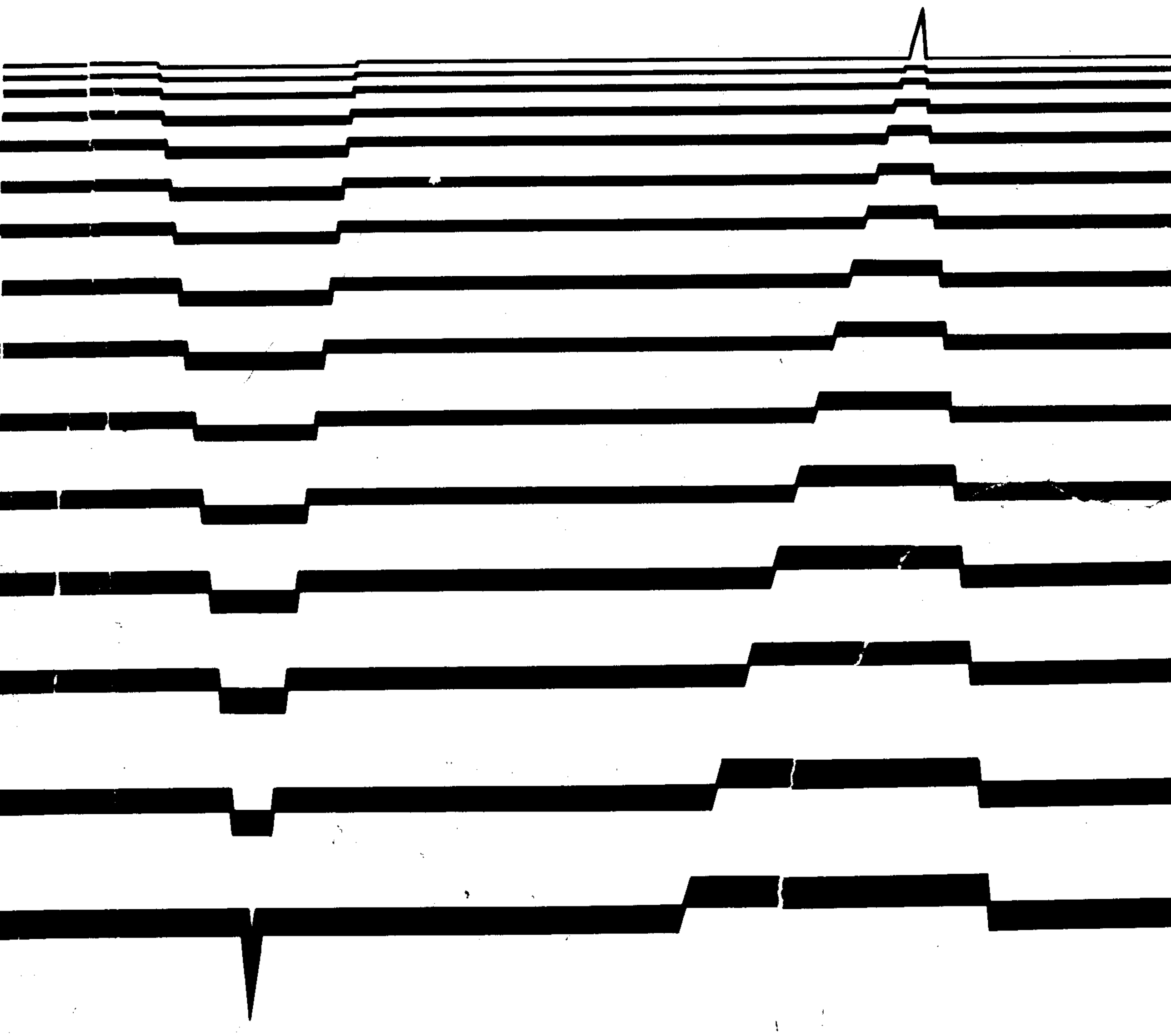
YAMAHA

MSX 2

# ЛОКАЛЬНАЯ СЕТЬ
## Версия 3.0

## РУКОВОДСТВО



# СОДЕРЖАНИЕ

## Общие сведения о локальной сети.

Назначение сети.

Данная локальная сеть разработана для связи компьютеров в учебном классе. Она позволяет соеденять преподавателя и 15 учеников, давая при этом возможность обмена программами, данными и сообщениями, как между преподавателем и учеником, так и между двумя учениками. Сеть работает не только в Бейсике, но и в MSX-DISK-BASIC, MSX-DOS и MSX-CP/M.

**Примечания**: 1. Ниже, для простоты записи, вместо "компьютер преподавателя" и "компьютер ученика" будет употребляться "преподаватель" и "ученик".

2. Как и в предыдущих версиях сети, в Версии 3.0 при обращении ученика к дисководу, команды сети Бейсика временно отключаются.

3. Для работы сети используется специальное ОЗУ (2 кбайт) в сетьевом блоке, обычное ОЗУ при работе с сетью не используется. Для обращения к сетьевому ОЗУ, используйте карту памяти.

4. Вызов сетьевой BIOS (системы базовых функций ввода-вывода) из MSX-DOS или MSX-CP/M ссылается на рабочую область из 8 байт в основном ОЗУ.

5. Если при работающей сети невозможно запустить некоторые прикладные программы (например MSX Painter), воспользуйтесь командой NETEND, а после нее повторите вызов Вашей программы.

# Список дополнительных операторов Бейсика

| | |
|---|---|
| **CALL HELP** ..... | выводит этот список с форматами. |
| **CALL WHO** ..... | возвращает номер Вашего компьютера. |
| **CALL SNDRUN** ..... | передача и запуск программы на Бейсике. |
| **CALL SEND** ..... | передача программы на Бейсике. |
| **CALL BSEND** ..... | передача программы в машинном коде или изображения с экрана. |
| **CALL RECIEVE** ..... | прием программы на Бейсике. |
| **CALL BRECIEVE** ..... | прием программы в машинных кодах или изображения с экрана. |
| **CALL SNDMAIL** ..... | передача информации в почтовый ящик преподователя. |
| **CALL RCVMAIL** ..... | прием информации из почтового ящика преподователя. |
| **CALL MESSAGE** ..... | передача сообщения преподавателем. |
| **CALL SNDCMD** ..... | передача команды. |
| **CALL RUN** ..... | запуск программы на Бейсике у ученика. |
| **CALL STOP** ..... | остановка программы на Бейсике у ученика |
| **CALL POKE** ..... | запись числа в память к ученику или в сетьевое ОЗУ. |
| **CALL PEEK** ..... | чтение числа из памяти у ученика и из сетьевого ОЗУ. |
| **CALL PON** ..... | начало упорядоченного опроса. |
| **CALL POFF** ..... | конец упорядоченного опроса. |
| **CALL CHECK** ..... | проверка кто из учеников подключен к сети. |
| **CALL TALK** ..... | передача сообщения учеником. |
| **CALL ONLINE** ..... | включение в сеть (только после команды NETINIT). |
| **CALL OFFLINE** ..... | отключение от сети (только после команды NETINIT). |
| **CALL NETEND** ..... | выключение сети. |
| **CALL NETINIT** ..... | инициализация сети. |
| **CALL ENACOM** ..... | разрешение передачи учеником. |
| **CALL DISCOM** ..... | запрещение передачи учеником. |

# HELP

**[Формат]** **CALL HELP**

**[Функция]** Показывает список команд с форматами.

Эта команда показывает список сетьевых команд Бейсика, которые используются при работе с сетью в классе. Экран должен быть в текстовом режиме. Так как наборы команд преподавателя и ученика различны, списки у них будут соответственно отличаться. У ученика, кроме того, указывается номер его компьютера. Эта функция у ученика работает вне зависимости от команды разрешения передачи (CALL ENACOM). Когда передача разрешена, в список добавляются команды: SEND, RECEIVE, BSEND, BRESEIVE и TALK.

# WHO

**[Формат]** **CALL WHO** [ ( <переменная> ) ]

**[Функция]** Возвращает номер Вашего компьютера.

**[Пример 1]**
```
CALL WHO (A)
PRINT A 3
```

Эта команда возвращает номер компьютера, установленный при помощи переключателя в сетьевом блоке (для преподавателя - 0, для учеников - 1-15). В этом примере, команда возвращает номер 3 ученика.

**[Пример 2]**
```
CALL WHO 3
```

Когда переменная отсутствует, номер сразу выводится на экран. Для использования этого формата Вы должны быть в командном режиме.

# BSEND

**[Функция]** **CALL BSEND** (<имя файла> [, [<номер ученика> ] [, [ <начальный адрес>] [, [<конечный адрес>] [, [<S>] ] ] ] )

**[Функция]** Передача программ в машинном коде или изображения с экрана.

**[Пример 1]** CALL BSEN ("DATA.BIN",12,&H8000)

Эта команда передает программу в машинных кодах или данные, записанные из Бейсика командой BSAVE или CALL BRECEIVE, по начальному адресу в память ученика, определенного номером. В этом примере файл "DATA.BIN" пересылается в память ученика номер 12 с начального адреса &H8000 . Если номер ученика не указан, программа или данные передается всем ученикам. Если не указан начальный адрес, программа или данные пересылаются с того адреса, с которого они были записаны.

**[Пример 2]** CALL BSEN (,12,&H0100,&H2000,S)

Когда Вы указываете <S>, содержимое Вашего экрана (видеопамяти) копируется в видеопамять ученика с указанным номером. Перед передачей режим видеопроцессора в принимающем компьютере автоматически меняется на тот режим, в котором находится передающий. В этом примере содержимое видеопамяти преподавателя с адреса &H0100 до адреса &H2000 передается по адресам с &H0100 по &H2000 на экран (в видеопамять) ученика номер 12. Когда номер ученика отсутствует, информация передается всем ученикам.

CALL BSEN (,12,,,S)
Когда используется <S> и адреса не указаны, то вместо начального берется &H0000, а вместо конечного - &HFFFF.

**[Пример 3]** CALL BSEN ("VDAT.BIN",12,&H0100,&H2000,S)

Эта команда передает данные с экрана (из видеопамяти), записанные с помощью команд Бейсика BSAVE или CALL BRESEIVE с использованием <S>, в область с начального по конечный адреса в видеопамять указанного ученика. Когда данные в видеопамять пересылаются из файла, режим видеопроцессора автоматически не устанавливается, т.е. он должен быть установлен до использования команды CALL BSEND. В этом примере данные из файла "VDATA.BIN" передаются по адресам с &H0100 по &H2000 в видеопамять

ученика с номером 12. Если номер ученика не указан данные передаются всем ученикам. Если не указан начальный адрес данные будут пересылаться начиная с того адреса, с которого они были записанны. Если этот адрес больше чем конечный, то используется именно он, а конечный игнорируется. А если меньше, то данные будут посланны лишь до конечного адреса, указанного в команде CALL BSEN.

# BRECEIVE

**[Формат]** **CALL BRECEIVE** (<имя файла> [,[<номер ученика>], [,[<начальный адрес>] [, [<конечный адрес>] [,<S>] ] ] ] ] )

**[Функция]** Прием программы в машинных кодах или изображения с экрана.

**[Пример 1]** CALL BREC ("DATA.BIN",10,&HA000,&HB000)

Эта команда записывает программу в машинных кодах или данные из памяти, с начального адреса по конечный, в файл на диск. Формат для записи такой же, как и в команде Бейсика BSAVE, только стартовым адресом всегда является начальный. В этом примере содержимое памяти ученика номер 10, с адреса &HA000 по &HB000, записывается в файл "DATA.BIN".

**[Пример 2]** CALL BREC (,10,&H0100,&H2000,S)

Когда используется <S>, содержимое экрана (видеопамяти) ученика с указанным номером принимается в видеопамять преподавателя. Перед этим видеопроцессор автоматически устанавливается в соответствующий режим. В этом примере содержимое видеопамяти ученика номер 10 с адреса &H0100 по &H2000 пересылается по этим же адресам на экран (в видеопамять) преподавателя. При использовании <S>, если Вы не указываете начальный адрес, используется - &H0000, а вместо конечного - &HFFFF.

**[Пример 3]** CALL BREC (" DATA.BIN ",12,&H0100,&H2000,S)

Содержимое экрана (видеопамяти) ученика принимается и записывается на диск. Формат для записи такой же, как и в команде Бейсика BSAVE при использовании <S>. Так как записываемый файл (содержимое видеопамяти) не содержит информацию о режиме видеопроцессора, необходимо при использовании такого файла предварительно установить соответствующий режим. В этом примере данные с экрана (из видеопамяти) ученика номер 12 с адреса &H0100 по &H2000 записываются на диск, в файл с названием "VDATA.BIN".
При использовании <S>, если Вы не указываете начальный адрес, используется - &H0000, а вместо конечного - &HFFFF.

# ENACOM

**[Формат]** **CALL ENACOM** ( [ <номер ученика > ] )

**[Функция]** Разрешает передачу учеником.

**[Пример]** CALL ENAC (12)

Эта команда разрешает указанному преподавателем ученику передачу в сеть. Этот пример разрешает связь ученику номер 12, т.е. он может использовать команды CALL SEND, CALL RESEIVE, CALL BSEND, CALL BRESEIVE. При указании 0 передача в сеть разрешается всем ученикам. Ученик, которому передача разрешена, может посылать сообщения другим ученикам командой TALK.

# DISCOM

**[Формат]** **CALL DISCOM** ( [ < номер ученика > ] )

**[Функция]** Запрещает передечу учеником.

**[Пример]** **CALL ENAC** (12)

Эта команда запрещает указанному преподавателем ученику передачу в сеть. Этот пример запрещает связь ученику номер 12, т.е. он не может использовать команды CALL SEND, CALL RESEIVE, CALL BSEND, CALL BRESEIVE. При указании 0 передача в сеть запрещается всем ученикам. При начале работы с сетью, передача в сеть запрещена всем ученикам.

# CHECK

**[Формат]** **CALL CHECK** ( [ <переменная> ] [, [ <переменная> ] ] )

**[Функция]** Проверяет кто из учеников подключен к сети и кому из учеников разрешена передача в сеть.

**[Пример 1]**

```
CALL CHECK (A)
PRINT BIN$(A)
110110111101100
```

Эта команда проверяет, кто из учеников подключен к сети и возвращает двоичное число, в котором младший разряд соответствует ученику номер 1, следующий - номер 2 и т. д. Когда соответствующий бит равен 0, это означает, что ученик подключен к сети, а когда равен 1 - отключен от нее. В этом примере к сети подключены ученики номер 1, 2, 5, 10 и 13. Когда все ученики подключены, значение возвращаемой переменной равно 0.
То, что ученик отключен от сети означает: либо, что компьютер выключен или сеть физически от него отключена, либо, что это сделано программно при помощи команды CALL OFFLINE.

**[Пример 2]**

```
CALL CHECK (,B)
PRINT BIN$(B)
111111111101110
```

Эта команда проверяет, кому из учеников разрешена передача в сеть, и возвращает двоичное число, в котором младший разряд соответствует ученику номер 1, следующий - номер 2 и т. д. Когда соответствующий бит равен 0, это означает, что передача в сеть этому ученику разрешена, а когда равен 1 - запрещена. В этом примере передача разрешена ученикам номер 1 и 5. Когда передача разрешена всем ученикам значение возвращаемой переменной равно 0. Разрешение и запрещение передачи учеником в сеть производится соответственно командами CALL ENACOM и CALL DISCOM.

# NETINIT

**[Формат]** **CALL NETINIT**

**[Функция]** Инициализация сети.

**[Пример]** CALL NETINIT

Эта команда используется для начала работы с сетью, если та не была инициализирована при включении компьютеров. В этом случае без подачи преподавателем этой команды сеть работать не будет. Обычно сеть инициализируется при включении.

# NETEND

**[Формат]** **CALL NETEND**

**[Функция]** Отключение сети.

**[Пример]** CALL NETEND

Эта команда используется для отключения сети, когда Вы хотите работать с прикладной программой, которая не может быть вызвана при работающей сети.

# POKE

**[Формат]** **CALL POKE** ( <число>,<адрес>,<номер ученика>,<N> )

**[Функция]** Запись числа в память ученика или в сетьевое ОЗУ.

**[Пример 1]** CALL POKE (100,&H7800) для преподавателя/ученика

Эта команда записывает указанное число по указанному адресу в сетьевое ОЗУ (&H7800-&H7FFF). В этом примере 100 записывается по адресу сетьевых сообщений &H7800.

**[Пример 2]** CALL POKE (100,&HB000,1) для преподавателя

Эта команда записывает данное число по указанному адресу в память означенному ученику. Если в качестве номера указан 0, то число записывается в память всем ученикам. В этом примере 100 записывается по адресу &HB000 в память ученику номер 1.

**[Пример 3]** CALL POKE (100,&H7A00,1,N) для преподавателя

Эта команда записывает данное число по указанному адресу в сетьевое ОЗУ (&H7800-&H7FFF) означенному ученику. В этом примере 100 записывается по адресу &H7A00 в сетьевое ОЗУ ученику номер 1.

# PEEK

**[Формат]** **CALL PEEK** ( <число>,<адрес>,<номер ученика>,<N> )

**[Функция]** Читает число из памяти ученика или из сетьевого ОЗУ.

**[Пример 1]** CALL PEEK (A,&H7800) для преподавателя/ученика

Эта команда читает ячейку по данному адресу в сетьевом ОЗУ (&H7800-&H7FFF) и возвращает в указанной переменной. В этом примере читаются данные из ячейки сетьевых сообщений с адресом &H7800 и возвращаются в переменной A.

**[Пример 2]** CALL PEEK (A,&HB000,1) для преподавателя

Эта команда читает данные по данному адресу из памяти указанного ученика и возвращает значение в заданной переменной. Если в качестве номера указан 0, то генерируется ошибка. В этом примере читаются данные из ячейки памяти с адресом &HB000 ученика номер 1 и возвращаются в переменной A.

**[Пример 3]** CALL PEEK (A,&H7A00,1,N) для преподавателя

Эта команда читает данные по данному адресу в сетьевом ОЗУ (&H7800-&H7FFF) указанного ученика и возвращает значение в заданной переменной. В этом примере читаются данные из ячейки сетьевого ОЗУ с адресом &HB000 ученика номер 1 и возвращаются в переменной A.

# MESSAGE

**[Формат]** **CALL MESSAGE** ( <сообщение> [, [ <номер ученика> ] ] )

**[Функция]** Передача сообщения преподавателем.

**[Пример]** CALL MESSAGE ( “Hello !!”,10)

Эта команда передает сообщение преподавателя одному или нескольким ученикам. Сообщение имеет максимальную длину 56 символов, но, когда оно высвечивается в 24 строке на экране ученика, его длина зависит от того, в каком режиме сейчас ученик. Если в момент передачи ученик находится в графическом режиме, то сообщение будет выведено к нему на экран, как только он выйдет в текстовой режим. Если параметр <номер ученика> пропущен, сообщение передается всем ученикам. Этот пример высвечивает сообщение “Hello !!” в 24 строке на экране у 10 ученика. Чтобы сообщение исчезло - нажмите пробел.

**Примечание:** Если у ученика выключена индикация функциональных клавиш (режим KEYOFF), то сообщение будет двигаться вверх по мере заполнения экрана, т.е. будет рассматриваться как обычный текст.

# TALK

**[Формат]** **CALL TALK** ( <сообщение> [, [ <переменная>] ] )

**[Функция]** Передача сообщения учеником.

**[Пример]** CALL TALK ("Не могу понять",B)

Когда ученику не разрешена передача, он может послать сообщение только преподавателю, а когда разрешена - и другим ученикам. Пользуясь этой командой, ученик должен определить номер ученика/преподавателя, используя переменную. Если значение переменной равно 0, сообщение передается преподавателю. Если от 1 до 15 - соответствующему ученику. Если передача прошла успешно, в переменной возвращается 0, если нет - 255. Сообщение имеет максимальную длину 56 символов, но, когда оно высвечивается в 24 строке на экране, его длина зависит от того, в каком режиме сейчас компьютер. В этом примере сообщение "Не могу понять" передается преподавателю/ученику, определенному в переменной B.

# SEND

**[Формат]** **CALL SEND** [( [[<имя файла>] [, [<номер ученика>] ]] )]

**[Функция]** Пересылает ученику программу на Бейсике.

**[Пример 1]** CALL SEND ("A:TEST.BAS",0)

Эта команда считывает заданную программу на Бейсике с диска и посылает ее указанному ученику. Если ученик работает с программой на Бейсике, его программа будет стерта и он получит новую. Во время передачи у него на экране будет высвечено сообщение "Wait" (ждите), а как только пересылка будет закончена - сообщение "OK". Если номер ученика опускается или равен 0 - программа передается всем ученикам. Эта команда не стирает содержимое памяти преподавателя и может быть использована в программном режиме. В этом примере программа на Бейсике "TEST.BAS" пересылается всем ученикам.

**[Пример 2]** CALL SEND (,10)

Эта команда передает программу, находящуюся в памяти преподавателя. В этом примере программа преподавателя пересылается ученику номер 10. Это команда может быть использована в программном режиме. Во время передачи, на экране ученика будет высвечено сообщение "Wait" (ждите). Если номер ученика опускается, программа передается всем ученикам.

# SNDRUN

**[Формат]** **CALL SNDRUN** [ ( [[<имя файла>] [,<номер ученика>]]] ) ]

**[Функция]** Пересылает ученику программу на Бейсике и запускает ее.

**[Пример 1]** CALL SNDRUN ("A:TEST.BAS",0)

Эта команда считывает данную программу на Бейсике с диска, посылает ее указанному ученику и запускает ее. Если ученик работает с программой на Бейсике, его программа будет стерта и он получит новую, которая и будет запущена. Во время передачи, на экране ученика будет высвечено сообщение "Wait" (ждите). Как только программа будет получена она сразу запускается. Если номер ученика опускается или равен 0 - программа передается всем ученикам. Эта команда не стирает содержимое памяти преподавателя и может быть использована в программном режиме. В этом примере программа на Бейсике "TEST.BAS" пересылается и запускается у всех учеников.

**[Пример 2]** CALL SNDRUN (,10)

Эта команда передает программу, находящуюся в памяти преподавателя и запускает ее у ученика. В этом примере программа преподавателя пересылается ученику номер 10 и запускается. Это команда может быть использована в программном режиме. Во время передачи на экране ученика будет высвечено сообщение "Wait" (ждите). Если номер ученика опускается, программа передается и запускается у всех учеников.

# RECEIVE

**[Формат]** **CALL RECEIVE** ( [ <имя файла> ], <номер ученика> )

**[Функция]** Принимает от ученика программу на Бейсике.

**[Пример 1]** CALL RECE ("B:TEST.BAS,14)

Эта команда принимает программу на Бейсике от указанного ученика и записывает ее на диск в заданный файл. Команда не стирает содержимое памяти преподавателя и может быть использована в программном режиме. Формат, в котором записывается файл такой же, как и при команде SAVE Бейсика. В этом примере программа ученика номер 14 записывается в файл "TEST.BAS". Если файл с таким именем уже есть на диске, старый будет стерт и взаменнего будет записана программа ученика. Если программа ученика в момент подачи команды работала, она будет остановлена и высветится сообщение "Wait" (ждите), а после окончания приема ее преподавателем - "OK".

**[Пример 2]** CALL RECE (,1)

Эта команда пересылает программу ученика в память преподавателя, стирая при этом программу находящуюся в памяти. Если команда используется в коммандном режиме, то старая программа прерывается в строке с этой командой, стирается, а взаменнее принимается программа ученика, после чего, высвечивается "OK".
Во время приема на экране ученика высвечивается сообщение "Wait", а после чего - "OK". В этом примере программа ученика номер 1 пересылается в текстовом области памяти перполавателя.

# SNDMAIL

**[Формат]** **CALL SNDMAIL** [ ( [ <номер ученика> ] ) ]

**[Функция]** Передает содержимое почтового ящика.

Почтовый ящик - это область памяти для передаваемой информации, которая резервируется как в памяти ученика, так и преподавателя. В каждом компьютере есть почтовый ящик для передачи и для приема. Под каждую из этих областей выделяется по 256 байт. Адреса почтовых ящиков записаны в рабочей области (смотрите соответствующий раздел). Передача заключается в пересылке содержимого передаточного ящика преподавателя в приемный ящик ученика. Почтовый ящик, таким образом - область для передачи данных. Как Вы ее будете использовать - полностью зависит от Вас.

**[Пример 1]**

```
10 FOR L=1 TO 5
20 CALL SNDM (L)
30 NEXT 1
```

Эта команда копирует содержимое передающего ящика преподавателя в принимающие ящики учеников. В этом примере - копируется ученикам 1-5.

**[Пример 2]**

```
CALL SNDM
```

Когда номер ученика пропущен, содержимое ящика преподавателя пересылается в ящики всех учеников, как в этом примере.

# RCVMAIL

**[Формат]** **CALL RCVMAIL** (<номер ученика>)

**[Функция]** Принимает содержимое ящика.

**[Пример]**

```
CALL RCVM (5)
```

Эта команда принимает содержимое передающего ящика ученика и записывает его в приемный ящик преподавателя. В этом примере - у пятого ученика.

# SNDCMD

**[Формат]** **CALL SNDCMD** ( <команда> [, [ <номер ученика> ] ] )

**[Функция]** Передает команду на Бейсике.

**[Пример 1]** CALL SNDC ("KEY OFF",3)

Эта команда передает указанному ученику команду на Бейсике и выполняет ее. В конце команды всегда добавляется код перевода каретки (CR). Если номер ученика пропущен, команда передается всем ученикам. В этом примере в результате выполнения команды, у ученика номер 3 перестают светиться функциональные клавиши.
Если команда набрана с ошибкой или это вообще не команда Бейсика, она все равно будет передана в компьютер ученика и он попытается ее выполнить. В этом случае ученик получит сообщение об ошибке. Будте внимательны, т.к. на компьютере преподавателя никакого сообщения не будет. Если во время передачи команды, у ученика работает программа, она будет остановленна, после чего передаваемая команда будет принята и выполнена.

**[Пример 2]**

```
10 A$="COLOR 15,4,7"
20 B$=CHR$(13)
30 C$="CLS"
40 CALL SNDC (A$+B$+C$)
```

Можно посылать несколько команд одновременно. В этом примере у всех учеников выполняется команда "COLOR 15,4,7", а затем - команда "CLS". При этом между отдельными командами надо вставлять код перевода каретки (CR).

# RUN

**[Формат]** **CALL RUN** [( [[<номер ученика>] [, [<номер строки>]]] )]

**[Функция]** Запускает программу на Бейсике у ученика.

**[Пример 1]** CALL RUN (1,100)

Эта команда запускает программу указнного ученика с заданного номера строки. В этом примере программа запускается с 100 строки у 1 ученика. Если в программе ученика не окажется 100 строки, эта команда вызовет ошибку, сообщение о которой появится на экране ученика. Если программа ученика была уже запущена, то она останавливается и начинается с указанной строки.

**[Пример 2]** CALL RUN

Когда номер строки опускается, программа запускается со своей первой строки. Когда опускается номер ученика, эта команда вызывает запуск программ на Бейсике у всех учеников.

# STOP

**[Формат]** **CALL STOP** [ ( <номер ученика> ) ]

**[Функция]** Останавливает программу на Бейсике.

**[Пример]** CALL STOP (5)

Когда у указанного ученика запущена программа на Бейсике, эта команда останавливает ее. Когда номер ученика не указан, эта команда вызывает остановку программ на Бейсике у всех учеников. В этом примере останавливается программа у ученика номер 5. Действия вызываемые этой командой аналогичны действию CTRL-STOP. Например если ученик находится в режиме ожидания прямого ввода, то при выполнении этой команды будет переведена строка, если в момент передачи команды у ученика выполняется команда LIST, то вывод текста программы будет прерван, и т. д.

# PON

**[Формат]** **CALL PON**

**[Функция]** Начинает упорядоченный опрос.

**[Пример]** CALL PON

Преподователь использует эту команду для начала упорядоченного опроса учеников. Помните, что если этот опрос не активирован, невозможно определить кто из учеников подключен к сети, а также невозможна какая-либо связь между учениками (TALK и т.д.). При инициализации сети, в том числе при включении компьютера преподавателя, сеть устанавливается в режим опроса.

# POFF

**[Формат]** **CALL POFF**

**[Функция]** Прекращает упорядоченный опрос.

**[Пример]** CALL POFF

Эта команда прекращает упорядоченный опрос учеников. В этом режиме ученики не могут посылать сообщения при помощи команды TALK, а также невозможна всякая связь между учениками.

# ONLINE

**[Формат]** **CALL ONLINE**

**[Функция]** Включает ученика в сеть.

**[Пример]** CALL ONLINE

Эта команда включает ученика в сеть после того, как он был отключен от нее при помощи команды CALL OFFLINE. Если ученик не включен в сеть какая-либо связь с ним невозможна. При включении компьютер подключается к сети.

# OFFLINE

**[Формат]** **CALL OFFLINE**

**[Функция]** Отключает ученика от сети.

**[Пример]** CALL OFFLINE

Эта команда отключает ученика от сети. Когда компьютер отключен от сети, он не может принять команду от преподавателя. Таким образом, эта команда используется, когда ученик хочет работать сам, без каких-либо прерываний от преподавателя. Чтобы снова включится в сеть, ученик должен использовать команду CALL ONLINE.

## Базовые функции ввода-вывода сети.

Данная сеть имеет систему стандартных функций ввода-вывода (BIOS), включающую в себя различные функции.
Обращаясь прямо к ней можно работать с сетью из программ на машинном языке в MSX-DOS.

Как обращаться к BIOS.

Для обращения к BIOS необходимо поместить номер функции в регистр C, первый адрес блока параметров в регистровую пару DE, а затем вызвать подпрограмму (CALL) по адресу F989H. Перед вызовом подпрограммы по этому адресу, неоходимо обратиться к подпрограмме по адресу F98EH, которая инициализирует сеть в MSX-DOS. Без инициализации, вызов любой стандартной функции вызовет ошибку.

Для окончания работы с сетью обратитесь к подпрограмме по адресу F984H. Этот обращение возможно только если система имеет сетьевое ПЗУ. Вы можете узнать, есть ли оно у Вас, проверив находится ли по адресу, который Вы вызываете, rst30 (F7H) или проверив идентификатор "RNT", который должен находится по адресам 4040H - 4042H в сетьевом ПЗУ.

Таким образом:

| | |
|---|---|
| Инициализация сети | F98EH |
| Обращение к BIOS | F989H |
| Конец работы с сетью | F984H |
| Номер функции в регистре C | 01H-1AH |
| Начальный адрес блока параметров в регистровой паре DE | начальный адерес блока параматров из 8 байтов. |

Помните, что некоторые части BIOS используются только преподавателем, некоторые - только учеником, а некоторые как тем, так и другим. (Не забудьте также о том, что во время обращения ученика к диску с помощью сетьевого MSX-DOS BIOS возможны паузы в работе системы).

## Список функций сетьевой MSX-DOS BIOS.

| Номер | и название | Функция |
|---|---|---|
| 0 | **INIT** | : Инициализация сеть. |
| 1 | **INTON** | : Разрешение прерываний по сети. |
| 2 | **INTOFF** | : Запрещение прерываний по сети. |
| 3 | **PON** | : Начало упорядоченного опроса учеников. |
| 4 | **POFF** | : Конец упорядоченного опроса учеников. |
| 6 | **WHO** | : Проверка номера. |
| 8 | **SHEX** | : Пересылка программы на машинном языке. |
| 9 | **SHEXS** | : Пересылка изображения (данных видеопамяти). |
| 11 | **RHEX** | : Прием программы на машинном языке. |
| 12 | **RHEXS** | : Прием изображения (данных видеопамяти). |
| 13 | **MESS** | : Передача сообщения от преподавателя ученику. |
| 14 | **TALK** | : Передача сообщения от ученика преподавателю. |
| 15 | **SNDM** | : Передача содержимого почтового ящика. |
| 16 | **RNDM** | : Прием содержимого почтового ящика. |
| 17 | **POKE** | : Запись в память. |
| 18 | **PEEK** | : Чтение из памяти. |
| 19 | **SNDCMD** | : Передача команды. |
| 22 | **BREAK** | : Передача кода остановки программы. |
| 23 | **CHECK** | : Проверка кто подключен к сети. |
| 24 | **ENDNET** | : Конец работы с сетью. |
| 25 | **ENACOM** | : Разрешение связи между учениками. |
| 26 | **DISCOM** | : Запрещение связи между учениками. |

Функции 5, 7, 10, 20, 21 не используются в MSX-DOS, т.к. они предназначены для работы в Бейсике.

| | | |
|---|---|---|
| 5 | HELP | : Высвечивает сетьевые команды Бейсика. |
| 7 | SEND | : Передача программы на Бейсике. |
| 10 | RECV | : Прием программы на Бейсике. |
| 20 | RUN | : Запуск программы на Бейсике. |
| 21 | STOP | : Остановка программы на Бейсике. |

## ФУНКЦИЯ 1 : INTON

Преподаватель и ученик

**[Функция]** Разрешает прерывания по сети.

**[Код]** C = 01H

**[Возвращает]** ----------

Эта функция разрешает прерывания по работе с сетью. Без этой команды какая-либо связь в классе невозможна. После ее вызова компьютер готов к работе с сетью.

## ФУНКЦИЯ 2 : INTOFF

Преподаватель и ученик

**[Функция]** Запрещает прерывания по сети.

**[Код]** C = 02H

**[Возвращает]** ----------

Эта функция запрещает прерывания по работе с сетью. Так как связь в классе поддерживается через прерывания процессора, когда они запрещены, какая-либо связь невозможна. Эта функция используется, если при работающей сети возникает необходимость ее отключить. Для того, чтобы снова начать работать с сетью, используйте Функцию 1 (INTON).

## ФУНКЦИЯ 3 : PON

Преподаватель

**[Функция]** Начинает упорядоченный опрос учеников.

**[Код]** C = 03H

**[Возвращает]** ----------

Эта функция начинает упорядоченный опрос учеников. Без него невозможно правильно определить, кто из учеников подключен к сети (т.е. правильность работы Функции 24 CHECK не гарантируется), невозможен ткаже прием сообщений от учеников (Функция TALK). Помните, вызов Функции 1 (INTON), не означает что опрос начат, т.е. при ее вызове опрос отключен и его надо активировать отдельно.

## ФУНКЦИЯ 4 : POFF

Преподаватель

**[Функция]** Прекращает упорядоченый опрос учеников.

**[Код]** C = 04H

**[Возвращает]** ----------

Прекращает упорядоченный опрос учеников. Для его начала используйте Функцию 3 (PON).

## ФУНКЦИЯ 5 : HELP

Преподаватель и ученик

**[Функция]** Высвечивает список сетьевых команд Бейсика.

**[Код]** C = 05H

**[Возвращает]** ----------

Эта функция выводит на экран список сетьевых команд Бейсика. Этот список различен для преподавателя и ученика (см. выше). Для ученика эта функция высвечивает также его номер. Экран должен быть в текстовом режиме.

## ФУНКЦИЯ 6 : WHO

Преподаватель и ученик

**[Функция]** Возвращает номер Вашего компьютера.

**[Код]** C = 06H

**[Возвращает]** A = номер преподавателя или ученика
00H = преподаватель
01H - 0FH = ученики

Эта функция возвращает в регистре A, номер Вашего компьютера, установленный переключателем в сетьевом блоке. В классе не может быть несколько компьютеров с одинаковыми номерами, если это не так, воспользуйтесь переключателями в сетьевом блоке.

**[Функция]** Передает программу на Бейсике.

**[Код]**
C = 07H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес блока контроля файлов (FCB)

**[Возвращает]**
CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция читает программу на Бейсике с диска и пересылает ее ученику или всем ученикам. Файл с этой программой определяется в блоке контроля файлов (FCB) и должен быть открыт до вызова этой функции. Ученик должен находиться в режиме Бейсика.

**[Функция 1]** Передает программу на машинном языке из памяти.

**[Код]**

C = 08H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2 = 0
PAR 3-4 = начальный адрес ученика
PAR 5-6 = начальный адрес преподавателя
PAR 7-8 = конечный адрес преподавателя

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция пересылает программу на машинном языке или данные из памяти преподавателя в память ученика или всех учеников.

**[Функция 2]** Передает файл на машинном языке.

**[Код]**

C = 08H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2 = любое значение кроме 0
PAR 3-4 = начальный адрес у ученика (если указать0FFFFH - будет начальный адрес файла)
PAR 5-6 = адрес блока контроля файлов (FCB)

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция читает программу на машинном языке с диска и пересылает ее ученику или всем ученикам. Файл с этой программой определяется в блоке контроля файлов (FCB) и должен быть открыт до вызова этой функции.

**[Функция 1]** Передает изображение (данные видеопамяти).

**[Код]**

C = 09H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2 = 0
PAR 3-4 = начальный адрес у ученика
PAR 5-6 = начальный адрес у преподавателя
PAR 7-8 = конечный адрес у преподавателя

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция пересылает изображение (данные видеопамяти) из видеопамяти преподавателя в видеопамять ученика или всех учеников.

**[Функция 2]** Передает файл на машинном языке.

**[Код]**

C = 09H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2 = любое значение кроме 0
PAR 3-4 = начальный адрес ученика (если указать 0FFFFH - будет начальный адрес файла) - будет начальный адрес файла)
PAR 5-6 = адрес блока контроля файлов (FCB)

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция читает изображение (данные видеопамяти) с диска и пересылает его ученику или всем ученикам. Файл с этими данными определяется в блоке контроля файлов (FCB) и должен быть открыт до вызова этой функции.

## ФУНКЦИЯ 10 : RECEIVE

**[Функция]** Принимает программу на Бейсике.

**[Код]**

C = 0AH
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес блока контроля файлов (FCB)

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция принимает программу на Бейсике у указанного ученика и записывает ее на диск. Файл для программы на Бейсике должен быть определен в блоке контроля файлов (FCB) до вызова этой функции. Ученик должен быть в режиме Бейсика.

**[Функция 1]** Принимает программу на машинном языке в память.

**[Код]**

C = 0BH
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2 = 0
PAR 3-4 = начальный адрес ученика
PAR 5-6 = конечный адрес ученика
PAR 7-8 = начальный адрес преподавателя

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция принимает программу на машинном языке или данные из памяти ученика в память преподавателя.

**[Функция 2]** Передает программу на машинном языке в файл на диске.

**[Код]**

C = 0BH
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2 = любое значение кроме 0
PAR 3-4 = начальный адрес ученика
PAR 5-6 = конечный адрес ученика
PAR 7-8 = адрес блока контроля файлов (FCB)

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция передает программу на машинном языке из памяти ученика и записывает ее на диск. Файл для этой программы должен быть определен в блоке контроля файлов (FCB) и открыт до вызова этой функции.

**[Функция 1]** Принимает изображение (данные видеопамяти).

**[Код]**

C = 0CH
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2 = 0
PAR 3-4 = начальный адрес ученика
PAR 5-6 = конечный адрес ученика
PAR 7-8 = начальный адрес преподавателя

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция принимает изображение (данные видеопамяти) из видеопамяти ученика в видеопамять преподавателя.

**[Функция 2]** Передает изображение (данные видеопамяти) в файл на диске.

**[Код]**

C = 0CH
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2 = любое значение кроме 0
PAR 3-4 = начальный адрес ученика
PAR 5-6 = конечный адрес ученика
PAR 7-8 = адрес блока контроля файлов (FCB)

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция передает изображение (данные видеопамяти) ученика и записывает его на диск. Файл для этой программы должен быть определен в блоке контроля файлов (FCB) и открыт до вызова этой функции.

**[Функция]** Передает сообщение от учителя одному или нескольким ученикам.

**[Код]**
C = 0DH
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес памяти, начиная с которого записано сообщение
PAR 4-5 = длина сообщения

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция передает сообщение учителя одному или всем ученикам. Если ученик находится в режиме Бейсика, сообщение будет выведено на экран. Если же ученик в MSX-DOSe сообщение выведено не будет и для его обработки необходима специальная подпрограмма. Ее работа возможна, так как при принятии сообщения, выставляется специальный флаг, который и должен обрабатываться подпрограммой.

## ФУНКЦИЯ 14 : TALK

**[Функция]** Передает сообщение ученика учителю.

**[Код]**

C = 0EH

DE = адрес блока параметров

PAR 1 = номер ученика

00H = преподаватель

01H - 0FH = номер отдельного ученика

PAR 2-3 = адрес памяти, начиная с которого записано сообщение

PAR 4-5 = длина сообщения

**[Возвращает]**

A = 00H - сообщение записано нормально

FFH - осталось старое сообщение

CARRY ON ошибка

Если ученикполучил разрешениеучаствовать всвязи посети, тоон может посылать сообщенияпреподавателю идругим ученикам. Еслнразрешение недано,то сообщенияот негомогут посылатьсятолько преподавателю.
(Для стираниясообщений нажимаютна любуюклавишу.)

## ФУНКЦИЯ 15: SNDMAIL

Преподаватель

**[Функция]** Передает содержимое почтового ящика.

**[Код]**
C = 0FH
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика

**[Возвращает]**
CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция копирует содержимое преподавательского почтового ящика для передачи, в приемный почтовый ящик указанного или всех учеников. Изначально длина ящика равна 256 байтам, но она может быть изменена, указанием новой длины в рабочей области. Будьте внимательны, так как если длина передающего ящика преподавателя больше, чем приемного ящика ученика, то часть информации будет потеряна. Использование почтовых ящиков полностью зависит от Вас.

## ФУНКЦИЯ 16 : RCVMAIL

Преподаватель

**[Функция]** Принимает содержимое почтового ящика.

**[Код]**
C = 10H
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика

**[Возвращает]**
CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция принимает содержимое передающего почтового ящика ученика и записывает его в приемный ящик преподавателя. Количество информации зависит от длины передающего ящика ученика. Будьте внимательны, так как если длина его передающего почтового ящика больше, чем приемного ящика преподавателя, часть информации будет потеряна.

## ФУНКЦИЯ 17 : POKE

**[Функция]** Записывает в память.

**[Код]**

C = 11H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес ячейки
PAR 4 = значение
PAR 5 = выбор
00H - память ученика
01H - сетьевая память ученика
02H - Ваша сетьевая память

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция, в зависимости от параметра PAR5, записывает указанное значение в обычную или сетьевую память отдельного или всех учеников, или (как для преподавателя, так и для ученика) записывает это значение в сетьевую память Вашего компьютера. Функция не проверяет в какое именно место памяти она пишет, и всегда выполняется нормально если только по указанному адресу не находиться ПЗУ или вообще отсутсвует память.

**[Функция]** Читает из памяти.

**[Код]**

C = 12H
DE = адрес блока параметров
PAR 1 = номер ученика
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес ячейки
PAR 5 = выбор
00H - память ученика
01H - сетьевая память ученика
02H - Ваша сетьевая память

**[Возвращает]** A = значение записанное по данному адресу.

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция, в зависимости от параметра PAR5, читает число, записанное по указанному адресу, из обычной или сетьевой памяти отдельного или всех учеников, или (как для преподавателя, так и для ученика) читает это значение из сетьевой памяти Вашего компьютера. Эта функция всегда выполняется нормально, кроме тех случаев, когда память по указанному адресу отсутствует.

## ФУНКЦИЯ 19 : SNDCMD

Преподаватель

**[Функция]** Передает команду на Бейсике.

**[Код]**

C = 13H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2-3 = адрес памяти, начиная с которого записана команда
PAR 4-5 = длина команды

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция передает одному или всем ученикам команду на Бейсике и выполняет ее. После приема команды у ученика ее в конце прибавляется код возврата каретки (CR). Ученик(-и) должны быть в режиме Бейсика (эта команда может и не быть командой Бейсика, но если интерпритатор ее не поймет, у ученика появится сообщение об ошибке).

## ФУНКЦИЯ 20 : RUN

Преподаватель

**[Функция]** Запускает программу на Бейсике.

**[Код]**

C = 14H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика
PAR 2-3 = номер стартовой строки (если указан 0FFFFH - первая строка программы)

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция запускает программу на Бейсике отдельного или всех учеников. Ученик(-и) должны быть в режиме Бейсика.

## ФУНКЦИЯ 21 : STOP

Преподаватель

**[Функция]** Останавливает программу на Бейсике.

**[Код]**
C = 15H
DE = адрес блока параметров
PAR 1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция производит такое же действие, как если бы ученик нажал CTRL-STOP. Если ученик находится в режиме Бейсика происходит остановка программы или переход на начало следующей строки (в командном режиме). Если ученик находится в MSX-DOSe, то команда игнорируется.

## ФУНКЦИЯ 22 : BREAK

Преподаватель

**[Функция]** Передает код остановки.

**[Код]** C = 16H

**[Возвращает]** ----------

Эта функция посылает ученикам код остановки. Когда их компьютеры получают этот код во время связи, они сбрасывают все полученные до этого данные и становятся готовыми к приему.

**[Функция]** Проверяет, кто из учеников подключен к сети.

**[Код]** C = 17H

**[Возвращает]** HL = побитовая информация, кто подключен к сети.

```
              H                    L
BIT   F E D C B A 9 8      7 6 5 4 3 2 1 0
      * * * * * * * *      * * * * * * * *
      :                          0: включен (ONLINE)
      :                          1: отключен (OFFLINE)
      :.............................→ всегда 0
```

DE = побитовая информация, кому разрешена работа с сетью.

```
              D                    E
BIT   F E D C B A 9 8      7 6 5 4 3 2 1 0
      * * * * * * * *      * * * * * * * *
      :                          0: разрешена (ENACOM)
      :                          1: запрещена (DISCOM)
      :.............................→ всегда 0
```

Эта функция проверяет, кто из учеников подключен к сети, кому из них разрешена связь с другими учениками. То что ученик отключен от сети означает, что его компьютер выключен или физически отключен от сети. Эта информация обновляется в каждом цикле упорядоченного опроса, если же опрос выключен, хранится последняя, полученная до его отключения информация. Информация о том, кому разрешена связь с другими учениками, зависит только от функций ENACOM и DISCOM.

## ФУНКЦИЯ 24 : ENDNET

Преподаватель

**[Функция]** Заканчивает работу сети.

**[Код]** C = 18H

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка

Эта функция аналогична вызову подпрограммы по адресу F98EH, заканчивающей работу с сетью. Выключение сети бывает необходимо для работы с некоторыми прикладными программами, вызов которых невозможен при работающей сети. Для включения сети используйте вызов подпрограммы по адресу F98EH.

## ФУНКЦИЯ 25 : ENACOM

Преподаватель

**[Функция]** Разрешает связь ученикам.

**[Код]**
C = 19H
DE = адрес блока параметров
PAR1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика

**[Возвращает]** CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция разрешает отдельному или всем ученикам связь между собой.

**[Функция]** Запрещает связь ученикам.

**[Код]**

C = 1AH
DE = адрес блока параметров
PAR1 = номер ученика
00H = всем ученикам
01H - 0FH = номер отдельного ученика

**[Возвращает]**

CARRY OFF - функция выполнена нормально.
CARRY ON - ошибка.

Эта функция разрешает отдельному или всем ученикам связь между собой.

# Classroom Network BIOS (for MSX-CP/M)

The classroom network has a basic input/output system (BIOS) with a variety of functions. It is possible to carry out telecommunications with machine language programs on the MSX by accessing this BIOS directly.

## How to access the BIOS

The BIOS is accessed by setting the function number in the C register, the lead address for the parameters in the DE register, and calling address 05H. It is necessary to call the MSX-CP/M network initialization routine at address F97FH before calling address 05H. Unless this is done, normal function calls will result in errors.

To end the machine language network, call address F984H. These call addresses can only be used when the system has the classroom network ROM. You can check whether the system has the classroom network ROM by checking whether or not the call addresses start with F7H or by checking whether the 3-byte ID "RNT" is written into addresses 4040H through 4042H of the classroom network ROM.

| | |
|---|---|
| Network initialization | = address F97FH |
| Network BIOS entry | = address 05H |
| Network ending | = address F984H |
| The function numbers for the C register | = 01H – 12H |
| The parameter address for the DE register | = the start address for an 8-byte parameter |

Note that some parts of the BIOS can only be used by the teacher, some parts can only be used by the students, and some parts can be used by either the teacher or the students.

Expansion BDOS calls are made exactly the same way as CP/M BDOS calls with Call 5 by setting the parameters in the registers.

— Set the function number in the C register.
— Set the other party's drive number in the B register.
— Set the FCB address in the DE register.
(FCBs for CP/M Version 2.2 are 36 bytes long, but expansion BDOS FCBs are 37 bytes long.)

— The values returned are set in the H and AF registers.

- The A register shows the BDOS status.
- CARRY OFF Function ended normally.
  CARRY ON Network error
- H register

| | |
|---|---|
| FFH | Normal end |
| FEH | Disk lock error |
| FDH | BDOS busy |
| FCH | Network error |
| FBH | Broadcast error |
| 0CH | Disk error |
| 00H | |

Expansion BDOS calls are distinguished from regular BDOS calls by setting Bit 8 of the C register to 1.

| C register | Function |
|---|---|
| 8FH | NET-OPEN-FILE |
| 90H | NET-CLOSE-FILE |
| 91H | NET-SEARCH-FIRST |
| 92H | NET-SEARCH-NEXT |
| 93H | NET-DELETE-FILE |
| 94H | NET-READ-FILE |
| 95H | NET-WRITE-FILE |
| 96H | NET-MAKE-FILE |
| 97H | NET-RENAME-FILE |
| B3H | NET-SEARCH-END |
| B4H | NETWORK-BIOS |
| B5H | ERROR-CONTROL |
| B6H | NETWORK-CONTROL |

1) 8FH-B3H are disk functions that return the same value in the A register as the regular BDOS functions. For more details, refer to the CP/M manual.

2) During the time that the functions for 8FH ~ B3H are operating (Directory SEARCH-FIRST to SEARCH-END), other BIOS functions are "busy" and operation is not possible.

3) The purpose of the B5H error-control function is to suppress display of the error message when a disk error occurs in CP/M.
(If the E register is 00H, error messages are displayed; if the E register is 01H, error messages are not displayed.)

4) The B6H network-control function is used to have the CP/M system set the Netend flag.
(If the E register is 00H, the network is active; if the E register is 01H, the network is ended.)

The MSX-CP/M network BIOS is called by setting the parameters in the registers with the MSX CP/M extension with the 5H call. The format is that B4H is set in the C register, the sub-function in the B register, and the lead address for the parameters in the DE register.

— Function No. Set B4H in the C register.
— Sub-function No. Set in the B register.

Lead address for the parameters Set in the DE register.

The values returned are set in the AF, HL, and DE registers.

B register

| | | |
|---|---|---|
| 01H | INTON | Enables hardware interrupts through telecommunications. |
| 02H | INTOFF | Disables hardware interrupts through telecommunications. |
| 03H | PON | Starts polling of the students. |
| 04H | POFF | Stops polling of the students. |
| 05H | CHECK | Checks which students are connected to the network. |
| 06H | WHO | Checks the teacher or student number. |
| 07H | ENACOM | Enables telecommunications between students. |
| 08H | DISCOM | Disables telecommunications between students. |
| 09H | POKE | Writes to memory. |
| 0AH | PEEK | Reads to memory. |
| 0BH | MESS | Sends messages from the teacher to students. |
| 0CH | TALK | Sends messages from students to the teacher. |
| 0DH | SNDCMD | Sends a command. |
| 0EH | SEND-VRAM-REGISTER | Sends the VRAM register. |
| 0FH | RECEIVE-VRAM-REGISTER | Receives the VRAM register. |
| 10H | SEND-VRAM-DATA | Sends VRAM data. |
| 11H | RECEIVE-VRAM-DATA | Receives VRAM data. |
| 12H | BREAK | Sends a break code. |

(Teacher and students)

**Function 1 INTON**

**[Function]** Enables hardware interrupts through telecommunications.

**[Path]** B = 01H
C = B4H

**[Parameters returned]** None

This function enables hardware interrupts through telecommunications. Classroom network telecommunications are impossible unless this function is executed. When Netinit is executed, the teacher computer and the student computers are put into Inton mode.

(Teacher and students)

**Function 2 INTOFF**

**[Function]** Disables hardware interrupts through telecommunications.

**[Path]** B = 02H
C = B4H

**[Parameters returned]** None

This function disables hardware interrupts through telecommunications. Since telecommunications in the classroom network are carried out by processing interrupts to the CPU, when hardware interrupts are disabled no telecommunications are possible whatsoever. This function is used when it would be a problem if telecommunications took place.
When a student or the teacher is in Intoff mode and wishes to resume telecommunications through the classroom network, he or she executes Function 1, Inton.

**Function 3 PON**

**[Function]** Starts polling of the students

**[Path]** B = 03H
C = B4H

**[Parameters returned]** None

This function starts polling of the students. Unless the students are polled, it is impossible to check which students are connected to the classroom network, the teacher can not receive messages from the students, and telecommunications between the students are impossible. When Netint is executed, polling is set on.



**Function 4 POFF**

**[Function]** Stops polling of the students

**[Path]** B = 04H
C = B4H

**[Parameters returned]** None

This function stops polling of the students. To restart polling, execute Function 3, Pon.

**Function 5 CHECK**

**[Function]** Checks which students are connected to the network.

**[Path]** B = 05H
C = B4H

**[Parameters returned]** HL = bit map of the connections to the network.

```
                H              L
BIT         FEDCBA98       76543210
            ********       ********
            |                                0: ONLINE
            |                                1: OFFLINE
            |------------------------------> always 0
```

DE = bit map of the students for whom telecommunications are enabled.

```
                D              E
BIT         FEDCBA98       76543210
            ********       ********
            |                                0: ENACOM
            |                                1: EISCOM
            |------------------------------> always 0
```

This function checks which students are connected to the network and which of those telecommunications are enabled for. When a student is not connected to the network, this means that the student's computer is powered off or in some other way physically not connected or that hardware interrupts have been disabled. The connection bit map is updated every time the teacher polls the students, but if polling is stopped, the connection bit map from before the polling is retained. The telecommunications enabled bit map is only updated by the Enacom and Discom functions.

## Function 6 WHO

**[Function]** Check a teacher or student number.

**[Path]** B = 06H
C = B4H

**[Parameters returned]** A = the teacher or student number
00H = teacher
01H-0FH = students

This function places the value set by the DIP switch in the network unit in Register A. The value is 00H for the teacher or 01H-0FH for a student. More than one student can not have the same student number. If they do, reset the DIP switches in the network unit.

## Function 7 ENACOM

**[Function]** Enables student telecommunications.

**[Path]**
B = 07H
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student

**[Parameters returned]**
CARRY OFF Function ended normally.
CARRY ON Error

This function enables telecommunications for the student whose number is specified.



## Function 8 DISCOM

**[Function]** Disables student telecommunications

**[Path]**
B = 08H
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student

**[Parameters returned]**
CARRY OFF Function ended normally.
CARRY ON Error

This function disables telecommunications for the student whose number is specified.

## Function 9 POKE

**[Function]** Writes numbers to network memory and student memory.

**[Path]**

B = 09H
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 2-3 = write address
PAR 4 = write value
PAR 5 = Option
00H = Student memory
01H = Student network memory
02H = Network memory

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function writes the numerical value into the specified address of the student memory or network memory or into the network memory. This function does not check whether the specified address is in memory. This function ends normally even if the specified address is in ROM or does not exist at all.

## Function 10 PEEK

**[Function]** Reads a number from network or student memory

**[Path]**

B = 0AH
C = B4H
DE = lead parameter address
PAR 1 = student number
01H-0FH = that one student
PAR 2-3 = write address
PAR 5 = Option
00H = Student memory
01H = Student network memory
02H = Network memory

**[Parameters returned]**

A = value written at the specified address
CARRY OFF Function ended normally.
CARRY ON Error

This function reads the numerical value from the specified address of the student's memory or into the network memory. This function ends normally even if nothing at all is connected to the specified address.

## Function 11 MESSAGE

**[Function]** Sends a message from the teacher to one or all of the students.

**[Path]**

B = 0BH
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 2-3 = the lead address for the memory in which the message is written
PAR 4-5 = the length of the message

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function sends a message from the teacher to one or all of the students. The message sent can be up to 567 characters long, but when displayed on the 24th row of the student's screen, the length of the message that can be displayed is determined by the width of the current screen. If the student is not in text mode, the message is displayed when the student goes into text mode.
(To erase the message, input a key.)

## Function 12 TALK

**[Function]** Sends a message from a student.

**[Path]**

B = 0CH
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = teacher
01H-0FH = that one student
PAR 2-3 = the lead address for the memory in which the message is written
PAR 4-5 = the length of the message

**[Parameters returned]**

A = 00H message stored successfully
= FFH message left from the last time

CARRY ON error

If telecommunications are not enabled for the student, he or she can only send messages to the teacher. If telecommunications are enabled for the student, he or she can send also messages to other students. (To erase the message, input a key.)

## Function 13 SNDCMD

**[Function]** Sends a command.

**[Path]**

B = 0DH
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 2-3 = lead address for the memory the command is written in
PAR 4-5 = command length

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function sends a command to one or all students and runs it. The student receiving this command enters CTRL-U before receiving it and adds a carriage return code to the end of the command after receiving it.

## Function 14 SNDVR

**[Function]** Sends the VRAM register.

**[Path]**

B = 0EH
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 2 = option
00H = current pallet
FFH = pallet specification
PAR 3-4 = register and pallet data lead address
PAR 5-6 = VRAM start address
PAR 7-8 = VRAM end address

**[Parameters returned]** CARRY OFF Function ended normally.
CARRY ON Error

This function sends the VRAM register and pallet data. This VRAM register and pallet data change the screen mode for the receiver. Note that the VRAM start and end addresses are used when VRAM data is sent with Function 16.
(A student for which communication is enabled by Function 7 ENACOM can use up to Functions 14 ~ 17.)

(Specification for PAR 2 option data)

PAR 2 = 00H
PAR 3-4 →
Address

| Register Data | 8 BYTE |
|---|---|

PAR 2 = FFH
PAR 3-4 →
Address

| Register Data | 8 BYTE |
|---|---|
| Pallet Data | 32 BYTE |

## Function 15 RECVR

**[Function]** Receives the VRAM register.

**[Path]**

B = 0FH
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 2 = option
00H = current pallet
FFH = pallet specification
PAR 3-4 = register and pallet data lead address
PAR 5-6 = VRAM start address
PAR 7-8 = VRAM end address

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function receives the VRAM register and pallet data.

## Function 16 SNDVRAM

**[Function]** Sends VRAM data.

**[Path]**

B = 10H
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 3-4 = Start address for the VRAM data buffer

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function sends 128 bytes beginning from the specified start address for the VRAM data buffer and writes it into the VRAM of the recipient starting at the VRAM start address specified in Function 14.

## Function 17 RECVRAM

**[Function]** Receives VRAM data.

**[Path]**

B = 11H
C = B4H
DE = lead parameter address
PAR 1 = student number
00H = all students
01H-0FH = that one student
PAR 3-4 = Address for the receive data buffer
PAR 5-6 = Address of the VRAM to be received

**[Parameters returned]**

CARRY OFF Function ended normally.
CARRY ON Error

This function receives 128 bytes beginning from the VRAM address specified by PAR 5-6 and sets the data into the receive buffer specified by PAR 3-4.

**Function 18 BREAK**

**[Function]** Sends a break code.

**[Path]** B = 12H
C = B4H

**[Parameters returned]** None

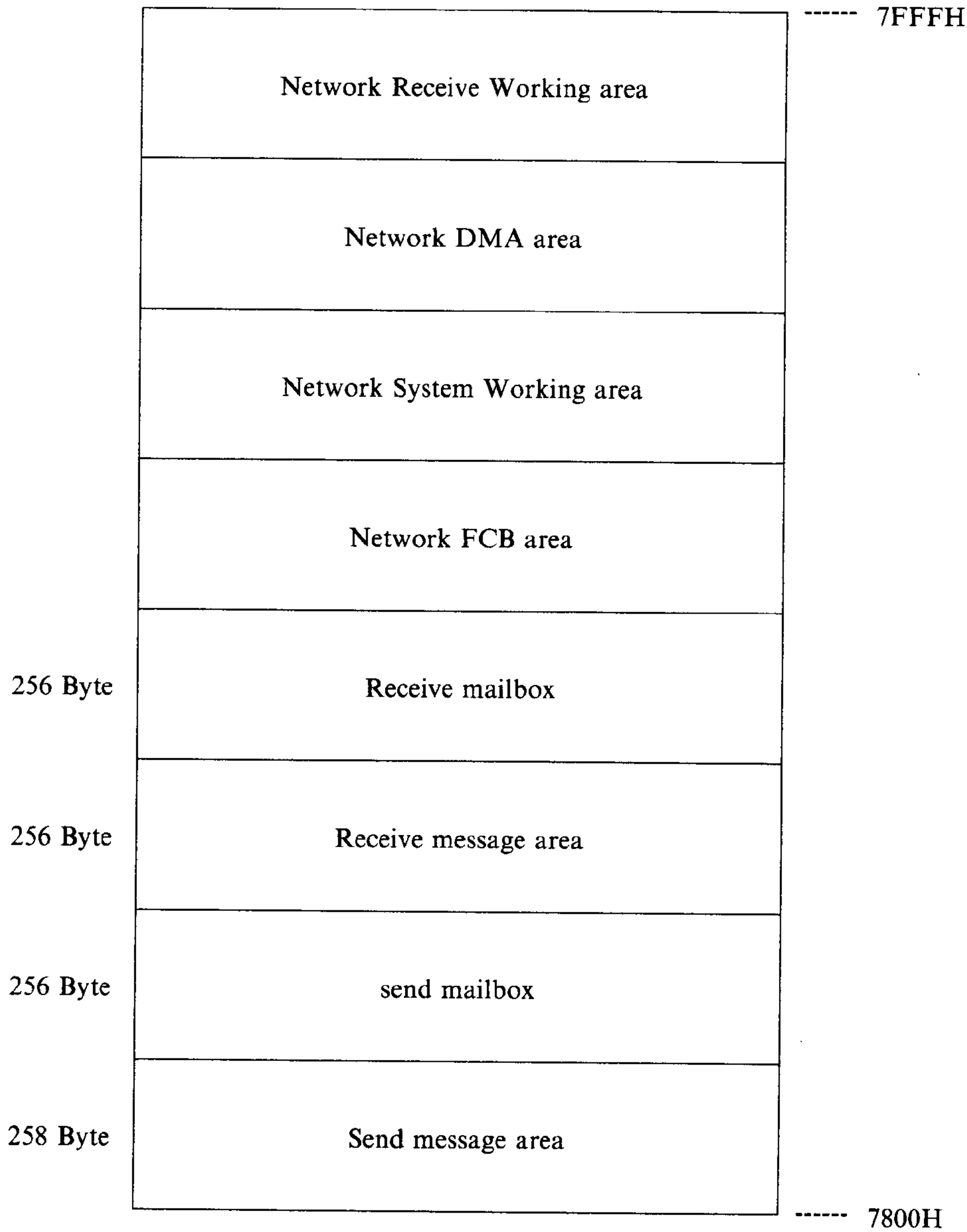
This function sends a break code to all the students. When a student's computer receives a break code during telecommunications, it throws out the data received up till then and resets the network's internal states.

# Classroom Network Work Areas

Version 3.0 of the classroom network has 2 kbytes of RAM. This section presents a map of those 2 kbytes and explains the various work areas of the network system. The notation is as shown below. The length of each work area is given in bytes.

Label (address, length) contents

Network RAM Memory Map (2 kbytes, Addresses 7800H-7FFFH, Slot 3-3)

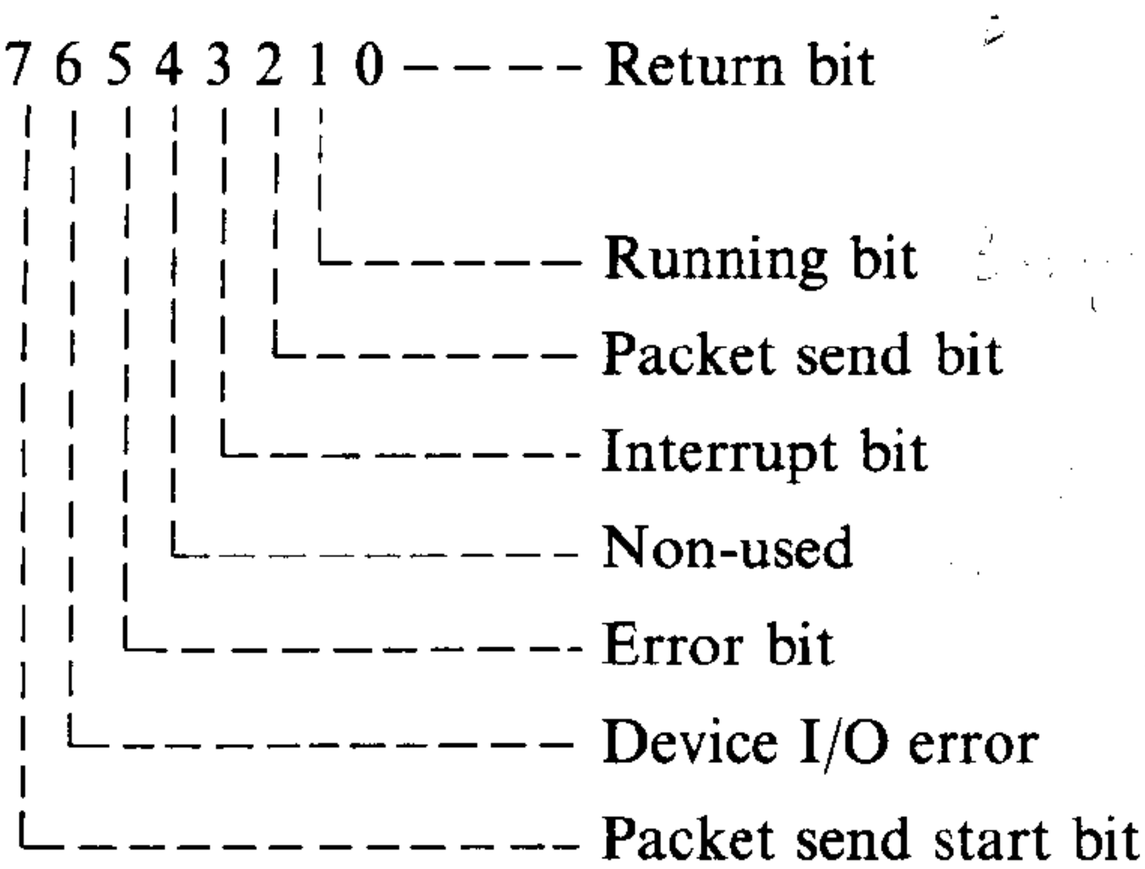


| | | • Contents |
|---|---|---|
| PAR 1 | (7C78H,1) | BIOS Parameter 1 area |
| PAR 2 | (7C79H,1) | BIOS Parameter 2 area |
| PAR 3 | (7C7AH,1) | BIOS Parameter 3 area |
| PAR 4 | (7C78H,1) | BIOS Parameter 4 area |
| PAR 5 | (7C7CH,1) | BIOS Parameter 5 area |
| PAR 6 | (7C7DH,1) | BIOS Parameter 6 area |
| PAR 7 | (7C7EH,1) | BIOS Parameter 7 area |
| PAR 8 | (7C7FH,1) | BIOS Parameter 8 area |
| SMSADR | (7C80H,2) | Send message area address |
| RMSADR | (7C82H,2) | Receive message area address |
| SMLADR | (7C84H,2) | Send mail area address |
| RMLADR | (7C86H,2) | Receive mail area address |
| W-STATUS | (7C88H,1) | Status of network BDOS call |
| W-FCBARD | (7C89H,2) | Network PCB area address |
| W-DMAADR | (7C8BH,2) | Network DMA buffer address |
| U-FLGS | (7C8DH,1) | Net-CP/M register flag |
| U-STATUS | (7C8EH,3) | Status area, H register, and AF register for CP/M BDOS |
| U-FCBARD | (7C91H,2) | User FCB address save area |
| U-DMAADR | (7C93H,2) | User DMA buffer save area |
| BIOSA | (7C95H,2) | Disk BASIC DISK BIOS entry address |
| BIOSB | (7C97H,2) | MSX-CP/M background BDOS entry address |
| BIOSC | (7C99H,2) | BASIC call address |
| BIOSF | (7C9BH,1) | Current network system flag |

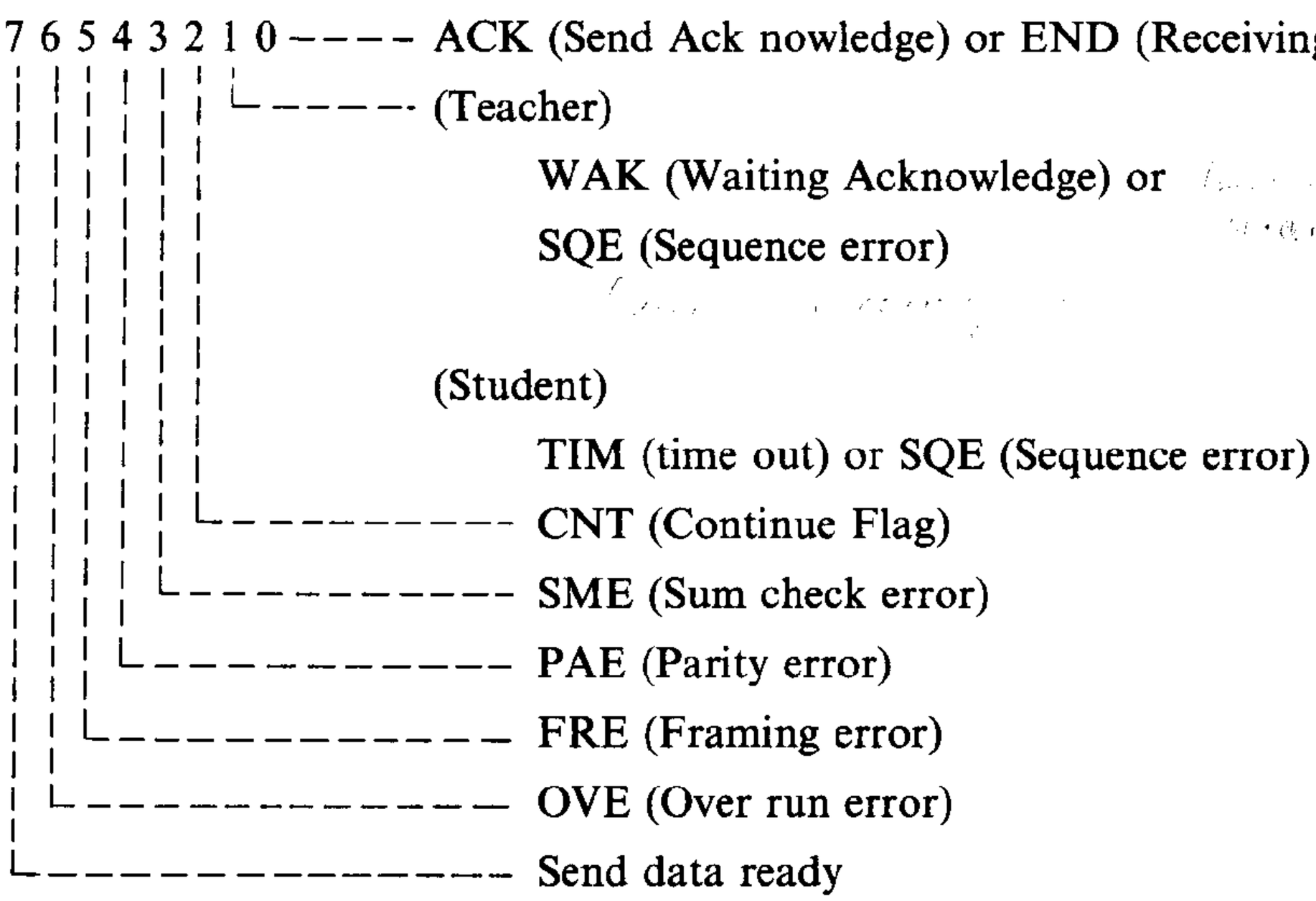
```
bit  7 6 5 4 3 2 1 0---- NULL non initilize
                 | | |
                 | | |
                 | |└---- DISK BASIC SYSTEM
                 | └----- MSX-DOS SYSTEM
                 └------- MSX-CP/M SYSTEM
```

| | | |
|---|---|---|
| DSPFLG | (7C9CH,1) | Screen display control flag |

| | | |
|---|---|---|
| ADR1 | (7C9DH,1) | Sender's number |
| ETBX | (7C9EH,1) | Packet ETB, ETX area |
| CMND | (7C9FH,1) | Packet command area |
| ADR2 | (7CAOH,1) | Receiver's number |
| LNGTH | (7CA1H,2) | Packet length area |
| BUFFR | (7CA3H,2) | Send buffer address |
| O-FLGS | (7CA5H,1) | Communications status flag |




| | | |
|---|---|---|
| O-FLGSS | (7CA6H,1) | 0 flag save area |
| O-SENDF | (7CA7H,1) | Communications right flag (teacher only) |
| O-STATUS | (7CA8H,1) | Communications control and error status |



| | | |
|---|---|---|
| TMVFL | (7CA9H,1) | T1 timer set flag |
| TM1CT | (7CAAH,2) | T1 timer counter |
| TM1FL | (7CACH,1) | T1 timer time out flag |
| TM2CT | (7CADH,2) | T2 timer counter |
| TM2FL | (7CAFH,1) | T2 timer time out flag |
| TM3CT | (7CBOH,2) | T3 timer counter |
| TM3FL | (7CB2H,1) | T3 timer time out flag |
| O-ENQF | (7CB3H,1) | Enquiry control flag (teacher only) |
| O-GENEC | (7CB4H,1) | Polling sub-interval counter |
| O-INTRV2 | (7CB5H,1) | Polling sub-interval "03H" |
| MESADR | (7CB6H,1) | Receive message control flag |
| MESLNG | (7CB7H,2) | Message receive length |
| SAV25F | (7CB9H,1) | Message display control flag |
| COMMAP | (7CBAH,2) | Telecommunications enabled bit map |
| COMMAPS | (7CBCH,2) | Telecommunications enabled bit map save (teacher only) |
| U-ERROR | (7CBEH,48) | Broadcast status area |
| R-CTRL | (7CEEH,1) | Reception data |
| R-ADR1 | (7CEFH,1) | Sender number for reception |
| R-ADR2 | (7CFOH,1) | Receiver number for reception |
| R-CMND | (7CF1H,1) | Reception command area |
| R-LNGTH | (7CF2H,2) | Reception packet length area |
| R-SUM | (7CF4H,2) | Reception sum check counter |
| R-ETBX | (7CF6H,1) | Reception ETB, ETX area |
| RECADR | (7CF7H,2) | Reception address set area |
| FILSIZ | (7CF9H,4) | Disk BASIC file size area |
| RECFB | (7CFDH,2) | Disk BASIC FCB address save area |
| KEYFLG | (7DFFH,1) | Receive key code flag |
| NXTRTN | (7DO0H,2) | Receive next routine address |
| ACTMAP | (7DO2H,2) | Active bit map |
| DOWNTL | (7DO4H,16) | Error counter table area |
| ERRMPA | (7D14H,2) | Error bit map |
| INTRV | (7D16H,1) | Polling interval "08H" |
| VDPC | (7D17H,1) | Polling interval counter |
| COUTLN | (7D18H,2) | Receive data length |
| MATCH | (7D1AH,1) | Receive command matching area |

| | | |
|---|---|---|
| COULNG | (7D1BH,2) | Receive sum check area |
| SLOT | (7D1DH,1) | Slot address |
| POLFLG | (7D1EH,1) | Polling control flag |
| MAILLN | (7D1FH,2) | Mail area length 256 bytes |
| PAKLNG | (7D21H,2) | Maximum length per packet 56 bytes |
| MYSLOT | (7D23H,1) | Current state area |
| OH-KEYI | (7D24H,5) | Old H key 1 area |
| OH-NEWS | (7D29H,5) | Old H news area |
| OH-READ | (7D2EH,5) | Old H read area |
| OH-MAIN | (7D33H,5) | Old H main area |
| H-REC | (7D38H,5) | Receive interrupt hook |
| VERSION | (7D3DH,1) | Version 3.0 "00H" |
| CALBAS | (7D3EH,5) | Calbas routine for the network |
| SYSCAL | (7D43H,12) | Syscal routine for the network |
| CALSLT | (7D4FH,5) | Calslt routine for the network |
| bb-TMPP | (7D54H,2) | Receive buffer address pointer |
| LOOPC | (7D56H,1) | Receive processing control flag |
| LOOPF | (7D57H,1) | Version number display flag for disk BASIC |
| SAVSTK | (7D58H,2) | Stack save area |
| H-ERRDS | (7D5AH,2) | Network disk error hook address |
| H-ERRDP | (7D5CH,2) | Network Disk Error Hook Address |
| SLTS-SV | (7D5EH,1) | Slot save area |
| SLTM-SV | (7D5FH,1) | Slot save area |
| SLTI-SV | (7D6OH,1) | Slot save area |
| SLT4S-SV | (7D61H,1) | Slot save area |
| SLT4M-SV | (7D62H,1) | Slot save area |
| SLT4I-SV | (7D63H,1) | Slot save area |
| REG35-SAV | (7D64H,1) | MCS Register 35 save area |
| V-SCRMOD | (7D65H,1) | VDP screen mode area for reception |
| V-LINLEN | (7D66H,1) | VDP line length for reception 40 or 80 |
| V-REGDAT | (7D67H,8) | VRAM register save area |
| V-PLTDAT | (7D6FH,32) | VRAM pallet save area |
| V-REGDARP | (7D8FH,2) | VRAM register pointer for reception |
| V-PATBASS | (7D91H,8) | Sprite pattern save area |
| V-ATRBASS | (7D99H,4) | Sprite attribute save area |
| SPRPOAT | (7D9DH,4) | Sprite data area |
| V-SAV25A | (7DAIH,2) | 25-line VRAM address save area |
| V-SAV25D | (7DA3H,80) | 25-line VRAM data save area |

In MSX-CP/M, the BIOS shown below is in the same addresses as the BASIC BIOS and the main ROM BIOS for BASIC can be used. The user can use BASIC sub-ROM BIOS calls as MSX-CP/M expansion functions.

# RDLST (000CH)

**[Function]** Selects the slot corresponding to the value of A and reads one byte of that slot's memory. When this routine is called, interrupts are disabled and remain so even after this execution of this routine is completed.

**[Input]** The slot number in A

[ F 0 0 0 E E P P ]

Basic slot number (0-3)
Expansion slot number (0-3)
"1" when specifying an expansion slot

The address for the memory read into HL

**[Output]** The value of the memory read into A

**[Registers]** AF, BC, DE

# WRSLT (0014H)

**[Function]** Selects the slot corresponding to the value of A and writes one byte into that slot's memory. When this routine is called, interrupts are disabled and remain so even after this execution of this routine is completed.

**[Input]** The slot is specified by A (in the same manner as for Rdslt). The write address is set in HL and that value in E.

**[Output]** None

**[Registers]** AF, BC, D

# CALSLT (001CH)

**[Function]** Calls out a routine for another slot (inter-slot routine)

**[Input]** The slot is specified with the upper 8 bits of the IY register (in the same manner as for Rdslt). The address being called is set in IX.

**[Output]** Depends on the routine called.

**[Registers]** Depends on the routine called.

# ENASLT (0024H)

**[Function]** Selects the slot corresponding to the value of A and enables it. When this routine is called, interrupts are disabled and remain so even after this execution of this routine is completed.

**[Input]** The slot is specified by A (in the same manner as for Rdslt).

**[Output]** None

**[Registers]** All

# CALLF (0030H)

**[Function]** Calls out a routine for another slot in the following manner:

RST 30H
n is the slot number (the same as for Rdslt).
nn is the address called.

**[Input]** As explained in the previous paragraph

**[Output]** Depends on the routine called.

**[Registers]** AF; other registers depend on the routine called.

# SUBROM (DA36H)

**[Function]** Calls the sub-ROM with an inter-slot call this way:

```
LD      IX, SUBROM ENTRY
CAL     SUBROM (Address DA36H)
DEFW    015FH
```

**[Input]** Set the address to be called in IX, then make the call as shown in the previous paragraph.

**[Output]** Depends on the routine called.

**[Registers]** The rear registers and IY are reserved.

# Classroom Network System Materials

Here are the names of the demonstration software and network utility files for the systems (MSX disk BASIC, MSX-DOS, and MSX- CP/M) used in the classroom network.

1) MSX disk BASIC demonstration software

a) Educational demonstration Run "EXER".

Files used:
EXER
EXERCISR
CRE
ANSWER
MENU

b) Graphics demonstration Run "CAL".

Files used:
CAL
VDISP
JAN.CAL
FEBRUARY.CAL
MARCH.CAL
APRIL2.CAL
IMAGE2.CAL
JUN.CAL
JULY.CAL
FALL.CAL
SEPTEM1.CAL
OCTOBER.CAL
CARP.CAL
FIRE.CAL

2) Utilities for the MSX disk BASIC network system

File used: LINCHK.BAS

3) MSX-DOS NETWORK SYSTEM

Files used

MSX DOS.SYS
COMMAND.COM
AUTOEXEC.BAT
NETINIT.COM
NUTL.COM

4) MSX-CP/M NETWORK SYSTEM

Files used

| (Floppy disk version) | (ROM version) |
|---|---|
| BASIC.COM | BASIC.COM |
| NUTL.COM | NUTL.COM |
| NPIP.COM | NPIP.COM |
| MOVCPM.COM | XDIR.COM |
| ASM.COM | SXUB.COM |
| DDT.COM | SUBMIT.COM |
| DUMP.COM | |
| ED.COM | |
| LOAD.COM | |
| XSUB.COM | |
| SUBMIT.COM | |
| PIP.COM | |
| STAT.COM | |
| SYSGEN.COM | |
| FORMAT.COM | |
| SETLIMIT.COM | |

## NUTL (network utility program) for MSX-CP/M

Nutl runs on the MSX-CP/M network system. There are different functions for the teacher and for the students. Inputting the number for a command that can not be used causes an error. During processing of any command, a help screen can be displayed by pressing the INS key and the processing can be returned to the previous level input state by pressing the ESC key. Since the system returns to the net command input state when execution of a command is complete, when finished with Nutl, simply press the return key.
(For details on each command, see the section on the network BIOS for MSX-CP/M.)

1) Student commands

1 = Online Network
2 = Offline Network
6 = Check My - No
9 = Poke
10 = Peek
12 = Send Message
18 = Network Initialize
19 = Network End

2) Teacher commands

3 = Polling Online
4 = Polling Offline
5 = Check line or Communication
6 = Check My - No
7 = Communication Enable
8 = Communication Disable
9 = Poke
10 = Peek
11 = Send Message
13 = Send Key Command
18 = Network Initialize
19 = Network End

## NUTL (Network Utility Program) for MSX-DOS

NUTL runs on the MSX-DOS network system and is used only by the teacher. During processing of each command, a help screen can be displayed by pressing the INS key and the immediately previous input state can be returned to by pressing the ESC key. After each command has been executed, the system returns to the Net Command input state, so to end NUTL, just press the return key.

0 = Network initialization
3 = Polling online
4 = Polling offline
6 = Check My-No
8 = Send hex or binary
9 = Send VRAM
10 = Receive hex or binary
12 = Receive VRAM
13 = Send message command
15 = Sending mail
16 = Receiving mail
17 = Poke command
18 = Peek command
19 = Send key command
23 = Check line or communication
24 = Network end
25 = Communication enable
26 = Communication disable

**Demonstration programs**

1. Files

| | |
|---|---|
| • EXER | Start-up program; executed by the teacher. The teacher sends the Answer program to the students and has them run it, then the teacher executes the Menu program. |
| • ANSWER | Student program; the student inputs numbers to answer the questions displayed on the screen. |
| • MENU | Teacher's management program The teacher uses this program to monitor the progress of the students and to enable or disable telecommunications between students. |
| • CRE | A program that randomly creates problems for the students to answer |
| • EXERCISE | The problem file created by Cre |
| • AUTOEXEC.BAS | Runs Exer. |

2. Start up

(1) Double check that the correct cable is connected. To double check which students are currently online, enter the following from the teacher's keyboard:

```
– CHECK (A,B)
? RIGHT$ ("000000000000000" + BIN$ (A) , 15)
```

For example, if the display is "111110000000000", this means that Students 1-10 are currently connected to the network.
If what the screen displays is not the same as the actual situation, check the cables and the DIP switch settings for the student numbers again.

(2) Starting up the program

After starting up all the students, starting up the teacher's system automatically starts the program.
(When executing (1) Test, press CRTL-STOP to stop the program, then run the test.)

After a short wait, 10 problems are displayed on the student screens and the cursor appears. A short while later, the menu is displayed on the screen.

(3) Student processing

The students input the answers to the problems. Only numbers and minus (−) may be input. The arrow keys, INS, DEL, BS, and the return key may be used. Pressing the ESC key during input cancels that input. Pressing the ESC key with the cursor at the head of the problem ends the answer for that problem and the grade is calculated. When telecommunications between students are enabled, students can Talk to each other by pressing the T key. At this time, the following message is displayed at the bottom of the screen:

To whom (0, 15)?

Input 0 to send a message to the teacher or a number from 1 to 15 to send a message to the student with that number.
Entering the desired message on the next row sends the message to specified recipient (if telecommunications between students are enabled).

(4) Teacher processing

The meaning of the menu items

1. (1 or F1 key) Degree of progress of each student

```
1        ○ x – – – – – – –
.        . . . . . . . . .   → Student 2 is offline
3        ○ ○ x – x ○ – – –   → ○ Right
                               x Wrong
                               – Did not answer yet
↑
Student No.
```

2. (2 or F2 key) The last problem and answer for each student

```
1   2        [      ]  [      ]  [      ]
  .
3   6        [      ]  [      ]  [      ]
↑   ↑          ↑          ↑          ↑
Student  Problem   Problem   Correct   Student
No.      No.                 answer    answer
```

3. (3 or F3 key) Monitors the screen of the specified student.
When a student number (1-15) is input, that student's screen is displayed.

4. (4 or F4 key) Enables telecommunications between students.

5. (5 or F5 key) Disables telecommunications between students.
The default setting is disable mode.

6. (6 or F6 key) Sends a message to the specified student.

# EXER

```
10 _SNDRUN("answer",0)
20 GOSUB 70
30 _BSEN(,0,&H0,&H400,S)
40 _SNDM
50 _POKE(1,MS,0,N)
60 RUN"menu"
70 CLS:SCREEN 0:WIDTH(40):KEY OFF
80 MS=&H7900
90 OPEN"exercise" FOR INPUT AS #1
100 GOSUB 230
110 FOR I=1 TO 10
120 F=I*8-6+MS
130 LINE INPUT #1,A$:B$=LEFT$(A$,20)
140 FOR J=0 TO 7
150 _POKE(ASC(MID$(A$,41+J,1)),F+J)
160 NEXT J
170 LOCATE 5,I*2
180 PRINTB$
190 NEXT I
200 CLOSE #1
210 _POKE(0,MS):_POKE(&HFF,MS+1)
220 RETURN
230 LOCATE 3,1:PRINT"Problem";:LOCATE 18,1:PRINT"Answer"
240 LOCATE 10,23:PRINT"ESC to finish";:LOCATE 28,23:PRINT"score(    )";
250 RETURN
```

## ANSWER

```
10 CLS:SCREEN 0:WIDTH(40):KEY OFF
30 MS=&H7900:MR=&H7B00:_POKE(0,MS):_POKE(&HFF,MS+1):CN=0
40 FOR I=2TO127:_POKE(0,MS+I):NEXT I
50 _PEEK(S,MS)
60 IF S<>1 THEN 50
70 BEEP:BEEP:BEEP
80 _WHO(N):LOCATE1,0:PRINT"Student :";N;
90 GET DATE D$:GET TIME T$
100 LOCATE 20,0:PRINTD$;:LOCATE30,0:PRINTLEFT$(T$,5)
120 I=1:QX=20:QL=8
130 QY=I*2:QA$=AT$(I):GOSUB1000:AT$(I)=QA$
140 GOSUB420
150 IFQK=27 THEN 260
160 IFQK=30 THEN GOSUB630:GOTO380
170 IFQK=31 ORQK=13 THEN GOSUB630:GOTO200
180 IFQK=84 ORQK=116THEN GOSUB630:GOSUB550:GOTO130
190 GOTO130
200 I=I+1:IF I<11 THEN 130
210 LOCATE 10,23:PRINT"Finish (y/n) ? ";
220 YN$=INKEY$:IF YN$=""THEN 220
230 PRINTYN$;
240 IF YN$="y" OR YN$="Y" THEN 260 ELSE LOCATE 10,23:PRINT"ESC to finish   ";:GO
TO120
260 _POKE(2,MS)
270 SC=0:GOSUB 460
300 SC$=STR$(SC):LOCATE 34,23,0:PRINTSC$;
310 B$=INKEY$
320 IF B$="" THEN 310
340 IF B$="e" OR B$="E" THEN 370
350 IF B$="t" OR B$="T" THEN GOSUB 550
360 GOTO 310
370 END
380 I=I-1:IF I=0 THEN I=10
390 GOTO 130
420 GET TIME T$:IF TX$=LEFT$(T$,5) THEN450
430 TX$=LEFT$(T$,5):LOCATE30,0,0:PRINTTX$;
440 LOCATE 20,I*2,1
450 RETURN
460 FOR I=1TO10:S=8*I-6+MS:R=8*I-6+MR:Y=0:R$=""
470 FOR K=0TO7
480 _PEEK(SD,S+K):_PEEK(RD,R+K):R$=R$+CHR$(RD)
490 IF SD<>RD THEN Y=1
500 NEXT K
510 IF Y=0 THEN SC=SC+1:GOTO530
520 LOCATE 28,I*2,0:PRINT"-->";R$
530 NEXT I
540 RETURN
550 LOCATE1,21,0:PRINT"To whom (0..15)       ":LOCATE17,21:INPUTA
560 IF A<0 OR A>15 THEN 550
570 LOCATE 1,22,1:LINEINPUTA$
575 IF A=0 THEN 590
580 _PEEK(B,MR):IFB<>1 THEN 600
590 _TALK(A$,A)
600 LOCATE1,21,0:PRINTSPACE$(38)
610 LOCATE1,22,0:PRINTSPACE$(38)
620 RETURN
630 IFQZ=0THENRETURN
640 A=VAL(QA$):AN$=LEFT$(STR$(A)+SPACE$(8),8)
650 J=I*8-6+MS:AN=VARPTR(AN$):AD=PEEK(AN+2)*256+PEEK(AN+1)
660 FOR K=0TO7
670 _POKE(PEEK(AD+K),J+K)
680 NEXT K
690 CN=CN+1:IF CN=16 THEN CN=0
700 _POKE(CN*16+I,MS+1):RETURN
1000 QC=1:QZ=0:QS=0:GOSUB1330:QT$=LEFT$(QA$+SPACE$(QL),QL)
1010 GOSUB1340
1020 LOCATEQX+QC-1,QY,1
1030 QK$=INKEY$:IF QK$=""THEN1030
1040 QK=ASC(QK$)
1050 IFQK=127THEN1290
1060 IFQK=116ORQK=84THEN1150
1070 IFQK=8THEN1260
1080 IFQK=13THEN1150
1090 IFQK=18THEN1200
1100 IFQK=45THEN1210
1110 IF47<QKANDQK<58THEN1220
```

```
1120 ONQK-26GOTO1140,1160,1180,1150,1150,1220
1130 GOTO1030
1140 IFQC<>1THEN1000
1150 GOSUB1340:QA$=QT$:RETURN
1160 QS=0:GOSUB1330:IFQC=QLORMID$(QT$,QC)=SPACE$(QL-QC+1)THEN1020
1170 QC=QC+1:GOTO1020
1180 QS=0:GOSUB1330:IFQC<>1THENQC=QC-1
1190 GOTO1020
1200 QS=(QS+1)MOD2:GOSUB1330:GOTO1020
1210 IFQC<>1THEN1030
1220 QZ=1:IFQC=QLTHENMID$(QT$,QC,1)=QK$:GOTO1010
1230 IFQS=0THENMID$(QT$,QC,1)=QK$:QC=QC+1:GOTO1010
1240 QW$="":IFQC<>1THENQW$=LEFT$(QT$,QC-1)
1250 QT$=QW$+QK$+MID$(QT$,QC,QL-QC):QC=QC+1:GOTO1010
1260 QZ=1:IFQC=1THENQT$=RIGHT$(QT$,QL-1)+" ":GOTO1010
1270 IFQC=2THENQT$=RIGHT$(QT$,QL-1)+" ":QC=QC-1:GOTO1010
1280 QT$=LEFT$(QT$,QC-2)+RIGHT$(QT$,QL-QC+1)+" ":QC=QC-1:GOTO1010
1290 IFQC=1THENQT$=RIGHT$(QT$,QL-1)+" ":QZ=1:GOTO1010
1300 IFQC=QLTHENQT$=LEFT$(QT$,QL-1)+" ":QZ=1:GOTO1010
1310 IFMID$(QT$,QC)=SPACE$(QL-QC+1)THEN1030
1320 QT$=LEFT$(QT$,QC-1)+RIGHT$(QT$,QL-QC)+" ":QZ=1:GOTO1010
1330 POKE&HFCAA,QS:RETURN
1340 LOCATEQX,QY,0:PRINTQT$:RETURN
```

# MENU

```
10 CLEAR 3000
20 ON ERROR GOTO 1540
30 GOSUB 120
40 GOSUB 240
50 A$=INKEY$
60 IF A$="" THEN 50
70 KY=ASC(A$):IF 48<KY AND KY<56 THEN KY=KY-48:GOTO 100
80 IF 128<KY AND KY<136 THEN KY=KY-128:GOTO 100
90 GOTO 50
100 ON KY GOTO 320,860,1190,1400,1410,1460,1420
110 GOTO 50
120 CLS:SCREEN 0:WIDTH(40):KEY OFF
130 MS=&H7900:MR=&H7B00:G$(0)="  -":G$(1)="  O":G$(2)="  X"
140 FOR I=1TO 10
150 KEY I,CHR$(&H80+I)
160 NEXT I
170 DIM X$(10),PD(15)
180 OPEN"exercise" FOR INPUT AS #1
190 FOR I=1 TO 10
200 LINEINPUT #1,X$(I)
210 NEXT I
220 CLOSE #1
230 RETURN
240 CLS:LOCATE 18,1:PRINT"Menu"
250 LOCATE 4,3:PRINT"1. Status in Progress  (All)"
260 LOCATE 4,5:PRINT"2. Last problem & answer  (All)"
270 LOCATE 4,7:PRINT"3. Display Student's screen  (Each)"
280 LOCATE 4,9:PRINT"4. Enable inter-student Talk (All)"
290 LOCATE 4,11:PRINT"5. Disable inter-student Talk (All)"
300 LOCATE 4,13:PRINT"6. Talk to student (Each/All)"
310 RETURN
320 GOSUB 630
330 _CHECK(A,B)
340 A$=RIGHT$("0000000000000000"+BIN$(A),15):EF=0
350 FOR I=1TO 15
360 IF MID$(A$,16-I,1)="1" THEN GOSUB 690:GOTO 550
370 LOCATE 5,I+4:PRINTRIGHT$(STR$(I),2);
380 _PEEK(M,MS+1,I,N)
390 IF PD(I)=M THEN 540
400 PD(I)=M
410 _RCVM(I)
420 FOR K=1 TO 10:F=K*8-6+MR
430 SP(K)=0
440 _PEEK(M,F)
450 IF M=0 THEN 520
460 SP(K)=1
470 FOR J=0 TO 7
480 _PEEK(AX,F+J)
490 A0=ASC(MID$(X$(K),41+J,1))
500 IF AX<>A0 THEN SP(K)=2:J=7
510 NEXT J
520 NEXT K
530 GOSUB 730
540 'GOSUB 840
550 K$=INKEY$
560 IF K$="" THEN 590
570 IF K$=CHR$(27) THEN EF=1:I=15:GOTO 590
580 IF K$=CHR$(132) THEN GOSUB 770
590 NEXT I
600 GOSUB 1430
610 IF EF=0 THEN 330
620 GOTO 40
630 CLS:GET DATE D$:GET TIME T$
640 LOCATE 20,0:PRINTD$;:LOCATE30,0:PRINTLEFT$(T$,5)
650 LOCATE 20,2:PRINT"Problem no."
660 LOCATE 2,3:PRINT"Student";:LOCATE10,3:PRINT"1  2  3  4  5  6  7  8  9 10"
670 FOR I=1TO15:PD(I)=0:NEXT I
680 RETURN
690 LOCATE 5,I+4:PRINT" .";
700 LOCATE 8,I+4
710 FOR L=1TO10:PRINT"  .";:NEXT L
720 RETURN
730 LOCATE 8,I+4
740 FOR K=1TO10
750 PRINTG$(SP(K));:NEXT K
760 RETURN
770 FORPL=0TO23
780 CN=0:BF$=""
```

```
790 X=VPEEK(PL*40+CN)
800 IF X=0 THEN 830
810 IF X=13 THEN 830
820 BF$=BF$+CHR$(X):CN=CN+1:IF CN<40 THEN 790
830 LPRINT BF$
840 NEXTPL
850 RETURN
860 GOSUB 1110
870 _CHECK(A,B)
880 A$=RIGHT$("0000000000000000"+BIN$(A),15):EF=0
890 FOR I=1TO15
900 IF MID$(A$,16-I,1)="1" THEN GOSUB 1170:GOTO 1030
910 LOCATE 1,I+4:PRINTRIGHT$(STR$(I),2);
920 _PEEK(M,MS+1,I,N)
930 IF PD(I)=M THEN 1030
940 PD(I)=M
950 M=M MOD 16:IF M<1 OR M>10 THEN 1030
960 F=M*8-6+MS
970 LOCATE 4,I+4:PRINTRIGHT$(STR$(M),2);
980 LOCATE 8,I+4:PRINTLEFT$(X$(M),15);MID$(X$(M),41,8);
990 FOR J=0TO7
1000 _PEEK(AX,F+J,I,N)
1010 PRINTCHR$(AX);
1020 NEXT J
1030 K$=INKEY$
1040 IF K$="" THEN 1070
1050 IF K$=CHR$(27) THEN EF=1:I=15:GOTO 1070
1060 IF K$=CHR$(132) THEN GOSUB 770
1070 NEXT I
1080 GOSUB 1430
1090 IF EF=0 THEN 870
1100 GOTO 40
1110 CLS:GET DATE D$:GET TIME T$
1120 LOCATE 20,0:PRINTD$;:LOCATE 30,0:PRINTLEFT$(T$,5)
1130 LOCATE 1,2:PRINT"St      Problem";
1140 LOCATE23,2:PRINT"Right   Answer";
1150 FOR I=1TO15:PD(I)=0:NEXT I
1160 RETURN
1170 LOCATE 1,I+4:PRINT" .";SPACE$(35)
1180 RETURN
1190 CLS
1200 PRINT"Student no. (1..15) ";:INPUT N
1210 IF N<1 OR N>15 THEN PRINT"Reenter.":GOTO 1200
1220 GOSUB 1390
1230 IF MID$(A$,16-N,1)="1" THEN PRINT"Student ";N;"is not connected.":GOTO 1200
1240 _BREC(,N,&H0,&H400,S)
1250 Y$=""
1260 K$=INKEY$
1270 IF K$="" THEN 1260
1280 Y=ASC(K$)
1290 IF Y=27 THEN 40
1300 IF Y=132 THEN GOSUB 770:GOTO 1240
1310 IF Y>48 AND Y<58 THEN 1340
1320 IF Y=13 THEN 1350
1330 GOTO 1240
1340 Y$=Y$+K$:GOTO 1260
1350 M=VAL(Y$):IF M<1 OR M>15 THEN 1240
1360 GOSUB 1390
1370 IF MID$(A$,16-M,1)="1" THEN BEEP:BEEP:BEEP:GOTO 1250
1380 N=M:GOTO 1240
1390 _CHECK(A,B):A$=RIGHT$("0000000000000000"+BIN$(A),15):RETURN
1400 _ENAC(0):_POKE(1,MR,0,N):GOTO40
1410 _DISC(0):_POKE(0,MR,0,N):GOTO40
1420 GOSUB 770:GOTO 40
1430 GET TIME T$:IF TX$=LEFT$(T$,5) THEN 1450
1440 TX$=LEFT$(T$,5):LOCATE30,0,0:PRINTTX$;
1450 RETURN
1460 LOCATE1,21,0:PRINT"To whom (0..15)       ";:LOCATE17,21:INPUTA
1470 IF A<0 OR A>15 THEN 1460
1480 LOCATE1,22,1:LINEINPUTA$
1490 IF A=0 THEN _MESS(A$):GOTO 1510
1500 _MESS(A$,A)
1510 LOCATE1,21,0:PRINTSPACE$(8)
1520 LOCATE1,22,0:PRINTSPACE$(8)
1530 GOTO 40
1540 _PEEK(FL,&H7CA5):_POKE(0,&H7CA5)
1550 EF$=RIGHT$("0000000"+BIN$(FL),7):IF LEFT$(EF$,1)="1" THEN 1570
1560 END
1570 ER=ER+1:RESUME
```

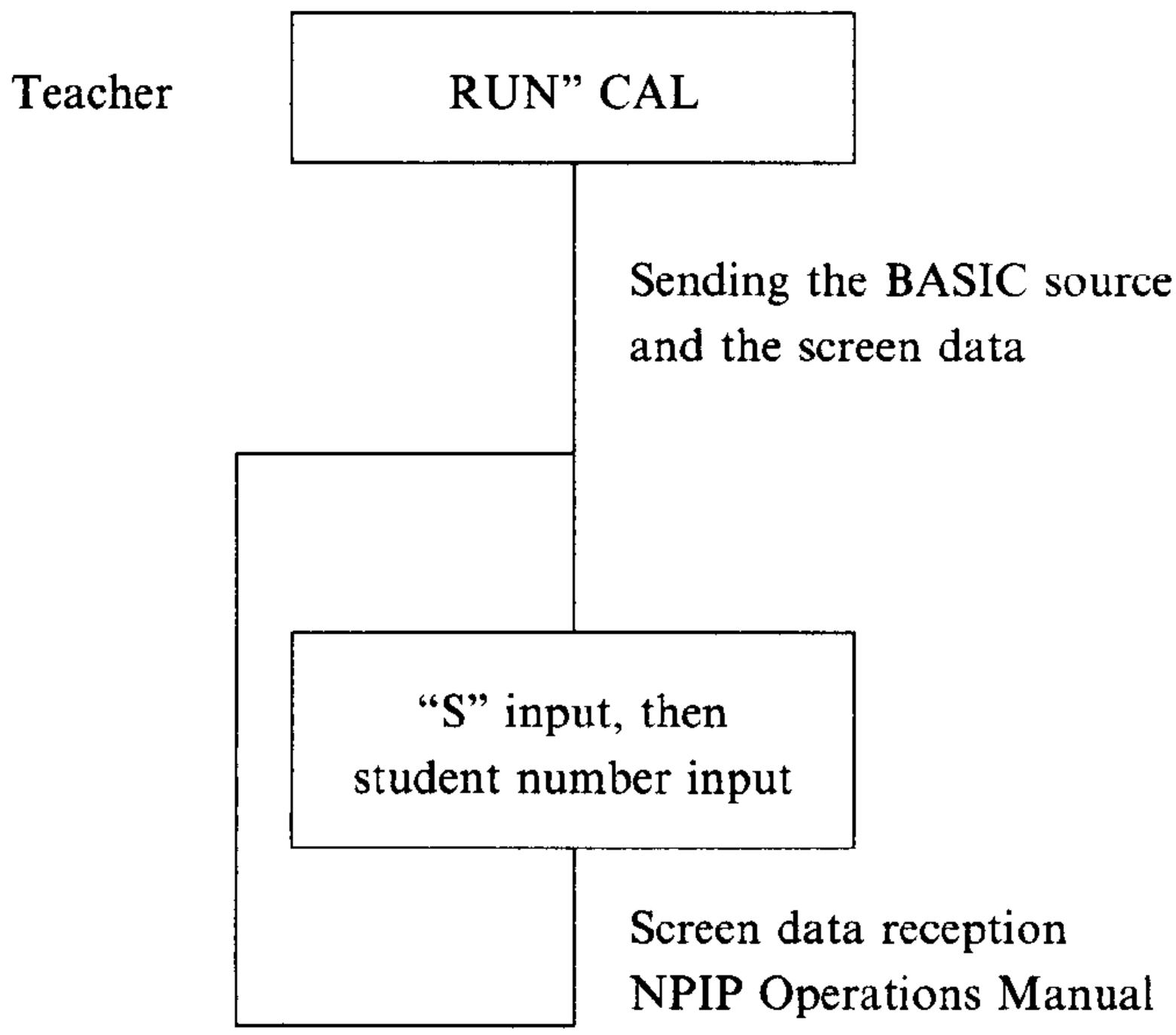
## CRE

```
10 DEF FN RD=INT(RND(-TIME)*18)-9
20 DEF FN R(X,Y)=INT(RND(-TIME)*Y)+X
30 I1=FNR(4,3):I2=FNR(2,3):IX=I1+I2-5
40 ON IX GOTO 60,70,80,90,100
50 GOTO 30
60 I3=2:GOTO 110
70 I3=FNR(1,2):GOTO 110
80 I3=FNR(0,5):GOTO 110
90 I3=FNR(0,2):GOTO 110
100 I3=0
110 I4=10-I1-I2-I3
120 OPEN"exercise" FOR OUTPUT AS #1
130 AX$=SPACE$(8):PT$=" 10 ":NU$=SPACE$(4)
140 FOR I=1TOI1:A=0:P$="":FOR J=1TO2:GOSUB220:NEXT J:GOSUB 250:NEXT I
150 FOR I=1TOI2:A=0:P$="":FOR J=1TO3:GOSUB220:NEXT J:GOSUB 250:NEXT I
160 IF I3=0 THEN 180
170 FOR I=1TOI3:A=0:P$="":FOR J=1TO4:GOSUB220:NEXT J:GOSUB 250:NEXT I
180 IF I4=0 THEN 200
190 FOR I=1TOI4:A=0:P$="":FOR J=1TO5:GOSUB220:NEXT J:GUSUB 250:NEXT I
200 CLOSE #1
210 END
220 PD=FNRD:A=A+PD:PD$=STR$(PD):IF LEFT$(PD$,1)=" "THEN PD$="+"+MID$(PD$,2)
230 P$=P$+PD$
240 RETURN
250 A$=LEFT$(STR$(A)+SPACE$(8),8):IF LEFT$(P$,1)="+" THEN P$=MID$(P$,2)
260 P$=LEFT$(P$+"="+SPACE$(40),40)
270 PRINTP$;A$;AX$;PT$;NU$
280 PRINT#1,P$;A$;AX$;PT$;NU$
290 RETURN
```

**Graphics demonstration program**

1. Entering the Cal call on the teacher's terminal sends the calendar graphics screen (1-12) for the student number to each student.

2. After the graphics screens have been sent to all the students, inputting "S" on the teacher's terminal, then inputting a student number receives that student's graphics screen on the teacher's screen.



## CAL

```
10 DIM M$(12),SN(15)
20 FOR I=1 TO 12
30 READ M$(I)
40 NEXT I
50 _SNDRUN("vdisp",0)
60 _CHECK(A,B)
70 A$=RIGHT$("0000000000000000"+BIN$(A),15)
80 FOR I=1TO 15
90 IF MID$(A$,16-I,1)="1" THEN 260
100 _PEEK(M,&H7900,I,N)
110 IF M<1 OR M>12 THEN 190
120 CLS
130 SCREEN 5
140 SET PAGE 0,0
150 BLOAD M$(M),S
155 COLOR=RESTORE
160 SET PAGE 0,0
170 _BSEN(,I,&H0,&H6A00,S)
180 _POKE(0,&H7900,I,N)
190 IF SN(I)=0 THEN 260
200 _POKE(&HFF,&H7901,I,N)
210 CLS:SCREEN 5:SET PAGE 0,0
220 _BREC(,I,&H0,&H6A00,S)
230 SET PAGE 0,0
240 SN(I)=0
250 _POKE(0,&H7901,I,N)
260 K$=INKEY$
270 IF K$="s" OR K$="S" THEN 280 ELSE 320
280 CLS:SCREEN 0:WIDTH(40)
290 PRINT"Student no. (1..15) ";:INPUT N
300 IF N<1 OR N>15 THEN PRINT"Reenter.":GOTO 290
310 SN(N)=1:CLS:SCREEN 5:SET PAGE 0,0
320 NEXT I
330 GOTO  60
340 DATA "jan.cal","feburary.cal","march.cal","april2.cal","carp.cal"
350 DATA "jun.cal","july.cal","image2.cal","septem1.cal","october.cal"
360 DATA "fall.cal","fire.cal"
```

## VDISP

```
10 _POKE(0,&H7900):_POKE(0,&H7901)
20 DEFINT A-Z
30   DIM X1(62),Y1(62),X2(62),Y2(62),X3(62),Y3(62),X4(62),Y4(62)
40   DIM DM(12),C1(12),C2(12),C3(12),C4(12),C5(12)
50 _WHO(M)
60 IF M<1 THEN M=1:GOTO 80
70 IF M>12 THEN M=M-12:GOTO 70
80 VDP(9)=&H4A
90 T0=0
100 GOSUB 430
110 COLOR 4,4,4
120 GOSUB 1150
130 GET DATE T$
140 A=PEEK(&H2B) : B=(A AND (&H70))
150 IF (B/16)=1 GOTO 480
160 IF (B/16)=2 GOTO 520
170 YY$=MID$(T$,1,2) : YY=VAL(YY$) : TY=YY
180 MM$=MID$(T$,4,2) : MO=VAL(MM$) : TM=MO
190 DD$=MID$(T$,7,2) : DD=VAL(DD$)
200 IF TM<>0 GOTO 220
210 T0=1 : TM=1 : MO=1
220 IF DD<>0 GOTO 240
230 T0=1 : DD=1
240 SCREEN5:SET PAGE 0,0:CLS
250 _POKE(M,&H7900)
260 _PEEK(N,&H7900)
270 IF N<>0 THEN 260
280 SET PAGE 0,0
290 COLOR 4,0,1
300 OPEN "grp:" FOR OUTPUT AS #1
310 ON INTERVAL=60  GOSUB 920 : INTERVAL ON
320 GOSUB 560
330  INTERVAL OFF
340 IF T0<>1 GOTO 360
350 PSET (18,170),C4(MO) : COLOR C4(MO),0,1 : PRINT #1,"Please set date."
360  GOSUB 940
370 _PEEK(L,&H7901)
380 IF L=&HFF THEN 410
390 IF L=LX THEN 360
400 INTERVAL ON:LX=L:GOTO 360
410 IF L=LX THEN 370
420 INTERVAL OFF:LX=L:GOTO 370
430 RESTORE 1200
440 FOR I=1 TO 12
450 READ C1(I),C2(I),C3(I),C4(I),C5(I)
460 NEXT I
470 RETURN
480 YY$=MID$(T$,7,2) : YY=VAL(YY$) : TY=YY
490 MM$=MID$(T$,1,2) : MO=VAL(MM$) : TM=MO
500 DD$=MID$(T$,4,2) : DD=VAL(DD$)
510 GOTO 200
520 YY$=MID$(T$,7,2) : YY=VAL(YY$) : TY=YY
530 MM$=MID$(T$,4,2) : MO=VAL(MM$) : TM=MO
540 DD$=MID$(T$,1,2) : DD=VAL(DD$)
550 GOTO 200
560 IF YY=0 THEN MY=2000 ELSE MY=1900+YY
570 IF MY=INT((MY)/4)*4 THEN RESTORE 1330 ELSE RESTORE 1320
580 FOR I=0 TO 12
590 READ DM(I)
600 NEXT I
610 Z=MY-1901
620 YT!=365*Z+INT(Z/4)
630 MT=DM(MO-1)
640 IF YT!>=32389 THEN YT!=YT!-32389
650 M=(YT!+MT+2) MOD 7
660 N=1
670 FOR Y=110 TO 160 STEP 9
680 FOR X= 150+M*15  TO 250 STEP 15
690   A$=MID$(STR$(N),2)
700 GOSUB 760
710 M=0
720 N=N+1
730 IF N=DM(MO)-DM(MO-1)+1 GOTO 750
740 NEXT X,Y
750 RETURN
760 INTERVAL OFF
```

```
770 COLOR C4(MO)
780 DRAW "BM=x;,=y;"
790 FOR I = 1 TO LEN(A$)
800  IF N >= 10 GOTO 820
810   DRAW "br6"
820 IF YY<>TY GOTO 860
830 IF MO<>TM GOTO 860
840   IF N<>DD GOTO 860
850    COLOR C5(MO),0,1
860    PRINT #1,MID$(A$,I,1);
870    DRAW "BL2"
880    COLOR C4(MO),0,1
890 NEXT I
900 INTERVAL ON
910 RETURN
920 BC=1
930 COLOR BC
940 GET TIME T$
950 HH$=MID$(T$,1,2) : HH=VAL(HH$)
960 MM$=MID$(T$,4,2) : MM=VAL(MM$)
970 SS$=MID$(T$,7,2) : SS=VAL(SS$)
980 IF SS=TS THEN RETURN ELSE TS=SS
990 IF HH>=12 THEN HH=HH-12
1000 I=HH*5+INT(MM/12)+1
1010 IF I=OH THEN  1040
1020 LINE(X3(OH),Y3(OH))-(X4(OH),Y4(OH)),BC
1030 OH=I
1040 LINE(X3(I),Y3(I))-(X4(I),Y4(I)),C1(MO)
1050 I=MM+1
1060 IF I=OM THEN 1090
1070 LINE(X1(OM),Y1(OM))-(X4(OM),Y4(OM)),BC
1080 OM=I
1090 LINE(X1(I),Y1(I))-(X4(I),Y4(I)),C2(MO)
1100 I=SS+1
1110 LINE(X1(OS),Y1(OS))-(X2(OS),Y2(OS)),BC
1120 LINE(X1(I),Y1(I))-(X2(I),Y2(I)),C3(MO)
1130 OS=I
1140 RETURN
1150 RESTORE 1340
1160 FOR I=0 TO 61
1170 READ X1(I),Y1(I),X2(I),Y2(I),X3(I),Y3(I),X4(I),Y4(I)
1180 NEXT I
1190 RETURN
1200 DATA 2,2,6,2,6
1210 DATA 2,2,10,11,2
1220 DATA 2,2,5,3,2
1230 DATA 2,2,5,6,2
1240 DATA 2,2,10,8,2
1250 DATA 2,2,5,4,11
1260 DATA 2,2,13,2,10
1270 DATA 2,2,6,3,12
1280 DATA 2,2,3,4,8
1290 DATA 2,2,6,4,2
1300 DATA 2,2,11,10,2
1310 DATA 12,12,2,10,2
1320 DATA 0,31,59,90,120,151,181,212,243,273,304,334,365
1330 DATA 0,31,60,91,121,152,182,213,244,274,305,335,366
1340 DATA 196,28,201,58,197,34,201,55
1350 DATA 200,28,200,58,200,34,200,55
1360 DATA 204,28,199,58,203,34,199,55
1370 DATA 208,28,197,58,206,35,198,55
1380 DATA 212,29,196,58,209,35,197,55
1390 DATA 216,30,194,57,211,36,197,54
1400 DATA 220,31,193,57,214,36,196,54
1410 DATA 223,32,192,56,216,37,195,54
1420 DATA 226,33,191,56,219,38,195,54
1430 DATA 229,35,190,55,221,40,194,53
1440 DATA 232,37,189,55,222,41,193,53
1450 DATA 234,39,188,54,224,42,193,52
1460 DATA 236,41,187,53,225,44,192,52
1470 DATA 238,43,187,52,226,45,192,51
1480 DATA 239,45,187,52,227,47,192,51
1490 DATA 239,48,186,51,227,48,192,51
1500 DATA 239,50,186,50,228,50,192,50
1510 DATA 239,53,186,49,227,52,192,50
1520 DATA 239,55,187,48,227,53,192,49
1530 DATA 237,57,187,48,226,55,192,49
1540 DATA 236,59,187,47,225,57,193,48
1550 DATA 234,61,188,46,224,58,193,48
```

```
1560 DATA 232,63,189,45,222,59,193,47
1570 DATA 229,65,190,45,220,61,194,47
1580 DATA 226,67,191,44,218,62,195,47
1590 DATA 223,68,192,44,216,63,195,46
1600 DATA 219,70,193,43,214,64,196,46
1610 DATA 216,71,195,43,211,64,197,46
1620 DATA 212,72,196,43,208,65,198,46
1630 DATA 208,72,197,42,206,65,198,45
1640 DATA 204,72,199,42,203,66,199,45
1650 DATA 200,73,200,42,200,66,200,45
1660 DATA 196,72,202,42,197,66,201,45
1670 DATA 191,72,203,42,194,65,202,45
1680 DATA 187,71,204,43,191,65,203,46
1690 DATA 184,71,206,43,189,64,203,46
1700 DATA 180,69,207,43,186,64,204,46
1710 DATA 177,68,208,44,184,63,205,46
1720 DATA 173,67,209,44,181,62,206,47
1730 DATA 170,65,210,45,179,60,206,47
1740 DATA 168,63,211,46,178,59,207,47
1750 DATA 166,61,212,46,176,58,207,48
1760 DATA 164,59,213,47,175,56,208,48
1770 DATA 162,57,213,48,174,55,208,49
1780 DATA 161,54,214,49,173,53,208,49
1790 DATA 161,52,214,49,173,51,208,50
1800 DATA 161,50,214,50,172,50,208,50
1810 DATA 161,47,214,51,173,48,208,51
1820 DATA 162,45,213,52,173,47,208,51
1830 DATA 163,43,213,53,174,45,208,52
1840 DATA 164,41,213,53,175,43,208,52
1850 DATA 166,38,212,54,176,42,207,52
1860 DATA 169,37,211,55,178,41,207,53
1870 DATA 171,35,210,55,180,39,206,53
1880 DATA 174,33,209,56,182,38,205,54
1890 DATA 177,32,208,57,184,37,205,54
1900 DATA 181,30,207,57,187,36,204,54
1910 DATA 185,29,205,57,189,36,203,54
1920 DATA 189,29,204,58,192,35,202,55
1930 DATA 193,28,203,58,195,35,202,55
1940 DATA 197,28,201,58,198,34,201,55
1950 DATA 201,28,200,58,201,34,200,55
1960 CLS:COLOR1,15:STOP
```