SOUTHERN CROSS COMPUTER

USERS MANUAL

Direct all enquires to:
DIY ELECTRONICS
PO BOX 88458, SHAM SHUI PO
HONG KONG.

Fax: 852 - 2725 0610

Email: peter@kitsrus.com

Web: http://kitsrus.com

June, 1997.

CHAPTER 1

INTRODUCTION

As we all know prices for Personal Computers (PC's) have dropped to such low levels that almost anyone today can put a huge amount of computing power into their hands at minimal cost. Buying a computer and software to go with it today are routine tasks no more complicated or expensive than buying a new TV, CD player or washing machine. With the coming of programs like Windows 3.1 you can easily have a completely computer illiterate person operating the latest programs just by moving a mouse and click - click - clicking. One never needs open the computer cover. The fact that the available literature is almost completely about how to operate various software packages reflects the popular swing.

However, along with this undoubted advance in human progress there is a negative side. This downside is that we have lost touch with how a computer actually works. Any popular books which mention computer hardware design do so only in a superficial way. One needs to go to computer engineering texts to get detailed material. For the beginning student these sources can be unfriendly and difficult to read. Journals like *Elektor* are also too advanced while other journal are too simple. There is very little in the way of simple but up to date hardware available anywhere.

This is not just cry for a return to the good old days! There is a very real need for basic, computer do it yourself Kits. The heroes of the real computer world are NOT the microProcessors (uP) we all read about - 80386, 80486 etc. Look inside a washing machine, a fax, CD player, photocopier, portable telephone, photocopier and you will find a microController (uC). They have names such as 8051, 80752, PIC16C54, 68705P3, 77C82. Not many people realise that there is more computing power in the latest model of car from Japan than in their 80386 PC sitting on their desk.

Learning to program micro Controllers generally means several years study at a technical institution or working for a company which programs uC chips for a particular appliance. There is almost no hardware available commercially which will teach it. The only books available are generally the uC Data Books from the manufacturer.

The Southern Cross Single Board Computer

The Southern Cross is a Single Board Computer (SBC) designed for the 1990's. It is aimed to teach two things:

- how a computer works. We start with machine code entry using the onboard hexadecimal keypad. When this is outgrown the Southern Cross can be connected to a PC or by using an EPROM emulator. A series of addon boards further increase the teaching potential of the system.
- the practical and philosophical methods of program development for modern microprocessors and microcontrollers.

In both cases the Southern Cross can be connected to a PC to use its power and facilities for program development, linking and assembly. The program can then be downloaded to the Southern Cross either into RAM or to an EPROM emulator. In the coming months we will develop addon and support boards for the Southern Cross which illustrate and teach these concepts.

The Southern Cross is built around the Z80 micro Processor. This is an 'old' chip having first come out in the late 1970's. However, it has been steadily upgraded and supported. It is readily available at low cost. (Do not think that 'old' means out of date; the popular 'Game Gear' game uses a Z80 chip.) We use a low power cmos version. All IC's on the Southern Cross are cmos.

This documentation has been designed to go straight into using the Southern Cross as soon as the Construction and Circuit Description has been covered. We have not attempted to revise or summarize computer basics. There are literally hundreds of books on the topic. The Southern Cross should be used in conjunction with several computer texts for reference. Similarly we have not dealt in great depth with the Z80 Instruction Set or Theory - again there are tens of books on this. In the Technical Manual the Data Sheet on the Z80 chip is reprinted. Also some pages from the Z80 Technical Manual are included.

Future of the Southern Cross

The Southern Cross is a stand alone, fully operational SBC. However, we have a program of addon boards, support boards and Monitor upgrades which will greatly enhance its power and teaching capabilities. When you Register with us you will automatically be notified of new releases. Comments about the Southern Cross and ideas for addon boards and demonstration programs are requested.

Support Boards now Available.

- EPROM emulator.

- 8x8 LED display. Up to three can be connected to I/O Port CN1.
- Relay Board.

Full documentation and program examples are included with all boards. A separate Technical Manual is available containing copies of all Data Sheets of the ICs used on the Southern Cross.

Components for Serial interface	1 packet
Keycap letters strip.	1
Southern Cross PCB	1

Southern Cross Documentation:

This User Manual.
Alphabetic and Numeric Z80 mnemonic code list.
Registration Form.
Floppy disk. See Read1.me file.

Southern Cross Components:

Resistors, 1/4W, 5%:		
100R (brown, black, brown)		1
330R (orange, orange,brown)		1
560R (green, blue, brown)		1
1K (brown, black, red)		7
2K2 (red, red, red)		1
10K (brown, black, orange)		1
22K (red, red, orange)		1
100K (brown, black, yellow)		2
10M (brown, black, blue)		1
20K KOA vertical adjust potentiometer		1
4 X 100R SIL discrete network, SIL1 & 2		2
SIL3, 22K resistor network, A type, 6 pin		1
Capacitors:		
33pF ceramic		2
100pF (101) ceramic		1
100n (104) monoblock		7
1uF electrolytic		2
3.3uF "		1
10uF "		2
1000uF "		1
BC547 transistors		7
4.00 MHz crystal		1
W02 bridge rectifier		1
16 pin IDC box header		1
40 pin IDC box header		1
Integrated Circuits:		
74HC273	IC1 & IC4	2
74HC138	IC2 & IC3	2
Z80CPU	IC5	1
27C64	IC6	1
6264	IC7	1
74HC244	IC8	1
74C923	IC9	1
74HCU04	IC10	1
74HC74	Single Stepper IC	1
IC Sockets:		
14 pin		1
16 pin		2
20 pin		4
28 pin		2
40 pin		1
5V Buzzer		1
LM7805CT		1
Heat sink, nut, bolt		1
Power Jack		1
CM1-5615S LED hi-eff red display		6
Keyswitches		21
Keyswitch caps		20
Tinned copper wire		5'
Rubber feet		5
SPDT PCB-mounted switch		1
5mm LED		1
6 pin harness & PCB-header		1 pair

CHAPTER 2

CONSTRUCTION

Constructing the Southern Cross is no more difficult than soldering any small electronic kit project; it just takes longer. Before you start the project you should have a quiet place to work with a light. Have a range of tools available to use. The minimum you will need are cutters, long nose pliers, multimeter and a low power soldering iron and solder. Keep your work area tidy.

The Southern Cross computer is built on a single sided 1.6mm fibre glass printed circuit board (PCB). The PCB has three layers:

- on the top of the board is the component overlay (sometimes called silk screen overlay) which shows where to put the components,wire links and gives other information.

- underneath the board are two layers. One has the copper tracks which connects the solder pads of the components. The second is a plastic solder mask which covers all the board except around the solder pads. The solder mask helps in soldering the pads.

A listing of all the components is given on the previous page. The first thing to do is place all the components into a container then check them off against this listing. If there is some item left out let us know and we will supply it to you.

The overlay is detailed and you should not any problems seeing where each of the components goes. The main thing to watch is the polarization of some of the components which means they must be put into their holes in the correct way around. The following components are polarized:

- electrolytic capacitors. It is usual to mark the positive lead of electrolytic capacitors on the PCB overlay even though it is the negative lead which is marked on the capacitor itself. There are six to watch for C1, C2, C3, C5, C11, C15.
- the buzzer B1 has the positive lead marked on it.
- the six LED displays. Match up the decimal point.
- 5mm LED1. The flat part on the LED (cathode) faces to the top of the PCB. The cathode lead is the shorter of the two leads.
- all IC's (integrated circuits) IC1 to IC10. All IC's have their pin 1 on the top left of their respective IC sockets. All IC's point to the top of the board. The bridge rectifier DB1 has a + to mark its position. The 7805 voltage regulator faces up after it is bent over.
- the keyswitches. Each switch has a flat part on one of its sides. This faces towards the bottom of the PCB. The flat part is marked on the overlay. All 21 keyswitches are identical. 16 of the same colour are supplied for the hex numbers 1 through F.
- transistors T1 to T7.

The following components are not polarized. They may be put on the PCB either way around:

- all resistors and the resistor networks SIL1 and 2, the 4MHz crystal.
- all other components not mentioned.

The following order of construction is suggested. It is usually easiest to assemble the lowest height components first then gradually solder in the taller components.

1. Links. First there are 54 links to make on the component side of the PCB. Each of them is marked with a white straight line. (One link is marked with a dotted line. This is the link to remove later when you add a DS1216B SmartWatch socket to the board. This is discussed in Chapter 8.) A length of tinned copper wire has been provided in the kit for this. All the links except one are vertical. Cut the length off that you need then use the needle nosed pliers to bend each end. Be careful that the link does not pop out of the hole when you solder it.

2. Resistors.

3. IC Sockets.

4. The heat sink goes under the 7805 voltage regulator. When you bend the 3 legs of the 7805 please use the pliers. Do not just push the case backwards as you may damage it. It is good practice not to put mechanical strain on the IC case itself.

5. LED Displays, keyswitches, minispeaker and other components.

6. The very last thing to do is insert the IC's themselves into their sockets. All IC's point to the top of the board. That is pin 1 is in the upper left corner.

How do you know that the Southern Cross works?

When all the links, components and other items have been put on the board stop and check it again for the following things:

- electrolytic capacitors around the correct way.
- IC's inserted into their sockets the right way around. All IC's on the Southern Cross point upwards.
- that all the links are on the board
- that all links, IC sockets and components are soldered
- put the SPEED switch in the F)ast position

If you are happy that everything is correct then plug in a plug pack output of 9 to 15V AC or 9 to 12V DC into the power jack. (You do not have to worry about polarity of the input jack; that is taken care of by the bridge rectifier.) The Southern Cross should beep, the power LED should light up and the numbers '2000' should appear in the group of 4 Address LED displays.

What to do if it does not work.

You should consider yourself lucky if the Southern Cross does NOT work. You will learn a lot about real world electronics by fixing thing as well as by just building them. The process of thinking about why a piece of equipment which should work but fails to work is an important one. (The first Southern Cross we built up did not work - we had put the Z80 chip in upside down. It took about 5 minutes to find the problem. Surprisingly the chip was not destroyed by the experience. The second prototype also did not work - one pad and a link was not soldered.)

Remember the problem almost certainly is a mistake you made during construction. Firstly, did you put in the links on the top (component side) of the board? Check that you got them all.

The most common cause of Kit failure is bad soldering - dry joints, forgot to solder a pad, solder shorting across two pads or the link/component dropped out of the solder pad but not far enough out of the component side to be easily noticed. So, inspect all soldering under a strong light. Check the trackwork at the same time to make sure that a track is not missing or cut accidently.

The next most common cause is putting in components the wrong way. Check the orientation of all IC's, electrolytic capacitors, keyswitches, buzzer, bridge rectifier and LED displays. Check that you have the correct IC's in the right place.

It may be a good idea to have someone else do this checking since many people cannot believe they can make such silly mistakes and only do brief checking.

If the Southern Cross is completely dead when the power is connected (and the LED1 next to the heat sink does not go on) then clearly the place to look is the bridge rectifier, 1mF elcap and the 7805 regulator. (Or maybe you put the LED in the wrong way?) Similarly if some of the board is active and parts are dead then this will indicate where to direct attention.

Use a multimeter to check that power is going to all the IC's. Look at the Data Sheets to see which pins are the power lines. Use the resistance meter to check that links and pads are soldered together.

Not all pads need to be soldered with a component in it. Two pads next to the 1mF capacitor are there as an optional power source to add-on boards. The pads attached to pin 21 of the Z80 CPU and pin 22 of the EPROM & RAM are for an optional expansion of the board to use a day/date/time/nonvolatile RAM chip. This is explained in Chapter 8. 5V and zero pads have been provided near the voltage regulator in case you want to power a logic probe.

Finally, if all else fails then we will you can send the board to us and we will get it going for you for a small charge plus return postage. We hope that you never have to do this but at least you know the service is available. To repeat: we have found that 99% of faults are non-soldered pads or dry-joints, IC pins bent, ICs around the wrong way or links missing.

CHAPTER 3

CIRCUIT DESCRIPTION

Today there are hundreds of books available which outline the way a computer works. They all start with block diagrams similar to the one on the right then progress in various ways through the topics of memory decoding, RAM, ROM, input/output, CPU architecture etc. etc. Thus we have decided not to even attempt to summarize this information here. We will give a description of the four circuit diagrams which describe the Southern Cross and ask that you read this together with reference books.

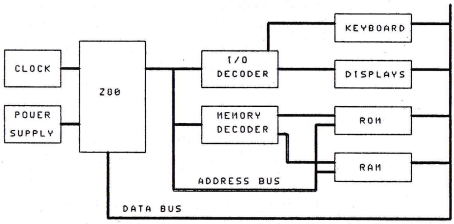

Figure 1. General Computer Architecture.

Figure 2 shows details of the **Power Supply Unit** and the Clock or **Oscillator**. The PSU is standard design. The bridge rectifier allows 9 to 15V, AC or DC input. This means that any power pack, either AC or DC, and of any tip polarity can be used provided the voltage is in the 9 to 15V range. The 7805 is a 5V voltage regulator. The LED indicates when the power is on. You can remove it if you do not want it. The 100nF monoblocks are the standard way to remove spike transients in the power rails. A lot of effort has gone in to reducing spikes and making sure there is enough power available in the circuit when it is required (IC switching, LED displays) and to supply the addon boards. Spike reduction is the reason the ground track seems to be redundant in places on the PCB.

The frequency locked and variable oscillators are also standard. 4.00MHz was chosen so that it is a easy number to use when calculating delay loops, software for timers, etc.

Figure 3 shows the major **Input Interface** for the Southern Cross - the keyboard circuit. Again this is a standard and proven circuit. See the data sheet in the Technical Manual for more information about the 923 IC which is specifically designed for a 5 x 4 matrix of key

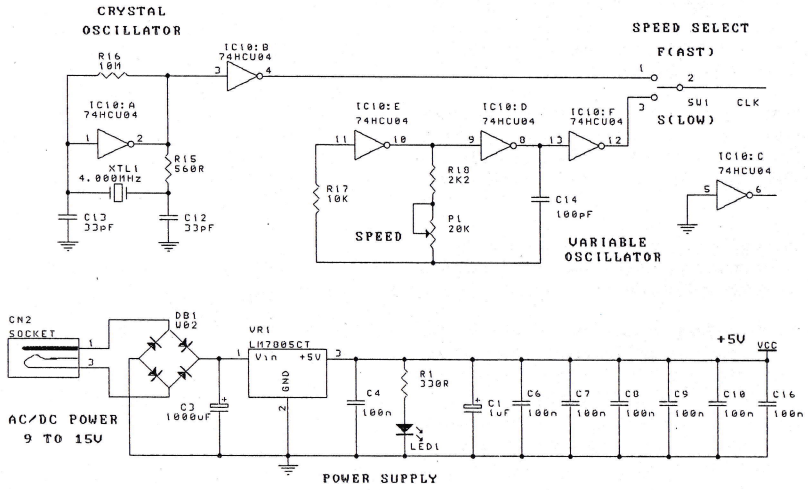

Figure 2. Power Supply Unit & Oscillator Schematic.

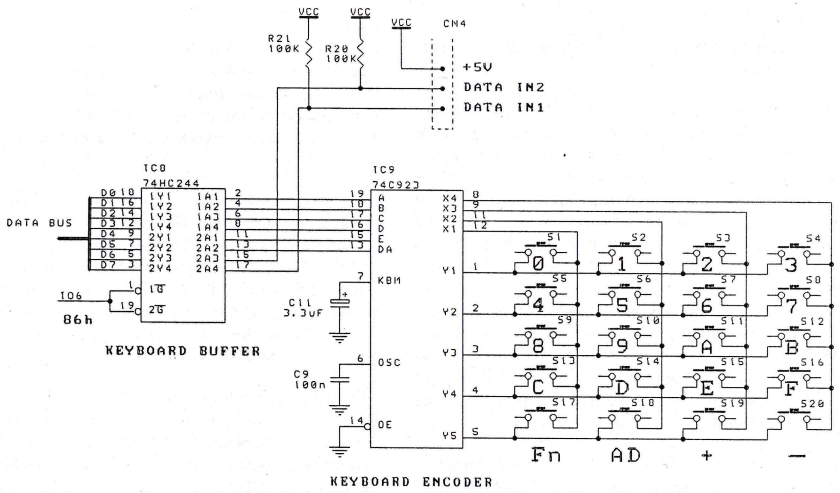

Figure 3. Input Interface.

switches. The 923 continuously looks for a keypress. When one is detected it encodes the key into a 5 bit number and a status out bit (Data Available) is set high. The Monitor constantly looks to see if this bit is set. Two capacitors are connected to the 923. C9 sets the speed at which the chip scans the keyswitch matrix looking for a keypress. C11 sets the time the 923 waits for the keypress to settle before reading it. IC8 is a buffer IC used to interface the 923 to the data bus. The data bus is a shared bus between several chips so data to go on the bus from the 923 must be held ready to be put on it. The Z80 controls IC8 through the I/O address decoder chip IC3. We will say more about this below.

The keyboard buffer IC8 has 2 unused input lines. These have been taken to CN4 where they are available for other uses that we will discuss later.

Note that in circuit diagrams like these it is usual not to show the connections of the power and ground pins of a chip. These connections are assumed. When Vcc and

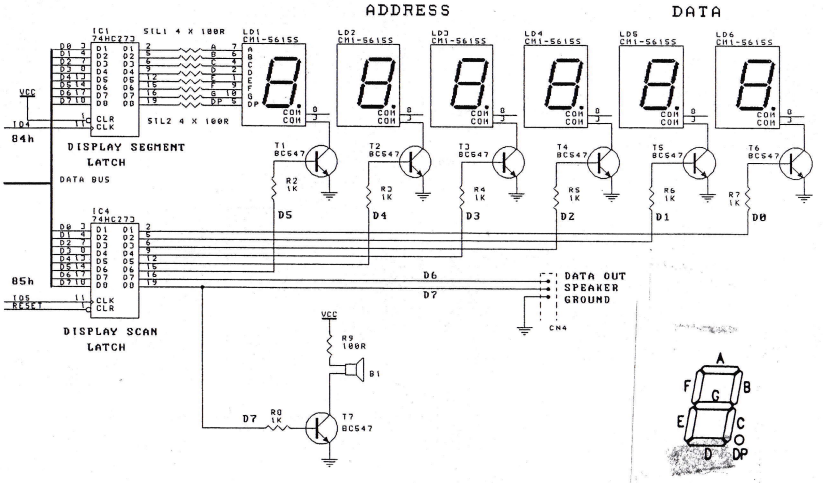

Figure 4. Output Interface.

GND connections are shown they are usually pull up or down connections.

Figure 4 shows the **Output Interface** of the Southern Cross. It consists of 6 seven segment common cathode LED displays and a one-bit speaker. Latch IC1 drives the display segments via 2 resistor networks SIL1 & 2. These resistors limit the current to the displays and prevent them from burning out. (In normal use as multiplexed displays these resistors would not be necessary, however, in the next Chapter we will be turning on the individual LED segments continuously. Thus current limiting resistors are desirable.) The definition of the segments from **a** to **g** and the **dp** (decimal point) are shown. The segments are connected to bits 0 to 7 respectively. Latch IC4 drives the common cathodes of each display as well as the speaker via 7 NPN transistors. Resistors limit the current into the base of the transistors as well as the speaker.

IC1 holds the data which lights each of the segments which makes up a particular number in a display. The scan latch IC4 is used to turn on each display in turn using the transistors T1 - T6 as switches. Pin 1 of IC4 (clear input) is connected to RESET. When the computer is RESET the latch is cleared and nothing appears on any display during the RESET time.Both IC1 & IC4 are controlled by the I/O decoder chip IC3.

IC4 has one unused output line. This is taken to CN4. The speaker output bit is also taken to CN4 as well as the two power lines.

We have left the core building blocks of the Southern Cross until last. Figure 5 shows the Z80, the I/O address decoder, memory decoder, RAM, EPROM, reset circuit and the connections of the computer to the 16 and 40 pin IDC's. Let us start from the left. The 40 pin IDC is the main expansion connector. It takes out all the pins of the Z80 for expansion projects too large for the other expansion sockets.

The reset circuit is simple but robust. The RESET pulse is active low and resets the Z80 CPU, the display scan latch, and is passed off the main board via the 16 and 40 pin IDC's. S21 is a manual reset switch. C15 is part of the power-on reset circuit. It pulls the reset line to ground immediately power is applied to the board. C15 charges up via R19 and the line goes high (and the reset is removed) after several milliseconds.

The Z80 chip comes next. Some specific information about the Z80, its operation and instruction set are given in the Technical Manual. Each device that the Z80 talks to must be connected to the three buses; the address bus, the data bus and the control bus. Not all the individual circuits in each bus may be used.

Memory Decoding. The full 64K address space of the Z80 is decoded into 8 blocks of 8K. This is done by the memory decoder IC2. This is a 3 to 8 line decoder. Three address lines A13, 14 & 15 are used as input. Each of the eight decoded lines becomes a CHIP SELECT signal for the memory chips it is connected to. Only two are used here; the EPROM from 0000h to 1FFFh and the RAM from 2000h to 3FFFh. See the data sheet in the Technical Manual for details about IC2.

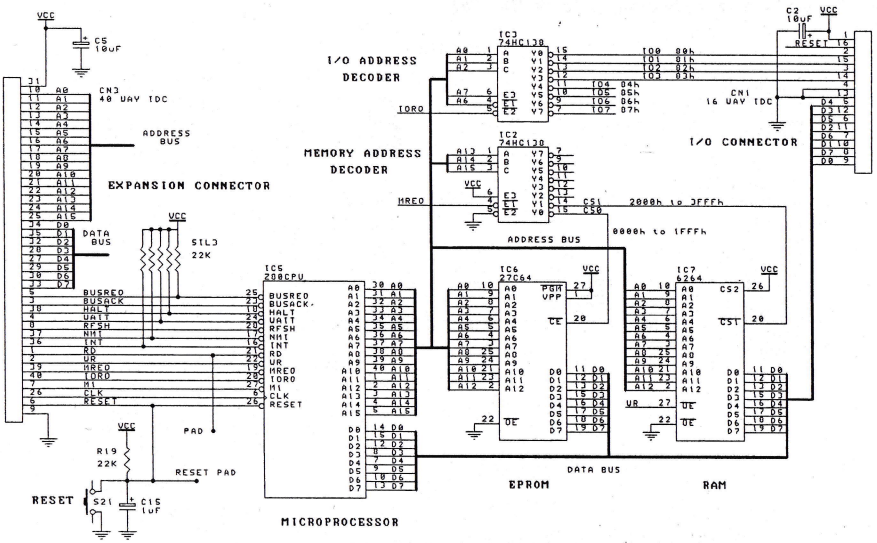

Figure 5. Core Southern Cross Components.

I/O Decoding. Each I/O device needs one I/O port address for itself. To get this unique address we need to decode one from the 256 I/O addresses provided by the Z80. This is how we have organised the decoding: the connection of address line 7, A7, to active high enable on IC3 divides the memory map into two halves. If A7 is low the decoder is disabled and no I/O ports on the Southern Cross are selected. This means that every I/O port selected must be from 80h to FFh.

The upper half of this memory map is further divided in half by address line 6, A6, connected to active low. Thus 64 locations from 80h to BFh are available to the Southern Cross. If A6 is high then a quarter of the address space from C0h to FFh are available for use by other devices. To get eight I/O ports from this 64 block address lines 0, 1 & 2 are used as input to a 3-to 8 decoder as described above for memory decoding. You can see the 8 decoded ports 80h to 87h on the diagram. Ports 80h to 83h are taken to the expansion port. Ports 84 and 85 communicate with the displays; port 86 connects to the keyboard. Port 87h is used in Chapter 7 for single stepping through programs to find errors.. Since A3, A4 and A5 are not decoded this means that there are 7 copies of each of the first 8 decoded ports to be found in the remaining 56 ports of the decoded port space. Each is 8 addresses higher.

The connections of the EPROM and RAM are standard.

Finally note that all chips on the Southern Cross are low power cmos versions.

- - - - - - - - - - - - - - - - - -

Summary of IC Function.

IC1,4	74HC273. Octal D flip flops with Clear.
IC2,3	74HC138. 3-to-8 line decoder.
IC5	Z84, cmos Z80 CPU, 4 or 6 MHz.
IC6	27C64. 8x8K EPROM.
IC7	6264. Random access memory.
IC8	74HC244. Octal tristate buffer.
IC9	74C923. 20 key encoder.
IC10	74HCU04. Unbuffered Hex Inverter.

'Octal' means there are eight of them on the one integrated circuit chip. 'Hex' means there are six of them. It does not stand for hexadecimal in this case.

- - - - - - - - - - - - - - - - - -

CHAPTER 4

A TOUR OF THE SOUTHERN CROSS

The Southern Cross has been constructed and when you apply power, press RESET, it goes beep-beep and '2000' appears in the ADDRESS displays. An undefined number appears in the DATA displays. The Southern Cross is ready. Check that the SPEED switch is across the F(ast) pins.

Please follow through EACH of the following steps. Do not be tempted to skip some because they look too easy. So much of what comes later depends on a full understanding of the concepts reviewed earlier. Nothing is reviewed which has no use later.

I. First Steps

The Southern Cross starts up at address 2000. The contents of address 2000 are shown in the Data displays.

Let us see what the keyswitches do.

- Keys 0 through to F enter digits of the base 16, or hexadecimal, numbering system. In computing they represent all 16 possible combinations of 4 digit digital (base 2) numbers from 0000 through to 1111. The table printed on the overlay to the right of the keypad shows you the full sequence of 4 digit digital number combinations which each hex number represents. Press some of the hex values and note what happens in the displays. The number pressed enters the display on the right and pushes the left-most digit out.If you are unfamiliar with hex digits then note in particular the way hex B and D are represented in the LED displays. Do not confuse a B with a 6.
- Press +. The current Address increases by 1. If you keep the + key depressed then the increment takes place automatically.
- Press -. The opposite to the + key occurs. The current Address decreases by 1. Play with the + & - keys.
- Press AD. The switch AD is a toggle. It allows hex digits entered on the keypad to go into either the Address or the Data displays. It switches back & forth between the Address and Data displays. The decimal points show where the switch is point to. For example, if you want to put byte 3F into Address 23EF press the AD switch so the 4 decimal points on the Address display are on. Enter 23EF. Press the AD switch so the keypad accesses the Data display. Now enter 3F. Address 23EF now contains 3F.
- Pressing RESET resets the computer.
- The Function key. By itself the key does nothing. It is used together with another key (or keys) to do things which the Monitor program has been programmed to do. For example, run programs. The next section gives examples.

To demonstrate the capability of the Southern Cross Function 8, Fn 9, and Fn C have been programmed to do things.

Function 8. Press RESET and then the Fn (Function) key. Now press 8. You have pressed 'Function 8'. A tune should start to play and it will continue to play until RESET is pressed. Press RESET.

Function 9. Now press Fn 9. That is, press the Fn key then 9. Another tune will play. Press RESET to stop it and reset the computer. Put the SPEED switch on S(low). Now press Fn 9. No prizes for guessing that the tune now plays much slower. Move the SPEED trimpot position. The speed of the tune changes.

Function B. If you do not like the beep each time a key is pressed you can toggle it off with Fn B. When the beep is off the displays will turn off briefly to indicate that a key has been pressed. This turn off time can be adjusted in software from zero (no dimming) to about half a second.

Function C. Put the switch back to F(ast). Now press Fn C. That is, press the Fn key then key C. This runs a game called Secret Number. The computer has generated a random, 4 digit hex number. You have 20 tries to guess what the number is. Enter your guess and press the AD key. The number of attempts you have made briefly flashes in the Data displays then the computer evaluates your guess and tells you how good it was.

The left of the two Data display (LD5) tells you how many numbers you have correct. The right Data display (LD6) tells you how many you have correct but are in the wrong place. Pressing the Fn key will take you out of the game.

This game will certainly help to stop you thinking in decimal and start to think in hexadecimal. In 20 guesses it is possible to find out what the random hex number is. (Our best number of guesses is 9.) Try to work out the system before you look at Section V below for the solution.

II. Review.

These first steps should have familiarised you with the 21 keyswitches of the Southern Cross.

Some questions may have occurred to you:

- what tells the computer to play a tune when Fn 9 is pressed?
- what tells the computer that pressing the plastic keyswitch labelled '4' registers '4' or '0100' with the computer?
- why does the computer start with '2000 xx' when it is RESET?

The answer to these and similar questions is to be found in full in the Monitor. The Monitor is an operating system which has been written so that the lifeless IC's, PCB and passive components which make up the Southern Cross can actually do something - that is, be a computer.

The Monitor is to be found in the EPROM IC. The EPROM is numbered IC6 on the overlay. It is a 27C64 IC and it is situated next to the Z80 CPU. A full listing of the Monitor code with comments is given in a text file on the floppy disk(SCMV1_2.PRN.) Please print it out using your work processor. Turn to the part which deals with Secret Number for example. Or find the part of the code which defines what each keypress stands for. The Monitor for the Southern Cross has deliberately been kept as simple as possible. You will not find programming tricks and shortcuts in it because its purpose is to teach, not to impress. Chapter 7 deals with the Monitor and how to use it in detail.

Another question you may ask is why does the Monitor program in the EPROM run when you hit RESET? The answer is that the Z80 is made so that this happens. Look at the information about the Z80 in the Technical Manual for information about this.

III. Review of Hex, Digital and Decimal notation.

It is essential that everyone understands the hex (hexadecimal) and binary numbering systems and why these two numbering systems are universally used in modern computing. The computer uses binary numbers 0 and 1 for memory, addresses and instructions. However, binary strings are error prone and unwieldy to use by people. Let us prove this so you understand exactly why we use hex.

Here are two columns of four 8 bit numbers. The numbers in each column are identical except for a one bit difference in one of the numbers. How long does it take you to spot this difference when the numbers are written in binary?

00101110	00101110
10101100	10101100
11001010	11011010
01101011	01101011

Errors do not show up easily. If we now look at the two columns expressed in hex the difference is much more obvious.

2E	2E
AC	AC
CA	DA
6D	6D

So it can be seen that hexadecimal numbers, numbers to the base 16, are a very convenient way of representing these binary patterns. The table on the right of the keypad on the overlay of the Southern Cross gives a full listing of all hex digits and the binary pattern each one represents.

To express a binary digit in hex, the binary bits are arranged in groups of four starting from the right. The hex value for each group is then assigned. It is usual to number binary digits from the right starting at zero. Thus the DATA displays are numbered D3-D0 for the rightmost hex digit (the low nibble as it is sometimes called), and D7-D4 for the high nibble. The ADDRESS displays which display values which are 16 bits wide are marked A15-A12, A11-A8, A7-A4 & A3-A0.

The ability to think in hex and picture the equivalent binary representation is a skill which should be mastered. You will be given a good chance to start practicing in the next section.

It is usual in texts like this not to keep pointing out when a number is a hex number and when it is decimal. The sense in which the number is used tells the reader what is meant. For example, if we say that the address 2231 contains 11, it is accepted that we are talking hex digits and not decimal. Of course, an address 2AEF which contains FF is obviously in hex.

Note that there are other numbering systems which are used in computing but none of them will be used here (eg, BCD, gray code, octal.)

IV. More Steps

1. Run a Program in EPROM with your own Data.

Function B. In the first section we showed that Function 8 (Fn key followed by pressing 8) played a tune. Function 9 did the same. The program and the notes of the tune are already in the computer. They are in the Monitor program which is in the EPROM. Function 8 and Function 9 have been programmed so that when they are pressed the tune will play. In both cases the program to run the tune and the tune itself is already written in the EPROM. Function B lets you run the tune program but you write your own notes. This is how it works.

In the Monitor program a musical Note Table going up 2 octaves has been defined as shown on the next page.

In the Table hex pair 01 plays a G, 02 plays G sharp etc. Enter a tune of your choice starting from address 2000. For example, press RESET. Address 2000 is shown in the address display and the contents of address 2000 may be some random byte. The decimal points in the Data displays should be on. If they are not then press the AD key to toggle back to the Data displays. Press 02. Press +. Address 2001 is now ready to receive a note table value. Press 08. Press +.

G	01	C	06	F	0B	A#	10	D#	15
G#	02	C#	07	F#	0C	B	11	E	16
A	03	D	08	G	0D	C	12	F	17
A#	04	D#	09	G#	0E	C#	13	F#	18
B	05	E	0A	A	0F	D	14	Rest	00

Continue 07 + 0A + 14 + 14 + 14 + 15 + 10 .... When you want to hear what you have written enter 1F instead of a byte from the Note Table. Press RESET. Then press Function A. (Fn key followed by A.) The tune will play then stop. If you would like the tune to repeat itself over and over until you stop it with RESET then change the contents of the address containing 1F to 1E. How do you do that?

Press down the + key and review the contents of the addresses from 2000 upwards until you find the address containing 1F. If you go passed it press the - key to go back. When you find say address 202A contains 1F then check that the 2 decimal points of the Data displays are lit. If they are not press the AD toggle. Now enter 1E, press RESET and then press Function A. The tune will play once then stop.

What you have done in this exercise is run a program which has already been written in the EPROM (can you find it in the Monitor listing?) but which used data that you entered yourself into RAM at address 2000 and up. The data has been terminated with a 1E or 1F depending on what you wanted to do. It is now time to enter a complete program yourself and run it.

2. Running your own program.

Enter the following:

RESET 3E + 4F + D3 + 84 + 3E + 21 + D3 + 85 + 76
RESET Fn 0

(This means press the RESET key then the 3 key then the E key then the + key then the 4 key then .. etc. The last three keys to press are the RESET key then the Function key then the 0 key.)

If you did it correctly then the displays LD1 and LD6 should both display the digit '3'. All the other displays will be blank. You have just entered and run a complete program.

If the program did not do this then you must find out why. This is called 'debugging' the program. Press RESET. Address 2000 should be displayed. In the Data display you should see 3E. Press +. The Address will now show 2001 and the Data should show 4F. Proceed in this way. Address 2002 should contain D3, 2003 84, 2004 3E, 2005 21, 2006 D3, 2007 85, 2008 76. If all are correct press RESET then the Fn key then key 0.

Note that although we write the hex digits as capital letters the B and D are displayed as small b and d since this is the easiest way to show them.

Function 0 causes the computer to run the program starting at the address shown in the address displays. In this case at address 2000. When in future we say 'run' the program we mean return to address 2000, press the Function key then key 0.

Let us first rewrite the main program in another way. Look at Table 1 below.

There are 4 parts to what we have written. On the right after the ';' are comments about what is happening in the code on that line. Next to the comments are the Z80 mnemonic codes for the program we are entering. Study the codes with the comments. It is pretty obvious what the codes mean. We are going to leave studing the Z80 codes in detail for another Chapter.

On the very left is the address that we started out program at (2000) and the addresses that the bytes of program code can be found. For example, if you key in the address 2008 directly on the Address display then you will find 76 in the Data display. (Remember how to do this? Toggle the AD key so the 4 decimal points are on then enter the address 2008 on the hexpad.) Next to the addresses are the hex bytes which represent the machine language code which you enter into the computer to run the program. These four parts to the program - address, machine code, source code and comments - are called 'fields'. There is actually another field in this simple program which we have not used in this example. This is a field in the centre which is used for labels. All this will very quickly become familiar to you.

Before going on to the final program change the bytes at address 2001 and at 2005. (Then RESET and run the

```
2000    3E 4F       LD A,4F          ;load the accumulator register A with value 4F
2002    D3 84       OUT (84),A       ;output that value to port 84
2004    3E 21       LD A,21          ;load the A register with 21
2006    D3 85       OUT (85),A       ;and output to port 85
2008    76          HALT             ;that's all, now stop
```

Table 1.

program by hitting Fn 0.) Play around with setting them to different values. Work up from low values 01 02 03 etc. You will see a pattern as you increase the values up from 01. Try to work out why?

3. Running your second program.

This time we have two parts to the program, one at address 2000 (the address we go to automatically when RESET is pressed.) and a second program (a delay program) which we will put at address 2100. See Table 2 below. The delay program is called a subroutine. It is called by the main program when it wants to have a delay during the running of the main program. Here is the printout in a similar form to that given above. Push RESET to bring you to address 2000. The code for you to enter is in the second field; enter 3E at 2000 then press +. Enter 0F at 2001 & press +, etc. You can check yourself all the time by looking at the address and the data byte being entered to see they correspond with the program. All five fields have been used in this program.

The only tricky part of the program is to enter the delay program at **address 2100.** Remember to use the AD toggle. After you have entered the first part of the program hit AD to toggle to the Address display. Then enter 2100. You are now at address 2100. Press AD again and you are ready to enter data.

Run the program. See if you can follow what is happening. Too fast? Well hit RESET and move the SPEED switch from F(ast) to S(low). Hit RESET then Function 0. (Note because you are on Slow you will have to wait up to a second after hitting the RESET, Function key and 0 keys.) Now things are really slowed down. You can use the potentiometer to also vary the speed. Note that you are using a hardware delay together with a software delay!

Now at this much slower speed try to work out what the program does.You can change some values to see their effect. Vary the number of displays by changing the contents of address 2001, 01 for 1 display up to 3F (or higher) for all six displays. Vary the delay time by changing the contents of address 2102.

VI. Solution to the 'Secret Number' Game.

You must have a system to find the solution. For the first guess enter 0000 and press AD. Then 1111, then 2222, 3333, etc. When you get an answer like '10' then you know that one and only one of the four digits entered was in the correct position. Suppose you had entered 3333 to get '10'. So next enter 3444. At least you will get '01'. If you get '01' then the 3 is still correct of course but its correct position is not in the leftmost position. Also there are no 4's, so enter 5355. Do you see the system? We are moving the 3 along to find its right position while simultaneously testing the other digits .

You must modify the system depending on the response. Enough has been now given for you to go on to determine the number. It gets tricky when there are two numbers the same! You have to recognise the response. Keep playing and you will quickly be thinking in hex and nested loops.

- -

```
2000   3E 0F          LD A,0F       ;load 0F into accumulator
2002   D3 85          OUT (85),A    ;scan 0F displays
2004   06 00          LD B,00       ;load zero into register B
2006   78      LOOP1  LD A,B        ;put contents B into accumulator
2007   D3 84          OUT (84),A    ;output A to segment latch
2009   04             INC B         ;increase value in B by 1
200A   CD 00 21       CALL DELAY    ;go to delay subroutine at 2100
200D   C3 06 20       JP LOOP1      ;unconditional jump to2006

2100   11 FF 60 DELAY LD DE,60FF    ;load 60FF into DE registers. Note address 2100
2103   1B       LOOP2 DEC DE        ;decrement DE by 1
2104   7B             D A,E         ;load contents of E reg into A
2105   B2             OR D          ;logically OR A with D
2106   C2 03 21       JP NZ, LOOP2  ;if not 0 jump to address 2103
2109   C9             RET           ;when 0 return to main program
```

Table 2.

CHAPTER 5

FIRST PROGRAMS

We are going to go back and reexamine the two programs you entered and ran in the previous chapter. But this time we will discuss them in more detail.

Both programs output information to two ports, port 84 and port 85. Port 84 accesses the individual segments within the LED displays. Port 85 accesses the individual LED displays themselves, as well as the buzzer. A latch chip is used to provide an interface between the computer and both ports. A latching chip retains its logic states even when the input which caused the change of state is removed. IC1 latches the display segments; IC4 latches the displays themselves.

It will help here to look again at the schematic diagram of the displays. Note how the eight data lines service both the individual segments within a display as well as the displays themselves.

Port 84. Display Segments

The diagram of the LED segments which make up a display shows the A, B, C, D, E, F, G, DP designations of the individual display segments. These correspond to the eight data lines D0 to D7 respectively. Why is there this correspondence? Because that is how we physically designed the tracks on the PCB. The schematic diagram shows this. For example, if D1 and D2 are high then segments B and C are lit up. This makes the number 1 on the display.

Suppose we want to display the capital letter 'C'. We must place highs on segments A D E and F to turn on these segments. This corresponds to highs on data lines D0 D3 D4 D5. That is, the data bus must be 00111001. (Remember we go D7 D6 D5 D4 D3 D2 D1 D0). But as we saw in the previous Chapter dealing in binary is very difficult. So convert it to hex. 0011 is 3 in hex; 1001 is 9. So to display the letter 'C' output 39 to port 84. This can be written 39h so there is no ambiguity.

Let us do one more. Say we want the number '3'. We want segments A B C D and G turned on. The byte to put on the data bus is (can you do it in your head) 0100 1111, or 4F.

Now look at the first program you entered in the last Chapter - reprinted on the next page. At address 2001 we are loading 4F into the A register of the Z80. At 2002 we tell the Z80 to output 4F to port 84. So now you know why the number 3 is displayed when you run this program.

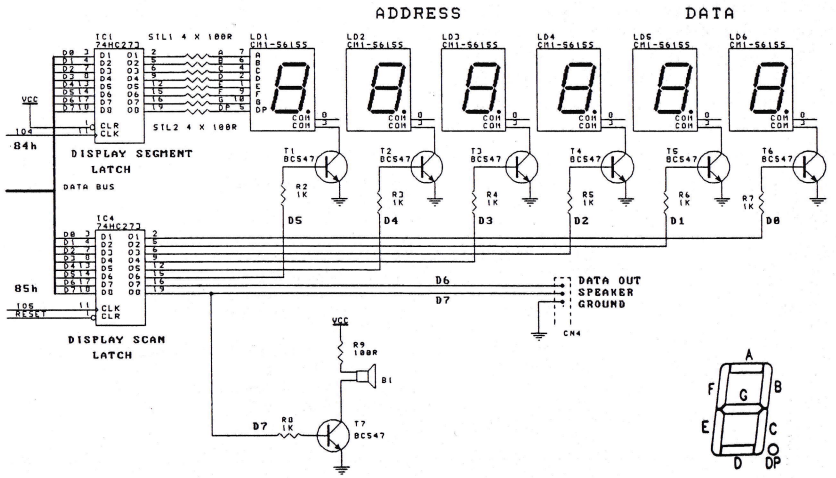

Figure 6.

Port 85. Display Scan

But, you say, the number 3 is displayed on only two of the displays and not the other four; why is that? An examination of the schematic will tell you why. D0 is connected to LD6, D1 to LD5 and so on up to D5 to LD1. So a high on D0 and D2 will turn on the 2 displays they run to. And by the analysis in the previous section a byte on the data bus of xx00 0101 will do this. That is, 05. Since x is a don't care bit there are four other values which will do the same thing. xx can be 00 01 10 or 11, so you can work out what the values are.

Segment	Data bus	
A	D0	A high on D0 turns on segment A
B	D1	A high on D1 turns on segment B
C	D2	" D2 " C
D	D3	" D3 " D
E	D4	" D4 " E
F	D5	" D5 " F
G	D6	" D6 " G
DP	D7	" D7 turns on decimal point

Table 3.

In our first program reprinted below address 2005 contains 21. In binary that is 0010 0001. So it should be apparent to you why the LD6 and LD1 displays show the number 3.

Z80 Instructions

Instructions like LD, OUT, HALT are mnemonic assembly language representations of Z80 instructions. They are easily understandable symbolic representations of the actual binary machine code of the instruction. Remember we showed in the previous Chapter how hard it is to work with binary numbers directly. For binary addresses and bytes we convert the binary into hex and work with the hex. For binary instructions like LOAD the A register with a byte, the hex of the binary code for LOAD (3E in this case) is still meaningless. It is easier to work with a mnemonic - an abbreviation which give a good hint as to what the instruction does. LD is easily remembered as standing for LOAD; OUT and HALT are immediately obvious, INC, OR, JP, CALL, JP and DEC (see next program) are also easy to remember.

By working through the program examples in the next few Chapters start to learn the assembly language mnemonics. When you write your own programs it is these symbolic instructions you will use.

Initially you will hand assemble your programs. That means you will convert the mnemonic instruction to its hex equivalent by hand. A complete listing of the Z80 codes is supplied with this Kit to enable you to do this. Later on when we look at bigger programs you will use an Assembler program & a PC to do this conversion for you. The floppy disk contains one of these programs to run on PC's using DOS.

Working through the Second Program

The second program presented in the previous Chapter and reprinted at the top of the next page, is an extension of the first program. The first two lines turn on 1 or more displays depending on the byte at address 2001. We have loaded 00001111 which turns on the four right displays. Then the rest of the program counts from 00 to FF and outputs the number to the displays which are turned on. Thus displays cycle through all combinations of seven segments plus the decimal point. When all segments are turned on (at FF) the program restarts at 00. Watch the program at a very slow speed and try to predict the next segment to turn on.

The delay subroutine is a standard delay. The byte at 2102 is the critical value to set the delay. The byte at 2101 is less critical. Do not worry to much if you cannot understand exactly how the delay works. It sets a number then decreases it by 1. When the number gets to zero control returns to the main program. A 'tricky' but commonly used way test for zero.

More programs.

1. Back and Forth

Enter this program in Table 6 and try to work out why it does what it does. What you should see is a pattern bouncing back and forth along the six displays.

2. Back & Forth - all Segment Combinations

A small modification can cycle all combinations of the display segments as the display oscillates back & forth. Add the code in Table 7 at address 2080

Change the byte at 2025 to 82 and run the program. To start the cycle at zero you can change (2001) to 00 or FF. (Putting '2000' in brackets means the 'contents of address 2000'.) You can play with the delay at (2102) as well.

The extra code counts from 00 to FF and each time puts the result on the data bus to be latched by the display segment latch.

```
2000  3E 4F    LD A,4F       ;load the accumulator register A with value 4F
2002  D3 84     OUT (84),A    ;output that value to port 84
2004  3E 21     LD A,21       ;load the A register with 21
2006  D3 85     OUT (85),A    ;and output to port 85
2008  76        HALT          ;that's all, now stop
```

Table 4. First Program.