Программный эмулятор ЭЦВМ М-20
(первое поколение советских ЭЦВМ).
Руководство пользователя.
Начальные сведения по программированию ЭЦВМ М-20.

Автор:
Дмитрий Викторович Стефанков

Версия 0: 1 декабря 2014 года.
Версия 9: 10 марта 2015 года.

1. Введение

Данное руководство содержит описание порядка работы пользователя с эмулятором ЭЦВМ М-20.
Общие сведения по М-20 могут найдены в литературе, список которой приведен в общем описании проекта.
Сведения по системе команд, примеры и приемы программирования для ЭЦВМ М-20 лучше всего изложены в книге [1963 Ляшенко].

2. Эмулятор ЭЦВМ М-20

Эмулятор ЭЦВМ М-20 состоит из одного исполняемого файла:
m20.exe (платформа Microsoft Windows) или **m20** (платформа Unix/Linux) .
Версия с русскими сообщениями (для М-20, не SIMH)– **m20ru.exe**.
Версия с английскими сообщениями – **m20.exe**.
Эмулятор может быть собран в виде 32-разрядного или 64-разрядного приложения.
Эмулятор является консольным приложением с текстовым пользовательским интерфейсом.
Для версий с русским языком есть поддержка нескольких кодовых страниц.
Эмулятор поддерживает интерактивный и пакетный режимы работы.

В интерактивном режиме работы с эмулятором пользователь в командной строке эмулятора вводит соответствующую команду и ждет завершения ее выполнения.
В пакетном режиме работы с эмулятором пользователь готовит файл сценарии, который содержит директивы (команды) для исполнения без участия пользователя.
В руководстве SIMH можно найти описание директив (команд) для любого из этих режимов работы.
Форматы директив (специфичных для М-20) и форматы файлов для устройств ввода/вывода М-20 можно найти в соответствующем описании для эмулятора М-20.

3. Эмулятор ЭЦВМ М-20 (первый сеанс)

Для работы с эмулятором лучше всего сделать отдельный начальный каталог, где можно расположить сам эмулятор, сценарии, входные и выходные файлы данных, и другие рабочие файлы (например: **c:\emulators\m20**).
Для команд SIMH можно использовать сокращения (см. руководство по SIMH).

Эмулятор ЭЦВМ М-20 запускается из командной строки (прав администратора не требуется):
c:\emulators\m20\m20ru.exe
После запуска мы получим приглашение эмулятора:
M-20 simulator V4.0-0 Beta git commit id: c317f685
sim>

Наберем команду “show config” или “show configuraion”:
sim> show config

Эмулятор должен вывести начальное состояние М-20:
M-20 simulator configuration
CPU
4096W, long symbolic instruction name
CDR
not attached, no input extended format
CDP
not attached, no output extended format
LPT
not attached, no new output format , no octal help format
DRUM 3 units
DRUM0 4096W, not attached

DRUM1 4096W, not attached
DRUM2 4096W, not attached
MT 4 units
MT0 75KW, not attached
MT1 75KW, not attached
MT2 75KW, not attached
MT3 75KW, not attached
sim>

Наберем команду “quit” или “bye” и завершаем первый сеанс работы с эмулятором:
sim> quit
Goodbye

Во время первого запуска мы увидели начальную конфигурацию М-20 и теперь готовы набрать свою первую программу для М-20.

Для реальной работы с М-20 рекомендуется использовать пакетный режим, используя сценарии.

3. Эмулятор ЭЦВМ М-20 (первый интерактивный сеанс)

В первых ЭЦВМ вычисление **2+2** было такой же традицией как сегодня «**Hello world**». Не будем отступать от этой традиции и попробуем сделать то же самое.

Все числа будут представлены в восьмеричном (или для упрощения в ряде случаев в десятичном виде, но будем оговаривать это отдельно).

Пусть ячейки памяти с 20 по 23 будут рабочими и распределим их так:
ячейка 20 - первый операнд (=2)
ячейка 21 - второй операнд (=2)
ячейка 22 - результат операции (=0)
ячейка 23 - ожидаемый результат (=4)

Пусть ячейки памяти с 30 по 31 будут командами и распределим их так:
ячейка 30 - команда сложения
ячейка 31 - команда останова

Загружаем эмулятор ЭЦВМ М-20 из командной строки :
c:\emulators\m20\m20ru.exe

Получаем приглашение эмулятора:
M-20 simulator V4.0-0 Beta git commit id: c317f685
sim>

Набираем следующие команды SIMH для ввода данных:
sim> de 0020 102400000000000
sim> de 0021 102400000000000
sim> de 0022 0
sim> de 0023 103400000000000

Набираем следующие команды SIMH для проверки ввода данных:
sim> ex 0020-0023
20: 1 02 4000 0000 0000
21: 1 02 4000 0000 0000
22: 0 00 0000 0000 0000
23: 1 03 4000 0000 0000
sim>

Набираем следующие команды SIMH для ввода команд М-20:
sim> de 0030 001002000210022
sim> de 0031 077000000000000

Набираем следующие команды SIMH для проверки ввода команд М-20:
sim> ex 30-31
30: 0 01 0020 0021 0022
31: 0 77 0000 0000 0000
sim> ex -m 30-31
30: [op=01 mod=0] слож_он 0020, 0021, 0022
31: [op=77 mod=0] останов_077 0000, 0000, 0000
sim>

Набираем команду SIMH для запуска нашей программы для М-20:
sim> run 0030
Останов, KRA: 0032 ([op=00 mod=0] пересылка 0000, 0000, 000
sim>

Набираем команду SIMH для проверки результатов:
sim> ex 20-23
20: 1 02 4000 0000 0000
21: 1 02 4000 0000 0000
22: 1 03 4000 0000 0000

23: 1 03 4000 0000 0000
sim>

Видим следующее из состояния ЦПУ и дампа памяти:
(1) программа завершилась по команде останова (но счетчик команд указывает на следующую команду после останова);
(2) ячейка 22 содержит теперь 4 вместо 0, что равняется содержимому ячейку 23.

Завершаем работу с эмулятором М-20:
sim> quit
Goodbye

Наши поздравления! Добро пожаловать в 1958 год!

4. Эмулятор ЭЦВМ М-20 (первый пакетный сеанс)

В первых ЭЦВМ вычисление **2+2** было такой же традицией как сегодня «**Hello world**».
Не будем отступать от этой традиции и попробуем сделать то же самое.
Все числа будут представлены в восьмеричном (или для упрощения в ряде случаев в десятичном виде, но будем оговаривать это отдельно).

Пусть ячейки памяти с 20 по 23 будут рабочими и распределим их так:
ячейка 20 - первый операнд (=2)
ячейка 21 - второй операнд (=2)
ячейка 22 - результат операции (=0)
ячейка 23 - ожидаемый результат (=4)

Пусть ячейки памяти с 30 по 31 будут командами и распределим их так:
ячейка 30 - команда сложения
ячейка 31 - команда останова

Создаем нужные файлы для запуска программы в эмуляторе М-20.

Программа hello.m20.

; Пример сложения
; десятичный ввод

```
:0020
=2
=2
=0
=4

; или восьмеричный ввод
;:0020
;1 02 4000 0000 0000
;1 02 4000 0000 0000
;1 00 0000 0000 0000
;1 03 4000 0000 0000

:0030
0 01 0020 0021 0022
0 77 0000 0000 0000

@0030                   ; Старт
```

Программа hello.simh.

```
; Пример программы
load hello.m20
ex 20-23
echo
ex 30-31
echo
ex -m 30-31
echo
echo Start
run
show queue
show time
;show throttle
ex 20-23
echo
quit
```

Вызываем эмулятор:
c:\emulators\m20\m20ru.exe hello.simh >hello_ru.out

Результат в файле **hello_ru.out.**

M-20 simulator V4.0-0 Beta git commit id: c317f685
20: 1 02 4000 0000 0000
21: 1 02 4000 0000 0000
22: 1 00 0000 0000 0000
23: 1 03 4000 0000 0000
30: 0 01 0020 0021 0022
31: 0 77 0000 0000 0000

30: слож_он 20, 21, 22
31: останов_077

Start
hello.simh-10> run
Останов, KRA: 0032 (пересылка)
M-20 event queue empty, time = 53, executing 50000 instructios/sec
Time: 53

20: 1 02 4000 0000 0000
21: 1 02 4000 0000 0000
22: 1 03 4000 0000 0000
23: 1 03 4000 0000 0000

Goodbye

В файле результатов мы видим следующее до исполнения программы:
(1) по адресам 20-23 введенные нами числа;
(2) по адресам 30-31 простейшую программу сложения содержимых ячеек 20 и 21 с записью результата сложения в ячейку 22.

После исполнения (после команды **run**) видим следующее:
(1) программа завершилась по команде останова (но счетчик команд указывает на следующую команду после останова);
(2) ячейка 22 содержит теперь 4 вместо 0, что равняется содержимому ячейку 23.
Наши поздравления! И вновь добро пожаловать в 1958 год!

Другие примеры и программы можно найти в комплекте поставки эмулятора М-20 или посмотреть примеры в литературе, список которой приведен в общем описании проекта.

5. Эмулятор ЭЦВМ М-20 (пакетный сеанс загрузки с ЧУ)

Перенесем вычисление **2+2** на перфокарты и попробуем провести загрузку и исполнение программы почти как на реальной ЭЦВМ М-20.
Все числа будут представлены в восьмеричном (или для упрощения в ряде случаев в десятичном виде, но будем оговаривать это отдельно).

Пусть ячейки памяти с 1000 по 1004 будут рабочими и распределим их так:
ячейка 1000 - не используется
ячейка 1001 - первый операнд (=2)
ячейка 1002 - второй операнд (=2)
ячейка 1003 - результат операции (=0)
ячейка 1004 - ожидаемый результат (=4)

Пусть ячейки памяти с 1 по 3 будут командами и распределим их так:
ячейка 1 - команда сложения
ячейка 2 - команда останова
ячейка 3 - команда останова (на всякий случай)

Создаем нужные файлы для запуска программы в эмуляторе М-20.

Программа boot.cdr.

```
; Запуск рабочей программы (2+2) с перфоркарты (пример начальной загрузки);
 (2015 Стефанков Д.В.)

; Card 0
; Simple Program: 2+2=4
; :1
1       0 01 1001 1002 1003   0
1       0 77 0000 0000 0000   0
1       7 77 7777 7777 7777   0
; :1000
0     0 00 1000 0000 0000   1   ; КА
1     0 00 0000 0000 0000   0
1       1 02 4000 0000 0000   0
```

```
1         1 02 4000 0000 0000   0
1         1 00 0000 0000 0000   0
1         1 03 4000 0000 0000   0
;
; end-of-input marker and checksum
1      5 07 6001 1002 1004   1
```

Контрольную сумму можно получить следующими способами:
(1) включить отладку для устройства cdr, посмотреть вычисленную к/с при вводе и затем ввести ее вместе фиктивной (обыкновенно я так и делаю);
(2) ввести карту суммирования, а за ней ввести массив кодов для суммирования, далее как пункте 1;
(3) в программе суммирования ввести массива кодов, запустить программу, далее как в пункте 1.

Программа boot.simh.

```
; Запуск рабочей программы (2+2) с перфоркарты (пример начальной загрузки)
; (2015 Стефанков Д.В.)
;
! del boot_debug.txt
! del boot.lst
! del boot.cdp
;
set console debug=boot_debug.txt
set cpu debug
set lpt debug
set cdp debug
set cdr debug
;
de DEBUG_DUMP_REGS 1
de DEBUG_DUMP_MEM  1
de DEBUG_DUMP_MODERM_MEM 1
;de ARITHMETIC_OP_DEBUG 1
;
att lpt boot.lst
att -r cdr boot.cdr
attach cdp boot.cdp
;
ex 1-5
echo
```

```
ex -m 1-5
echo
ex 1000-1005
echo
;
echo Boot
boot cdr
;
show queue
show time
;show throttle
ex 1-5
echo
ex -m 1-5
echo
ex 1000-1005
echo
quit
```

Вызываем эмулятор:

```
c:\emulators\m20\m20ru.exe boot.simh >boot_ru.out
```

Результат в файле **boot_ru.out.**

```
M-20 simulator V4.0-0 Beta       git commit id: c317f685
Debug output to "boot_debug.txt"
LPT: creating new file
CDR: unit is read only
CDP: creating new file
1:      0 00 0000 0000 0000
2:      0 00 0000 0000 0000
3:      0 00 0000 0000 0000
4:      0 00 0000 0000 0000
5:      0 00 0000 0000 0000

1:      [op=00 mod=0] пересылка              0000, 0000, 0000
2:      [op=00 mod=0] пересылка              0000, 0000, 0000
3:      [op=00 mod=0] пересылка              0000, 0000, 0000
4:      [op=00 mod=0] пересылка              0000, 0000, 0000
5:      [op=00 mod=0] пересылка              0000, 0000, 0000
```

1000: 0 00 0000 0000 0000
1001: 0 00 0000 0000 0000
1002: 0 00 0000 0000 0000
1003: 0 00 0000 0000 0000
1004: 0 00 0000 0000 0000
1005: 0 00 0000 0000 0000

Boot

***** Command time profile stat *****
opcode=01 count=1 times=28.50 avg_time=28.50 (слож_он)
opcode=10 count=1 times=500000.00 avg_time=500000.00 (ввод_пфк_останов)
opcode=77 count=1 times=24.00 avg_time=24.00 (останов_077)
Summary: times=500052.50 count=3 avg_time=166684.17
****

boot.simh-32> boot cdr

Останов, KRA: 0002 ([op=77 mod=0] останов_077 0000, 0000, 0000)
M-20 event queue empty, time = 500053, executing 50000 instructios/sec
Time: 500053

1: 0 01 1001 1002 1003
2: 0 77 0000 0000 0000
3: 7 77 7777 7777 7777
4: 0 00 0000 0000 0000
5: 0 00 0000 0000 0000

1: [op=01 mod=0] слож_он 1001, 1002, 1003
2: [op=77 mod=0] останов_077 0000, 0000, 0000
3: [op=77 mod=7] останов_077 @-1, @-1, @-1
4: [op=00 mod=0] пересылка 0000, 0000, 0000
5: [op=00 mod=0] пересылка 0000, 0000, 0000

1000: 0 00 0000 0000 0000
1001: 1 02 4000 0000 0000
1002: 1 02 4000 0000 0000
1003: 1 03 4000 0000 0000
1004: 1 03 4000 0000 0000
1005: 0 00 0000 0000 0000

Goodbye
Debug output disabled

В файле результатов мы видим следующее до исполнения программы:
(1) по всем адресам нули как по включении питания:
(2) и программы тоже нет.

После исполнения (после команды **boot**) видим следующее:
(1) программа завершилась по команде останова (но счетчик команд указывает на следующую команду после останова);
(2) ячейка 1002 содержит теперь 4 вместо 0, что равняется содержимому ячейку 1003.
В трассировочном файле отладки можно посмотреть более детальную картину загрузки и исполнение программы.
Наши поздравления! И вновь добро пожаловать в 1958 год!

6. Интерпретирующая система ИС-2

Для ЭЦВМ М-20 была разработана интерпретирующая система (ИС-2) для возможности загрузки, настройки и запуска готовых библиотечных программ (стандартных программ или стандартных подпрограмм).
К сожалению, версия ИС-2 1961 года от Шуры-Буры не работает верно.
В качестве текущей версии ИС-2 взята версия из книга Ляшенко 1963 года.
Тексты СП были взяты из книги Шуру-Буры 1961 года.
Готовые примеры можно найти в соответствующем каталоге.
Краткое описание каждого примера можно посмотреть в файле **files.txt**.
В примерах есть сборка ИС-2 и СПП для МБ и МЛ.
Ниже приводится список СП, которые доступны для эмулятора.

Номер СП (восьмеричный)	**Назначение**	**Статус (состояние)**
0	программа обмена	тест пройден
1	x в степени y (x**y)	тест пройден
2	перевод чисел из десятичной системы исчисления в двоичную (10 -> 2)	тест пройден
3	e в степени x (e**x)	тест пройден
4	Натуральный логарифм x (ln x)	тест пройден

5	синус x (sin x)	тест пройден
6	арксинус x (arcsin x)	тест пройден
7	выдача восьмеричных кодов на десятичную печать с возможностью изменения их нумерации	тест пройден
10	перевод чисел из десятичной системы исчисления в двоичную (10 -> 2)	тест пройден
11	тангенс x (tgx)	тест пройден
12	арктангенс x (arctgx)	тест пройден
13	тест печати	Тест пройден
15	печать материала с любым числом кодов	тест пройден
16	перфорация материала с любым числом кодов	тест пройден
27	печать материала с одновременным переводом в десятичную систему и сохранением двоичного материала в МОЗУ	тест пройден
30	перевод материала в десятичную систему и вывод на печать с тестом печати	тест пройден
32	умножение диагональной матрицы на вектор	тест пройден
33	умножение матрицы на вектор	тест пройден
34	умножение матрицы на диагональную матрицу	тест пройден
35	умножение диагональной матрицы на полную матрицу	тест пройден
36	умножение матриц	тест пройден
37	обращение матрицы	тест НЕ пройден
40	перевод материала в десятичную систему и перфорация с тестом перфорации	тест пройден

42	групповой перевод чисел из десятичной системы счисления в двоичную	тест пройден
43	перевод материала в десятичную систему и вывод на печать без теста печати	тест пройден
47	выдача восьмеричных кодов на десятичную печать	тест пройден
50	перестановка стандартных программ	тест НЕ пройден (проходится, но требуется ряд ручных действий)

7. Список программ для эмулятора М-20

Примеры программ для эмулятора ЭЦВМ М-20 можно найти в соответствующем каталоге.
Краткое описание каждой программы можно посмотреть в файле **files.txt**.
Образы ИС-2 и СПП есть на МБ и МЛ (все это можно найти в каталоге ИС-2).
Образы перфокарт можно также найти в соответствующем каталоге.

8. Параметры для устройств М-20

Для просмотра опций (модификаторов) устройств нужно набрать команду для вывода списка всех модификаторов «**show modifiers**» или команду для вывода модификаторов одного устройства «**show cpu mod**» [«**show device-name mod**»].

1. Устройство ЦПУ (CPU модуль)
«**set cpu short_sym_opcode**» - использовать короткие символические имена инструкций
«**set cpu long_sym_opcode**» - использовать длинные символические имена инструкций
«**de RUN_MODE**» - режим работы ЭЦВМ М-20 (0=авто,1=цикл,2=импульс)
«**de MOSU_MODE**» - режим работы МОЗУ (1=режим I, 2=режим II)
«**de DEBUG_DUMP_REGS**» - дамп регистров текущей команды при отладке (1=да,0=нет)
«**de DEBUG_DUMP_MEM**» - дамп памяти текущей команды при отладке (1=да,0=нет)
«**de (DEBUG_DUMP_MODERM_MEM**» - дамп памяти текущей команды при отладке в

современном представлении чисел (1=да,0=нет)
«**de ENABLE_M20_PRINT_ASCII_TEXT**» - печать АЦПУ вместо ЦПУ (1=да,0=нет)
(Историческая справка: на М-20 не было алфавитно-цифровой печати до 1961 года)
«**de DISABLE_IS2_TRACE**» - выдача трассировки ИС-2 (1=да,0=нет)
«**de PRINT_SYS_STAT**» - выдача статистики по исполненным командам (1=да,0=нет)
«**de PRINT_STAT_ON_BREAK**» - выдача статистики по исполненным командам на точке остановки breakpoint (1=да,0=нет)
«**de DIAG_PRINT**» - печать принятых входных строк программы (1=да,0=нет)
«**de MEMORY_45_CHECKING**» - проверка выхода за границу 45 разрядов (1=да,0=нет)
«**de ARITHMETIC_OP_DEBUG**» - отладочная печать для проверки арифметических операций (1=да,0=нет)
«**de USE_NEW_ADD**» - использовать другой вариант операции сложения/вычитания/вычитания_по_модулю (1=да,0=нет)
«**de USE_NEW_MULT**» - использовать другой вариант операции умножения (1=да,0=нет)
«**de USE_NEW_DIV**» - использовать другой вариант операции деления (1=да,0=нет)
«**de USE_NEW_SQRT**» - использовать другой вариант операции извлечения квадратного корня (1=да,0=нет)
«**de USE_ADD_SBST**» - использовать иной вариант операции сложения/вычитания/вычитания_по_модулю (1=да,0=нет)

2. Устройство МЛ (МТ модуль)
«**de TAPE_AUTO_SKIP_ZERO_ADDRESS**» - автоматический пропуск нулевого адреса при операциях чтения/записи (1=да,0=нет)
«**de TAPE_MAP_CHECK**» - проверка допустимости трансляции логических номеров в номера физических устройств (1=да,0=нет)
«**de LOG_TAPE_0_MAP**» - трансляция логической МЛ 0 на физическую МЛ (0-3)
«**de LOG_TAPE_1_MAP**» - трансляция логической МЛ 1 на физическую МЛ (0-3)
«**de LOG_TAPE_2_MAP**» - трансляция логической МЛ 2 на физическую МЛ (0-3)
«**de LOG_TAPE_3_MAP**» - трансляция логической МЛ 3 на физическую МЛ (0-3)
«**de TAPE_0_ACCESS_MODE**» - режим доступа к физической МЛ 0 (0=нет доступа,1=чтение,2=запись,4=разметка и комбинации этих разрядов)
«**de TAPE_1_ACCESS_MODE**» - режим доступа к физической МЛ 1 (0=нет доступа,1=чтение,2=запись,4=разметка и комбинации этих разрядов)
«**de TAPE_2_ACCESS_MODE**» - режим доступа к физической МЛ 2 (0=нет доступа,1=чтение,2=запись,4=разметка и комбинации этих разрядов)
«**de TAPE_3_ACCESS_MODE**» - режим доступа к физической МЛ 3 (0=нет доступа,1=чтение,2=запись,4=разметка и комбинации этих разрядов)
«**de TAPE_READ_DATA_DUMP**» - дамп данных после чтения с МЛ
«**de TAPE_WRITE_DATA_DUMP**» - дамп данных перед записью на МЛ

«**de TAPE_FORMAT_DATA_DUMP**» - дамп данных перед форматированием зоны МЛ

3. Устройство МБ (DRUM модуль)
«**de DRUM_AUTO_SKIP_ZERO_ADDRESS**» - автоматический пропуск нулевого адреса при операциях чтения/записи (1=да,0=нет)
«**de DRUM_MAP_CHECK**» - проверка допустимости трансляции логических номеров в номера физических устройств (1=да,0=нет)
«**de LOG_DRUM_0_MAP**» - трансляция логического МБ 0 на физический МБ (1-3)
«**de LOG_DRUM_1_MAP**» - трансляция логического МБ 1 на физический МБ (1-3)
«**de LOG_DRUM_2_MAP**» - трансляция логического МБ 2 на физический МБ (1-3)
«**de LOG_DRUM_3_MAP**» - трансляция логического МБ 3 на физический МБ (1-3)
«**de DRUM_0_ACCESS_MODE**» - режим доступа к физическому МБ 0 (0=нет доступа,1=только-чтение,2=только-запись,3=чтение-запись)
(Формально его нет, но появился по причине реализации из-за особенностей SIMH)
«**de DRUM_1_ACCESS_MODE**» - режим доступа к физическому МБ 1 (0=нет доступа,1=только-чтение,2=только-запись,3=чтение-запись)
«**de DRUM_2_ACCESS_MODE**» - режим доступа к физическому МБ 2 (0=нет доступа,1=только-чтение,2=только-запись,3=чтение-запись
«**de DRUM_3_ACCESS_MODE**» - режим доступа к физическому МБ 3 (0=нет доступа,1=только-чтение,2=только-запись,3=чтение-запись)
«**de DRUM_READ_DATA_DUMP**» - дамп данных после чтения с МБ
«**de DRUM_WRITE_DATA_DUMP**» - дамп данных перед записью на МБ

4. Устройство БПУ (LPT модуль)
«**set lpt newextfmt**» - использовать при десятичной печати современное представление десятичного числа
«**set lpt nonewextfmt**» - не использовать при десятичной печати современное представление десятичного числа (по умолчанию)
«**set lpt octhelpfmt**» - выводить дополнительно при восьмеричной печати современное представление восьмеричного числа
«**set lpt noocthelpfmt**» - не выводить дополнительно при восьмеричной печати современное представление восьмеричного числа (по умолчанию)
«**de LPTWIDTH N**» - установить ширину печати в N чисел (по умолчанию = 7)
«**de decimal_print_type V**» - установить тип десятичной печати (если не используется современное представление), по умолчанию = 0

5. Устройство ЧУ (CDR модуль)
«**set cdr extfmt**» - разрешить ввод чисел в стандарте IEEE (=-1E+1)

«**set cdr noextfmt**» - запретить ввод чисел в стандарте IEEE (по умолчанию)

6. Устройство ПФ (CDP модуль)
«**set cdp extfmt**» - разрешить вывод адресных кодов на перфокарты
«**set cdp noextfmt**» - запретить вывод адресных кодов на перфокарты (по умолчанию)
«**de BCDPRINT [0|1]**» - 0=вывести как восьмеричное число,1=как двоично-десятичное число