форме литерала на исходном языке программирования.

Формат:

EVALUATE выражение[,выражение...]

Параметры

Выражение - задает любое допустимое выражение на исходном языке, но может быть скомбинированно из чисел и одного или нескольких знаков операций, операндов или разделителей.

Квалификаторы:

/BINARY - указывает, что результат будет выдан с двоичным основанием.

/CONDITION_VALUE - указывает, что выражение должно интерпретироваться как значение условия (отчасти значение условия можно было бы задать, используя механизм обработки условий). Затем выдается текст сообщения, соответствующий этому значению условия. Указываемое значение должно быть целой величиной.

/DECIMAL - указывает, что результат будет выдан с десятичным основанием.

/HEXADECIMAL - указывает, что результат будет выдан с шестнадцатеричным основанием.

/OCTAL - указывает, что результат будет выдан с восьмеричным основанием.

Описание.

Если выражение содержит символические имена с различными генерируемыми компилятором типами, то отладчик использует правила преобразования типа текущего языка для оценки этого выражения.

Если указан командный квалификатор режима основания,

то отладчик воспроизводит значение этого выражения в форме литерала с этим основанием. Если квалификатор режима основания не указан, то действия те же, что и при обработке литералов.

С помощью одной команды EVALUATE можно оценивать несколько выражений, разделяя их запятыми.

Используя данную команду, можно выполнять арифметические вычисления, которые могут быть и не связаны с программой пользователя. Таким образом, отладчик может быть использован в качестве калькулятора.

Детальная информация о допустимых операторах и типах данных приведена в приложении.

Примеры:

1. DBG>EVALUATE 100.34*(14.2+7.9)

   2217.514

Эта команда использует отладчик как калькулятор, умножая 100.34 на (14.2+7.9).

2. DBG>EV/OCTAL X

   1512

Эта команда оценивает символическое имя X и выдает результат с восьмеричным основанием.

1.33. Команда EVALUATE/ADDRESS

Команда EVALUATE/ADDRESS выполняет оценку адресного выражения и воспроизводит результаты в форме литерала с указанным или установленным по умолчанию режимом основания.

Формат:

EVALUATE/ADDRESS адресное_выражение[,адресное_выражение...] .

Параметры:

Адресное_выражение - указывает адресное выражение, которое может быть комбинацией из чисел для символических имен, а также из одного или нескольких знаков операций, операндов или разделителей.

Квалификаторы:

/BINARY - указывает, что результат выдается с двоичным основанием.

/DECIMAL - указывает, что результат выдается с десятичным основанием.

/HEXADECIMAL - указывает, что результат выдается с шестнадцатеричным основанием.

/OCTAL - указывает, что результат выдается с восьмеричным основанием.

Описание.

Данная команда используется для определения виртуального адреса (или адресов), соответствующего указанному адресному выражению.

Если указан командный квалификатор режима основания, то отладчик воспроизводит величину адресного выражения в форме литерала с этим основанием.

Одной командой EVALUATE/ADDRESS можно оценивать несколько адресных выражений, указав их через запятую.

Примеры:

1. DBG>EVALUATE/ADRESS MODNAME\%LINE 110

Эта команда определяет значение адресного выражения MODNAME\%LINE 110

2. DBG>EVALUATE/ADRESS/HEX Y

Эта команда определяет величину адресного выражения Y и выдает результат с шестнадцатеричным основанием.

3. DBG>EVALUATE/ADRESS A,B,C

Эта команда определяет величину адресных выражений A, B, C

1.34. Команда EXAMINE

Команда EXAMINE воспроизводит величину об'екта в ячейке, указанной адресным выражением, и в типе, связанном с этой ячейкой.

Формат:

EXAMINE [адресное_выражение[:адресное_выражение][,...]

Параметр:

Адресное_выражение - определяет просматриваемый об'ект.

Отладчик оценивает адресное выражение в соответствии с его собственными, независимыми от языка программирования правилами для получения виртуального адреса и интерпретирует этот адрес как ячейку, в которой находится просматриваемый об'ект.

Адресное выражение может комбинироваться из чисел и символических имен, а также из одного или нескольких знаков операций, операндов или разделителей.

Если должен быть просмотрен ряд об'ектов, то значение адресного выражения, которое указывает первый об'ект в этом ряду, должно быть меньше значения адресного выражения, указывающего последний об'ект данного ряда. Если должен быть просмотрен список об'ектов, то адресные выражения, которые их определяют, могут быть приведены в команде в любом порядке.

Квалификаторы:

/ASCIC - выдает адресное выражение как подсчитанную строку КОИ-8.

/ASCID - интерпретирует адресное выражение как адрес дескриптора строки, указывающий на строку КОИ-8. Поля CLASS и DTYPE дескриптора не проверяются, а поля LENGTH и POINTER определяют длину и адрес строки КОИ-8. Затем строка выдается на дисплей.

/ASCII:N - выдает на дисплей рассматриваемый об'ЕКТ или об'ЕКТы в виде символов кода КОИ-8 (длина N байтов). Длина определяет как число байтов просматриваемой памяти, так и число выдаваемых на дисплей символов кода кои-8. Если величина N не приводится, то отладчик пытается определить длину из типа адресного выражения.

/ASCIW - интерпретирует адресное выражение как адрес строки переменной длины (двухбайтовое поле, за которым сразу следует строка символов кода КОИ-8 указанной длины). Этот тип данных встречается в языках Паскаль и Пл/1.

/ASCIZ - выдает адресное выражение как строку КОИ-8, оканчивающуюся нулем.

/BINARY - выдает рассматриваемый об'ект или об'екты с двоичным основанием.

/BYTE - указывает, что соответствующий просматриваемый об'ект или об'екты должны воспроизводиться в типе байтового целого (длина 1 байт).

/CONDITION_VALUE - интерпретирует адресное выражение как значение условия (тип условия надо задавать, используя механизм обработки состояний).затем выдается текст сообщения, соответствующего этому значению условия.

/D_FLOAT - выдает рассматриваемый об'ект или об'екты в виде вещественного числа формата D (длина 8 байтов). Величины этого типа могут иметь диапазон от .29*10**(-38) до 1.7*10**38 с точностью приблизительно 16 значащих цифр.

/DATE_TIME - выдает рассматриваемый об'ект или об'екты как 64-битовое представление даты и времени.

/DECIMAL - выдает рассматриваемый об'ект или об'екты с десятичным основанием.

/DEFUALT - выдает рассматриваемый об'ект или об'екты с основанием по умолчанию.

/FLOAT - выдает рассматриваемый об'ект или об'екты в виде вещественного числа формата F (длина 4 байта). Величины этого типа могут иметь диапазон от .29*10**(-38) до 1.7*10**38 с точностью приблизительно 7 значащих цифр.

/G_FLOAT - выдает рассматриваемый об'ект или об'екты в виде вещественного числа формата G (длина 8 байтов). Величины этого типа могут иметь диапазон от .56*10**(-308) до .9*10**308 с приблизительной точностью 15 десятичных цифр.

/H_FLOAT - выдает рассматриваемый об'ект или об'екты в виде вещественного числа формата H (длина 16 байтов). Величины этого типа могут иметь диапазон от .84*10**(-4932) до .59*10**4932 с точностью приблизительно 33 десятичных цифр.

/HEXADECIMAL - выдает рассматриваемый об'ект или

об'екты с шестнадцатеричным основанием.

/INSTRUCTION - выдает рассматриваемый об'ЕКТ или об'ЕКТы в виде команд ассемблера СМ 1700 (с переменной длиной). Длина может меняться в зависимости от числа операндов и типов режимов адресации, используемых в этой команде ассемблера.

/[NO]LINE - управляет выдачей ячеек программы в виде номеров строк (%LINE) или в виде "программа + смещение в байтах". По умолчанию отладчик выдает номера строк.

/LONG - указывает, что соответствующий просматриваемый об'ект или об'екты должны воспроизводиться в виде длинного целого числа (длина 4 байта).

/OCTAL - выдает рассматриваемый об'ект или об'екты с восьмеричным основанием.

/OCTAWORD - указывает, что соответствующий просматриваемый об'ект или об'екты должны воспроизводиться в виде целого октаслова (длина 16 байтов).

/PACKED - выдает рассматриваемый об'ект или об'екты как "упакованные" представления чисел.

/PSL - выдает рассматриваемый об'ект или об'екты как длинное слово состояния процессора, интерпретируя различные битовые поля.

/PSW - выдает содержимое второго слова длинного слова состояния процессора, интерпретируя различные битовые поля.

/QUADWORD - указывает, что соответствующий просматриваемый об'ект или об'екты должны воспроизводиться в виде целого квадрослова (длина 8 байтов).

/SOURCE - указывает, что отладчик выдает исходную строку (строки), соответствующие ячейке или ячейкам, определенным адресным выражением (или выражениями).

/[NO]SYMBOL - управляет появлением символизации. По умолчанию отладчик символизирует все адреса. (следовательно, команда EXAMINE эквивалентна EXAMINE/SYMBOL). /NOSYMBOL можно использовать для подавления символизации и увеличения скорости обработки команд.

/WORD - указывает, что соответствующий просматриваемый об'ект или об'екты должны воспроизводиться в виде целого слова (длина 2 байта).

Описание.

С помощью одной команды EXAMINE можно просматривать несколько об'ектов, указав несколько адресных выражений через запятые.

Можно просматривать диапазон об'ектов с помощью одной данной команды, указывая адресное выражение, которое определяет первый об'ект этого диапазона, двоеточие и адресное выражение, которое указывает последний об'ект этого диапазона.

Можно так же просматривать список диапазонов об'ектов, указав диапазоны через запятые.

Если пользователь определяет квалификатор типа, то отладчик воспроизводит соответствующий об'ект (или об'екты) в этом типе.

Если пользователь определяет командный квалификатор режима основания, то результат выдается с этим основанием. Квалификатор режима основания не влияет на интерпретацию отладчиком самого об'екта. При использовании квалификатора /SYMBOL отладчик воспроизводит адресные выражения и операнды в символической форме (если это возможно). При определении командного квалификатора /NOSYMBOL отладчик воспроизво-

дит символическую информацию в эквивалентной числовой форме.

При просмотре длинного слова состояния процессора (PSL) в символьном или несимвольном режиме отладчик воспроизводит его содержимое в форматированном виде.

При задании квалификатора /SOURCE отладчик воспроизводит исходную строку (или строки), соответствующую ячейке (или ячейкам), указанной адресным выражением (или выражениями).

Просмотр совокупности элементов (составные структуры данных, такие, как массив или запись) отображается в виде отдельных ее компонентов. Это позволяет просматривать совокупность как одно целое: нет необходимости указывать каждый элемент отдельно. Выходная информация о массиве показывает индекс и значение каждого его элемента. Выходная информация о записи показывает имя и значение каждой ее компоненты. Многие языки программирования допускают также массивы записей и записи, содержащие массивы. В этом случае получение выходной информации является рекурсивной операцией.

Подобным же образом рассмотрение массива выдает часть его элементов, называемую "кусок массива". Это свойство позволяет рассматривать части массива как целое; нет необходимости определять и рассматривать каждый элемент массива в отдельности. Ряды индексов определяются параметрами адресных выражений; индексы составляют ограниченный, но непрерывный ряд значений. Как и обычные значения индексов, они могут отделяться точкой с запятой. Каждое такое значение индекса может определяться языковым выражением соответствующего типа, которым обычно является целое число или перечисление (перечень). Верхний предел всегда должен быть

больше или равен нижнему пределу. Ряды индексов могут смешиваться с отдельными индексами в одном и том же списке. Например, "NAMEARRAY(2,1:10,5)" является допустимым куском массива. Фактически, кусок массива является просто обычным связанным массивом, пределы которого более жестки, чем пределы основного массива. Например, если массив A языка фортран имеет размерность A(10,10), тогда определение "A(2:4,7:8)" даст кусок массива, составленный из элементов A(2,7), A(2,8), A(3,7), A(3,8), A(4,7), A(4,8).

Можно также обозначить ряд индексов одной звездочкой. Звездочка представляет собой полный ряд индексов основного массива. Например, команда EXAMINE NAMEARRAY(5,*) выдает все элементы ряда 5 массива NAMEARRAY.

Примеры:

1. DBG>EXAMINE/ASC WORK+20

DETAT\WORK+20: ABCD

Эта команда отображает значение ячейки WORK+20 как строку кодов КОИ-8.

2. DBG>EXAMINE/WORD MAIN

MAIN\MAIN: 483c

Эта команда выдает значение ячейки MAIN как целое число 483c длиной в слово.

3. DBG>EXAMINE/I MAIN+2

MAIN\MAIN+02: MOVAL L^MAINA,R11

Эта команда выдает инструкцию MOVAL из ячейки MAIN+2.

4. DBG>EXAMINE/DEC/WORD WORK:WORK+6

DETAT\WORKDATA: 256

DETAT\WORKDATA+2: 770

DETAT\WORKDATA+4: 1284

```
DETAT\WORKDATA+6: 1798
```

Этот пример показывает возможность отображения ряда значений от адресного выражения WORK до адресного выражения WORK+6.

```
5. DBG>EXAMINE/NOSYM WORKDATA
00000B6F:  03020100
```

Эта команда выдает значение адреса WORKDATA(0000086F) как адрес виртуальной памяти (03020100).

```
6. DBG>EXAMINE/HEX FIDBLK
FDEX1\MAIN\FIDBLK
    (1):        00000008
    (2):        00000100
    (3):        000000AB
```

Эта команда выдает значение адреса FIDBLK с шестнадцатеричным основанием.

1.35. Команда EXIT

Команда EXIT заканчивает сеанс отладки или выполнение команд командной процедуры или последовательности DO.

Формат:

EXIT

ПарАметры

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Когда пользователь выдает данную команду с терминала, происходит упорядоченное окончание сеанса отладки. При этом выполняется обработчик выхода отладчика, выдается на экран информация состояния выхода, а управление возвращается интерпретатору команд DCL. При этом нельзя продолжить отладку программы выдачей команд DEBUG или CONTINUE. Для возобновления работы отладчика следует снова запустить программу.

Когда команда EXIT выполняется в командной процедуре, управление возвращается в точку, из которой была вызвана эта командная процедура. Например, если эта командная процедура была вызвана из последовательности DO, управление возвращается этой последовательности, после чего отладчик выполняет следующую команду (если в этой последовательности еще остались команды).

Когда отладчик выполняет команду EXIT в последовательности DO, он игнорирует все оставшиеся в этой последовательности команды и воспроизводит свой запрос.

Пример.

```
DBG>EXIT
$
```

Эта команда заканчивает процесс отладки и возвращает пользователя на уровень команд DCL.

1.36. Команда EXITLOOP

Команда EXITLOOP завершает цикл WHILE, REPEAT или FOR.

Формат:

```
EXITLOOP [N]
```

Параметр:

N - определяет количество циклов, после которого необходимо осуществить выход. Параметр является целым числом.

Квалификаторы:

Отсутствуют.

Примеры:

```
1. DBG>WHILE X1 .LT. X2 DO(EXAM X2)
DISTANCE#MAIN\X2: 2.000000
DISTANCE#MAIN\X2: 2.000000
DISTANCE#MAIN\X2: 2.000000
```

Этот пример показывает команду WHILE, которая генерирует бесконечный цикл.

```
2. DBG>WHILE X1 .LT. X2 DO(EXAM X2;EXITLOOP)
DISTANCE#MAIN\X2: 2.000000
```

Этот пример показывает, как команда EXITLOOP заканчивает бесконечный цикл, сгенерированный командой WHILE.

1.37. Команда FOR

Команда FOR позволяет неоднократно выполнять список команд отладчика определенное количество раз.

Формат:

FOR имя=выражение1 TO выражение2 [BY выражение3]-
DO(список_команд)
DTES 3

Параметры:

Имя - указывает переменную.

Выражение1 - определяет значение типа целого числа или перечисления. Параметры выражение1 и выражение2 должны быть всегда одного и того же типа.

Выражение2 - определяет значение типа целого числа или перечисления.

Выражение3 - обозначает целое число.

Список_команд - указывает список команд отладчика.

Квалификаторы:

Отсутствуют.

Описание.

Поведение команды FOR меняется при различных значениях параметра выражения3. Если выражение3 положительно, переменная имя увеличивается от значения выражение1 на значение выражение3 до тех пор, пока оно не станет больше значения выражение2. Если выражение3 отрицательно, то переменная имя уменьшается от значения выражение1 на величину выражение3 до тех пор, пока не станет меньше выражение2.

Если выражение3 равно нулю, отладчик возвращает сооб-

дение об ошибке.

Если выражение3 полностью отсутствует, отладчик предполагает, что его значение равно +1.

Примеры:

1. DBG>FOR I=10 TO 1 BY -1 DO (EXAMINE A(I))

В этом примере массив просматривается в обратном порядке.

2. DBG>FOR I=1 TO 10 DO (DEPOSIT A(I)=0)

В этом примере массив А обнуляется.

1.38. Команда GO

Команда GO начинает или продолжает выполнение программы.

Формат:

GO [адресное_выражение]

Параметр:

Адресное_выражение - указывает ячейку, с которой возобновляется выполнение программы.

Квалификаторы:

Отсутствуют.

Описание.

Если команда GO определяется без адресного выражения, выполнение возобновляется с точки приостанова или, в случае запуска отладчика, с адреса передачи. Применение адресного выражения в качестве параметра может дать непредсказуемые результаты, если адресное выражение указывает ячейку программы, которая еще не выполнялась.

Примеры:

1. DBG>GO

ROUTINE START AT MAIN\MAIN

ROUTINE BREAK AT SUB1\SUB1

Эта команда начинает выполнение программы с адреса MAIN\MAIN. (адрес передачи является ячейкой скомпиллированной программы, определенной ключевыми словами исходной программы и указывающей начало программы).

2. DBG>GO MAIN

ROUTINE START AT MAIN\MAIN

ROUTINE BREAK AT SUB1\SUB1

Эта команда продолжает выполнение программы с адреса MAIN.

3. DBG GO

ROUTINE START AT SUB1\SUB1

%DEBUG-I-EXITSTATUS, IS %SYSTEM-S-NORMAL,NORMAL/SUCCESSFUL/
COMPLETION

Эта команда начинает выполнение программы с адресА передачи SUB1\SUB1 и успешно ее завершает.

1.39. Команда HELP

Команда HELP обеспечивает воспроизведение следующей информации относительно команд отладчика: описание команды, формат команды, а также параметры и квалификаторы, которые могут быть определены с этой командой.

Формат:

HELP тема[подтема...] .

Параметры:

Тема - определяет имя команды, о которой требуется получить информацию.

Подтема - определяет конкретный квалификатор или параметр, о котором пользователь хочет получить дополнительную информацию, или ключевое слово команды, которое дает информацию о ряде квалификаторов или параметров или о тех и других вместе.

Квалификаторы:

Отсутствуют.

Описание.

Если требуется информация относительно конкретного квалификатора или параметра, то следует определить его в качестве подтемы.

Если требуется информация относительно всех квалификаторов команды, то в качестве подтемы следует указать "QUALIFIER". Если требуется информация относительно всех параметров, то следует в качестве подтемы указать "PARAMETER". Если требуется информация относительно всех параметров и всех квалификаторов, то следует в качестве подтемы указать звездочку.

Пример.

```
DBG>HELP DEFINE
   DEFINE

            определяет одно или несколько символических
            имен и присваивает им указанные адреса на
            время соответствующего сеанса отладки
```

Формат:

DEFINE символическое_имя=выражение [,символическое_имя=выражение...]

Дополнительная информация:

Параметры:

Эта команда выдает информацию о команде DEFINE.

1.40. Команда IF

Команда IF позволяет выполнять список команд отладчика при условии, что языковое выражение имеет значение "истина"(TRUE).

Формат:

IF логическое_выражение THEN (список_команд)-
[ELSE(список_команд)]

Параметры:

Логическое_выражение - определяет языковое выражение, которое вычисляется как булевское значение: истина или ложь (TRUE или FALSE).

Список_команд - указывает список команд отладчика. Команды должны разделяться точкой с запятой.

Квалификаторы:

Отсутствуют.

Описание.

Команда IF оценивает языковое выражение как булевское. Если его значение - "истина" (TRUE) для текущего языка, выполняется список команд отладчика в предложении THEN. Если значение - "ложь" (FALSE), выполняется список команд в

предложении ELSE (если оно имеется).

Пример.

```
DBG>SET BREAK R DO (IF X .LT. 5 THEN (GO) ELSE (EXAMINE X))
```

Эта команда приказывает отладчику приостановить выполнение программы в ячейке R и возобновить выполнение программы, если значение X меньше 5. Если значение X равно 5 или более, отладчик выдаст это значение.

1.41. Команда QUIT

Команда QUIT заканчивает процесс отладки, если она не выдана из командной процедуры или последовательности команд DO. Не выполняет никаких заявленных обработчиков выхода.

Формат:

```
QUIT
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Выдавая команду QUIT после запроса DBG>, пользователь осуществляет окончание процесса отладки: файлы регистрации закрываются, экран и малая клавиатура переводятся в свои начальные состояния, а управление передается к интерпретатору команд DCL. Выполняется обработчик выхода отладчика, но не выполняются заявленные пользователем обработчики выхода (при выдаче команды EXIT выполняются заявленные обработчики выхода). После этой команды нельзя продолжить

отладку программы выдачей команд DEBUG или CONTINUE. Для повторного запуска отладчика надо вновь запустить программу.

Когда отладчик выполняет команду QUIT в последовательности DO или командой процедуре, он прекращает их выполнение. В этом случае отладчик не заканчивает процесс отладки.

Пример.

```
DBG>QUIT
¤
```

Эта команда, выданная после запроса DBG>, заканчивает процесс отладки и возвращает пользователя на уровень команд DCL.

1.42. Команда REPEAT

Команда REPEAT предлагает способ многократного выполнения последовательности команд отладчика.

Формат:

```
REPEAT языковое_выражение DO (список_команд)
```

Параметры:

Языковое выражение - указывает выражение на текущем языке программирования, которое вычисляется как целое положительное число.

Список_команд - указывает последовательность команд отладчика, разделенных точками с запятой.

Описание.

Команда REPEAT позволяет повторно выполнять команду или список команд. Она подобна команде FOR.

Пример.

DBG>REPEAT 10 DO (STEP)

Эта команда заставляет отладчик выполнять команду STEP 10 раз.

1.43. Команда SAVE

Команда SAVE запоминает содержимое существующего экранного кадра в новом кадре.

Формат:

SAVE старый_кадр AS новый_кадр [,...]

Параметры:

Старый_кадр - указывает кадр, содержимое которого необходимо сохранить.

Новый_кадр - указывает имя нового создаваемого кадра, в который затем помещается содержимое старого кадра.

Квалификаторы:

Отсутствуют.

Описание.

Команда SAVE позволяет сделать "моментальный снимок" имеющегося кадра в новый кадр для дальнейшего использования. Новый кадр создается с тем же текстовым содержимым, что и в имеющемся кадре. Новому кадру придаются все свойства или характеристики старого, за исключением того, что новый кадр является удаленным с экрана и никогда не корректируется автоматически. Новый кадр можно вызвать на экран терминала с помощью команды DISPLAY.

Пример.

```
DBG>SAVE REG AS OLDREG
```

Эта команда перемещает содержимое кадра REG в заново созданный кадр OLDREG.

1.44. Команда SCROLL

Команда SCROLL передвигает экранное окно в определенном направлении над текстом кадра, делая видимым дополнительное его содержимое.

Формат:

```
SCROLL [имя_кадра]
```

Параметр:

Имя_кадра - указывает предназначенный для прокрутки кадр. Если опустить этот параметр, установленный командой SELECT, то будет сдвигаться текущий прокручиваемый кадр.

Квалификаторы:

/BOTTOM - прокрутка производится в самую нижнюю часть текста кадра.

/DOWN:[N] - прокрутка выполняется вниз по тексту кадра на N строк для того, чтобы сделать видимым текст, расположенный ниже текущего кадра. Если N не задано, окно сдвигается приблизительно на 3/4 своей высоты.

/LEFT:[N] - сдвигает окно влево над текущим прокручиваемым или указанным пользователем кадром. Если значение N не указано, происходит сдвиг вправо на 8 позиций. Сдвигать левее первой позиции невозможно.

/RIGHT:[N] - сдвигает окно вправо над текущим прокручиваемым или указанным пользователем кадром. Если значение

N не указано, происходит сдвиг вправо на 8 позиций.

/TOP - устанавливает окно над верхней частью текста кадра.

/UP[:N] - сдвигает вверх по тексту кадра на N строк. Если значение N не определено, окно сдвигается вверх примерно на 3/4 высоты окна.

Описание.

До тех пор, пока явно не указано имя кадра, который необходимо прокрутить, прокручивается текущий прокручиваемый кадр, заданный ранее в команде SELECT.

Примеры:

1. DBG>SCROLL/LEFT

Эта команда сдвигает окно влево на 8 позиций.

2. DBG>SCROLL/UP:4

Эта команда сдвигает окно вверх по тексту на 4 строки.

1.45. Команда SEARCH

Команда SEARCH направляет отладчик на поиск исходного текста, содержащего указанную цепочку символов, и последующее его отображение.

Для срабатывания команды SEARCH необходимо скомпиллировать программу с отладчиком.

Формат:

SEARCH диапазон цепочка

Параметры:

Диапазон - указывает район программы, в котором должен выполняться поиск. Можно выполнять поиск в любом модуле

программы от начала или с указанного номера строки до конца или до указанного номера строки программы. Диапазон ограничивает поиск отладчиком появлений цепочки определенными районами программы. Эти районы программы могут быть определены в любом из следующих форматов:

- имя_модуля - указывает поиск в определенном модуле от строки с номером 0 до конца этого модуля;

- имя_модуля\номер_строки - указывает поиск в определенном модуле от указанного номера строки до конца модуля;

- имя_модуля\номер_стоки:номер_строки - указывает поиск в определенном модуле, начиная от указанного с левой стороны двоеточия номера строки до конца строки, номер которой указан с правой стороны от двоеточия;

- номер_строки - указывает поиск в модуле, определяемом текущей установкой диапазона, от указанного номера строки до конца этого модуля;

- номер_строки:номер_строки - указывает поиск в модуле, определенном текущей установкой диапазона, начиная с номера строки, указанного с левой стороны от двоеточия, и кончая номером строки, указанным с правой стороны от двоеточия;

- нуль (т.е. отсутствие записи параметра) указывает поиск в том же самом модуле, из которого последний раз воспроизводилась исходная строка (например, в результате команды TYPE, EXAMINE/SOURCE или SEARCH), начиная с первой строки после последней воспроизводимой строки и до конца модуля.

Цепочка - определяет симзолы исходного кода, поиск которых выполняется. Если данный параметр не указан, то отладчик использует последнюю указанную искомую цепочку

поиска, т.е. параметр цепочки, определенный в последней команде SEARCH. Если такой цепочки поиска не было, то выдается сообщение об ошибке.

Параметр цепочки может быть заключен в кавычки или в апострофы, но также может указываться без разделителей.

Если цепочка выделена кавычками или апострофами, то в ней могут содержаться пробелы и символы табуляции, а также любые алфавитно-цифровые или специальные символы.

Если цепочка заключена в кавычки, то содержащиеся в ней кавычки указываются двойными кавычками. Если цепочка заключена в апострофы, то содержащиеся в ней апострофы указываются двойными апострофами.

Однако, если цепочка не выделена этими символами, то имеют место ограничения:

- цепочка не должна иметь начальных и конечных пробелов или символов табуляции;

- цепочка не должна иметь вложенной точки с запятой;

- параметр диапазона не должен быть нулем, т.е. диапазон должен быть указан в явной форме.

Квалификаторы:

/ALL - определяет, что отладчик должен выполнять поиск всех местоположений цепочки в указанном диапазоне и воспроизводить каждую строку, содержащую эту цепочку.

/IDENTIFIER - определяет, что отладчик должен выполнять поиск местоположения цепочки в указанном диапазоне, но воспроизводить эту цепочку только в том случае, если она не ограничена с любой стороны символом, который может быть частью идентификатора на данном языке программирования.

/NEXT - определяет, что отладчик должен выполнять

поиск первого местоположения цепочки в указанном диапазоне и воспроизводить только ту строку, которая содержит это местоположение. Действует по умолчанию.

/STRING - определяет, что отладчик должен выполнять поиск и воспроизведение цепочки в том виде, как она указана, а не интерпретировать контекст, окружающий эту цепочку, как в случае квалификатора /IDENTIFIER. Действует по умолчанию.

Описание.

Квалификаторы данной команды определяют, ищет ли отладчик все местоположения данной цепочки (/ALL) или только ее следующее появление (/NEXT), а также воспроизводит ли отладчик любое местоположение этой цепочки (/STRING) или только те, в которых она не ограничена с любой стороны символом, который может быть частью идентификатора на текущем языке (/IDENTIFIER).

Примеры:

```
1. DBG>SEARCH/STRING/ALL 40:50 D
MODULE COBOLTEST
     40:02   D2N   COMP-2 VALUE -234560000000.
     41:02   D     COMP-2 VALUE  222222.33.
     42:02   DN    COMP-2 VALUE -222222.333333.
     47:02   DR0   COMP-2 VALUE  0.1.
     48:02   DR5   COMP-2 VALUE  0.000001.
     49:02   DR10  COMP-2 VALUE  0.00000000001.
     50:02   DR15  COMP-2 VALUE  0.0000000000000001.
```

Эта команда отыскивает все местонахождения буквы D в строках от 40 до 50 модуля COBOLTEST.

```
2. DBG>SEARCH/IDENTIFIER/ALL 40:50 D
MODULE COBOLTEST
    41:02   D       COMP-2 VALUE  222222.33.
```

Эта команда отыскивает все местонахождения буквы D в строках от 40 до 50 модуля COBOLTEST. Отладчик выдает только ту строку, где буква D не ограничена с обеих сторон знаком, который может быть частью идентификатора в текущем языке.

```
3. DBG>SEARCH/NEXT 40:50 D
MODULE COBOLTEST
   40:02   D2N    COMP-2 VALUE -234560000000.
```

Эта команда отыскивает следующее появление буквы D в строках от 40 до 50 модуля COBOLTEST.

```
4. DBG>SEARCH/NEXT
MODULE COBOLTEST
    41:02   D       COMP-2 VALUE  222222.33.
```

Эта команда отыскивает следующее появления буквы D. Отладчик предполагает, что искомой строкой является D, потому что D была последней введенной искомой строкой, а больше ни одна строка указана не была.

1.46. Команда SELECT

Команда SELECT назначает указанный экранный кадр в качестве текущего кадра инструкций, вывода, прокрутки или текущего кадра исходного текста.

Формат:

```
SELECT [имя_кадра]
```

Параметры:

Имя_кадра - указывает имя устанавливаемого кадра. Можно задать любое из следующих:

1) предопределенные отладчиком: SRC, OUT, INST, REG;

2) предварительно созданные командой SET DISPLAY;

3) псевдо-имена: %CURDISP, %NEXTDISP, %NEXTINST, %NEXTOUTPUT, %NEXTSCROLL, %NEXTSOURCE.

Если этот параметр опущен, происходит отмена действующей установки:

- если указывается команда без квалификатора (или квалификатор /SCROLL), то снимается текущий прокручиваемый кадр;

- если указывается квалификатор /INSTRUCTION, то снимается текущий кадр команд ассемблера;

- если указывается квалификатор /OUTPUT, то снимается текущий кадр выходной информации;

- если указывается квалификатор /SOURCE, то снимается текущий кадр исходного текста.

Квалификаторы:

/INSTRUCTION - устанавливает указанный кадр в качестве текущего кадра команд ассемблера. Это означает, что выходная информация всех команд EXAMINE/INSTRUCTION направляется в этот кадр. Указываемый кадр должен быть кадром команд ассемблера. Если не указывается имя кадра, то ранее установленный текущий кадр команд ассемблера становится неустановленным (снимается): ни один кадр не получает информацию по командам ассемблера.

/OUTPUT - устанавливает указанный кадр в качестве текущего кадра выходной информации. Это означает, что вся

обычная выходная информация отладчика помещается в этот кадр. Указываемый кадр не может быть кадром исходного текста или кадром команд ассемблера.

/SCROLL - устанавливает указанный кадр в качестве текущего прокручиваемого кадра. Это значит, что команда SCROLL по умолчанию будет прокручивать этот кадр, пока явно не будет задан другой кадр. Если квалификаторы не указываются, то по умолчанию принимается /SCROLL. Если не указывается имя кадра, то ранее установленный текущий прокручиваемый кадр снимается: ни один кадр не прокручивается.

/SOURCE - устанавливает указанный кадр в качестве текущего кадра исходного текста. Это означает, что выход всех команд TYPE и EXAMINE/SOURCE направляется в этот кадр. Указываемый кадр должен быть кадром исходного текста. Если не задается имя кадра, то ранее установленный текущий кадр исходного текста снимается: ни один кадр не получает исходный текст.

Описание.

Для направления информации типа команд ассемблера, исходного текста или обычной выходной информации используется команда SELECT с соответствующими квалификаторами /INSTRUCTION, /SOURCE или /OUTPUT. Для установки заданного кадра как прокручиваемого используется команда SELECT без квалификаторов или с квалификатором /SCROLL.

Для снятия установленного кадра используется команда SELECT без параметра. Неустановленный (снятый) кадр больше не помечен как кадр команд ассемблера, исходного текста, выходной информации или как прокручиваемый кадр. Для повторного направления выходной информации или установки прок-

ручиваемого кадра нужно указать другую команду SELECT.

Примеры:

1. DBG>SELECT/SOURCE/SCROLL SRC2

Команда SELECT/SOURCE/SCROLL назначает кадр SRC2 текущим прокручиваемым кадром исходного текста.

2. DBG SELECT/SOURCE

Команда SELECT/SOURCE снимает (удаляет атрибут типа кадра) ранее назначенный текущий кадр исходного текста. Выход команды TYPE теперь направляется в ранее назначенный текущий кадр выходной информации.

1.47. Команда SET

Команда SET определяет или изменяет различные функции или характеристики (точки приостанова, режимы, типы и так далее) в соответствии с указанным ключевым словом.

Более подробная информация на эту тему приводится в описаниях отдельных команд данного класса (подразделы 1.49 - 1.73).

Формат:

SET ключевое_слово параметр

Параметры:

Ключевое_слово - определяет устанавливаемый элемент. Могут быть использованы следующие ключевые слова: BREAK, DEFINE, EXCEPTION BREAK, DISPLAY, LANGUAGE, LOG, MARGIN, MAX_SOURCE_FILES, MODE, MODULE, OUTPUT, KEY, PROMPT, RADIX, SCOPE, SEARCH, SOURCE, STEP, TERMINAL, TRACE, TYPE, WATCH или WINDOW.

Параметр - зависит от указанного ключевого слова.

Квалификаторы:

Зависят от указанного ключевого слова.

1.48. Команда SET BREAK

Команда SET BREAK обеспечивает установку точки приостанова в ячейке, определяемой адресным выражением.

Формат:

SET BREAK [адресное_выражение[,...]]-[WHEN(выражение)]-
[DO(команда[;...])] .

Параметры:

Адресное_выражение - определяет ячейку, в которой должна быть установлена точка приостанова. Ее значение зависит от выбранных квалификаторов. Например, если квалификаторы не указаны, то адресное_выражение относится к адресу определенной программной команды и при каждом выполнении данной команды отладчик приостанавливает выполнение. Для получения дополнительной информации по интерпретации параметра адресного выражения см. Описание квалификаторов.

Команда - это любая команда отладчика, которую отладчик должен выполнять в качестве части последовательности команд DO при достижении точки приостанова. Если пользователь указывает несколько команд, то следует разделить их точкой с запятой.

Выражение - указывает какое-либо выражение на текущем языке программирования, которое необходимо оценивать каждый раз при достижении точки приостанова. Если выражение имеет значение "истинна" (TRUE), происходит приостанов, и выдается соответствующее сообщение.

Однако, если выражение является ложным, приостанова не происходит. В этом случае сообщение не поступает, команды, указанные предложением DO, не выполняются, а выполнение программы продолжается.

Квалификаторы:

/AFTER:N - указывает, что действие приостанова должно выполняться с N-го появления точки приостанова. После этого простанов происходит каждый раз в этой точке при условии, что предложение WHEN имеет значение "истина". Величина N интерпретируется в режиме десятичного основания независимо от того, какие установки режима основания действуют в это время.

Другими словами, отладчик подсчитывает количество появлений данной точки, но не приостанавливает программу при первых N-1 активациях ячейки приостанова.

Команда SET BREAK/AFTER:1 имеет то же действие, что и команда SET BREAK.

/BRANCH - указывает, что отладчик должен приостанавливаться при каждой встречающейся команде ветвления (включая BEQL, BGTR, BLEQ, BGEQ, BLSS, BGTRU, BLEQU, BVC, BVS, BGEQU, BLSSU, BRB, BRW, JMP, BBS, BBC, BBSS, BBCS, BBSC, BBCC, BBSSI, BBCCI, BLBS, BLBC, ACBB, ACBW, ACBL, ACBF, ACBD, ACBG, ACBH, AOBLEQ, AOBLSS, SOBGEQ, SOBGTR, CASEB, CASEW, CASEL) во время выполнения. Параметр адресного выражения для этого квалификатора не имеет значения, так что его можно опустить.

/CALL - указывает, что отладчик должен приостанавливаться при каждой команде вызова (включая команды CALLS, CALLG, BSBW, BSBB, JSB, RSB и RET), встреченной при выпол-

нении. Параметр адресного выражения для этого квалификатора не имеет значения, так что его можно опустить.

/EXCEPTION - указывает, что действие приостанова должно происходить каждый раз при сигнализации исключительной ситуации. Приостанов происходит до вызова обработчика исключительной ситуации, написанного пользователем. Другими словами, команда SET BREAK/EXCEPTION идентична команде SET EXCEPTION BREAK. Для этого квалификатора параметр адресного выражения не имеет значения, так что он опускается.

/INSTRUCTION - указывает, что отладчик должен приостанавливаться на каждой выполненной команде. Для этого квалификатора адресное выражение не имеет значения, так что он опускается.

/INSTRUCTION =(код_операции[,...]) - указывает, что приостанов должен осуществляться на каждой команде, чей код операции дается одной из перечисленных мнемоник. Можно задать множество мнемоник кодов операции, разделив их запятыми и поместив весь список в круглые скобки. Квалификатор адресное выражение не имеет значения, так что его можно опустить.

/LINE - указывает на приостанов в начале каждой новой строки программы. Для этого квалификатора адресное выражение значения не имеет, так что он опускается.

/MODIFY - указывает, что приостанов должен происходить на каждой команде, которая записывает в ячейку, определяемую адресным выражением, или модифицирует значение такой ячейки.

Команда SET BREAK/MODIFY действует точно так же, как команда SET WATCH, и срабатывает при тех же ограничениях (подпункт 3.2.3.1 2б.00152-01 34 07-1. Если в параметре

адресного выражения указывается абсолютный адрес, отладчик не может связать этот адрес с конкретным об'ектом данных. В этом случае отладчик использует длину по умолчанию в 4 байта. Однако эту длину можно изменить установкой типа или на WORD (которое изменяет длину по умолчанию до 2 байтов) или на BYTE (длина уменьшается до 1 байта).

/RETURN - указывает, что приостанов должен происходить на команде RETURN (RET) - выхода из подпрограммы. Этот квалификатор может применяться только к подпрограммам, вызываемым командами CALLS или CALLG; к подпрограммам, вызываемым по JSB, он не применяется. Приостанов на команде RET позволяет проверить локальную обстановку до того, как команда RET удалит кадр вызова данной подпрограммы из стека вызовов.

Для этого квалификатора параметром адресного выражения является адрес команды внутри подпрограммы, вызываемой с помощью CALLS или CALLG. Он может просто быть именем подпрограммы; в этом случае он указывает начальный адрес подпрограммы. Однако можно также указать любую другую ячейку в подпрограмме, но тогда пользователь сможет зафиксировать только те возвраты, когда выполнение подпрограммы проходит через указанную ячейку.

/[NO]SILENT - управляет выдачей сообщения на терминал при выполнении приостанова. /NOSILENT указывает на выдачу сообщения о приостанове (действует по умолчанию). /SILENT указывает, что сообщение выдавать не надо.

/[NO]SOURCE - управляет выдачей исходного текста в случае приостанова. /SOURCE указывает на необходимость выдачи исходного текста (действует по умолчанию). /NOSOURCE указывает, что исходный текст в точке приостанова не нужен.

/SILENT отменяет /SOURCE.

/TEMPORARY - указывает, что точку приостанова после ее активизации следует убрать. Другими словами, точка приостанова работает один раз, после чего она отменяется.

Описание.

При активизации точки приостанова отладчик выполняет следующие действия:

1) приостанавливает выполнение программы;

2) оценивает выражение WHEN (если оно было указано при установке точки приостанова): если его значение "ложь", выполнение программы продолжается, а следующие три действия пропускаются;

3) воспроизводит ячейку точки приостанова (и воспроизводит строку исходного кода, соответствующую программной команде в этой ячейке, если действует параметр SOURCE в результате выполнения предыдущей команды SET STEP SOURCE);

4) выполняет командную последовательность DO (если такая последовательность была указана при установке соответствующей точки приостанова);

5) выдает запрос DBG>.

Примеры:

1. DBG>SET BREAK/AFTER:3 SUB2

Эта команда заставляет отладчик приостановить выполнение программы, когда он встречает ячейку SUB2 в третий раз.

2. DBG>SET BREAK LOOP 1 DO (EXAMINE D;STEP;EXAMINE Y;GO)

Эта команда заставляет отладчик приостановить выполнение программы в ячейке LOOP 1 и выполнить следующие команды:

```
EXAMINE D
STEP
EXAMINE Y
GO
```

```
3. DBG>SET BREAK/ TEMPORARY 1440
DBG>SHOW BREAK
BREAKPOINT AT 1440 [TEMPORARY]
```

Эта команда заставляет отладчик приостановить выполнение программы в ячейке 1440. После активизации точки приостанова в ячейке эта точка ликвидируется.

1.49. Команда SET DEFINE

Команда SET DEFINE определяет, как интерпретировать команды DEFINE. Команда DEFINE может связывать символ с адресом, с командной строкой или со значением.

Формат:

```
SET DEFINE тип_имени
```

Параметры:

Тип_имени может быть следующим:

1) ADDRESS - последующие команды DEFINE воспринимаются как DEFINE/ADDRESS. Принимается по умолчанию;

2) COMMAND - последующие команды DEFINE воспринимаются как DEFINE/COMMAND;

3) VALUE - последующие команды DEFINE воспринимаются как DEFINE/VALUE.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SET DEFINE VALUE
```

Эта команда указывает, что последующие команды DEFINE интерпретируются как DEFINE/VALUE.

1.50. Команда SET DISPLAY

Команда SET DISPLAY определяет появление одного или более указываемых кадров на экране терминала.

Формат:

```
SET DISPLAY имя_кадра [AT спецификация_окна]-
[тип_окна][,...]
```

Параметры:

Имя_кадра - указывает имя определяемого кадра. Если уже существует кадр с таким именем, его надо уничтожить до определения нового.

Можно указывать несколько кадров, каждый со своими необязательными параметрами спецификация_окна и тип_кадра.

Спецификация_окна - определяет экранное окно, в которое должен отображаться кадр. Можно указать следующее:

- предопределенное отладчиком окно. Например, FS (весь экран), H1 (верхняя половина), Q3 (третья четверть), T3 (вторая треть), Q23 (средние две четверти);

- окно, предварительно определенное командой SET WINDOW;

- спецификацию окна в форме начальная строка, количество строк. Например, (12,4) обозначает окно, начинающееся со строки 12 и размером в 4 строки.

Если этот параметр опускается, кадр помещается в окне

H1 или H2 по умолчанию.

Тип_кадра - указывает тип нового кадра, если его надо изменить. Ключевые слова для этого параметра включают:

1) DO(список_команд) - задает автоматически корректируемый кадр выходной информации. Список_команд выполняется каждый раз, когда управление переходит к отладчику. Их выход формирует содержимое данного кадра;

2) INSTRUCTION - задает кадр команд ассемблера, который автоматически не корректируется. Этот кадр содержит выход команд EXAMINE/INSTRUCTION. В этот кадр ничего не записывается до тех пор, пока он не устанавливается как текущий кадр команд ассемблера с помощью команды SELECT/INSTRUCTION;

3) INSTRUCTION(список_команд) - задает автоматически корректируемый кадр команд ассемблера. В списке команд необходимо указать команду EXAMINE/INSTRUCTION. Кадр корректируется каждый раз, когда управление переходит к отладчику;

4) NORMAL - задает кадр обычной выходной информации. В него ничего не записывается до тех пор, пока он не установлен как текущий кадр выходной информации с помощью команды SELECT/OUTPUT;

5) REGISTER - задает автоматически корректируемый кадр содержимого регистров. Кадр корректируется каждый раз, когда управление переходит к отладчику;

6) SOURCE - задает автоматически некорректируемый кадр исходного текста, который отображает выход команд TYPE или EXAMINE/SOURCE. В этот кадр ничего не записывается до тех пор, пока он не устанавливается как текущий кадр исходного текста с помощью команды SELECT/SOURCE;

7) SOURCE(список_команд) - задает автоматически корректируемый кадр исходного текста. В списке команд должна быть указана команда TYPE или EXAMINE/SOURCE. Кадр корректируется каждый раз, когда управление переходит к отладчику.

Если параметр тип_кадра опускается, создается кадр типа NORMAL.

Квалификаторы:

/HIDE - маскирует ("прячет") указанный экранный кадр под другими кадрами, которые занимают ту же область на экране. При этом другие кадры, ранее скрытые данным кадром, становятся видимыми.

/MARK_CHANGE - отмечает строки, которые изменяются на кадрах типа DO(список_команд) каждый раз при выполнении корректировки кадра. Строки, которые изменились со времени последней корректировки, высвечиваются в негативном (обратном) изображении. Этот квалификатор особенно удобен, когда необходимо выделить какие-либо переменные на автоматически корректируемом кадре при их изменении. Этот квалификатор не применяется к другим типам кадров.

/REMOVE - указывает, что кадр не должен отображаться на экране до тех пор, пока он не будет специально запрошен командой DISPLAY. Кадр помечается как удаленный с экрана, хотя продолжает существовать.

/SIZE:N - устанавливает максимальный размер текста кадра равным N строк. Если в кадр записывается строк больше N, наиболее ранние строки теряются и на них записываются новые. Если этот квалификатор опускается, размер кадра по умолчанию составляет 50 строк. Для кадра исходного текста N

дает число исходных строк, которое можно поместить в буфер памяти за один раз. Однако кадр исходного текста можно прокрутить над всем исходным файлом.

Примеры:

1. DBG>SET DISPLAY DISP2 AT T3

DBG>SELECT/OUTPUT DISP2

Команда SET DISPLAY создает новый кадр под именем DISP2 в нижней трети экрана. Затем команда SELECT/OUTPUT выбирает DISP2 в качестве текущего кадра выходной информации.

2. DBG>SET WINDOW TOP AT(1,8)

DBG>SET DISPLAY NEWINST AT TOP INSTRUCTION

DBG>DISPLAY/INST NEWINST

Команда SET WINDOW создает окно под именем TOP размером в 8 строк, начиная со строки 1. Команда SET DISPLAY создает кадр команд ассемблера под именем INST для отображения в окне TOP. Команда DISPLAY/INST выбирает NEWINST в качестве текущего кадра команд ассемблера.

1.51. Команда SET EXCEPTION BREAK

Команда SET EXCEPTION BREAK определяет рассмотрение отладчиком исключительного условия, генерируемого программой пользователя, в качестве точки приостанова.

Формат:

SET EXCEPTION BREAK

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание:

При выдаче этой команды, когда бы программа не генерировала условие исключительной ситуации, отладчик отвечает приостановом выполнения программы, сообщая об условии исключительной ситуации и запрашивая ввод.

Следовательно, при активизации точки приостанова по исключительной ситуации есть возможность выдать команды отладчика. Если необходимо продолжить выполнение программы, можно выдать одну из следующих команд GO:

1) команду GO без параметра адресное_выражение. В этом случае отладчик вновь сигнализирует об исключительной ситуации, позволяя, таким образом, срабатывать программам обработки исключительной ситуации, заявленным пользователем;

2) команду GO с параметром адресное_выражение. В этом случае отладчик позволяет продолжить выполнение программы в указанной ячейке, задерживая выполнение каких-либо обработчиков исключительной ситуации, заявленных пользователем.

Необходимо отметить, что при активизации точки приостанова по исключительной ситуации нельзя продолжить выполнение программы с помощью команды STEP. Отладчик выдает предупреждающее сообщение, потому что команда STEP является в этом контексте незаконной.

Команда SET EXCEPTION BREAK обладает таким же действием, что и команда SET BREAK/EXCEPTION.

Пример.

```
DBG>SET EXCEPTION BREAK
```

Эта команда обязывает отладчик рассматривать условие исключительной ситуации как точку приостанова, что позволяет пользователю вводить команды отладчика.

1.52. Команда SET KEY

Команда SET KEY изменяет текущее состояние определения клавиши. Функции клавиши определяются с помощью команды DEFINE/KEY.

Формат:

```
SET KEY
```

Параметры:

Отсутствуют

Квалификаторы:

/[NO]LOG - управляет выдачей отладчиком сообщения, указывающего, что состояние клавиши было установлено. По умолчанию работает /LOG.

/[NO]STATE[=имя_состояния] - указывает состояние клавиши, которое должен установить отладчик. Именем состояния может быть любая буквенно-цифровая строка. Если опускается квалификатор /STATE или используется /NOSTATE, текущее состояние остается неизменным. Состоянием по умолчанию является DEFAULT.

Описание.

При определении клавиш малой клавиатуры с помощью команды DEFINE/KEY определению клавиши можно присваивать

определенное имя состояния. В дальнейшем перед нажатием этой клавиши необходимо установить соответствующее состояние, иначе определение данной клавиши не обрабатывается. Для установки необходимого состояния и служит команда SET KEY.

Пример.

```
DBG>SET KEY/STATE=PROG3
```

Команда SET KEY изменяет состояние клавиши на состояние PROG3. Теперь можно использовать определения клавиши, связанные с этим состоянием.

1.53. Команда SET LANGUAGE

Команда SET LANGUAGE обеспечивает установку текущего языка программирования.

Формат:

```
SET LANGUAGE имя_языка
```

Параметры:

Имя_языка - определяет имя языка программирования. Могут быть использованы следующие языки: Бэйсик, Блисс, Си, Кобол, Фортран, Паскаль, Пл/1 и Макро.

Квалификаторы:

Отсутствуют.

Описание.

При первоначальном запуске отладчика в качестве текущего языка программирования устанавливается тот, на котором написан модуль, содержащий адрес передачи. Пользователь может (и должен) изменить этот язык с помощью команды SET

LANGUAGE перед отладкой модуля, написанного на другом исходном языке.

Текущий язык программирования оказывает воздействие на параметры отладчика по умолчанию (т.е. На параметры типа, основания и величины шага). Кроме того, языки программирования отличаются такими элементами, как преобразование типа, оценка выражений, допустимый синтаксис, специальные символы и символические имена.

Примеры:

1. DBG>SET LANGUAGE COBOL

Эта команда устанавливает в качестве текущего язык кобол.

2. DBG>SET LANGUAGE PASCAL

Эта команда устанавливает в качестве текущего язык Паскаль.

1.54. Команда SET LOG

Команда SET LOG определяет имя файла регистрации, в который отладчик выполняет запись при установке параметра вывода на LOG.

Формат:

SET LOG спецификация_файла

Параметры:

Спецификация_файла - это спецификация файла регистрации. Если спецификация файла начинается со специального символического имени, например такого, как квадратная скобка, то пользователь должен заключить спецификацию файла в кавычки или апострофы. В противном случае заключать специ-

фикацию файла в эти разделители не требуется.

Квалификаторы:

Отсутствуют.

Описание.

Если параметр вывода установлен на LOG, но с помощью данной команды никакой файл регистрации не определен, то по умолчанию отладчик записывает свой вывод в файл DEBUG.LOG.

Если отладчик выполняет запись в файл регистрации, а пользователь определяет с помощью данной команды другой файл регистрации, то отладчик закрывает старый файл и начинает записывать в файл, определенный командой SET LOG.

Если пользователь определяет имя файла, но не указывает тип файла в этой команде, то по умолчанию отладчик предполагает тип файла LOG.

Если пользователь указывает номер версии в параметре спецификации файла данной команды, а этот номер версии файла уже существует, то отладчик не может открыть новый файл. Вместо этого отладчик выполняет запись в указанный файл, добавляя регистрацию сеанса отладки в конце данного файла.

Следует отметить, что команда SET LOG определяет только имя файла регистрации. Она не заставляет отладчик создавать такой файл или записывать в него. Эти функции выполняет команда SET OUTPUT LOG.

Примеры:

1. DBG>SET OUTPUT LOG

SET LOG CALC

После выдачи команды SET OUTPUT LOG команда SET LOG обязывает отладчик помещать свой выход в файл под именем

```
CALC.
2. DBG>SET OUTPUT LOG
SET LOG "[CODEPROJ]FEB29.TMP"
```

После выдачи команды SET OUTPUT LOG, команда SET LOG обязывает отладчик помещать свой выход в файл со спецификацией [CODEPROJ]FEB29.TMP.

1.55. Команда SET MARGIN

Команда SET MARGIN определяет позицию самого левого символа исходной строки, с которой должно начинаться воспроизведение строки исходного кода (левая граница), и позицию самого правого символа, на которой заканчивается воспроизведение исходного кода (правая граница).

Формат:

```
SET MARGIN правая_граница
           левая_граница:правая_граница
           левая_граница:
           :правая_граница .
```

Параметры:

Левая_граница - определяет позицию символа в исходной строке, с которого должно начинаться воспроизведение исходного кода.

Правая_граница - определяет позицию символа исходной строки, на котором должно заканчиваться воспроизведение строки исходного кода.

Квалификаторы:

Отсутствуют.

Описание.

По умолчанию отладчик воспроизводит исходную строку, начиная с первой позиции исходной строки. Первая позиция исходной строки фактически является позицией 9 на экране терминала. Первые 8 позиций на экране терминала заняты номером строки и не могут изменяться с помощью данной команды.

Если пользователь определяет одно число, отладчик устанавливает левую границу на 1, а правую границу - в соответствии с этим числом.

Если пользователь указывает два числа, разделенных двоеточием, то отладчик устанавливает левую границу в соответствии с числом с левой стороны от двоеточия, а правую границу - в соответствии с числом с правой стороны от двоеточия.

Если пользователь указывает одно число, за которым приводится двоеточие, то отладчик устанавливает левую границу в соответствии с указанным числом, оставляя правую границу без изменения.

Если пользователь указывает двоеточие, за которым следует одно число, то отладчик устанавливает правую границу в соответствии с этим числом, оставляя левую границу без изменения.

Увеличение левой границы особенно полезно для устранения воспроизведения пустых мест с левой стороны. Тогда в строке терминала появляется больше места для воспроизведения исходного кода.

Уменьшение правой границы (по сравнению с ее установкой по умолчанию на 255) предотвращает продолжение длинных строк путем их усечения. Так как длинные строки исходного

кода требуют двух строк на экране терминала, то их усечение обеспечивает возможность воспроизведения большего числа строк исходного кода на терминале.

Команда SET MARGIN оказывает воздействие только на воспроизведение исходных строк, т.е. на воспроизведение результата выполнения таких команд, как TYPE и EXAMINE/SOURCE. Команда SET MARGIN не оказывает воздействия на воспроизведение результатов выполнения команд, которые не выдают исходного текста (например EXAMINE, EVALUATE, SHOW MODE и т.д.). Если команда воспроизводит исходный код вместе с другой информацией (например, команда STEP/SOURCE), то воспроизведение исходного кода подвергается влиянию текущей установки границ, а другая информация не подвергается влиянию этой команды.

Примеры:

```
1. DBG>SHOW MARGIN
LEFT MARGIN: 1, RIGHT MARGIN: 255
DBG>TYPE 14
MODULE FORARRAY
  14:         DIMENSION IARRAY(4:5,5),VECTOR(10),I3D(3,3,4)
```

Этот пример демонстрирует установку границ по умолчанию для строки исходного кода (1 и 255).

```
2. DBG>SET MARGIN 39
DBG>SHOW MARGIN
LEFT MARGIN: 1, RIGHT MARGIN: 39
DBG>TYPE 14
MODULE FORARRAY
   14:         DIMENSION IARRAY(4:5,5), VECTOR
```

Этот пример показывает, как изменяется строка исходно-

го кода при изменении установки правой границы с 255 на 39.

```
3. DBG>SET MARGIN 10:45
DBG>SHOW MARGIN
LEFT MARGIN: 10, RIGHT MARGIN: 45
DBG>TYPE 14
MODULE FORARRAY
   14:  IMENSION IARRAY(4:5,5),VECTOR(10),
```

На этом примере показано изображение одной и той же строки исходного кода после изменения обеих границ.

```
4. DBG>SET MARGIN: 100
DWG>SHOW MARGIN
LEFT MARGIN: 10, RIGHT MARGIN: 100
```

На этом примере показано, как изменить установку правой границы, сохраняя предыдущую установку левой границы.

```
5.. DWG>SET MARGIN  5:
DWG>SHOW MARGIN
LEFT MARGIN: 5, RIGHT MARGIN: 100
```

Этот пример показывает, как изменить установку левой границы, не меняя установку правой границы.

1.56. Команда SET MAX_SOURCE_FILES

Команда SET MAX_SOURCE_FILES определяет максимальное число исходных файлов, которые отладчик может держать открытыми одновременно.

Формат:

```
SET MAX_SOURCE_FILES N
```

Параметр:

N - указывает максимальное число исходных файлов, которые отладчик может держать открытыми одновременно. Значение N не может превышать 20; значение по умолчанию - 5.

Квалификаторы:

Отсутствуют.

Описание.

По умолчанию отладчик может держать открытыми пять исходных файлов одновременно. Для открытия исходного файла требуется канал ввода-вывода, что ограничивается ресурсами системы. Каналы ввода-вывода используются как программой, так и отладчиком. Чтобы быть уверенным, что отладчик не задействует все имеющиеся каналы ввода-вывода и, следовательно, не вызвет отказа программы (из-за отсутствия канала), можно выдать команду SET MAX_SOURCE_FILES с указанием максимального числа исходных файлов (и, следовательно, каналов ввода-вывода исходных файлов), которые отладчик может использовать одновременно.

Надо отметить, что значение SET MAX_SOURCE_FILES не ограничивает число открываемых отладчиком исходных файлов: оно ограничивает число открытых одновременно файлов. Следовательно, если отладчик доходит до этого предела, он должен закрыть один из файлов, чтобы открыть другой.

Надо также отметить, что установка MAX_SOURCE_FILES на слишком малое число может сделать использование отладчиком исходных файлов неэффективным.

Пример.

```
DBG>SHOW MAX_SOURCE_FILES
MAX_SOURSE_FILES: 5
DBG>SET MAX_SOURCE_FILES 8
DBG>SHOW MAX_SOURCE_FILES
MAX_SOURCE_FILES: 8
```

Команда SET MAX_SOURCE_FILES 8 позволяет отладчику иметь одновременно восемь открытых файлов.

1.57. Команда SET MODE

Команда SET MODE устанавливает режимы, действующие по умолчанию.

Формат:

```
SET MODE режим[,...]
```

Параметры:

Режим - указывает режим, определяемый следующими ключевыми словами:

1) DYNAMIC - позволяет динамическую установку модуля. Динамическая установка модуля действует по умолчанию. При этом отладчик автоматически устанавливает необходимые модули во время выполнения программы, так что нет необходимости выдавать команду SET MODULE для этих модулей. При выдаче запроса отладчика (при прерывании отладчиком выполнения), отладчик автоматически устанавливает модуль, включающий ячейку PC, и выдает информационное сообщение. Если модуль уже установлен, динамическая установка не действует;

2) NODYNAMIC - отменяет динамическую установку модуля. В этом случае надо устанавливать модули с помощью команды

SET MODULE;

3) G_FLOAT - заставляет отладчик интерпретировать константы с плавающей запятой двойной точности, имеющиеся в выражениях, как вещественные числа формата G;

4) NOG_FLOAT - заставляет отладчик интерпретировать константы с плавающей запятой двойной точности, имеющиеся в выражениях, как числа формата D. Действует по умолчанию;

5) KEYPAD - разрешает режим работы с малой клавиатурой, который является разрешенным по умолчанию. В этом случае можно использовать клавиши малой клавиатуры для выполнения предопределенных отладчиком функций. Несколько наиболее удобных для экранного режима команд отладчика уже связаны с клавишами, но можно заново определить функции клавиш с помощью команды DEFINE/KEY;

6) NOKEYPAD - отменяет режим работы с малой клавиатурой. При этом клавиши не имеют определенных отладчиком функций, а так же нельзя определить этим клавишам функции с помощью команды DEFINE/KEY;

7) LINE - указывает, что по возможности отладчик отображает программные ячейки в виде номеров строк. Этот режим является разрешенным по умолчанию;

8) NOLINE - указывает, что отладчик выдает программные ячейки в виде смещения от начального адреса в байтах, а не в виде номеров строк;

9) SCREEN - разрешает экранный режим, который подробно описан в разделе 8 26.00152-01 34 07-1;

10) NOSCREEN - блокирует экранный режим. Экранный режим отменен по умолчанию;

11) SCROLL - разрешает режим прокрутки. Этот режим разрешен по умолчанию. В режиме прокрутки кадры выходной

информации экранного режима корректируется путем сдвига строк по мере их появления;

12) NOSCROLL - отменяет режим прокрутки. В этом случае кадры выходной информации экранного режима корректируется только после выполнения команды отладчика вместо построчной корректировки. Отмена режима прокрутки уменьшает количество экранных корректировок, которые имеют место, и особенно удобна при использовании с медленно действующими терминалами;

13) SYMBOLIC - разрешает символический режим. Этот режим является допустимым по умолчанию. В символическом режиме отладчик выдает ячейки, указанные адресными выражениями, и операнды команд в символическом виде (если возможно). PSL выдается всегда в форматированном виде;

14) NOSYMBOLIC - блокирует символический режим. В этом случае отладчик выдает ячейки, указанные адресными выражениями, в виде числовых адресов, которые по умолчанию выдаются с десятичным основанием. Для задания другого основания может использоваться команда SET RADIX.

Квалификаторы:

Отсутствуют.

Описание.

В одной команде SET MODE можно указать сразу несколько режимов, отделив их запятыми. Режимы, устанавливаемые данной командой, могут быть отменены командными квалификаторами режима.

Хотя и можно установить режим основания с помощью ключевых слов (например, SET MODE OCTAL), команда SET RADIX является предпочтительным способом установки режима основа-

ния.

Пример.

```
DBG>SET MODE SCREEN
```

Эта команда переводит отладчик в экранный режим.

1.58. Команда SET MODULE

Команда SET MODULE копирует информацию относительно символических имен указанного модуля (модулей) или всех модулей в таблицу текущих символических имен (RST).

Формат:

```
SET MODULE [имя_модуля[,...]]
```

Параметры:

Имя_модуля - указывает имя модуля, символические имена которого должны копироваться в таблицу RST.

Квалификаторы:

/ALL - указывает, что символьные записи для всех модулей должны копироваться в таблицу RST.

/ALLOCATE - расширяет пул памяти отладчика. Отладчик расширяет свой пул памяти за счет области P0 каждый раз, когда его внутренний пул памяти расходуется. Отладчик распределяет дополнительную память с помощью LIB$GET_VM. Если программа пользователя также использует LIB$GET_VM, это может повлиять на распределение ее памяти. В этом случае может возникнуть необходимость избежать использования квалификакотора /ALLOCATE.

Описание.

Символьные записи должны быть представлены в таблице RST для того, чтобы отладчик мог интерпретировать соответствующие символические имена в сеансе отладки. При начальном запуске отладчика символьные записи для модуля, содержащего адрес передачи, копируются в таблицу RST. Для включения символьных записей других модулей следует указать эти модули в команде SET MODULE.

Таблица RST достаточно велика для размещения символьных записей для шести наибольших модулей программы пользователя.

Однако команда SET MODULE автоматически расширяет пул памяти, если она выдается до передачи управления программе пользователя. Поэтому выдача команды SET MODULE/ALL в начале сеанса отладки ограждает от появления сообщений отладчика "нет свободной памяти" (NO FREE STORAGE). С другой стороны, команда SET MODULE/ALL, для завершения которой требуется продолжительное время, не будет остановлена досрочно из-за выхода за пределы памяти.

Следует отметить, что если параметр в команде SET SCOPE указывает программную ячейку модуля, символьных записей которого еще нет в таблице RST, то отладчик копирует эти записи для этого модуля в таблицу RST при выполнении команды SET SCOPE. Другими словами, команда SET SCOPE может иметь побочный эффект установки модуля.

Примеры:

1. DBG>SET MODULE SUB1

Эта команда копирует информацию о символических именах модуля SUB1 в таблицу RST.

2. DBG>SET MODULE/ALL

Эта команда копирует информацию о символических именах всех модулей в таблицу RST.

1.59. Команда SET OUTPUT

Команда SET OUTPUT определяет форму выходной информации отладчика, т.е. способ, в соответствии с которым воспроизводятся и записываются ответы отладчика на команды.

Формат:

SET OUTPUT параметр[,параметр...]

Параметры:

LOG - указывает, что выходная информация отладчика, а так же входная информация пользователя должны записываться в файл регистрации. Если пользователь определяет имя файла регистрации с помощью команды SET LOG, то запись осуществляется в этот файл. В противном случае отладчик выполняет запись в файл со спецификацией по умолчанию DEBUG.LOG.

NOLOG - указывает, что выход отладчика и ввод пользователя записываться в файл регистрации не будут. Этот параметр действует по умолчанию.

[NO]SCREEN_LOG - указывает, что выход отладчика и вход пользователя при работе в экранном режиме записываются в файл регистрации. Если имя файла регистрации определяется командой SET LOG, отладчик делает запись в этот файл, иначе отладчик записывает в файл со спецификацией файла по умолчанию DEBUG LOG.

TERMINAL - указывает, что вывод отладчика должен воспроизводиться на терминале. Это параметр вывода по умолча-

нию.

NOTERMINAL - указывает, что вывод отладчика (за исключением диагностических сообщений) не будет воспроизводиться на терминале.

VERIFY - указывает, что отладчик включает в свой вывод каждую входную командную строку каждой выполняемой командной процедуры или последовательности команд DO.

NOVERIFY - указывает, что отладчик не включает в свой вывод каждую входную командную строку выполняемой командной процедуры или последовательности команд DO. Это параметр вывода по умолчанию.

Квалификаторы:

Отсутствуют.

Описание.

Пользователь может сообщить отладчику о необходимости отображения его выходной информации на терминале, записи в файл регистрации или и то и другое одновременно. Кроме того, пользователь может сообщить отладчику о необходимости включения в его вывод каждой входной командной строки любой командной процедуры или выполняемой им последовательности команд DO.

По умолчанию отладчик записывает свой вывод на терминал, а не в файл регистрации, и не включает в свой вывод каждую входную командную строку любой командной процедуры или последовательности команд DO.

Для изменения конфигурации вывода, действующей по умолчанию, следует использовать команду SET OUTPUT с любым из шести следующих параметров: TERMINAL, NOTERMINAL, LOG, NOLOG, VERIFY, NOVERIFY.

В одной данной команде можно указать несколько параметров через запятые.

Пример.

```
DBG>SET OUTPUT/LOG/NOTERMINAL
```

Эта команда указывает, что отладчик выполняет следующее:

- выводит каждую входную командную строку командной процедуры или последовательности DO, которую он выполняет;
- записывает выход отладчика и вход пользователя в файл регистрации;
- не отображает выходную информацию на терминале (за исключением диагностических сообщений).

1.60. Команда SET PROMPT

Команда SET PROMPT позволяет изменять символы запроса отладчика с DBG> на символы по выбору пользователя.

Формат:

```
SET PROMPT строка_символов
```

Параметры:

Строка_символов - указывает набор символов, который должен стать новым обозначением запроса отладчика. Если набор имеет пробелы, точки с запятой или знаки нижнего регистра, она должна заключаться в одиночные или двойные кавычки.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SET PROMPT "¤"
¤ SET PROMPT "D B G
D B G   SET PROMPT DBG>
DBG>
```

Последовательные команды SET PROMPT изменяют знак запроса отладчика с DBG> на ¤, затем на D B G :, после чего обратно на DBG>.

1.61. Команда SET RADIX

Команда SET RADIX изменяет режимы основания по умолчанию для ввода и вывода на требуемые.

Формат:

```
SET RADIX основание
```

Параметр:

Основание - указывает основание системы счисления, которое устанавливается как основание по умолчанию. Действительны следующие ключевые слова:

1) BINARY - устанавливает двоичное основание;

2) DECIMAL - устанавливает десятичное основание;

3) OCTAL - устанавливает восьмеричное основание;

4) HEXADECIMAL - устанавливает шестнадцатеричное основание.

Квалификаторы:

/INPUT - устанавливает режим основания только для входной информации.

/OUTPUT - устанавливает режим основания только для выходной информации.

Описание.

Реально команда SET RADIX является наиболее слабым способом установки основания. При задании режима основания командой SET RADIX отладчик интерпретирует и выдает числовые константы, включая целые числа и адреса, как величины с этим основанием. Однако другие значения (например, величины плавающего типа или величины перечисления) интерпретируются и выдаются в своем нормальном виде.

Основание, установленное этой командой, может быть отменено или командой SET RADIX/OVERRIDE или квалификатором основания в некоторых других командах отладчика (таких, как EVALUATE и EXAMINE).

Примеры:

1. DBG>SET RADIX HEX

Эта команда устанавливает шестнадцатеричное основание по умолчанию.

2. DBG>SET RADIX/INPUT OCT

Эта команда устанавливает восьмеричное основание по умолчанию для входной информации.

3. DBG>SET RADIX/OUTPUT BIN

Эта команда устанавливает двоичное основание по умолчанию для выходной информации.

1.62. Команда SET RADIX/OVERRIDE

Команда SET RADIX/OVERRIDE изменяет основание по умолчанию для выходной информации на заданное, отменяя всю имеющуюся информацию о типе.

Формат:

SET RADIX/OVERRIDE основание

Параметр:

Основание - указывает основание, отменяющее основание по умолчанию. Действительны следующие ключевые слова:

1) BINARY - устанавливает двоичное основание;

2) DECIMAL - устанавливает десятичное основание;

3) OCAL - устанавливает восьмеричное основание;

4) HEXADECIMAL - устанавлизает шестнадцатеричное основание..

Квалификаторы:

/OUTPUT - устанавливает режим основания, отменяющий другие, для выходной информации.

Описание.

Команда SET RADIX/OVERRIDE является вторым мощным средством установки основания. Она отменяет действие команды SET RADIX. Если выдается, например, команда SET RADIX/OVERRIDE HEX, то все значения будут представлены как шестнадцатЕричные целые числа, независимо от генерированного компиляторэм типа. Для отмечы основания, установленного этой командой, применяется команда CANCEL RADIX/OVERRIDE.

Команда SET RADIX/OVERRIDE работает только для режима основания выходной информации. Однако, надо отметить, что использование квалификатора основания является наиболее действенным способом установки основания.

Примеры:

1. DBG>SET RADIX/OVERRIDE HEX

Эта команда устанавливает основание по умолчанию для

выходной информации на шестнадцатеричное.

2. DBG>SET RADIX/OVERRIDE OCTAL

Эта команда устанавливает основание по умолчанию для выходной информации на восьмеричное.

1.63. Команда SET SCOPE

Команда SET SCOPE указывает одну или несколько ячеек программы, которые должны быть использованы при интерпретации символических имен.

Формат:

SET SCOPE ячейка[,ячейка...].

Параметры:

Ячейка - указывает одну или несколько программных ячеек, определяемых именами пути или другими специальными символами:

- имя пути - это одна или несколько меток программных ячеек, разделенных символом обратной косой черты (\), определяющих ячейку программы или диапазон программных ячеек. Метки программной ячейки могут быть именами модуля, подпрограммы или блока, а также номерами строк и числовыми метками. Общий формат имени пути выглядит следующим образом: модуль\программа\блок;

- целые 0, 1, 2,... - это ряд десятичных целых, представляющих вызовы активной в данный момент программной единицы; нуль представляет самый последний ее зызов, единица представляет предпоследний вызов и т.д.;

- обратная косая черта (\) - представляет набор всех программных ячеек, в которых находится описание какого-либо

глобального символического имени в программе.

Квалификаторы:

/MODULE имя_модуля - устанавливает область действия на указанный модуль. Если модуль еще не установлен, отладчик устанавливает этот модуль и определяет, что область действия - данный модуль.

Описание.

Команда SET SCOPE определяет область действия или местонахождение в программе пользователя, которая используется отладчиком для разрешения символических ссылок. При просмотре переменной X отладчик определяет по умолчанию, какую переменную пользователь имеет в виду, в зависимости от текущего положения программы. Такая область действия называется диапазоном 0.

Если необходимо рассмотреть переменные, расположенные в другом месте (например, в другой подпрограмме или модуле), можно выдать команду SET SCOPE с указанием подпрограммы или модуля. Например, для доступа к переменной X из программы YODA модуля JEDI следует выдать команду SET SCOPE\JEDI\YODA. Отладчик будет трактовать ссылки к X соответственно как JEDI\YODA\X.

Отладчик позволяет интерпретировать символические имена без префиксов имени пути интерпретацией символического имени как если бы оно появилось в ячейках, обозначенных первым параметром команды SET SCOPE. Если отладчик не может интерпретировать имя, используя первый параметр, он использует второй параметр команды SET SCOPE и продолжает этот процесс до тех пор, пока или не разрешит символическое имя или пока не кончится перечень параметров.

Если в команде указывается более одного параметра, то тем самым устанавливается список поиска диапазона для интерпретации имен без префиксов имени пути.

Помимо префиксов имени пути, в команде SET SCOPE можно использовать в качестве параметров обратную косую черту и набор целых десятичных чисел (0,1,2,...). Обратная косая черта представляет набор всех ячеек программы, в которых имеется объявление какого-либо глобального символического имени. Числа представляют вызовы действующих программных единиц: 0 представляет наиболее последний (недавний) вызов, 1 - следующий наиболее последний (предыдущий от недавнего) вызов и т.д.

При задании имени пути команда SET SCOPE пытается установить диапазон на подпрограмму с указанным именем. Если подпрограммы с таким именем нет в таблице RST, но существует модуль с таким именем, то этот модуль устанавливается (заносится в RST). Затем диапазон устанавливается на подпрограмму с данным именем, если такая существует. Если все же такой подпрограммы нет, то диапазон устанавливается на модуль.

Поэтому, если необходимо явно установить диапазон на модуль, когда существует подпрограмма с таким же именем, необходимо использовать команду SET SCOPE/MODULE.

Например, если программа содержит и модуль и подпрограмму с именем MAIN, команда SET MODULE MAIN установит диапазон на подпрограмму. Для установки диапазона на модуль необходима команда SET SCOPE/MODULE MAIN. Однако, если программа содержит только модуль MAIN, то SET MODULE MAIN установит диапазон на модуль.

Пример.

DBG>SET SCOPE 0,1,2,3,4,5

Эта команда указывает отладчику первоначальный поиск имен в диапазоне, определенным текущим значением PC (диапазон 0), затем обследовать подпрограмму, вызывающую текущую, затем подпрограмму, вызывающую данную, и т.д.

1.64. Команда SET SEARCH

Команда SET SEARCH устанавливает текущие параметры поиска для использования отладчиком в случае, когда не был задан квалификатор команды SEARCH.

Формат:

SET SEARCH [параметр,параметр]

Параметры:

ALL - указывает, что отладчик отыскивает все местоположения цепочки в заданной области и отображает каждую строку, содержащую эту цепочку.

IDENTIFIER - определяет поиск отладчиком местоположения цепочки в заданной области, но выдачу его только тогда, когда цепочка не граничит с каждой стороны символом, который может быть частью идентификатора в текущем языке программирования.

NEXT - определяет поиск отладчиком первого встретившегося местоположения цепочки в указанной области и выдачу строки, содержащую это местоположение. Данный параметр работает по умолчанию.

STRING - определяет поиск и отображение отладчиком указанной цепочки, при этом отладчик не рассматривает кон-

текст вокруг местоположения цепочки, как это делается в случае параметра IDENTIFIER.

Квалификаторы:

Отсутствуют.

Описание.

Параметры SEARCH определяют, будет ли отладчик:

1) выполнять поиск всех местоположений цепочки (ALL) или только следующего (NEXT);

2) воспроизводить все местоположения этой цепочки (STRING) или только тех ее появлений, в которых цепочка с любой стороны не ограничена символом, который может быть частью идентификатора в данном языке программирования (IDENTIFIER).

Если пользователь не определяет параметры в данной команде, то отладчик использует значения NEXT и STRING по умолчанию.

Пользователь может отменить текущие параметры поиска на время выполнения одной команды SEARCH, указав требуемые параметры поиска в качестве командных квалификаторов команды SEARCH.

В одной команде SET SEARCH можно указать несколько параметров поиска, разделив их запятыми.

Пример.

```
DBG>SHOW SEARCH
SEARCH SETTINGS: SEARCH FOR NEXT OCCURRENCE, AS A STRING
DBG>SET SEACH IDENTIFIER
DBG>SHOW SEARCH
```

```
                AS AN IDENTIFIER

DBG>SHOW SEARCH ALL

DBG>SHOW SEARCH

SEARCH SETTINGS: SEARCH FOR ALL OCCURRENCE, A AN IDENTIFIER
```

Команда SET SEARCH IDENTIFIER обязывает отладчик отыскать местонахождение цепочки в указанной области, но выдать цепочку только в том случае, если она не ограничена с обеих сторон символом, который может быть частью идентификатора в текущем языке.

Команда SET SEARCH ALL обязывает отладчик найти (и выдать) все местонахождения цепочки в указанной области.

1.65. Команда SET SOURCE

Команда SET SOURCE направляет отладчик на обследование указанного каталога или каталогов для поиска исходных файлов. Эта команда используется для направления отладчика к исходному файлу, который был перемещен в другой каталог после компилирования.

Формат:

```
SET SOURCE[/MODULE=имя_модуля]имя_каталога[,имя_-
каталога ...]
```

Параметры:

Имя_каталога - определяет обследуемый каталог.

Следует отметить, что имя_каталога может состоять из одного, нескольких или всех полей полной спецификации файла, хотя обычно это только имя каталога. Полная спецификация файла имеет следующий формат:

узел::устройство:[каталог]имя_файла.тип_файла;номер_версии

При определении любого из данных. Полей следует включать соответствующую пунктуацию этого поля.

Квалификаторы:

/MODULE=имя_модуля - определяет, что указанный список обследуемых каталогов должен быть использован для нахождения исходных файлов только для данного модуля (имя_модуля).

Описание.

По умолчанию отладчик предполагает, что исходный файл находится в том же каталоге, в который его поместили во время компилирования.

Если в одной такой команде SET SOURCE пользователь указывает более одного каталога, разделяя имя каждого каталога запятой, то создается список обследуемых каталогов для поиска исходных файлов. Отладчик обрабатывает этот список поиском в первом указанном каталоге для нахождения исходного файла соответствующего модуля, затем поиск выполняется во втором каталоге и т.д., вплоть до нахождения исходного файла или окончания списка каталогов.

Если указывается квалификатор /MODULE=имя_модуля, то отладчик использует заданный список каталогов только для нахождения исходных файлов указанного модуля.

Если выдается команда SET SOURCE без квалификатора /MODULE=имя_модуля, то отладчик использует указанный список каталогов для нахождения исходных файлов всех модулей, которые не были заданы в предыдущей команде SET SOURCE/MODULE=имя_модуля.

Примеры:

1. DBG>SHOW SOURCE

NO DIRECTORY SEACH LIST IN EFFECT

(список каталогов отсутствует)

DBG>SET SOURCE [PROJA],[PROJB],DISK:[PETER.PROJC]

DBG>SHOW SOURCE

SOURCE DIRECTORY SEACH LIST FOR ALL MODULES:

(список каталогов для всех модулей:)

[PROJA]

[PROJB]

DISK:[PETER.PROJC]

Эта команда сообщает отладчику о необходимости обследования каталогов [PROJA], [PROJB] и DISK:[PETER PROJC].

2. DBG>SET SOURCE/MODULE=COBOLTEST DEVICE:[PROJD],[014,015]

DBG>SHOW SOURCE

SOURCE DIRECTORY SEACH LIST FOR COBOLTEST:

(список поиска каталогов для модуля COBOLTEST:)

DEVICE:[PROJD]

[014,015]

SOURCE DIRECTORY SEACH LIST FOR ALL OTHER MODULES:

(список поиска каталогов для всех других модулей:)

[PROJA]

[PROJB]

DISK:[PETER.PROJC]

Эта команда сообщает отладчику о необходимости обследования каталогов DEVICE:[PROJD] и [014,015] для поиска исходных файлов COBOLTEST.

1.66. Команда SET STEP

Команда SET STEP устанавливает текущие параметры шага для случая, когда не был определен квалификатор команды STEP.

Формат:

SET STEP параметр[,параметр...]

Параметры:

Параметр - принимает одно из следующих значений:

1) BRANCH - пошаговый режим вызывает переход к следующему семейству команд ветвления (BRANCH);

2) CALL - пошаговый режим вызывает переход к следующим командам вызова или возврата;

3) EXCEPTION - пошаговый режим вызывает переход до следующего условия исключительной ситуации;

4) INSTRUCTION - пошаговый режим вызывает выполнение одной инструкции от ячейки, где было последний раз приостановлено выполнение;

5) INSTRUCTION=код_операции - пошаговый режим вызывает переход отладчика к инструкции, чей код операции определен той же мнемоникой. Пользователь может определять несколько мнемоник кода операции, отделив их запятыми, а весь перечень мнемоник заключив в круглые скобки;

6) INTO - аход "в" вызываемые подпрограммы в пространстве программы пользователя (а также "в" вызываемые подпрограммы в системном пространстве, если также действует параметр SYSTEM). Это означает, что при выполнении шагов отладчик не делает различия между кодом в пределах вызываемой подпрограммы и кодом вне вызываемой подпрограммы;

7) LINE - пошаговый режим вызывает выполнение всего кода программы пользователя от ячейки, где было последний раз приостановлено выполнение, и до границы следующей строки;

8) OVER - проход "поверх" вызываемых подпрограмм в пространстве программы пользователя и в системном пространстве. Это означает, что отладчик рассматривает любой код, выполненный в результате инструкции CALL, от инструкции CALL до соответствующей инструкции RETURN как часть одного шага (STEP). Следует при этом отметить, что сама подпрограмма при этом выполняется;

9) RETURN - шаги осуществляются до следующей инструкции возврата, которая выдается из текущей подпрограммы. Этот параметр действует только для подпрограмм типа CALLS и CALLG;

10) [NO]SILENT - регулирует распечаткой сообщения на терминале, когда отладчик работает в пошаговом режиме. NOSILENT определяет, что сообщение выдается. Этот параметр является параметром по умолчанию. SILENT устанавливает, что никакое сообщение о выполнении шагов не воспроизводится;

11) SOURCE - воспроизведение строки исходного кода, которая соответствует команде (или командам), выполяемой при каждом шаге. Следует отметить, что данный параметр вызывает также воспроизведение отладчиком строк исходного кода при приведении в действие точки приостанова или точки просмотра;

12) NOSOURCE - данный параметр указывает, что строки исходного кода не должны воспроизводиться при выполнении команды STEP. Следует также отметить, что данный параметр приводит к тому, что отладчик не воспроизводит строки

исходного кода при приведении в действие точки приостанова или точки просмотра;

13) [NO]SYSTEM - управляет пошаговым прохождением в системной области. SYSTEM разрешает вход "в" подпрограммы системной области (при условии, что действует также параметр INTO), т.е. при выполнении шагов отладчик не делает различия между кодами внутри и вне вызванной подпрограммы из системной области. Таким образом, выполнение команды STEP в этом случае позволяет останавливаться в системном пространстве. NOSYSTEM приводит к выполнению шагов "поверх" подпрограмм системной области, т. е. отладчик рассматривает любой выполняемый в системной области код от команды CALL до соответствующей команды RETURN как один шаг. (но при этом сама подпрограмма выполняется).

Квалификаторы:

Отсутствуют.

Описание.

Параметры, указанные в данной команде, определяют шаги отладчика по строкам или по программным командам, вход отладчика "в" или проход "поверх" вызываемых подпрограмм, а также воспроизведение отладчиком строк исходного кода по мере выполнения этих шагов.

Каждый язык программирования устанавливает параметры шага по умолчанию. При изменении пользователем данного языка программирования параметры STEP устанавливаются в соответствии с новым языком. Пользователь может изменить эти параметры с помощью команды SET STEP.

Пользователь может определять в данной команде несколько параметров, разделяя их запятой.

Примеры:

1. DBG>SET STEP INS/INTO

По этой команде отладчик выполняет одну программную инструкцию от ячейки, где была приостановлена программа последний раз. Команда также разрешает отладчику пошаговый вход в вызываемые подпрограммы, одинаково обрабатывая код в пределах вызванной подпрограммы и вне ее.

2. DBG>SET STEP SOURCE

По этой команде отладчик выдает строки исходного кода по мере пошагового выполнения программы.

1.67. Команда SET TERMINAL

Команда SET TERMINAL устанавливает ширину и/или высоту терминального экрана, используемую отладчиком при форматировании экранной и другой выходной информации.

Формат:

SET TERMINAL/квалификатор1:N[/квалификатор2:M].

Параметры:

Отсутствуют.

Квалификаторы:

Пользователь должен указать по крайней мере один квалификатор: либо /PAGE, либо /WIDTH. Он может указать оба квалификатора /PAGE и /WIDTH. Пользователь должен установить значение для каждого использованного квалификатора.

/PAGE:N - указывает, что высота терминального экрана должна быть установлена на N строк. Можно использовать любое значение от 11 до 100.

/WIDTH:N - указывает, что ширина терминального экрана

должна быть установлена на N знаков (позиций). Можно использовать любое значение от 20 до 255. Для терминала серий VT100 или VT200 N обычно либо 80 либо 132. Для терминальных станций это значение может быть больше.

Описание.

Команда SET TERMINAL (использованная с квалификатором /PAGE или /WIDTH) не устанавливает высоту или ширину реального экрана: она устанавливает высоту или ширину экрана, используемые отладчиком для форматирования выходной информации.

Команда SET TERMINAL/PAGE устанавливает число строк на экране, используемое отладчиком при переводе предопределенных окон экранного режима, таких как Q1, T1, H1 и так далее. Команда SET TERMINAL/WIDTH устанавливает ширину окна, которую отладчик использует при форматировании экрана и выходной информации. Ни /PAGE, ни /WIDTH не уменьшают и не увеличивают динамически размер экранных кадров. Поэтому после применения команды SET TERMINAL можно использовать команду DISPLAY для повторного воспроизведения этих кадров в заново определенных окнах. (Можно также использовать сочетание клавиш малой клавиатуры, такие как BLUE-MINUS, для ввода предопределенных команд DISPLAY).

После изменения размеров экран очищается.

Пример.

```
DBG>SET TERMINAL/WIDTH:110/PAGE:36
```

Эта команда указывает, что ширина терминального экрана должна быть 110 знаков, а высота 36 строк.

1.68. Команда SET TRACE

Команда SET TRACE устанавливает точки отслеживания в ячейке, определяемой указанным адресным выражением.

Формат:

SET TRACE [адресное_выражение[,адресное_выражение,...]]-
[WHEN(выражение)][DO(команда[;команда...])]

Параметры:

Адресное_выражение - указывает ячейку, в которой должна быть установлена точка отслеживания. Значение параметра зависит от выбранных квалификаторов. Например, если квалификаторы не указаны, то адресное выражение относится к адресу отдельной инструкции, и отладчик устанавливает точку отслеживания на этой инструкции. Для более полной информации об интерпретации параметра адресное_выражение необходимо обратиться к описанию квалификаторов данной команды.

Команда - указывает любую команду отладчика, которая выполняется отладчиком как часть командной последовательности DO при достижении точки отслеживания. Если указывается более одной команды, то необходимо отделять их точкой с запятой.

Выражение - указывает любое выражение на текущем языке программирования, которое, необходимо вычислить каждый раз, когда встречается точка отслеживания. Если значение выражения - "истина" (TRUE), то выполняется отслеживание, и пользователь получит соответствующее сообщение.

Однако, если выражение - "ложь" (FALSE), то отслеживания не происходит, сообщение не выдается, командная последовательность DO не выполняется, а программа выполняется

дальше.

Квалификаторы:

/AFTER:N - указывает, что отслеживание в данной точке не происходит до N-го прохождения через нее. Затем (с N-го раза) точка отслеживания действует каждый раз, когда обеспечиваются условия TRUE в предложении WHEN. Значение N интерпретируется в режиме десятичного основания независимо от установок режима основания.

/BRANCH - указывает, что точки отслеживания должны быть установлены на каждом из следующих членов семейства инструкций BRANCH: BEQL, BGTR, BLEQ, BGEQ, BLSS, BGTRU, BLEQU, BVC, BVS, BGEQU, BLSSU, BRB, BRW, JMP, BBS, BBC, BBSS, BBCS, BBSC, BBCC, BBSSI, BBCCI, BLBS, BLBC, ACBB, ACBW, ACBL, ACBF, ACBD, ACBG, ACBH, AOBLEQ, AOBLSS, SOBGEQ, SOBGTR, CASEB, CASEW и CASEL. Параметр адресное_выражение не имеет значения для этого квалификатора, и его нужно опустить.

/CALL - указывает, что точки отслеживания должны быть установлены на каждом из следующих членов инструкций семейства CALL: CALLS, CALLG, BSBW, BSBB, JSB, RSB и RET. Параметр адресное_выражение не имеет значения для этого квалификатора, поэтому нужно опустить его.

/EXCEPTION - указывает, что отслеживание должно быть каждый раз, когда сигнализируется исключительная ситуация. Отслеживание происходит до вызова пользовательского обработчика исключительной ситуации. Параметр адресное_выражение нужно опустить.

/INSTRUCTION - указывает, что отладчик должен установить точку отслеживания на каждой программной команде.

Параметр адресное_выражение нужно опустить.

/INSTRUCTION=(код_операции[,...]) - указывает, что отслеживание должно быть для каждой команды, чей код операции дается одной из указанных мнемоник. Можно указывать несколько мнемоник кода операций, отделенных друг от друга запятыми и заключенных в круглые скобки. Параметр адресное_выражение не имеет значения для этого квалификатора, и его необходимо опустить.

/LINE - указывает, что отслеживание должно происходить в начале каждой новой строки программы. Параметр адресное_выражение нужно опустить.

/MODIFY - указывает, что отслеживание должно выполняться на каждой команде, которая записывает и изменяет значение ячейки (ячеек), указанной параметром (параметрами) адресного выражения. Если пользователь задает абсолютный адрес в качестве адресного_выражения, то отладчик не может ассоциировать адрес с определенным об"ектом данных. В этом случае отладчик использует длину по умолчанию в четыре байта. Можно, однако, изменить эту длину установкой типа WORD (который меняет длину по умолчанию на 2 байта) или BYTE (который меняет длину по умолчанию на 1 байт).

/RETURN - указывает установку отладчиком точки отслеживания на инструкции RETURN встретившейся подпрограммы. Этот квалификатор может быть применен только к подпрограммам, вызываемым командами CALLS или CALLG; он не может использоваться с подпрограммами JSB. Отслеживание инструкции RET также позволяет проверить локальную среду перед удалением командой RET кадра вызова подпрограммы из стека вызова. Для этого квалификатора параметр адресное_выражение является адресом команды в пределах подпрограммы, вызыва-

мой с помощью CALLS или CALLG. Это может быть просто именем подпрограммы, в этом случае он указывает начальный адрес подпрограммы. Однако можно также указывать другую ячейку в подпрограмме, в таком случае можно увидеть только те возвраты, которые происходят после прохождения через эту ячейку.

/[NO]SILENT - управляет распечаткой сообщения. Этот квалификатор иногда полезен с оператором DO.

/[NO]SOURCE - управляет воспроизведением исходного текста программы при достижении точки отслеживания. /SOURCE указывает на необходимость выдачи исходного текста. Действует по умолчанию. /NOSOURCE указывает, что исходный текст не нужен.

/TEMPORARY - указывает на необходимость, чтобы точка отслеживания после отработки исчезла. Другими словами, точка отслеживания не сохраняет постоянной установки.

Описание.

Когда приводится в действие точка отслеживания, отладчик приостанавливает выполнение программы, выдает сообщение о достижении точки отслеживания и возобновляет выполнение программы.

Примеры:

1. DBG>SET TRACE %LABEL 5

Эта команда устанавливает точку отслеживания в ячейке %LABEL 5.

2. DBG>SET TR/BRA

Эта команда заставляет отладчик отслеживать все команды семейства BRANCH.

3. DBG>SET TR/CA

Эта комэнда заставляет отладчик отслеживать все инструкции вызова подпрограмм.

1.69. Команда SET TYPE

Команда SET TYPE обеспечивает установку типа по умолчанию для привязки к программной ячейке без типа и, при определении квалификатора /OVERRIDE, установку типа для привязки как к программным ячейкам без типа, так и к тем программным ячейкам, которые имеют сгенерированные компилятором типы.

Формат:

SET TYPE[/квалификатор] ключевое_слово_типа

Параметры:

Ключевое_слово_типа - может представлять один из следующих типов:

1) ASCIC - устанавливает тип на подсчитанную строку кои-8;

2) ASCID - устанавливает тип дескриптора строки кои-8. Поля CLASS и DTYPE дескриптора не проверяются, а поля LENGTH и POINTER определяют длину и адрес строки КОИ-8. Затем строка отображается на экране;

3) ASCII:N - тип символов кода КОИ-8 (длина N байтов). Длина указывает как число байтов памяти, необходимых для просмотра, так и число символов КОИ-8, необходимых для воспроизведения. В данном случае каждый символ занимает один байт памяти. Если пользователь не определяет величину N, то отладчик принимает длину по умолчанию в 4 байта.

величина N интерпретируется с десятичным основанием;

4) ASCIW - устанавливает тип для переменной строки КОИ-8 (двухбайтовое поле длины, за которым сразу следует строка КОИ-8 указанной длины). Этот тип данных встречается в Паскале и Пл/1;

5) ASCIZ - устанавливает тип строки КОИ-8, заканчивающейся нулем;

6) BYTE - тип байтового целого (длина 1 байт);

7) DATE_TIME - устанавливает тип дата-время (выдача времени);

8) D_FLOAT - тип числа с плавающей точкой формата D (длина 8 байтов). Величины данного типа имеют диапазон .29*10**-38 - 1.7*10**38 с точностью приблизительно 16 десятичных цифр;

9) FLOAT - тип числа с плавающей точкой (длина 4 байта). Величины данного типа имеют диапазон .29*10**-38 - 1.7*10**38 с точностью приблизительно 7 десятичных цифр;

10) G_FLOAT - тип числа с плавающей точкой формата G (длина 8 байтов). Величины данного типа имеют диапазон .56*10**-308 - .9*10**308 с точностью приблизительно 15 десятичных цифр;

11) H_FLOAT - тип числа с плавающей точкой формата H (длина 16 байтов). Величина данного типа имеет диапазон .84*10**-4932 - .59*10**4932 с точностью приблизительно 33 десятичные цифры;

12) INSTRUCTION - тип инструкции системы МОС ВП, длина которого является переменной в зависимости от числа операндов инструкции и используемых режимов адресации;

13) LONG - тип целого длинного слова (длина 4 байта);

14) OCTWORD - тип целого октаслова (длина 16 байтов);

14) PACKED:N - устанавливает тип целое упакованное число (длина 4N байт);

15) QUADWORD - тип целого квадрослова (длина 8 байтов);

16) WORD - тип целого слова (длина 2 байта).

Квалификатор:

/OVERRIDE - указывает, что заданный тип должен быть связан с программными ячейками без типа, а также с теми ячейками, которые имеют сгенерированные компилятором типы.

Описание.

Следующие типы отладчика являются приемлимыми параметрами в командах SET TYPE или SET TYPE/OVERRIDE: ASCIC, ASCID, ASCII:N, ASCIW, ASCIZ, BYTE, WORD, LONG, QUAIWORD, OCTWORD, FLOAT, D_FLOAT, G_FLOAT, H_FLOAT, PACKED:N, DATE_TIME и INSTRUCTION.

Типы по умолчанию и типы отладчика, отменяющие другие типы, оказывают влияние на интерпретацию и воспроизведение программных объектов, определяемых с помощью адресных выражений в таких командах отладчика как EXAMINE, DEPOSIT и EVALUATE.

Типы, указанные в качестве командных квалификаторов в командах отладчика, имеют приоритет над типами, указанными в качестве параметров в командах SET TYPE и SET TYPE/OVERRIDE.

Примеры:

1. DBG>SET TYP ASC:8

Эта команда устанавливает восьмибайтовую строку символов КОИ-8 как тип по умолчанию, связанный с программными

ячейками без типа.

2. DBG>SET TYP/OVERR LONG

Эта команда устанавливает тип целого длинного слова в качестве типа по умолчанию, связанного как с программными ячейками без типа так и с теми ячейками, которые имеют сгенерированные компилятором типы.

3. DBG>SET TYP D_FLOAT

Эта команда устанавливает тип числа формата D в качестве типа, связанного с программными ячейками без типа.

1.70. Команда SET WATCH

Команда SET WATCH обеспечивает установку точки просмотра в ячейке, которая определяется квалификатором или параметром адресного выражения.

Формат:

SET WATCH адресное_выражение[,...][WHEN(выражение)]-
[DO(список_команд)]

Параметры:

Адресное_выражение - определяет ячейку, в которой должна быть установлена точка просмотра.

Команда - указывает любую команду отладчика, которую он должен выполнить как часть последовательности команд DO при просмотре. Если задается несколько команд, надо разделить их точкой с запятой.

Выражение - указывает любое выражение текущего языка, которое пользователь хочет вычислять каждый раз, когда встречается точка просмотра. Если выражение - верно (TRUE), то просмотр осуществляется, и пользователь получит соот-

ветствующее сообщение.

Однако, если выражение ложное (FALSE), то просмотр не осуществляется. В этом случае сообщение не выдается, команды, указанные оператором DO, не выполняются, выполнение программы продолжается.

Квалификаторы:

/AFTER:N - указывает, что просмотр осуществляется с N-го случая прохождения через указанную точку. Затем точка просмотра действует каждый раз, когда значение предложения WHEN - "истина" Значение N интерпретируется в режиме десятичного основания независимо от установок режима основания.

/[NO]SILENT - указывает, распечатывать или нет сообщение просмотра на терминал при достижении точки.

/[NO]SOURCE - указывает, воспроизводить или нет исходный код; /SOURSE (по умолчанию) указывает, что исходный код надо воспроизводить, /NOSOURSE - нет.

/TEMPORARY - указывает, что пользователь хочет, чтобы точка просмотра исчезла после ее активизации. Другими словами, точка просмотра не сохраняет постоянной установки.

Описание.

Когда программная команда вызывает модификацию (изменение) просматриваемой ячейки, отладчик выполняет следующие действия:

1) приостанавливает выполнение программы после окончания выполнения этой команды;

2) вычисляет выражение WHEN (если такое указывается при установке точки просмотра); если его значение - "ложь", то выполнение продолжается и отладчик не выполняет следующие шаги;

3) выдает сообщение о местоположении точки просмотра;

4) идентифицирует команду, которая модифицировала просматриваемую ячейку (и строку исходного кода, соответствующую этой команде, если параметр SOURCE действует в результате предыдущей команды SET STEP SOURCE);

5) выполняет последовательность DO (если такая указывается при установке точки просмотра);

6) сообщает значение в просматриваемой ячейке до модификации;

7) сообщает новое или модифицированное значение в просматриваемой ячейке;

8) выдает запрос DBG>.

Если просматриваемая ячейка имеет сгенерированный компилятором тип, то отладчик использует длину в байтах, связанную с этим типом, для определения длины просматриваемой ячейки. Если просматриваемая ячейка не имеет сгенерированного компилятором типа, то отладчик просматривает четыре байта виртуальной памяти, начиная с байта, определяемого адресным выражением.

Пользователь может установить точки просмотра на агрегаты (т.е. На целые массивы или записи). Точка просмотра, установленная на массиве или записи, будет срабатывать, если какой-либо элемент массива или записи изменяется. Таким образом, пользователю нет необходимости устанавливать точки просмотра на отдельные элементы массива или компоненты записи. Однако пользователь не может установить точку просмотра на записи переменной длины.

Нельзя установить точку просмотра в динамически распределяемой ячейке памяти.

Примеры:

1. DBG>SET WATCH MAXCOUNT

Эта команда устанавливает точку просмотра в ячейке MAXCOUNT.

2. DBG>SET WATCH ARR

DBG>GO

WATCH OF SUBR\ARR AT SUBR\%LINE 12+8

OLD VALUE:

(1) 7

(2) 12

(3) 3

NEW VALUE:

(1) 7

(2) 12

(3) 28

BREAK AT SUBR\%LINE 14

В этом примере команда SET WATCH устанавливает точку просмотра на трехэлементном целом массиве ARR. Затем выполнение возобновляется командой GO. Точка просмотра запускается, когда какой-нибудь элемент массива изменяется. В данном случае изменился третий элемент.

1.71. Команда SET WINDOW

Команда SET WINDOW позволяет определять окна на экране терминала.

Формат:

SET WINDOW имя_окна AT(начальная_строка,количество_строк) .

Параметры:

Имя_окна - указывает имя определяемого окна.

Начальная_строка - указывает номер начальной строки окна. Эта строка отображает заголовок окна. Верхняя строка на экране всегда является строкой 1.

Количество_строк - указывает число строк текста в окне. Строка заголовка окна не включается в общее количество строк, сумма значений параметров начальной строки и количества строк не должна превышать 20.

Квалификаторы:

Отсутствуют.

Описание.

Экранным окном является область на экране терминала, через которое можно видеть кадр. С помощью команды SET WINDOW можно определить параметры окна. Определение связывает имя окна с областью на экоане, которая указывается в виде начальной строки и высоты (количества строк).

Для размещения кадров на экране можно использовать имена окон, определенных в командах DISPLAY и SET DISPLAY.

Пример.

DBG>SET WINDOW TEMP AT(5,10)

Этой командой определяется окно с именем TEMP, которое

имеет длину в 10 строк, начиная со строки 5.

1.72. Команда SHOW

Команда SHOW отображает различную информацию, соответствующую указанному ключевому слову.

Формат:

SHOW ключевое_слово [параметр]

Параметры:

Ключевое слово - определяет тему для отображения. Могут быть использованы следующие ключевые слова: AST, BREAK, CALLS, DEFINE, DISPLAY, EXIT_HANDLERS, KEY, LANGUAGE, LOG, MARGIN, MAX_SOURCE_FILES, MODE, MODULE, OUTPUT, RADIX, SCOPE, SEARCH, SELECT, SOURCE, STEP, SYMBOL, TERMINAL, TRACE, TYPE, WATCH, WINDOW.

Параметр - зависит от указанного ключевого слова.

Квалификаторы:

Квалификаторы зависят от указанного ключевого слова.

1.73. Команда SHOW AST

Команда SHOW AST показывает, разрешено или нет поступление в программу пользователя асинхронного системного прерывания (AST). Команда SHOW AST не определяет асинхронное системное прерывание (AST), выдача которого находится в ожидании. Поступление AST разрешено по умолчанию и с помощью команды ENABLE AST. Поступление AST блокируется командой DISABLE AST.

Формат:

SHOW AST

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW AST
ASTS ARE ENABLED
DBG>DISABLE AST
DBG>SHOW AST
ASTS ARE DISABLED
DBG>
```

Команда SHOW AST показывает, разрешено ли поступление AST в программу пользователя.

1.74. Команда SHOW BREAK

Команда SHOW BREAK вызывает воспроизведение отладчиком точек приостанова, которые установлены с помощью команды SET BREAK.

Формат:

SHOW BREAK

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Отладчик выдает всю информацию о каждой установленной точке приостанова.

Если пользователь установил точку приостанова с использованием квалификатора /AFTER:N в команде SET BREAK, то команда SHOW BREAK вызывает воспроизведение текущего значения десятичного целого N, т.е. начально указанное значение минус единица для каждого случая достижения данной точки приостанова. (отладчик уменьшает величину N на единицу при каждом достижении точки приостанова до тех пор, пока величина N не станет равной нулю, после чего отладчик выполняет действие приостанова).

Пример.

```
DBG>SHOW BREAK
BREAKPOINT AT SUB1\LOOP
BREAKROINT AT MAIN\MAIN+1F DO (EX SUB1\D;
EX/SYMBOL PSL;GO)
ROUTINE BREAKPOINT /AFTER:2 AT SUB2\SUB2
```

Эта команда выдает информацию о трех установленных точках приостанова: SUB1\LOOP, MAIN\MAIN и SUB2\SUB2.

1.75. Команда SHOW CALLS

Команда SHOW CALLS вызывает воспроизведение отладчиком информации относительно последовательности активных в данный момент вызовов процедур или числа кадров вызова в стеке.

Формат:

SHOW CALLS [N]

Параметр:

N - определяет число кадров вызова, относительно которых пользователю требуется информация. Если данный параметр не указан, то отладчик воспроизводит информацию относительно всех кадров вызова.

Квалификаторы:

Отсутствуют.

Описание.

Необязательный параметр N определяет количество вызовов или число кадров вызова, о которых необходимо получить информацию. Этот параметр может быть целым десятичным числом в диапазоне 0 - 32767. Если пользователь не определяет данный параметр, то воспроизводится информация относительно всех кадров вызова.

Для каждого кадра вызова отладчик воспроизводит одну строку информации. В первой строке содержится информация относительно верхнего кадра вызова (т.е. кадра, который представляет самую последнюю вызванную процедуру); в следующей строке содержится информация относительно следующей (т.е. предпоследней) вызванной процедуры и т.д.

Каждая строка информации, воспроизводимая отладчиком, содержит имя вызванной подпрограммы, имя модуля, в котором содержится вызванная подпрограмма, номер строки вызова (только в тех языках программирования, которые ориентированы на строки кода) и величину счетчика программы (PC) в вызывающей подпрограмме во время передачи управления вызы-

ваемой подпрограмме.

Величина счетчика программы (PC) воспроизводится двумя способами: как абсолютный виртуальный адрес и как виртуальный адрес относительно виртуального адреса имени подпрограммы.

Пример.

```
DBG>SHOW CALLS
MODULE NAME    ROUTINE NAME    LINE      REL PC       ABS PC
(модуль)(имя подпрограммы)(строка)(относительный(абсолютный
                                        счетчик)      счетчик)
                                        программы     программы
 SUB2          SUB2                      00000002     0000085a
*SUB1          SUB1                   5  00000014     00000854
*SUB2          MAIN                  10  0000002C     0000082C
```

Эта команда выдает информацию о последовательности активных вызовов процедур.

Звездочка указывает на установленные модули.

1.76. Команда SHOW DEFINE

Команда SHOW DEFINE показывает, какая установка команды DEFINE (/ADDRESS, /COMMAND или /VALUE) была определена предыдущей командой SET DEFINE. (для распознавания символического имени, определенного командой DEFINE, а также установки этой команды надо пользоваться командой SHOW SYMBOL/DEFINED).

Формат:

```
SHOW DEFINE
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW DEFINE
DEFINE ADDRESS
DBG>
```

Команда SHOW DEFINE показывает, что команда DEFINE устанавлена для определения с помощью адреса.

1.77. Команда SHOW DISPLAY

Команда SHOW DISPLAY показывает все действующие определения экранных кадров, перечисляя имя каждого кадра, его максимальный размер, экранное окно и тип кадра (включая список команд отладчика).

Формат:

```
SHOW DISPLAY
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW DISPLAY
DISPLAY SRC AT H1, SIZE=50
     KIND = SOURCE (EXAMINE/SOURCE .%SOURCE_SCOPE\%PC)
```

```
DISPLAY INST AT H1, SIZE=50, REMOVED
     KIND = INSTRUCTION (EXAMINE/INSTRUCTION .0\%PC)
DISPLAY REG AT Q3, SIZE=4, REMOVED, KIND = REGISTER
DISPLAY OUT AT H2, SIZE=100, KIND = NORMAL
DISPLAY SRC2 AT Q3, SIZE=4, KIND = SOURCE
```

Команда SHOW DISPLAY перечисляет все имеющиеся экранные кадры. Они включают четыре предопределенных отладчиком кадра (SRC, INST, REG и OUT) и определенный пользователем кадр SRC2. Кадры INST и REG являются удаленными с экрана: чтобы выдать их на экран, следует использовать команду DISPLAY.

1.78. Команда SHOW EXIT_HANDLERS

Команда SHOW EXIT_HANDLERS идентифицирует обработчики выхода, заявленные в программе пользователя.

Формат:

```
SHOW EXIT_HANDLERS
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Команда SHOW EXIT_HANDLERS идентифицирует обработчики выхода, заявленные в программе пользователя. Подпрограммы обработчиков выхода выдаются в том порядке, в каком они будут вызываться (т.е. "последним пришел, первым обслужен"). Если возможно, имя подпрограммы изображается симво-

лически. Если нет, выдается его адрес. Обработчики выхода отладчика не выдаются.

Пример.

```
DBG>SHOW EXIT_HANDLERS
EXIT HANDLER AT STACKS\CLEANUP
```

Команда SHOW EXIT_HANDLERS идентифицирует подпрограмму обработчика выхода CLEANUP, которая заявлена в модуле STACKS.

1.79. Команда SHOW KEY

Команда SHOW KEY заставляет отладчик выдать определения клавиш, созданные командой DEFINE/KEY. Перед этим необходимо установить режим работы с малой клавиатурой.

Формат:

```
SHOW KEY [имя_клавиши]
```

Параметр:

Имя_клавиши - указывает имя клавиши, чье определение необходимо посмотреть.

Квалификаторы:

/ALL - заставляет отладчик выдать все определения клавиш для указанных состояний. Если используется квалификатор /ALL, задавать имя клавиши не требуется. Если состояние не задано, выдаются все определения клавиш для текущего состояния. Для указания одного или более состояний используется квалификатор /STATE.

/BRIEF - выдается только определение клавиши. По умолчанию можно увидеть все квалификаторы, связанные с опреде-

лением клавиши, включая любое указанное состояние, если только не используется квалификатор /BRIEF.

/DIRECTORY - выдает имена всех состояний, для которых были определены клавиши.

/[NO]STATE=(имя_состояния[,...]) - определяет имя состояния, для которого должны выдаваться указанные определения клавиш. Если указываются два или более имени, необходимо разделить их запятыми и заключить список в скобки. Имена состояний могут быть представлены любой соответствующей буквенно-цифровой строкой.

Если опускается квалификатор /STATE или используется квалификатор /NOSTATE, отладчик выдает определения клавиши для текущего состояния.

Примеры:

1. DBG>SET MODE KEYPAD

DBG>SHOW KEY/ALL

На этом примере показано, как получить все действующие определенния клавиш.

2. DBG>SET MODE KEYPAD

DBG>SHOW KEY KP8

DEFAULT KEYPAD DEFINITIONS

KP8 = "SCROLL/UP" (NOECHO,TERMINATE,NOLOCK)

На этом примере показано, как получить определение клавиши 8.

3. DBG>SHOW KEY/BRIEF KP8

DEFAULT KEYPAD DEFINITIONS:

KP8="SCROLL/UP"

Этот пример показывает, как получить определение клавиши 8 без каких-либо связанных с ним квалификаторов или

состояний.

4. DBG>SHOW KEY/DIRECTORY

GOLD

DEFAULT

BLUE

По этой команде выдаются имена состояний, для которых были определены клавиши.

1.80. Команда SHOW LANGUAGE

Команда SHOW LANGUAGE вызывает воспроизведение отладчиком текущего языка программирования.

Текущим языком программирования является тот язык, который был установлен последней командой SET LANGUAGE, или язык, установленный при запуске отладчика.

Формат:

SHOW LANGUAGE

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

DBG>SH. LANG

LANGUAGE: BASIC

(язык: бэйсик)

Эта команда выдает имя текущего языка бэйсик.

1.81. Команда SHOW LOG

Команда SHOW LOG вызывает воспроизведение отладчиком имени текущего файла регистрации, а также факта записи отладчиком в этот файл.

Текущим файлом регистрации является тот файл, который установлен командой SET LOG или файл регистрации по умолчанию DEBUG.LOG.

Формат:

```
SHOW LOG
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW LOG
NOT LOGGING TO DEBUG.LOG
```

Эта команда выдает имя текущего файла регистрации DEBUG.LOG (файл регистрации по умолчанию) и сообщает, что отладчик не ведет в него запись.

1.82. Команда SHOW MARGIN

Команда SHOW MARGIN вызывает воспроизведение текущих установок границ для кадров исходного текста.

Установка границ выполняется с помощью команды SET MARGIN. По умолчанию отладчик устанавливает левую границу на 1, а правую на 255.

Формат:

SHOW MARGIN

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Примеры:

1. DBG>SHOW MARGIN

LEFT MARGIN: 1, RIGHT MARGIN: 225

Эта команда выдает установку границ по умолчанию от 1 до 225.

2. DBG>SET MARGIN 50

DBG>SHOW MARGIN

LEFT MARGIN:1, RIGHT MARGIN: 50

Эта команда отображает установку левой границы по умолчанию (1) и модифицированную установку правой границы (50).

3. DBG>SET MARGIN 10:60

DBG>SHOW MARGIN

LEFT MARGIN: 10, RIGHT MARGIN: 60

По этой команде отображаются измененные установки левой и правой границ (10 и 60).

1.83. Команда SHOW MAX_SOURCE_FILES

Команда SHOW MAX_SOURCE_FILES обеспечивает воспроизведение максимального числа исходных файлов, которые могут поддерживаться отладчиком открытыми одновременно.

Максимальное число исходных файлов, которые отладчик может держать открытыми одновременно, может быть определено с помощью команды SET MAX_SOURCE_FILES или может иметь значение по умолчанию 5.

Формат:

```
SHOW MAX_SOURCE_FILES
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW MAX_SOURCE_FILES
MAX_SOURCE_FILES: 5
```

Эта команда показывает, что отладчик может иметь максимум 5 открытых исходных файлов одновременно.

1.84. Команда SHOW MODE

Команда SHOW MODE обеспечивает воспроизведение отладчиком текущих режимов (экранный или нет, динамический или нет и т. д.) и основания системы счисления.

Текущими режимами являются режимы, которые были установлены последней командой SET MODE, или режимы по умолчанию, связанные с текущим языком программирования.

Формат:

SHOW MODE .

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW MODE
MODES: SYMBOLIC, LINE, D_FLOAT, SCREEN, SCROLL, KEYPAD,
       DYNAMIC
INPUT RADIX   DECIMAL
OUTPUT RADIX: DECIMAL
```

Эта команда выдает текущие режимы и текущие основания входной и выходной информации.

1.85. Команда SHOW MODULE

Команда SHOW MODULE вызывает воспроизведение отладчиком следующей информации относительно каждого модуля программы пользователя:

- имя модуля;
- язык программирования, на котором написан этот модуль;
- содержит или нет таблица RST информацию относительно символических имен данного модуля;
- пространство (в байтах), требуемое в таблице RST для символических имен данного модуля;
- общее число модулей, выбранных для отображения;
- число неиспользованных байтов в пространстве,

которое отведено для таблицы RST.

Формат:

SHOW MODULE [имя_модуля[,...]]

Параметры:

Имя_модуля - указывает один или несколько модулей, информация о которых должна быть отображена на экране. Можно использовать символ обобщения (*). Если имя модуля не указывается, отображается информация о всех модулях программы. Информация о системных модулях выдается только при указании квалификатора /SHARE.

Квалификаторы:

/[NO]SHARE - управляет выдачей отладчиком информации о разделяемых образах, скомпонованных с программой пользователя, но являющихся внешними по отношению к ней. Это, в первую очередь, образы исполнительной библиотеки, а также любые разделяемые образы, вызываемые программой пользователя. По умолчанию (/NOSHARE), никакие системные разделяемые образы не включаются в выходную информацию данной команды.

Пример.

```
DBG> SHOW MODULE FOO,MAIN,SUB*
MODULE NAME    SYMBOLS              LANGUAGE    SIZE
FOO            да                   MACRO       432
MAIN           нет                  FORTRAN     280
SUB1           нет                  FORTRAN     164
SUB2           нет                  FORTRAN     204
всего модулей:4.                    Оставшийся размер: 60720.
```

Эта команда выдает информацию о модулях FOO и MAIN и всех модулях, имеющих префикс SUB.

1.86. Команда SHOW OUTPUT

Команда SHOW OUTPUT вызывает воспроизведение отладчиком текущей формы выдачи его выходной информации.

Формат:

SHOW OUTPUT

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Текущая конфигурация вывода отражает следующие действия отладчика:

- отображает ли он входную командную строку при выполнении командных процедур и последовательностей команд DO;
- выдает ли выходную информацию на терминал;
- записывает ли выходную информацию в файл регистрации.

Пример.

```
DBG>SHOW OUTPUT
OUTPUT: NOVERIFY, TERMINAL, NOT LOGGING TO DEBUG.LOG
```

Эта команда показывает, что отладчик не выдает входную командную строку при выполнении командных процедур и последовательности команд DO, что выходная информация выдается на терминал и что эта информация не записывается в файл регистрации DEBUG.LOG.

1.87. Команда SHOW RADIX

Команда SHOW RADIX выдает текущие установки основания системы счисления для входной и выходной информации.

Формат:

SHOW RADIX

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW RADIX
INPUT RADIX   DECIMAL
OUTPUT RADIX: DECIMAL
```

Команда SHOW RADIX отображает, что для входной и выходной информации действует десятичное основание системы счисления.

1.88. Команда SHOW SCOPE

Команда SHOW SCOPE вызывает воспроизведение отладчиком текущего списка поиска диапазона, т.е. списка поиска диапазона, который установлен последней командой SET SCOPE.

Формат:

SHOW SCOPE

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Текущий список поиска диапазона определяет одну или несколько программных ячеек (указанных именами пути или другими специальными символами), которые должны быть использованы при интерпретации символических имен, указанных без префиксов имени пути в командах отладчика.

Следует отметить, что если пользователь применил десятичное целое число в команде SET SCOPE для представления вызванной подпрограммы, то отладчик пытается воспроизвести имя подпрограммы, представленной этим десятичным числом.

Пример.

```
DBG>SHOW SCOPE
SCOPE: MAIN, SUB1, 0[=SUB2],\
```

Этот пример показывает, что поиск символического имени X (EXAMINE X) будет осуществляться сначала в MAIN, затем в подпрограмме SUB1, после чего в подпрограмме, содержащей текущий программный счетчик, и, наконец, как глобальная переменная.

1.89. Команда SHOW SEARCH

Команда SHOW SEARCH воспроизводит текущие параметры поиска.

Формат:

```
SHOW SEARCH
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Текущие параметры поиска устанавливаются либо командой SET SEARCH, либо являются величинами по умолчанию NEXT и STRING.

Параметры поиска определяют:

1) будет ли отладчик искать все местоположения (ALL) цепочки или только следующее (NEXT);

2) будет ли отладчик воспроизводить все местоположения цепочки (STRING) или только те, в которых эта цепочка не ограничена с любой стороны символом, который может быть частью идентификатора в текущем языке программирования (IDENTIFIER).

Пример.

```
DBG>SHOW SEARCH
SEARCH SETTINGS: SEARCH FOR NEXT OCCURENCE, AS A STRING
DBG>SET SEARCH IDENT
DBG>SHOW SEARCH
SEARCH SETTINGS: SEARCH FOR NEXT OCCURENCE,
     AS AN IDENTIFIER
DBG>SET SEARCH ALL
DBG>SHOW SEARCH
SEARCH SETTINGS: SEARCH FOR ALL OCCURENCE,
     AS AN IDENTIFIER
```

Первая команда SHOW SEARCH выдает параметры поиска по

умолчанию. По умолчанию отладчик идет и выдает ближайшее следующее местоположение заданной цепочки символов.

Вторая команда SHOW SEARCH показывает, что отладчик будет искать следующее появление цепочки, но выдает строку только в том случае, если она не ограничена с одной стороны знаком, который может быть идентификатором в текущем языке.

Третья команда SHOW SEARCH показывает, что отладчик будет искать все местоположения цепочки, но выдает только строки, не ограниченные с одной стороны знаком, который может быть идентификатором в текущем языке.

1.90. Команда SHOW SELECT

Команда SHOW SELECT выдает информацию о выбранных в текущий момент кадрах для каждого из четырех атрибутов: выходной информации, прокручиваемом, исходного текста и команд ассемблера, назначенных в качестве текущих кадров.

Формат:

SHOW SELECT

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Четыре атрибута кадров имеют следующие свойства:

- кадр, имеющий атрибут выходной информации, отображает нормальный выход отладчика;

- кадр, имеющий атрибут исходного текста

- кадр, имеющий атрибут исходного текста, выдает исходный код и корректируется при вводе команды TYPE или EXAMINE/SOURCE;

- кадр, имеющий атрибут команд ассемблера, выдает команды ассемблера и корректируется при вводе команды EXAMINE/INSTRUCTION;

- кадр с атрибутом прокрутки, прокручивается при вводе команд SCROLL.

Пример.

```
DBG>SHOW SELECT
DISPLAY SELECTIONS:
     OUTPUT = OUT
     SCROLL = SRC
     SOURCE = SRC
     INSTRUCTION = NONE
```

Команда SHOW SELECT показывает выбранные в текущий момент для каждого из четырех атрибутов (выходной информации, прокрутки, исходного текста и команд ассемблера) кадры.

1.91. Команда SHOW SOURCE

Команда SHOW SOURCE выдает текущий список обследуемых каталогов.

Формат:

```
SHOW SOURCE
```

Параметры.

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Команда SET SOURCE/MODULE=имя_модуля устанавливает список обследуемых каталогов для конкретного модуля. Команда SET SOURCE определяет список обследуемых каталогов для всех модулей, которые явно не указаны в команде SET SOURCE/MODULE=имя_модуля.

Если список каталогов не был установлен с помощью команд SET SOURCE или SET SOURCE/MODULE=имя_модуля, то команда SHOW SOURCE показывает, что в данное время такого списка нет. В таком случае отладчик предполагает, что каждый исходный файл находится в том каталоге, в котором он был во время компилирования.

Пример.

```
DBG>SHOW SOURCE
NO DIRECTORY SEARCH LIST IN EFFECT
DBG>SET SOURCE [MARY],[JOHN]
DBG>SHOW SOURCE
SOURCE DIRECTORY SEARCH LIST FOR ALL MODULES:
       [MARY]
       [JOHN]
```

На этом примере показано, что отладчик ведет поиск исходных файлов в двух каталогах: [MARY] и [JOHN].

1.92. Команда SHOW STEP

Команда SHOW STEP вызывает воспроизведение отладчиком текущих условий пошагового выполнения.

Формат:

SHOW STEP

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Описание.

Текущие условия пошагового выполнения определяют следующие действия отладчика:

1) выполнение шагов построчно или покомандно;

2) выполнение шагов "в" или "поверх" подпрограммы в программе пользователя;

3) выполнение шагов "в" или "поверх" подпрограммы в системном пространстве;

4) выдачу строк исходного кода при выполнении шага.

Текущими условиями пошагового режима работы являются условия, установленные с помощью последней команды SET STEP, или условие пошагового режима по умолчанию, определяемые текущим языком программирования.

Пример.

```
DBG>SET STEP NOSYSTEM, INSTRUCTION, OVER, NOSOURCE
DBG>SHOW STEP
STEP TYPE: NOSYSTEM, NOSOURCE, NOSILENT, CALLS,
     BY INSTRUCTION, OVER ROUTINE
```

Эта команда показывает, что отладчик будет:

1) выполнять шаги "поверх" вызванных подпрограмм в системном пространстве;

2) выполнять отдельную команду в ячейке, где выполнение было приостановлено последний раз;

3) выполнять шаги "поверх" вызванных подпрограмм в пространстве программы пользователя;

4) не будет выдавать строки исходного кода при выполнении шагов.

1.93. Команда SHOW SYMBOL

Команда SHOW SYMBOL вызывает выдачу отладчиком всех символических имен из таблицы RST, совпадающих с данным или имеющих заданную форму, а так же выдачу всех имен, заявленных в данной лексической единице (такой, как модуль, подпрограмма или лексический блок), или перечисление дополнительной информации (такой, как типы данных или спецификацию адресов) для символических имен.

Формат:

SHOW SYMBOL спецификация_имени[IN диапазон[,диапазон...]]

Параметры:

Спецификация_имени - указывает требуемое символическое имя. Этот параметр может состоять просто из одного имени или имени, включающего символы *, которые могут заменять любое количество символов имени. Символическим именем является отдельный идентификатор или метка в виде %LABEL N, где "N" - целое число. Сложные имена (такие как RECORD.FIELD или ARRAY[1,2]) - недопустимы.

Диапазон - указывает имя модуля, подпрограммы, лексического блока или числовой диапазон. Он имеет тот же синтаксис, что и спецификация области действия в команде SET SCOPE, и может включать определение имени пути. Все указываемые диапазоны действия должны находиться в установленных модулях.

Команда SHOW SYMBOL выдает только те имена из RST, которые соответствуют заданному имени и об'явлены в лексическом об'екте, указанном параметром диапазона. Об'явление в пределах лексического об'екта означает, что символическое имя должно быть об'явлено в этом об'екте или в любом вложенном в него лексическом об'екте. Если параметр диапазона опущен, то обследуются все установленные модули и таблица GST (таблица глобальных символических имен) для поиска имени с заданной спецификацией.

Квалификаторы:

/ADDRESS - отображается спецификация адреса для каждого выбранного символического имени. Адресная спецификация является методом вычисления адреса имени. Это может быть просто виртуальным адресом имени, но так же может включать косвенность или смещение от значения регистра. Некоторые имена имеют слишком сложные адресные спецификации, чтобы представить их в более или менее понятной форме. Такие адресные спецификации обозначаются просто как "сложные адресные спецификации".

/DEFINED - отображаются символические имена, определенные командой DEFINE.

/DIRECT - выдаются только те символические имена, которые об'явлены непосредственно в параметре диапазона.

имена, об'явленные в лексических об'ектах, которые вложены в область действия, указанную параметрами диапазона, не выдаются.

/LOCAL - выдается символические имена, определенные квалификатором /LOCAL.

/TYPE - выдается информация о типе данных для каждого выбранного символического имени.

Описание.

Основным назначением команды SHOW SYMBOL является показ информации, которую имеет отладчик о данном символическом имени. Эта информация может отличаться от информации, которую имел компилятор, или даже от информации, имеющейся в исходном тексте. Именно поэтому она полезна для понимания того, почему отладчик действует так, а не иначе.

Примеры:

1. DBG>SHOW SYMBOL I

DATA FORARRAY\I

Эта команда показывает, что имя I находится в модуле FORARRAY.

2. DBG>SHOW SYMBOL INTARRAY1

DATA FORARRAY\INARRAY1

Эта команда показывает, что имя INTARRAY1 размещено в модуле FORARRAY.

3. DBG>SHOW SYMBOL/ADDRESS INARRAY1

DATA FORARRAY\INTARRAY1

DESCRIPTOR ADDRESS: 0009DE8B

Эта команда показывает, что имя INTARRAY1 размещено в модуле FORARRAY и имеет виртуальный адрес 0009DE8B.

4. DBG>SHOW SYMBOL *TARR*

DATA FORARRAY\INTARRAY1

DATA FORARRAY\INTARRAY2

DATA FORARRAY\INTARRAY3

Эта команда выдает все имена, содержащие строку "IARR", и то, что они размещены в модуле FORARRAY.

5. DBG>SHOW SYMBOL/TYPE/ADDRESS *

Эта команда выдает всю информацию о всех символических именах.

1.94. Команда SHOW TERMINAL

Команда SHOW TERMINAL показывает текущую высоту экрана терминала и его ширину, используемые для форматирования выходной информации.

Формат:

SHOW TERMINAL

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW TERMINAL
TERMINAL WIDTH: 80
          PAGE: 24
```

Эта команда выдает ширину и высоту текущего экрана терминала (80 позиций и 24 строки соответственно).

1.95. Команда SHOW TRACE

Команда SHOW TRACE вызывает воспроизведение отладчиком точек отслеживания, установленных с помощью команды SET TRACE.

Формат:

```
SHOW TRACE
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG> SHOW TRACE
TRACEPOINT AT CALL\MULT
TRACING /CALL INSTRUCTIONS: CALLS, CALLG, BSBW, BSBB, JSB,
RSB AND RET
```

Когда отладчик встречает ячейки CALLS, CALLG, BSBW, BSBB, JSB, RSB и RET, он приостанавливает выполнение программы, сообщает о достижении точки отслеживания и возобновляет выполнение программы.

1.96. Команда SHOW TYPE

Команда SHOW TYPE вызывает воспроизведение отладчиком текущего типа по умолчанию или, если указан командный квалификатор /OVERRIDE, текущий тип, отменяющий другие типы.

Формат:

SHOW TYPE .

Параметры:

Отсутствуют.

Квалификаторы:

/OVERRIDE - указывает на отображение текущего типа, отменяющего другие.

Пример.

```
DBG>SHOW TYPE
TYPE: LONG INTEGER
```

Эта команда выдает, что текущим типом по умолчанию является тип целого длинного слова. Другими словами, отладчик интерпретирует и выдает об'екты программы как целые числа длиной в два слова (пока не будет указан другой тип).

1.97. Команда SHOW WATCH

Команда SHOW WATCH вызывает воспроизведение отладчиком ячеек, в которых установлены точки просмотра с помощью команды SET WATCH, а также сообщает длину в байтах каждой просматриваемой ячейки.

Формат:

SHOW WATCH

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW WATCH
WATCHPOINT AT MAIN\ALPHA FOR 4. BYTES
WATCHPOINT AT SUB2\TABLE+20 FOR 4. BYTES
```

Эта команда выдает две точки просмотра: одну в ячейке MAIN\ALPHA, другую в ячейке SUB\TABLE+20.

1.98. Команда SHOW WINDOW

Команда SHOW WINDOW выдает список всех определений экранных окон. Отображается имя и местоположение на экране каждого существующего окна.

Формат:

```
SHOW WINDOW
```

Параметры:

Отсутствуют.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SHOW WINDOW
```

Эта команда выдает имя и положение на экране всех имеющихся окон.

1.99. Команда SPAWN

Команда SPAWN обеспечивает способ временного выхода с уровня отладчика на уровень диалогового командного языка DCL.

Формат:

SPAWN [DCL_команда]

Параметр:

DCL_команда - указывает команду языка DCL. Если после команды SPAWN задается команда DCL, создается подпроцесс, который выполняет указанную команду DCL. Управление возвращается отладчику после окончания выполнения команды DCL.

Если параметр DCL_команда опущен, команда SPAWN создает подпроцесс и присоединяет терминал к этому подпроцессу, с которого затем можно вводить команды DCL. Выход из этого подпроцесса или переподключение к основному процессу позволяют продолжить сеанс отладки.

Если указываемая пользователем команда DCL имеет точку с запятой, необходимо заключить всю команду в кавычки. Иначе отладчик будет интерпретировать точку с запятой как разделитель команд отладчика. Для включения кавычек в командную строку необходимо указывать две последовательные кавычки.

Квалификатор:

/NOWAIT - позволяет вводить команды отладчика во время работы подпроцесса. Если используется этот квалификатор, надо задавать команду DCL: команда DCL выполняется в подпроцессе. После ввода команды SPAWN с квалификатором /NOWAIT сразу выдается запрос отладчика DBG>, так что можно продол-

жать отладку без ожидания окончания подпроцесса. Когда подпроцесс завершается, на терминал выдается сообщение. С помощью квалификатора /NOWAIT можно, например, скомпилировать исходную программу, продолжая процесс отладки.

Описание.

Действие команды SPAWN отладчика точно такое же, что команды SPAWN языка DCL. Можно редактировать файлы, распечатывать файлы, читать почту или решать задачи, обычно выполняемые на терминале. Все это делается без окончания процесса отладки.

Кроме того, можно подключить команду SPAWN языка DCL следующим образом:

SPAWN SPAWN/OUTPUT=OUTFILE.LOG OUTFILE

Интерпретатор DCL сам берет вторую команду SPAWN и интерпретирует квалификатор.

1.100. Команда STEP

Команда STEP вызывает выполнение отладчиком программы пользователя построчно или покомандно, в зависимости от текущих условий шага.

Формат:

STEP [N]

Параметр:

N - определяет число строк или программных команд, которые должны быть выполнены по команде STEP. Если N не задано, то отладчик выполняет одну строку или одну команду программы. Параметр N всегда интерпретируется как десятичное целое число.

Квалификаторы:

/BRANCH - вызывает выполнение шага до следующей команды ветвления. STEP/BRANCH выполняет то же, что и SET BREAK/BRANCH;GO за исключением того, что она не создает постоянной точки приостанова.

/EXCEPTION - вызывает выполнение шага до следующего условия исключительной ситуации. STEP/EXCEPTION выполняет то же, что и SET BREAK/EXCEPTION;GO, но не создает постоянной точки приостанова.

/CALL - вызывает выполнение шага до следующей команды вызова или возврата. STEP/CALL выполняет то же, что и SET BREAK/CALL;GO, но не создает постоянной точки приостанова.

/INSTRUCTION - вызывает выполнение одной программной команды в ячейке, на которой последний раз было приостановлено выполнение;

/INSTRUCTION=(код_операций[,...]) - вызывает выполнение программной команды, чей код операции дается одной из указанных мнемоник. Можно задавать несколько мнемоник кодов операций, разделенных запятыми, заключив весь список в скобки.

/INTO - вызывает вход в шаговом режиме "В" вызываемые подпрограммы в пространстве программы пользователя (и "В" вызываемые подпрограммы в системном пространстве в случае определения квалификатора /SYSTEM). При выполнении шагов отладчик не делает различий между кодом в пределах вызываемой подпрограммы и кодом вне вызываемой подпрограммы.

/LINE - один шаг вызывает выполнение всего кода программы пользователя между ячейкой, где последний раз было приостановлено выполнение, и началом следующей строки.

/OVER - проход в пошаговом режиме "поверх" вызываемых

подпрограмм как в пространстве программы пользователя, так и в системном пространстве. При этом отладчик рассматривает любой код, выполняемый в результате инструкции CALL, от CALL и до соответствующей инструкции RETURN как часть одного шага (STEP).

/RETURN - указывает отладчику выполнить шаг до следующей команды возврата, выполняемой текущей подпрограммой. Этот квалификатор действителен только для подпрограмм CALLS и CALLG.

/[NO]SILENT - управляет выдачей сообщения "шаг выполнен", связанного с командой STEP.

/SOURCE - вызывает воспроизведение исходной строки, соответствующей инструкции или строке программы, которая следует за последней инструкцией, выполненой в результате команды STEP.

/NOSOURCE - указывает, что исходную строку воспроизводить не надо. Команда STEP/NOSOURCE используется в тех случаях, когда действует параметр SOURCE (в результате предыдущей команды SET STEP SOURCE), но воспроизведение исходного кода с этой конкретной командой STEP не требуется.

/SYSTEM - заход в пошаговом режиме "В" вызываемые подпрограммы в системном пространстве при условии того, что указан также и квалификатор /INTO. В этом случае при выполнении шагов отладчик не делает различий между кодом в пределах вызываемой подпрограммы и кодом самой программы. Таким образом, выполнение команды STEP может привести к приостановке выполнения в системном пространстве.

/NOSYSTEM - проход в пошаговом режиме "поверх" вызываемых подпрограмм в системном пространстве. В этом случае отладчик рассматривает любой код, выполняемый от команды

CALL до соответствующей команды RETURN, в качестве части одного шага (STEP).

Описание.

Когда пользователь выдает данную команду, отладчик выполняет следующие действия:

1) выполняет набор программных команд (или одну команду);

2) сообщает команду или строку, которая следует за последней выполненой программной командой;

3) выдает исходную строку, соответствующую строке или инструкции, которая следует за последней выполненной инструкцией, только в том случае, если действует параметр SOURCE (в результате предыдущей команды STEP/SOURCE или SET STEP SOURCE) и имеются исходные строки;

4) выдает свой запрос (DBG>).

Если пользователь указывает параметр N (целое десятичное), то отладчик выполняет N инструкций или строк.

Каждый язык программирования устанавливает условия пошагового режима по умолчанию. Когда пользователь изменяет текущий язык программирования, то условия выполнения в пошаговом режиме устанавливаются в соответствии с определениями нового языка. Пользователь может изменить эти условия с помощью команды SET STEP.

Можно отменить текущие условия выполнения в пошаговом режиме путем определения другого условия в качестве квалификатора команды STEP.

Примеры:

1. DBG>STEP
STEPPED TO MAIN\MAIN+14: MOVL 222,R0

По этой команде делается шаг до ячейки MAIN\MAIN+14 и выполняется команда MOVL.

2. DBG>S/LINE

STEPPED TO MAIN\MAIN+30: ADDL R0,R1,R3

По этой команде делается шаг до MAIN\MAIN+30 и выполняются все команды от ячейки MAIN\MAIN+14 до ячейки MAIN\MAIN+30.

3. DBG>S/INTO

STEPPED TO ROUTINE SUB1: MOVAL L^00000500,R11

По этой команде отладчик входит в подпрограмму до ячейки SUB1 и выполняет команду MOVAL.

1.101. Команда SYMBOLIZE

Команда SYMBOLIZE позволяет преобразовывать виртуальный адрес в символическое представление.

Формат:

SYMBOLIZE адресное_выражение [,адресное_выражение...]

Параметр:

Адресное_выражение - указывает преобразуемый адрес. Если адрес является постоянным, он будет изображен символически как ближайшее предшествующее символическое имя плюс смещение. Если это так же адрес кода операции, то в символизацию будет включен номер строки, если можно найти номер исходной строки, в которой содержится этот адрес.

Если адрес является адресом регистра, тогда отладчик выдает все символические имена из всех установленных модулей, которые связаны с этим регистром. Выдается полное имя пути каждого такого имени, так что становится ясной область

действия символического имени. Также выдается само имя регистра ("%R5", например).

Если адресом является ячейка стека в кадре вызова подпрограммы установленного модуля, отладчик будет отыскивать все символические имена в этой подпрограмме, чьи адреса относятся к указателю кадра (FP) или указателю стека (SP). Ближайшее предыдущее символическое имя плюс сдвиг выдаются как символизация адреса. Имя, чья спецификация адреса слишком сложна, в этом поиске игнорируется.

Если отладчик не может найти символизацию для адреса, выдается сообщение об этом.

Квалификаторы:

Отсутствуют.

Пример.

```
DBG>SYMBOLIZE %R5
PROG\SUB1\X
PROG\SUB2\Y
```

Эта команда показывает, что локальная переменная X подпрограммы SUB1 и локальная переменная Y подпрограммы SUB2 размещены в регистре 5.

1.102. Команда TYPE

Команда TYPE воспроизводит строку (строки) исходного кода программы по ее (их) номеру (номерам).

Формат:

```
TYPE [[Имя_модуля\]номер_строки[:номер_строки]-
    [,[Имя_модуля\]номер_строки[:номер_строки]...]] .
```

Параметры:

Номер_строки – указывает любой номер, сгенерированный компилятором для метки оператора или операторов исходного языка программирования.

Квалификаторы:

Отсутствуют.

Описание.

Номера строк, которые используются отладчиком для идентификации строк исходного кода, генерируются компилятором и приводятся в распечатке (листинге) программы.

Если пользователь указывает один номер строки, то отладчик воспроизводит исходный код, который соответствует этому номеру строки.

Если пользователь указывает список номеров строк, разделяя их запятыми, то отладчик воспроизводит исходный код, который соответствует каждому из номеров строк, приведенных пользователем.

Если пользователь указывает диапазон номеров строк, разделяя начальный и конечный номера строк в этом диапазоне двоеточием, то отладчик воспроизводит исходный код, который соответствует указанному диапазону номеров строк.

Пользователь может считывать все операторы исходного языка программирования своей программы путем определения диапазона номеров строк, начиная со строки с номером 1 и кончая номером, который равен или больше максимального номера строки в распечатке этой программы.

После воспроизведения одной строки исходного кода пользователь может воспроизвести следующую строку путем выдачи команды TYPE без номера строки, т.е. Путем выдачи

команды TYPE, а затем нажав клавишу RETURN. Затем можно воспроизвести следующую строку и последующие строки путем повторения этой последовательности, выполняя, таким образом, построчное считывание своей исходной программы.

Пользователь может определить имя модуля вместе с номером (или номерами) строки для указания того, что эта строка (или строки) находится в данном модуле. В этом случае пользователь должен ввести имя модуля, обратную косую черту (\) и номер (или номера) строки (строк) без каких-либо пробелов.

Если вместе с номером строки (или с номерами строк) пользователь не приводит имя модуля, то отладчик использует текущую установку диапазона для определения используемого модуля. В этом случае текущим диапазоном является либо первый модуль, указанный в команде SET SCOPE, либо, если эта команда не была ранее использована, модуль, содержащий текущий счетчик программы (PC).

Примеры:

```
1. DBG>TYPE 160
MODULE COBOLTEST
     160: START-IT-PARA.
DBG>TYPE
MODULE COBOLTEST
     161:          MOVE SC1 TO ES0.
```

Первая команда воспроизводит строку 160 исходного кода, вторая - следующую.

```
2. DBG>T 160:163
MODULE COBOLTEST
     160: START-IT-PARA.
```

```
     161:          MOVE SC1 TO ES0.
     162:          DISPLAY ES0.
     163:          MOVE SC1 TO ES1.
```

Эта команда выдает строки 160-163 исходного кода.

```
3. DBG>TYPE COBOLTEST\160,22:24
MODULE COBOLTEST
     160: START-IT-PARA.
MODULE COBOLTEST
     22: 02    SC2V2   PIC S99V99   COMP VALUE  22.33.
     23: 02    SC2V2N  PIC S99V99   COMP VALUE -22.33.
     24: 02    CPP2    PIC PP99     COMP VALUE  0.0012.
```

Эта команда выдает строку 160 и строки с 22 по 24 модуля COBOLTEST.

1.103. Команда UNDEFINE

Команда UNDEFINE удаляет определение указанного символического имени из таблицы глобальных или локальных символических имен. Глобальным называется символическое имя, определенное командой DEFINE без квалификатора /LOCAL. Локальным является символическое имя, определенное в косвенном командном файле отладчика с помощью команды DEFINE/LOCAL, так что его определение ограничено этим командным файлом. Команда UNDEFINE идентична команде DELETE.

Формат:

```
UNDEFINE [символическое_имя[,...]]
```

Параметр:

Символическое_имя - указывает имя, чье определение должно быть удалено из таблицы локальных или глобальных символических имен. Если используется /ALL, не требуется указывать какой-либо параметр. Если используется /LOCAL, указываемое символическое имя должно быть ранее определено командой DEFINE/LOCAL. Если /LOCAL не задается, данное имя должно быть ранее определено командой DEFINE без квалификатора /LOCAL.

Квалификаторы:

/ALL - удаляет все определения глобальных символических имен. Если при этом указывается /LOCAL, удаляются все определения локальных символических имен, связанные с текущим косвенным командным файлом (но не с определениями глобальных имен).

/LOCAL - удаляет определения заданных символических имен из текущего косвенного командного файла. Эти имена должны быть заранее определены командой DEFINE/LOCAL.

Примеры:

1. DBG>DEFINE X=INARR, Y=OUTARR
DBG>UNDEFINE X,Y

Команда DEFINE определяет X и Y как глобальные символические имена, соответствующие INARR и OUTARR. Команда UNDEFINE удаляет эти два определения из таблицы глобальных символических имен.

2. DBG>UNDEFINE/ALL/LOCAL

Команда UNDEFINE/ALL/LOCAL удаляет все определения локальных имен из текущего косвенного командного файла.

1.104. Команда UNDEFINE/KEY

Команда UNDEFINE/KEY уничтожает определение клавиши, установленное командой DEFINE/KEY.

Формат:

UNDEFINE/KEY [имя_клавиши]

Параметр:

Имя_клавиши - указывает имя клавиши, чье определение надо исключить. Этот параметр не используется, если указывается квалификатор /ALL.

Квалификаторы:

/ALL - исключаются все определения клавиш для заданного состояния. При его использовании имя клавиши не указывается. Если состояние не задается, то стираются все определения клавиш для текущего состояния. Для указания состояния служит /STATE.

/[NO]LOG - управляет выдачей сообщения, указывающего, что указанные определения клавиш удалены. По умолчанию такое сообщение выдается(/LOG).

/[NO]STATE=(имя_состояния[,...]) - указывает имя состояния, для которого должны удаляться обозначенные определения клавиши. Если указывается только одно имя состояния, круглые скобки можно опустить. Именами состояний могут быть любые подходящие буквенно-цифровые строки.

Если квалификатор /STATE опускается или употребляется /NOSTATE, исключаются определения клавиш для текущего состояния.

Описание.

Чтобы исключить определения клавиши, нужно установить режим работы с малой клавиатурой. Команда UNDEFINE/KEY обладает тем же действием, что и команда DELETE/KEY.

Пример.

```
DBG>UNDEFINE/KEY/STATE=GOLD KP2
```

Команда UNDEFINE/KEY удаляет определение для клавиши KP2 в состоянии GOLD.

1.105. Команда WHILE

Команда WHILE позволяет выполнять списки команд отладчика до тех пор, пока какое-нибудь заданное языковое выражение не будет оценено как ложное (FALSE).

Формат:

```
WHILE логическое_выражение DO (список_команд) ,
```

Параметры:

Логическое_выражение - указывает языковое выражение, которое оценивается как булева величина (TRUE или FALSE) в текущем языке программирования.

Список_команд - указывает список команд отладчика. Команды должны отделяться точкой с запятой.

Квалификаторы:

Отсутствуют.

Описание.

Команда WHILE вычисляет языковое выражение как булево выражение. Если его значение - "истина" (как определено в текущем языке), выполняется список команд отладчика в пред-

ложении DO. В дальнейшем команда выполняет последовательность, определенную списком команд DO, заново оценивая булево выражение и выполняя список команд до тех пор, пока выражение не будет оценено как ложное (FALSE).

Если булево выражение является ложным, команда WHILE завершается.

Пример.

```
DBG>WHILE (X.EQ.0) DO (STEP/SILENT)
```

Эта команда заставляет отладчик продолжать пошаговое выполнение программы, пока x не перестанет быть равным 0.

2. Сообщения оператору

В процессе работы на терминал могут выдаваться сообщения, текст которых, а также ответные действия оператора приведены в [1].

Приложение 1

Дополнительная информация по отладчику

1. Поддержка отладчиком языка бэйсик

Информация по языку бэйсик приведена в табл. 1-3.

Таблица 1

Знаки операций в языковых выражениях бэйсика

Тип	Символ	Функция
Префикс	+	Плюс
префикс	-	минус
инфикс	+	сложение, соединение в строку
инфикс	-	вычитание
инфикс	*	умножение
инфикс	/	деление
инфикс	**	возведение в степень
инфикс	^	возведение в степень
инфикс	=	равно
инфикс	<>	не равно
инфикс	><	не равно
инфикс	>	больше, чем
инфикс	>=	больше, чем или равно
инфикс	=>	больше чем или равно
инфикс	<	меньше, чем
инфикс	<=	меньше чем или равно
инфикс	=<	меньше чем или равно

Продолжение табл. 1

Тип	Символ	Функция
Префикс	NOT	Логическое не с битами
инфикс	AND	логическое и с битами
инфикс	OR	логическое или с битами
инфикс	XOR	логическое исключающее или с битами
инфикс	IMP	логическая импликация с битами
инфикс	EQV	эквивалентность с битами

Таблица 2

Поддерживаемые конструкции в языковых и адресных выражениях для бэйсик

Символ	Конструкция
()	Индексация
	выбор компонента записи

Таблица 3

Поддерживаемые типы данных бэйсик

Тип в бэйсике	Имя типа для СМ 1700
BYTE	B - байтовое целое число
WORD	W - целое число длиной в слово
LONG	L - длинное целое
SINGLE	F - вещественное число формата F
DOUBLE	D - вещественное число формата D
GFLOAT	G - вещественное число формата G
HFLOAT	H - вещественное число формата H

Продолжение табл. 3

Тип в бэйсике	Имя типа для СМ 1700
DECIMAL	P - упакованное десятичное
STRING	T - текст кодов ASCII
RFA	нет
ARRAYS	нет
RECORS	нет

Примечания:

1. Если при вычислении выражения в бэйсике происходит переполнение, то это не обязательно приводит к переполнению при вычислении этого выражения отладчиком. Отладчик попытается получить корректный результат даже в том случае, когда правила бэйсик предусматривают переполнение. Эта разница особенно влияет на вычисления с десятичным основанием.

2. Константы бэйсика формата [основание]"числовая_строка"[тип] (как например, "12.34"GFLOAT) или N% (например, 25% для целого числа 25) отладчиком не поддерживаются.

2. Поддержка отладчиком языка блисс

Этот раздел содержит информацию о поддержки отладчиком языка блисс (табл. 4-6).

Таблица 4

Знаки операций в языковых выражениях блисс

Тип	Символ	Функция
Префикс		Косвенность
префикс	+	унарный плюс
префикс	-	унарный минус
инфикс	+	сложение
инфикс	-	вычитание
инфикс	*	умножение
инфикс	/	деление
инфикс	MOD	остаток
инфикс	^	сдвиг влево
инфикс	EQL	равно
инфикс	EQLU	равно
инфикс	EQLA	равно
инфикс	NEQ	неравно
инфикс	NEQU	неравно
инфикс	NEQA	неравно
инфикс	GTR	больше, чем
инфикс	GTRU	больше, чем без знака
инфикс	GTRA	больше, чем без знака
инфикс	GEQ	больше, чем или равно
инфикс	GEQU	больше, чем или равно без знака
инфикс	GEQA	больше, чем или равно без знака
инфикс	LSS	меньше чем
инфикс	LSSU	меньше чем без знака
инфикс	LSSA	меньше чем без знака
инфикс	LEQ	меньше чем или равно

Продолжение табл. 4

Тип	Символ	Функция
Инфикс	LEQU	Меньше чем или равно без знака
инфикс	LEQA	меньше или равно без знака
инфикс	NOT	логическое не для битов
инфикс	AND	логическое и для битов
инфикс	OR	логическое или для битов
инфикс	XOR	логическое исключающие или для битов
инфикс	EQV	логическая эквивалентность для битов

Таблица 5

Поддерживаемые конструкции в языковых и адресных выражениях языка блисс

Символ	Конструкция
[]	Индексация
[имя_поля]	выборка поля
<P,S,E>	выборка поля битов

Таблица 6

Поддерживаемые типы данных блисс

Тип в блисс	Имя типа для СМ 1700
BYTE	B - байтовое целое число
WORD	W - целое число размером в слово
LONG	L - целое число размером в длинное слово
BYTE UNSIGNED	BU - байт без знака
WORD UNSIGNED	WU - слово без знака

Продолжение табл. 6

Тип в блисс.	Имя типа для СМ 1700
LONG UNSIGNED	LU - длинное слово без знака
VECTOR	отсутствует
BITVECTOR	отсутствует
BLOCK	отсутствует
BLOCKVECTOR	отсутствует
REF VECTOR	отсутствует
REF BITVECTOR	отсутствует
REF BLOCK	отсутствует
REF BLOCKVECTOR	отсутствует

3. Поддержка отладчиком языка Си

Информация о поддержке языка Си приведена в табл. 7-9.

Таблица 7

Знаки операций в языковых выражениях Си

Тип	Символ	Функция
Префикс		Косвенность
префикс	&	адрес
префикс	SIZEOF	размер
префикс	-	унарный минус (отрицание)
инфикс	+	сложение
инфикс	-	вычитание
инфикс	*	умножение
инфикс	/	деление

Продолжение табл. 7

Тип	Символ	Функция
Инфикс	%	Остаток
инфикс	<<	сдвиг (смещение) влево
инфикс	>>	сдвиг (смещение) вправо
инфикс	==	равно
инфикс	!=	неравно
инфикс	>	больше чем
инфикс	>=	больше чем или равно
инфикс	<	меньше чем
инфикс	<=	меньше чем или равно
префикс	(знак тильды)	логическое не для битов
инфикс	&	логическое и для битов
инфикс	!	логическое или для битов
инфикс	^	логическое исключающее или для битов
префикс	!	логическое не
инфикс	&&	логическое и
инфикс	!!	логическое или

Таблица 8

Вспомогательные конструкции в языковых и адресных выражениях языка Си

Символ	Конструкция
[]	Индексация выборка компоненты структуры

Продолжение табл. 8

Символ	Конструкция
->	отличительный признак указателя

Таблица 9

Поддерживаемые типы данных языка Си

Тип в С	Имя типа для СМ 1700
INT	L - целое длинное слово
UNSIGNED INT	LU - длинное слово без знака
SHORT INT	W - целое число размером в слово
UNSIGNED SHORT	WU - слово без знака
INT	
CHAR	B - целое число длиной в байт
UNSIGNED CHAR	BU - байт без знака
FLOAT	F - вещественное число формата F
DOUBLE	D - вещественное число формата D
ENUM	отсутствует
STRUCT	отсутствует
UNION	отсутствует
POINTERS	отсутствует
ARRAYS	отсутствует

Примечания:

1. Имена символов чувствительны к регистру для языка си, что означает, что буквы верхнего и нижнего регистров рассматриваются как различные символы.

2. Поскольку восклицательный знак (!) является оператором в языке си, он не может применяться в качестве разде-

лителя комментария. Когда язык установлен на си, отладчик воспринимает /* как разделитель комментария. Комментарий продолжается до конца текущей строки. (соответствующий символ */ не нужен и не распознается). Чтобы использовать файлы регистрации отладчика как входные, отладчик все же распознает символ ! Как разделитель комментария, если он является первым непустым символом на строке.

3. Отладчик допускает префикс звездочку (*) как знак косвенности в выражениях языка Си и в адресных выражениях отладчика. В адресных выражениях префикс "*" идентичен префиксу . или "@", когда язык установлен на Си.

4. Отладчик не поддерживает ни один из операторов присваивания в языке с (или любом другом языке), чтобы предотвратить непреднамеренные модификации отлаживаемой программы. Поэтому такие операторы, как =, +=, -=, ++ и -- не распознаются отладчиком. Если надо изменить содержимое ячейки памяти, необходимо делать это явно с помощью команды DEPOSIT.

4. Поддержка отладчиком языка Кобол

Этот раздел включает информацию о средствах поддержки языка Кобол (табл.10-12).

Таблица 10

Знаки операций языка Кобол в языковых выражениях

Тип	Символ	Функция
Префикс	+	Унарный плюс

Продолжение табл. 10

Тип	Символ	Функция
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Деление
Инфикс	**	Возведение в степень
Инфикс	=	Равно
Инфикс	NOT =	Не равно
Инфикс	>	Больше чем
Инфикс	NOT <	Больше чем или равно
Инфикс	<	Меньше чем
Инфикс	NOT >	Меньше чем или равно
Инфикс	NOT	Логическое не
Инфикс	AND	Логическое и
Инфикс	OR	Логическое или

Таблица 11

Поддерживаемые конструкции в языковых и адресных выражениях для языка Кобол

Символ	Конструкция
()	Индексация
OF	Выборка компоненты записи
IN	Выборка компоненты записи

Таблица 12

Поддерживаемые типы данных для языка кобол

Тип в Коболе	Имя типа для СМ 1700
COMP	Целое длинное (L, LU)
COMP	Целое слово (W, WU)
COMP	Байтовое целое число (B, BU)
COMP	Целое квадрослово (Q, QU)
COMP-1	F - вещественное формата F
COMP-2	D - вещественное формата D
COMP-3	P - упакованное десятичное
INDEX	L - целое длинное
буквенно-числовой	T - текст ASCII
записи	T - текст ASCII
числовой без знака (NU)	?
головной разделительный знак (NL)	?
головной знак перфорации (NLO)	?
хвостовой разделительный знак (NR)	?
хвостовой знак перфорации (NRO)	?

Примечания:

1. Отладчик может отображать исходный текст, включенный в программу с помощью COPY или COPY REPLACING. Однако при использовании COPY REPLACING отладчик всегда отображает первоначальный исходный текст как появившийся до замены текста. Другими словами, показывается первоначальный исход-

ный файл вместо модифицированного исходногоо текста, сгенерированного с помощью COPY REPLACING.

2. Отладчик не может отображать первоначальные исходные строки, связанные с программой для секции REPORT. Можно получить исходные строки секции данных, связанные с REPORT, но не исходные строки, связанные с скомпилированным кодом, который генерирует сообщение.

5. Поддержка отладчиком языка Фортран

Этот раздел включает информацию о средствах фортрана, поддерживаемых отладчиком (табл. 13-16).

Таблица 13

Знаки операций языка фортран

Тип	Символ	Функция
Префикс	+	Унарный плюс
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Деление
Инфикс	**	Возведение в степень
Инфикс	//	Конкатенация
Инфикс	.EQ.	Равно
Инфикс	.NE.	Неравно
Инфикс	.GT.	Больше
Инфикс	.GE.	Больше или равно

Продолжение табл. 13

Тип	Символ	Функция
Инфикс	.LT.	Меньше чем
инфикс	.LE.	меньше или равно
Префикс	.NOT.	Логическое не
Инфикс	.AND.	Логическое и
Инфикс	.OR.	Логическое или
Инфикс	.XOR.	Исключающее или
Инфикс	.EQV.	Эквивалентность
Инфикс	.NEQV.	Исключающее или

Таблица 14

Поддерживаемые конструкции в языковых и адресных выражениях для языка Фортран

Символ	Конструкция
()	Индексация
	Выбор компоненты записи

Таблица 15

Поддерживаемые предопределенные символические имена для языка Фортран

Имя	Значение
.TRUE.	Логическое истина
.FALSE.	Логическое ложь

Таблица 16

Поддерживаемые типы данных фортрана

Тип в Фортране	Имя типа для СМ 1700
LOGICAL*1	BU - байт без знака
LOGICAL*2	WU - слово без знака
LOGICAL*4	LU - длинное слово без знака
INTEGER*2	W - целое число длиной в слово
INTEGER*4	L - целое длинное
REAL*4	F - число с плавающей запятой формата F
REAL*8	D - число с плавающей запятой формата D
REAL*8	G - число с плавающей запятой формата G
REAL*16	H - число с плавающей запятой формата H
COMPLEX*8	FC - комплексное число формата F
COMPLEX*16	DC - комплексное число формата D
COMPLEX*16	GC - комплексное число формата G
CHARACTER	T - текст ASCII
Массивы	Отсутствует
Записи	Отсутствует

Примечания:

1. Хотя для внутреннего описания логических типов данных используются типы целых без знака (BU, WU, LU), отладчик (подобно компилятору) обрабатывает логические переменные и значения, использованные в языковых выражениях, как числа со знаком.

2. Отладчик печатает числовые значения логических переменных или выражении вместо TRUE или FALSE. Обычно только значение последнего бита логических переменных или значений является значимым (0 - это ложь-FALSE, а 1 -

истина-TRUE). Однако фортран МОС ВП позволяет манипулировать всеми битами логической величины и использовать логические значения в целых выражениях. По этой причине необходимо иногда смотреть значение всего целого числа логической переменной или выражения, чтобы определить, что показывает отладчик.

3. Комплексные константы, такие как (1.0,2.0), отладчиком не поддерживаются.

6. Поддержка отладчиком языка MACRO

В этом разделе дана информация о средствах макро, поддерживаемых отладчиком (табл. 17-19).

В языке макро нет таких выражений, как в языках высокого уровня. Допускаются только выражения уровня ассемблирования и ограниченный набор знаков операций. Отладчик обеспечивает несколько знаков операций в языковых выражениях макро, которых нет в самом языке MACRO, для того, чтобы позволить программисту использовать выражения во время отладки так же свободно, как и в других языках. В частности, отладчик допускает полный набор знаков операций сравнения и булевских операций, созданных по образцу Блисс. Он также допускает знак косвенности и обычные арифметические знаки операций (см. табл. 17).

Таблица 17

Знаки операций языка Макро

Тип	Символ	Функция
Префикс	@	Косвенность
Префикс		Косвенность
Префикс	+	Унарный плюс
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Деление
Инфикс	MOD	Остаток
Инфикс	@	Сдвиг влево
Инфикс	EQL	Равно
Инфикс	EQLU	Равно
Инфикс	NEQ	Неравно
Инфикс	NEQU	Неравно
Инфикс	GTR	Больше
Инфикс	GTRU	Больше без знака
Инфикс	GEQ	Больше или равно
Инфикс	GEQU	Больше или равно без знака
Инфикс	LSS	Меньше
Инфикс	LSSU	Меньше без знака
Инфикс	LEQ	Меньше или равно
Инфикс	LEQU	Меньше или равно без знака
Префикс	NOT	Логическое не для битов
Инфикс	AND	Логическое и для битов
Инфикс	OR	Логическое или для битов

Продолжение табл. 17

Тип	Символ	Функция
Инфикс	XOR	Логическое исключающее или для битов
Инфикс	EQV	Логическая эквивалентность для битов

Таблица 18

Поддерживающие конструкции в языковых и адресных выражениях языка Макро

Символ	Конструкция
<P, S, E>	Выбор битового поля как в языке блисс

Таблица 19

Поддерживаемые типы данных языка Макро

Тип в Макро	Имя типа для СМ 1700
?	BU - байт без знака
?	WU - слово без знака
?	LU - длинное слово без знака
?	B - целое число длиной байт
?	W - целое число длиной в слово
?	L - целое число размером в длинное слово

7. Поддержка отладчиком языка Паскаль

Этот раздел включает информацию о средствах Паскаля, поддерживаемых отладчиком (табл. 20-24)

Таблица 20

Знаки операций языка Паскаль в языковых выражениях

Тип	Символ	Функция
Префикс	+	Унарный плюс
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение, конкатенация
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Реальное деление
Инфикс	DIV	Целочисленное деление
Инфикс	MOD	Основание системы счисления
Инфикс	REM	Остаток
Инфикс	**	Возведение в степень
Инфикс	IN	Установка членства
Инфикс	=	Равно
Инфикс	<>	Неравно
Инфикс	>	Больше
Инфикс	>=	Больше или равно
Инфикс	<	Меньше
Инфикс	<=	Меньше или равно
Префикс	NOT	Логическое не
Инфикс	AND	Логическое и
Инфикс	OR	Логическое или

Таблица 21

Поддерживаемые конструкции в языковых и адресных выражениях языка Паскаль

Символ	Конструкция
[]	Индексация
.	Выбор компоненты записи
^	Отличительный знак указателя

Таблица 22

Поддерживаемые предопределенные символические имена языка Паскаль

Имя	Значение
TRUE	Истина для булева выражения
FALSE	Ложь для булева выражения
NIL	Указатель нуля

Таблица 23

Поддерживаемые встроенные функции языка Паскаль

Имя	Значение
SUCC	Логический преемник
PRED	Логический предшественник

Таблица 24

Поддерживаемые типы данных языка Паскаль

Тип в Паскале	Имя типа для СМ 1700
INTEGER	L - целое число размером в длинное слово
INTEGER	W, WU - целое число размером в слово
INTEGER	B, BU - целое число размером в байт
UNSIGNED	LU - длинное слово без знака
UNSIGNED	WU - слово без знака
UNSIGNED	BU - байт без знака
SINGLE	F - число с плавающей запятой формата F
DOUBLE	D - число с плавающей запятой формата D
DOUBLE	G - число с плавающей запятой формата G
QUADRUPLE	H - число с плавающей запятой формата H
BOOLEAN	Отсутствует
CHAR	T - текст кодов ASCII
VARYING OF CHAR	VT - изменяющийся текст
SET	Отсутствует
FILE	Отсутствует
Перечисления	Отсутствует
Подобласти	Отсутствует
Введенные указатели	Отсутствует
Массивы	Отсутствует
Записи	Отсутствует

Продолжение табл. 24

Тип в Паскале	Имя типа для СМ 1700
Различные записи	Отсутствует

Примечание. Отладчик разрешает использование наборов констант паскаля, таких как [1,2,5,8..10] или [RED, BLUE], в языковых выражениях паскаль.

8. Поддержка отладчиком языка Пл/I

В данном разделе содержится информация о поддержке отладчиком языка Пл/I (табл. 25-27).

Таблица 25

Знаки операций языка Пл/I в языковых выражениях

Тип	Символ	Функция
Префикс	+	Унарный плюс
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Деление
Инфикс	**	Возведение в степень
Инфикс	!!	Конкатенация
Инфикс	=	Равно
Инфикс	^=	Неравно
Инфикс	>	Больше

Продолжение табл. 25

Тип	Символ	Функция
Инфикс	>=	Больше или равно
Инфикс	^<	Больше или равно
Инфикс	<	Меньше
Инфикс	<=	Меньше или равно
Инфикс	^>	Меньше или равно
Префикс	^	Логическое не для битов
Инфикс	&	Логическое и для битов
Инфикс	I	Логическое или для битов

Таблица 26

Поддерживаемые конструкции для языка Пл/I

в языковых и адресных выражениях

Символ	Конструкция
()	Индексация
	Выбор структурной компоненты
->	Отличительный признак указателя

Таблица 27

Поддерживаемые типы данных языка Пл/I

Тип в Пл/I	Имя типа для СМ 1700
FIXED BINARY	L - целое число размером в длинное слово
FIXED DECIMAL	P - упакованное десятичное
FLOAT BINARY	F - число с плавающей запятой формата F

Продолжение табл. 27

Тип в Пл/I	Имя типа для СМ 1700
FLOAT DECIMAL	F - число с плавающей запятой формата F
FLOAT BIN/DEC	D - число с плавающей запятой формата D
FLOAT BIN/DEC	G - число с плавающей запятой формата G
FLOAT BIN/DEC	H - число с плавающей запятой формата H
BIT	V - бит
BIT	VU - неориентированный бит
CHARACTER	T - текст кодов ASCII
CHARACTER VARYING	VT - изменяющийся текст
FILE	Отсутствует
Метки	Отсутствуют
Указатели	Отсутствуют
Массивы	Отсутствуют
Структуры	Отсутствуют

Примечание. Отладчик обрабатывает все числовые константы вида N или N.N в языковых выражениях Пл/I как упакованные десятичные константы, а не как целочисленные константы или константы с плавающей запятой, для соответствия правилам языка Пл/I. Поэтому внутреннее представление 10 это шестнадцатеричное 0C01, а не шестнадцатеричное 0A. Можно вводить константы с плавающей запятой, используя синтаксис NEN или N.NEN. В языке Пл/I нельзя вводить константы с внутренним представлением длинного целого числа. Данное ограничение обычно не является значительным во время отладки, поскольку отладчик обеспечивает правила преобразования типа Целочисленные константы можно вводить, используя операторы отладчика %HEX, %OCT и %BIN.

9. Поддержка отладчиком языка UNKNOWN

Данный раздел содержит информацию о поддержке отладчиком языка UNKNOWN (любой другой или неопределенный язык) (табл. 28-29).

Таблица 28

Знаки операций языка UNKNOWN в языковых выражениях

Тип	Символ	Функция
Префикс	+	Унарный плюс
Префикс	-	Унарный минус (отрицание)
Инфикс	+	Сложение
Инфикс	-	Вычитание
Инфикс	*	Умножение
Инфикс	/	Деление
Инфикс	**	Возведение в степень
Инфикс	&	Конкатенация
Инфикс	//	Конкатенация
Инфикс	=	Равно
Инфикс	<>	Неравно
Инфикс	/=	Неравно
Инфикс	>	Больше
Инфикс	>=	Больше или равно
Инфикс	<	Меньше
Инфикс	<=	Меньше или равно
Инфикс	EQL	Равно
Инфикс	NEQ	Неравно
Инфикс	GTR	Больше
Инфикс	GEQ	Больше или равно

Продолжение табл. 28

Тип	Символ	Функция
Инфикс	LSS	Меньше
Инфикс	LEQ	Меньше или равно
Префикс	NOT	Логическое не
Инфикс	AND	Логическое и
Инфикс	OR	Логическое или
Инфикс	XOR	Исключающее или
Инфикс	EQV	Эквивалентность

Таблица 29

Поддерживаемые конструкции в языковых и адресных выражениях языка UNKNOWN

Символ	Конструкция
[]	Индексация
()	Индексация
	Выбор компоненты записи
^	Отличительный признак указателя

Когда устанавливается язык UNKNOWN, отладчик понимает все типы данных, допускаемых другими языками, за исключением некоторых узко-специальных языков, таких как тип изображения и типы файла. В выражениях языка UNKNOWN отладчик позволяет использовать большинство стандартных типов данных МОС ВП СМ 1700.

Примечания:

1. В языке UNKNOWN отладчик допускает точечное обозна-

чение для выбора компоненты записи. Если C является компонентой записи B, которая в свою очередь является компонентой записи A, то к компоненте C можно обращаться как "A.B.C". К любым компонентам массива могут быть присоединены индексы, если, например, B - это массив, то к C можно обращаться как "A.B[2,3].C".

2. Для языка UNKNOWN отладчик разрешает как круглые, так и квадратные скобки. Следовательно, A[2,3] и A(2,3) являются эквивалентными.

Перечень ссылочных документов

1. Операционная система МОС ВП. Подсистема управления. Сообщения системы и действия по восстановлению. Справочный материал. 00152-01 97 02.

Перечень ссылочных документов

1. Операционная система МОС ВП. Подсистема управления. Сообщения системы и действия по восстановлению. Справочный материал. 26.00152-01 97 02

Лист регистрации изменений

Изм	Номера листов (страниц): изменённых	заменённых	новых	аннулированных	Всего листов (страниц) в докум.	N документа	Входящий N сопроводительного докум. и дата	Подп	Дата