G P INDUSTRIAL ELECTRONICS LTD.
Skardon Place, North Hill
Plymouth PL4 8HA Tel. (0752) 28627

PROLOGUE - WHAT THE MANUAL COVERS.

This manual deals with the construction and use of SOFTY which is a bench-tool for the system designer. A professional level of understanding is expected and assumed.

No apology is offered to the beginner about the absence of sections on the aspects of hardware and software design of microsystems in general. There are no explanations of Computer Architecture, Boolean Algebra or even Hexadecimal Arithmetic and there is nothing about programming MPUs in general or even about the INS 8060 in particular. Instead a list of useful publications is appended.

SOFTY is about hardware and software design rather than about programming. The world has many programmers but very few designers. If you find these contents baffling take heart: all the MPU does is to turn bits on and off. Meanings are often obscured in esoteric language but the concepts behind the words are always simple. Each instruction causes bits in registers to change in a straightforward and obvious way. Complex functions are performed by sequences of such simple instructions and that is all that programming is about.

SOFTY - DEFINITION.

SOFTY is a small computer which is designed to be a complete and useful tool in its own right. It is not a minimal system to which memory cards and peripherals should be added. SOFTY has all the peripherals it needs and it would not be easy to add extra memory in addressing space.

SOFTY already has a Visual Display Unit which makes internal operation totally transparent. It also has a cassette interface for storing programs and an EPROM programmer for making programs a permanent part of any microsystem. It has two eight bit I/O ports which are capable of a wide variety of data transfers and a serial I/O interface. SOFTY has a keyboard for entering machine language programs and a set of assembler functions for manipulating memory contents. Once a program has been written SOFTY may action it using the internal MPU or offer it to an external MPU for action. SOFTY is therefore a universal program development system.

SOFTY is not designed to support a full-scale programming language (like BASIC). It cannot directly be used to balance your household accounts or play intergalactic snakes and ladders.

SOFTY is for the development of new PRODUCTS which are also microsystems. These new products themselves might easily be accounting machines or videogames. (In fact SOFTY was used to develop a TV chess game and is currently working on a Space Raiders type of TV game)

The keyboard functions themselves form a special high level language which interprets the requirements of the system designer.

SOFTY AS A CONTROLLER.

Most MPUs will not be used in formal computers which have alphanumeric keyboards and VDUs. The majority will be used in communication and control systems. Most programs will be developed in machine-code or assembler-language and contain relatively few instructions. When such controllers are produced in large numbers they will be in the form of single chip computers which incorporate the program in masked ROM. Low volume applications will use EPROM for the program because the numbers of units would not justify a masking charge.

SOFTY can be used by itself as a controller. It can also be used as a development-system for an INS 8060 three chip microcomputer. The three chips would be 2708 (or 2716 if 2K of program memory was required) INS 8060 and INS 8154. The cost of such a control computer would be less than £30 in quite small quantities. SOFTY is such a minimal system with a VDU, EPROM programmer, Keyboard etc added

SOFTY has a microcycle time of exactly one microsecond, controlled by a crystal oscillator. The INS 8060 has an on-chip programmable timer which will count microsecond increments and therefore provide a real-time clock to synchronize events. It is possible to calculate the execution time of any program EXACTLY TO THE MICROSECOND, provided that it does not depend on random inputs.

When supplied SOFTY contains firmware in the form of a 1K x 8 EPROM TYPE 2708. The firmware forms a useful monitor, editor and assembler for the development of programs. There is no reason to retain this firmware in many control applications. SOFTY can be reprogrammed to do only those tasks which are required.

A program in EPROM may also be run from the program socket. This permits the retention of useful programs in EPROM which may be plugged into the system when required.

Many novices believe that their particular application requires a 'powerful' processor which has a complex instruction set. This is often untrue. The powerful processors have limited application because they are not versatile. Simple processors will have real benefits, such as control or flag lines and on-chip clock, and the INS 8060 is often more cost-effective than the Z80. The only circumstance where a powerful processor will be advantageous is one which will use a lot of memory, and the additional complexity of the hardware can be offset against memory costs.

Actual sales of MPUs reflect how readily they may be incorporated into real products: a popularity poll would produce quite different favourites.

CONSTRUCTION.

Looking at the top of the board which has the component positions printed, every pad visible requires a link to make contact between tracks on the upper and lower sides. A wire link should be inserted into each of these holes, soldered and trimmed.

Next the switches should be installed. Clean the pins by scraping to ensure a reliable soldered joint and insert them into the board. Take care not to press on the button part of the switch because this may damage the domed disc inside.

Do not do any soldering on the lower side of the board until all the components and sockets have been placed correctly with their leads bent and trimmed. Take particular care to orient transistors, diodes and capacitors correctly. Solder the components from the lower side and the pin-throughs all at the same time. This way you are less likely to miss any joints.

The only holes which will not contain components are the two dotted wire links for the cassette interface and the EPROM selection. Use of these will be explained in the text and they may be left until the end of the testing stage.

When SOFTY is to be used by itself, and in any case to test the device, the lines NCARD and NENIN should be connected to ground.

If SOFTY is to be used for program development in an external microprocessor system the address and data lines should be connected via the card edge and not via the EPROM socket. Therefore the ribbon cable should be connected to a 43- way edge connector. Remember to connect a common ground line, NCARD, NENIN and NBREQ in some appropriate method to control bus access. The 24 pin EPROM-type plug is for the non-SOFTY end of the umbilical cord: this plug replaces the 2708 or 2716 EPROM device which SOFTY will later provide.

If SOFTY is to be used only as an EPROM programmer for your system then you should devise a serial or parallel link between your computer and SOFTY. The serial link can be via the serial input(DIN socket) or any bit of either port. The parallel link can be to port A with the handshake signals going to the top two lines of port B. Consult manufacturers data sheet on the INS 8154 I/O port. The actual routine which performs the transaction can be run in SCRATCHPAD or programmed into an EPROM which is placed in the program socket when needed. Manual entry of programs from hex listings is time consuming and prone to error.

Do not connect the program voltage at any time (other than during actual programming). Be careful with this 27 volt line which is capable of destroying every device on the board.

WHAT TO DO IF IT WILL NOT WORK.

1) Inspect the board with a magnifier in a strong light. Look for solder bridges between adjacent lines, dry joints and components inserted the wrong way round. This is by far your best chance of finding the fault.

2) Use an oscilloscope for this and the following tests. Check the power supply pins of each IC in turn. Any feedthrough of signal shows that the pin is unconnected.

3) Trace the divider chain through from the crystal. Assume that any line which is running between high and low levels is working correctly. Investigate any line which appears to be conflicting or appears to be open-circuit. Check any pin which is clamped at high or low levels to find a logical reason for this.

4) Investigate the address and data busses, first at the card edge and then at each device visited. Make sure that all lines are running and not shorted to each other or to one of the supply rails.

5) Start looking at the control signals at the MPU. Try to isolate the fault in one particular region by thinking about it logically.

6) Even if the fault is still present it is still almost certainly due to a solder bridge, a dry joint or a misplaced component.

SERVICE.

It could happen that you construct SOFTY with care and attention and yet it refuses to work. Naturally you will suspect the components which have been supplied so you should identify them with a mark before returning the SOFTY with the standard fee of £20.

The detection and correction of errors in assembly is a task which can be as time consuming as that of building the machine and in many cases replacement of the PCB is the only cure.

Any parts which have been spoiled will be charged also - but if it appears that a defective part was originally supplied the £20 will be refunded. Frankly this is a slight hope.

GETTING A PICTURE.

SOFTY will produce a picture on a VIDEO MONITOR or a TV SET. If a monitor is to be used the UHF modulator may be left off the board but a wire-link should be made in its place - the ground rail passes through it.

The modulator supplied will work with most sets of UK and European origin designed to receive 625 line transmissions - but not French second-series sets which require another pattern of modulator.

Tune a picture in channel 36. There may be pictures in other channels but these will be inferior. The left hand edge of SOFTY's picture is in the same place as the left hand edge of the test-card. If the TV width control is wrongly set it should be readjusted. A TV set which receives the SOFTY picture may be used for normal viewing without readjustment.

KEYBOARD CHECKS.

First check that all the keys work. On keying RESET you should see four shaded bands of hexadecimal characters and at least one of the lines of inverted video above them. Inverted video means white characters on a black background.

It can happen that the domed disc inside a keyswitch becomes displaced from its central position leading to unreliable entry. The cure is to remove the top of the switch and recentralize the disc. Once you are sure all the switches are working reliably the panel should be fitted and secured with bolts.

Keys 0 to F enter hexadecimal information into memory. There are also a series of functions which may be used after the SHIFT key has been pressed. The SHIFT key is next to the RESET key and is marked with upward and downward arrows denoting change of case. WHEN SOFTY IS AWAITING AN UPPERCASE (FUNCTION) KEYSTROKE THE MODE FLAG IN THE STATUS LINE CONTAINS FF.

HOW SOFTY WORKS: THE HARDWARE.

The on-chip oscillator of the INS 8060 is controlled at 4 MHZ by a crystal. This master clock frequency is divided down by LS and CMOS counters. Sync, blank and shade signals are derived and combined with serial hexform data to produce a composite video signal. The video signal is fed to a UHF modulator to produce an RF signal which will drive a TV set via the aerial socket.

The VDU divider chain also produces horizontal and vertical addresses which pass through a tri-state buffer on to the main address bus. This tri-state buffer is enabled by the NENOUT signal from the INS 8060 microprocessor. This allows the VDU to scan memory by placing signals on the address bus when the INS 8060 is not using the bus itself.

The eight data lines are connected to a Quad multiplexer which selects the high and low four bits in turn. These four outputs are used to address a PROM which is also addressed by line-select addresses from the VDU divider chain. The PROM converts the four binary inputs into a visual hexadecimal form. Each character so formed is three bits wide and another bit forms the space between characters. The PROM outputs are connected to a shift register and clocked out at 4 MHZ as serial hexform data.

As all eight data bits are used a ninth bit is added to memory to form the cursor. The cursor memory is a single bit 1K RAM chip. This is written to from a flag line of the MPU (F0). The MPU does not read from this memory device.

The screen memory (WORKING RAM) is a pair of 2114 chips giving 1K x 8. Not all of this can be presented on screen at the same time - each screen page shows the contents of 512 addresses. The screen page selected depends upon the position of the cursor.

SOFTY's own working program is contained in a 2708 EPROM. Although the contents of the firmware EPROM cannot be presented on-screen directly the firmware can be copied into working RAM by a keyboard instruction.

A programming socket for EPROM devices is also provided and data contents of devices can be read from this socket. In this case the contents can also be observed directly on-screen.

Also present in addressing space is an INS 8154 RAM I/O device which contains 128 bytes of RAM and two eight bit I/O ports. Bit A7 of the address bus which forms the memory-I/O select to the chip is inverted and this allows part of the chip's RAM to appear as inverted video at the top and bottom of the screen. This RAM is used as SCRATCHPAD by SOFTY for storage of routines and data which does not belong to the program under development.

The last line of the SCRATCHPAD which appears directly above the main screen contents is called the STATUS LINE. The particular significance of each location is explained in the diagram and these should be memorized or the TV screen labelled.

The keyboard is connected in a matrix across ports A and B. SOFTY would scan a matrix of 64 keys but only 20 are present The reset key is not part of the matrix: it is conditioned with an R/C combination and a couple of gates and used to reset the INS 8060 and the INS 8154.

Seven lines from port B and Flag 2 from the MPU are connected to an eight-input NAND gate which drives the CONT input of the INS 8060. When the MPU has no more work to do it returns to the keyscan routine which places flag 2 high. If no key is depressed all lines of port B will be high and the CONT input will be driven low. The MPU then hands the bus to the VDU scanner by placing NENOUT low.

The page of information to be scanned is derived from P1 (the cursor position pointer) and latched into the lowest two bits of port A. Two screen pages of EPROM and two of RAM can be selected in this way with parts of the SCRATCHPAD at the top and bottom of the screen.

The INS 8060 cannot reclaim the bus until a key is pressed which will drive CONT high. The keymatrix is then scanned and the particular function requested is actioned before the program returns to the keyscan routine and flag 2 is placed high again.

In applications which demand the use of ports A or B the keyboard presents no problems: if no key is pressed it may be ignored.

When the MPU is using the bus the addresses are not being scanned by the VDU in an orderly fashion and the picture will be blurred and meaningless. Whilst the action is in the keyscan subroutine as described above and a program is being entered the user has a clear view of memory contents.

Use of the PORTS A & B for serial and parallel data transfers and use of the Serial Input and the Serial Output will be covered in detail later.

THE EPROM PROGRAMMING SOCKET.

The program socket may contain an EPROM device of the 2708 or 2716 variety as a board-strap option. During programming the NHOLD input of the INS 8060 is driven low which 'freezes' the data and address busses whilst the program pulse is applied. A 555 timer determines the length of the program pulse and R/C combinations observe the correct set-up and hold times.

A high output on Flag 1 of the INS 8060 forms the program-enable. 2708s are programmed with F linked to E. For 2716s link E to D. F then becomes the most significant bit of the address and may be connected to Vcc or ground to select which half of the 2716 will be programmed. The effect of the link is to change which pin receives the program-enable input.

DO NOT ATTEMPT TO PROGRAM SINGLE RAIL DEVICES SUCH AS THE INTEL 2716.

The 27 volt programming voltage is connected at the point marked P at the bottom right corner of the keyboard. The program-voltage should not be left connected except when it is actually required to program an EPROM.

BEWARE that if the program voltage is connected alone; without the other voltage rails DESTRUCTION OF THE EPROM IS GUARANTEED.

Care should also be taken that the high voltage line does not contact any other devices on the board - this is why it is not brought to the card edge.

The EPROM socket is not only used for programming. It forms a useful extension for firmware because programs which have been placed in EPROM may be actioned from this socket.

KEYBOARD FUNCTIONS.

When power is first applied each RAM cell takes up an arbitrary state which results in a random display.

RESET zeros the internal registers of the INS 8060 and the INS 8154. The first instruction will then be fetch from location #0001. (The # sign will be used for a hexadecimal number) RESET may be used to terminate any program which is running.

The SHIFT key is on the left of the RESET key. It is used to select the functions which are described in the upper half of each keylabel. When a keystroke should be preceded by a SHIFT keystroke the $ symbol will be used in the text.

When SOFTY is awaiting an UPPERCASE instruction, immediately after the $ key is pressed, the byte FF will appear in the MODE FLAG of the STATUS LINE.

Pressing the $ key again will exit from the uppercase mode and at the same time remove any spurious cursor highlights.

The effect so far would be to give a screenful of random data with a single cursor at the top left corner.

$F (which means that the SHIFT key is struck and then the key marked F is struck) will now clear memory and the screen will be filled with FF bytes. The reason why an all-clear of highs (FF) is chosen rather than all lows (00) is that an unprogrammed EPROM is all-high. Therefore it permits the selective writing of programs into EPROM a section at a time. $F clears only those locations which are BEYOND the cursor.

Hexadecimal numbers may now be entered into memory sequentially from the keyboard. All the keys labelled 0 to F should be tried and correct operation confirmed.

CURSOR keys will move the cursor backwards and forwards in memory. The cursor will step if the key is held down and then jump a line at a time until the key is released.

NOTE that the first two digits in the STATUS LINE show the first two digits of the absolute cursor location, and the second two digits are the lower cursor address.

$0 is an exit from the SHIFT mode and is similar in effect to keying $ twice.

$1 stores the current cursor location for later return or the calculation of jumps and displacements. When the cursor is moved subsequently the hex difference between the two locations is shown in the penultimate byte of the STATUS LINE. The stored cursor address appears in bytes 3 & 4.

$2 permits the definition of a block of data by moving the cursor BACKWARDS. Start with the cursor on the last location of the intended block. The block may immediately afterwards be stored away in SCRATCHPAD or moved to a different part of memory. The limiting size for storage in SCRATCH is 108 bytes. For block shifts 127 bytes is the limit. This is why the screen is divided into shaded bands of 128 bytes: for most MPUs 128 bytes forwards or backwards is the limit of a single byte displacement of program-counter relative instructions. The shaded bands help in the estimation of whether a particular location is within reach or not.

$3 will permit the shifting of the block as defined by $2 forwards or backwards as directed with the cursor keys. This is a system for rearranging memory contents. Intervening data is not destroyed - it is shifted to the other side of the block. A single $ stroke will exit from this mode. NOTE that this function will only work in RAM - data in EPROM cannot be rearranged.

$4 lifts the defined block into SCRATCHPAD and stores it as a recursive subroutine. In effect this enables the user to program keyfunctions. For example a bipolar PROMburner, Teletype I/O, parallel interface to a Z80 system for which SOFTY was acting as EPROM programmer and a subroutine to convert hexadecimal into Baudot have all been fitted into SCRATCHPAD as keyfunctions. Incidentally a recursive subroutine is one which exits at its entry point and can therefore be called time and time again without reloading pointers. If it is possible to see the first line of inverted video below the screen the effect of $4 can be viewed directly and the mechanism of the recursive subroutine observed in action. NOTE that it IS POSSIBLE to lift sections out of EPROM using $4 but it is NOT POSSIBLE to block-shift out of EPROM using $3.

$5 writes back the block in SCRATCHPAD defined by $4 starting with the current location of the cursor. This process has NO EFFECT on the contents of SCRATCH and it may be repeated.

$6 hands control to a user program in the EPROM socket. The user can make a library of useful routines in EPROM which are placed in the EPROM socket when required and called by $6.

$7 transmits the contents of working RAM in serial form from the DIN socket in a TRANSWIFT transmission which may be stored on cassette tape. When finished the MODE FLAG contains an AA byte.

$8 samples the serial input for an incoming TRANSWIFT transmission. As received the data is written into screen RAM (the process is visible) and when complete the transmission is checked for validity and the checkword placed in the MODE FLAG. If the transmission is valid the byte in the MODE FLAG will be AA.

$9 copies back the contents of SOFTY'S own firmware on screen.

$A compares the contents of screen RAM with the contents of the EPROM in the programming socket. The number of differences will be counted in the MODE FLAG and any differing locations will be highlighted. One use for this function is to check that an EPROM is erased by keying $F then $A.

$B is the EPROM programming function. The 27 volt line should be connected at the point marked P at the bottom right corner of the keyboard first. On pressing $B the contents of RAM will be transferred to the EPROM. Afterwards the contents will be read and compared as described for $A. The correct linkage for 2708 or 2716 should be made as described elsewhere. During the 100 hit cycles prescribed for EPROM programming a counter is kept in the MODE FLAG. The process takes about 2 minutes per 2708.

$C copies the contents of EPROM into screen RAM.

$D is the matchbyte function. When followed by two hexadecimal digits all locations containing that particular byte will be highlighted. The matchbyte function can be repeated to find a particular section of code.

$E exchanges the contents of the stored cursor of $1 with the current cursor. This function is useful for making long range jumps and calculating the difference between locations as a hexadecimal number. It will be noted that the operation of $E will cause the penultimate byte in the STATUS LINE to complement.

$F is the all-clear function as previously explained.

RECOMMENDED READING.

PUBLISHER.	TITLE.
National Semiconductor.	Pub No 426305290-001C INS8060 Single-Chip 8-Bit N-Channel Microprocessor (SC/MP Family)
N.S.	SC/MP Technical Description.
N.S.	SC/MP Microprocessor Applications Handbook.
N.S.	SC/MP Microprocessor Assembly Language Programming Manual.
N.S.	INS8154 N-Channel 128-by-8 Bit RAM Input/Output (RAM I/O)
KEMITRON	GUIDE TO SC/MP PROGRAMMING by Dr. Drury.
ELEKTOR MAGAZINE.	Series 'Experimenting with SC/MP' Nov 77 to Mar 78 and other articles.

SERIAL INTERFACES.

The INS 8060 has pins dedicated to serial input and serial output which communicate directly with the Extension Register. The serial output line is latched. Serial communications may readily be handled by software.

Conditioning circuitry is interposed between the Sin and Sout pins and the DIN socket via which serial communications are handled. In the case of Sout a 10K resistor is used merely to protect the MPU pin against external high voltage transients. This resistor could be removed to drive TTL directly and replaced by a wire link.

The Sin line first passes through a capacitor and a 10K resistor (which could also be removed for specific applications which require signal reproduction down to d.c.) and then to the input of an EX-OR gate which is suspended in the centre of its transfer range by a 100K variable resistor from Vcc to Ground. This gate performs two functions: firstly it amplifies and squares up the input signal and gives some control of the mark/space ratio (using the VR) and secondly it provides an easy way of inverting the input when necessary. If the other input of the EX-OR gate is at Vcc the input will be inverted - if it is at ground the signal will pass unchanged. A dotted line will be found on the PCB marked A----B----C. If A is linked to B the input signal will be unchanged. If B is linked to C it will be inverted.

SOFTY is capable of handling serial transmissions at 110, 300, 600, 1200, 2400, 4800, 6400, or 9600 baud using a subroutine containing less than 35 bytes of code, but such routines do not form a part of the provided firmware. When required such programs should be run in EPROM or SCRATCHPAD, unless the user chooses to modify the firmware.

TRANSWIFT. THE CASSETTE INTERFACE.

In order to leave the Serial Lines free for other forms of synchronous and asynchronous data transmissions, such as a CUTS FSK tape interface, teleprinter I/O or a direct line to a larger computer, a method of storing data on tape has been devised which entirely software.

TRANSWIFT is a software modem. Patent application has been made for this novel method of serial data transmission. The name TRANSWIFT describes a preferred form of data transmission in which each bit of a succession of binary bits is conveyed as the interval of time between two successive changes in magnitude. These intervals are predetermined by whether the bit has a value 0 or 1. Inversion of the waveform has no effect upon the essential succession of intervals. TRANSWIFT has no clock and cannot be regarded as either synchronous or asynchronous.

Recovery of data is performed by squaring the waveform at its points of transit across an intermediate voltage level to produce a serial waveform of absolute binary values. This waveform is sampled at a STEP interval from each positive (or negative) transition and the value of the STEP may be calculated automatically from the incoming waveform.

In effect TRANSWIFT solves many problems with regard to data transmissions in limited bandwidth media: it is relatively insensitive to changes in speed and level and is therefore less prone to the effects of 'dropouts'; it will run at a data transmission speed which is close to the upper frequency limit of the band and virtually no hardware is required. The limiting factor in speed is the ability of the processor to sample and store the incoming data, rather than the limit imposed by bandwidth in cassette storage applications.

TRANSWIFT - SETTING UP THE CASSETTE INTERFACE.

The DIN socket is wired to tape recorder standard: pin 1 is the input, pin 3 the output and pin 2 is connected to Ground. A mirror lead should be used to make direct data transfers between SOFTY and a cassette recorder. It is also possible to make data transfers between two SOFTIES.

Unfortunately not all tape recorder manufacturers use DIN standard I/O but SOFTY should work with any recorder once the correct method of interconnection has been found.

It is helpful to be able to hear the recorded data. On some recorders the insertion of a plug into the output socket operates a cut-out switch: in such cases it is suggested that the operation of this switch be defeated by bridging it with a low value resistor. It may be that the setting of the recorder's volume control will affect the output signal - but this is not always so.

Observe the voltage on pin 24 of the INS 8060 and confirm that adjustment of VR1 will cause it to swing between Vcc and Ground. Leave VR 1 set at the point of change. This should be very close to the centre of travel.

Record some data which is easy to identify on tape and then clear screen memory. Check for a loud and continuous tone lasting about 5 seconds on replay. With SOFTY in the RECALL mode attempt to recover the data. Even though the screen will be obscured by constant action of the MPU the data should be clearly seen as it is written into screen memory. When the transmission is complete SOFTYshould return automatically to normal operation.

When the data has been recovered SOFTY will EXOR all the data bytes together with a check-byte which was appended at time of transmission and place the result in the MODE FLAG. If the transmission has been received correctly the byte flagged will be AA.

If attempts to recover data with AB linked are unsuccessful then BC should be linked instead. Some slight improvements may be effected by adjusting VR1 slightly, but such adjustments are a waste of time if the system does not work at all.

PARALLEL INTERFACES USING PORTS A & B.

The two eight-bit programmable I/O ports are a part of the facility of the INS 8154 RAM I/O chip. A more detailed description is available in the manufacturers data sheet.

Each port consists of an eight bit output data latch and an eight bit input data latch. Any bit may be defined either as an input or as an output. It is also possible to set, clear or read any bit individually. Moreover, port A may be operated in strobed input or strobed output modes.

Associated with each port is an output definition register (ODR). Each ODR is an eight bit latch which defines which of the bits will be used as outputs. The ODR is a write-only register.

An Interrupt Request line from the INS 8154 is connected to Sense A of the INS 8060 and will generate an interrupt. This signal is only active when port A is used in the strobed mode.

SOFTY's master reset line is connected to the INS 8154. After reset all I/O ports will be in the basic I/O mode and configured as inputs. If SOFTY's firmware is retained port A is then redefined as outputs and used to select the on-screen page and to scan the keymatrix which is read from port B. As there are eight bits in each port SOFTY would scan a matrix of up to 64 switches.

A port can have some inputs and some outputs since there is an ODR latch for each bit in the port. A write to a bit defined as an input will load a new value into the output data latch but it will have no effect on the I/O line.

A data read from lines defined as outputs will read the data from the output data latch, provided there is no short circuit on the I/O line.

When port A is to be used in one of the strobed modes two bits of port B are used for handshake control functions. Accordingly only six bits of port B are available for normal I/O. PB 7 and PB 6 are masked from parallel writes to port B when port A is in the strobed mode - but not from single bit writes. When initiating strobed mode a write to port B (ODR should be performed first of all with 0 in PB 7, 1 in PB 6 and the other bits as 1 (outputs) or 0 (inputs) as required.

The mode definition register (MDR) is also a write-only register which defines the operating mode of port A only.

It is a convention on INS8060 systems to use P2 as the RAM pointer. SOFTY uses P2 to point to the RAM I/O: locations 00 to 7F will address the RAM part of the INS8154. Locations 70 to 7F are in fact the STATUS LINE of information - memory locations which are used internally by SOFTY's firmware.

Here is a table showing the manner in which the ports and their associated registers may be addressed.

Instruction.	Action.	Remarks.
CA 9X	Set Bit.	Bit selected by X. If X = 0 to 7 bit is in port A. If X = 8 to F bit is in port B. The accumulator and data bus are not involved in bit set/clear operations.
CA 8X	Clear bit.	
C2 8X	Read bit.	The bit read does not appear in its normal place in the accumulator. It becomes the most significant bit and the other bits become 0. As only the sign bit is affected a bit read can be followed by a conditional jump-if-positive.
C2 A0	Read port A.	
C2 A1	Read port B.	
CA A0	Write to port A.	
CA A1	Write to port B.	
CA A2	Write to ODR A.	A 1 in the ODR register defines the associated line as an output. A 0 defines it as an input.
CA A3	Write to ODR B.	
CA A4	Write to MDR.	The MDR defines mode of operation of port A. On initiation normal I/O mode is selected. A data write will select other modes as follows:

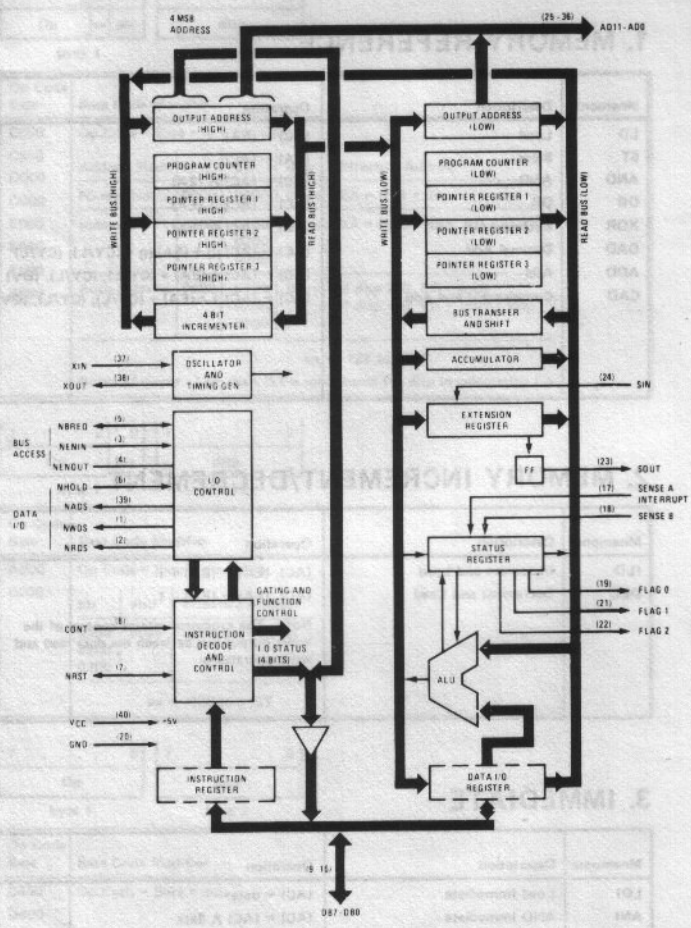
Data.	Mode.
00	Normal I/O
20	Strobed Input.
60	Strobed Output.
E0	Strobed Output (Tri-State)

Strobed Input Mode.

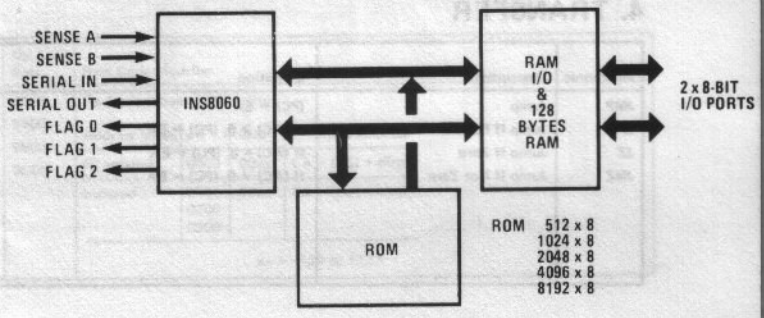
The peripheral device places data on those lines of port A which have been defined as inputs and applies a negative-going strobe signal (STB) to PB 7. The data will be latched into port A on the rising edge of STB and this will happen whether the INS 8154 is selected or not. The MPU can be performing some other task at the time.

When data is in the latch SOFTY will output a high signal on PB 6 to indicate to the peripheral that the input buffer is full (IBF).

This automatically generates an internal interrupt request which is ANDed with the output data latch of PB 7 which now performs an interrupt enable function. (This latch may be set or cleared by a single bit write to PB 7)



INS8060 Internal Architecture



SC/MP Minimum System

SC/MP STATUS REGISTER

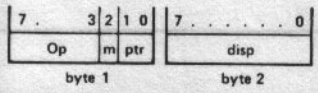
7	6	5	4	3	2	1	0
CY/L	OV	S_B	S_A	IE	F_2	F_1	F_0

Flags	Description
$F_0 - F_2$	User assigned flags 0 through 2.
IE	Interrupt enable, cleared by interrupt.
S_A, S_B	Read-only sense inputs. If IE = 1, S_A is interrupt input.
OV	Overflow, set or reset by arithmetic operations.
CY/L	Carry/Link, set or reset by arithmetic operations or rotate with Link.

© 1976 National Semiconductor Corp.

1. MEMORY REFERENCE

Mnemonic	Description	Operation
LD	Load	(AC) ← (EA)
ST	Store	(EA) ← (AC)
AND	AND	(AC) ← (AC) ∧ (EA)
OR	OR	(AC) ← (AC) ∨ (EA)
XOR	Exclusive-OR	(AC) ← (AC) ∀ (EA)
DAD	Decimal Add	(AC) ← $(AC)_{10} + (EA)_{10}$ + (CY/L); (CY/L)
ADD	Add	(AC) ← (AC) + (EA) + (CY/L); (CY/L), (OV)
CAD	Complement and Add	(AC) ← (AC) + ~(EA) + (CY/L); (CY/L), (OV)



Op Code Base	Base Code Modifier
C000 C800 D000 D800 E000 E800 F000 F800	Op Code = Base + m + ptr + disp

Address Mode	m	ptr	disp	Effective Address
PC-relative	0000	0000	00xx	EA = (PC) + disp
Indexed	0000	0100 0200 0300	00xx	EA = (ptr) + disp
Auto-indexed	0400	0100 0200 0300	00xx	If disp ≥ 0, EA = (ptr) If disp < 0, EA = (ptr) + disp

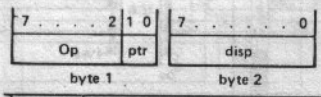
xx = -128 to +127

Note: If disp = -128, then (E) is substituted for disp in calculating EA.

2. MEMORY INCREMENT/DECREMENT

Mnemonic	Description	Operation
ILD	Increment and Load	(AC), (EA) ← (EA) + 1
DLD	Decrement and Load	(AC), (EA) ← (EA) - 1

Note: The processor retains control of the input/output bus between the data read and write operations.



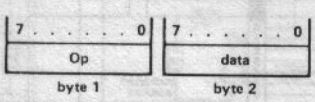
Op Code Base	Base Code Modifier
A800 B800	Op Code = Base + ptr + disp

ptr	disp	Effective Address
0100 0200 0300	00xx	EA = (ptr) + disp

xx = -128 to +127

3. IMMEDIATE

Mnemonic	Description	Operation
LDI	Load Immediate	(AC) ← data
ANI	AND Immediate	(AC) ← (AC) ∧ data
ORI	OR Immediate	(AC) ← (AC) ∨ data
XRI	Exclusive-OR Immediate	(AC) ← (AC) ∀ data
DAI	Decimal Add Immediate	(AC) ← $(AC)_{10}$ + $data_{10}$ + (CY/L); (CY/L)
ADI	Add Immediate	(AC) ← (AC) + data + (CY/L); (CY/L), (OV)
CAI	Complement and Add Immediate	(AC) ← (AC) + ~data + (CY/L); (CY/L), (OV)



Op Code Base	Base Code Modifier
C400 D400 DC00 E400 EC00 F400 FC00	Op Code = Base + data

4. TRANSFER

Mnemonic	Description	Operation
JMP	Jump	(PC) ← EA
JP	Jump If Positive	If (AC) ≥ 0, (PC) ← EA
JZ	Jump If Zero	If (AC) = 0, (PC) ← EA
JNZ	Jump If Not Zero	If (AC) ≠ 0, (PC) ← EA

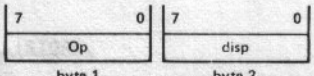
Op Code Base	Base Code Modifier
9000 9400 9800 9C00	Op Code = Base + ptr + disp

Address Mode	ptr	disp	Effective Address
PC-relative	0000	00xx	EA = (PC) + disp
Indexed	0100 0200 0300	00xx	EA = (ptr) + disp

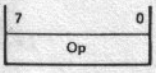
xx = -128 to +127

5. DOUBLE-BYTE MISCELLANEOUS

Mnemonic	Description	Operation
DLY	Delay	count AC to -1, delay = 13 + 2(AC) + 2 disp + 2^9 disp microcycles



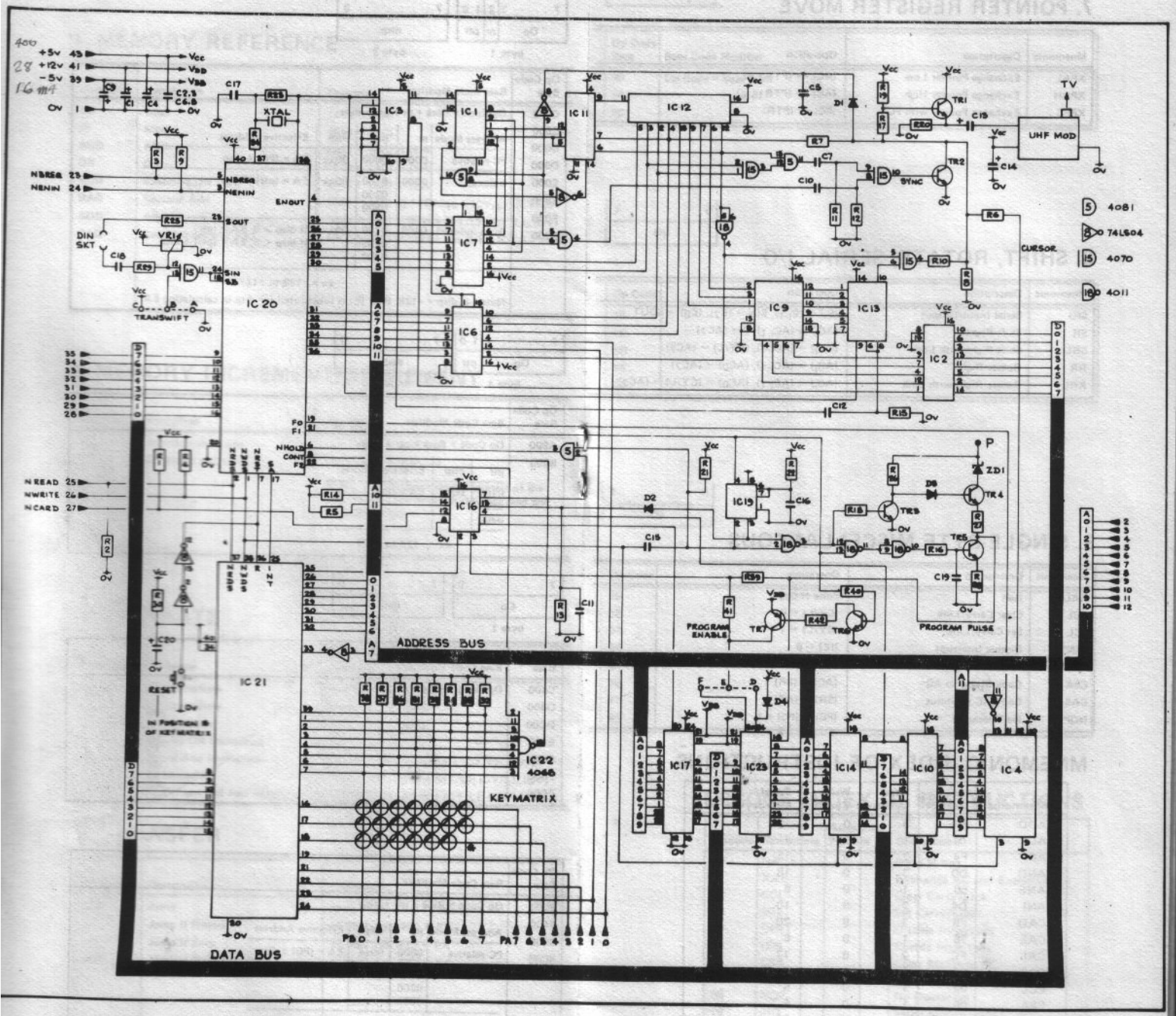
Op Code Base	Base Code Modifier
8F00	Op Code = Base + disp



6. EXTENSION REGISTER

Mnemonic	Description	Operation
LDE	Load AC from Extension	(AC) ← (E)
XAE	Exchange AC and Extension	(AC) ↔ (E)
ANE	AND Extension	(AC) ← (AC) ∧ (E)
ORE	OR Extension	(AC) ← (AC) ∨ (E)
XRE	Exclusive-OR Extension	(AC) ← (AC) ∀ (E)
DAE	Decimal Add Extension	(AC) ← $(AC)_{10}$ + $(E)_{10}$ + (CY/L), (CY/L)
ADE	Add Extension	(AC) ← (AC) + (E) + (CY/L); (CY/L), (OV)
CAE	Complement and Add Extension	(AC) ← (AC) + ~(E) + (CY/L); (CY/L), (OV)

Op Code	
40 01 50 58 60 68 70 78	



I.C.s

1. 74LS93
2. 74LS157
3. 74LS90
4. MM2102
5. CD4081
6. CD4503
7. CD4503
8. 74LS04
9. 93427 (74S287)
10. MM2114
11. 74LS73
12. CD4040
13. 7495
14. MM_114
15. CD4070 (74C86)
16. 74LS42
17. MM2708
18. CD4011
19. LM555
20. INS8060
21. INS8154
22. CD4068

TRANSISTORS.

1,2,3,4,5,6, 7	BC182 (NPN) BC212 (PNP)

DIODES.

1,2,3,4	1N914 or 1N4148
ZD1	2.7volt Zener.

XTAL.

4 MHZ CRYSTAL.

CAPACITORS.

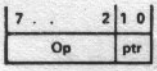
1.	16 volt (or more) Elect.
4,9,13,14,20	6 volt (or more) Elect.
2,3,6,8,	Decoupling discs.
5,11	5000 or 4700pf
7,16	.01uf Mylar
18	.1 or .082uf Mylar

RESISTORS.

1,2,4,5,9,14,16,18,19,25,26,29,30,31,32,33,34,35,36, 37,38,40,42			all 10K
3,17			6K8
6,11,12			22K
7			47K
8			3K9
10			15K
13,24,20			1K
15			330
21,23			100K
22			82K
27			47
28			18
39,41			2K2
10	220pf	C	T = Tantalum Bead
12,15	330pf	C	M = Mylar
17	22pf	C	C = Ceramic Disc
19	560pf	C	

7. POINTER REGISTER MOVE

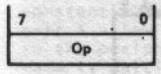
Mnemonic	Description	Operation
XPAL	Exchange Pointer Low	(AC) ↔ ($PTR_{7:0}$)
XPAH	Exchange Pointer High	(AC) ↔ ($PTR_{15:8}$)
XPPC	Exchange Pointer with PC	(PC) ↔ (PTR)



Op Code Base	Base Code Modifier
30	Op Code = Base + ptr
34	
3C	

8. SHIFT, ROTATE, SERIAL I/O

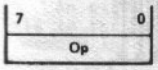
Mnemonic	Description	Operation
SIO	Serial Input/Output	(E_i) → (E_{i-1}), SIN → (E_7), (E_0) → SOUT
SR	Shift Right	(AC_i) → (AC_{i-1}), 0 → (AC_7)
SRL	Shift Right with Link	(AC_i) → (AC_{i-1}), (CY/L) → (AC_7)
RR	Rotate Right	(AC_i) → (AC_{i-1}), (AC_0) → (AC_7)
RRL	Rotate Right with Link	(AC_i) → (AC_{i-1}), (AC_0) → (CY/L) → (AC_7)



Op Code	
19	
1C	
1D	
1E	
1F	

9. SINGLE-BYTE MISCELLANEOUS

Mnemonic	Description	Operation
HALT	Halt	Pulse H-flag
CCL	Clear Carry/Link	(CY/L) ← 0
SCL	Set Carry/Link	(CY/L) ← 1
DINT	Disable Interrupt	(IE) ← 0
IEN	Enable Interrupt	(IE) ← 1
CSA	Copy Status to AC	(AC) ← (SR)
CAS	Copy AC to Status	(SR) ← (AC)
NOP	No Operation	(PC) ← (PC) + 1



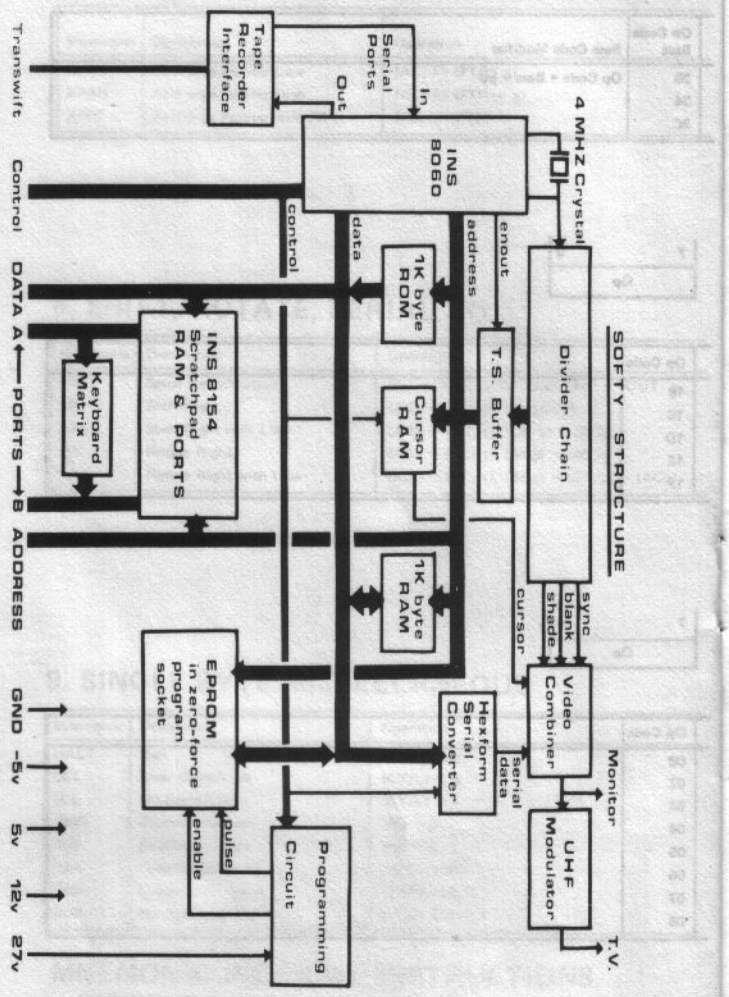
Op Code	
00	
02	
03	
04	
05	
06	
07	
08	

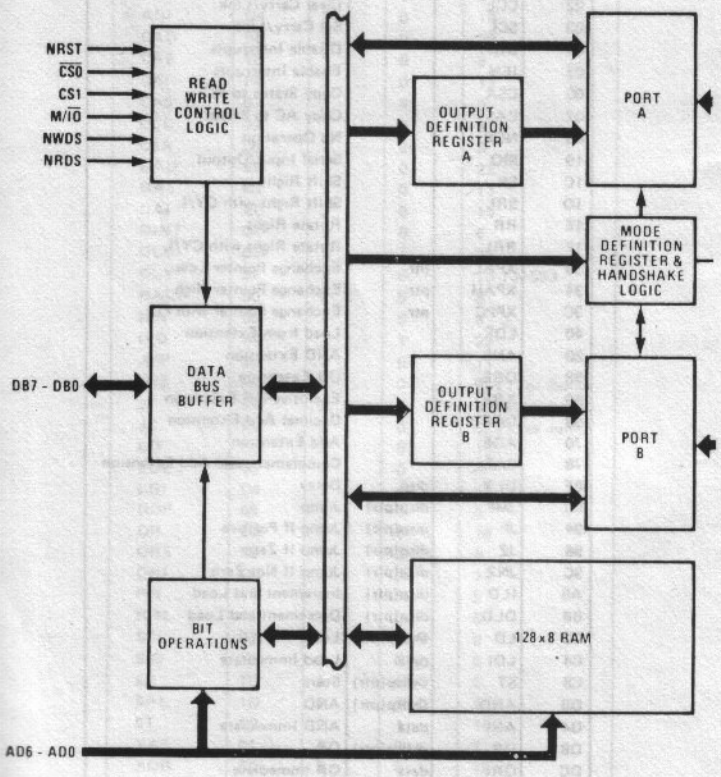
MNEMONIC INDEX OF INSTRUCTIONS

Mnemonic	Opcode	Read Cycles	Write Cycles	Total Microcycles
ADD	F0	3	0	19
ADE	70	1	0	7
ADI	F4	2	0	11
AND	D0	3	0	18
ANE	50	1	0	6
ANI	D4	2	0	10
CAD	F8	3	0	20
CAE	78	1	0	8
CAI	FC	2	0	12
CAS	07	1	0	6
CCL	02	1	0	5
CSA	06	1	0	5
DAD	E8	3	0	23
DAE	68	1	0	11
DAI	EC	2	0	15
DINT	04	1	0	6
DLD	B8	3	1	22
DLY	8F	2	0	13-131593
HALT	00	2	0	8
IEN	05	1	0	6
ILD	A8	3	1	22
JMP	90	2	0	11
JNZ	9C	2	0	9, 11 for Jump
JP	94	2	0	9, 11 for Jump
JZ	98	2	0	9, 11 for Jump
LD	C0	3	0	18
LDE	40	1	0	6
LDI	C4	2	0	10
NOP	08	1	0	5
OR	D8	3	0	18
ORE	58	1	0	6
ORI	DC	2	0	10
RR	1E	1	0	5
RRL	1F	1	0	5
SCL	03	1	0	5
SIO	19	1	0	5
SR	1C	1	0	5
SRL	1D	1	0	5
ST	C8	2	1	18
XAE	01	1	0	7
XOR	E0	3	0	18
XPAH	34	1	0	8
XPAL	30	1	0	8
XPPC	3C	1	0	7
XRE	60	1	0	6
XRI	E4	2	0	10

OPCODE INDEX OF INSTRUCTIONS

Opcode	Mnemonic	Assembler Format	Operation
00	HALT		Pulse H-flag
01	XAE		Exchange AC and Extension
02	CCL		Clear Carry/Link
03	SCL		Set Carry/Link
04	DINT		Disable Interrupts
05	IEN		Enable Interrupts
06	CSA		Copy Status to AC
07	CAS		Copy AC to Status
08	NOP		No Operation
19	SIO		Serial Input/Output
1C	SR		Shift Right
1D	SRL		Shift Right with CY/L
1E	RR		Rotate Right
1F	RRL		Rotate Right with CY/L
30	XPAL	ptr	Exchange Pointer Low
34	XPAH	ptr	Exchange Pointer High
3C	XPPC	ptr	Exchange Pointer with PC
40	LDE		Load from Extension
50	ANE		AND Extension
58	ORE		OR Extension
60	XRE		Exclusive-OR Extension
68	DAE		Decimal Add Extension
70	ADE		Add Extension
78	CAE		Complement and Add Extension
8F	DLY	disp	Delay
90	JMP	disp(ptr)	Jump
94	JP	disp(ptr)	Jump If Positive
98	JZ	disp(ptr)	Jump If Zero
9C	JNZ	disp(ptr)	Jump If Not Zero
A8	ILD	disp(ptr)	Increment and Load
B8	DLD	disp(ptr)	Decrement and Load
C0	LD	@disp(ptr)	Load
C4	LDI	data	Load Immediate
C8	ST	@disp(ptr)	Store
D0	AND	@disp(ptr)	AND
D4	ANI	data	AND Immediate
D8	OR	@disp(ptr)	OR
DC	ORI	data	OR Immediate
E0	XOR	@disp(ptr)	Exclusive-OR
E4	XRI	data	Exclusive-OR Immediate
E8	DAD	@disp(ptr)	Decimal Add
EC	DAI	data	Decimal Add Immediate
F0	ADD	@disp(ptr)	Add
F4	ADI	data	Add Immediate
F8	CAD	@disp(ptr)	Complement and Add
FC	CAI	data	Complement and Add Immediate





INS8154 128-by-8-Bit RAM I/O

If the output data latch of PB 7 contains a 1 the interrupt is enabled and the interrupt line of the INS 8060 will be driven high. This line is the sense A line. It is worth mentioning that this does not automatically generate an interrupt: the MPU also has an interrupt enable flag which is held in the status register.

When port A is read the interrupt is automatically removed and IBF is brought low to indicate to the peripheral that SOFTY is ready to accept more data.

Strobed Output.

In this mode those lines of port A defined as outputs are constantly driven. When a data write has been performed to port A a low going signal (OBF) is output on PB 6 to indicate to the peripheral device that the output buffer is full and data is waiting to be read.

The peripheral outputs a low going signal (ACK) to PB 7 when it reads the data from port A. Immediately upon ACK going low SOFTY places OBF high again. When ACK goes high an interrupt request is made and if the interrupt is enabled (as defined for strobed input) an interrupt is generated by placing sense A of the INS 8060 high. This interrupt is automatically removed upon the next write to port A.

Strobed Output with Tri-State Control.

In this mode port A is normally high-impedance. It is equal to the previous Strobed Output mode in all respects but one: port A is only driven when ACK (PB7) is low.

An example of use in this mode would be when port A is in the addressing space of another MPU and read as a memory location.

INTERNAL USES OF THE I/O PORTS.

SOFTY uses the I/O ports for certain internal functions. The two lowest bits of port A are used as a latch to select which page of information is shown on-screen when the INS 8060 releases the bus. The three highest lines of port A and the 7 lowest lines of port B are used to scan the keymatrix (in fact the internal program scans a bigger matrix of keys: only 20 keys are physically present in the matrix. The other keycodes can be experimentally derived by shorting individual lines of port A and port B. A unique keyword will be written to its position in the Status Line and may be read from this location by the user's program)

The user need not consider SOFTY's internal use of the ports or the keymatrix: so long as no key is pressed the keyboard is effectively absent. But any program which manipulates ports, their associated registers or the contents of pointers 1 or 2 may make a return to firmware impossible. The effect will be a constantly shifting display and no response to keyboard inputs - except Reset. This fact does not mean that the user's program is not working - in fact it is certain that the MPU is doing exactly what it has been instructed to do!

NOTE that all lines of port B have a 10K pullup to Vcc on the SOFTY card and because of this port B is often the better choice for input signals. Also note that all port B lines except PB 7 are connected to an eight input NAND gate the eighth input of which is driven by flag 2 of the INS 8060. If the program places flag 2 high and all 7 bits of port B (not PB 7) are also high then the MPU will immediately suspend operations. The NAND gate output is connected to the MPU CONT input.

This function is necessary to force the INS 8060 to release the bus so that the VDU can scan memory contents and present a data picture. Flag 2 is placed high as a part of the keyscan subroutine and operations are suspended until the next key stroke. When CONT goes high flag 2 is reset by the program and the keymatrix scanned to find the keyword. When the keyword has been actioned the program returns to the keyscan subroutine and the process repeats.

There is an exception to this rule: if the keyword was a cursor key input flag 2 is not placed high. The program introduces a delay and then 'falls-through' to give a cursor repeat function. Repeats are counted in the Cursor Speed Counter in the Status Line and when a whole line has been traversed the delay is only introduced at the end of every line.

Ports are capable of considerable drive and will illuminate LEDs. It follows that a seven-segment calculator display of eight digits may be driven directly by matrixing. One port can handle segments and the other digits. The output of the display can be any alphanumeric symbols which the user can devise from the seven segments and decimal point.

EXECUTION OF PROGRAMS BY THE INTERNAL MICROPROCESSOR.

Programs written for the INS 8060 are essentially RELOCATABLE: they will run from any point in addressing space. Only the values of the Pointers and the Program Counter are absolute; it is usually a simple matter to track down those instructions which assign values to the Pointer Registers and modify them.

SOFTY may be used to action user programs in four different ways:

1) By placing the program in EPROM and substituting it for the firmware EPROM.
2) By placing the program in EPROM and then using $6 to run that program whilst the EPROM remains in the programming socket.
3) By transferring the program to SCRATCHPAD with $2 and $4 as a recursive subroutine. This makes it available as a keyfunction $EX. (marked GOSUB on the keyboard)
4) By running it directly in screen RAM with the benefit of an Editing Facility. The key marked EX transfers control to RAM using P1 and execution starts at the location which follows the cursor. A breakpoint may be set by inserting a 3D byte. When the program reaches such a byte, which causes it to return to a point in SOFTY's firmware, the contents of the internal registers will be displayed in the STATUS LINE. The registers displayed are P3, which by convention is the subroutine pointer, the Accumulator, the Extension Register and the Status Register. It may be found helpful to label these location on-screen.
When execution is restarted no loss of status results because the program first loads the registers from their stored locations. The 3D byte should be substituted for a normal program instruction and not appear in the final program.
Certain assumptions are made for this editing feature to work: P1 should not be reloaded or no return to the editor is possible and P2 should not be reloaded because it points to SCRATCHPAD and PORTS. The user program may use P2 to access the PORTS and SCRATCHPAD. If the user program does reload P1 or P2 the user program will still work but no return to the editor is possible and therefore no meaningful readout will be presented on the screen. It would be possible for the user program to call subroutines from SOFTY firmware using P3 - the KEYSCAN routine might prove helpful and this is located at #033A.

DEMONSTRATION PROGRAMS.

Simple examples of programming follow. More complex examples are given later as applications for SOFTY.

1) Decimal addition of two numbers:

```
08      No operation. (the first memory location is unused.)
C4      Load immediate (the byte which follows)
55      Data
EC      Decimal-add immediate
26      Data
C8      Store (the accumulator)
05      Displacement (meaning present location plus #05)
3D      Breakpoint.
```

Return the cursor over the entry point (08 byte) and key EX. Change the values of the two numbers (data) noting that if the total exceeds 100 the carry flag in the Status Register is set and added-in to the next calculation. Use this information to write a program to add two four digit decimal numbers.

2) This is an example of an extra keyfunction - a user program which runs in SCRATCHPAD.

```
    XY      Any data byte you please.
Loop A8FE   Increment and load the byte two memory locations
              back (the one you just wrote in)
    CD01    Store the result at P1 (cursor) and autoincrement P1.
    35      Copy P1 high to Accumulator (to test if memory
              is full. NOTE that P1 will not become #1000 when
              it increments from #0FFF because of the internal
              paging structure of the INS 8060. P1 becomes
              #0000.)
    9803    Jump out of this program if P1 high is zero.
    35      Otherwise restore P1 high
    90F6    Jump back to loop.
```

Step the cursor back to F6 byte and key $2. Using the cursor keys extend the cursor to cover the whole block and lift it into SCRATCHPAD with $4. If you can see the first line of inverted video below the main screen contents you will be able to see the program in SCRATCHPAD.

To verify the existence of the program clear the screen and key $5 which will write it back.

Run the program with $EX. A count will be written into screen memory starting from the location of the cursor at the data byte which you specified.

PROGRAM EXECUTION BY AN EXTERNAL MICROPROCESSOR.

The INS 8060 has control lines which govern bus access because the manufacturers realized that its special virtues would often be required in multiprocessor applications. A multiprocessor application is one in which address lines, data lines and memory or peripheral devices are shared by more than one microprocessor.

It is possible for the screen memory etc to be placed in the addressing space of a microprocessor in an external system under development. This provides such an external system with SOFTY's facilities on a temporary basis for the purpose of program development. Microsystems often do not need a keyboard, VDU, cassette storage or an EPROM programmer after the program development stage has been completed.

The system under development may have and usually will have different address locations for the data presented on SOFTY's screen: that depends only on the manner in which the address lines are connected.

Before any connection with an external system is attempted the user should make sure that the operation of the address selection logic, bus-access logic and control signals is understood.

CONTROL SIGNALS - ACCESS TO THE ADDRESS BUS.

SOFTY has an internal 12 bit address bus which is accessible by the INS 8060 and the VDU divider chain. Apart from the most-significant bit which is pulled up to Vcc when not driven internally, the address lines and the data lines are also available to an external device such as a microprocessor.

The two most significant bits of the address bus are decoded to select which of the memory devices in addressing space is enabled. As the most significant bit is not available externally it follows that an external microprocessor cannot select SOFTY's own firmware or the RAM I/O but that the WORKING RAM or an EPROM in the programming socket may be selected.

An external device may use NBREQ and NENIN to gain access to SOFTY's address bus. At the start of any read or write cycle the INS 8060 examines the state of NBREQ which is pulled to Vcc by a resistor. If any other device is already holding NBREQ low bus access is denied. If NBREQ is high the INS 8060 issues a bus request by pulling NBREQ low. NBREQ is not then released until the memory transaction has been completed. If NENIN is low or later goes low bus access is granted and the INS 8060 places an address on the bus and a read or write action is performed.

The NENOUT line is used internally to enable the VDU scanner. NENOUT is low when the INS 8060 is not using the bus (NBREQ high) and NENIN is low: at other times it is high. It should now be apparent that if NENIN is high neither the INS 8060 nor the VDU scanner may use the bus - and therefore an external device may gain bus access by placing NENIN high.

If there is no external system NENIN should be wired low.

CONTROL SIGNALS - MEMORY SELECT LOGIC.

The decoding device which selects which of the memory devices is enabled has an input NCARD which is available externally. If NCARD is high none of the devices is selected and the data bus remains in a high impedance state. When SOFTY is acting as program memory for an external system a memory-read or write is performed by placing NCARD low.

If there is no external system NCARD should be wired low.

When SOFTY's working RAM is selected the external system may perform a data write by placing a low going pulse on the NWRITE input.

NREAD is normally pulled low but it may also be used for memory selection by an external device which requires such an input to memory.

If the external device pulls A10 low the EPROM in the program socket will be selected. If A10 remains high or is unconnected the SCREEN RAM is selected. This enables the development of programs in RAM and later transference to a 2708 or 2716 type EPROM; with an intermediate stage in the case of the 2716 in which half the program is in RAM and the other half is in EPROM.

INS8154 128-by-8-Bit RAM I/O

PRODUCT DEVELOPMENT USING SOFTY.

SOFTY is intended for the development of programs which will eventually become firmware - which is software residing in ROM and forming a part of a microsystem.

During the gestation period of an infant system SOFTY will be connected in place of the firmware ROM via an umbilical cord terminated in a 24 pin DIL plug. The umbilical cord is a ribbon cable which carries address, data and control signals.

In the end, when the program is complete and working, the header plug is removed and replaced by an EPROM device programmed by SOFTY. Then the new system will work all by itself.

It is assumed that the infant system is completely developed except for a lack of program memory. In fact it is often possible to interchange software and hardware problems en route to a final design.

A prototype of the new system should be constructed using some system which permits alteration - wire-wrap is an ideal prototyping method.

The use of EPROMS which have three voltage rails does not commit a designer to using these devices in the end-product. It is convenient to use EPROMS in the early stages because of the program revisions which may be required. No designer will ever commit to masked devices until he is absolutely sure of his program.

At this present time the three rail device is cheaper and much more easy to obtain than the single rail device. Therefore is was chosen for use by SOFTY for reasons which are practical rather than academic.

The designer is not restricted to designing systems with only 2K bytes of firmware. This is the limit which one SOFTY can handle at any one time - but the system can have any number of kilobytes already developed in EPROM.

Program development is greatly simplified when SOFTY can be connected to the infant system with no loss of facility on either side. The problems associated with bus-access should be given careful thought along these lines:

1) Can SOFTY and the infant system share the busses freely with the employment of all the existing control signals?

2) Can the control signals be connected to switches which will give access to one side or the other?

3) If the external MPU has no input which will cause it to release the busses, can an intermediate tri-state buffer be incorporated into the infant prototype to simulate the same effect?

4) Can SOFTY be physically disconnected from the infant system whilst the test-program is being written and edited?

5) Can program development proceed at a satisfactory rate by the recycling of a few EPROMS? (Remember that selective writing is possible - but erasure must be total)

APPROACHES TO PROGRAM DEVELOPMENT.

1) Any approach to program design should start with a statement of the results the program is expected to achieve. The process by which the results may be achieved is then expressed as a sequence of steps. The steps themselves are then broken down to a sequence of elemental 'building blocks' which each may be expressed by a sequence of machine code instructions.

It is often helpful to write the program in a high-level language even when there is no computer available to run it. The discipline involved in the writing makes later coding simpler.

There is a species of computer program known as a 'compiler' which produces compiled code but it does not produce the ultra-condensed code that a human compiler can achieve. The computer program may be several times as long. A compiling approach should be chosen when memory space is limited (perhaps because the program must fit into a single ROM) or when speed of execution is the main criterion.

2) An interpretive approach is also possible. The microprocessor has many intricate instructions which manipulate bits. The microprocessor system can be considered as an entity and a language made up of machine code routines which manipulate inputs and outputs. Such a language has a few, powerful instructions rather than a lot of fiddly ones.

The inputs to an interpreter can be hexadecimal, ASCII or even analogue inputs from the outside world which are converted into digital quantities.

The keyboard instructions of SOFTY itself is an example of an interpretive language. Most small computers have an interpretive language - such as BASIC.

SOFTY's program was compiled by hand in machine code and many program-economies were made to fit it into a single 2708.

Another example of an interpretive language is given in the appendix.

AN INTERPRETIVE LANGUAGE.

As an example of the interpretive approach a language is listed here in about 400 bytes. The language may be of practical value to some SOFTY users. It is meant to be run in EPROM (in the programming socket) and called by $6.

The first instruction will be fetched from the first screen location of RAM.

There are 16 variables, each named by a single hex digit. Variables 0 to 9 are assigned by the user during the course of the program. A and B are ports A and B. C and D are their respective ODRs. E is the mode definition register associated with port A. F is a keyboard variable - if it is used the program will wait for an input byte of two keystrokes from the keyboard.

The reads and writes to ports A and B are parallel operations. When single bit operations are required -such as those which enable interrupt when port A is used in the strobed mode - a machine code subroutine must be called.

Each instruction consists of two bytes. The variables are kept in the first line of SCRATCHPAD which is the first line of inverted video below the main screen contents.

The point of an interpretive language is that it is designed for the machine AS AN ENTITY. The instructions are simple and powerful, like a super-machine-code.

INST.	OPERATION.	
0LLL	Do machine language subroutine starting at address location defined by three hex digits. Subroutine must end in 3F byte.	
1VDD	Let variable V contain data DD.	
2XVV	Perform operation X on variables.	
	If X = 0 Skip next instruction if V1 = V2.	
	If X = 1 Skip if V1 does not equal V2.	
	If X = 2 Skip if V1 > V2.	
	If X = 3 AND V1, V2.	(Result to V1)
	If X = 4 OR V1, V2.	(Result to V1)
	If X = 5 EXOR V1, V2.	(Result to V1)
	If X = 6 DECIMAL-ADD V1, V2	(Result to V1)
	If X = 7 ADD V1, V2.	(Result to V1)
	If X = 8 SUBTRACT (V1 - V2)	(Result to V1)
30AA	GOTO displacement byte AA. The displacement is a normal signed hex byte for a program counter relative jump.	
60DD	Delay by data in tenths of a second. The data is a hex quantity which gives a programmed delay of up to 25 seconds.	

INTERPRETER LISTING.

```
08 C4 06 CA A0 C2 A1 E4 FF 01 8F 20 C2 A1 E4 FF 0
58 9C F2 C4 06 36 C4 00 32 C4 00 CE 01 32 98 03 1
32 90 F6 C4 06 CA 0A C4 FF CA 0C C2 0E CA A4 C2 2
0D CA A3 C2 0C CA A2 C2 0B CA A1 C2 0A CA A0 C2 3
A0 CA 0A C2 A1 CA 0B C4 0C 35 C5 01 1C 1C 1C 1C 4
9C 09 C1 FF 37 C5 01 33 3F 90 D0 01 C4 01 60 9C 5
0B C1 FF D4 0F 01 C5 01 CA 80 90 BF C4 02 60 9C 6
77 02 C5 01 1C 1C 1C 01 C4 0F 60 9C 16 C4 00 7
CA 0F C4 09 37 C4 73 33 3F C2 0F 1E 1E 1E 1E CA 8
0F 3F 00 00 C2 80 CA 10 C1 FF D4 0F 01 C4 0F 60 9
9C 16 C4 00 CA 0F C4 09 37 C4 73 33 3F C2 0F 1E A
1E 1E 1E CA 0F 3F 00 00 C2 80 CA 11 C1 FE D4 0F B
9C 0A C2 10 E2 11 9C A2 C5 02 90 9E 01 C4 01 60 C
9C 08 C2 10 E2 11 98 F2 90 EE C4 02 60 9C 0B 03 D
C2 10 FA 11 94 E2 90 82 90 58 C4 03 60 9C 06 C2 E
10 D2 11 90 41 C4 04 60 9C 06 C2 10 DA 11 90 36 F
C4 05 60 9C 06 C2 10 E2 11 90 2B C4 06 60 9C 06 0
C2 10 EA 11 90 20 C4 06 60 9C 06 C2 10 EA 11 90 1
15 C4 07 60 9C 06 C2 10 F2 11 90 0A C4 08 60 9C 2
0F 03 C2 10 FA 11 01 C1 FF 1C 1C 1C 1C 01 CA 80 3
90 A4 C4 03 60 9C 09 C5 01 01 31 02 70 31 90 F0 4
C4 06 60 9C 0B C5 01 01 8F 78 02 70 9C F9 90 EE 5
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 08 6
C4 0F 01 3F C4 06 CA A0 C2 A1 E4 FF 01 8F 20 C2 7
A1 E4 FF 58 9C F2 C4 04 07 C4 00 07 C2 A1 E4 FF 8
01 8F 20 C2 A1 E4 FF 50 98 EC 1C 98 06 01 AA 0F 9
01 90 F7 C4 98 CA 7E BA 7E 01 C4 92 60 98 D1 CA A
80 C2 A1 E4 FF 98 B8 02 C2 0F F4 07 CA 0F 90 E7 B
```

HARD COPY - ADDING A PRINTER TO SOFTY.

The listing of programs is a tedious task. A printer to make permanent records is a useful facility. Printers are generally expensive items costing many times the price of SOFTY but a low-cost alternative has been devised.

The printer chosen was one which is widely available on the surplus market: an obsolete ex-GPO Baudot teletype Creed Type 7, which cost all of £25. (A paper tape reader was thrown in with the deal for good value) There are plenty of these marvellous old machines available in good working order. They were designed to last virtually forever.

Baudot is a 5 bit code. It is transmitted to the teletype serially at 50 baud which means that each bit is 20 milliseconds in duration. In an asynchronous transmission the 5 bits are preceded by a start bit (low) and followed by 1 or more stop bits (high) which define the beginning and the end of a character.

As there are only 32 different combinations in 5 bits a case change character is sent to change from figure case to letter case or vice-versa. It is not necessary to send a shift character before each printed character: only when it is needed to change case.

Baudot has about 50 different printed characters plus space, line-feed and carriage return.

An eight bit machine (like SOFTY) can use Baudot code with a start bit and a stop bit already incorporated. The eighth bit is left over and can be used to determine case. If the case bit is chosen to be the most significant bit or sign bit then figures can be defined as positive and letters negative.

To make the serial output from SOFTY's DIN socket drive the teletype an interfacing circuit is required. The original used an optocoupler for good isolation driving a high voltage transistor type MJE 340. A relay could have been used instead.

The program is meant to be placed in EPROM. The EPROM is inserted into the programming socket and the program called with $6 when a printout is required.

Two programs are given. The first lists memory location, op code and the destination of any jump instructions. This is a partial dis-assembler which could be made into a full dis-assembler by adding a printout of the op-codes. The second program lists data as at appears on screen and appends a single digit line number at the end of each line.

PROGRAM LISTER.

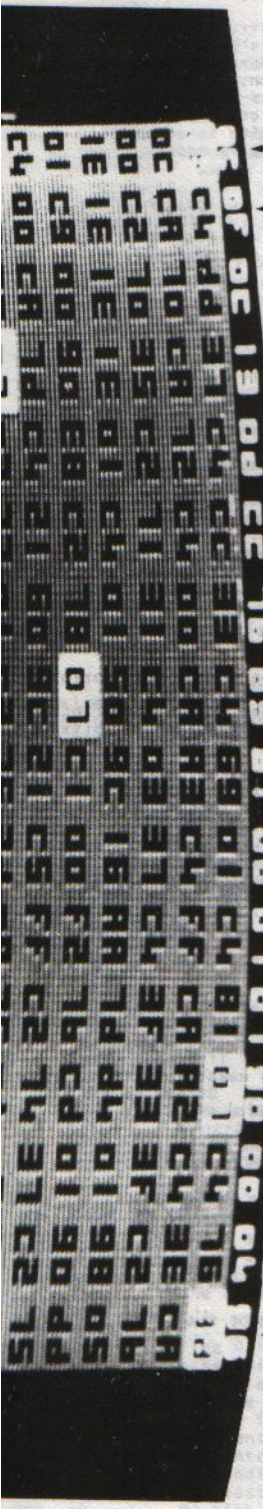
```
C00  02          The columns on the left were printed out by the
C01  C270        program itself on the teletype.
C03  35          First P1, the cursor pointer, is reloaded from
C04  C271        the first two bytes in the Status Line.
C06  31
C07  C4FF        Then the serial output is placed high.  FF is
C09  01          loaded and transferred to the extension register
C0A  19          And a serial shift is made.
C0B  8FFF        Delay to isolate the Start Bit
C0D  C408        P3, the subroutine pointer is loaded with
C0F  37          the SENDWORD subroutine address.
C10  C499 Line
C12  33
C13  C401        Flag 0 in the Status register is set and the
C15  07          other bits are cleared.
C16  C4D0        A carriage return character is loaded
C18  3F          And the subroutine called which will send it.
C19  C4C4        A line feed character is loaded
C1B  3F          And sent.
C1C  C7EF        The entry point to the subroutine is modified
C1E  35          to handle two digit hex numbers.
C1F  01          The last digit of P1 High is printed first
C20  40          P1 high is copied to the E-reg and restored
C21  35
C22  40          P1 high is copied from the E-reg to Accumulator.
C23  3F          And the subroutine to print it is called.
C24  C7F3        The entry point to the subroutine is now modified
C26  31          to handle the first digit of a byte.
C27  01          And P1 low is copied to Acc, then to E-reg
C28  40          P1 low is restored
C29  31
C2A  40          P1 low is copied from E-reg to Acc.
C2B  3F          The first digit of P1 low is printed.
C2C  C7EF        The subroutine entry point is modified for
C2E  31          the second digit.
C2F  01          P1 low is copied and restored again.
C30  40
C31  31
C32  40
C33  3F          And the second digit is printed.
C34  C4C8        A space character is loaded
C36  3F          A space is 'printed'
C37  C4C8        Another space.
C39  3F
C3A  C7F3 Disp.  The subroutine entry point is modified for first
C3C  C100        digit of the byte.  The byte to be sent is loaded
C3E  C900        into the accumulator and then stored back in its
C40  3F          location.  Because Flag 0 is high this will highlight
C41  C7EF        the location on-screen.  The first digit is sent
C43  C501        the entry point modified and the second digit sent.
C45  3F          When the second load took place P1 autoincremented.
C46  06          The status register is copied and +0 tested if it
C47  D401        is clear the result will be zero and the second
C49  9809  J54   byte will have been printed. (Jump to Test)
C4B  C1FF        Read the op-code again.  If it is positive a
C4D  94C1  J10   jump is made to Line -- all single byte op codes
C4F  C400        are positive.  Zero is loaded and copied to Status.
C51  07          Copy Acc to Status to clear Flag 0.
C52  90E6  J3A   Jump to Disp.
C54  C4C8  Test  This routine will test the instruction to find
C56  3F          whether it is a jump.  If it is a letter J followed
C57  C4C8        by the absolute destination address (last two digits)
C59  3F          are printed in the third column. First two spaces
C5A  C1FE        are sent.  Then the op-code is reloaded from P1
C5C  D4F3        The instruction is masked and compared with 90
C5E  E490        If a zero results the instruction is a jump
C60  9CAE  J10   If not zero a jump back to Line is made.
C62  C4D6        The Baudot letter J is loaded
C64  3F          And printed.
C65  C7F3        Now the absolute destination is calculated from
C67  31          the displacement and current value of P1. The sub
C68  01          -routine entry point is first modified for first
C69  40          digit.  P1 is copied to E-reg and restored.
C6A  31
C6B  02          The carry flag is cleared
C6C  C1FF        The displacement is loaded
C6E  70          The E-reg (which contains P1 low) is complemented
C6F  CA7E        and added to the accumulator.  And stored in the
C71  3F          penultimate byte of the status line.  The first
C72  C7EF        digit is sent.  The second digit is then loaded
C74  C27E        from the status line and printed (again after entry
C76  3F          point adjustment)
C77  9097  J10   Now the line is certainly finished and a jump is
C79  6C    (0)   made back to Line.
C7A  6E    (1)   Locations C79 to C88 are the table of conversions
C7B  66    (2)   from Hexadecimal digits to Baudot Figures/letters.
C7C  42    (3)   The 1st bit is always low because it is the start bit.
C7D  54    (4)   The 7th bit is always high because it is the stop bit.
C7E  60    (5)   The eighth bit determines case.  This baffles the
C7F  6A    (6)   line printer which is used to op-codes and not
C80  4E    (7)   tables.  Therefore it prints the figures as single
C81  4C    (8)   byte instructions and the letters as double byte
C82  70    (9)   instructions.
C83  C6F2  (AB)
C85  DCD2  (CD)
C87  C2DA  (EF)
C89  D40F  SECOND This is the entry point for the second digit print.
C8B  9004  J91   The first digit is masked off.  Jump to Both.
C8D  1C    FIRST This is the entry point for the first digit print
C8E  1C          the first digit is rotated into the low four bits
C8F  1C          of the accumulator
C90  1C
C91  02    BOTH  Now both digits are handled the same.  The carry
C92  F4E3        bit is first cleared and an offset is added which
C94  01          will permit the Baudot character to be found in
C95  C080        the table above.  The sum is placed in E-reg and
C97  9001  J9A   Read from the table (Disp #80 uses E-reg) Jump ENTER
C99  3F    EXIT  This is the common exit from the subroutine
C9A  01    ENTRY This is the entry point.  Using adjacent locations
C9B  40          for entry and exit saves reloading the subroutine
C9C  E27F        pointer.  First the digit to be sent must be tested
C9E  941D  JBD   to find if it is the same case as the last-sent
                 character.

                 Word to be sent is placed in E-reg and copied
                 back to be EXORed with the word which was sent when
CA0  40          a case character was last sent.  This word was
CA1  CA7F        stored in the MODE FLAG (P2 7F)  If a case change
CA3  19          is required the result will be negative.  If positive
CA4  8F27        a jump to sendword will be made.  Otherwise the
CA6  C4FD        word is stored in MODE FLAG and a start bit sent.
CA8  01          Delay 20 ms for start bit.
CA9  19          Load FD, place in E-reg and send bit.
CAA  C27F        Load the stored word from the MODE FLAG.
CAC  9404  JB2   Jump to Figcase if positive.
CAE  8FFD        Otherwise send letter case.  As lettercase is all-high
CB0  9008  JBA   a single delay will suffice.  Then jump LOADWORD.
CB2  8F4E  FIGCASE  Delay 40 ms
CB4  19          Sendbit
CB5  8F27        Delay 20 ms.
CB7  19          Sendbit
CB8  8F88        Delay 70 ms
CBA  C27F  LOADWORD   Loadword to be sent from MODE FLAG.
CBC  01    SENDWORD   Word to E-reg.
CBD  19          Send startbit
CBE  8F27        Delay 20 ms.
CC0  19          Send bit 1
CC1  8F27        Delay 20 ms.
CC3  19          Send bit 2
CC4  8F27        Delay 20 ms.
CC6  19          Send bit 3
CC7  8F27        Delay 20 ms.
CC9  19          Send bit 4
CCA  8F27        Delay 20 ms
CCC  19          Send bit 5
CCD  8F27        Delay 20 ms.
CCF  19          Send stop bit
CD0  8F3A        Delay 30 ms.
CD2  90C5  J99   Jump to EXIT.
```

SCREENPRINTER.

The program prints out the screen contents as they appear and appends a line number. The coding is quite similar to the last program. Again the effect is produced by the program operating on itself.

```
00 C2 70 35 C2 71 31 C4 FF 01 19 8F FF C4 6B 33 0
C4 08 37 C4 D0 3F C4 C4 3F 08 08 08 08 08 08 C4 1
01 07 C1 00 1C 1C 1C 1C F4 2F 01 C0 80 3F C1 00 2
CD 01 D4 0F 02 F4 22 01 C0 80 3F 08 08 08 C4 C8 3
3F 31 01 40 31 40 D4 0F 9C D5 06 D4 01 98 C4 C4 4
00 07 C4 FF 70 1C 1C 1C 1C 90 D9 6C 6E 66 42 54 5
60 6A 4E 4C 70 C6 F2 DC D2 C2 DA 3F 01 40 E2 7F 6
94 1D 40 CA 7F 19 8F 27 C4 FD 01 19 C2 7F 94 04 7
8F FD 90 08 8F 4E 19 8F 27 19 8F 88 C2 7F 01 19 8
8F 27 19 8F 27 19 8F 27 19 8F 27 19 8F 27 19 8F 9
27 19 8F 3A 90 C5 FF FF FF FF
```

Scratchpad Information



- 0 ← Cursor Pointer High Byte
- 1 ← Cursor Pointer Low Byte
- 2 ← Alternate Cursor High Byte
- 3 ← Alternate Cursor Low Byte
- 4 ← Subroutine Pointer High Byte
- 5 ← Subroutine Pointer Low Byte
- 6 ← Accumulator Contents
- 7 ← Extension Register Contents
- 8 ← Status Register Contents
- 9 ← Parallel Parity Word
- A ← Matchbyte (Highlighted)
- B ← Cursor Speed Counter
- C ← Keyword
- D ← End Around Carry
- E ← Hexadecimal Difference Between Cursors
- F ← Mode Flag Etc.