Microelectron. Reliab. Vol.18, pp.65-76 0026-2714/78/0801-0065 $02.00/0

SCRUMPI 3—A MICROPROCESSOR WITH LOW COST I/O

John H. Miller-Kirkpatrick

Bywood Electronics, 68 Ebberns Road, Hemel Hempstead, Herts

Microprocessor development/training kits are now available in many forms and at many prices. The original manufacturers units such as the Motorola 'Exorciser', National's Low Cost Development system for SC/MP or PACE and Intel's various development kits, are still available but expensive. Even the low cost units intended for the Amateur market can be used as production development units but they are limited by their size and capabilities.

For the small development company or indeed for the amateur, most of the kits mentioned so far require one major item of expenditure which can more than double the total cost of a development system, this item is the Teletype device used for the main Human/MPU/Human communication. Even the 'Glass Teletypes' now beginning to proliferate at lower costs are still not cheap enough in comparison to the cost of the actual MPU kit. For example, it is possible to buy several MPU systems at under £500 each, a new ASR33 teletype or Silent 700 terminal will cost over £1000 and a 'Glass Teletype' will cost about £500. A further limitation which this approach puts on the designer is that he is tempted to end up with an end product which also uses a teletype, thus making his end equipment expensive and probably over-cumbersome.

Bywood Electronics is a company which specialises in the supply of LSI technology components to small volume users. To ease the design of equipment based on these components, a set of simple basic kits was designed for such things as digital clocks, timing and counting circuits. The typical customer was either an amateur constructor or a designer developing a larger piece of equipment who did not have the time or resources to investigate these chips from first principles. The kit allowed him to have the basic clock or counter operational within a couple of hours of receiving the kit; he could thus spend his time in interfacing it to his equipment rather than spending time interfacing it to a power supply and displays.

Enter the dragon.

In this case in the form of National Semiconductors. The release of the NS SC/MP chip was heralded by NS with the introduction of two development kits - the LCDS (Low Cost Development System) at £350 and the Introkit at £60. Bot required a teletype device as the main I/O media although the LCDS also had an optional simple keyboard and LED display media built into it. The SC/MP was designed as a simple replacement for a boxfull of TTL or even mechanical logic in such applications as control systems, amusement arcade games, intelligent data transmission systems, etc. Obviously none of these products are likely to use a teletype as the I/O medium in the finished article and so the use of a teletype in the development of such products is not only expensive but also questionable.

The approach to the designer of this type of equipment assumes that he is capable of designing the necessary software to drive his product. Software programming is a relatively new art practised mainly be people who may or may not have seen the hardware on which the program is to run. The programs are usually large 'file handling' systems such as office routines (Ledgers, Payrolls, Cost Accounting, etc.) or large complicated mathematical models which will repetitively do the same complex calculation until the correct (or best) answer is obtained. The relationship between programming an IBM 370 in PL/1 or COBOL and programming a SC/MP in machine code is roughly like expecting a brain surgeon to be a whizz kid with Meccano!

The designer of what might be called generically 'control systems' is probably most at home with TTL logic gates or he may even long for the days when calculators were more correctly called 'calculating machines'. This man still wants to debug his project with a screwdriver, soldering-iron, oscilloscope and hammer, it is no use expecting him to rush into using an MPU debugging monitor without first explaining how to use it and what tricks he can do with it.

At Bywood we were confronted with the problem of getting our own MPU units up and running simply and at the lowest cost and although we had programmers on the staff, they only had IBM experience. We eventually decided that the best approach was to take an MPU chip and add as few extras as possible onto it in order to get a minimum system running so that the capabilities of MPUs could be investigated. This simple system used LED lamps to show the status of the 8 bit data lines, the 12 bit address lines and four single bit I/O lines; so as not to unduly load these MPU busses we ran the LEDs from CMOS drivers.

One of the main ideas of an MPU is that the instruction set can be

1. Easily modified

2. Executed repetitively at high speed.

For these reasons we had to include some form of memory device. ROMS and PROMs were ignored as these were either difficult or expensive (or both) to generate or modify. Paper or magnetic storage media were rejected on the grounds of speed, interface cost and general complexity. In the end we plumped for MM2112 type TTL compatible Read/Write memory chips.

The MM2112 device is organised as 256 words of 4 bits each, as the MPU requires 8 bits we needed only two memory chips at about £3 each to act as both program and temporary data storage areas for small programs.

The SC/MP MPU can be easily single-stepped, that is it can be persuaded to execute instructions at a very slow speed, manually controlled by the designer at his own speed. This allows simple debugging using the LED lamps plus the designers other tools if necessary - but hopefully not the hammer! At each step the designer can see exactly what effect the previously executed instruction had on the equipment as a whole and if necessary modify that instruction to do something different. This is also the method used to enter the program into the memory; as each instruction is entered on the input switches it can optionally be loaded into memory so that the MPU can eventually execute the program automatically. In the full automatic mode of operation the MPU can perform between 10,000 to 200,000 operations per second where each operation can be an 8 bit logical or arithmetical operation or comparison.

The breakdown of the instruction set of the SC/MP is -

1. Memory reference. Load, Store, AND, OR, EXOR, DEC ADD, HEX ADD, COMP ADD.

   Each instruction has 8 modes giving a total of 64 possibles.

2. INCR/DECREMENT.

   Two instructions with four modes each giving a total of 8 possibles.

3. JUMP, Conditional JUMPs.

   Four jump instructions with four modes each equals 16 possibles.

4. EXTENSION REGISTER (i.e. Accumulator B).

   Eight single byte instructions similar to 1. above.

5. POINTER REGISTER.

   12 instructions allow indexed memory reference or jumps.

6. Others.

   13 other single byte instructions facilitate serial I/O, internal bit shifting, control and flag operations.

From this list it can be seen that the 53 instructions claimed by

National for the SC/MP in fact works out to 122 possibles out of a total of 256 combinations. It is possible to externally extend the instruction set to perform hardware operations customised to particular needs if the 123 instructions are insufficient. At this point it would probably be fair to again liken the SC/MP to a box full of mixed TTL chips; most products which could be built from such a box of TTL can be built using a SC/MP.

SCRUMPI 1 and 2.

After designing and building our own prototyping board for SC/MP as described above, we decided that if a company like ours could use it for both training and development, then so could other people. It was marketed under the name SCRUMPI (as an unforgettable name) and sold exceptionally well. National introduced the lower cost and lower powered NMOS SC/MP2 chip during 1977 and we modified SCRUMPI to use this chip, A RAM and PROM socket was introduced on SCRUMPI 2 late in 1977.

KEYBOARDS and THINGS

One of the first applications for our own SCRUMPI was to make the I/O of instructions and data by more experienced MPU designers a little easier and faster. As mentioned earlier the National Semiconductor LCD unit has an optional simple keyboard and LED digital display, NS also introduced an add-on kit to the Introkit to perform similar tricks. As the MPU only talks in Hexadecimal codes there are only 16 different 'numeric' keys required (0-9 and A-F) plus a few operation keys such as RUN, STEP, RESET, etc. The output from such a unit uses LED digital displays such as might be found in a calculator, in fact NS used an old NOVUS calculator case, keyboard and display for add-on to INTROKIT. The Introkit and KBDKIT were not designed as a pair and were thus not perhaps the best solution to the problem; the concept has now been simplified and has recently been offered on the market. It is, however, basically a software training tool and not a designers development tool.

The Digital LED type of display is limited by the number of digits (usually 6 or 8) and the fact that the seven segment type of display severely limits the number of understandable non-numeric characters which can be displayed. If the end-user is not electronically orientated then character styles which may be legible to you are a foreign language to the user. For example the following messages -

```
UNIT = 4        (UNIT = 4)
RESET           (RESET)
```

are just about comprehensible - but what happens if a segment fails?

What we were after was a low cost, quiet, legible output device capable of displaying textual messages which were not so stylised that a layman would have difficulty in understanding them. The idea of using a video output to a TV monitor or commercial TV set is quite new and as we had already designed and sold several type of VDU character generation systems, we decided to cost out a minimum configuration.

VDU CONCEPTS

This type of VDU (Visual Display Unit) contains several basic units -

1. TV synchronisation signal generation. This part of the circuit generates the line and frame sync signals used to synchronise the 'picture' on the screen. Basically we have to generate a line sync and the necessary blanking signals every 64 μs and the similar frame signals every 20 ms (figures refer to 625 line TVs.)
2. Within the visible 48 μs horizontal time we have to define a number of character slots and similarly define a number of character rows in each visible vertical scan.
3. Within each character slot we have to define a number of horizontal dots and vertical lines, each to include inter-character spaces.
4. Each character slot has to be able to display several (in fact 64) different characters made up from light and dark dots and lines.
5. A memory is required to remember the character required at each of all the possible character locations defined in 2. above. This memory must be accessable by the VDU and by the MPU.

The minimum memory within reason is two MM2112 RAM chips, thus giving a possible total of 256 character slots. We decided to split this up into 8 rows of 32 character slots and to fill the screen with the resultant display, thus the characters would be quite large in both height and width. The resultant display was thus suitable for personal display on a 2" screen such as the Sinclair monitor or for public display on a 26" commercial TV screen and for other applications with 6", 9", 12", etc. screens.

Referring to the VDU circuit diagram, a 7.02 MHz crystal is used as the main timing source from which all sync and character timing signals are derived. Counter C defines the number of dots which make one character width,

the character times are then counted by counters D and E and gated in F and L to give horizontal blanking, sync and counter reset pulses.

Similarly counters M, N and O together with gates P, T and L give vertical blanking, sync and counter resets. These counters reset after 312 vertical lines, i.e. no interlacing has been attempted.

Thus we have five character address lines to give addresses 0-31 and eight row address lines to give rows 0-7, any character can thus be defined as being character number x in row number y or more simply as character address z where z is 0-255. The resultant eight character address lines go into a Tri State 8 bit buffer (G) which is normally enabled so that the counter address lines are allowed to access the RAM. Alternatively the counter buffer is disabled and the MPU address buffer (J) is enabled thus allowing the MPU to access the RAM. At the same time the MPU data bus buffers (R, W) would be enabled thus allowing the MPU to read or write at the selected RAM address, the VDU RAM is thus completely transparent to the MPU - it cannot tell this RAM from any other memory device of a similar type.

The data from the RAM is normally selected by the counters and is brought out to the character generator system. The lower 6 bits go into a DM8678 type of character generator which also includes latches and PISO shift registers in one 16 pin package. The output is a set of logic 1 or 0 'dots' which can be optionally inverted by memory data bit 7 and thus used to produce white characters on a black background or black characters on a white background. This option is individually selectable for each of the 256 character slots. The same option is available as a 'whole screen' option by the INV input which can be controlled by a manual switch or by one of the SC/MP Flag outputs. NB. Double inverting of a white character results in a white character not black.

The generated character signal, the blanking signal and the sync signals are digitally and analogue mixed to give a 1v p-p composite video signal suitable for feeding a video monitor, modified TV set or UHF modulator, and thus end up as a pattern of dots on a TV screen; the pattern of dots is in fact our 256 characters set out as 32 x 8 matrix.

As the characters were to be quite large, it was decided to use a matrix of 7 dots wide by 9 lines high for each character. Each character slot is defined as 9 dots wide (7 + 2 spaces) and 16 lines high (9 + 7 spaces). As each character line is made up from two of our 312 TV scan lines, each character is thus 32 TV lines high, eight rows are thus 8 x 32 (256) TV scan lines high; this is about 90 percent of the total visible picture height.

QWERTY or $\overline{2B}$

The keyboard had to be low cost, simple to assemble and use but still give as many character code inputs as possible. We decided to run the keyboard as a 16 key block plus four 'mode' keys and an Interrupt key. The decoding of the code from the key depression (s) was to be done by the MPU rather than using an external encoder to save on costs, component count and to give maximum flexibility. Thus in the end product the designations of the keys are controlled by the software and can be labelled to user requirements. One example uses the INT key to simulate a Carriage Return/Line Feed function used to indicate the end of a Human to MPU command string or operation. The 16 key block is used to define characters in the 64 character ASCII set and three of the mode keys define which part of the ASCII set the 16 key block defines. In this example it is possible to enter 65 different codes by use of only 20 of the 21 keys available. The keys could be simply re-labelled for control functions which are completely unrelated to the ASCII character set.

WHAT ELSE?

On any MPU system it is useful to have both serial and parallel communication with the outside world. We decided to install a UART to give high speed formatted serial communication to other MPUs or equipment such as printers which might contain a similar UART. The TTY interface is not connected to the UART but is run from software in a manner identical to that used by National in both the Introkit and the LCDS to give external compatability with those units.

Late in 1977, NS introduced a single 40 pin package containing two 8 bit bidirectional I/O ports together with a 128 byte RAM. As 128 bytes is sufficient RAM for a large number of applications, we decided to include one of these devices and to use the 128 bytes as the system working storage RAM, thus releasing the main RAM (if used) totally for program or data storage.

In its minimum configuration the SC/MP has only 12 address pins and without an additional latch to get at the other 4 address pins is thus only capable of driving up to 4096 bytes of memory or devices. At this stage, our design had already used up 1024 bytes with the VDU, Keyboard, RAM/PORTS and UART. Thus we had the option of limiting our design and adding only 3k of devices, or expanding the address bus and allowing another 63k bytes of expansion; we opted for the former on the basis that it would cover most simple applications and that the expansion could be carried onto another board up to the maximum 64k by the user if so required.

Of our 3k possible memory locations, we had to have 1k dedicated to some form of monitor or control program and to give maximum flexibility we decided to offer 1k more of PROM and 1k of RAM as user options and so we did the simplest thing, which was to design-in sockets for optional user expansion. The block diagram shows the MPU and all of the related units including the device selection logic which in fact comprises several ICs of different types scattered liberally around the system.

The Question Everybody loves to ask.

'What do I need an MPU for? What can I do with it?' The answer to these questions is, 'just about anything'. The problem which the designer may have is that if he knows little or nothing about MPUs he may not be able to rethink his end product in terms of MPUs. The concept of using an MPU in a piece of equipment is similar to that of using a calculator chip in a calculating machine or a clock chip in a clock. The input, output and instruction entry methods are so radically different from what is considered normal, that it can be difficult to ignore several years of mechanical or digital product growth and redesigns and think in terms of a complete redesign of the product from first principles. I can imagine clock designers not being able to understand how the LSI clock chip is capable of driving the hands of a clock; designing a clock with about 12 hands capable of showing time and date from month to 1/100ths of seconds would be inconceivable and ignored on the basis that it would never sell. You can now buy watches with these features in your local jewellers at under £50, not with hands but with a digital display. This type of radical, back to first principles, rethink is what is needed when considering using an MPU in a new version of an existing product.

Let us assume that the existing product is required to accept information in digital or analogue form from say 13 input lines and is required to control devices via 5 output lines (or vice-versa). Data must also be periodically entered from BCD switches or a keyboard and messages or figures displayed on an operators panel. Examples: simple petrol forecourt system, pressure/temperature control of machinery, production line monitoring, complex timing control system, Amusement and/or vending machines, pill or component counter/sorter, etc.

Let us also assume that any complex or difficult environmental interfaces already exist in the present equipment and are already at or could easily be brought down to 5v logic levels. The MPU equipment can easily interface to

such equipment via serial UARTs or parallel PORTs and can thus evaluate the information being presented at the inputs or control equipment via TTL compatible outputs.

All that is required now is the appropriate MPU hardware and some software to run it. The former is the unit we have now designed and called SCRUMPI 3, the latter is something which is probably unique to the end product.

SOFTWARE HURTS

The old definition of hardware and software was that hardware hurts when you kick it, referring of course to the large IBM type of hardware. In todays definition it could be said that hardware hurts the foot and software hurts the pocket; software can be very expensive to design yourself or to get designed by an outside agency. IN SCRUMPI 3 half of the software is already designed to run the keyboard and VDU, etc. The half of the software required to run your product can be very simple to write and debug as it can be done in situ on the same SCRUMPI 3 which will eventually become the final prototype. Alternatively, our consultancy division have experience with SCRUMPI 3 software and may well have recently written software for a similar SCRUMPI 3 application and therefore be able to write software to your specifications quickly and at reasonably low cost.

Let us take an example of a product called a 'SUPATIMA' which is a complex timing apparatus capable of controlling up to 8 outputs at preselected times during a year to the second. Up to 25 alarm times can be entered for operation per minute, per hour, per day, per month or per year, manual and external over-ride facilities are included and the alarm times can be easily changed at any time. Basically the SCRUMPI 3 is used to extend the facilities of a time and date alarm clock chip which is interfaced to SCRUMPI 3 via one of the two eight bit ports. If your product was similar but required 150 alarm times per day to minute, rather than seconds accuracy, then the software for such a product could be based on the software used in the first product and may thus only take a week or so to write at a cost of under £1000. Here we can see one of the other main advantages of using MPU based systems such as SCRUMPI 3, it is feasible that the prototype could be available within two weeks of finallising the specification. In the case of urgent one-off specials, this sort of time scale could make the difference between your getting the order or it going to a competitor who is already using MPUs.

SCRUMPI 3 is ideal for the small volume user as the expensive and time

consuming hardware development is already 90 percent completed (including suitable casing and power supplies, etc.). If we assume that the software development costs are under £2000 for the average product and we amortise this over 50 units per year then each cased product would cost under £200 to manufacture; in most cases this would represent about 10 percent of the present costs of such equipment.

SO HOW DO I START

If you feel like writing your own software, then the SCRUMPI 3 development system includes a simple but effective DEBUG monitor system in two PROMS. One PROM contains the software routines for controlling the keyboard and VDU and printing numbers and/or text on the VDU screen; this PROM can be used in your final product. The second PROM contains routines for modifying memory (eg for entering programs or data), displaying memory in several ways, executing main or sub-routines written by the user and reporting on the results of that program execution. After development, your programs can be dumped into PROMs so that the programs will be retained in case of a power supply failure. The total cost of such a development system is about £160 for SCRUMPI 3, £70 for a 12" television and a day with a soldering iron,(note how this compares to the £1500 or more required for those systems mentioned in the opening paragraph).

If you have no desire to build your own SCRUMPI 3 or design your own software, you should analyse your current product down into interface hardware and control hardware and then work out what you would hope or expect to see in the proposed product in the way of features and facilities. You should then contact a Microprocessor consultant (preferably ourselves) and arrange to buy a day's consultancy to discuss your proposals and do a simple costing and feasibility exercise. For the sake of £100 or so spent at this stage you could save yourself a lot of money and heartache later. You know your product and the consultant knows his, between the two of you, you could revolutionise your product area.

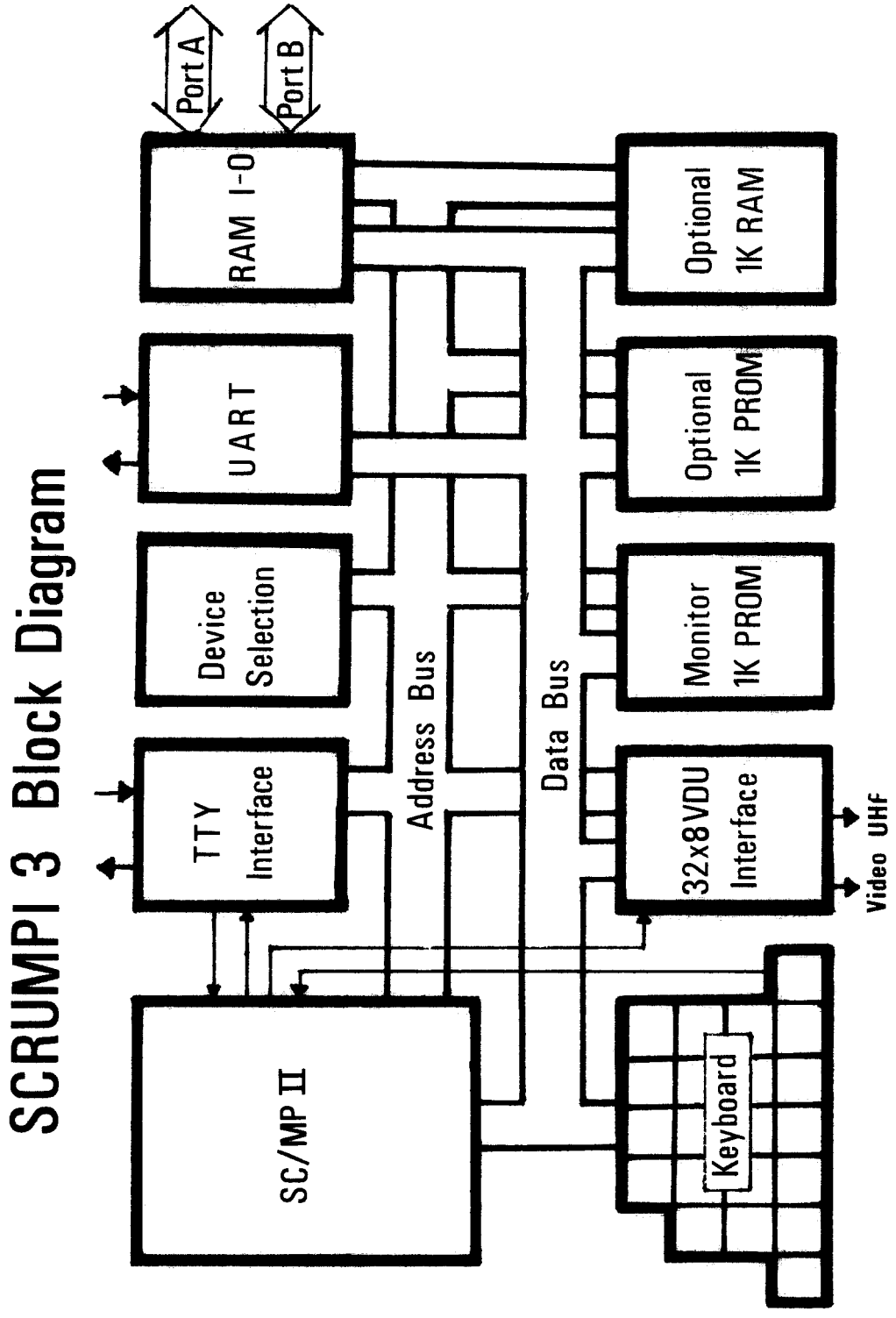
PERSONAL COMPUTING ASPECTS

The SCRUMPI 3 type of product does not talk BASIC (yet!) and is not therefore suitable for the computer programmer who wants to keep his mother's recipes on file for instant access. SCRUMPI 2 and 3 are designed for the TTL or mechanical logic fan who wishes to investigate the applications of microprocessors in his hobby or business. In this type of application SCRUMPI 2 or 3 or both together will allow the TTL fanatic to build up his latest project without the necessity of caniballising the previous project for bits. Most of the bits will

be common to both projects and will be in the form of a SCRUMPI 2 or 3 as the main PCB with the extras built onto Veroboard and plugged into a PORT, UART or MPU busses. As a logic fanatic myself, I can assure you that this is a very effective and efficient way of trying out a new project. The use of a SCRUMPI 2 to assist in the debugging of a SCRUMPI 3 project means that the single step and LED lamp facilities can be added to the keyboard and VDU facilities of SCRUMPI 3 to give one very powerful MPU development or training tool at a ridiculously low cost.

STOP ME AND BUY ONE

Whether we like it or not, microprocessors are going to enter our everyday life in the same way that transistors have done. Just as products such as the CBM PET computer are leaders in the use of microprocessors in home and business information systems, products such as SCRUMPI are leaders in the home and industrial control systems. When Mr. Jones from next door invites you round for a quick drink and game of Star Trek, you can counter by being late because you had to program your baby sitter!


SCRUMPI 3 Block Diagram
Port A
Port B
RAM I-O
UART
Device Selection
TTY Interface
SC/MP II
Address Bus
Data Bus
Optional 1K RAM
Optional 1K PROM
Monitor 1K PROM
32x8 VDU Interface
Video UHF
Keyboard