1.0 INTRODUCTION

EMP-20 refers to the programmer hardware and EMP20 refers to the software control program. Most other supporting files within the software also begins with EMP20.

1.1 EMP-20 PHYSICAL DESCRIPTION

EMP-20 is a low cost device programming tool, supporting PLD's, Microcontrollers, and most memory devices. The EMP-20 is housed in an attractive plastic enclosure, 9.5 inches long, 5.75 inches wide and 1.25 inches high. The unit has a parallel interface to the PC host. Operation is controlled by an IBM compatible PC, XT, AT, 386, 486 and above machines running the supplied EMP device programming software. An LED indicator in lower left corner indicates the power is on and the control power supply is operating.

Located in the lower left area is the main device programming station consisting of a 48 pin ZIF. The ZIF will handle wide or narrow pattern dual in-line devices. Adapters are use to accommodate non DIP package types. Also in this area is the device "Family Module" connector. The devices that can be programmed at any given time are determined by which Family Module is installed.

Family module boards reconfigure supply voltage, ground and pin driver capabilities for different device families (i.e. architectures). The EMP-20 can support many device families by simply changing or flipping the family module board. Devices supported for the as shipped EMP-20 are handled by 3 included Family Modules. The devices supported are listed below by Family Module.

Family Module #01, EPROM's:

All 8 bit EPROM's, all manufactures. All 16 bit EPROM's, all manufactures. EEPROM's 2804-28256, all manufactures. Flash EPROM's 28F256-28F020, most manufactures.

Family Module #02, MICRO's:

87 and 87C41, 42, 42AH, 55, 48, 49, 48H, 49H, 51, 51FA, 51FB, 51FC, 51RA, 51RB, 51RC, 751, 752, from all manufactures.

Family Module #03, PLD's:

PALCE and GAL 16V8, 20V8, 22V10 from AMD, Lattice, NSC and others.

Many additional devices are supported with other Family Modules and adapters.

1.2 Supported Devices

EPROM’s; are cell based where each bit has is cell that is either charged or not charged to determine a logic one, (true), or a logic zero, (false). They can be bulk erased with ultraviolet light and reused many times. Many other devices use EPROM cells for the internal programmable memory section.

EEPROM’s; are very similar to the EPROM’s except they are bulk erased electrically. This is done either by a command sequence or a high voltage (i.e. 12V). Most are programmed without high voltage.

FLASH MEMORIES; these devices are also electrically erasable but may be erased and/or programmed block by block. They require special algorithms to access internal registers in order to do any operations on the device.

MICROCONTROLLERS; these are CPU type devices capable of executing strings of instructions. They have some type of internal programmable memory made up from any of the types above.

PLD’s; these are made up of logic arrays or groups of logic arrays that are configured by programming interconnect information to the internal memory. This memory may be made up of any of the above memory technology. Whole logic functions taking the place of many individual components may be programmed into a single PLD. There is also a class of PLD’s, not supported by the EMP-20, that are configured by blowing fuses or connections not required or creating connections, (antifuse) for the desired function. PLD’s are programmed according to a “fuse map” usually contained in a JEDEC file.

NOVRAMS and SRAM’s MODULES; these are generally made up of battery backed up SRAM and require only 5 volts to program. Erasing is not usually require but can be done by an interruption of power to the SRAM. Memory data is retained when external power is removed.

SERIAL EEPROMS; these are very similar to EEPROM’s except they are programmed and read serially instead of parallel.

1.3 SUPPORTED FILE FORMATS

BINARY — Where the data is a binary string with no header or end-of-file markers.

INTEL HEX — File format usually output from an assembler when the target processor is an Intel device. Data is in ASCII format and contains various fields for address, data and checksum.

MOTOROLA S RECORD — File format used with a Motorola processor. Data is in ASCII format and contains multiple “records”, each consisting of a single line with fields for address, data and checksum.

HEX (ASCII) — With this format, hex values which are typed into an editor (which are actually ASCII values) are recognized and transferred into the buffer just as they were saved in the text file.

JEDEC — This format is used mostly with programmable logic devices. It follows the JEDEC standard for translating files that are output from most logic compilers.

See section 6.0 for more detailed information on file formats.

1.4 MINIMUM SYSTEM REQUIREMENTS

The systems which can be used with EMP-20 include the IBM PC, XT, AT, 386, 486 and above machines. You must have at least 640KB of RAM to start the EMP20 software. It is recommended that you copy the floppy disk to hard disk for operation. You can use the floppy disk provided rather than operate off of a hard disk. There is limited room on the floppy disk and you will be limited as to which devices you will be able to program.

It is best to run the EMP20 software by itself in your system. TSR's, print queues or other memory load and stay resident programs can cause indeterminate problems.

2.0 MAIN MENU COMMANDS

To invoke a command or setting option, type the alphanumeric character displayed to its left or use arrow keys to scroll the cursor bar (if using arrow keys, press <ENTER> after highlighting the desired selection).

2.1 ERASE DEVICE (0)

This command only appears when the part can be electrically erased via software. It will erase the entire device. First, ensure device is correctly inserted into ZIF socket and latch is down. When ready, press '0' to erase the device.

See section 2.11.2 for information about user selectable block erase capability. This feature is available for devices with user-selectable block locking.

2.2 PROGRAM WITH SELECTED ALGORITHM (1)

This command programs a device with the contents of the data buffer. First, ensure device is correctly inserted into ZIF socket and latch is down. Check Buffer Range-D (K), Device Range (L), Split (O) and Set (P) settings before proceeding— 'K' and 'L' values default to the device's size, 'O' and 'P' default to "1".

When a device is selected, its default algorithm is called. The algorithm name, typically “default”, is displayed in the “status/settings” box’s ‘J’ field (i.e. “Algorithm” display; see section 3.2 for information about selecting other algorithms). When ready, press ‘1’ to program the device. Verification automatically follows programming.

Note: The ‘J’ selection allows optional algorithms if available.

2.3 VERIFY DEVICE IS ERASED (2)

This command determines if device is in the erased state. First, ensure device is correctly inserted into ZIF socket and latch is down; otherwise the device may appear to be erased, even though it isn’t. Additionally, check Device Range (L). When ready, press ‘2’ to run the erase-verify operation.

If all bytes are erased, the message “device successfully erase verified” appears on screen. Pressing any key returns the main menu.

If an unerased cell is encountered, an on-screen indication appears containing the first erase error. Pressing any key, returns the main menu (and terminates the erase verify operation). Alternatively, the ‘S’ or ‘F’ keys allow the erase verify to continue, outputting all erase errors to the screen or to a file.

2.3.1 DISPLAYING MULTIPLE ERASE ERRORS TO THE SCREEN

The ‘S’ key outputs errors to the screen. The first error displayed is the one that was shown when the first unerased cell was encountered during the Verify Device is Erased (3) command. Additional errors get listed below. After it sends a page to the screen, EMP20 waits for a key to be pressed before displaying more errors. Pressing ‘/’ disables pagination <CONTROL-S> then governs pause/continue. <ESC> at any time aborts the error display and returns the main menu.

2.3.2 WRITING MULTIPLE ERASE ERRORS TO A FILE

The ‘F’ key outputs erase errors to a file. When ‘F’ is pressed, a box appears prompting for a filename (make sure this file does not exist on disk). The message “cannot create file” appears if this file already exists; the verify erase function will only write into new files.

Once an appropriate filename has been entered, erase errors are displayed to the screen in the manner described in 2.3.1 above. The text, however, is not paginated— <CONTROL-S> starts/stops text. <ESC> at any time aborts the erase verify and closes the erase error file. All erase error information prior to this point will be in the newly created file.

2.4 VERIFY DEVICE EQUALS BUFFER (3)

This command compares device contents to the buffer through the specified Ranges (L and K respectively). When ready, press ‘3’ to run the compare operation.

If all bytes compare, the message “device successfully verified” appears on screen. Pressing any key returns the main menu.

If a miscompare occurs, an on-screen indication appears containing the first error. Pressing any key, at this point, returns the main menu (and terminates the compare operation). Alternatively, the ‘S’ or ‘F’ keys allow the verify to continue, outputting all errors to the screen or a file.

2.4.1 DISPLAYING MULTIPLE PROGRAMMING ERRORS TO SCREEN

The ‘S’ key outputs errors to the screen. The first error displayed is the one that was shown from the miscompare during the Verify Device Equals Buffer (3) command. Additional errors get listed below. After it sends a page to the screen, EMP20 waits for a key to be pressed before displaying more errors. Pressing ‘/’ disables pagination— <CONTROL-S> then governs pause/continue. <ESC> at any time aborts the error display and returns the main menu.

2.4.2 WRITING MULTIPLE PROGRAMMING ERRORS TO FILE

The ‘F’ key outputs errors to a file. When ‘F’ is pressed, a box appears prompting for a filename (make sure this file does not exist on disk). The message “cannot create file” appears if this file already exists; the verify function will only write into new files.

Once an appropriate filename has been entered, errors are displayed to the screen in the manner described in 2.4.2 above. The text, however, is not paginated— <CONTROL-S> starts/stops text. <ESC> at any time aborts the verify and closes the error file. All error information prior to this point will be in the newly created file.

2.5 READ DEVICE INTO BUFFER (4)

This command reads device contents into the data buffer. First, ensure device is correctly inserted into ZIF socket and latch is down. Check Buffer Range-D (K), Device Range (L), Split(O) and Set (P) settings before starting— ‘K’ and ‘L’ values default to the device’s size, ‘O’ and ‘P’ default to “1”.

Caution: Reading a device into the data buffer destroys buffer contents through the specified range. Make sure anything needed has been saved. When ready, press ‘4’ to load device.

2.6 SELECT DEVICE (5)

EMP20's device selection menu lets you choose a manufacturer and one of their devices that will be used for all main-menu operations and status/settings displays. All supported manufactures and devices are listed in these menus. Family module and socket module (if required) information is shown next to the device part numbers.

---- Selecting a Manufacturer ----

All of the supported manufacturers are displayed in the left box. To select a manufacturer, enter the number displayed to the left of the manufacturer or use the arrow keys to scroll the cursor bar to the appropriate manufacturer. The manufacturers list may exceed the capacity of the screen. In this case, continue to use the arrow keys to scroll to the right past the screen. The screen will automatically adjust so that the unseen manufactures will display.

As the cursor bar scrolls down each manufacturer, a new list appears in the right-hand box. These are the devices that are supported for this manufacturer. Once the manufacturer has been selected, press <CR>. If you need to re-select the manufacturer name, press <TAB> to bring the cursor back to the Manufacturer menu.

---- Selecting a Device ----

Once the manufacturer has been selected, the cursor bar moves to the right-hand box, and all devices supported for the selected manufacturer are displayed.

To select a device, the same procedure is used as above. Enter the number displaying to the left of the device (not the device number itself), or use the arrow keys to scroll the cursor bar to the appropriate part. Some lists exceed the capacity of the screen. In this case, continue to use the arrow keys to scroll past the screen. The screen will automatically adjust so that the unseen devices will display.

Once the device has been selected, press <CR>. This will select the device and return EMP to the main menu or overlay a socket/family module display. If a socket/family module overlay appears on screen, note the information specified, then hit any key to return the main menu. Make sure that the correct device is displayed under the 'Manufacturer: ' and 'Device: ' display, just above option 'J'. in the upper right-hand corner of the box labeled 'Status/Settings'. If the device has not been selected properly, press '5' again to reselect the device.

2.6.1 Socket/Family Module Display

A socket/family module overlay may appear after device selection if a socket module is required (for non-DIP selections). This overlay provides socket module placement and part number information for that device. Any key clears this display and returns the main menu.

2.7 PRINT BUFFER TO ASCII FILE (6)

This command writes buffer contents in ASCII form to a disk file or printer. When asked for a filename, enter “PRN” for printer or the output filename. The ASCII data is represented in exactly the same way it is in the editor (i.e. hex mode). Before writing from buffer check Buffer Range-F (S) settings. Press ‘6’ to write to selected output.

2.8 BUFFER EDITOR (7)

This command invokes the buffer editor display. Data read in from a file or device is placed in a data buffer. The buffer editor allows you to view and/or edit this data. When loading a file, note File Range (T) to make sure the buffer gets loaded through the range of the device. Also note Buffer Range-F (S) to see where the device data will be loaded. Press ‘7’ to enter buffer editor display. Once in the editor, <ESC> returns the main menu.

See section 4.0 for more detail on the buffer editor.

2.9 LOAD FILE FROM DISK (8)

This command copies a file to the data buffer. Before loading the file, check Buffer Range-F (S) and File Range (T). Make sure File Type (U) is compatible with the file being loaded, and Filename (V) has been entered correctly.

See section 3.0 for more detail on files.

Buffer contents, through the specified range, are overwritten by the new file. If needed, make sure anything in the buffer has been saved. Press ‘8’ to load buffer from file.

2.10 SAVE FILE TO DISK (9)

This command saves buffer contents to a file. Before saving to file, check Buffer Range-F (S) and File Range (T). Make sure File Type (U) is set to the format you’re wanting to save the file as, and Filename (V) has been entered correctly.

The contents of the buffer through the specified File Range (T) will be written into the Filename (V), completely erasing any existing file with the same name. Before saving to disk, make sure that no file with the same name is needed. Press ‘9’ to save buffer contents to a file.

2.11 ADDITIONAL COMMANDS

Beyond the standard list of main-menu commands described above, some devices have features that require additional commands. The appropriate additional command(s) are displayed when such a device is selected. See below for currently-supported additional commands. Note that the letter 'A' is not used.

2.11.1 EDIT ENCRYPTION ARRAY (A) (Microcontrollers)

An encryption array prevents user access to the device's true EPROM contents without the correct hex codes. Encryption arrays can be 16 byte, 32 byte or 64 byte data combinations that are programmed into a special location on the device. In order to read out the correct data, the same combination must be entered in this selection. When an encryption array has been programmed, the data output will be the code data XNOR'ed with the encryption data.

To enter encryption data, hit "A" and enter the desired hex values into the array. Hit return if you want to use the displayed values, or hit "L" to load an encryption array you previously stored on disk. To save the current displayed encryption array to disk, hit "S". You have to name the encryption file you want to save to or read from disk. By hitting TAB you can make the encryption array values active (encryption mode on) or inactive (encryption array off).

2.11.2 EDIT/USE TEST VECTORS (A) (PLD's)

To edit or use test vectors for GAL/PAL devices, hit "A" when a GAL/PAL has been selected. Test Vectors are used to do functional tests on programmed GAL/PAL devices. To access the Test Vector menu, four things must have been done:

1. A GAL/PAL device must be selected.
2. A JEDEC file must be already loaded.
3. The correct device must be properly installed in the EMP-20.
4. The proper verify voltages must be selected from the main menu,(option "R").

Once all of these thing have been accomplished, press "A" to access the Test Vector menu. (JEDEC files cannot be created by EMP, they are created by PAL design software such as PALASM, ABEL, CUPL, etc..)

- Vector Menu Layout -

The total number of Test Vectors in the JEDEC file being used is shown in the upper right hand corner. The current vector line number and device pin number are shown just below it and the Verify High and Verify Low voltages that determine what voltages will be used when testing the entire set of vectors is shown below that. The red arrow (<) shown next to one of the voltages shows which of the two voltages will be used when testing one vector at a time. To toggle between high and low press “F4”.

To test only one vector at a time, make sure the correct VCC voltage is selected then press “F2”. The selected VCC voltage will be turned on , the vector will be run, then VCC will be turned off. To test all vectors in one sequence, press “F3”. EMP will run all Test Vectors twice, first using the Verify High Voltage and again using Verify Low Voltage.

- Editing Test Vectors -

To change Test Vectors simply use the arrow keys to move the cursor over the desired vector, then press any valid vector key:

0.	Drive pin low
1.	Drive pin high
C.	Clock pin (low, high, then low
H.	Test for output high
K.	Clock pin (high, low, then high)
L	Test for output low
N.	Power pin or untested output pin
X.	Do not test or drive pin: Use “default test condition”
Z.	Test for high impedance

After editing vectors the new file may be saved in JEDEC format using option “9” from the main menu.

2.11.3 SETTING SECURITY BITS (B) When Applicable)

Security bits on some Microcontrollers and most PLD’s are used to prevent unauthorized access to the contents of the EPROM array on the device. Once the security bit(s) have been set, there is no way to program or read the device unless the entire device is erased. Devices that support encryption tables and/or security bits fall into 4 categories:

1. Single bit systems
2. Two bit systems with 16 byte encryption array
3. Two bit systems with 32 byte encryption array
4. Three bit systems with 64 byte encryption array

In a single bit system there is only one bit which is either inactive (unprogrammed), or active (programmed). Once set it is not possible to read the device contents.

In a two bit system, the bit combinations are as follows:

U=unprogrammed P=programmed

1.UU Both bits unlocked. When both bits are unlocked, there is no lock on the device. However, any encryption array programmed into the device will be valid and if used will encrypt the data on the device.

2.PU Security bit 1=ON, bit 2=OFF. This disables the device from further programming and MOVC instructions executed from external memory are disabled from fetching code bytes from internal memory.

3.PP Security bit 1=ON, bit 2=ON. All the above features are active and the device can no longer be verified.

In a three bit system, the bit combinations are as follows:

1.UUU All bits unlocked. When all the bits are unlocked, there is no lock on the device. However, any encryption array programmed into the device will be valid and if used will encrypt the data on the device.

2.PUU Security bit 1=ON, bit 2=OFF, bit 3=OFF. This combination disables the device from further programming and MOVC instructions executed from external memory are disabled from fetching code bytes from internal memory.

3.PPU Security bit 1=ON, bit 2=ON, bit 3=OFF. Same as above and the device can no longer be verified.

4.PPP Security bit 1=ON, bit 2=ON, bit 3=ON. All the above features are active and external execution is disabled

2.11.4 BYTE SWAP (B)

This command is only available for 16-bit wide devices. It does not act upon the device itself, but rather modifies the data in the buffer. Default mode for 16-bit devices sets low-byte portion of word-wide data to even addresses and high-byte data to odd addresses. Byte swap switches low bytes to odd addresses and high bytes to even addresses.

2.11.5 BLOCK LOCKING/ERASE ©

This command is only available for devices with user-selectable block locking capability. When selected, an overlay is displayed that represents the individually-erasable blocks of the device’s memory array with a flag for each that can be set to invoke locking of that block. Any number and/or sequence of blocks can be locked. This function is usually selected after programming.

Additionally, each block has a flag that can be set for erasing that block. Any number and/or sequence of blocks can be erased. This reduces erase time by only erasing those blocks requiring modification; i.e. other blocks are not affected. Additional time is saved because blocks not requiring modification do not need to be reprogrammed with the old contents.

Note: Buffer Range-D (K) and Device Range (L) settings must be appropriately adjusted to correctly direct data to the desired blocks when reprogramming a device that has been erased in the manner described in this section, i.e. individual block(s) vs. full device.

2.11.6 PROGRAM ENCRYPTION ARRAY © (Microcontrollers)

By selecting “C” and confirming by hitting “Y” when requested you will program the array into the device. Thereafter, you will have to use the same displayed encryption array anytime you read back this device

3.0 STATUS/SETTINGS DISPLAY

3.1 MANUFACTURER DISPLAY

Upper right hand corner. Selecting “5” first allows the selection of or change to a new manufacturer. The selected manufacturer is shown here. When EMP20 first comes up, “Use option 5” is displayed since a manufacturer and a device have not been selected (unless the special initial macro in the IFLASH3.INI file defines a device).

3.2 DEVICE DISPLAY

After selecting a manufacturer, as in 3.1 above, the device menu for the selected manufacturer is shown. From this menu select or change to a new device. The selected device is shown here just under the manufacturer display.

3.3 ALGORITHM (J)

The algorithm employed during programming is indicated just under the device display (upper right-hand corner). When a device is first selected, EMP20 defaults to the fastest algorithm recommended by the manufacturer based on the latest programming data available to Needham’s Electronics. To select a different algorithm, press ‘J’. A menu displaying all algorithms available for the selected device appears. Selecting an alternative algorithm may adversely affect the long term data retention for the device.

In some cases there will be more than one algorithm for a particular device; the additional algorithm(s) may be available to support special programming considerations or customized needs. In other cases, a single algorithm will appear; this means the default algorithm is the only one available for the selected device.

Note: When EMP20 first comes up, the algorithm is displayed as ‘None Selected’; this is because a device has not yet been selected.

3.4 BUFFER RANGE (K)

When programming a device, data comes from buffer locations as determined by Buffer Range-D (K) settings. Similarly, when reading a device into the buffer, data is placed at locations established by ‘K’ settings.

Press ‘K’ to set Buffer Range-D— a box appears over the Buffer Range-D start address. Type in a new number if desired. <ENTER> makes EMP20 accept your input, <ESC> retains the original value and proceeds to the Buffer Range-D end-address box (apply same procedure used for start-address box).

For example, you want EMP20 to read 200h bytes of data starting at device location 000h and place the data into the buffer starting at address 100h and ending at address 300h. Set ‘K’ to 00000100, 00000300 and Device Range (L) to 00000000, 00000200. Select Load Device (4) to complete the operation.

Buffer Range-D (K) and Buffer Range-F (S) both refer to the data buffer. ‘K’ is used for device-related operations and ‘S’ for file-related operations.

Note: ‘K’ is linked with Device Range (L), Splits (O) and Sets (P). When entering ‘K’, ‘L’ or even ‘K’ itself may automatically adjust to new values depending on ‘O’, ‘P’ and previous buffer/device values.

3.5 DEVICE RANGE (L)

Device Range (L) is the range of device addresses to be acted on during read, program or verify operations.

Press ‘L’ to select Device Range— a box appears over the Device Range start address. Type in a new number if desired. <ENTER> makes EMP20 accept your input, <ESC> retains the original value and proceeds to the Device Range end-address box (apply same procedure used for start-address box).

Note: ‘L’ is linked with Buffer Range-D (K), Splits (O) and Sets (P). When entering ‘L’, ‘K’ or even ‘L’ itself may automatically adjust to new values depending on ‘O’, ‘P’ and previous buffer/device values.

3.6 BUFFER CHECKSUM (M)

Press ‘M’ to have EMP20 perform a checksum of buffer contents. Make sure Buffer Range-D (K) is correct before starting this operation, as the checksum is done for that range. The 2’s complement of the checksum is shown directly after the buffer checksum, denoted by ‘cmpl’.

Note: see section 5.8 for details on checksums for 16 bit devices.

EMP20 includes “FF”s in its buffer checksum calculation. Some checksum methods disregard “FF”s. Note that checksum calculation methods differ at the various programmer manufacturers, therefore the value generated by EMP20 may not coincide with one done on the same data by other means.

3.7 DEVICE CHECKSUM (N)

Press ‘N’ to have EMP20 perform a checksum of the device. Make sure Device Range (L) is correct before starting this operation, as the checksum is done for that range. Also make sure the correct device was selected. The 2’s complement of the checksum is shown directly after the device checksum, denoted by ‘compl’.

Note: see section 5.8 for details on checksums for 16 bit devices.

EMP20 includes “FF”s in its device checksum calculation. Some checksum methods disregard “FF”s. Note that checksum calculation methods differ at the various programmer manufacturers, therefore the value generated by EMP20 may not coincide with one done on the same data by other means.

3.8 SPLITTING DEVICES (O)

Splitting is used to program 8-bit devices for applications which are more than 8-bits wide. For example, to program an 8-bit device for a 16-bit wide system, a split of 2 is selected. For a 32-bit system, a split of 4 is used.

Press ‘O’ to change a split value— two user-input boxes appear in order. The first box prompts you for the split factor (i.e. how many 8-bit devices you’re using); the second, for the current split device. If a split factor of 4 is selected, there are 4 devices in that group; you must increment the split number for each device as you program it.

The following graphics describe the flow of data from the buffer to the devices: <u>EXAMPLE FOR SPLIT OF 2</u>

```
DEVICE 1

device location
-------------------------------------------------------------        50      00
|               ---------------------------------------------        55      01
|               |               -----------------------------        5A      02
|               |               |               -------------        44      03
|               |               |               |
00      01      02      03      04      05      06      07      BUFFER LOCATION
50      AA      55      A5      5A      20      44      41      BUFFER DATA
DEVICE 2

device location
        -----------------------------------------------------        AA      00
        |               -------------------------------------        A5      01
        |               |               ------------------------ 20      02
        |               |               |               -------- 41      03
        |               |               |               |
00      01      02      03      04      05      06      07      BUFFER LOCATION
50      AA      55      A5      5A      20      44      41      BUFFER DATA
```

After programming, the device contents will look like this:

DEVICE 1	DEVICE 2	MEMORY MAP
8 bits	8 bits	16 bits
50	AA	50AA
55	A5	55A5
5A	20	5A20
44	41	4441

Similarly, EMP20 can be used to split 16 bit devices for a 32 bit, 64 bit or larger system.

3.9 SETS (P)

Sets (P) allows you to put buffer data into multiple devices. For example: If buffer contents is 8MB in length and 28F016SV (2MB) devices are being programmed, 4 28F016SVs are needed. Press ‘P’ to change Set values.

Two user-input boxes appear in order the first box prompts you for the number of sets; the second, for the current set number. When a set is selected, EMP20 adjusts programming by taking the device length (e.g. 2MB) and adding it to the buffer start for each successive device. For example, if 4 sets were selected and set #3 is currently selected, EMP20 points to the area for which set #3 would be in memory. For this example, it would be buffer starting address plus 4MB [i.e. buffer start + 2MB (added for device1) + 2MB (added for device2)].

Graphically, data is drawn from the buffer like this (assuming device length is 2MB):

buffer start + 0 bytes	= start for device 1
buffer start + 2MB	= start for device 2
buffer start + 4MB	= start for device 3
buffer start + 6MB	= start for device 4

Note: When entering the set factor, EMP20 automatically readjusts the current buffer size. For instance, moving from a set factor of 1 to 2 doubles the size of the buffer, and moving from 1 to 4 quadruples it. Make sure there is enough buffer space by setting the DMEM option in EMP20.INI.

3.10 PROGRAMMING VOLTAGES (Q)

When selecting a device, EMP20 enables the proper default programming voltages. You can manually change V_{PP}/V_{CC} by pressing 'Q'. EMP20 then overlays a table with lines having V_{PP}/V_{CC} options; select one and press <ENTER> to enable the voltages on that line.

Note: Voltages are changed for programming and verify cycles.

Note: When reselecting a part, voltages are set to their default values.

Warning: Setting improper voltages can destroy a device. If you are unsure of the correct voltage settings, reselect the device (5).

3.11 VERIFY HIGH/LOW ®

To provide external read-back margining for the programmed data, V_{CC} can be adjusted to higher or lower voltages than was employed during programming. This capability is allowed for the selected device if 'R' is fully illuminated. If not available for the selected device, 'R' is dimly illuminated.

3.12 BUFFER RANGE (S)

When saving buffer contents to a file, data comes from buffer locations as determined by Buffer Range-F (S) settings. Similarly, when reading a file into the data buffer, data is placed at locations established by 'S' settings.

Press 'S' to set Buffer Range-F— a box appears over the Buffer Range-F start address. Type in a new number if desired. <ENTER> makes EMP20 accept your input, <ESC> retains the original value and proceeds to the Buffer Range-F end-address box (apply same procedure used for start-address box). Check 'V' for correct Filename.

For example, you want EMP20 to read 200h bytes of data starting at file location 000h and place the data into the buffer starting at address 100h and ending at address 300h. Set 'S' to 00000100, 00000300 and File Range (T) to 00000000, 00000200. Select Load File to Buffer (8) to complete the operation.

Buffer Range-D (K) and Buffer Range-F (S) both refer to the data buffer. 'K' is used for device-related operations and 'S' for file-related operations.

Note: 'S' is linked with File Range (T), Splits (O) and Sets (P). When entering 'S', 'T' or even 'S' itself may automatically adjust to new values depending on 'O', 'P' and previous buffer/device values.

3.13 FILE RANGE (T)

File Range (T) is the range of file addresses to be acted on during a read or write file operation.

Press ‘T’ to select File Range— a box appears over the File Range starting address. Type in a new number if desired. <ENTER> makes EMP20 accept your input, <ESC> retains the original number and proceeds to the File Range end-address box (apply same procedure used for start-address box).

File Range (T) may start and end at virtually any address, however an attempt is made to keep ‘T’ within the limitations of ‘S’. Additionally, you must match the ‘T’ value to the offset in an Intel Hex or Motorola S Record file, if it is not zero.

Note: ‘T’ is linked with the Buffer Range-F (S), Splits (O) and Sets (P). When entering ‘T’, ‘S’ or ‘T’ itself may automatically adjust to new values depending on ‘O’, ‘P’ and previous buffer/device values.

3.14 FILE TYPE (U)

Select one of the file types from Binary, Intel Hex, Motorola S Record, Hex (ASCII)or JEDEC. Section 6.0 details the different file types.

3.15 FILENAME (V)

Filename (V) identifies the file that is loaded into or saved from the data buffer. To set or change Filename, press ‘V’. A box comes up with the current filename. If the box is blank, simply type a filename and <ENTER>. If a filename already exists in this box, <ENTER> alone retains the same filename. Otherwise, enter a new filename over the old filename. <CONTROL-END> deletes to the end of the box— this clears the line so the new filename may be entered. When entering the filename, most editing keys work (e.g. arrow keys, DEL, INS, END and HOME).

Note: When writing to a file, if one already exists with the same name, that file will be overwritten by the new file. Whenever this occurs, EMP20 asks if it is OK to do so. Before continuing, make sure the file to be overwritten is not needed.

3.15.1 SEARCHING FOR A FILE

EMP20 allows wildcards or DIR names. Entering “*.DAT” displays all .DAT files in the current directory. As an example, entering “C:\EMP20\*.DAT” displays all .DAT files in the directory C:\EMP20, entering “C:\EMP20” alone displays all files in C:\EMP20.

<F1> at the Filename prompt brings up a list of all files (i.e. **.**) in the current directory. From there, other directories may be searched. Once at the file listing, valid sub-directories are displayed to the left and files to the right.

3.15.2 CHANGING DIRECTORIES

Use the arrow keys to highlight a new directory (or select with mouse). <ENTER> then displays all files in that directory. Note that the directory names displayed are sub-directories of the directory specified. For instance, if the directory was specified as "C:\EMP20", and the directory name "DATA" is showing, <ENTER> on "DATA" displays files for C:\EMP20\DATA, not already at the root directory <ENTER> when ".." is highlighted moves to the parent directory.

The directory and currently-selected (highlighted) filename is displayed in the upper-right menu. Thus, although the sub-directories of the currently used directory are displayed, the full path may be seen in the upper-right menu.

3.15.3 SELECTING A FILENAME

To select a Filename, press the right-arrow key to move to the file selection menu. Use PAGE-UP, PAGE-DOWN, or the arrow keys to move about and select the file. To move up or down with the mouse, click on the words PGUP and PGDN located at the bottom of each menu. Once a filename has been chosen, (i.e. highlighted), press <ENTER> or click with the mouse. This will become the current filename, displayed under selection 'V' in the status/settings box.

Remember that the only files displayed are those that match the wildcard values. For instance, if "*.DAT" is entered, only .DAT files display. Some directories may appear to be blank for this reason. These directories may contain files, just not files matching the file specification (i.e. *.DAT).

3.16 PORT BASE (W)

The port address is determined by EMP20. If EMP-20 is not connected to a valid parallel port, EMP20 will request that you enter the port address (just in case a non-standard address is used). If EMP-20 is still not found or no address is entered, EMP20 enters the Demo Mode. A message to this effect is displayed in the lower right of the command/settings screen. If a successful connection is made, the port address is displayed at 'W'.

4.0 USING THE BUFFER EDITOR

Buffer data is that data which gets programmed into a device. It is read in from a master device or file. The editor allows you to view or alter that data. You may move around the buffer using special quick move scroll keys, page up/down/home keys, arrow keys, the direct GOTO address key or the mouse.

4.1 DISPLAY CHARACTERISTICS

The editor has two data display modes, Hex and Binary (Hex is the default mode). <F4> toggles between these modes.

4.1.1 HEX / BINARY DATA DISPLAY MODES

In Hex data display mode there are two data fields:

1. a 32 bit nibble (16 byte) x 16 line hex field
2. a 16 character x 16 line ASCII field

<F2> positions a blinking data entry cursor in the Hex field and <F3> positions it in the ASCII field. Either action enables buffer data entry or modification.

In the Binary data display mode there are three data fields:

1. a 32 bit x 16 line Binary field
2. an 8 nibble (4 byte) x 16 line Hex field
3. a 4 character x 16 line ASCII field

<F2> positions a blinking data entry cursor in the Binary field and <F3> positions it in the Hex field. Either action enables buffer data entry or modification.

4.1.2 HEX / BINARY ADDRESS DISPLAY MODES

In Binary data display mode, there two possible address display modes Hex and Decimal (Hex is the default mode). <F6> toggles between modes.

Note: In Decimal address display mode, buffer editing is limited to 64KB. For example, if the buffer is sized to 3FFFFh, (i.e. DMEM=2 0r 256KB), editing would be restricted to 00000H to 0FFFFh. Once back in Hex address display mode full editing capability is restored.

4.2 ENTERING DATA VIA THE EDITOR

To enter data into the editor, you must first hit an appropriate key to allow changes (<F1>, <F2>, <F3>, or <F8>), invoke the desired mode. Hex mode is default. While in Hex mode:

F2	= Enter Hex data at cursor address
F3	= Enter ASCII data at cursor address
F4	= Toggle between Hex and Binary data display modes

While in Binary mode:

F2	= Enter Binary data at cursor address
F3	= Enter Hex data at cursor address
F4	= Toggle between Hex and Binary data display modes
F6	= Toggle between Hex and Decimal address modes

In Binary mode all manipulations done to the buffer are done in binary. The display is altered, changing to a field of binary ones and zeroes, followed by the hex and ASCII equivalents.

Note: When you are in binary mode, the addresses on the left are in hex, but represent BIT locations not BYTE locations.

Note: When done editing data, hit F8 to prevent further changes and end the edit.

4.1.2 Other Editor Functions

F1	= Goto new address	ESC	= Exit to main menu
TAB	= Switch modes (F1,F2,F3)	F10	= Help
F8	= Toggle edit enable	F9	= Enter DOS shell

To move immediately to an address:

Press <F1> input the goto address, then hit <ENTER>.

4.2.2 Keypad Functions

Up/Down arrow	= Up/down a line	^Pgup	= Scroll up 1000 bytes
Left/right arrow	= Left/right a column	^Pgdn	= Scroll down 1000 bytes
Pgup	= Scroll up a page	HOME	= Move to top of screen
Pgdn	= Scroll down a page	^HOME	= Go to address 0000 (home)

4.2.3 Quick-Move Keys (^ = Ctrl key)

^Q = Scroll up 10,000,000 bytes	^A = Scroll down 10,000,000 bytes
^W = Scroll up 1,000,000 bytes	^S = Scroll down 1,000,000 bytes
^E = Scroll up 100,000 bytes	^D = Scroll down 100000 bytes
^R = Scroll up 10,000 bytes	^F = Scroll down 10000 bytes
^T = Scroll up 1,000 bytes	^G = Scroll down 1000 bytes
^Y = Scroll up a page	^H = Scroll down a page
^U = Scroll up a line	^J = Scroll down a line

4.3 COPY, FILL, SEARCH, INVERT, SERIALIZE

The COPY, FILL and SEARCH commands manipulate the buffer data in the indicated manner.

<ALT-C> allows you to COPY a block of data from anywhere in the buffer to anywhere else in the buffer. <ALT-F> allows you to FILL a block of buffer with a string of hex or ASCII characters up to 30 characters in length.

<ALT-S> allows you to SEARCH any part of the buffer for a hex or ASCII value. You must precede any ASCII characters with an apostrophe ('), e.g. 'cdefghi, not just cdefghi.

To INVERT a block of buffer data, press <ALT-I> and input beginning and ending addresses. This function XOR's data within the specified range.

To use the SERIALIZATION feature, press <ALT-E> to input the address which will be incremented with the serial number. Then, with each <ALT-Z>, that specific location will be incremented one hex value. You can set up a macro to do this each time you program a device.

ALT-C	= Copy block of data	ALT-E	= Select serialization address
ALT-F	= Fill block of data	ALT-I	= Invert field of data
ALT-S	= Search block of data	ALT-Z	= Increment serial number

5.0 USING MACROS WITHIN THE EMP20.INI INITIALIZATION FILE

5.1 EMP20.INI OVERVIEW

The initialization file, EMP20.INI, preconfigures EMP20 before execution. When EMP20 is started, it loads information from EMP20.INI into memory. EMP20.INI lets you control or define items such as macros, file definitions, memory used and colors.

To change or add anything in EMP20.INI, edit it with any ASCII/non-document editor, then save it back to disk. Before editing EMP20.INI, make sure the original is backed up on the current disk, or at least on the original diskettes.

5.2 EMP20.INI COMMANDS- OVERVIEW

DMEM = Determines amount of disk space for the virtual buffer; i.e. size of EMP.VIR and the last address in buffer area. Ranges from 0 (64KB) to C (256MB).

MEM= Determines the amount of system memory to use for buffer space. Ranges: 0 = 64KB, 1 = 128KB, 2 = 256KB, 3 = 512KB.

Note: You should only have DMEM or MEM active at any particular time; MEM takes precedence if both are active.

PORT	= Sets port base for EMP20 before EMP20 starts
DFILE	= Sets default path and filename for Virtual buffer (EMP.VIR)
E20HELP	= Sets default path and filename for Help file (EMP20.HLP)
VIDEO	= Overrides automatic color/monochrome selection by EMP20
;=	= Term to specify that line is commented out, i.e. takes no action

In addition to these commands are MACRO, FILE and COLOR definitions.

Note: EMP20.INI must be located in currently logged directory; i.e., if EMP20.INI is in C:\EMP20 and EMP20 is executed from C:\ or any other directory, EMP20.INI will not load. This is also true for all help files used by EMP20.

If EMP20.INI is not on disk or does not contain the proper data fields, everything in the file is ignored and EMP20 uses the following default values for initialization:

DMEM	= 0	64KB area used for virtual buffer space
MEM	= 0	64KB used for system memory
PORT	= 378	Uses port 378H for EMP-20 port base

More than one .INI file may be used. To specify a different .INI file, type “EMP20 filename”, where “filename” is the .INI file to use. To use ALAN.INI, “EMP20 ALAN.INI” is entered. Otherwise, EMP20 defaults to EMP20.INI, or in the case of finding no .INI file, defaults to preset values.

5.3 EMP20.INI COMMANDS- DETAILED DESCRIPTION

5.3.1 DMEM

DMEM determines how large the virtual buffer will be. DMEM ranges from 0-C. Each number makes the buffer twice the size of the previous number. Buffer sizes range from 64KB to 256MB, and are as follows:

0	= 64KB (smallest possible buffer)
1	= 128KB
2	= 256KB
3	= 512KB
4	= 1MB
5	= 2MB

6	= 4MB
7	= 8MB
8	= 16MB
9	= 32MB
A	= 64MB
B	=128MB
C	= 256MB

To set a buffer of 1MB, set “DMEM=4” inside EMP20.INI. If EMP20.INI or the DMEM option doesn’t exist, the default buffer size is 64KB.

The buffer size set with DMEM affects two things:

1. The maximum allowed address within the buffer editor. The memory size selected by DMEM also determines the buffer range. If DMEM=2 is selected, the buffer will be 256KB in length (the maximum address allowable will be 3FFFF).
2. The actual file size on disk. When EMP20 opens a virtual buffer, it creates a file called EMP.VIR to the length of the buffer size specified. Therefore, the buffer on disk (EMP.VIR) will be as large as the buffer. So, if DMEM=2, EMP20.VIR will be 256KB. If DMEM=8, EMP20.VIR will be 16MB. The disk path EMP.VIR uses to load from and store to is controlled by DFILE. If the DMEM settings are set smaller than the size of DFILE, then only part of DFILE will be loaded. See section 5.3.3 for more information on DFILE.

5.3.2 MEM

MEM determines how large an in-memory buffer will be. EMP20 keeps the specified amount of memory open, reserving 64-KB blocks. MEM ranges from 0-3; each number makes the buffer twice the size of the previous number. Buffer sizes range as follows:

0	= 64KB (smallest possible buffer)
1	= 128KB
2	= 256KB
3	= 512KB (largest possible buffer)

To reserve 256KB of memory, set “MEM=2” inside EMP20.INI. If EMP20.INI or the DMEM option does not exist, the default amount of memory reserved is 64KB.

On a 640KB system with all of memory free, setting “MEM=2” to reserve 256KB is easily done.

Note, however, that although memory for use with the MEM command is available, using it limits the flexibility of the computer. For instance, using a shell within EMP20 may be impossible (due to lack of memory), or even if it is possible, some programs may not fit with the limited memory available. EMP20 does not utilize extended or expanded memory.

You can either have MEM active or DMEM active, you must comment out the other command (either DMEM or MEM) with a ";=". If both are active, MEM overrides.

5.3.3 DFILE

DFILE automatically sets default path and file name for the to and from path to the virtual buffer file on your HDD. The virtual buffer is basically the data buffer.

Normally, EMP20 uses the file EMP.VIR on the current directory for virtual buffer storage. However, unless EMP20 is always executed from the same directory, it creates new EMP.VIR files wherever it's executed from.

For example: Executing EMP20 in the directory C:\EMP20 creates the file C:\EMP20\EMP.VIR, executing EMP20 in the directory C:\MYDIR creates the file C:\MYDIR\EMP.VIR.

EMP allows EMP.VIR to stay current, i.e. the buffer is kept current, even when EMP20 exits to DOS. Also, EMP.VIR can take up to 16MB of space, and having multiple EMP.VIR files on disk is unnecessary.

Setting DFILE eliminates both of these problems. To set a virtual buffer file, simply enter "DFILE=<filename>" where <filename> is any filename. This filename takes the place of EMP.VIR. A path may also be entered in <filename>. For instance, entering "DFILE=C:\MYDIR\EMP.VIR" causes EMP20 to use C:\MYDIR\EMP.VIR for its virtual buffer regardless of which directory EMP was executed from.

Note: To use the same virtual buffer file (i.e. EMP.VIR), the .INI file must always specify the virtual buffer. When EMP20 is executed, EMP20.INI is searched for in the current directory. If EMP20.INI is not found, EMP.VIR in the current directory is defaulted to. In different directories, EMP20.INI should specify DFILE. Otherwise, if the .INI file is in a specific directory, this may be entered on the command line. For instance, if EMP20.INI is in C:\EMP20, entering "EMP20 C:\EMP20.INI" causes EMP20 to default to C:\EMP20\EMP20.INI for the .INI file, even if there is a .INI file in the current directory.

5.3.4 HELP

HELP allows the help file "EMP20.HLP" to be specified in another directory. Usually, EMP20 looks for EMP20.HLP in the current directory. If EMP20.HLP is not found, this has the effect of eliminating all HELP from

EMP20 and also removes the socket/family module overlays that appear prior to programming or reading/verifying a device.

EMP20 may be executed from any directory, as long as an EMP20.INI file is available, either in the directory or with a path name to one in the command line. It is unnecessary to have EMP20.HLP in each directory. To set a path for the help file, enter “HELP=<filename>“ in EMP20.INI, where <filename> is any valid path, followed by EMP20.HLP.

Note: see section 5.9 for details on calling EMP20 without an EMP20.INI file in that directory

If EMP20.HLP is located in the directory C:\EMP20, and EMP20 is executed from C:\MYDIR, EMP20 will not find the help file.

Setting “HELP=C:\EMP20\EMP20.HLP” in the EMP20.INI file allows EMP20 to look to C:\EMP20\EMP20.HLP for the help file, solving the problem.

5.3.5 VIDEO

When EMP20 is executed, it determines the difference between color and monochrome systems, and adjusts all colors accordingly. However, EMP20 cannot determine whether systems with color cards (i.e. CGA, MCGA, EGA, VGA) have color monitors attached to them. Monochrome monitors attached to color systems will not display some colors used with EMP20. On systems with color cards, but with monochrome monitors, the VIDEO switch may be set to counteract the usage of color in EMP20. To do this, set VIDEO=M, where

VIDEO = M sets a default of monochrome

VIDEO = C sets a default of color

Note: see section 5.6 COLORS for more information on color selection.

VIDEO works only with pop-up menus, overlays and the HELP system. VIDEO has no effect on the colors described in section 5.6 (Defining Colors). Note, these are preset for monochrome systems and should not need adjustment.

5.4 WRITING MACROS

Macros are defined in the form of:

m<n>=macro name,macro text

Where <n> is the macro number being defined. Macros can be numbered from 0-I (i.e. 0-9, A-I).

The macro number must be followed by an equals ("=") sign. Note that there are no spaces between <n> and "macro name". EMP20 will not recognize the macro, and therefore not load it into memory, if spaces occur.

"Macro name" is the name of the macro printed on the macro list and is for printing only. The macro name may be any length, but is concatenated to the number of characters used in the menu. "Macro name" must be followed by a comma (",") (as before, without spaces).

"Macro text" is the actual macro input. Macro text is usually straight-forward. For instance, to change Buffer Range-D (K) to 010000-01FFFF with a macro, enter the keystrokes into EMP20.INI as you would in EMP20; i.e. "K010000^M01FFFF".

A more explicit example might be macro number 0 entitled "buffer macro". The definition would look like this:

m0=buffer macro,K010000^M01FFFF

Keys not normally associated with ASCII may be entered as well. Keys such as the arrow keys, page up, page down, insert, etc. may be used. Since these type are not single-key sequences, they must be entered differently.

To use these keys in EMP20, preface the code with a "$0". As example, "$048" is used for the up-arrow key. Make sure there are always three numbers after the "$", e.g. if a key's code is 3H, type "$003" as input. See the hex equivalents of extended ASCII characters in section 5.7.3.

Typically, these keys are not used. For instance, up-arrow is mostly used for moving around in menus to make a selection. Alternatively, hitting the desired selection number (or letter) is faster.

Another key of special value is <ENTER>. <ENTER> is one of the control-character set, i.e. ASCII values under 20h (spacebar). To input a control character, type "^" before its associated ASCII value (section 5.7.3), i.e. type "^M" (control-M) to simulate <ENTER>.

As an example, selecting a part might look like this:

m1=part selection macro,516^M3^M

Where macro #1, "part selection macro", selects manufacturer #16 and device #3.

Macros are limited to 256 bytes total size (macro name + macro text).

5.4.1 Special Initial Macro

There is one other macro besides macros (0-I). This is a macro that is defined as:

m=macro text

This macro is an initial macro that executes upon entering EMP20. For example, to automatically select a part (i.e. keystrokes 5 40 <ENTER>) upon execution of EMP20, input:

m=516^M3^M

Initial macros are also limited to 256 characters.

5.4.2 Activating a Macro

To activate a macro, hold the <RIGHT SHIFT> key while pressing the number/letter desired.

Note: when manufacturers and/or parts are added to the menu, numbers in the above macros may need to be edited.

5.5 SETTING FILE DEFINITIONS

File definitions allow you to set all file specifications with a single key. File definitions are declared in this manner:

f<n>=title,file type,filename,buffer start,buffer end,file start,file end

<n> is the file number. Files are numbered 0-I (i.e. 0-9, A-I).

Note: There's no space between the file number <n> the equal sign ("=") and the "title". EMP20 will not recognize the file definition if spaces appear on either side of the "=" sign.

"title" is only used for the file menu, it has no other use. "title" may be any length, but is cutoff in the menu display.

A comma (",") must follow every definition. Spaces are not allowed (i.e. "title,file type", not "title , file type"), otherwise EMP20 will not recognize the file definition.

"file type" is the default file type of the file being declared. File types are Binary, Motorola S Record, Intel Hex and plain ASCII-Hex.

The "file type" indicator is one-letter, and is as follows:

B	= Binary file	M	= Motorola S record
I	= Intel Hex file	H	= ASCII-Hex file
J	= JEDEC file		

"filename" is the actual filename, including path.

For example, if the current directory is C:\EMP20, and FLASH.DAT is in C:\DATA, the filename would be "C:\DATA\FLASH.DAT".

Buffer start and Buffer end refer to Buffer Range-F (S), i.e. simulates 'S' option in EMP20. To choose a range from 1000-1FFF, enter "1000,1FFF" for "buffer start, buffer end".

"file start" and "file end" refer to File Range (T). They behave exactly as above (i.e. "1000,1FFF").

Examples:

f1=data file 1,b,c:\b.dat,0,fffff,0,fffff

This specifies that the file definition named "data file 1" (file 1) will be a binary file with the name "C:\B.DAT". Buffer Range-F (S) and File Range (T) ranges will be 0-FFFF.

f2=Alan's file,m,c:\alan\this.dat,1400,15FF,0,1FF

This specifies that the file entitled "Alan's file" (file 2) will be a Motorola S Record file with the name "C:\ALAN\THIS.DAT". Buffer Range-F (S) will be 1400-15FF, File Range (T) will be 0-1FF.

5.5.1 SPECIAL INITIAL FILE

There is another file definition available beyond the normal file definitions 0-I. This is an initial file definition, used when EMP20 is first brought up. The only other difference is that there is no name or number associated with it. The format is:

f=,file type,filename,buffer start,buffer end,file start,file end

Example

f=,m,c:\alan\this.dat,1400,15FF,0,1FF

Note that there's a comma preceding "file type" ("m" in this example). When this definition as active, EMP20 defaults to the above values upon execution.

5.5.2 ACTIVATING A FILE DEFINITION

To activate a file definition, hold the <LEFT ALT> key while pressing the number/letter desired.

5.6 DEFINING COLORS

Most colors used in EMP20 may be re-defined. To define a color in EMP20.INI, the format is:

C<n>=<Mono color><Color color>

Where:

<n> is the color associated with a EMP20 function (e.g. pop-up menus, buffer editor, etc.).

<Mono color> is the color to be used for monochrome displays. Usually only 07h (normal text) and 70h (inverted text) are used with monochrome displays.

<Color color> is the color to be used on color systems. Each color is defined in a 2-digit HEX format, i.e. the first digit is the BACKGROUND, the second digit is the FOREGROUND (printed text). The last bit of the first digit is used for blinking text. For example:

```
COLOR

Bits 0-3 are the Foreground Color --------------------------------------
Bits 4-6 are the Background Color -----------------------------  |         |
Bit 7 controls Blinking--------------------------------------- |     |
MONO                                                            |     |         |
Bits 0-3 are the Foreground Color ----------- ----|         |     |
Bits 4-6 are the Background Color ------           |         |     |         |
Bit 7 controls Blinking------------        |           |         |     |         |
                                       |     |           |         |     |         |

                                     <0 / 0 0 0 / 0 0 0 0>  <0 / 0 0 0 / 0 0 0 0>
```

Colors are as follows:

0	= BLACK	6	= BROWN	C	= BRIGHT RED
1	= BLUE	7	= WHITE	D	= BRIGHT MAGENTA
2	= GREEN	8	= DARK GRAY	E	= BRIGHT YELLOW
3	= CYAN	9	= BRIGHT BLUE	F	= BRIGHT WHITE
4	= RED	A	= BRIGHT GREEN		
5	= MAGENTA	B	= BRIGHT CYAN		

The foreground color may be any color above. However, since one bit is used for the blink indicator, only the first 8 (0-7) may be used for the background color.

For instance, to used the foreground colors WHITE, YELLOW and GREEN with a BLUE background, the colors would be defined as:

1F—WHITE (bright) on BLUE

1E— YELLOW (bright) on BLUE

1A— GREEN (bright) on BLUE

To make them all blink:

9F—Blinking WHITE (bright) on BLUE

9E—Blinking YELLOW (bright) on BLUE

9A—Blinking GREEN (bright) on BLUE

The selectable colors (0-A) are defined in EMP20 as follows:

C0	= text with titles	C1	= menu options
C2	= body text	C3	= buffer editor items
C4	= extra control	C5	= editor's hex display
C6	= editor's ASCII display	C7	= borders
C8	= pop-up borders	C9	= pop-up items
CA	= pop-up selections		

Example:

C6 = 701F Defines editor's ASCII display in MONO as BLACK on WHITE (inverted), in COLOR as WHITE on BLUE.

Note: On some systems using a color card with a Monochrome monitor or some LCD display systems the conversion of color signals to gray shades may make some menus unreadable with certain color choices. It is recommended that all settings be set to 0707 or 7070.

5.7 MACRO-KEY CONVERSION

5.7.1 SPECIAL MACRO KEYS

There are three special keys used in macros: "$", "@" and "~". "$" is used to specify extended ASCII keys, "@" is used in conjunction with "$" to specify another macro, and "~" is used to bypass error menus.

Some ASCII characters may not be used directly when specifying macros in EMP20.INI. For instance, it is not possible to place an <ENTER> in a macro directly, since in a text file this would start a new line.

5.7.2 CONTROL CHARACTERS

To specify a control character such as "ENTER", use a "^", followed by its ASCII equivalent. For instance, <ENTER> is "Control-M" (^M).

Control Characters in EMP20.INI macros:

ESC	= ^[
ENTER	= ^M

For example, "M1=55^M1^M^[Y" sets macro 1 to select part #5, programs it and returns to DOS. "^M" represents <ENTER> sequences, while "^[" represents an <ESCAPE> sequence.

5.7.3 EXTENDED ASCII CHARACTERS

extended ASCII characters are keys not represented normally by the keyboard. These include all keypad functions (up/down, etc.), function keys, ALT-keys, etc. To specify an Extended ASCII character in a macro definition, enter “$0” followed by its Hex equivalent (listed below).

To select file definition 1 within a macro, “$078” simulates the <ALT-1> key and loads in file definition 1.

Hex equivalents of extended ASCII characters:

Alt-1	= $078	Alt-A	= $01E	F1	= $03B
Alt-2	= $079	Alt-B	= $030	F2	= $03C
Alt-3	= $07A	Alt-C	= $02E	F3	= $03D
Alt-4	= $07B	Alt-D	= $020	F4	= $03E
Alt-5	= $07C	Alt-E	= $012	F5	= $03F
Alt-6	= $07D	Alt-F	= $021	F6	= $040
Alt-7	= $07E	Alt-G	= $022	F7	= $041
Alt-8	= $07F	Alt-H	= $023	F8	= $042
Alt-9	= $080	Alt-I	= $017	F9	= $043
Alt-0	= $081	Alt-S	= $01F	F10	= $044

PGUP	= $049	PGDN	= $051	HOME	= $047
END	= $04F	INS	= $052	DEL	= $053

UP ARROW	= $048	DOWN ARROW	= $050
RIGHT ARROW	= $04D	LEFT ARROW	= $04B

5.7.4 CHAINING MACROS

Macros may execute other macros. To do this, add “$@<macro#>“ at the end of the macro definition. <macro#> is any valid macro from 0-I.

For example, “M1=1^M3^M$@2” programs the current device, verifies it, then executes macro 2.

Macro Definitions are as follows:

Right-Shift-1	= $@1	Right-Shift-A	= $@A
Right-Shift-2	= $@2	Right-Shift-B	= $@B
Right-Shift-3	= $@3	Right-Shift-C	= $@C

Right-Shift-4	= $@4	Right-Shift-D	= $@D
Right-Shift-5	= $@5	Right-Shift-E	= $@E
Right-Shift-6	= $@6	Right-Shift-F	= $@F
Right-Shift-7	= $@7	Right-Shift-G	= $@G
Right-Shift-8	= $@8	Right-Shift-H	= $@H
Right-Shift-9	= $@9	Right-Shift-I	= $@I
Right-Shift-0	= $@0		

5.7.5 MACRO ERROR HANDLING

When an error occurs, any macro currently executing in memory is halted.

If a macro is defined by “M1=1^M0^M$@1” (which programs and erases a device, then re-executes the same macro), it will halt if an error occurs.

To disable halting on errors, use “~” to bypass errors. As an example, “M1=1^M~0^M$@1” allows the macro to continue even through an error condition. This happens because the error menu, when displayed, looks for an input. If a “~” is entered, this signals EMP20 to disregard the error and not halt the macro execution. If no error occurs, “~” is ignored.

5.8 SPECIAL EMP20.INI COMMANDS

5.8.1 CHECKSUM FOR 16 BIT WIDE DEVICES

The EMP20’s default method for generating a checksum for 16 bit devices is the compliment (cmpl) of an additive summation of each 16 bit word within the device. By adding the statement: CHECKSUM 8=Y into the EMP20.INI file, the software will generate an 8 bit checksum on the bytes within a 16 bit device. This will correspond with the checksums generated on many other programmers.

5.8.2 SOUND ON THE EMP-20

SOUND tells the software to beep after programming, reading, operation complete, etc. Placing the statement SOUND in the EMP20.INI set to no, ‘N’ after the equals sign, turns off the beeps. The default value is YES and is not necessary to add.

For example:

```
SOUND=N
```

5.9 CALLING EMP20 FROM ANY DIRECTORY

The EMP20.EXE when starting must be able to load certain parameters. These parameters are contained within an EMP20.INI file so it must be accessible as part of the startup. The simplest way is to start EMP20 from a directory that also contains all the other files initially supplied by Needham's Electronics. If you need to be able to load or call EMP20 from any directory with out establishing EMP20.INI files in all the possible directories, the following must method must be used. When starting EMP20 a path to a valid EMP20.INI file must be a part of the command line. For instance, if EMP20.INI is in C:\EMP20, entering "EMP20 C:\EMP20.INI" from the root directory causes EMP20 to default to C:\EMP20\EMP20.INI for the .INI file, even if there is a .INI file in the current directory. The EMP20.INI file, no matter where it is located must contain the paths to all the other files that EMP20 needs, i.e. EMP20LIB.DAT, EMP20.HLB, etc.. All of these requirements could be included in a batch file

6.0 OVERVIEW OF FILE FORMATS

6.1 BINARY FORMAT

The binary format is usually used to save memory data, which has been read into the buffer, to disk. Also you can read back these files as binary files. This format is just for a binary string of data which does not have any other header or footer data, it is just raw binary data which will be read into the buffer and displayed as Hex values.

6.2 INTEL HEX FORMAT

Each line in an Intel hex file corresponds to a "record". Each record tells the loader (i.e. EMP20) what to do with that line. Intel hex format uses the following record types:

Data Record

Start Segment Address Record

Extended Segment Address Record

Extended Linear Address Record

Start Linear Address Record

End-of-File Record

All these record types follow the format as shown below. These record types will be further described in section 6.2.7.

```
:    09    514E    00    50AA55A55A20202020    29
|    |     |       |     |                     |
Record Mark------      |      |         |      |                              |
Record Length---------|      |         |      |
Address-----------------------|         |      |
Record Type---------------------------|      |
Data------------------------------------------  |
Checksum---------------------------------------------------------------------
```

6.2.1 RECORD MARK

Each record starts with a colon (":"), called a record mark. If the record mark does not exist, the line should be ignored by the loader. When a record is specified, it must follow a certain order so that the loader will read it properly:

When entering a record, all data (except the record mark,":") is entered in ASCII hex. That is, when the word "byte" is used, it refers to the two-digit hex number; i.e. "01 02 03 04" is 8 digits wide, but is only 4 bytes of data when converted.

6.2.2 RECORD LENGTH

The record length is always one byte and determines how wide the data field is. For instance, if there are 10 bytes of data in the current record (i.e. on the current line), the record length will be "0A". Remember that all bytes are entered as their 2-digit ASCII hex values. "A" is invalid and must always be "0A".

6.2.3 ADDRESS FIELD

The address field is two bytes (i.e. 4 digits) in length, and tells the loader where to put the current line of data. Address fields are only valid in data records. For record types that do not use the address field, the address field is always "0000".

6.2.4 RECORD TYPE

The record type is one byte in length (2 digits). Record types are as follows:

00	Data Record
01	End-of-File Record
02	Extended Segment Address Record
03	Start Segment Address Record
04	Extended Linear Address Record (32-bit format only)
05	Start Linear Address Record (32-bit format only)

The record type determines what the current record (i.e. the current line) will instruct the loader to do. For instance, a record type of “00” specifies that this record contains data; record type “01” terminates loading of the file and so on.

6.2.5 DATA FIELD

The data field contains the actual data for the loader. If the record type is a data record, the data field will contain actual file data. If the record type is a start-address or extended-address, the data field will contain these addresses. The EOF record does not contain a data field. In this case the data field is non-existent.

The data field is directly related to the record length. The record length determines how large the data field will be. For instance, if the record length is 05, the data field will contain 5 bytes of information.

6.2.6 CHECKSUM

The last byte of the record is a 8-bit checksum of the entire record (excluding the record mark “:” and the checksum itself). The checksum simply adds all of these numbers and takes the negative as a checksum. For instance, the line

:09514E0050AA55A55A2020202029

will have the checksum of “29”. Note that “29” is the last byte of the record. When a line is loaded, the negative of the sum of all bytes prior to the checksum should be equal to the checksum. Or, in another way, the sum of all bytes including the checksum should be 0. If this is not the case, the line should be reloaded and tried again. If the record has an incorrect checksum, the appropriate error message will be displayed, and EMP20 will discontinue loading the file.

6.2.7 RECORD TYPES

6.2.7.1 Data Records

The data record specifies that the data field in the current record will be file data. This is the actual data that is being loaded.

```
                  :   09  514E  00  50AA55A55A20202020   29
                      |   |     |   |                    |
Record Length------|   |     |   |
Address--------------------|     |   |
Record Type------------------------|   |
Data-------------------------------------- |

Checksum-------------------------------------------------------------
```

Note that the record length is 09 and there are 9 bytes of data in the data field. Since the address is 514E, data for this record will be loaded sequentially starting at address 514E (relative to the start of the buffer).

6.2.7.2 End-of-File Record

The EOF record specifies the end of the file. At this point, the loader discontinues loading the file.

```
                              :        00        0000     01        FF
                                       |         |        |         |
Record Length------------------------- |         |        |
Address(Blank)---------------------------------- |        |
Record Type (01=EOF)------------------------------------- |
Checksum------------------------------------------------------------
```

Note that the data field is 0 bytes in length and that the address contains “0000”.

There is another form of EOF record, referred to as the Alternate EOF Record

```
                              :        00        0000     00        00
                                       |         |        |         |
Record Length------------------------- |         |        |
Address----------------------------------------- |        |
Record Type--------------------------------------------   |
Checksum------------------------------------------------------------
```

Although the record type is a data record, there is no data so the record is taken as the end of file. EOF records are not variable records. That is, they always appear as they do here.

6.2.7.3 Extended Segment Address Record

The extended segment address record specifies the segment that data will be loaded into. For instance, if the segment is specified as “58A2” the loader will change the segment to 58A2H, which will cause data to be loaded at address 58A20H. In the following data records, the address specified in the address field will be added to the new extended address. For example, if the address in the data record is “12C4” and the extended address is “58A2”, the actual address will be 58A2:12C4 or 58A20+12C4 or 59CE4.

The extended address will continue to offset data record addresses until a new extended address record is specified. The record length is only “02” since the segment address is the only data and is only 2 bytes long.

```
                              :        02       12C4    02       58A2    02
                                       |        |       |        |       |
Record Length------------------------- |        |       |        |
Load Offset-------------------------------------|       |        |
Record Type-------------------------------------------- |        |
Upper Segment Base Address-------------------------------------- |
Checksum---------------------------------------------------------------------
```

6.2.7.4 *Start Segment Address Record*

The start segment address record specifies the execution start address for the object file. The value given is the 20-bit segment address for CS and IP registers of 8- and 16-bit Intel processors.

```
                      :     04    12C4    03    58DFE4A2    02
                            |     |       |     |           |
Record Length---------------|     |       |     |
Load Offset---------------------- |       |     |
Record Type-------------------------------|     |
CS/IP Register Contents-------------------------|
Checksum---------------------------------------------------------
```

6.2.7.5 *Extended Linear Address Record*

The extended linear address record is for 32-bit processors only. It is used to specify bits 16-32 of the linear base address (LBA). It may appear anywhere within a 32-bit hexadecimal object file. Its value remains in effect until another extended linear address record is encountered.

```
                              :        02       12C4    02       58A2    02
                                       |        |       |        |       |
Record Length------------------------- |        |       |        |
Load Offset-------------------------------------|       |        |
Record Type-------------------------------------------- |        |
Upper LBA------------------------------------------------------  |
Checksum----------------------------------------------------------------------
```

6.2.7.6 Start Linear Address Rec ord

The start linear address record is used to specify the execution address of a 32-bit Intel object file only.

```
                    :    04   0000    02   58ED353A   46
                         |    |       |    |           |
Record Length---------|    |       |                   |
Load Offset-----------------|      |                   |
Record Type-------------------------- |                |
IEP Register Contents---------------------             |
Checksum-------------------------------------------------
```

6.3 MOTOROLA S RECORD FORMAT

Motorola S record format is very similar to Intel hex format. The source file is an ASCII hex file, with each line in the file representing a record.

```
                S  1  23  08B0  737420636F6D706C6574652073657420  48
                |  |  |   |     |                                  |
Record Mark-       |  |   |     |                                  |
Record Type-----      |   |     |                                  |
Record Length------|      |     |
16-bit Address------------|     |
Data------------------------------- |
Checksum--------------------------------------------------------------------------
```

6.3.1 RECORD MARKER

The record marker in a Motorola S record is the “S”. That is, each line is preceded by an “S”.

6.3.2 RECORD TYPE

Following the “S” is the record type. Unlike Intel files, this record type is only one digit in length. For instance, record type 1 is represented by “S1<...>“, not “S01<...>“. The record types are discussed in section 6.3.6.

6.3.3 RECORD LENGTH

Following the record type is the record length. It is a two-digit hex value specifying the number of data bytes in the record, including the address and the checksum. Each byte is represented as two hex characters. After the record length, the address follows. This determines where in memory the current record (i.e. the current line) will load. Instead of using extended address record types to load data above 64KB, the Motorola file uses the actual address. For this reason, the address field may vary in size. This size of the address field may be 4, 6 or 8 bytes, (or 16, 24 or 32 bits), and is determined by the record type (discussed below).

6.3.4 FILE DATA

Following the address is the actual file data. The number of bytes of data is determined by the record length indicator. The EOF record does not contain file data.

6.3.5 CHECKSUM

The checksum is the last byte (2 digits) in the record, and follows the file data. The checksum is calculated differently than the Intel hex format. All bytes with the exception of the “S” record type digit, and the checksum itself, are added and then negated.

6.3.6 RECORD TYPES

Motorola S record files have the following record types:

1	16-bit Addressing
2	24-bit Addressing
3	32-bit Addressing
7	EOF Record
8	EOF Record
9	EOF Record

6.3.6.1 16-bit Addressing

Files with less than 64KB only need 16 bits to address every location in the file. When files are this small, the record type used is “1”.

```
            S  1  23  08B0  737420636F6D706C6574652073657420  48
               |      |
Record Type 1-    |
16-bit Address-----------
```

6.3.6.2 24-bit Addressing

With files greater than 64KB, more bits are needed to address every location within the file. If 24 bits are needed, record type "2" is used to specify 24-bit addressing.

```
                  S   2  24   11408A   50AA55   2020   06
                      |       |
Record Type 2-------          |
24-bit Address-----------------
```

6.3.6.3 32-bit Addressing

In the case where more than 24 bits are needed, record type "3" may be used to specify 32-bit addressing:

```
                  S   3  25   020057AE   50AA55   2020   04
                      |       |
Record Type 3-------          |
32-bit Address------------------
```

Note that in each case, the record length increased. Specifically, the record length for 16-bit addressing was 23. When the record length increased to 24-bits, the record length moved to 24. The record length increased to 25 when 32-bit addressing was used. This is due to the fact that one byte in each case was added due to the extra byte in the address. Since the record length includes the address, it must be increased also.

6.3.6.4 End-of-File Record

Record types 7, 8 and 9 represent the end of the file. Record type "3" (32-bit addressing) uses EOF record type 7, record type "2" (24-bit addressing) uses EOF record type 8, and record type "1" (16-bit addressing) uses EOF record type 9.

An example of record type 9:

```
                          S       9       03       0000      FC
                          |       |       |        |         |
Record Marker---------            |       |        |         |
Record Type---------------------- |       |        |
Record Length------------------------------|        |
Null Address------------------------------------------ |
Checksum---------------------------------------------------------
```

6.4 HEX (ASCII) FORMAT

This format reads in a file which has ASCII characters relating to hex values. That is, if you were to type in hex values "0-F" in an editor, these would be ASCII values which will be converted to their corresponding hex values in the buffer.

6.5 NOTE ON FILE POINTERS

When using EMP20 with Intel hex files, the message "0 bytes loaded" may appear. In this case, the offset in record type "02" may be out of range for EMP20. For instance, if the extended segment address record is at "F8000" and File Range (T) settings are from 0-FFFF, no data would be loaded. In this case, change the file offset in 'T' to coincide with the extended segment address range in the Intel hex file. That is, in this case, set the value of the start-address box (T) to "F8000" (the ending address adjusts automatically). This then allows the file to load properly.

In the case of Motorola S Record files, there is no extended address. The address is encoded directly in the address field. In this case, move 'T' to the initial address value.

6.6 JEDEC STANDARD FORMAT

The JEDEC file format defines the standard for transferring data for PLD's from a development system to a device programmer. The standard defines fuse, test, identification and comment information in an ASCII representation. It does not define device specific information such as algorithms and methods for actually programming the device.

The JEDEC standard utilizes a transmission protocol that has error checking in the form of a checksum. A start transmission character and an end transmission character is used when a device programmer shares the serial port with a programmer.

The following is an example of a JEDEC standard format file:

```
<STX>File for PLD 12S8 Created on 11-NOV.-91 5:08PM

6809 memory decode 123-456-789

Joseph Shmo Pretzel Logic Corp.

QP20* QF448* QV8*
F0* X0*
L0000111110111111111111111111111*
L0028101111111111111111111111111*
L0056111011111111111111111111111*
L0112010101110111101111111111111*
L0224010101111011101111111111111*
L0336010101110111011111111111111*
V0001000000XXXNXXXHHHLXXN*
V0002010000XXXNXXXHHHLXXN*
V0003100000XXXNXXXHHHLXXN*
V0004110000XXXNXXXHHHLXXN*
V0005111000XXXNXXXHLHHXXN*
V0006111010XXXNXXXHHHHXXN*
V0007111100XXXNXXXHHLHXXN*
V0008111110XXXNXXXLHHLXXN*
C124E*<ETX>8946
```

For more information on the JEDEC standard, contact Needham's Electronics Inc. at (916) 924-8037.

7.0 INSTALLATION AND TROUBLESHOOTING

7.1 INSTALLATION

You must be in a DOS environment and have one megabyte of space available on your hard drive. Additional space will be required for EMP.VIR, the main buffer file, if internal RAM is not used. To install the software:

Load the supplied diskette into a floppy drive and switch to that drive. Then type:

```
x:\>INSTALL <programmer name> <from drive> <to drive>
```

<programmer name> = EMP20

<from drive> = Floppy drive, (with colon), you are loading from

<to drive> = Hard drive, (with colon), you are loading to

For example, to install EMP20 software from drive A: to drive C:, from the A: drive prompt you would type:

```
A:\>INSTALL EMP20 A: C: <ENTER>
```

The Batch File will create an EMP20 directory, switch to it, load the main compressed software file, expand it and terminate in the EMP20 directory that it just created.

To install and start operation of the EMP-20 programmer, plug the wall transformer, (supplied), into a 110VAC, 60Hz source, (for US operation), and the transformer output plug into the rear of the EMP-20. Connect a standard printer cable, (supplied), to connector in the rear of the EMP-20 and the other end to your PC. Turn the EMP-20 on with the on/off switch in the rear of the EMP-20. From the EMP20 directory, described above, type EMP20.

The opening screen will appear, follow the instructions on the screen. The EMP software will attempt to communicate with the EMP-20 though all the PC's standard printer port addresses. If you are using a printer on another port at the time you may notice it acting a bit strange. This is normal for first time installations and will not occur the next time the software is used. If the printer port hooked to the EMP-20 is a non standard printer port address, you will be asked to enter the base address of the port by the software. When communication has been established, the correct port address is stored in a created file, (EMP.IOP), for future use and the main menu appears.

Important!: Software is shipped and down loaded in self expanding compressed form. To upgrade the EMP20 software, copy the file EMP20ARC.EXE from the Needham's Electronics Website on the Internet by using the following address: "http://www.needhams.com".When upgrading, you may not want to over write your EMP20.INI, just type no when ask if you want it over written. Because the EMP20.INI holds your custom operating instructions, you may want to back it up in another file or location.

When you are ready to start programming devices, you will need to install the correct Family Module (shown with your device selection). They are installed into the SIMM connector to the left of the ZIF connector at the lower left hand corner of the EMP-20 unit. Insert the Family Module with the leading edge facing downward at a 45 degree angle. When fully inserted push down the trailing edge until it is flat or horizontal. Make sure the aligning pins of the SIMM connector are in the Family Module's alignment holes.

7.1.1 EMP-20 FILES

The following files are supplied with your new programmer. They are compressed into a self extracting file called EMP20ARC.EXE. When uncompressed, the following list will be in your directory. This process may be automated by using the INSTALL.BAT file, (listed at the end of this section).

-- EMP20.EXE—

EMP20.EXE is the main executable file. This is the software that allows you to read, program, verify, the device. To execute EMP20, simply enter " EMP20 <CR> :"

-- EMP20.INI—

EMP20.INI is an optional file that is read in by EMP20.EXE upon execution. EMP20.INI contains information such as default port-base (see EMP20.IOP), Macros, File definitions, Colors, Memory configurations, etc.

To use another .INI file, type EMP20 <filename>, where <filename>   is the name of the .INI file to use. For instance, to use MY.INI, enter: "EMP20 MY.INI <CR>

.Note: EMP20.INI must be located in currently logged directory; i.e., if EMP20.INI is in C:\EMP20 and EMP20 is executed from C:\ or any other directory, EMP20.INI will not load. This is also true for all help files used by EMP20.

If EMP20.INI is not on disk or does not contain the proper data fields, everything in the file is ignored and EMP20 uses the following default values for initialization:

DMEM	= 0	64KB area used for virtual buffer space
MEM	= 0	64KB used for system memory
PORT	= 378	Uses port 378H for EMP-20 port base

-- EMP20.IOP—

EMP20.IOP contains the last IO-PORT configuration. Whenever the Port Base is changed in EMP20, EMP20 writes the new information to EMP20.IOP. When EMP20 is reloaded, it reads EMP20.IOP and automatically sets the new port base.

This may be overridden by deleting EMP20.IOP or setting the 'PORT' value in EMP20.INI (see Help in main-menu commands). EMP20.IOP also contains the last data-encryption array stored in memory.

-- EMP20.VIR—

This is where EMP20 keeps it's virtual buffer on disk. When a file is loaded into memory, it is kept current in EMP.VIR. Thus, when DOS is returned to, the buffer still remains in EMP.VIR.

-- EMP20.HLP—

EMP20.HLP is the where all help-text is stored. IF you are running EMP-20 from a floppy disk, the system will appear to be slow whenever the help system is used. This is due to the slow disk-access time of the floppy, and since DOS must reload directory information every time the disk is not used for a few seconds.

```
-- INSTALL.BAT—
@echo off
if "%2" == "" goto SYNTAX
if "%3" == "" goto SYNTAX
if "%2" == "%3" goto SYNTAX
if "%1" == "PB10" goto PB10
if "%1" == "pB10" goto PB10
if "%1" == "Pb10" goto PB10
if "%1" == "pb10" goto PB10
if "%1" == "PB-10" goto PB10
if "%1" == "pB-10" goto PB10
if "%1" == "Pb-10" goto PB10
if "%1" == "pb-10" goto PB10
if "%1" == "emp20" goto EMP20
if "%1" == "emP20" goto EMP20
if "%1" == "eMp20" goto EMP20
if "%1" == "eMP20" goto EMP20
if "%1" == "Emp20" goto EMP20
if "%1" == "EmP20" goto EMP20
if "%1" == "EMp20" goto EMP20
if "%1" == "EMP20" goto EMP20
if "%1" == "emp-20" goto EMP20
if "%1" == "emP-20" goto EMP20
if "%1" == "eMp-20" goto EMP20
if "%1" == "eMP-20" goto EMP20
if "%1" == "Emp-20" goto EMP20
if "%1" == "EmP-20" goto EMP20
if "%1" == "EMp-20" goto EMP20
if "%1" == "EMP-20" goto EMP20
if "%1" == "sa" goto SA
if "%1" == "sA" goto SA
if "%1" == "Sa" goto SA
if “%1” == “SA” goto SA
:SYNTAX
echo -----------------------------------------------------------------------------------
echo |   SYNTAX:        (to install from A: drive to C: drive)                           |
echo |                                                                                   |
echo |   INSTALL PB10 A: C:   (for PB10 - internal programmer)                           |
echo |   INSTALL EMP20 A: C:  (for EMP20 - Parallel port programmer)                     |
echo |   INSTALL SA A: C:     (for SA10/20 - Stand alone programmers)                    |
echo |                                                                                   |
echo |                     (See the READ.ME file for details)                            |
echo -----------------------------------------------------------------------------------
goto end
:EMP20
:SA
:PB10
%3
IF EXIST %3\%1\*.* goto D_EXISTS
echo ***  MAKING DIRECTORY  ***
MD\%1
CD\%1
copy %2\%1\*.* %3\%1
IF NOT EXIST %3\%1\*.* GOTO NO_COPY
ECHO ***  DECOMPRESSING SOFTWARE  ***
if “%1” == “PB10” goto PB10_2
if “%1” == “pB10” goto PB10_2
if “%1” == “Pb10” goto PB10_2
if “%1” == “pb10” goto PB10_2
if “%1” == “emp20” goto EMP20_2
if “%1” == “emP20” goto EMP20_2
if “%1” == “eMp20” goto EMP20_2
if “%1” == “eMP20” goto EMP20_2
if “%1” == “Emp20” goto EMP20_2
if “%1” == “EmP20” goto EMP20_2
if “%1” == “EMp20” goto EMP20_2
if “%1” == “EMP20” goto EMP20_2
if “%1” == “sa” goto SA_2
if “%1” == “sA” goto SA_2
if "%1" == "Sa" goto SA_2
if "%1" == "SA" goto SA_2
:PB10_2
EMP_DIST.EXE
DEL EMP_DIST.EXE
GOTO GOOD_INS
:EMP20_2
EMP20ARC.EXE
DEL EMP20ARC.EXE
GOTO GOOD_INS
:SA_2
GOTO GOOD_INS
:GOOD_INS
copy %2\util\util.*
echo ***  %1 SOFTWARE INSTALLED  ***
GOTO END
:D_EXISTS
echo ERROR: Directory already exists.
goto end
:NO_COPY
echo ERROR: Unable to copy files onto hard drive.
:end
```

7.1.2 DEFINITIONS AND TERMS

Device—Device always refers to the device in the ZIF socket that is being programmed, read, verified, or otherwise manipulated by the EMP20.

Buffer—The buffer is the area of memory (disk and/or RAM) into which a disk file of the contents of a device is read. The Buffer may be from 64k to 16 Megabytes. See EMP20.INI.

Main-menu—The main-menu is the initial menu brought up by EMP20. This is where reading, programming, and most other file and device functions are performed. There are two sub-menus, the part-selection menu, and the Buffer editor.

Part-selection menu—To program, read, or verify a device, it must first be selected. To do this, press '5' at the main menu to enter the part-selection menu. Once there, you may select a manufacturer and then the appropriate part.

Buffer Editor—To view or edit the data in the buffer, press '7' at the main-menu. Once in the editor, press 'F10' for help, or '<ESC>' to exit.

<CR> and <ESC> -- <CR> refers to the Return/Enter key on the keyboard. <ESC> refers to the ESCAPE key located on the upper-left of some keyboards, or on the upper-left of the numeric keypad on others.

7.1.3 SCREEN SIZE

EMP20 supports different screen sizes. For users of EGA and VGA, EMP20 will support any screen size available. To use another screen size, simply use the program normally used to change the screen size, usually a special program that comes with the display adapter. EMP20 will automatically adjust to the new screen size. The higher-density screen sizes are useful when viewing large amounts of data (i.e. the Editor), or for use with Macros and File Definitions.

7.2 TROUBLESHOOTING

On EMP-20 there are not many areas to check. Below is a list of things to look at:

Make sure device selection matches device being programmed.

Make sure family module is correctly oriented and properly seated, and that it's the correct one for device being programmed.

Make sure socket module is correctly oriented and seated properly, and that the ZIF socket lever is down.

Make sure parallel cable is securely attached at programmer and PC.

Make sure power cord is securely attached to programmer and power pack to wall socket.

Make sure power switch is in the ON position.

8.0 WARRANTY AND TECHNICAL SUPPORT

8.1 WARRANTY INFORMATION

EMP-20 is sold on a 30 day satisfaction quarantine. The EMP-20 is further warrantied against failure for one (1) year. Covered is replacement or repair, at our option, at no cost for parts, labor and second day return shipping. Needham's Electronics Inc. is not responsible for incidental damages that may arise out of use of the EMP-20. If a failure is experienced, call Needham's Customer Service Monday through Friday between the hours of 8:00AM and 5:00PM at (916) 924-8037 for a Return Material Authorization. After the warranty period expires, the EMP-20 can be repaired for a flat fee of $35.00. This includes all parts, labor, upgrading to the latest revision, recalibration and second day return shipping within the continental US. Use the same RMA procedure as above. No other warranties apply or are implied.

Needham's Electronics Inc. reserves the right to modify the terms and conditions of the EMP-20 warranty at any time. Failures due to abuse or misuse of the EMP-20 will void the warranty and will only be repaired on a time and material basis, an estimate will be provided and agreed to before any work is started.

8.2 TECHNICAL SUPPORT

When you call Needham's Electronics Inc., a person will answer, we do not use a call forwarding system. The person that answers will most likely be able help you with any questions without transferring your call or putting you on hold. We want you to be totally satisfied with your EMP-20 and the company that makes it. Technical Support and Customer Service is available Monday through Friday between the hours of 8:00AM and 5:00PM by calling (916) 924-8037.

8.3 USEFUL PHONE NUMBERS

NEEDHAM'S ELECTRONICS	(916) 924-8037
NEEDHAM'S BBS	(916) 924-8094
NEEDHAM'S FAX	(916) 924 8065

INTERNET ACCESS TO NEEDHAM'S ELECTRONICS

WWW	URL: http//www.quiknet.com/~needhams
Software updates	URL: ftp://ftp.quiknet.com/pub/users/needhams

INTEL LITERATURE ORDER LINE 1-800-548-4725

Order #	Handbook Title
210830	Memory Handbook
270645	8-bit Embedded Controllers

ATMEL	(408) 441-0311
CYPRESS	(408) 943-2600
DALLAS SEMICONDUCTOR	(214) 450-0400
EPSON	(213) 534-4234
FUJITSU LIMITED	(408) 432-1300
HITACHI	(800) 448-2244
INTEL	(408) 986-8086
MITSUBISHI	(800) 556-1234
MOTOROLA	(800) 311-6456
NATIONAL SEMICONDUCTOR	(800) 538-8510

NEC	(800) 632-4636
SEEQ	(408) 432-7400
SIGNETICS	(408) 991-2000
TEXAS INSTRUMENTS	(800) 232-3200
TOSHIBA AMERICA	(800) 457-7777
WAFERSCALE	(415) 656-5400
XICOR	(408) 432-8888

Source for Socket Adapters not available from Needham's(i.e. DIPtoPLCC, etc..)

CIC	(916) 626-6168
EDI	(702) 735-4997
EMULATION TECHNOLOGY	(408) 982-0660
LOGICAL SYSTEMS CORPORATION	(315) 4780722

Programmable Devices Parts Suppliers

B&G MICRO	800 276-2206
C&P DISTRIBUTING	(219) 256 -1138
DIGIKEY	800 344-3539
KRUGER	800 245-2235
R & D ELECTRONICS	(714) 773-0240

GRAMMAR ENGINE	(614) 471-1113

EPROM Emulators

ENERTEK (215) 362-0966

ASSEMBLERS and CROSS ASSEMBLERS

PROMDISK (619) 7448087

Make PCROMDISK from programmed EPROMs

SMART COMMUNICATIONS (U.K.) 44 081 441 3890

EPROM emulators, cross assemblers, simulators, disassemblers, compilers, in circuit emulators, and socket adapters.