4

C3PO

VOL 5 NO 5 JUNE 1982

MB

SPECIAL EDITION

Contents

ΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩ

VOL 5 NO 5 JUNE 1982

ΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩ

ΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩΩ

We extend our thanks to
DATABASE and DATA AGE
for their assistance in the
production of this issue of C3PO.

Editorial

DONALD COOK

The following EDITORIAL was first published in our bumper (MICROBITION 81) edition of C3PO in June last year ...

The Cape Computer Club is just three years old. It all started in May 1978 when a small group of interested amatures who were frustrated by the lack of any commercial interest in micros in the Western Cape planned a get-together at the house of Elliot Sacher. At least Elliot thought it would be a small group. But word got around, and such was the underground inerest in micros that over sixty people turned up.

At the first meeting a working committee was appointed to draw up a constitution, fix and collect dues, organise formal monthly meetings and publish amonthly newsletter. This was all done in a very short time and Elloit Sacher was formally elected as the first chairman of the newly constituted Cape Computer Club.

The first few meetings were held in premisis in Sea Point but later the venue was moved to a school which was more central and offered abundant parking.

The first project tackled by the C.C.C. was a glass teletype, and very successful it was. Many are still up and running without modification to this day. The glass teletype board naturally led to a long, and often heated,discussion about a suitable bus organisation. About the only fact that everyone agreed to was that it should be a standard used by all club members. Eventually the C3 bus was proposed and accepted. This bus is thought to have been the first standard bus organisation in South Africa and was published overseas as well.

About his time various interest groups started forming in the club, particularly the 6800 group and the Z80 group (which now includes users of all 8080 compatable micros).

The 6800 group has designed a complete micro system from scratch including a single board computer, dynamic and static memory boards, input/outpt board and floppy disc controller board among others. As large and sophisticated monitor was designed andwritten by the group for use with the system. These designs are available to Club members. All 6800 boards have been designed for the C3 bus.

The Z80 group has made extensive use of commercial boards based on the SA bus. However, here too, a number of boards have been designed as the commercial boards were unsuitable or unobtainable. This group has been able to digress into morer esoteric areas. For example, a number of interesting video graphic systems are being developed.

While initially most interest was in hardware, there has been a gradual swing towards software as more and more members have up and running systems. The Club recognised and encouraged this by organising a series of talks on Software Design. These proved very popular with members.

Recently an Apple users group and a TRS 80 users group have become active with much exchange of tips and software among the group members.

The Club newsletter has grown into a substantial monthly magazine with editorial, articles, regular columns, a monthly competition and more. The professional publications regularly reprint original articles from C3PO.

The venue for monthly meetings has changed again recently and meetings are now held on the first Thursday of every month in the Community Centre, Rondebosch. A major undertaking at present is the serch for a suitable permanent premises for the club to house the club library, a number of microcomputers the club hopes to obtain, a small workshop for the use of club members and a general meeting place where people with like interests can relax together.

The Cape Computer Club is just three years old, but in that time this group of keen and dedicated persons has achieved a vast amount and created an organisation of which all the members can justifiably be proud.

COLIN RUDOLPH

... Well, some important developments have occured since that was written the most significant falling under the following headlines:-

CLUB MAKES FREINDS WITH MRS VONK.
(Of Atheneaum fame)

DYNAMIC NEW GROUP FORMS.
(yet another tree?)

'NOG GROUP FORMED
(At last someone to use the trees)

NOT 'NOG GROUP?
(Anyone got a key board?)

With out doubt our most notable achievement was making friends with Mrs Vonk, the caretaker at the Atheneaum. Have you tried getting your hands on a homemade biscuit after the rush? We certainly have an excellent meeting place with just the right atmosphere. I'm sure you all join me in thanking Mrs Vonk for the way in which she helps us.

Following the examples of the 680X and Z80 groups the special interest groups that cater for the commercial machines have grown like mushrooms.

These started with the APPLE users group and a quick glance at APPLE TURNOVER reveals that this is still an enthusiastic group. Anthony is untiring and industrious; his ingenuity never fails to amaze onlookers. (Hey! thats my RAM chip!!) He has implemented such unlikely things as scopes and TV controllers, not to mention disk drives on his Apple.

Peter Reber is responsible for the fast growing and active Acorn group that is paving the way for the BBC micro that is to hit the scene shortly. Peter not only has expertise with a soldering iron (he has made a number of mods to his Acorn), he has given some very informative software talks to the software group, not to mention having his work published in a popular overseas magazine. The Acorn fans are lucky to have a person of this calibre to guide their efforts.

Earlier this year the Commodore Operators Group was formed. Yes, we have our very own COG. Ephy Chesler has proved a very capable organiser and with his right hand man, Geoff Sturges, nothing can go #$%!"#!!... Geoff has a delightful sense of humour and is one of those lucky souls with the ability to make the most complicated concept sound simple.

Our most recent addition to our ranks is the ZX80/1 group - This could be our biggest and most active group. So watch, you oldies. The low cost and availability of this machine will obviously attract the younger enthusiast. Roger van Rensburg and David Long are in the driving seat and things look very promising. (look for the 1st ZX80 report in this issue.)

THE FUTURE

It seems that now most of the homebrew hackers have got their machines running, the trend in the club is more towards software, its design and implementation. This will nodoubt be the trend for a long time to come. I wonder if the next developement will be that of software SIGs?

We will shortly celebrate our 4th birthday. Looking back over this year we can be justly proud; the club has increased it's membership by over 50% which is good by any standards.

See you at MICROBITION.

This Month's Meeting

THURSDAY, 03 JUNE 1982

THE ATHENEAUM, NEWLANDS

19h30	SOFTWARE CLASS Donald Cook
20h00	CLUB BUSINESS
20h15	THE BBC MICRO Peter Reber Peter Lawson
21h30	TEA AND BISCUITS
	20c for non-members

SUBS

MONEY, MONEY, MONEY
CASH, CASH, CASH

Press reset and start again.

"This is your Treasurer speaking. It hardly seems possible that another year has gone by and in next month's C3PO you will all be advised of the A.G.M. Financially speaking our year ends on the 31st May, and very recently joined members who have just paid a full years subscription can relax, this is not aimed at you. You have paid in full and your membership takes you through to 31st May 1983. Then there is a small band of you who joined about 4 or 5 months ago and were only required to pay a half subscription, to you and all the long standing members I must say, without apology, that from the 1st June onwards subscriptions again fall due.

The rates are:

Fullmember	R15.00
Country member	R 8.00
Scholar	R 8.00

I hope I shall not be accused, like dentists of yesteryear of making painful extractions; it is up to you to put on a brave face, open your purse or cheque book and pay up so that I can shut up. You can send your sub, together with your name, latest address and telephone number to our Box 6251, Roggebaai, 8012 when a receipt will be posted to you, or pay it to me at our meeting when a receipt will be handed to you".

The Program has been written - very basic, just press "enter and return".

Peter Llewellyn.

C3PO Scratchpad

GEOFF STURGES

The last Club Meeting will not be forgotten for a long time by those who were there to witness the pyrotechnics of the Apple Ayotollah. Signature Analysis may still be greek to some of us but "frying chips" has taken on a new meaning.

Did you see our resident T.V. personality on the box the other night. Those ties were really TRS 80 Stringy Floppies in disguise!

Only a few more weeks to go and it is "Microbition 82". Is your machine ready? Are you supporting your Group? Don't let us have the same old faces; new blood is always needed and most welcome. See Anthony or Peter Reber before it is too late.

All the stands have been sold so our stand has to be outside the main hall, in the foyer. We are going to a lot of trouble to have a first class stand so all aspects of the Club's activities need to be represented.

On the first day, Friday June 18th, the doors open at 10h00, and at 18h00 the official opening by a guest speaker takes place, so do try to be there, with your Computer Widow if you have one. Don't forget to show your Membership Card if you wish to get in free.

I have never understood why official openings do not coincide with actual openings (was Hitler guilty of coincide?).

Later that evening, at 20h00 when the doors close, "refreshments" will be served to Club Members and Exhibitors (not "flashers") so prepare yourselves with "Guronsan" beforehand.

On Saturday the 19th the opening and closing times are the same (but unfortunately not another party). We would ask you (those who read this rubbish that is) to volunteer to act as guides, marshals, interpreters (not BASIC), etc. on both Friday and Saturday. We also need help for preparation on Thursday and for cleaning up on Sunday.

Another important date is July 1st when the A.G.M. is to be held. Again we will have a Cheese and Wine sans Cheese type of affair afterwards except for the Apple Group who only qualify for fruit juice! The cost will be paid for with money we haven't been able to swindle out of Club funds!

It is rumoured that after the A.G.M. proper Messrs Tedelex (distributors of the best range of micros on the market according to an independent stooge) will present a VIC 20 microcomputer to the winner of their contest at "Microbition 82", and another machine to the Club for use by Members (bags first go). Now's your chance to learn PET BASIC.

I hear that the Club library has been donated the following books:-

The SABUS primer
by Reed and Wright

Non-Standard Peripherals
by I.O.Port

Dynamic Memory Limitations
by Ramsbottom and Topp

CMOS Technology
by I.C.Pinn

May I thank the donors on behalf of the Club. Other books are always most welcome.

The most worstest comment you'll hear this month is that my puns come from a Cornucopia! How else can I fill this page each month. I tell you its tough at the bottom, in fact as tough as a T-bone steak from Mike's Kitchen!

See you at the next meeting, and, for the last time, please support "Microbition 82".

The following was first published in VOL 3 No.9

The Impact of the Microprocessor

COLIN RUDOLPH

To appreciate the full scale of the impact of microcomputers and minicomputers and to understand what these terms mean it is necessary to briefly look at the development of electronic digital computers from the mid 1940s onwards. Indeed, electronic computers themselves must be viewed against the long history of man's attempts to calculate by machine.

It is appropriate to start with Charles Babbage (1792 - 1871), the father of digital computing. His Analytical Engine was conceived as a mechanical device and employed decimal arithmetic. It was beyond the resources of the age but included most of the basic principles used in modern machines.

There were three motivations for his work, apart from his own interest. These are still the motivations for computer development today. They comprise commercial applications, scientific computation and the stimulus of defense projects.

In the latter part of the 19th century the electric telegraph permitted both domestic and business communication efficiently and comparatively cheaply. Mechanical power was abundant with coal as well as water the motive force. The age of the entrepreneur had dawned. The Victorian age, though many people experienced great poverty, was an age of conspicuous spending both on what are now called durables and on inessentials. Both were made possible by a cheapening of the manufacturing processes and advances in technology due to mechanisation. Even the lot of artisans improved through enhanced sanitation, street lighting and the like.

The worst social effects of the Industrial Revolution occurred in the textile industry where there was a rapid transition from small labour intensive activity, typically in the home, to large scale mechanisation in the factory. Only a small number of operatives could handle weaving machines for which patterns were pre-programmed on Jacquard cards. The scale of industrialisation in the textile industry was great.

Through the Victorian age and indeed well into the 20th century, only limited use was made of Babbage's pioneering work. Very simple data processing - that required for census work - was first carried out by Herman Hollerith in the U.S. Census of 1890. Small scale technical calculations were performed using mechanical desk calculators.

The advent of World War 2 gave the impetus to the development of the electronic computer. Probably the first digital electronic computer was built at the British Post Office Research Station to speed up the decyphering of German signals. For security reasons its existence was made known only thirty years after the war and even today very few details are known about it.

Most texts place ENIAC as the first true electronic digital calculating machine. Built at the University of Pennsylvania, it was designed specifically for ballistic calculations. Some of the vital statistics are:

it occupied a room 12m x 6m
it contained about 18000 valves
it consumed 150kW of power
it operated on numbers of 10 decimal digits
it could do 5000 additions per second
it could store 20 numbers for immediate recall

By the mid 1950's computers were becoming well established - at least at universities. I would like to quote from the Preface to a book "Faster than Thought" published in 1953:
"A rough count showed that about 150 digital computers are being built at this moment, most of them in universities and other research establishments. It will be interesting to see if these machines play in the next decade the part of cyclotrons and high voltage generators in the thirties. In those days every university had to have a cyclotron on the campus; they were mysterious and expensive and they gave tone to the place; they impressed distinguished visitors and attracted endowments; their construction gave the whole of the Physics Department plenty of healthy exercise, and kept them happy, out of mischief and covered in oil; the united efforts of the staff were required to keep the machine on the verge of

operation, and those who so wished could postpone into the indefinite future the embarrassing decision as to what was to be done with the machines when they actually started working."

These first generation machines were, by modern standards, of limited power, expensive and unreliable. They made huge demands on programmers since programs were written exclusively in what are now called low-level languages. Still they generated popular interest and invoked earnest discussions about the relationships between computer and brain, the nature of free will, "could computers think", and so on. People experienced a reflection, though only faint, of the great religion versus science controversies of the previous century centred on Darwin's "Origin of the Species". The automation revolution seemed just around the corner. But the revolution never did quite come off. Hindsight clarifies some of the reasons.

These computers incorporated many of the features of modern machines. However high level languages were unavailable. In certain quarters they were even regarded as "unsporting". As a result programmers were specialist personnel requiring long and expensive training and individual programs were not written lightly. Programs took a long time to write and as a result were expensive to produce. Peripheral interface presented problems making input and output a major task.

Thus the introduction of a computer system to handle commercial data processing or in process control was a decision not to be taken lightly. Machines were expensive, staff training was expensive and extensive programming time was required. All these factors resulted in the immediate revolution not occurring.

Rather a gradual development took place. A distinction appeared between main-frame and minicomputers. The former developed in general purpose machines with sophisticated operating systems and using high level languages. The minicomputer, on the other hand, adopted the role as process controller. Low level languages were retained for efficiency, though this required expert programmers.

Since then minicomputers have become increasingly powerful, incorporating high level languages and more comprehensive operating systems. Today's minicomputers look like yesterday's main-frames, but with reduced physical size and cost. Though traditionally thought of in terms of scientific computing, often in the process control field, it is used increasingly in banking transactions, wages calculations, in airline and theatre booking systems, and it even finds a place in the classroom.

Here one important aspect must not be overlooked. This is the degree to which young people take the computer for granted. Even primary school children will not flinch from simple programs. For example, a series of books for the youngest readers (the Ladybird Book series) devotes a volume to computers. The importance of this aspect should not be minimised. Increased knowledge brings increased confidence and a reduction in fear of the machine. So we must be clear of what the microprocessor really offers - or fails to offer.

The microprocessor is a new device providing substantial computing power on a single chip of miniature proportions, is readily available and at low cost. The performance of a wide range of existing devices will be enhanced through incorporating these chips. In industry and in business, microprocessors will make possible the totally automated factory or office at an early date. As a result many people will lose their jobs and a general reduction in working hours will make for substantial problems as to how to employ the new-found leisure.

This is the myth projected by the media, intent on a spectacular approach.

What is the reality? At present it is true to say that the microprocessor forms a component of the microcomputer; that the microcomputer is part of the overall spectrum of computing power; that the microcomputer costs not more than a few hundred rand; that for the same computing power it appears to be cheaper than a comparable minicomputer.

However hardware cost is only the "tip of the iceberg" as so many have come to appreciate - somtimes the hard way. An example may make this clearer.

The microprocessor is often held out as the device which will make for safer and more efficient motoring. We have a massive market. Why not incorporate a microprocessor based instrument system? Work in these areas is in progress but the problem is not so easy of solution.

Suppose we wish to display fuel consumption per distance. A little reflection will reveal that a flowmeter of requisite performance and appropriate price and which incorporates digital readout suitable for interface to a microprocessor involves formidable problems. Further environmental and interface problems peculiar to the motor car must be considered and the solution at an economic price becomes difficult. Such a situation is repeated with each parameter to be monitored. Production costs, even spread over a considerable volume, will be far more trivial.

Apart from hardware, a robust software suite must be provided. This too is a formidable task. From the above it will appear that unit costs will be high even if the microprocessor was free.

Schemes for maintenance must be set up; training arrangments for staff made. With all these factors, it is perhaps not surprising that the impact of the microprocessor in the motoring field has to be minimal.

This example is not intended to belittle the role that the microprocessor will play in industry in general. The purpose of the example is to show that an instant and low-cost impact cannot be expected.

Where are the contributions likely to occur? Process control has always been a typical application forr minicomputers. Considerable economies have resulted in, for example, the chemical industry. With the microprocessor further economies will accrue. Certain areas where a minicomputer was not economic are open to exploitation. For example it is possible to use microprocessors to control traffic intersections in an urban network with only limited data transfer to and from a central processor. Such a procedure would not be viable if a minicomputer had to be provided at each intersection.

There is also a possibility of multiprocessor configurations for higher reliability. It has been shown that onboard computer systems for aircraft can be made economically with a reliability ten times that of the aircraft structure itself.

Until recently the software requirements of the microcomputer have been supported at a low level only. This has led to expensive software development and applications programs are not lightly written. Only if software is to be widely used are unit costs reduced to mangeable proportions.

The realistic image shows many similarities to the initial computer revolution of the 1950s and 1960s. This was a major revolution with substantial long term impact on the lives of every one of us. But it was a gradual - and therefore relatively painless - one.

The microprocessor has been with us for several years already. Much of the automation revolution expected of microprocessors has already been achieved with minicomputers. Contrary to popular belief the real costs associated with microprocessors are not negligable. The use of high level languages has already begun. This lead to wider exploitation of the minicomputer and it will no doubt be true of microprocessors as well. The fear that the microprocessor will lead to loss of jobs on a large scale is unfounded in my view.

There is the possibility of many entirely new developments. Improved data transmission via domestic TV (Prestel for example) is a possibility. While some will be as ephemeral as TV games, others will doubtless be more valuable and may indeed enhance "the quality of life". Computer aided instruction is an example.

The analogy with the Victorian era should not be forgotten. The first Industrial Revolution enhanced production, lowered unit costs and increased prosperity (though unevenly) and as a result new markets opened up.

This article appeared in VOL 2 : NO 5 (JUNE 1979) when people with systems were realising their software needs.

SOFTWARE ENGINEERING

Neil Walsh

The discipline is generally a new and blooming field in electronics. Software engineering was previously restricted to the computer industry but is now emerging into other branches in electronics particularly where automatic control is required.

We are all aware of some of the effects microprocessor technology is having on every day life. These new versatile and complex digital integrated micro-electronic circuits when put to task, often replace hundreds of "discrete" logic IC's. Hardware development costs are now drastically reduced by using standardised printed circuit boards and by providing customised software or programs.The instruction codes are then masked or burned into permanent read only memory components, and then plugged into the PC board.

While software engineering is closely allied to computer programming, it has some distinct difference. Because of low component costs, microprocessors are being used in applications very alien to normal computer operations. Programming is a user function while software engineering is required to make a system usable. Without on board operations control software known as "firmware", a system is dead and incapable of performing any task. However with a control program ready waiting and instantly availible after switch on, the system will be ready to receive commands from its user. These commands may be simple requirements like reading a magnetic tape to fetch a program into memory and then jump to the loaded program. In a dedicated application where the machine's capability is restricted to a certain task, the system may just wait for simple parameters like the date and time.

The nature of the firmware provided by the original engineer will definitely dictate the expandabiity and capability of the system. If the system was provided with external data and address bus lines,the user can re-engineer his system by replacing the original ROM with his own firmware and possibly upgrade the machine's operating performance.

Software engineering tasks could also directly replace hardware circuitry functions. A good example would be with dual ramp analogue to digital conversion under microprocessor control. For instance, the program would use an input-output port to perform the following operation. It would first switch on the ramp control and then with the use of a software loop, time the ramp up duration and when this expires switch over the ramp control to begin its integrating count. This would be achieved by counting the number of discrete software loops allowed until the comparator output changes. At this stage the conversion will be complete and the processor under program control would have to make any count adjustments necessary. Thereafter a binary-BCD conversion may be necessary. The main advantage of this technique is that only a few input and output lines are required, there are are no external clocks (apart from the MPU clock), the number of bits of the conversion is easily altered by the

engineer, the ramp up and integrating period can be altered to user's requirement, and finally there is greater hardware simplicity. Single chip dual ramp controllers are easily available.
Less conventional techniques are also finding their way into the new software field. Another good example of this is theinterfacing unusual devices to control systems. There are numerous new and easily available 24 hour clock-calendar chips but few that give out BCD codes. The MPU under program control can be made to read in the seven segment codes and to store the pattern for each digit and thereafter to look up the patterns on a stored code table.

To make modern computer terminals more sophisticated manufacturers would normally have needed to make the internal logic far more complicated. With the new technology available they now make better terminals with less components using simple circuitry.

These dedicated microprocessors, under control of their own firmware, scan keyboard switches looking for closure or abnormal conditions. Once a correct closure is detected the key code is looked up ona pattern table stored in firmware form and then output it in the manner determined by its designers. Because CRT scanning is a bit too fast for present microprocessors special complex IC's have been produced that actually handle the formating of the sync pulses and character refreshing. These can be supervised by the dedicated MPU. These controllers are programmable and the engineer designing the terminal can predetermine the lengths of the vertical and horizontal pulses together with the margin sizes and obviously the number of characters per line and the number of lines. It is the task of the software engineer to develop the required firmware to perform this duty. The advantage of being able to adjust the video signals is because all monitors are not all the same.

In low cost microcomputer systems the MPU can be programmed to alternate between being a terminal supervisor and the main microcomputer processor. To achieve versatility these systems can be designed so that the supervisor actually echoes the data to a (serial) port and then returns from the interrupt service routine and then the host or monitor program then reads it back in a conventional mode. In this case a normal terminal can be inserted if needed. However the software engineers may decide to use a completely unconventional technique and simply use unusual input/output subroutines. Often the hardware is oversimplified and puts a definite restriction on upgradability.

It is often difficult to understand why microprocessor controlled systems are so expensive, especially when one looks inside the system. This is because software engineering results are invisible to the eye. One often expects things that can't be seen, like fresh air, to cost nothing. It may have taken a team of software engineers six months to develop. A ten thousand Rand cost can easily be written off if one sells ten thousand units. However this is seldom the case. One must be constantly aware that some tasks can sometimes be cheaper to perform with hardware. Therefore the software engineermust be completely conversant with hardwired logic and not be too proud to replace software with a few gates or simple counters when advantageous.

The following was first published in VOL 4 No.2

Do you need it or do you want it?

ALAN DAY

Why do people spend up to three grand on a home computer? Is it because they are swept up by full-colour, real time space games? Do you think you will ever tire of TV games like "Space Invaders"...? No!? Go and read Farmers' Weekly.

Playing Star-Trek is great fun, the first few times, but I have found that computer games (with the exception of chess) pall after an amazingly short period of time.

This is largely an article of questions like, "Do you know why you bought/built your home computer?" and " What do you want to DO with a microprocessor?". Having asked many people these questions, I'd say that more than half of them don't really know!

Personally, I built my micro just to acquaint myself with something new that was creeping into my field (at about light speed), but had no idea why I wanted to build a computer. "It can be adapted as my needs change," thought I (convenient excuse).

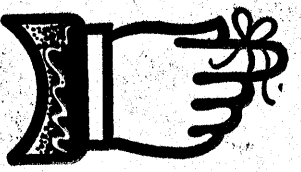
Ah! Now there's the rub ... the needs. If you want a home computer, probably nothing will stop you getting one. However, before you flatten your bank balance, look closely at those needs, not the wants ... the needs.

Look at what you're doing and think about using a computer to do it. (Not too hard if you want to buy one, eh?) Now think about using a microprocessor instead. (That'll work as well, mmm?) And there is still likely to be some cheaper and easier way. What's that? The computer can do other things as well?

Granted, it would be great to have your computer make tea and toast for breakfast, wake you up in time to consume it and get to work, after have sufficed as a burglar alarm during the night, as well as monitoring the water level in your swimming pool, the dryness of the garden, the time your teenagers get home, rhubarb, etc. Now try and play Star Trek on what's left.

Remember, big organisations don't just use one computer to do everything. There's a good reason, it's messy. Several dedicated units of smaller size are always more efficient.

So, buy a digital clock to wake your wife up, she can make the tea (Only kidding, ladies!) and build your little warning circuits from electronics magazines - they cost next to nothing. And save your computer for playing Space Invaders, until you think of a use for it.



18 AND 19 JUNE

DROMEDARIS HALL

GOOD HOPE CENTRE

The following was first published in VOL 4 No.6

Floppy Disk Sector Sequencing

PETER LLEWELLYN

In order to get my first floppy disk drive going a few months ago it was necessary to start off with a formatted disk. These do not appear to be available in Cape Town but I was able to prevail on Ian McQueen to format a disk for me on his TRS-80. This was in the form of 10 sectors of 256 bytes per track, to the IBM format. This enabled me to get going and I was soon able to format my own disks to either this format or the popular alternative of 16 sectors of 128 bytes. Nominally the former layout will contain 2,5k bytes per track and the latter only 2k bytes. However if the last byte of a program goes into a new sector the remaining 255/127 bytes will not be used, so the choice is yours.

One of the first things I noticed when I had formatted my own disk was that the TRS-80 format enabled a much quicker read or write, for the same length of program, than my formatting. Why? In using Western Digital's 1771 controller chip, there is an instruction "Read Track" which enables one to read everything on a track from index pulse to index pulse, both the written data and the inter-sector coding. Amongst other things this inter-sector coding gives the track and sector numbers. From this it was clear that the TRS-80 format did not have the sectors running sequentially.

TABLE 1

	SECTOR NUMBERING SEQUENCE																ALGORITHM FOR SECTOR COUNT	TIMING FOR	
																		"W"	"R"
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	+1	27.3	14.4
2	1	4	7	10	13	16	3	6	9	12	15	2	5	8	11	14	+3 if > 16 - 16	22.6	9.5
3	1	14	11	8	5	2	15	12	9	6	3	16	13	10	7	4	+13 if > 16 - 15	17.6	4.6
4	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16	+4 if > 16 - 15	17.6	4.6
5	1	12	7	2	13	8	3	14	9	4	15	10	5	16	11	6	+11 if > 16 - 16	16.8	3.6
6	1	9	2	10	3	11	4	12	5	13	6	14	7	15	8	16	+8 if > 16 - 15	27.2	14.3

TABLE 2

	SECTOR NUMBERING SEQUENCE											ALGORITHM FOR SECTOR COUNT	TIMING FOR	
													"W"	"R"
1	1	2	3	4	5	6	7	8	9	10		+1	14.1	7.7
2	1	3	5	7	9	2	4	6	8	10		+2 if > 10 - 9	11.1	4.5
3	1	8	5	2	9	6	3	10	7	4		+7 if > 10 - 10	10.1	3.4
4	1	6	2	7	3	8	4	9	5	10		+5 if > 10 - 9	9.3	2.7

The controller chip has a busy time doing its "housekeeping". On a read and a write it must find the track and sector number on the disk which agrees with that in its own track and sector registers. It must do a CRC, or cyclic redundancy check, a sort of checksum, which is written on to the disk on a "write" and checked backed again on a "read". Since a "write" always incorporates a read back, it does take longer than a "read" on its own. Because this "housekeeping" does take a certain short, but finite, time, sequential sectors do take longer to read or write purely because there just is not time between sectors and it becomes necessary for the drive to do a complete revolution before coming to the next sector.

What then is the sector sequence which gives the shortest read and write times? I did a series of experiments with different sequencing and the following two tables give the results. The timings in seconds by stopwatch are from initiating the read or write of 8k data single density to reset on completion. The controller chip is the FD 1771, clocked at 1MHz, the CPU is the Z80 clocked at 2MHz. The drive is a 5,25" Siemens. Table 1 covers the timings for 64 sectors of 128 bytes, while Table 2 is for 32 sectors of 256 bytes. The algorithm is for the sector count to be incorporated into the formatting program. In each case I have assumed a sector numbering of 1 to 16 and 1 to 10 respectively though it can equally well be for 0 to 15 and 0 to 9. Times for other of the FD series of controllers, double or quad density, different MPU or different clock speeds may result in different timings from those given below, but the principle would be the same. Experiment with different sequences and then use the one which is quicker for you. As you will see below, time savings can be quite dramatic. And one final point, the times given include a 1 second delay in hardware to enable the drive to come up to operating speed from stationery.

The following was first published in VOL 4 No.10

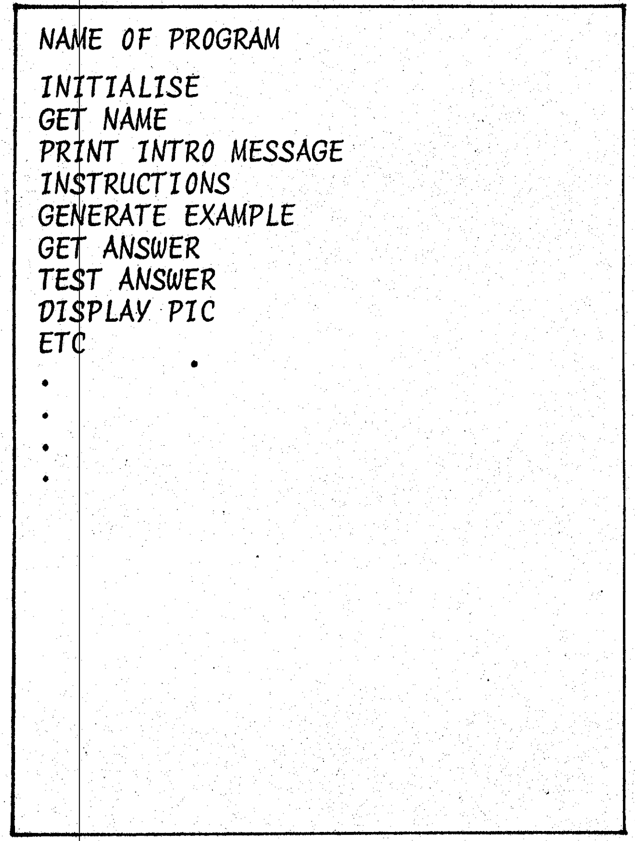
Educational Forum

DONALD COOK

This month's offering may, I hope, provide a few ideas to help with those super challenging CAL programs you all rushed to start on after reading last month's Educational Forum.

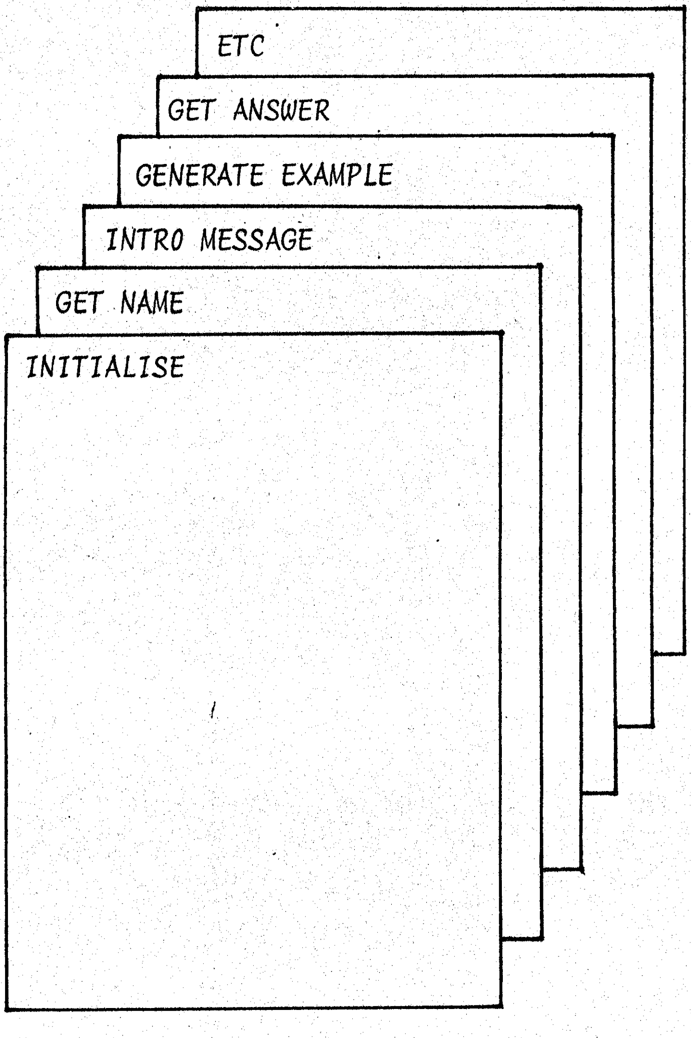
Firstly, a suggestion to help get things organised:

Write out on one page (or more) your program in the following form:

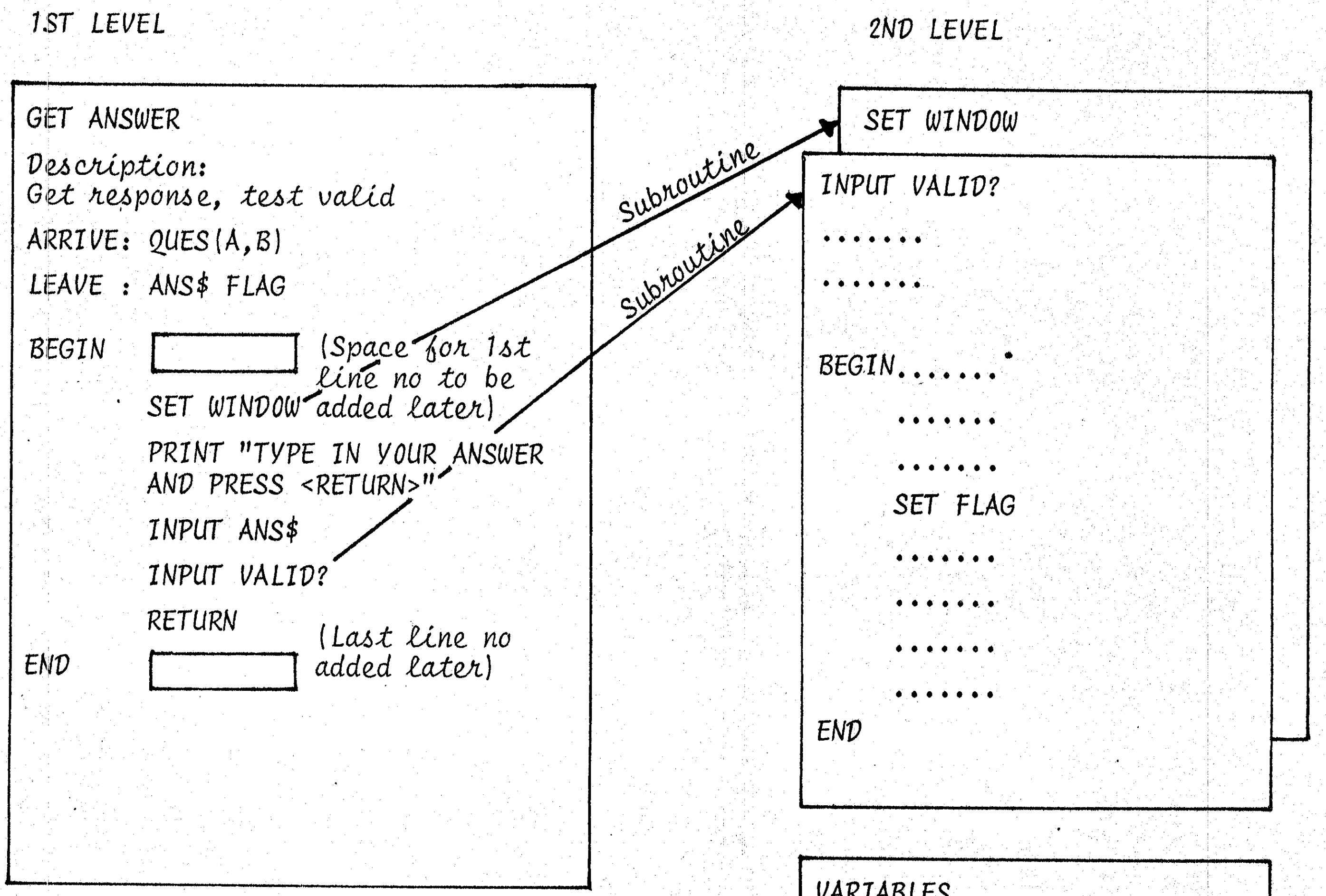


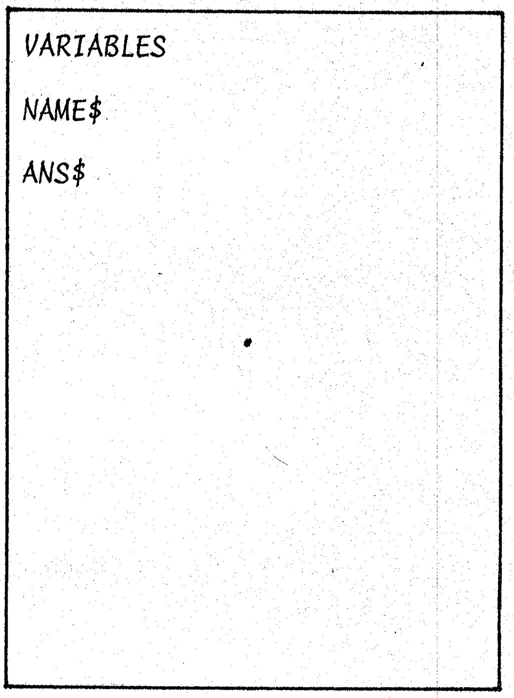
```
NAME OF PROGRAM

INITIALISE
GET NAME
PRINT INTRO MESSAGE
INSTRUCTIONS
GENERATE EXAMPLE
GET ANSWER
TEST ANSWER
DISPLAY PIC
ETC
.
.
.
.
```

Then write the title of each procedure at the top of a clean page:



Then take one sheet at a time, leaving INITIALISE until last of course, and specify the routine in detail in the following manner, making the necessary entries on the sheet labled VARIABLES as you proceed. Don't do the 2nd level routine until you've completed all of the 1st level. That way you will get maximum sharing of 2nd level routines.





I don't have space to give you the whole thing in absolute detail here. If you'd like some further explanation we can discuss it over tea at the meeting.

In the above example you will note that I used the 2nd level routine of SETWINDOW. Now I do this so often that rather than having a lot of POKE'S around when I'm writing a program I simply insert GOSUB SETWINDOW. It's easier to write and understand.

Such a routine is very simple but makes things much easier to comprehend. Try going back to a program 6 months and 20 programs later:

```
5000 REM *SETWINDOW*
5010 POKE 32,L : POKE 33,WD :
     POKE 34,T : POKE 35,B
5015 RETURN
5020 REM *END SETWINDOW*
```

A similar idea is useful for "waiting", i.e. when text is being read on the screen - one can use two ideas.

Firstly, a simple for next loop. For example:

```
FOR A=1 TO 3000 : NEXT
```

However, this means lots of for - nexts dotted around spoiling the reading of your work of art. I like to do it this way:

```
**WAIT**
FOR  W=0  TO WT : FOR Q=1 TO 1000
: NEXT Q : NEXT W
RETURN
**WAIT**
```

Then I can call it with: WT=3 : GOSUB WAIT. Hardly erudite but it's part of a whole philosphy that makes readable code. Try it on other things and see how it helps. Ho! Of course you've probably noticed you can't GOSUB SETWINDOW or WAIT. That's why somebody wrote PLE - PGE; you can "CHANGE" GOSUB WAIT to GOSUB (LINE NUMBER) when you've got everything (subroutines and all) installed.

You will have noticed that this philosphy is not applicable to "development on the keyboard" types. I personally feel you can't do serious stuff without defining your goals and writing out a reasonable skeleton to guide your typing in.

I find that structuring things as I have indicated produces a completed working program much faster than an unstructured approach.

Before I rush back to Appleforth a quick message about a new group that's started up:

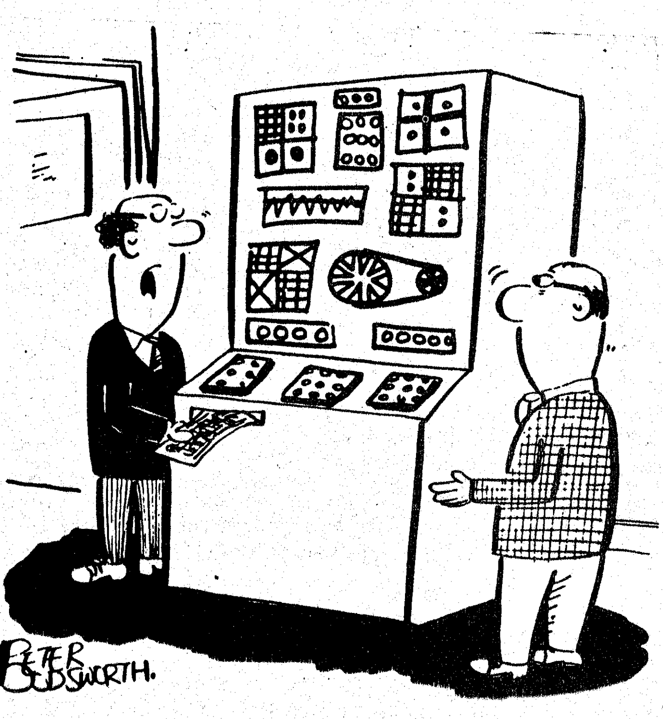
S.C.E.G.

Schools Computers in Education Group

The Chairman is P.Williamson
Phone 462380
or 451236

The Secretary is P.Waker
Phone 445355

Membership is open to all interested in computers for educational purposes. The aim of the group in broad terms is to inform and assist people who are interested in getting started in a school (includeing at PTA level) and developing software of an educational nature.



"WE WON'T GET MUCH OUT OF THIS FOR THE NEXT FEW DAYS, — THIS IS A SICK NOTE!"

The following was first published in
VOL 4 NO.11

1998 And All That

A BRIEF HISTORY OF COMPUTERS

BY GEOFF STURGES
(With apologies to Sellar and Yeatman and the Cape Computer Club.)

For those who have only recently entered the world of computers here is a brief history of those magnificent men and their computing machines who have helped to bring computers to the stage that they are today.

We feel that a basic knowledge of the general history of any subject is usually beneficial to one's overall knowledge of that subject. We hope that your interest in your new-found hobby is enhanced and invigorated by the following account.

To begin with we should explain that there are two basic types of computers, "Antilog" and "Digital". "Antilog Computers" are so called because they work like old-fashioned watches, and you need to look up tables to understand the results, consequently you don't often come across them.

"Digital Computers", on the other hand, are so called because we used to count on our fingers, or digitalis as the ancient Romans used to call them.

Most of us, of course, have ten fingers. However, in the last century a Yorkshire clergyman by the name of George Boole was involved in an accident with his traction engine, and, finding that he had only two fingers left, promptly invented "Binary Arithmetic" and, later, "Boolean Algebra", which he named after an arab.

However, people quickly forgot this rude person who kept giving the two-up, and it wasn't until the 1920's when the famous Red Baron von Neumann (of Snoopy fame) emigrated to the U.S.A. as a P.O.W. and learnt to read (American) English.

The story goes that he was having a bath when suddenly he jumped out of the bath and ran down the street waving his digitals and shouting "By George!". He had, of course, rediscovered Boole's inventions.

He also discovered Boole's missing fingers (which had been preserved in the British Museum) and promptly invented "Octal Arithmetic", which is still mentioned in the cheaper textbooks.

But the history of computing goes way back to ancient times when the men who counted export bananas used to cut "Tally Marks" on sticks. Later, they made marks on wet clay tablets which they then baked in the sun to prevent forgery.

The Chinese then invented the "Abacus", which they sold to the Japanese for mass production. The Japanese still use it to this day as the keys on their calculators are not big enough for Japanese characters.

But man (and woman) yearned for a mechanical means to help him (and her) with his (and her) calculations, and the search for a working computer (and women's lib) was on.

The big breakthrough, though, was in the mid 17th Century when the first "Programmable Calculator" was constructed by a child prodigy called Blaze Pascal, so named because he had a white triangular mark on his forehead (which came to be known as "Pascal's Triangle"). Pascal programmed his machine in a language which he called "Wirth" because, although he was very good at arithmetic, he couldn't spell.

There is also an unfounded theory that in fact this language was not an improvement on the one that he had written previously and that Pascal lisped.

Pascal died just before the Great Plague of London, but he didn't start it as he died in France.

The next giant leap for mankind was in the 18th Century when a certain Mr. Napier, who couldn't master the one times table, took some bones from an old skeleton that he had lying around in a cupboard and made the first "Antilog Computer" which he called a "Slide Rule".

The first textbook on trigonometry, called "The Compleat Angler", was written by Sir Isaac Walton. It could have been the standard textbook today but in it he kept going off at a tangent and discussing things like fishing.

It was a laborious process calculating the tables of sines, cosines, etc. so Walton asked another keen fisherman, Sir Isaac Newt, to help, which he did with great gravity.

Another century or so passed (with the M.C.C. batting) when a gentleman by the name of Charles Baggage built a "Different Engine" with the help of a lady named Lady Jenny Arkwright who wanted to use punched cards for her loom down at Mill. This machine was based on an idea by Leonardo da Vici, and was so called because it didn't work. In fact da Vinci's design was for a submersible helicopter, not a computer.

When manufacturing processes became more reliable mechanical calculators were produced, but then "Electricity" was invented by Benjamin Franklin who tied his front door key to the tail of a kite. However, he received a fatal shock when the kite flew into a cherry tree which he then cut down, but his father found out what he had done. Franklin postumously changed his name to George Washington and then owned up.

In a factory owned by Charles Atlas, which made these new-fangled electric light bulbs, a mistake on the production line produced a light bulb with two elements. Whilst idly connecting this freak to a battery the foreman, a Mr. Edison, realised that he held the first "Diode Valve" in his hands. He was so surprised that he dropped it.

In the U.S.A. valves are called "Inner Tubes" because their Constitution, as amended from time to time, stated that "......it is a fundamental right of all men to give an object another name or to spell it differently if it annoys King George III" (who has since died).

In the 193Ø's von Neumann, whom we have mentioned before whilst he was in a state of undress, built the first "Electronic Computer" which was the size of a large block of flats, consumed enough electricity to power a small town, and caused the first blackout.

These early computers were used during World War II to break the Japanese codes and to calculate the trajectories of atomic bombs. The Japanese couldn't crack our codes, however, because, as you may recall, they only use the "Abacus".

In 1948 two world shattering events took place. In that year Prince Charles was born (he was unmarried at the time and certainly not a father) and two gentlemen named Bang and Olafsen, who both worked in the Bell Telephone Exchange, invented the "Transistor Radio".

"Transistors" are usually made from one of two materials, "Silicone" or "Geranium", the latter being known as "Flower Power". "Silicone" was first discovered in a valley in California, U.S.A., by a skinny lady who was training to be a plastic surgeon.

It wasn't long before people were manufacturing transistors on the heads of pins, and someone had the idea of making "Integrated Circuits", which are several transistors made on the same piece of silicone dipped into arsenic and old lace.

At first these "I.C.s", as they are known, were only simple "AND", "OR" and "OT" gates, but these were quickly followed by "NAND, "NOR" and "NOT" gates as well as five-barred gates.

Integrated circuits are ideal for machines which use a system known as "Bussing". South Africa has yet to produce fully integrated circuits.

Computers which used these I.C.s instead of valves or individual transistors were known as "Mini-Computers" because the ladies who operated them wore short skirts.

As manufacturing techniques improved far more complicated circuits were produced until "Large Scale Integration", or "L.S.D.", came into being. In the early 1970s the first "Microprocessor" was made. This was a four bit device which cost 50 cents. This device was known as the "4004" and this number was derived from the wheel arrangement of the toy locomotive owned by the inventor's son.

It was called a microprocessor because it was known even then that one day it would be the main component used in "Microcomputers".

Computers were originally programmed in "Machine Code", but later on "High Level Languages" were introduced such as "COBALT", "EPOL", and then "ACOL" which was invented by the manufacturers of the "Bridge Challenger". In 1963 "BASIC", which is an acrimony for "FORmula TRANslator", was invented at the "Dartmouth Royal Naval College".

Finally, due to its long evolution, the computing world is full of little foibles which are likely to remain with us for a very long time. For example, if there are eight things to be numbered these are always numbered from "0" to "7", not from "1" to "8", which is why we have "7-segment" displays and not "8-segment". You will also have noticed that the numeral "0" has a line through it. This is because von Neumann was actually Danish.

We hope that with this brief account we have whetted your appetite to delve further into the colourful pages of history of the world of computers.

CAPE COMPUTER CLUB

MB

MICROBITION 82

This is the last report on MICROBITION, Peter L. has informed me that all the stands have been let and deposits paid (he has a wonderful way with these things has our Peter).

As you may know by now Tedelex are making available two VIC-20 machines to the Club. One is to be given away to a lucky winner at MICROBITION; the other is to be given to the Club. Both will be presented at the Club A.G.M. during the July meeting.

We approached Prof. Christo Viljoen, Dean of the faculty of Engineering, to say a few words at the opening and he has kindly accepted to do so.

One thing that is very noticeable from this side of the fence is the excellent organisational groundwork laid down by Colin Rudolph and his team last year - we have reaped enormous benefits from their hardwork.

I would like to record my thanks to all those who have helped in any way to MICROBITION 82 the success it will undoubtedly be. To those who will put their equipment onto the Club stands (even at the risk of having them covered with hamburgers and chips) - thank you. To the stand organisers, Peter Reber and Anthony Rose - thank you.

My greatest debt is to Peter Llewellyn. He has watched everything like a hawk - he has thought of all the little things and he has put in hours of unstintng labour. Peter - a really warm thank you.

Peter, in turn I know, joins me in thanking the exhibitors for this suppoet and in all that they have done to make our task so much easier - we wish you a successful MICROBITION 82.

Atomic Matter

Peter Reber

At the last meeting we had a demonstration of the BBC computer by Colly Myers who agreed to it on very short notice. I think all people present were very impressed by this new machine.
Many people who have not yet a computer will wonder what they should buy, an ATOM or a BBC. My own opinion, based on the demonstration and on various reviews is to buy the BBC. If you buy an ATOM and upgrade to something that can be compared to the BBC you will have to buy a colour encoder, the floating point ROM, something like the toolbox, additional RAM. You have to add the cost of the power supply and you end up with a price higher than the BBC and you will still not have all its facilities like sound generator, a considerable improved screen format, faster processor, user definable function keys etc. To this add the BBC's BASIC and you will see that not even an upgraded ATOM compares well. In fact you can take any micro even at twice the price of the BBC and it will probably come out second best. Should you buy the model A or B? If you can afford it buy the model B. To upgrade at a later stage will be approx. 30% more expensive. To upgrade the machine yourself will not be as easy as it is with the ATOM. The memory chips they used are not widely available and I suspect that some of the other chips may present an even greater problem.

If you have an ATOM already then you may wonder what to do with it if you buy a BBC. Actually these two computers do not compete but rather complement each other. While the BBC is certainly great for programs running in BASIC it is a different story if you want to interface it to the outside world. The very often mentioned central heating system control hardly applies here in SA, a more likely candidate for computer control in this part of the world could certainly be your burglar alarm.

The BBC has 32k of RAM and 32k of ROM and its addressing space is used up. It offers one user I/O port with 8 bidirectional lines but this may not be enough for your application.
On the other hand the ATOM still has a lot of unused addressing space and it is easy to add on some more VIA's, PIA's or UART's.
I was also assured by Colly Myers that they will continue to support the ATOM.

To carry on with last months subject of PEEK and POKE I offer a short program that makes extensive use of these facilities.

```
  10 $#2800="!#80=71+256*72;?#84=70;
GOTO 30"
  20 ?16=0;?17=#28;GOTO 100
  30 PRINT $6,$15,$7''
  40 PRINT "ERROR "?#84" LINE"!#80'
  50 $#2800="LIST      "
  60 I=LEN(#2800)-1
  70 DO I?#2800=!#80%10+#30;
!#80=!#80/10
  80 I=I-1;UNTIL !#80=0
  90 Q
 100 REM MAIN PROGRAM STARTS HERE
```

First a few notes. In line 50, LIST must be followed by at least 5 spaces. More doesn't do any harm, less may crash the program under certain conditions. Line 90 looks and is wrong but is actually the key to this program.
What does it do? If your main program contains an error instead of just getting an error message and the line number the line containing the error will also appear on the screen. This may save you quite a lot of typing LIST xxxx, while debugging a newly typed program.

The error handler in the ATOM works in a fairly simple manner. A pointer to the error handling program is held in memory location 16 and 17. If an error is found the pointer is moved to location 5 and 6, which hold a pointer to the next BASIC statement to be executed, and jumps to the BASIC interpreter. Line 10 sets up the new error handling program and line 20 revectors the pointer. The mainprogram is then executed. In case of an error the line number is saved in !#80, the error number in ?#84 and the program

Jumps to line 30. The screen is then turned on and is put into the unpaged mode. This ensures that whatever will be printed appears on the screen. The bell is sounded and we output two linefeeds. The normal error message is printed. We then set up new instructions for the error handler (line 50). The spaces after LIST are essential because they reserve space for the line number which we then POKE into the string (line 60 to 80). The line number temporarly stored in !#80 is converted from a binary to a decimal number and the digits are further converted to ASCII. Now we just have to get the interpreter to execute the new instructions. This is done with the deliberate error in line 90.
It would be even more helpful if the location where the program got stuck would be indicated. I wrote such a program but unfortunately it does not perform correctly in all cases yet.

```
  20REM ***********************
  30REM ***********************
  40REM ** ATOMIC CLOCK ******
  50REM ***********************
  60REM ** AUTHOR:M.JENKINS***
  70REM **        14/05/'82***
  80REM ***********************
  90REM ***********************
 100REM
 110REM THIS PROGRAM GENERATES
 120REM THE MACHINE CODE FOR A
 130REM CLOCK & DISPLAYS TIME
 140REM ON THE TOP RIGHT HAND
 150REM CORNER OF THE SCREEN.
 160REM
 170REM THE TIME IS UPDATED,
 180REM PROVIDED IN MODE 0,
 190REM EVERY SECOND WITHOUT
 200REM ANY DIFFERENCE TO THE
 210REM NORMAL OPERATION OF
 220REM THE ACORN ATOM!
 230REM
 240REM THE FOLLOWING MEMORY
 250REM IS USED,BUT CAN BE
 260REM ALTERED IF REQUIRED.
 270REM
 280REM
 290REM  NR INTS    : #A0
 300REM
 310REM    HOURS    : #A1
 320REM
 330REM  MINUTES    : #A2
 340REM
 350REM  SECONDS    : #A3
 360REM
```

```
370REM  INITIALISING
380REM  ROUTINE    : #21C-#23E
390REM
400REM  INTERRUPT : #2800 BUT
410REM  HANDLER    : RELOCATBL
420REM
430REM
440 PRINT $12$21
450 DIM LL(9)
460 GOTO w
470wP=#021C
480[
490:LL0 SEI IGNORE INTERRUPTS
500\
510      LDA @#00  REVECTOR
520      STA #0204 IRQ TO OWN
530      LDA @#28  INTERRUPT
540      STA #0205 HANDLER
550\
560      LDA @#C0  SET UP VIA
570      STA #B80B T1 FREE RUN
580      STA #B80E ENABLE INT
590\
600      LDA @#4E  LOAD LOW
610      STA #B804 COUNT
620      LDA @#7A  LOAD HIGH
630      STA #B805 COUNT &
640\                START T1
650      LDA @#14  LOAD 20 INT
660      STA LL5   PER SECOND
670\     BECAUSE 1 INT=50MSECS
680      CLI
690      RTS
700]
710REM LINES 490-700 ARE THE
720REM CODE TO INITIALISE CLK
730 P=#2800
740REM VECTOR INTRPT HANDLER
750[
760      LDA #B804 RESTART T1
770      DEC LL5   NOTE INT
780      BNE LL3 N.RETURN
790      SED       Y.DECIMALS
800      CLC
810      LDA LL8
820      ADC @#1 INC SECONDS
830      STA LL8
840      CMP @#60 SECONDS=60?
850      BCC LL2 N.INIT NRINTS
860      LDA @#0 Y.CLEAR SECS
870      STA LL8
880      ADC LL7 INC MINUTES
890      STA LL7
900      CMP @#60 MINUTES=60?
910      BCC LL2 N.INIT NRINTS
920      LDA @#0 Y.CLEAR MINS
930      STA LL7
940      ADC LL6 INC HOURS
950      STA LL6
960      CMP @#24 HOURS=24?
```

```
 970      BCC LL2 N.INIT NRINTS
 980      LDA @#0 Y.CLEAR HOURS
 990      STA LL6
1000:LL2 LDA @#20 NRINTS PER
1010      STA LL5  SECOND=20
1020      CLD
1030:LL3 LDA @#10 IN WHAT GRAF
1040      BIT #B000   MODE??
1050      BNE LL4 (1234 FINISH)
1060      LDA LL6   DISPLAY...
1070      JSR LL9   HOURS TENS
1080      ORA @#30  CONVERT ASC
1090      STA #8018 PRINT IT
1100      LDA LL6
1110      AND @#0F  HOURS UNITS
1120      ORA @#30  CONVERT ASC
1130      STA #8019 PRINT IT
1140      LDA @#3A  SEPERATOR=:
1150      STA #801A PRINT IT
1160      LDA LL7
1170      JSR LL9   MINS TENS
1180      ORA @#30
1190      STA #801B PRINT IT
1200      LDA LL7
1210      AND @#0F  MINS UNITS
1220      ORA @#30  CONVERT ASC
1230      STA #801C PRINT IT
1240      LDA @#3A  SEPERATOR=:
1250      STA #801D PRINT IT
1260      LDA LL8
1270      JSR LL9   SECS TENS
1280      ORA @#30
1290      STA #801E PRINT IT
1300      LDA LL8
1310      AND @#0F  SECS UNITS
1320      ORA @#30  CONVERT ASC
1330      STA #801F PRINT IT
1340:LL4 PLA   ROM DOES A PHA
1350      RTI
1360:LL9 LSR A;LSR A S/R TO
1370      LSR A;LSR A GET TENS
1380      RTS         VALUE.
1390]
1400RETURN
1410REM BASIC INITIALISATION
1420wLL5=#A0
1430 LL6=#A1
1440 LL7=#A2
1450 LL8=#A3
1460 GOSUB v
1470 GOSUB v
1480 PRINT $6
1490 PRINT "ATOMIC CLOCK"'
1500 PRINT "************"
1510 PRINT "            "
1520 PRINT "********"''
1530 PRINT "PLEASE ENTER TIME"
1540 PRINT " AS"'"H"'"H"'"M"'
1550 PRINT "M"'"S"'"S"'
1560 @=1
1570 A=0
1580 FOR I=6 TO 8
1590     INPUT"TENS"T
1600     ?LL(I)=T*16
1610     INPUT"UNITS"T
1620     PRINT $11$11
1630     ?LL(I)=?LL(I)+T
1640 NEXT I
1650 @=4
1660 LINK LL0
1670 END
```

Notes:

1. If break is accidently depressed, restart with... 'LINK #21C'.

2. COS commands stop the clock until the I/O operation is finished. Correct the time with :-

    ?#A2=#(mins)
    ?#A3=#(secs).

3. If you require the real variables, goto (4).

4. To relocate the routine. Change the address at 780,510 and 530.

5. Clocks vary! Logic timing clocks I mean, you may correct for differences by 'tuneing' locations at lines 600 and 620.

6. If no display is required, delete :-

    1030 thru' 1330
    1360 thru' 1380

   Change :-
    LL3 to LL4 at 780.

7. Noteable addresses:-

   204(H), 205(H)
   Interupt req. vector.

   B80B(H)
   VIA Mode select register

   B80E(H)
   VIA interupt enable register

CONTD ON PAGE 23

Commodore Comments

Ephy Chesler

The Commodore Users Group (CUG) held it's May meeting on the 3rd of May at it's by now usual venue in Plumstead. The 10 or so members who were there were treated to another of Geoff Sturges dissertations on the use and purpose of the Memory Map. Specifically, he dealt with Variables, both string and numeric, and how they were stored and handled in machine code. Lest those of you who were not present consider this too esoteric, let me hasten to assure you it is!

Notwithstanding all the complexities of the subject, Geoffs manner conquered all and even I ended up with a fair knowlwedge of what he was explaining. Copious notes were taken by most of us present, and should any CUG's wish to catch up on what they missed, they need only contact me and I will put them in touch with one of our number who is geographically convenient to them.

The March issue of MicroComputer Printout has a Disk Checker Program by the ubiquitous Jim Butterfield. If any CUG's have entered this whole long program on disk, could they please bring the disk with to our next meeting! It'll save those of us who are poor typists hours of boring work. I have entered it, so if any CUG feels like the challenge of debugging a badly entered program, he, she or it is welcome to a copy (from my 8050 Disk Drive, bring your own Drive if yours is different).

Which brings me to a similar point. Should any of us subscribe to the various magazines catering to our micro's come across an interesting article or program, please let the others know. Particularly if you've actually gone so far as to enter the program; I for one, would love to take a copy off you! This works both ways, as I believe Bob Bailey has actually debugged the lousy program that I entered from the back of the 8050 Disk Operators Manual. Geoff says it's a pretty poor example of what happens in Random Files anyway. Perhaps Bob will let me recopy the lousy program from him one day so that I can decide for myself just how bad it is!

The June meeting will take place on Monday the 7th at 8 pm. Venue as usual is Chesler Cantrell Television, cnr Main & Gabriel Rds, Plumstead. Phone me at 711721 (Bus) or 703107 (Home) if you hve any enquiries.

CONTD FROM PAGE 22

B804(H)
VIA reg. - lo byte of count

B000(H)
PIA reg. - mode select bit
4 set if not alphanumeric
8018-801F(H)
last eight bytes of to R/hand
corner of VDU.

021C-023E(H).RAM

2800(H) RAM

B805(H)
VIA reg. - hi byte of count

Sinclair Users Group Report

DAVID LONG

INTRODUCTION:

At last it has happened, a Sinclair User Group has been formed in conjunction with the Cape Computer Club. Sinclair users need no longer be lonely. There will be monthly meetings at the Athenium in Rondebosch as well as an article containing software, hardware, reviews and advice for Sinclair users in the Cape Computer Club magazine, C3PO.

AIMS:

We aim to bring you the sinclair owner, software for 16K as well as 1K machines. We also hope to be able to produce hardware for all users. What we also want to do is to to have a letter page for comments as well as problems that you have encountered with your Sinclair. We will do our best to answer your questions and the most common questions will be published with suggested advice, so start writing now!

MEMBERSHIP:

To become a member, all that you need do is to contact me for a membership form and details. All that you need is the Sinclair ZX80 or 81. My number is 641901 and I can be contacted between 3pm and 7pm on most days.

PROBLEMS:

Here is our first most common problem - LOADing and SAVEing programs. After hours of enthusiastic typing you finish the program. You haul out your cassette recorder and plug it in. You SAVE the program and label it. But the next time you want it, it refuses to LOAD! What are you going to do?
1. Before you begin to SAVE your program, check that the earphone cable is disconnected at the cassette recorder side. (The most effective method is to remove it completely.)

2. Check that your computer is not next to the T.V. or monitor or right up against it. (The monitors and T.V's have powerful magnetic fields which can cause problems with your tapes) 3. Make sure that your RAMpack does not hum or buzz very loudly as this may well be picked up by the recorder. (Check by listening to the RAM in a quiet room and listen to the 5-second silence at the beginning of your programs.) I hope that these suggestions help but if they don't, then you may have a more serious problem which needs expert advice.)

SOFTWARE:

We will try to cater for all types (excuse the expression). We will try to bring you 1K as well as 16K programs, both games and serious applications.

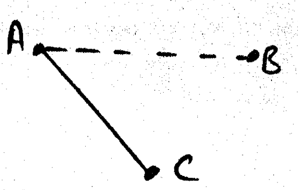
To kick off, here is a 1K game called SPIDER. It was written by Alastair Knight, a well known Sinclair programmer from Cape Town.

The aim of the game is to get the spider, which is heavily disguised as an 'X', to the top of the wall using the keys 5, 7 and 8. There is an element of luck involved, as the spider falls off the wall at random, so it is advisable to rest on the black blocks. This program will not work in 16K.

```
  10 PRINT AT 0,0;"████████████████████"
  20 FOR X=1 TO 18
  30 PRINT "█                  █"
  40 NEXT X
  50 PRINT "███SPIDER███████"
  60 FOR X=1 TO 40
  70 PRINT AT RND*18,RND*3+1;"▒"
  80 NEXT X
  90 LET X=PEEK 16396+PEEK 16397
*256+220
 100 POKE X,61
 110 LET A$=INKEY$
 120 IF RND>.7 THEN LET A$="1"
 130 IF A$="" THEN GOTO 110
 140 LET A=(A$="8")-(A$="5")+12*
(A$="1")-12*(A$="7")
 150 IF PEEK (X+A)=137 THEN RUN
 160 IF PEEK (X+A)<>0 THEN GOTO
110
 170 POKE X,0
 180 LET X=X+A
 190 POKE X,61
 200 IF A$="1" THEN GOTO 140
 210 GOTO 110
```

HARDWARE:

We would very much like to get hold of a circuit diagram which would allow ZX81 users the pleasures of a black screen with white text. Here follows the instructions for inverting the ZX80 screen: You will need: A piece of wire 3cm. long, A sharp knife, A soldering iron and solder. Step 1: Remove retaining rivets from computer. Take board out of the case and lay it down with the keyboard facing down. Step 2: Look for a track which looks like this:



It will be on the right hand side (on the solder side) about half-way up. Step 3: Now cut the track marked AC and connect A to B with the wire. Solder this. Make sure there is a clean break in AC and close the computer. You will now have a fully inverse screen. Don't forget to write to me for any problems and ideas for inclusion in articles, at this address:

David Long
7 Feldhausen Ave.
Claremont
7700
Cape Town

See you at the meeting on Tuesday 27,JULY, at the Athenium at 7.30pm.

CAPE COMPUTER CLUB
MB
MICROBITION 82

TRS~80 Group Report

IAN MCQUEEN

***** A CHEAP START *****

Another month older and still more TRS-80 users appearing. Some of our enthusiastic model I owners are upgrading (?) to model III's and this of course means that there are some fine model I machines becoming available at second-hand prices. Certainly not a system to be sneered at, so if anybody out there wants one and can't find it - give me a ring or phone Derek (71-7879) who also keeps his finger on the pulse.

***** COMMUNICATIONS *****

Communication by various routes between users in Johannesburg and those in Cape Town is hotting up with quite a lot of model I/III software appearing in the Mother City. Some super stuff out there too! Of particular note are some of the latest arcade type games and I believe, although I have not seen it yet, that there is a very good demonstration program illustrating the capabilities of Visicalc.

** FROM YOUR LOCAL NEWSAGENT **

I don't know how many of you subscribe to Computerweek, but for those of you who haven't seen the issue of March 29 I would like to quote the following extract from an article concerning the TRS-80.

"TRS-80 microcomputers will be sold through the Central News Agency shops from next month (April).
This fulfills the tentative agreement between Lucem Holdings and CNA signed in December 1980 when CNA bought Central Data Systems, local agents for the Prime and TRS ranges which gave CNA the right to expand its books and stationery offerings to include TRS electronic equipment.
The first across-the-counter computer department will open in CNA's flagship store in Commissioner Street, Jhb.
Another four are planned in quick succession at CNA's big branches at the Carlton Centre, Rosebank, Sandton and Randburg for a trial period of three months.
If there is a good consumer response, this will be followed up with computer outlets in other big cities and in time in smaller centres throughout the country.
CNA will retail Radio Shack's microcomputer products including printers. In the United States the TRS-80 accounts for 30% of the micro market.
At the lowest end hardware will be available from R500 upwards and will be backed by Radio Shack's extensive software library, a major part of which concerns education, as well as stocks of books, manuals and magazines of interest to computer users, hobbyists and the like.
Mark Devenney, managing director of CDS business systems division, said "Everyone buying a computer will have the back-up service offered by the CDS national network - support throughout the life of the machine, software application support, training schools (registered with the department of manpower which qualify for tax relief) and advice at any point."

"Maintenance contracts can also be taken out on both hardware and software," Devenney said.
Devenney stressed that much effort had gone into training CNA staff who will man the computer sections and into familiarising them with the range of products so that they have a good basic understanding of microcomputer functions.
The design of the computer counters has also been given a lot of thought. Large TV systems are being installed to show the public what the TRS-80 and the programs available for the systems can do.
The Radio Shack range will be expanded to include pocket computers."

Maybe one our Johannesburg user's has already visited one of the stores mentioned and could drop us a line about his impressions. It looks like good news for TRS-80 marketing in S.A. - I look forward to a trip on a model III in the Sea Point branch.

***** ANOTHER SORT *****

This short sort program appeared in the December 1981 issue of TRS-80 microcomputer news from Tandy. Try it - it works!

```
5 CLEAR 1000
10 CLS
20 INPUT"ENTER THE NUMBER OF
WORDS TO BE ALPHABETIZED"; N
30 DIM A$(N)
40 FOR X = 1 TO N : PRINT "ENTER
WORD NUMBER"; X : INPUT A$(X)
50 CLS : NEXT X
60 FOR X = 1 TO N : PRINT A$(X),
: NEXT X : PRINT
70 PRINT : PRINT
80 FOR I = 1 TO N
90 FOR J = 1 TO N - 1
100 IF A$(J) > A$(J+1) THEN 110
ELSE 140
110 T$ = A$(J)
120 A$(J) = A$(J+1)
130 A$(J+1) = T$
140 NEXT J, I
150 FOR X = 1 TO N
160 PRINT A$(X), : NEXT X
```

***** CALENDARS AGAIN *****
As I know we have several number crunching calandar freaks amongst our members, the last offering for this month - also from the same Tandy publication, is yet another program to generate a monthly calendar.

See you again next month !

Monthly Calendar

by Dean Beazly
Mansfield, IL

This program allows you to make a calendar for any month from the year 1700 on.

```
5 LPRINT TIME$"  PROGRAM FOR MAKING A MONTHLY
     CALENDAR BY DEAN BEAZLY"
1Ø CLEAR
     : A=Ø
     : INPUT "# OF MONTH, # OF YEAR FOR THE
     CALENDAR";M,YR
2Ø Y=(YR)-17ØØ
25 LD%=(Y+1ØØ)/4ØØ
3Ø LD%=(LD%+(Y/4))-INT(Y/1ØØ)
35 LD=LD%
36 PRINT"LEAP";LD
4Ø DA=LD+(Y*365)+5
41 IF Y/4=INT(Y/4)THEN FD=1
42 IF M<2 AND FD=1 THEN GOSUB 4ØØ
43 IF M=2 AND FD=1 THEN GOSUB 4Ø2
     : PRINT"FD=";FD
45 ON M GOSUB 21Ø, 22Ø, 23Ø, 24Ø, 25Ø,
     26Ø, 27Ø, 28Ø, 29Ø, 295, 299, 3ØØ
49 LPRINT"SUN     MON     TUE          WED          THU
     FRI    SAT"
5Ø WK%=DA/7
52 DW%=DA-(WK%*7)
55 B=DW%
56 IF M=2 THEN E=E+FD
6Ø FOR L=Ø TO B
     : A$(L)=""
     : NEXT
65 FOR W=L TO 7
66 IF W=1 THEN L=1
7Ø A=A+1
71 IF A>(E) THEN S=1
72 A$(L)=STR$(A)
73 IF S=1 THEN A$(L)=" "
75 L=L+1
79 NEXT
8Ø LPRINT USING"% %     % %      % %          % %
     % %     % %      % %  ";A$(L-7), A$(L-6),
     A$(L-5), A$(L-4), A$(L-3), A$(L-2), A$(L-1)
85 L=1
86 IF S=1 THEN LPRINT DA-5;"DAYS FROM
     1/1/17ØØ"
     : GOTO 1Ø
9Ø GOTO 65
21Ø DA=DA
     : E=31
     : LPRINT TAB(2Ø)"JAN";YR
     : RETURN
22Ø DA=DA+31
     : E=28
     : LPRINT TAB(2Ø)"FEB.";YR
     : RETURN
23Ø DA=DA+59
     : E=31
     : LPRINT TAB(2Ø)"MARCH";YR
     : RETURN
```

```
240 DA=DA+90
     : E=30
     : LPRINT TAB(20)"APRIL";YR
     : RETURN
250 DA=DA+120
     : E=31
     : LPRINT TAB(20)"MAY";YR
     : RETURN
260 DA=DA+151
     : E=30
     : LPRINT TAB(20)"JUNE";YR
     : RETURN
270 DA=DA+181
     : E=31
     : LPRINT TAB(20)"JULY";YR
     : RETURN
280 DA=DA+212
     : E=31
     : LPRINT TAB(20)"AUG";YR
     : RETURN
290 DA=DA+243
     : E=30
     : LPRINT TAB(20)"AUG";YR
     : RETURN
295 DA=DA+273
     : E=31
     : LPRINT TAB(20)"OCT.";YR
     : RETURN
299 DA=DA+304
     : E=30
     : LPRINT TAB(20)"NOV.";YR
     : RETURN
300 DA=DA+334
     : E=31
     : LPRINT TAB(20)"DEC.";YR
     : RETURN
400 FD=0
401 LD=LD-1
402 DA=DA-1
     : RETURN
```

CAPE COMPUTER CLUB

MB

MICROBITION 82

C3PO Flea Market

One SA-BUS wire-wrap card with conector

R 20

Also Programming the Z80 By Rodney Zaks

R 10

Phone Alan Day : 489909

Free to a good home one NASCOM i Z80 system QWERTY keyboard, VDU and parallel/serial I/O (all in a good looking box)

Manual for above system - R200.

note:- System and manual cannot be parted & must go to the same home.

Contact Robin Cristie

Phone: stellenbosch 2740

HP 29c

with manuals, charger etc. (as new) has never done long calculations. Only used by little old lady for Saturday morning shopping.

R 75. O.N.C.O.

Phone Alan Day : 489909

Roger Van Rensberg is looking for a keyboard any offers???

phone Roger at 612809.

Apple Turnover

Anthony Rose

This month:
Build a Ø-3Ø MHZ frequency counter
Add a real-time clock to your Apple
NAKed chips

Having procured some money from the club, the Apple Group can now afford some computer literature, and I think CALL APPLE would be the best buy with our limited funds. The magazines will be brought to all meetings for members to peruse, and you will then be able to borrow the magazines until the next meeting.

I am told that the club has now sold 25ØØ disks, of which I would say at least 15ØØ have gone to Apple owners. I wonder what how you filled those two hundred megabytes of disk space...

Build a frequency counter

I had a need for a frequency counter, but it seemed a little expensive for the few times I would use it, so I built a simple one for the Apple which uses a combination of hardware and software to provide gate times for one tenth, one, ten, one hundred and one thousand milliseconds. The sequence of events performed to measure the input frequency is:

```
1) N=1
2) Reset counter
3) Start counter
4) Delay 1Ø*N cycles
5) Stop counter
6) Get count
7) If count < 4ØØ then inc. N, goto 2
8) Multiply count by 1.Ø1562 & print
9) Goto 1
```

Steps 2-5 are performed by a machine code routine with a seperate delay routine for each delay required. Because a 12-bit counter is used, the maximum count is 4Ø95, so if the count is less than 4ØØ, the delay time is multiplied by ten to obtain a more accurate result. The Apple clock frequency is not exactly 1MHZ, American versions having a frequency of about 1.Ø23MHZ and European ones 1.Ø15MHZ. A local Apple was found to have a clock frequency of 1.Ø1562MHZ, so this is the multiplier used to get the correct frequency. The software to drive the board consists of the machine code delay routines and a controlling BASIC program which does the autoranging, multiplication and printing.

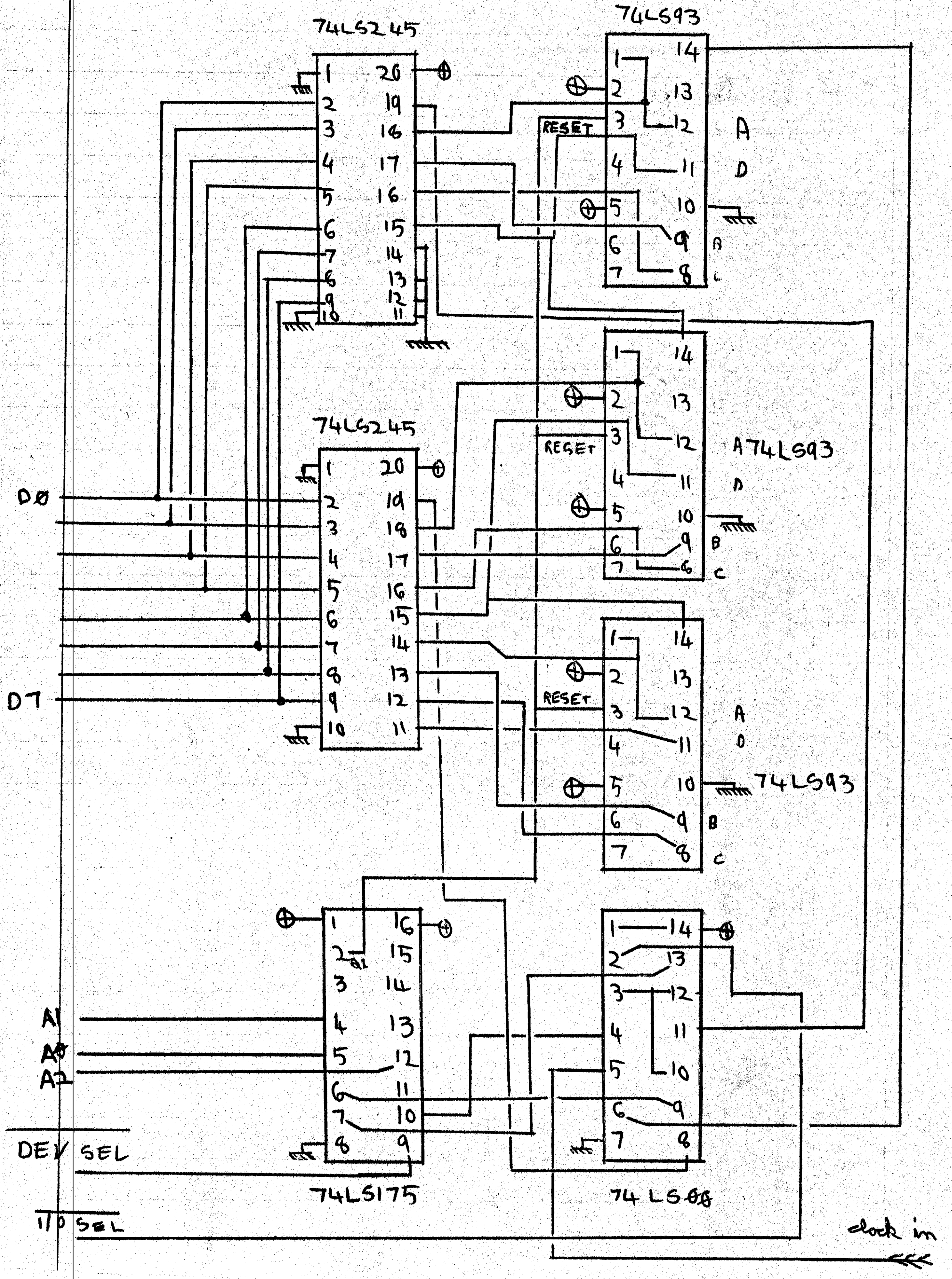
The hardware is very simple - three 4-bit counters form a counter chain, the outputs of which are fed onto the data bus via two 74LS245 buffers. The 74LS175 latch serves to generate the reset and start/stop signals for the board, while the input signal is gated via a nand gate, thus stopping the count when the control input goes low. The address lines latched by the LS175 do the following:

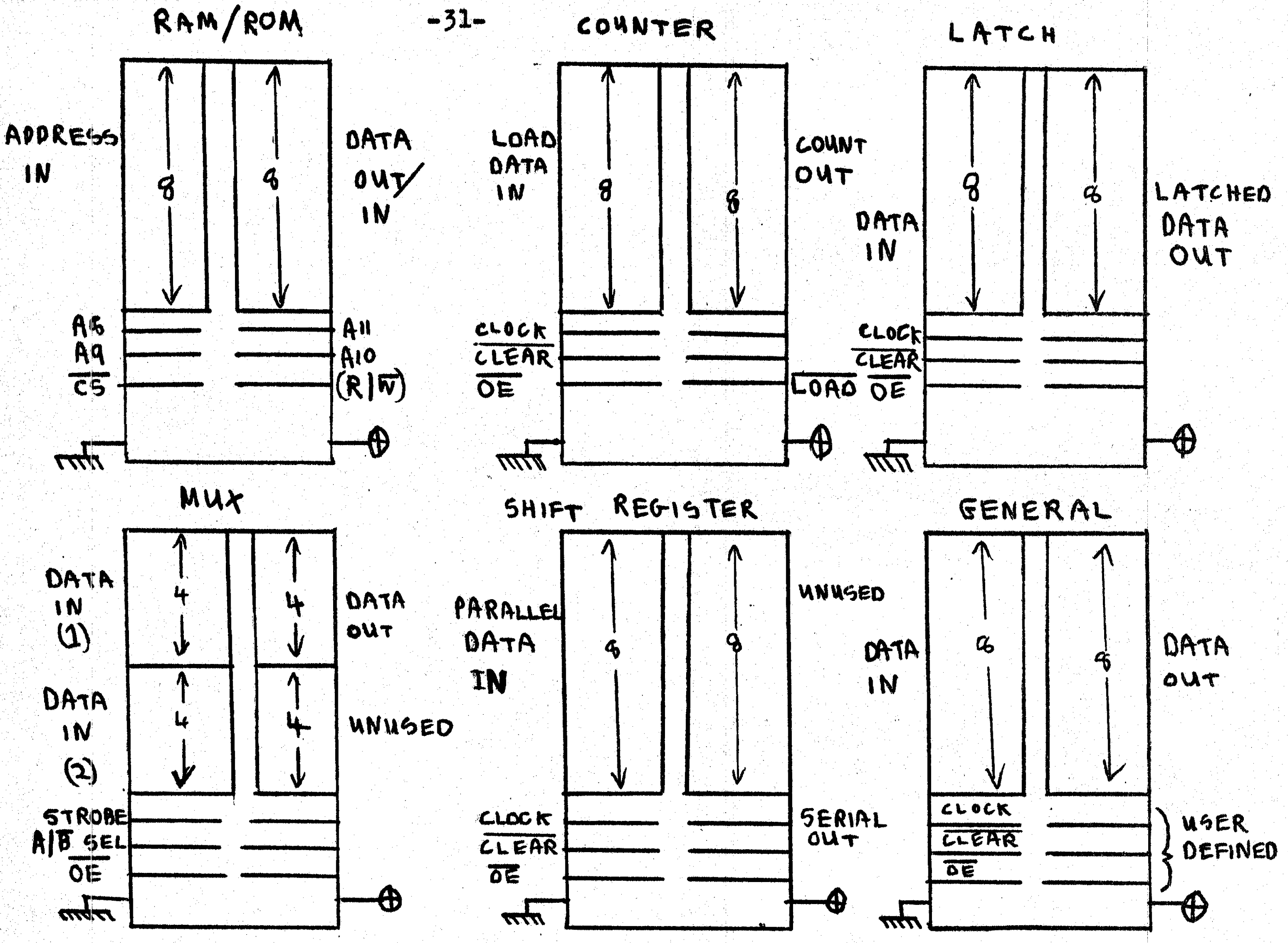
AØ: Count to be read from first or second two counter(s)
A1: Counter reset
A2: counter start/stop

It was suggested that a counter/timer chip could have been used instead, and indeed it could. The reasons it was not were that the circuit was constructed over a weekend when no CTMs (counter/timer modules) were available, also, the maximum count rate of most CTMs is about 2MHZ, so a readable prescaler would have to be used, and we're back to the present circuit. Using a software delay of 1ØØ cycles, the maximum input frequency would be 32MHZ, which corresponds approximately to the maximum clock frequency of a 74LS93.

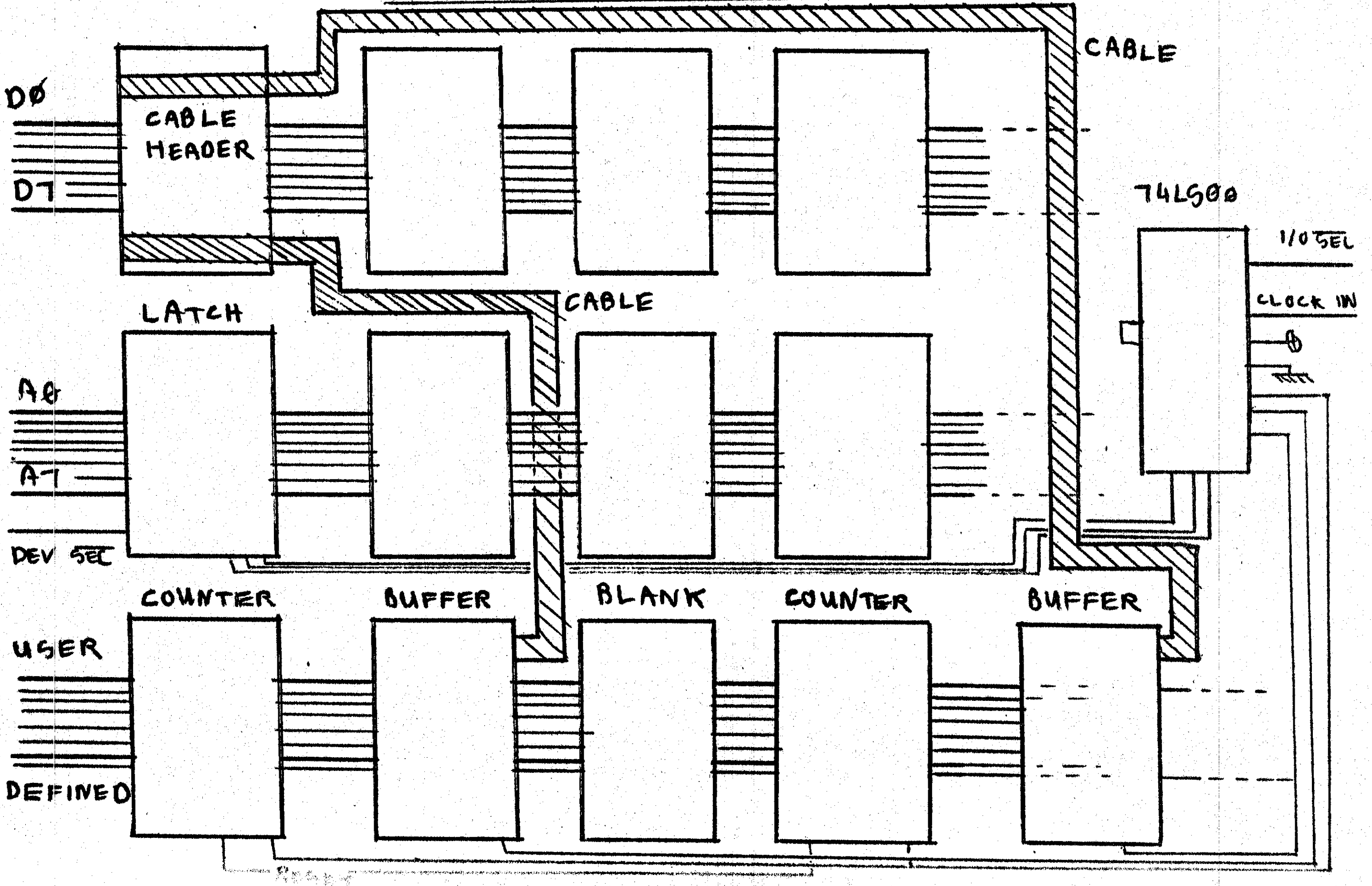
To use, simply connect the frequency to be measured to the clock input, load the machine code routines and run the controlling BASIC program, and the frequency will be displayed on screen. If the frequency to be measured is not TTL compatible, signal conditioning and/or amplifying circuitry may be necessary.

FREQUENCY COUNTER (12/4/82)





IMPLEMENTING A FREQUENCY COUNTER



```
:LIS

1000 ********************************
1010 * DELAY ROUTINES FOR FREQUENCY *
1020 * COUNTER. DELAY 1,10,100,1000 *
1030 * MILLISECONDS & 100 MICROSECS *
1040 * A0 = READ 1ST OR 2ND COUNTER *
1050 * A1 = COUNTER RESET WHEN HIGH *
1060 * A2 = COUNTER RUN (HI) / STOP *
1070 ********************************
1080            .OR $300
1090 START      .EQ $C0C4
1100 STOP       .EQ $C0C0 (FOR SLOT #4)
1110            JMP TENTH
1120            JMP ONE
1130            JMP TEN
1140            JMP HUNDRED
1150            JMP THOUS
1160
1170 ***************
1180 * 0.1MS DELAY *
1190 ***************
1200 TENTH      BIT START
1210            LDX #17
1220 LOOP       DEX
1230            BNE LOOP
1240            NOP
1250            NOP
1260            NOP
1270            NOP
1280            NOP
1290            BIT STOP
1300            RTS
1310
1320 *************
1330 * 1MS DELAY *
1340 *************
1350 ONE        BIT START
1360            LDY #199
1370 LOOP1      DEY
1380            BNE LOOP1
1390            BIT STOP
1400            RTS
1410
1420 **************
1430 * 10MS DELAY *
1440 **************
1450 TEN        BIT START
1460            LDY #104
1470 LOOP2      LDX #18
1480 LOOP3      DEX
1490            BNE LOOP3
1500            DEY
1510            BNE LOOP2
1520            PHA
1530            PLA
1540            NOP
1550            NOP
1560            BIT STOP
1570            RTS
1580
1590 ***************
1600 * 100MS DELAY *
1610 ***************
1620 HUNDRED    BIT START
1630            LDY #99
1640 LOOP4      LDX #200
1650 LOOP5      DEX
1660            BNE LOOP5
1670            DEY
1680            BNE LOOP4
1690            LDX #79
1700 LOOP6      DEX
1710            BNE LOOP6
1720            NOP
1730            NOP
1740            BIT STOP
1750            RTS
1760
1770 ****************
1780 * 1000MS DELAY *
1790 ****************
1800 THOUS      BIT START
1810            LDY #243
1820 LOOP7      LDX #0
1830 LOOP8      PHA
1840            PLA
1850            NOP
1860            NOP
1870            DEX
1880            BNE LOOP8
1890            DEY
1900            BNE LOOP7
1910            LDX #229
1920 LOOP9      PHA
1930            PLA
1940            NOP
1950            DEX
1960            BNE LOOP9
1970            NOP
1980            BIT STOP
1990            RTS
:PR#0

]LIST

10  HOME :AD = 12 * 4096 + 12 * 1
     6: REM  SLOT 4
15 IO = 12 * 4096 + 4 * 256
17 F = 1.01562: REM CLOCK SPEED
20 RES = AD + 2:R1 = AD:R2 = AD +
     1
25  VTAB 10: PRINT "FREQUENCY = "

30  POKE RES,0: REM RESET COUNT
32  CALL 768: GOSUB 300: IF N < 4
     01 THEN 40
34  VTAB 10: HTAB 13: PRINT  INT
     (N * 10 * F);" KHZ      ": GOTO
     30
40  POKE RES,0: CALL 771: GOSUB 3
     00: IF N < 401 THEN 100
```

```
60  VTAB 10: HTAB 13: PRINT  INT
     (N * F);" KHZ     ": GOTO 30

100  POKE RES,0: CALL 774: GOSUB
     300: IF N < 401 THEN 200
120  VTAB 10: HTAB 13: PRINT  INT
     (N * F * 100);" HZ     ": GOTO
     30
200  POKE RES,0: CALL 777: GOSUB
     300: IF N < 401 THEN 250
210  VTAB 10: HTAB 13: PRINT  INT
     (N * 10 * F);" HZ     ": GOTO
     30
250  POKE RES,0: CALL 780: GOSUB
     300
260  VTAB 10: HTAB 13: PRINT  INT
     (N * F);" HZ     ": GOTO 30
300  POKE R1,0:A =  PEEK (IO) * 1
     6: POKE R2,0:B =  PEEK (IO):
     N = A + B: RETURN

]
```

Add a real time clock to your Apple

Few microcomputers have real-time clocks, which is a pity, as they are very useful. Apart from allowing your computer to turn on the TV at a set time, a real-time clock could be used to time programs in order to find the optimum solution, or the interrupts produced by the real-time clock could allow the computer to operate in a foreground/background mode, in other words execute a second program while the first is running. This is accomplished by interrupting the processor, say every hundredth of a second, and sending it off to execute another program, returning to the first one ten milliseconds later.

Adding a real-time clock is a lot easier than it used to be. A single CMOS chip with built-in oscillator replaces many counters and other circuitry, all of which consumed power, and, since the real-time clock must operate even when the computer is turned off, larger batteries had to be used. I am using the National MM58174 clock chip, which uses 1ma when operating with a supply of 5v, and an amazing ten micro-amps at a stand-by voltage of 2.2v. The chip has the following timer outputs: tenths of seconds, seconds, minutes, hours, days, day of week and month, and has built in leap year calculation. It costs R15 at Elektrolink. As can be seen from the circuit, the clock does not turn out to be a single chip, as address and data latches are needed for it. The reason for this is that the time from selecting the chip to read a digit, until the data finaly arrives, is some 600ns, whereas the 6502 is only prepared to give it 300ns. Likewise the clock chip expects data written to it to be stable for 600ns, and again the 6502 only gives it 300ns. A cure to this problem is to latch the address and data going to the chip. To read a register, the register address is first latched, then a couple a microseconds later the data is read. This would take the form of two succesive read operations. To write to the chip, a normal write operation is performed, since the data out is latched, then the following instruction clears the write strobe to the chip (see diagram).

The chip's data bus is four bits wide, so only one digit can be read at a time. If, during a read, the register is updated, a false reading would occur, so the chip produces an illegal BCD output of $0F, and the CPU then simply does another read. The registers and their addresses on the chip are as follows:

00:	Test only
01:	Tenths of seconds
02:	Units of seconds
03:	Tens of seconds
04:	Units of minutes
05:	Tens of minutes
06:	Units of hours
07:	Tens of hours
08:	Units of days
09:	Tens of days
0A:	Day of week
0B:	Units of months
0C:	Tens of months
0D:	Leap year calender - write only
0E:	Clock stop / start
0F:	Interrupt / status

To set up the chip, the test mode register, which facilitates production testing, has to have a $00 stored in it. The chip has an interrupt output (open drain) which can produce an interrupt every 0.5, 5 or 6 seconds. However, the interrupt status register must be read each time an interrupt occurs, or no more shall occur. This means that after a disk access, when interrupts are disabled, our real-time clock will cease to call the CPU to update the time display. Messy. To get around this problem, I simply used a 452Ø and a 4Ø2Ø CMOS binary ripple counter to divide down the 1MHZ Apple clock to produce an interrupt every Ø.5 seconds, or other period.

The 32.768KHZ crystal used by the clock chip can be expensive, and the cheapest way of aquiring one may be to buy a digital watch and remove it from there - most use 32KHZ crystals. A 6-36PF trimmer capacitor is used to fine adjust the frequency, and, if it is not included, the oscillator is not self-starting. The reason a 452Ø counter is used to divide the Apple 1MHZ clock to 32768HZ, rather than use the 32768HZ crystal frequency, is that, when running on battery, only the clock chip would be powered up, and the input clamping diodes on the 4Ø2Ø would then short out the clock, so the time would cease to update. If the 4Ø2Ø is powered from the battery as well, more current is used, especially since it is driving a load.

The address lines latched by the 74LS174 hex latch are:

A0-A3: Register address
A4: Read or write operation
A5: Chip select

Loading a BCD digit from the clock chip will then be accomplished as follows:

```
GETBYTE   LDA SLOTADDRESS+REG+$3Ø
          LDA SLOTADDRESS+REG+$3Ø
          BIT SLOTADDRESS
          AND #$ØF
          CMP #$ØF  (illegal code?)
          BEQ GETBYTE  (read again)
```

The second load will get the data, while the BIT instruction will deselect the chip after the read. If the code is illegal, the process is repeated. SLOTADDRESS is the first address to cause I/O SELECT for that slot to go low, REG is the register to be read (ØØ-ØF) and the $3Ø offset signifies a read operation. Note that address lines A4 and A5 are inverted on the board. The following sequence will write a byte:

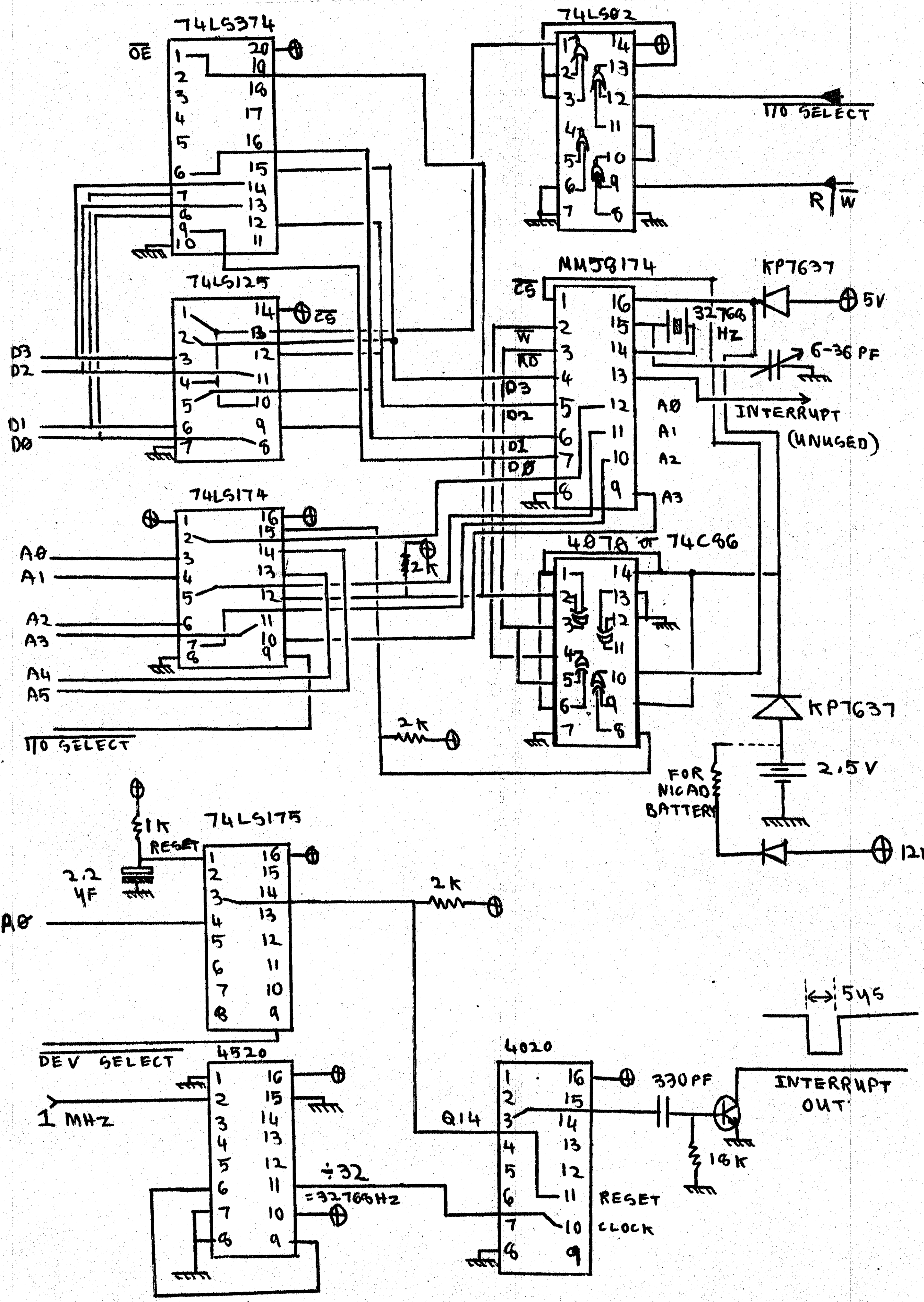
```
WRITEBYTE STA SLOTADDRESS+REG+$2Ø
          BIT SLOTADDRESS
```

Here, the $2Ø offset signifies a write, and again the BIT instruction clears the strobe to the clock chip. I would advise you to get the specifications for the chip. The machine code routine supplied has two sections, the first displaying the time (hours, minutes, seconds) on the top right of the screen and updating the string 'T$' in a BASIC program, while the second part, which need only be called once, will set up the clock and prompt you for the time, which is stored in the clock chip, and will remain there until the batteries go flat. These batteries can be any cells that add up to more than 2.5v, as there is a Ø.3v drop across the diode supplying the chip. If a nicad battery is used, a diode & resistor from +12v will recharge it while the computer is turned on. The resistor value should be chosen to recharge the nicad at about Ø.Ø6 times its discharge current rating. Minimum standby voltage according to the specifications is 2.2v, but I have run it down to about 2v, although the oscillator will not self-start if the chip is turned on at this voltage. Two nicad batteries with a nominal cell voltage of 1.2v should suffice, especially since during the first third of their discharge cycle the output voltage exceeds the nominal output voltage. The current consumption increases with increasing voltage, so, if you want the batteries to last as long as possible, keep the voltage on the 58174 as close to 2.2v as possible, bearing in mind the voltage drop across the diode. The diodes should be germanium types as they have a smaller voltage drop. I am using the KP7637.

Because of the latched address and data, the timing is not critical and this circuit can be used on computers other than the Apple, bearing in mind that I/O Select would then be any decoded chip select signal that asigns at least 64 bytes to that input/output port, and Device Select would be any other decoded address that allows two bytes for that port.

The Device Select line clocks a second latch that enables or disables interrupts. LDA $CØC1 for the clock in slot four will enable interrupts, while LDA $CØCØ will disable them by keeping the 4Ø2Ø CMOS counter permanently reset. If desired, a one-of-eight selector can be used to select between

REAL-TIME CLOCK



various interrupt periods produced by the 4020 ripple clock chain.

Another problem with the circuit arose due to battery backup. As the 58174 has active low chip select and read and write strobes, problems would occur with the rest of the circuit pulling these inputs low. The inputs do have pullup resistors, but these merely serve to increase power drain when a few hefty TTL devices (with a resistance of some 15k each between Vcc and an output) are connected. The solution is to use a CMOS inverter (I used a 74C86 ex-or package) to drive these inputs, and to keep the inverter package powered by the battery. The inputs to the inverter are pulled low, so the outputs go high. Yippy. The 4070 and 74C86 are pin-compatible, but the 74C86 is prefered as no input pull-up resistors are required to make the TTL compatible outputs of the other chips CMOS compatible.

Unfortunately, a bus conflict seems to have claimed the life of my chip, and the local supplier will not be in stock only until the middle of August. Why can't they plan ahead? Anyway, all the software and hardware with the exception of the 74C86 inverter and battery backup were completed before the dastardly event, so all should work.

Finally, to use the software required BRUNning it or CALLing 16384 when it is loaded will run the part that displays the time and updates 'T$' in the BASIC program. CALLing 16387 will run the configure part of the routine, which will prompt you for the time (hours and minutes only) and when you hit <return>, will start the clock going, with the seconds reset to 00. This section must be run each time the clock chip loses the time, as when the batteries go flat.

The section that updates 'T$' has only one requirement, and that is that 'T$' = "XX/XX/XX" is the first statement in the first line ofyour program. For instance,

10 T$="HH/MM/SS"

will do. The string name must be T$, but the string can be as long as you like, with any delimiters you chose. In the case above, only HH, MM and SS in the string would be updated, leaving the delimiters and the rest of the string as it was. If the time-keeping routine does not find a T$ as the first statement, it simply does not update and your program runs normally (eg/ totally unpredictably, crashing, wiping disks, throwing up error messages, etc). The reason the routine is written with the time setup section between the section which updates the display and T$ is that the routines were written in the order in which they appear, and my assembler doesn't have any merge or text relocate commands.

Previously only laboratories could justify real-time clocks, but now for only R50, you could have your very own computer tell you:

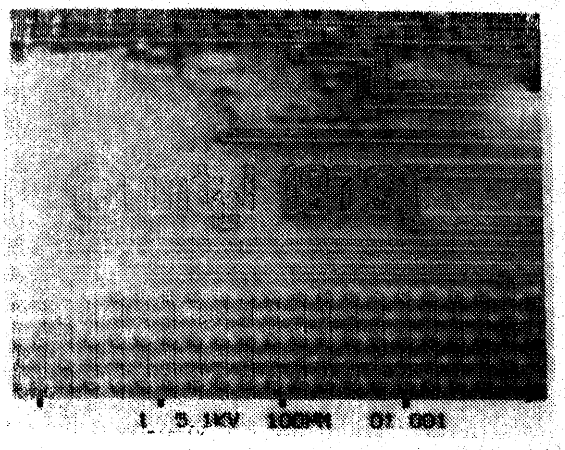
TOO LATE ERROR: IT IS NOW BEDTIME.

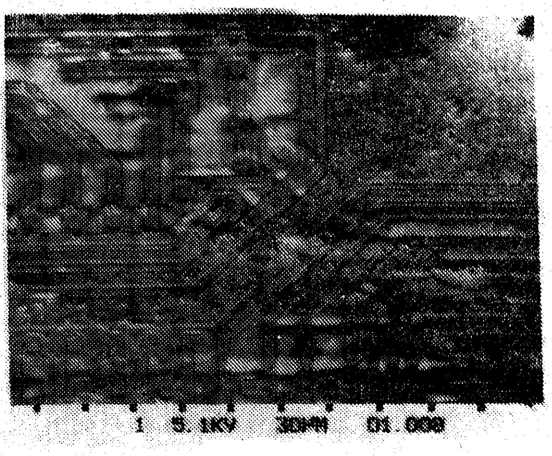
```
1000 ******************************
1010 * ROUTINE TO ACCESS MM58174 *
1020 * REAL TIME CLOCK IN SLOT 4 *
1030 ******************************
1040 * THE VARIOUS  SECTIONS ARE *
1050 * INDEPENDENT. UNUSED PARTS *
1060 *      CAN BE DELETED        *
1070 ******************************
1080           .OR $4000
1090           JMP SETUP
1100           JMP WRITE
1110
1120 SETUP     PHA
1130           LDA #INT   SET VECTORS
1140           STA $3FE   FOR INTERRUPTS
1150           LDA /INT   TO POINT TO
1160           STA $3FF   'INT'
1170           PLA
1180           STA $C0C1 ENABLE INTS.
1190           CLI
1200           JMP $3D0
1210
1220 INT       TXA        COME HERE ON
1230           PHA        INTERRUPT
1240           TYA        SAVE REGISTERS
1250           PHA
1260           JMP BEGIN
1270 END       PLA     RESTORE REGISTERS
1280           TAY
1290           PLA
1300           TAX
1310           LDA $45
1320           RTI
1330
1340 BEGIN     BIT $C410
1350           LDA #$AF   STORE A '/'
1360           STA $425   BETWEEN DIGITS
1370           STA $422   ON SCREEN
1380           LDY #$17
```

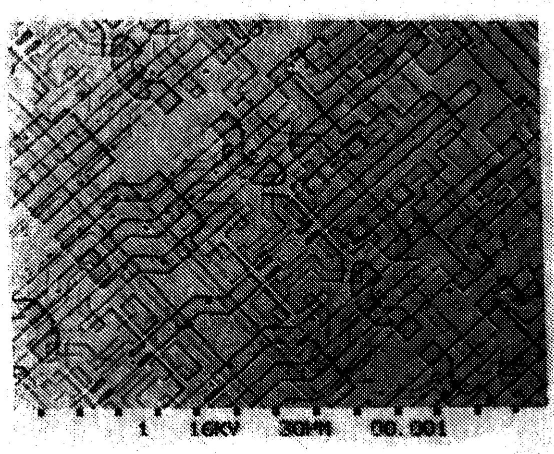
```
139Ø BEGIN1    LDA $C42Ø,Y
14ØØ           LDA $C42Ø,Y
141Ø           BIT $C41Ø TURN OFF CS
142Ø           AND #$ØF
143Ø           ORA #$BØ  FORM ASCII
144Ø           CMP #$BF  READ COLLISION
145Ø           BEQ BEGIN1
146Ø           PHA
147Ø           TYA
148Ø           EOR #$1F  GET OFFSET
149Ø           SEC         INTO OFFSET
15ØØ           SBC #$Ø8      TABLE OF
151Ø           TAX               SCREEN
152Ø           LDA OFFSET,X  OFFSETS
153Ø           CLC
154Ø           ADC #32
155Ø           TAX
156Ø           PLA
157Ø           STA $4ØØ,X STORE ON
158Ø           DEY        SCREEN
159Ø           CPY #$11   NEXT DIGIT
16ØØ           BNE BEGIN1
161Ø           JMP STRING
162Ø
163Ø ********************************
164Ø * ROUTINE TO WRITE TIME TO CHIP *
165Ø ********************************
166Ø WRITE     SEI
167Ø           LDA #$ØØ
168Ø           BIT $C41Ø
169Ø           STA $C42Ø  RST TEST MODE
17ØØ           BIT $C41Ø
171Ø           TAY
172Ø PRINT     LDA TABLE,Y
173Ø           JSR $FDFØ
174Ø           INY
175Ø           CPY #22
176Ø           BNE PRINT
177Ø           LDY #$ØØ
178Ø           JSR $FD6F  GET INPUT
179Ø           BIT $C41Ø
18ØØ           LDA #$ØØ   STOP CLOCK
181Ø           STA $C42E
182Ø           BIT $C41Ø
183Ø
184Ø           LDY #$ØØ
185Ø NEXT      LDX OFFSET,Y
186Ø           LDA $2ØØ,X WRITE TO CHIP
187Ø           BIT $C41Ø
188Ø           LDX CHIPAD,Y
189Ø           STA $C42Ø,X
19ØØ           BIT $C41Ø
191Ø           INY
192Ø           CPY #$Ø4
193Ø           BNE NEXT
194Ø           LDA #$ØF
195Ø           STA $C42E  START CLOCK
196Ø           BIT $C41Ø
197Ø           JMP SETUP
198Ø
199Ø ******************************
2ØØØ * ROUTINE TO INSERT TIME INTO *
2Ø1Ø * A STRING IN A BASIC PROGRAM *
2Ø2Ø ******************************
2Ø3Ø STRING    LDA $8Ø5  CHECK FOR T$
2Ø4Ø           CMP #$54  IN FIRST LINE
2Ø5Ø           BNE RETURN IS IT 'T' ?
2Ø6Ø           LDA $8Ø6
2Ø7Ø           CMP #$24   IS IT '$' ?
2Ø8Ø           BNE RETURN
2Ø9Ø
21ØØ           LDX #$17    GET TIME
211Ø TIME      LDA $C42Ø,X
212Ø           LDA $C42Ø,X
213Ø           BIT $C41Ø
214Ø           AND #$ØF
215Ø           ORA #$3Ø
216Ø           CMP #$3F READ COLLISION?
217Ø           BEQ TIME
218Ø           PHA
219Ø           TXA
22ØØ           EOR #$1F
221Ø           SEC
222Ø           SBC #$Ø8
223Ø           TAY
224Ø           LDA OFFSET,Y
225Ø           TAY
226Ø           PLA
227Ø           STA $8Ø9,Y
228Ø           DEX
229Ø           CPX #$11
23ØØ           BNE TIME
231Ø RETURN    JMP END
232Ø
233Ø TABLE     .HS 8D
234Ø           .AS-"ENTER HOURS/MINUTES: "
235Ø OFFSET    .HS ØØØ1Ø3Ø4Ø6Ø7
236Ø CHIPAD    .HS Ø7Ø6Ø5Ø4
237Ø YSAVE     .DA *
238Ø XSAVE     .DA *
:PR#Ø
```

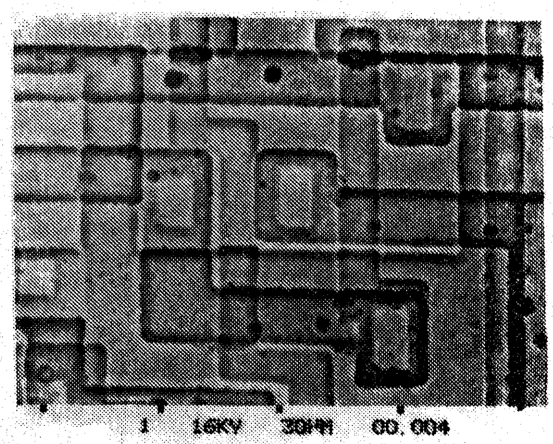
Simulate it!

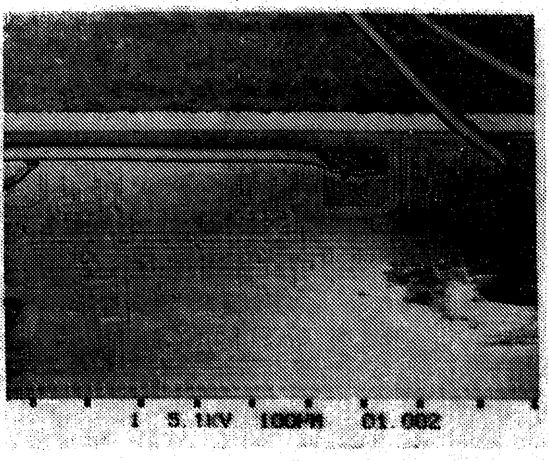
Why did they sink Chief Belgrano? Would it not have been less wasteful if, upon discovering Chief Belgrano in their sights, the British sub would have reported it to Mrs Snatcher, who could have phone Mr Galtieri and said something like 'Listen, Galtieri, we can sink your ship in ten seconds. Why don't you just take it back to port and scrap it, instead of our having to sink it ?', whereupon he would have had no option but to scrap it. This may not seem like a good idea to you, but it probably would be to the few hundred souls languishing two thousand leagues

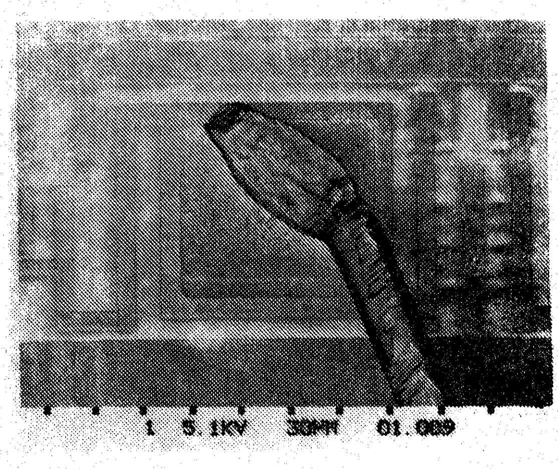

1 5.1KV 100MM 01.001


1 5.1KV 30MM 01.008


1 16KV 30MM 00.001


1 16KV 30MM 00.004


1 5.1KV 100MM 01.002


1 5.1KV 30MM 01.009

down. Taken a stage further, when that bomber could have taken off to attack the Sheffield, why didn't each side agree that the bomber would have got through, so the Sheffield would now be recycled Coke cans instead of contaminating the sea bed. Why indeed have any weapons, instead simulating all on a computer. For instance, each missile costs one million units, so if you want one your GNP must be decremented by that amount, so fewer hospitals can be built, etc. There will be thus a playoff between instruments against life and instruments for it. If too few roads are built, the people will get unhappy and will elect a government that will build roads. It would be an extremely complex simulation, and not everyone would like to lose that easily. Perhaps they would prefer having two thousand metres of water above their heads....

NB/ Certain series of Apple disk drives have a defect in that the disk centering hub was manufactured slightly smaller than it should have been, with the result that disks are not always centred correctly. If a disk gives I/O errors, try removing it, and, with a finger at the centre of the disk, push the disk to one side of the jacket, and try reading again. This may help you to load the data to make a backup.

NAKed chips

Wouldn't you like to know what the inside of all those chips in your computer look like? The accompanying photo's were taken by the UCT electron microscope. At the bottom of each picture are three bits of information: the leftmost number is the energy of the electrons hitting the target; the central number is the distance represented by the scale above it; and the rightmost number is simply the number of the photograph. For instance, if the distance between the gratings is 10mm and each represents 20 micrometres, then the magnification is 500. The objects seen are a SC/MP MPU which had its top removed several weeks before, hence the contamination, and a 2716 EPROM which had its top rather forcibly removed. The photo's look quite attractive, so I hope they have reproduced well in the magazine.

Z80 Group Report

J.W. nearly recovered from 'flu.

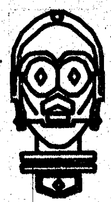
J.V. nearly got disk drive working.

J.W. nearly got 32K RAM working.

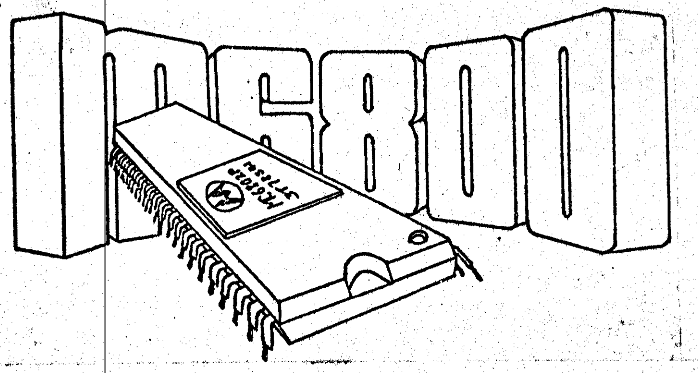
D.S. nearly got Z-80 Big Board working.

Welcome to Mike Moncrieff - a visitor to the Z-80 Group Meeting - who is getting a Heath System from the States.

Z-80 Group nearly got a Group Report!



BEFORE the advent of the computer, only a wife could record all one's transgressions in a permanent memory bank.



MICROPHILE

Neil Walsh

6809 Review.

With all the different microprocessor chips on the market it is very difficult to know which is the best to use in a system. This review will show you just how good the Motorola microprocessor is. The main question is - would you ever use anything less than a 6809? There are some very new and exciting 16 bit microprocessors coming onto the scene like the 8086, Z8000 and the 6809's bigger brother the 68000 which (of course) takes the cake.

Are the earlier microprocessors really inadequate? After all, they all run sophisticated high level software just like the 6809. These micros include the Z80, 6502 and 8085 (the 6802 is obsolete).

These microprocessors were designed in the days when the need for high level software on micros was not really established. The 6809 removes some of the risks involved with undetected high level "bugs" that can hide in a microcomputer. There is a tendency for people to use microcomputers for tasks previously reserved for bigger computers and often get very surprised when strange, wierd and wonderful things happen.

Of all the outstanding features of the 6809, the most attractive is that it is now possible to write truly modular software and be able to safely test these modules individually. This is because the 6809 was the first microprocessor which was designed to run position independent software. The position independent machine code assembled from a correctly written source program will run anywhere in memory. The operating system has to remember where it has put the modules in the available memory and then to call them when needed. The absolute address in memory of the modules is then not important provided there is sufficient memory available for them at the time. On Z80 systems transient programs have to load into an absolute memory area - thus only one transient program can be resident at one time because they must occupy the same area of memory. The 6809 instructions usually reference code or data tables relative to the absolute running address of the program at the time of execution. Stack and page handling make the referencing of variable data or workspace a dream.

High level or complex programs make extensive use of data pointers. These software pointers, pointing to memory locations containing data, have to be manipulated and updated by the running program. The earlier microprocessors do not have address register compare instructions which make it easy to test and validate pointer registers. On the 6809 it is possible to compare any of the four 16-bit pointer registers to an unsigned or signed value (which may be a constant or variable data) and make a branch decision based on the result. There are no less than fourteen separate decisions allowed for this purpose. This makes for safer running software. On the earlier microprocessors one had to write sometimes lengthy subroutines which could themselves have 'bugs'. On the 6809 the test is a single instruction! The power of the processor extends far beyond this important requirement. With a single machine instruction it is possible to add or subtract a 7 or 15 bit value from any of the pointers without disturbing the accumulators. This value could be an immediate or constant value or it could be a variable value from the accumulator or accumulators.

The pointer or index registers of the 6809 can all be used with 16 powerful indexed memory referencing modes. These include constant or variable table offsetting of 0, 5, 7 or 15 bits (all positive or negative) making it possible to perform complex table accesses with only a single instruction. Previous micros again would require complex subroutines to achieve this with the associated risk of hidden software bugs that show themselves months or sometimes years after the software was thought to be completely tested. Of course complex programs require more than four pointer registers. The 6809 facilitates the easy saving or stacking of registers so that multiple pointers can be easily maintained. Because of its powerful and extensive indirect addressing modes, high level software can be super fast and also safe.

Home computers exploit memory mapped screen display for diagrams, drawings, moving pictures or text. This is for educational or instructive reasons or fun. Because of the above mentioned indexing modes the 6809 makes screen handling easy and effective and of course very fast. The binary multiplication feature of the processor make the calculation of screen cursor addresses simple and fast. The Computer Club 680X machine makes extensive use of the features of the 6809 MPU to achieve its good screen handling. These facilities would have been extremely difficult to achieve with any of the earlier microprocessors, yet with the 6809 it was a straightforward task.

The 6809 has all of the arithmetic and logic functions found on the Z80 and its predecessors. Bit shifting and testing, addition and subtraction are easily accomplished. Unlike any of the other 8 bit micros (excluding the 6802) the 6809 has two full function ALU accumulators which can be joined together to form a 16 bit accumulator when required. When it comes to comparing accumulators with constant or variable data the 6809 can test for greater than, less than, greater than or equal, less than or equal (all with signed or unsigned data), equality, positive or negative or zero conditions with just one instruction unlike earlier micros which often need two or more tests to fully establish branch conditions for "decision" making.

The Z80 has its much acclaimed block move or search facilities. The 6809 does not have these but because of it auto indexing feature it can handle these with similar ease and speed. This is where a set block of memory data has to be moved to another area of memory or where a data pattern has to be located in a set area of memory. The advantage of the 6809 is that it can handle varying size blocks between two absolute or relative memory locations and in any direction.

One may wonder why it may be better to use the 6809 instead of one of the new 16 bit microprocessors. The question is reasonably easy to answer. Here we are looking at home computers and these do not have to process vast amounts of data. The far superior arithmetic handling capabilities of say the 68000 would be wasted on a home computer. Most home users would be perfectly prepared to wait a few extra milliseconds for the answer to a complex calculation. It is unlikey that a home user would require to run more than one program at a time or that more than one member of the family would want to use the computer at one time. One would not require very fancy file management or system security because outsiders wouldn't be able to access your computer anyway.

There is adequate high level software for the 6809. This is because the architecture of the 6809 was determined by software engineers. The ease of writing compilers and run time software for this processor has resulted in the major software houses being able to produce really good software. Of course there are Basic interpreters, word processors and editor programs as well. Up to now the 6809 has not been widely used as a home computer processor mainly because the 6502 got in earlier and found its way into the famous APPLE computer. The new TRS80 colour computer does use the 6809 however. Screen based games and educational packages tend to be very hardware dependent and the "APPLE II" has really been the only standard if you can call wide usage a standard.

The Cape Computer Club members have developed a machine using the 6809 processor giving facilities even better than the Apple or TRS80 colour computer and just might set some sort of standard, who knows. The more people that one can get to write games and educational programs, the more successful the project will be. The microcomputer scene has shied away from any form of standardisation resulting in an often excessive diversity of machines. The 6809 microprocessor offers some kind of standard which could serve as a basis for further development and improvement. Would you use any lesser microprocessor than the 6809?

CAPE COMPUTER CLUB

MB

MICROBITION 82

DROMEDARIS HALL, GOOD HOPE CENTRE

FRIDAY, 18, AND SATURDAY, 19 JUNE 1982