UNDERSTUDY — XP640 Programmer and XM520 Emulator

It's always satisfying to see a unit perform on a real job and even more pleasing when everything shapes up. This programmer and emulator proved themselves invaluble for this engineer's act.

RECENTLY, A PROJECT being designed in the ETI lab required some software development. The software is in machine code form, serving the dual functions of circuit hardware checking and circuit applications. It requires a lot of small, tedious subroutines to be written, burnt into the EPROM and plugged into the target system (the circuit you want to test) and tested. It very soon becomes a never ending battle of burning-erasing and reburning of EPROMs. That might be OK for someone who has mega stock of EPROMS but the time investment is just too much for us.

A quick shop around found us the helpful XP640 EPROM programmer and XM512 ROM emulator from Elmeasco Instruments. The XP640 can be used as a stand alone programmer and is endowed with very powerful programming features. The XM512 Emulator could be viewed as a supporting device to enhance the facilities on the programmer maximally. They are both encased in lightweight aluminium conveniently ready to be carried around.

XP640 EPROM programmer

To start with, this XP640 EPROM programmer can do a lot more than just burn EPROMs. It can also burn and erase the recently introduced EEPROM and automatically check the correct insertion of the EPROM in the socket with 64K bytes of internal memory. The programmer incorporates a LED display for data and commands.

One handy design in this unit enables the user to hook up the video monitor to the programmer, thus allowing a much bigger portion of the memory in the system or multiple commands to be examined on the screen simultaneously.

The entire keyboard can be divided into four sections functionally: data entry, edit, command and cursor control sections. To save space and cost, the editing and data entry keys are multiplexed together by a single function key. If you happen to be a poor engineer who has to key in machine code manually, these keys are a must.

The editing function allows the user to define an address within the 64K for the cursor to move to instantly. The user can define a block of memory of any size and do something to it later. The most common thing is to shift the data in that block of memory to another part of the memory. You can also copy it to another address, invert (complement) the data in a defined block, or fill the entire block with whatever data you like.

Apart from the block operations, the user can move the cursor with the 'up', 'down', 'left' and 'right' arrow keys to the start of data, replace it with something else, or delete it with automatic shifting of the following data to fill the 'hole'.

Having to insert a few bytes in the middle of a program after the entire thing has been entered is a real pain, unfortunately unavoidable. In this programmer, you are catered for. You simply move the cursor to the place where you want to insert and hit the insert key. A 'hole' is automatically generated for you in the memory with the rest of the following data shifted back one location. Hit the insert key again and two holes will be generated and the data you have to insert is simply keyed in.

The ability to handle 16-bit machine programming really puts the unit forward in the market. The 'split' command divides the internal RAM into two blocks as specified by the type of device selected. All data at even addresses is stored in the lower half of the block, and all odd address data is stored in the top half. The effect is that if 16-bit data had been loaded into the RAM (from the serial port), it can be split so that two EPROMs can be programmed: one containing the data at even addresses, the other containing data at odd addresses. The 'shuffle' converses the split to interleave the data in the top half of the block with data in the lower half, that is, a 16-bit to 8-bit shuffle. On top of all these handy features, a specific data byte can be located with the 'search' command. The entire data entry and editing sections are protected while you are away for a coffee break if a 3-digit code has been entered using the 'lock' command. Further data editing can only be resumed if the keyboard is unlocked with that selected 3-digit code.

Command keys section

There are two ports (one serial RS232 and one Centronics) in the unit to allow communications to the external peripherals. The 15 keys in the command section control the ports, the programming sequence, the printing, menu of EPROM, emulation and a lot of checking during data transfer.

The keys that handle the programming sequence enable the user to check if the EPROM is blank, verifying that the correct program is burnt in, and calculating the check sum or a cyclic redundancy check on the entire program. Of all these, I have found the illegal bit check command is most useful. Very often, a corrupted EPROM or an EPROM which has not been erased properly will have some of the bits stay at logic 1. If these bits match with the corresponding machine code you are trying to burn in the EPROM, there is indeed no need to erase the EPROM again.

Ports

The 'port' key used in conjunction with the data entry keys, enables the user to ▶

S. K. Hui

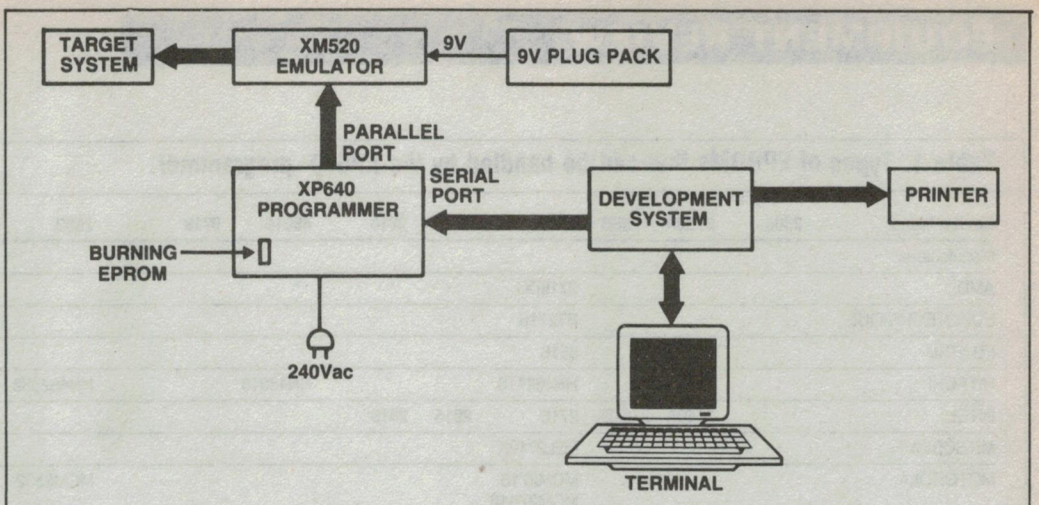

Figure 1. A typical set-up for software development employing the XP640 programmer and XM520 emulator.

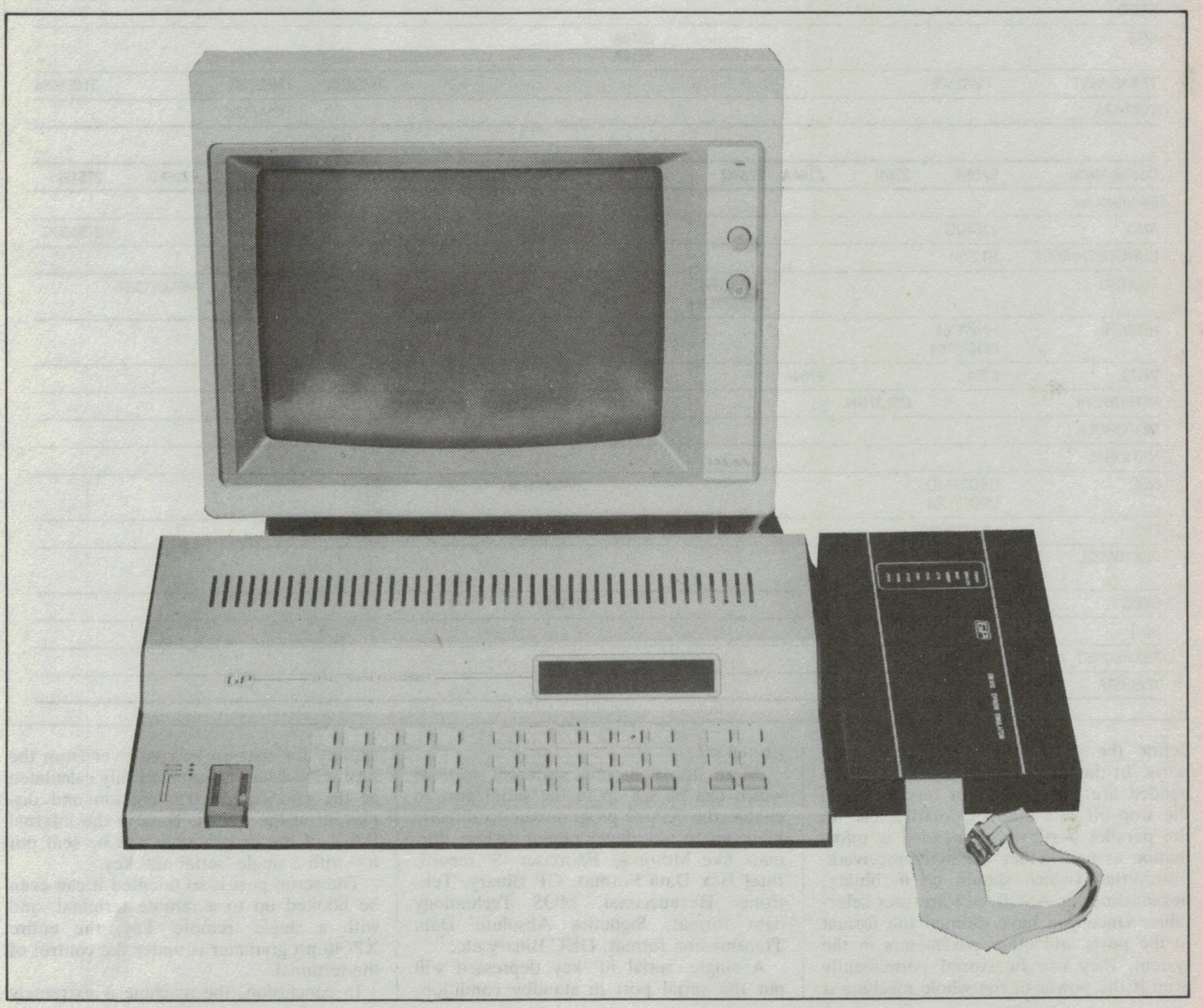

Table 1. Types of EPROMs that can be handled by the XP640 programmer.

Device Menu	2508	2758A	2758B	2716	2815	2816	48016	9716	2532	2732	2732A	2564
Manufacturer												
AMD				2716DC						273DC	2732ADC	
EUROTECHNIQUE				ET2716						ET2732		
FUJITSU				8516						MBM2732	MBM2732A	
HITACHI				HN462716			HN48016		HN482532	HN482732		
INTEL		2758A	2758B	2716	2815	2816						
MITSUBISHI				M5L2716K						M5L2732K		
MOTOROLA				MCM2716 MCM27A16					MCM2532			
NATIONAL				MM2716 NMC27C16		NMC2816		NMC9716	NMC2532	NMC2732 NMC27C32		
NEC				UPD2716D						UPD2732D	UPD2732AD	
OKI				2716							2732A	
ROCKWELL						R5213				R87C32		
SEEQ												
SGS						2816A 5516A						
TEXAS INST	TMS2508								TMS2532	TMS2732		TMS2564
TOSHIBA										TMM2732D		

Device Menu	2764N	2764I	2764A	2764Q	27128N	27128I	27128A	27128Q	27256I	27256Q	27512I
Manufacturer											
AMD	2764DC					27128DC			27256DC		27512DC
EUROTECHNIQUE	ET2764										
FUJITSU				MBM2764 MBM27C64				MBM27128		MBM27C256	
HITACHI	HN27C64 HN482764				HN4827128						
INTEL	2764		2764A			27128			27256		
MITSUBISHI		M5L2764K									
MOTOROLA											
NATIONAL											
NEC	UPD2764D UPD27C64					UPD27128D					
OKI	MSM2764RS										
ROCKWELL	R2764 R87C64										
SEEQ		2764				27128					
SGS	M2764										
TEXAS INST	TMS2764										
TOSHIBA	TMM2764D				TMM27128D						

define the parameters required in both ports. In the RS232 serial port, the things needed are the number of bits for data, the stop bit, the kind of polarity, etc. In the parallel port, what's needed is information as to whether the bytes received/-transmitted (which should be in binary, hexadecimal or ASCII etc) are user selectable. Once you have defined the format in the ports and other parameters in the system, they can be stored permanently even if the power of the whole machine is turned off.

There are numerous different formats which can be set up in the serial port to enable the XP640 programmer to directly hook up to any development system, formats like Motorola Exorciser 'S' record, Intel Hex Data Format, GP Binary, Tektronix Hexadecimal, MOS Technology data format, Signetics Absolute Data Transmission format, DEC Binary etc.

A single 'serial in' key depressed will put the serial port in standby condition, waiting for data to be sent over from the host. Checksum is automatically calculated at the end of the transmission and displayed on the screen. Data in the internal RAM of the programmer can be sent out too with a single 'serial out' key.

The serial port is so talented it can even be hooked up to a remote terminal, and with a single 'remote' key, the entire XP640 programmer is under the control of the terminal.

In conclusion, the machine is extremely

versatile with many little handy features built in. The only less congratulatory comments are for the $1900 price and the 'rigid touch' keys used in the machine.

XM512 RAM/ROM emulator

So far, the programmer has only given programming capability and redundancy built in for emulation. To realise the emulation, another unit is needed. The XM512 emulator will cost you another $600-$700.

The machine has sockets on board for the memory to be easily expanded up to 64K bytes to match the memory size of the mother machine (XP640). The emulator has a CPU on board to mastermind the data transfer from the mother machine, shuffle data during 16-bit emulation, control the display, etc.

There are several arrays of switches on the bottom of the unit for the user to input parameters to the machine. You have to tell the machine the RAM chips on board are 2K byte or 8K byte each, the type of EPROM you are trying to emulate and the mode of operation required. The self-test mode is a rather useful feature in the machine. Anyone buying the unit can turn this mode on to find out instantly whether the machine is functioning or not.

Although the machine had been designed primarily for emulating ROMs, it can also be used to emulate RAM. The major difference between the two is that during the ROM emulation, the target system cannot write anything into the internal RAMs of the emulator, thus avoiding possible program corruption during the software development. The target system is free to write to the emulator during RAM emulation.

There is an optional line connecting to the reset of the target system. A download from the mother machine to the emulator will automatically reset your target system.

Functionally, it is better to divide the emulator and the programmer into two separate units. This arrangement has some advantages such as when the emulator is busy communicating with the target system, the programmer may be talking to a remote terminal, receiving a new program from the development system, or editing the data in its own internal RAM etc. The only drawback of this scheme is the extra bugs for the emulator.

However, with intelligence in the XP640 programmer control software, the 64K internal memory of the programmer could immediately 'emulate' the ROM/RAM in your target system. This architecture immediately precludes the above advantages offered to the scheme where separate units are used for programming and emulating, but does the latter approach justify the extra cost? ●