MILLIMETER WAVE TWT LIFE TEST MONITOR

Leon L. Stevens, James Perretta, Curt Belusar
This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-90-202 has been reviewed and is approved for publication.

APPROVED:  

JOSEPH J. POLNIAZEK, JR.  
Acting Chief, Surveillance Technology Division  
Directorate of Surveillance

APPROVED:  

FRED J. DEMMA  
Acting Director of Surveillance

FOR THE COMMANDER:  

JAMES W. HYDE III  
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (OCTP) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.
This report presents information on a monitor which can be used to capture and store life test data in real time. The hardware is specifically tailored to capture data without delay in real time. In addition to data acquisition, this report presents information on hardware and programs which can be used to transfer data to larger multitask computer systems where data analysis is more efficient. Because all multitask computer systems (even those as small as PCs) devote considerably amounts of time to tasks other than running the user's test program, they are not directly compatible with real-time data acquisition. Outside hardware is required to buffer real-time data and pass it to the computer system where final data reduction takes place. The interface hardware described in this report consists of two single task microprocessor systems. One system is based on the Motorola 6800 cpu with an SS50 mother board and SS30 input/output slots. The second system is a single board Motorola 6802 microprocessor.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>Life Test Hardware</td>
<td>2</td>
</tr>
<tr>
<td>Millimeter Wave TWT Life Test Procedure</td>
<td>2</td>
</tr>
<tr>
<td>Single Board 6802 Microprocessor System</td>
<td>5</td>
</tr>
<tr>
<td>Single Board 6802 System Memory Map</td>
<td>10</td>
</tr>
<tr>
<td>Single Board 6802 Assembly Language Programs</td>
<td>15</td>
</tr>
<tr>
<td>Single Board 6802 Wire Wrap List</td>
<td>48</td>
</tr>
<tr>
<td>Wire Wrap List Programs in C</td>
<td>71</td>
</tr>
<tr>
<td>SS50 6800 Microprocessor System</td>
<td>76</td>
</tr>
<tr>
<td>SS50 Real Time Clock/Calendar</td>
<td>98</td>
</tr>
<tr>
<td>SS50 Eight Bit Analog to Digital Converter</td>
<td>112</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Life Test Procedure</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Single Board 6802 System Layout Top View</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Single Board 6802 System Schematic</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>SS50 Mother Board Schematic</td>
<td>77</td>
</tr>
<tr>
<td>5</td>
<td>SS50 CPU Board Schematic</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>SS50 CPU Board Top View</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>SS50 CPU Board Bottom View</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>SS30 Serial I/O Board Schematic</td>
<td>83</td>
</tr>
<tr>
<td>9</td>
<td>SS30 Serial I/O Board Top View</td>
<td>84</td>
</tr>
<tr>
<td>10</td>
<td>SS30 Serial I/O Board Bottom View</td>
<td>85</td>
</tr>
<tr>
<td>11</td>
<td>SS30 Standard Parallel I/O Board Schematic</td>
<td>87</td>
</tr>
<tr>
<td>12</td>
<td>SS30 Standard Parallel I/O Board Top View</td>
<td>88</td>
</tr>
<tr>
<td>13</td>
<td>SS30 Standard Parallel I/O Board Bottom View</td>
<td>89</td>
</tr>
<tr>
<td>14</td>
<td>SS30 Open Collector I/O Board Schematic</td>
<td>90</td>
</tr>
<tr>
<td>15</td>
<td>SS30 Open Collector I/O Board Top View</td>
<td>91</td>
</tr>
<tr>
<td>16</td>
<td>SS30 Open Collector I/O Board Bottom View</td>
<td>92</td>
</tr>
<tr>
<td>17</td>
<td>SS50 CMOS Static Memory Board Top View</td>
<td>95</td>
</tr>
<tr>
<td>18</td>
<td>SS50 CMOS Static Memory Board Bottom View</td>
<td>96</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>SS50 32K CMOS RAM Schematic (8KX8 Chips)</td>
<td>97</td>
</tr>
<tr>
<td>20</td>
<td>Real Time Clock/Calendar Schematic</td>
<td>107</td>
</tr>
<tr>
<td>21</td>
<td>SS50 Address Decode for Clock and A/D Converter</td>
<td>108</td>
</tr>
<tr>
<td>22</td>
<td>SS50 Analog to Digital Schematic</td>
<td>109</td>
</tr>
<tr>
<td>23</td>
<td>SS50 Clock/Calendar and A/D Board Top View</td>
<td>110</td>
</tr>
<tr>
<td>24</td>
<td>SS50 Clock/Calendar and A/D Board Bottom View</td>
<td>111</td>
</tr>
</tbody>
</table>
INTRODUCTION:

This report presents both hardware designs and software programs which can be used to capture and store life test data in real time. The hardware is specifically tailored to capture data without delay in real time. In this way, we hope to be able to record the rare (perhaps once in 5 years), very rapid (perhaps less than 200 milliseconds) catastrophic failure of a millimeter wave traveling wave tube. Millimeter wave tubes fail quickly because their parts have relatively low thermal mass and their electron beams have high power densities. If one is able to record sufficient tube parameters during such a failure, one may be able to decipher the root cause of the failure and correct the defect in the next generation of millimeter wave amplifiers. In addition to data acquisition, this report will present information on hardware and programs which can be used to transfer data to larger multitask computer systems where data analysis is more efficient.

BACKGROUND:

Rome Air Development Center has been actively involved in the development of Millimeter Wave Traveling Wave Tubes since the inception of such devices. The classical method of determining the failure mechanism of a microwave tube is to do a postmortem (cut it apart and inspect it). However, millimeter wave tubes are so small and fragile that determining the root cause of the failure by this method has very low credibility. Either the physical opening process or some secondary damage mechanism tends to obscure any visible evidence of the initial failure mechanism. Millimeter wave tube amplifiers are so expensive that testing a few thousand devices to failure is not possible. Learning as much as possible from each actual failure is the most cost effective method to improving these devices. This system which senses the onset of failure and records all significant amplifier parameters during failure (for later analysis) appears to be a sound practical approach.
to gaining knowledge about failure mechanisms.

LIFE TEST HARDWARE:
Because all multitask computer systems (even those as small as PCs) devote considerable amounts of time to tasks other than running the user's test program, they are not directly compatible with real-time data acquisition. Outside hardware is required to buffer real time data and pass it to the computer system where final data reduction takes place. The interface hardware described in this report consists of two single task microprocessor systems. One system is based on the Motorola 6800 cpu with an SS50 mother board and SS30 input/output slots. The second system is a single board Motorola 6802 microprocessor.

MILLIMETER WAVE TWT LIFE TEST PROCEDURE:
The following are typical parameters which would be continuously monitored:

- Filament (Heater) Voltage
- Filament Current
- Focus Anode Voltage
- Focus Anode Current
- Body Current
- Collector Voltage
- Collector Current
- Second Collector Voltage
- Second Collector Current
- RF Input Power
- RF Output Power
- Thermocouple 1
- Thermocouple 2
- Thermocouple 3
- Thermocouple 4
- Time & Date
If as few as 16 parameters (four bytes each) are recorded every few (10) milliseconds, a fantastic quantity of data (200,000,000,000 bytes) is recorded each year. In fact a 20 megabyte hard disk is filled in less than an hour. Obviously the vast majority of this data is redundant and need not be recorded at all. It is only necessary to record data when a significant change has occurred in some parameter. As long as parameters are being compared against limits to determine if it should be recorded, it makes sense to compare the parameters against a second set of limits to determine if the tube under test is in a safe operating range, or if the system should be shut down (crowbared) to protect the tube from damage caused by cooling or power conditioner failures.

In practice, the critical tube parameters are checked first to determine if crowbar is necessary. Next, each parameter is tested against its upper and lower limits of "normal" variation. If any parameter is outside its normal range, all the parameter values are recorded. To keep the system from triggering again on this same value, the "normal variation range" (of this specific parameter) is now centered about its current value. The recentering prevents redundant data from being recorded (as parameters slowly vary over their safe operating range) and still provides chronologic data on all significant parameter variations. Figure one on page four should make the test procedure more clear.
Life Test Procedure

- Parameter Value = _________
- Safety Limits = _________
- Normal Variation limits = _________

The safety limits were set based on theoretical considerations of the tube operating system.

The normal variation limits at point "A" were experimentally determined by recording the highest and lowest value of the parameter over a 24 hour period and multiplying that range by three. The normal variation limits were centered around the present value of the parameter when the life test commenced.

At points "B", "C", and "D" the parameter value hit the upper limit of the normal variation range. All parameters were recorded. The normal variation limits were recentered around the present value.

At point "D" the parameter value hit the safety limit. The system was shut down (crowbarred) and all parameters were continuously recorded until the recording media was full or standby power was lost.

(FIGURE 1)

-4-
SINGLE BOARD MICROPROCESSOR SYSTEM:

(Please refer to the schematic diagram on pages 13 & 14)

IC-4 (Motorola 6802 8 bit microprocessor) is the heart of the system. It has a built-in clock which requires only a 4 MHz external crystal. In this application, external capacitors were added between the crystal leads and ground, to provide a stronger (more stable) clock signal. The sixteen outgoing address lines of IC-4 are buffered by non-inverting line drivers (IC-6 and IC-7). The address lines from the 6802 to the line drivers are labeled with small letters (a1, a2, etc.). The amplified address lines out of the buffers (IC-6 & IC-7) are labeled with capital letters (A1, A2, etc). The eight bidirectional data lines from the 6802 (IC-4) are buffered by IC-8. Small letters (d1, d2, etc) indicate data lines between the 6806 (IC-4) and the line driver (IC-8). Capital letters (D1, D2, etc) represent data lines between the bidirectional line driver (IC-8) and the rest of the system. This convention allows data and address lines to be labeled rather than drawn on the schematic diagram. The read/write output of the 6802 (IC-4) is buffered by one section of IC-16, and uses the same convention. The enable output of IC-4 is buffered by a section of IC-16, and uses the same convention. Valid memory address (VMA) is buffered and inverted by section B of IC11 (7400). VMA and E are combined in section A of IC11 to provide "NOT" of the quantity (E*VMA). Buffered R/W is inverted in section B of IC12 (7404) to provide NOT (R/W).

All input/output (I/O) is memory mapped in the Motorola MC 6802, that is, no additional data lines or address lines are provided for I/O. The microprocessor merely reads from or writes to an apparent memory location, and hardware which decodes chip-select and R/W at that apparent memory location provides the real-world interface. IC-15 (MC6821) provides 16 bits of parallel
data interface between the microprocessor and the outside world. The 6821 is called a peripheral interface adaptor (PIA). IC-18 (MC6850) provides both a serial input interface and a serial output interface. The 6850 is an asynchronous communications interface adaptor (ACIA). IC-19 (MC14411) is a band rate generator chip. In conjunction with a 1 meg ohm resistor and a 1.8432 MHz crystal, it produces a variety of stable clock signals which can be used by the ACIA (6850) to clock serial data either in or out of the ACIA. Although the MC6850 is asynchronous (the same clock need not be provided to both ends of the serial communications link), the clocks must have the same (or very nearly the same) frequency.

Primary memory chip select decoding (including that for I/O) is provided by two 74154 decoders (IC-9 and IC-13) each 74154 has four bits of positive sense binary data input and one of sixteen lines of negative sense chip select as output.

The MC6802 (IC-4) has 16 address lines and can address 65,536 memory locations directly. To read a memory location the 6802 places the address of that location on the address lines, sets the R/W line to read (+5 volts) and sets VMA high (+5 volts). The address lines tell the hardware which location. The read line at +5 volts tells the memory location to place its stored data on the 8 data line. The VMA high is required because the microprocessor often puts garbage on the address lines while it is performing internal processing. Without VMA power would be wasted by selecting unwanted output drivers or worse yet garbage could be written into memory destroying the good data there.

One additional signal is required to write data into a memory location. The enable line (E) is used to synchronize when the memory chip should "capture" the information which is on the data lines. The 6802 places
the desired address on the address line, places R/W to write (zero volts), places E high (+5 volts), places the data to be stored in the memory chip on the data lines, and when the good data has been on the data lines as long as possible (for a given clock speed) pulls the E line to ground (zero volts). This falling edge of the enable (E) line is the signal for the memory location to capture the data now. Several chips like MC6821 and MC6850 have a direct connection to the buffered enable signal. However, some chips like the 5517 memory do not have a separate enable pin. These memory chips "capture" data when their chip select lines go high (they are deselected). To account for this, the primary memory select decode chips (IC-9 and IC-13) are only enabled when E and VMA are high. Not (E*VMA) from PIN-3 of IC-11 is connected to NOT (chip enable) PIN-19 of IC-9 and IC-13. The VMA part of this control signal limits memory selection to valid memory addresses.

Address decoding for NOT chip selects is provided by IC-9 and IC-13. It is impractical to run all 16 address lines to each and every memory chip or I/O chip used in the microprocessor system.

Sixteen address lines allows 64K (65,536K actually) of direct access locations. The most significant binary bit (A15) can be used to differentiate between upper and lower 32K memory blocks (upper 32K when A15 is +5 volts and lower 32K when A15 is 0 volts). IC-9 and IC-13 have a NOT enable on pin-18. NOT A15 from pin-2 of IC-12 is connected to pin-18 of IC-13 (NOT enable). Thus IC-13 is enabled for the upper 32K block ($8000 through $FFFF). A15 from Pin-3 of IC-6 is connected to Pin-18 of IC-9 (NOT enable). Thus IC-9 is enabled for the lower 32K block ($0000 through $7FFF). Buffered address lines A14, A13, A12, and A11 are connected to the D, C, B, and A inputs of both IC-13 and IC-9. These four lines (representing 16 different states) break each of the 32K blocks down into 2K blocks (2,048 actually).
<table>
<thead>
<tr>
<th>ADDRESS IN HEX</th>
<th>CHIP SELECT GOES LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0000 - $07FF</td>
<td>IC-9 PIN-1</td>
</tr>
<tr>
<td>$0800 - $0FFF</td>
<td>IC-9 PIN-2</td>
</tr>
<tr>
<td>$1000 - $17FF</td>
<td>IC-9 PIN-3</td>
</tr>
<tr>
<td>$1800 - $1FFF</td>
<td>IC-9 PIN-4</td>
</tr>
<tr>
<td>$2000 - $27FF</td>
<td>IC-9 PIN-5</td>
</tr>
<tr>
<td>$2800 - $2FFF</td>
<td>IC-9 PIN-6</td>
</tr>
<tr>
<td>$3000 - $37FF</td>
<td>IC-9 PIN-7</td>
</tr>
<tr>
<td>$2800 - $3FFF</td>
<td>IC-9 PIN-8</td>
</tr>
<tr>
<td>$4000 - $47FF</td>
<td>IC-9 PIN-9</td>
</tr>
<tr>
<td>$4800 - $4FFF</td>
<td>IC-9 PIN-10</td>
</tr>
<tr>
<td>$5000 - $57FF</td>
<td>IC-9 PIN-11</td>
</tr>
<tr>
<td>$5800 - $5FFF</td>
<td>IC-9 PIN-12</td>
</tr>
<tr>
<td>$6000 - $67FF</td>
<td>IC-9 PIN-13</td>
</tr>
<tr>
<td>$6800 - $6FFF</td>
<td>IC-9 PIN-14</td>
</tr>
<tr>
<td>$7000 - $77FF</td>
<td>IC-9 PIN-15</td>
</tr>
<tr>
<td>$7800 - $7FFF</td>
<td>IC-9 PIN-16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDRESS IN HEX</th>
<th>CHIP SELECT GOES LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>$8000 - $87FF</td>
<td>IC-13 PIN-1</td>
</tr>
<tr>
<td>$8800 - $8FFF</td>
<td>IC-13 PIN-2</td>
</tr>
<tr>
<td>$9000 - $97FF</td>
<td>IC-13 PIN-3</td>
</tr>
<tr>
<td>$9800 - $9FFF</td>
<td>IC-13 PIN-4</td>
</tr>
<tr>
<td>$A000 - $A7FF</td>
<td>IC-13 PIN-5</td>
</tr>
<tr>
<td>$A800 - $AFFF</td>
<td>IC-13 PIN-6</td>
</tr>
<tr>
<td>$B000 - $B7FF</td>
<td>IC-13 PIN-7</td>
</tr>
<tr>
<td>$B800 - $BFFF</td>
<td>IC-13 PIN-8</td>
</tr>
<tr>
<td>$C000 - $C7FF</td>
<td>IC-13 PIN-9</td>
</tr>
<tr>
<td>$C800 - $CFFFF</td>
<td>IC-13 PIN-10</td>
</tr>
<tr>
<td>$D000 - $D7FF</td>
<td>IC-13 PIN-11</td>
</tr>
<tr>
<td>$D800 - $DFFF</td>
<td>IC-13 PIN-13</td>
</tr>
<tr>
<td>$E000 - $E7FF</td>
<td>IC-13 PIN-14</td>
</tr>
<tr>
<td>$E800 - $EFFF</td>
<td>IC-13 PIN-15</td>
</tr>
<tr>
<td>$F000 - $F7FF</td>
<td>IC-13 PIN-16</td>
</tr>
<tr>
<td>$F800 - $FFFF</td>
<td>IC-13 PIN-17</td>
</tr>
</tbody>
</table>
The eleven lower order buffered address lines (AO through A10) are connected to IC-1, IC-2, IC-3, and IC-5. The five higher order address lines (All through A15) are decoded by IC-9 and IC-13. This information is passed to IC-1, IC-2, IC-3, and IC-5 in the form of an active low chip select (which is applied to output enable and chip enable). Thus the memory chips are fully decoded. Each memory is actually represented by a unique 8-bit memory location.

IC-18 (MC6850 ACIA) is not fully address decoded. This chip is selected by IC-13 at any of the 2K addresses between $E800 and EFFF. However, only one of the lower eleven address lines is connected to IC-18. AO is connected to PIN-11 (the register select line). If the microprocessor reads "memory" location $E801 it gets the data register of IC-18. If location $E800 is read it gets the status register of IC-18. $E803 is the data register of IC-18 (the same as $E801). $E802 is the status register of IC-18 (the same as $E800). $E804 is the same as $E800, etc. The same registers "echo" through the entire 2,048 addresses. All even addresses in this range get the status register because AO is low. All odd addresses in this range get the data register because AO is high. Actually, there are two more registers in the ACIA (transmit data and the control register). These are selected at $E801 and $E800 respectively when the microprocessor performs a write operation. These also echo through the full 2K. Unlike an actual memory chip where reads and writes done to the same location access the same storage register, a read and write to the same location in an ACIA access different storage registers.
IC-14 and IC-15 (MC6821) are not fully address decoded. AO and A1 are connected to register select 0 and register select 1 respectively. In IC-14 $E000, $E001, $E002, and $E003 echo from $E000 to $E7FF. In IC-15 $0800, $D801, $D802, and $D803 echo from $D800 to $DFFFF.

Only seven of the thirty-two 2K memory blocks defined by the address decoders are used in the design shown. It would have been possible to save one chip and decode into eight 8K memory blocks. Use seven of these and allow echoing through the full 8K on each. However, a good base design should have several unused chip selects decoded. This allows a clock or an A to D converter or more memory or more I/O to be added at a later date with almost no more effort than if they had been implemented initially. Without unused chip selects, adding function is a major hassle.

MEMORY MAP OF THE SINGLE BOARD MICROPROCESSOR SYSTEM

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FFFF</td>
<td>IC-5 ROM 3K BY 8</td>
</tr>
<tr>
<td>$F800</td>
<td>Unused PIN-16 IC-3</td>
</tr>
<tr>
<td>$F000</td>
<td>Unused IC-18 MC6850 ACIA</td>
</tr>
<tr>
<td>$E800</td>
<td>IC-14 MC6821 PIA</td>
</tr>
<tr>
<td>$E000</td>
<td>IC-15 MC6821 PIA</td>
</tr>
<tr>
<td>$D800</td>
<td>Unused PIN-11 IC-13</td>
</tr>
<tr>
<td>$D000</td>
<td>Unused PIN-10 IC-13</td>
</tr>
<tr>
<td>$C800</td>
<td>Unused PIN-9 IC-13</td>
</tr>
<tr>
<td>$C000</td>
<td>Unused PIN-8 IC-13</td>
</tr>
<tr>
<td>$B800</td>
<td>Unused PIN-7 IC-13</td>
</tr>
<tr>
<td>$B000</td>
<td>Unused PIN-6 IC-13</td>
</tr>
<tr>
<td>$A800</td>
<td>Unused PIN-5 IC-13</td>
</tr>
<tr>
<td>$A000</td>
<td>Unused PIN-4 IC-13</td>
</tr>
<tr>
<td>$9800</td>
<td>Unused PIN-3 IC-13</td>
</tr>
<tr>
<td>$9000</td>
<td>Unused PIN-2 IC-13</td>
</tr>
<tr>
<td>$8800</td>
<td>Unused PIN-1 IC-13</td>
</tr>
<tr>
<td>$8000</td>
<td>Unused PIN-17 IC-9</td>
</tr>
<tr>
<td>$7800</td>
<td>Unused PIN-16 IC-9</td>
</tr>
<tr>
<td>Address</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>$6800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$6000</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$5800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$5000</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$4800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$4000</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$3800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$3000</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$2800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$2000</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$1800</td>
<td>UNUSED</td>
</tr>
<tr>
<td>$1000</td>
<td>IC-1</td>
</tr>
<tr>
<td>$0800</td>
<td>IC-2</td>
</tr>
<tr>
<td>$0000</td>
<td>IC-3</td>
</tr>
</tbody>
</table>
THE FOLLOWING ASSEMBLY LANGUAGE PROGRAMS WERE WRITTEN FOR THE SINGLE BOARD 6802 MICROPROCESSOR SYSTEM DESCRIBED IN THIS REPORT. ALTHOUGH MOST WERE CREATED SPECIFICALLY FOR THE PROBLEMS FOUND ON THIS PARTICULAR BOARD, THEY SHOULD BE SUFFICIENTLY GENERAL TO PROVIDE INSIGHT INTO THE METHODOLOGY OF TROUBLESHOOTING AND OPERATING THIS TYPE OF SYSTEM.

DOWN AND DOWNROM ARE ASSEMBLY LANGUAGE PROGRAMS WHICH RUN IN THE SS50 6800 SYSTEM AND THE SINGLE BOARD 6802 SYSTEM RESPECTIVELY. THESE ALLOW PROGRAMS TO BE ASSEMBLED IN THE SS50 SYSTEM AND THEN TRANSFERRED TO AND RUN IN THE SINGLE BOARD SYSTEM. THIS ALLOWS SOFTWARE TO BE DEVELOPED FOR THE SINGLE BOARD SYSTEM WITHOUT HAVING TO BURN A NEW EPROM EVERY TIME.

TST13 IS A DATA TRANSFER PROGRAM WHICH ALLOWS DATA TO BE TRANSFERRED FROM A MICROPROCESSOR SYSTEM TO A PC(AT) VIA RS-232-C.

INITIAL TEST PROGRAMS WERE WRITTEN DIRECTLY INTO A 2716 (2K BY 8 EPROM DECODED AT $F800-$FFFF) WITH THE RESTART VECTOR AT $FFFE AND $FFFF POINTING TO $F800. INITIAL TEST PROGRAMS SHOULD BE LOOP TYPE PROGRAMS WITH AS FEW STEPS AS POSSIBLE. THE LOOP SHOULD ACCESS A SINGLE DEVICE WITH A SINGLE READ OR WRITE. THIS ALLOWS THE SCOPE TO USE CHIP SELECT AS A SYNC INPUT AND GIVE AS BRIGHT A DISPLAY AS POSSIBLE.

TESTS WERE CONDUCTED WITH THE MINIMUM NUMBER OF INTEGRATED CIRCUITS INSTALLED IN THE BOARD. ALL POWER AND GROUND CONNECTIONS WERE CHECKED ON THE SOCKETS BEFORE THE INTEGRATED CIRCUITS WERE INSTALLED.
NAM TST1

* THIS PROGRAM LOADS THE A ACCUMULATOR WITH
* A CAPITAL "A" ($41) AND STORES THIS IN
* MEMORY LOCATION $0000. THEN LOOPS BACK TO
* THE STORE IN $0000 ENDLESSLY.

OBJECT CODE: 86 41 *THIS LOADS THE A
    *ACCUMULATOR WITH "A"
    97 00 *THIS STORES THE A
    *ACCUMULATOR IN $0000
    20 FD *THIS LOOPS BACK ONE
    *LINE TO 97 00

WITH SCOPE SYNC ON THE CHIP SELECT LINE GOING TO
MEMORY LOCATION $0000, IT APPEARED THAT $FF WAS BEING
WRITTEN INTO $0000. THE EXPECTED VALUE OF $41 WAS NOT
BEING WRITTEN.

*******************************************************************************

NAM TST2

THIS PROGRAM WAS ASSEMBLED TO SEE WHAT
CODE WOULD EMULATE THE CONDITION OF ALL
DATA LINES TIED HIGH (+5 V) ON INPUT TO
THE MICROPROCESSOR. THIS WAS DONE BECAUSE
THE SCOPE HAD SHOWN THE INPUT DATA LINES
HIGH ALMOST ALL OF THE TIME WHEN ATTEMPTING
TO RUN TST1.

OPT O,S
ORG $100

0100 FF FFFF STX $FFFFFF
0103 FF FFFF STX $FFFFFF
END

TOTAL ERRORS 00000
STORE THE INDEX REGISTER EXTENDED IN $FFFF FITS THE BILL. THIS PROGRAM WILL STORE THE FIRST HALF OF WHAT IS IN THE INDEX REGISTER IN MEMORY LOCATION $FFFF AND STORE THE SECOND HALF OF WHAT IS IN THE INDEX REGISTER IN MEMORY LOCATION $0000.

THIS SHOULD GIVE ALTERNATING CHIP SELECTS TO IC-3 AND IC-5 (WITH THE R/W LINE LOW). IN FACT, THE SCOPE SHOWED ALTERNATING CHIP SELECTS OF IC-3 AND IC-5 IN THE WRITE MODE.

BASED ON THESE RESULTS ALL CONNECTIONS WERE TRACED OUT ON THE DATA LINES IN QUESTION. THE WIRING APPEARED CORRECT. THE NEXT THING WAS TO REPLACE THE 74245 BIDIRECTIONAL DATA DRIVER CHIP. TST1 NOW RAN CORRECTLY.

******************************

NAM TST3

CONTINUALLY WRITES THE A ACCUMULATOR (WHATEVER HAPPENS TO BE IN THE A ACCUMULATOR) TO MEMORY LOCATION $0001

OPT O,S
ORG $800

0800 97 01 LP1 STA A 1
0802 20 FC BRA LP1
END

LP1 0800

TOTAL ERRORS 00000
THIS PROGRAM APPEARED TO RUN CORRECTLY WITH OCCASIONAL BURSTS OF ERRATIC BEHAVIOR. PART OF THE TIME THE PROGRAM OPERATED AS EXPECTED WITH THE PROPER NUMBER OF CLOCK CYCLES PER LOOP. PART OF THE TIME IT WOULD RUN IN A LOOP WITH PERHAPS THREE OR FOUR TIMES THE NUMBER OF CYCLES EXPECTED IN THE LOOP AND WHAT WAS BEING DONE IN THE LOOP WAS NOT OBVIOUS.

*NAM TST4
  (LATER RENAMED DOWNROM)*

*NAM  DOWNROM
OPT  O,S,NOP
ORG  $0800

THIS PROGRAM RESIDES IN ROM ON THE SINGLE BOARD 6802 MICROPROCESSOR THIS PROGRAM DOWNLOADS (RECEIVES) AND THEN EXECUTES A PROGRAM SENT FROM ANOTHER COMPUTER.

IT WAITS UNTIL IT RECEIVES AS$55 IN THE ACIA INPUT DATA (AS A SIGNAL) AND:

USES THE NEXT TWO BYTES TO TELL WHERE IN MEMORY TO START PUTTING THE PROGRAM

USES THE NEXT TWO BYTES TO TELL HOW MANY BYTES TO DOWN LOAD INTO MEMORY

USES THE NEXT TWO BYTES TO TELL WHERE IN MEMORY TO EXECUTE THE PROGRAM WHEN FINISHED DOWNLOADING

-18-
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>8E 17FF</td>
<td>LDS #$17FF</td>
</tr>
<tr>
<td>0803</td>
<td>86 03</td>
<td>LDA A #3</td>
</tr>
<tr>
<td>0805</td>
<td>B7 E800</td>
<td>STA A ACCR</td>
</tr>
</tbody>
</table>
| 0808    | 86 15       | LDA A #$15  | *WAS $9D FIRST
          |             |             | THIS ENABLED REC IRQ ($9D) |
| 080A    | B7 E800     | STA A ACCR  |
| 0810    | 84 01       | AND A #$01  |
| 0812    | 27 F9       | BEQ MAIN    |
| 0814    | B6 E801     | LDA A ACRD  | (THE ACIA DATA REGISTER) |
| 0817    | 81 55       | CMP A #$55  |
| 0819    | 26 F2       | BNE MAIN    |
| 081B    | BD F851     | JSR INPUT+E |
| 081E    | 97 00       | STA A START |
| 0820    | BD F851     | JSR INPUT+E |
| 0823    | 97 01       | STA A START+1 |
| 0825    | BD F851     | JSR INPUT+E |
| 0828    | 97 02       | STA A QUIT  |
| 082A    | BD F851     | JSR INPUT+E |
| 082D    | 9B 01       | ADD A START+1 |
| 082F    | 97 03       | STA A QUIT+1 |
| 0831    | 96 02       | LDA A QUIT  |
| 0833    | 99 00       | ADC A START |
0835 97 02  STA A  QUIT
0837 BD F851 JSR INPUT+E
0839 97 04  STA A  EX
083C BD F851 JSR INPUT+E
083F 97 05  STA A  EX+1
          *
0841 DE 00 XFER  LDX  START
0843 BD F851 XF2 JSR INPUT+E
0846 A7 00  STA A  0,X
0848 08  INX
0849 9C 02  CPX  QUIT
084B 26 F6  BNE  XF2
          *
084D DE 04  LDX  EX
084F 6E 00  JMP  0,X
          *
0851 B6 E800 INPUT  LDA A  ACSR
0854 84 01  AND A  #$01
0856 27 F9  BEQ  INPUT
0858 B6 E801 LDA A  ACRD
085B 39  RTS
            END

ACRD     E801
ACSR     E800
ACCR     E800
START    0000
QUIT     0002
EX       0004
E        F000
MAIN     080D
XFER     0841
XF2      0843
INPUT    0851

TOTAL ERRORS 00000

******************************************************************************

-20-
NAM DOWN
OPT O,S
ORG $2000

THIS PROGRAM RUNS IN SS50 6800 SYSTEM
AND DOWNLOADS (TRANSMITS) TO THE SINGLE
BOARD 6802 MICROPROCESSOR SYSTEM (PROGRAMS
WHICH THE 6802 SYSTEM WILL RUN)

LOCATIONS $1000,1001 CONTAIN THE
STARTING ADDRESS OF DESTINATION PROGRAM IN
THE 6802 SYSTEM

LOCATIONS $1002,1003 INDICATE THE
NUMBER OF BYTES TO BE TRANSFERED

LOCATIONS $1004,1005 CONTAIN THE
TRANSFER ADDRESS FOR THE DESTINATION
PROGRAM (WHERE TO EXECUTE THE PROGRAM IN
THE SINGLE BOARD 6802 SYSTEM)

LOCATIONS $1006,1007 HAVE THE STARTING
ADDRESS OF THE PROGRAM IN SS50 6800 SYSTEM

LOCATIONS $1008,1009 CONTAIN THE
ENDING ADDRESS+1 OF THE PROGRAM IN THE SS50
6800 SYSTEM

E1D1 OUT1 EQU $E1D1
1006 START EQU $1006
1008 STOP EQU $1008
2000 86 55 STARD LDA A #$55
2002 BD 203A JSR OUT

2005 B6 1000 LDA A $1000
2008 BD 203A JSR OUT
200B B6 1001 LDA A $1001
200E BD 203A    JSR    OUT

2011 B6 1002    LDA A $1002
2014 BD 203A    JSR    OUT
2017 B6 1003    LDA A $1003
201A BD 203A    JSR    OUT

201D B6 1004    LDA A $1004
2020 BD 203A    JSR    OUT
2023 B6 1005    LDA A $1005
2026 BD 203A    JSR    OUT

2029 FE 1006    LDX    START
202E BD 203A    JSR    OUT
2031 08        INX
2032 BC 1008    CPX    STOP
2035 26 F5     BNE    LP1

2037 7E 7103    JMP    $7103

203A BD E1D1    OUT    JSR    OUT1
203D 86 FF      LDA A #$FF
203F 4A         DLY2    DEC A
2040 26 FD      BNE    DLY2
2042 39         RTS

OUT1    E1D1
START   1006
STOP    1008
STARD   2000
LP1     202C
OUT     203A
DLY2    203F

TOTAL ERRORS 00000

******************************************************************************
NAM TST5

* 
* TST5 CHECKS IF SECTION ONE OF S2 IS ON OR OFF.

IF THE BIT IS ON, AN A($41) IS PUT OUT ON THE A SIDE OF IC14(PIA).

IF THE BIT IS OFF, THEN A B($42) IS PUT OUT.

THE PROGRAM RUNS IN AN ENDLESS LOOP, CONTINUOUSLY CHECKING SECTION 1 OF S2 WHICH IS CONNECTED TO PB0 OF IC14(PIA) AT $E000.

GET ASSEMBLED PROGRAM, RUN <DOWN> TO TRANSFER THE OBJECT CODE TO THE MICROPROCESSOR.

* *

OPT O,S,NOP
ORG $0100

0100 *

* CONSTANT DECLARATIONS

E000 PRA EQU $E000 * DDR IF CR(2) IS 0
E001 CRA EQU $E001
E002 PRB EQU $E002 * DDR IF CR(2) IS 0
E003 CRB EQU $E003
1000 WHERE EQU $1000 * WHERE STORED ON BOARD
1000 EXEC EQU $1000 * WHERE EXECUTED ON BOARD

* PIA INITIALIZATIONS

0100 86 FF START LDA A #$FF
0102 B7 E000 STA A PRA . * SET A SIDE FOR OUTPUT ON ALL LINES

-23-
0105 86 04 LDA A #$04
0107 B7 E001 STA A CRA * ACCESS PERIPHERAL REGISTER
010A B7 E003 STA A CRB * ACCESS PRB

* MAIN LOOP

010D B6 E002 LP1 LDA A PRB
0110 84 01 AND A #$01
0112 27 07 BEQ AR1
0114 86 41 LDA A #$41
0116 B7 E000 STA A PRA
0119 20 F2 BRA LP1
011B 86 42 AR1 LDA A #$42
011D B7 E000 STA A PRA
0120 20 EB BRA LP1

* END MAIN LOOP

* VARIABLE DECLARATIONS

0122 0001 NEXT RMB 1
0022 LENG EQU NEXT-START
0100 ORG $1000
1000 1000 FDB WHERE
1002 0022 FDB LENG
1004 1000 FDB EXEC
1006 0100 FDB START
1008 0122 FDB NEXT

END

PRA E000
CRA E001
PRB E002
CRB E003
WHERE 1000
EXEC 1000
START 0100
LP1 010D
AR1 011B
NEXT   0122
LENG   0022

TOTAL ERRORS 00000

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
NAM    TST6
*      *
*      * THE PROGRAM TST6 CHECKS TO SEE IF SECTION1 OF S2 IS SET OR NOT.

IF IT IS SET, THE HEX VALUE CORRESPONDING TO THE DIP SWITCH SETTINGS OF S2 IS
PUT OUT OVER THE ACIA AND SENT TO SECTION A OF THE PIA.

IF SECTION1 OF S2 IS NOT SET, A B($42) IS SENT TO SECTION A OF THE PIA.

THE PROGRAM RUNS IN AN ENDLESS LOOP CONTINUOUSLY CHECKING S2 SECTION 1 WHICH IS
CONNECTED TO PB0 OF IC14 (PIA) AT $E000.

AFTER GETTING THE ASSEMBLED PROGRAM, RUN <DOWN> TO TRANSFER THE OBJECT CODE
TO THE MICROPROCESSOR.

OPT O,S
ORG $0100

* CONSTANT DECLARATIONS

  E000  PRA EQU $E000   * DDR IF CR(2) IS 0
  E001  CRA EQU $E001
  E002  PRB EQU $E002   * DDR IF CR(2) IS 0
  E003  CRB EQU $E003
  E801  ACRD EQU $E801

-25-
E800  ACSR  EQU  $E800
1000  WHERE  EQU  $1000  * WHERE STORED ON BOARD
1000  EXEC  EQU  $1000  * WHERE EXECUTED ON BOARD

* PIA INITIALIZATIONS

0100  86  FF  START  LDA A  #$FF
0102  B7  E000  STA A  PRA  * SET A SIDE FOR OUTPUT
0105  86  04  LDA A  #$04
0107  B7  E001  STA A  CRA  * ACCESS PERIPHERAL REGISTER
010A  B7  E003  STA A  CRB  * ACCESSES PRB

* MAIN LOOP

010D  B6  E002  LP1  LDA A  PRB
0110  84  01  AND A  #$01
0112  27  12  BEQ  AR1
0114  B6  E002  LDA A  PRB
0117  B7  E000  STA A  PRA  *
011A  F6  E800  LP2  LDA B  ACSR
011D  C4  02  AND B  #$02
011F  27  F9  BEQ  LP2
0121  B7  E801  STA A  ACRD  *
0124  20  E7  BRA  LP1
0126  86  42  AR1  LDA A  #$42
0128  B7  E000  STA A  PRA
012B  20  E0  BRA  LP1

* END MAIN LOOP

*VARIABLE DECLARATIONS

012D  0001  NEXT  RMB  1
THE PROGRAM TST7 CHECKS SECTION 1 OF SWITCH TWO (S2).

IF THE BIT IS SET, THE HEX VALUES FOR 0-9 ARE CONTINUOUSLY OUTPUT TO THE ACIA.

IF THE BIT IS NOT SET, A B(42) IS OUTPUT TO SECTION A OF THE PIA.
THE PROGRAM RUNS IN AN ENDLESS LOOP, CONTINUOUSLY CHECKING SECTION 1 OF S2 WHICH IS CONNECTED TO PB0 OF IC14 (PIA) ADDRESSED AT MEMORY LOCATION $E000.

AFTER GETTING THE ASSEMBLED PROGRAM, RUN <DOWN> TO TRANSFER THE OBJECT CODE TO THE MICROPROCESSOR.

```
OPT  O,S,NOP
ORG  $0100

* CONSTANT DECLARATIONS

E000  PRA  EQU  $E000  * DDR IF CR(2) IS 0
E001  CRA  EQU  $E001
E002  PRB  EQU  $E002  * DDR IF CR(2) IS 0
E003  CRB  EQU  $E003
E801  ACRD EQU $E801
E800  ACSR EQU $E800 1000  WHERE EQU  $1000  * WHERE STORED ON BOARD
1000  EXEC EQU  $1000  * WHERE EXECUTED ON BOARD

* PIA INITIALIZATIONS

0100  86  FF  START  LDA A  #$FF
0102  B7  E000  STA A  PRA  * SET A SIDE FOR OUTPUT
0105  86  04  LDA A  #$04
0107  B7  E001  STA A  CRA  * ACCESS PERIPHERAL REGISTER
010A  B7  E003  STA A  CRB  * ACCESSSES PRB

* MAIN LOOP

010D  30  NUM  FCB  $30
```
010E B6 E002 LP1  LDA A   PRB
0111 84 01       AND A   #$01
0113 27 1F       BEQ    AR1
                   * LDA A   PRB
                   * STAA   PRA
0115 B6 010D     LDA A   NUM
                   *
0118 F6 E800 LP2  LDA B   ACSR
011B C4 02       AND B   #$02
011D 27 F9       BEQ    LP2
011F B7 E801     STA A   ACRD
                   *INC NUM COMP TO 39
0122 B6 010D     LDA A   NUM
0125 4C          INC A
0126 B7 010D     STA A   NUM
0129 81 3A       CMP A   #$3A
012B 26 05       BNE    AR4
012D 86 30       LDA A   #$30
012F B7 010D     STA A   NUM
                   *
0132 20 DA AR4    BRA    LP1
0134 86 42 AR1    LDA A   #$42
0136 B7 E000     STA A   PRA
0139 20 D3       BRA    LP1
                   * END MAIN LOOP
                   *
*VARIABLE DECLARATIONS

013B 0001 NEXT   RMB    1
003B LENG       EQU    NEXT-START
1000 ORG        $1000
1000 1000       FDB    WHERE
1002 003B       FDB    LENG
1004 1000       FDB    EXEC
1006 0100       FDB    START
1008 013B       FDB    NEXT
                    END
PRA E000
CRA E001
PRB E002
CRB E003
ACRD E801
ACSR E800
WHERE 1000
EXEC 1000
START 0100
NUM 010D
LP1 010E
LP2 0118
AR4 0132
AR1 0134
NEXT 013B
LENG 003B

TOTAL ERRORS 00000

****************************

NAM TST8
*
  PURPOSE: THE PROGRAM TST8 PUTS OUT NUMBERS 0-9, CR, LF AND HAS A TIME DELAY SO THAT THERE IS NO DATA LOSS WHEN TRANSFERRING TO AN IBM PC(AT).

GET THE ASSEMBLED PROGRAM, RUN <DOWN> TO TRANSFER THE OBJECT CODE TO THE MICROPROCESSOR.

0100
OPT O,S,NOP
ORG $0100

-30-
* CONSTANT DECLARATIONS

0F00 E EQU $F00  *ASMB AT $0100 TO RUN AT $100
E000 PRA EQU $E000  *(DDR) IF CR(2) IS 0
E001 CRA EQU $E001
E002 PRB EQU $E002  * DDR IF CR(2) IS 0
E003 CRB EQU $E003
E801 ACDR EQU $E801
E800 ACSR EQU $E800
1000 WHERE EQU $1000  * WHERE STORED ON BOARD
1000 EXEC EQU $1000  * WHERE EXECUTED ON BOARD
E002 SW2 EQU $E002  * DIP SWITCH 2 USED FOR INPUT

* PIA INITIALIZATIONS

0100 86 FF START LDA A #$FF
0102 B7 E000 STA A PRA  * SET A SIDE FOR OUTPUT
0105 86 04 LDA A #$04
0107 B7 E001 STA A CRA  * ACCESS PERIPHERAL REGISTER
010A B7 E003 STA A CRB  * ACCESSES PRB

* MAIN LOOP

010D B6 E002 LP1 LDA A PRB
0110 85 01 BIT A #$01
0112 27 F9 BEQ LP1
0114 B6 0156 LDA A NUM
0117 BD 1031 JSR OUTA+E
011A 7C 0156 INC NUM
011D 81 39 CMP A #$39
011F 26 EC BNE LP1
0121 BD 104B JSR CRLF+E
0124 86 30 LDA A #$30

-31-
STA A NUM
LDX #$2000
JSR DLY1+E
BRA LP1

* END MAIN LOOP *

OUTA PUTS OUT ON THE ACIA WHAT IS IN THE A ACCUMULATOR, DESTROYS THE B ACCUMULATOR, RETURNS NOTHING, AND PRESERVES THE INDEX REGISTER

LDA B ACSR
BIT B #$02
BEQ OUTA
STA A ACDR
RTS

INA INPUTS A CHARACTER FROM THE ACIA INTO THE A ACCUMULATOR, PRESERVES THE B ACCUMULATOR, AND PRESERVES THE INDEX REGISTER

LDA B ACSR
AND B #$01
BEQ INA
LDA A ACDR
RTS

DLY1 TAKES WHAT IS IN THE X REGISTER AND DECREMENTS IT TO ZERO CREATING A TIME DELAY AND DESTROYING WHATS IN THE INDEX REGISTER

DEX
BNE DLY1
RTS
CRLF PUTS OUT CARRIAGE RETURN & LINE FEED

TO ACIA
014B 86 0D CRLF LDA A #$0D
014D BD 1031 JSR OUTA+E
0150 86 0A LDA A #$0A
0152 BD 1031 JSR OUTA+E
0155 39 RTS

VARIABLE DECLARATIONS

0156 30 NUM FCB $30
0157 07 CNT2 FCB $7
0158 0001 NEXT RMB 1
 0058 LENG EQU NEXT-START
1000 ORG $1000
1000 1000 FDB WHERE
1002 0058 FDB LENG
1004 1000 FDB EXEC
1006 0100 FDB START
1008 0158 FDB NEXT
  
* END

E 0F00
PRA E000
CRA E001
PRB E002
CRB E003
ACDR E801
ACSR E800
WHERE 1000
EXEC 1000
SW2 E002
START 0100
LP1 010D
OUTA 0131
INA 013C
DLY1 0147
CRLF 014B

-33-
NUM 0156
CNT2 0157
NEXT 0158
LENG 0058

TOTAL ERRORS 00000

*****************************************************************************

NAM TST10
*

PURPOSE: PUTS OUT 0-9, CR & LF, LINE NUMBERS WITH ENOUGH DELAY SO NO CHARACTERS LOST BY THE PC(AT) ON THE RECEIVING END.

********+++PUTS GARBAGE OUT++++++++*
BECAUSE COUNTER NEEDS OFFSET IN MEMORY.

GET THE ASSEMBLED PROGRAM, RUN <DOWN> TO TRANSFER THE OBJECT CODE TO THE MICROPROCESSOR.

GPT O,S,NOP
ORG $0100

CONSTANT DECLARATIONS

0F00 E EQU $F00  * ASMB AT $0100 TO RUN AT $100
E000 PRA EQU $E000  * DDR IF CR(2) IS 0
E001 CRA EQU $E001
E002 PRB EQU $E002  * DDR IF CR(2) IS 0
E003 CRB EQU $E003
E801 ACDR EQU $E801
E800 ACSR EQU $E800
1000 WHERE EQU $1000  * WHERE STORED ON BOARD

-34-
1000  EXEC  EQU  $1000  * WHERE EXECUTED ON BOARD
E002  SW2  EQU  $E002  * DIP SWITCH 2 USED FOR INPUT

* PIA INITIALIZATIONS

0100  86  FF  START  LDA  A  #$FF  * SET A SIDE FOR OUTPUT
0102  B7  E000  STA  A  PRA
0105  86  04  LDA  A  #$04
0107  B7  E001  STA  A  CRA  * ACCESS THE A PERIPHERAL REGISTER
010A  B7  E003  STA  A  CRB  * ACCESSES PRB

* MAIN LOOP

* LP1
010D  B6  E002  LDA  A  PRB
0110  85  01  BIT  A  #$01
0112  27  F9  BEQ  LP1
0114  B6  016C  LDA  A  CNT2  * PUT LINE NUMBERS ON LINES
0117  8B  30  ADD  A  #$30
0119  BD  104B  JSR  OUTA+E
011C  86  20  LDA  A  #$20
011E  BD  104B  JSR  OUTA+E
0121  B6  016B  LDA  A  CNT1  * PRINT OUT STRING
0124  BD  104B  JSR  OUTA+E
0127  7C  016B  INC  CNT1
012A  81  39  CMP  A  #$39
012C  26  F3  BNE  LP2
012E  BD  1065  JSR  CRLF+E
0131  86  30  LDA  A  #$30
0133  B7  016B  STA  A  CNT1
0136  CE  2000  LDX  #$2000
0139  BD  1061  JSR  DLY1+E
013C  7C  016C  INC  CNT2

-35-
013F F6 016C       LDA B  CNT2
0142 C1 0A         CMP B  #$A
0144 26 C7         BNE   LP1
0146 7F 016C       CLR   CNT2
0149 20 C2         BRA   LP1

* END MAIN LOOP

OUTA PUTS OUT ON THE ACIA WHAT IS IN
THE A ACCUMULATOR, DESTROYS THE B
ACCUMULATOR, RETURNS NOTHING, AND PRESERVES
THE INDEX REGISTER

014B F6 E800 OUTA LDA B  ACSR
014E C5 02         BIT B  #$02
0150 27 F9         BEQ   OUTA
0152 B7 E801       STA A  ACDR
0155 39            RTS

INA INPUTS A CHARACTER FROM THE ACIA
INTO THE A ACCUMULATOR, PRESERVES THE B
ACCUMULATOR, AND PRESERVES THE INDEX
REGISTER

0156 F6 E800 INA  LDA B  ACSR
0159 C4 01         AND B  #$01
015B 27 F9         BEQ   INA
015D B6 E801       LDA A  ACDR
0160 39            RTS

DLY1 TAKES WHAT IS IN THE INDEX
REGISTER AND DECREMENTS IT TO ZERO CREATING
A TIME DELAY AND DESTROYING WHATS IN THE
INDEX REGISTER

0161 09            DLY1  DEX
0162 26 FD         BNE   DLY1
0164 39            RTS
CRLF PUTS OUT CARRIAGE RETURN & LINE
FEED
ONLY PUTS OUT CARRIAGE RETURN BECAUSE
PC UNDERSTANDS CARRIAGE RETURN AS CARRIAGE
RETURN AND LINE FEED.

0165 86 0D CRLF LDA A #$0D
0167 BD 104B JSR OUTA+E
  *LDAA #$0A
  *JSR OUTA+E
016A 39 RTS

*VARIABLE DECLARATIONS

016B 30 CNT1 FCB $30
016C 00 CNT2 FCB $0
016D 0001 NEXT RMB 1
  006D LENG EQU NEXT-START
1000 ORG $1000
1000 1000 FDB WHERE
1002 006D FDB LENG
1004 1000 FDB EXEC
1006 0100 FDB START
1008 016D FDB NEXT

*END

E 0F00
PRA E000
CRA E001
PRB E002
CRB E003
ACDR E801
ACSR E800
WHERE 1000
EXEC 1000
SW2 E002
START 0100
LP1 010D
TOTAL ERRORS 00000

*****************************************************************

NAM TST12

PURPOSE OF PROGRAM : SEND STRING OF CHARACTERS TO PC WITH LINE NUMBERS.

METHOD :

CHECK SWITCH ** OUTPUT <CR>AAA<CR>

PUT LINE NUMBERS ON LINES

PRINT STRING

REINITIALIZE CHAR COUNT TO 0

DELAY #$2000

INCREMENT LINE NUMBER

CHECK SWITCH, BRANCH IF STILL ON

OUTPUT BBB SO PC KNOWS IT IS THE END OF TRANSFER

DELAY #$FFFF
BACK TO BEGINNING, WAIT FOR SWITCH

GET ASSEMBLED PROGRAM, RUN <DOWN>
TO TRANSFER THE OBJECT CODE TO THE
MICROPROCESSOR.

OPT O,S,NOP
ORG $0100

* CONSTANT DECLARATIONS

0F00 E EQU $F00  * ASMB AT $0100 RUN
                 AT $1000
E000 PRA EQU $E000  * DDR IF CR(2) IS 0
E001 CRA EQU $E001
E002 PRB EQU $E002  * DDR IF CR(2) IS 0
E003 CRB EQU $E003
E801 ACDR EQU $E801
E800 ACSR EQU $E800
1000 WHERE EQU $1000  * WHERE STORED ON
                    BOARD
1000 EXEC EQU $1000  * WHERE EXECUTED ON
                    BOARD
E002 SW2 EQU $E002  * DIP SWITCH 2 USED
                    FOR CONTROL INPUT

*  
* PIA INITIALIZATIONS

0100 86 FF START LDA A #$FF
0102 B7 E000 STA A PRA  * SET A SIDE FOR
                         OUTPUT
0105 86 04 LDA A #$04
0107 B7 E001 STA A CRA  * ACCESS PERIPHERAL
                         REGISTER
010A B7 E003 STA A CRB  * ACCESSES PRB
* MAIN LOOP

* CHECK SWITCH

010D B6 E002 LP LDA A PRB
0110 85 01 BIT A #$01
0112 27 F9 BEQ LP
0114 86 0D LDA A #$0D
0116 BD 1083 JSR OUTA+E
0119 86 41 LDA A #$41
011B BD 1083 JSR OUTA+E
011E BD 1083 JSR OUTA+E
0121 BD 1083 JSR OUTA+E
0124 86 0D LDA A #$0D
0126 BD 1083 JSR OUTA+E

* PUT LINE NUMBERS ON LINES

0129 B6 10A4 LP1 LDA A CNT2+E
012C 8B 30 ADD A #$30
012E BD 1083 JSR OUTA+E
0131 86 20 LDA A #$20
0133 BD 1083 JSR OUTA+E
0136 B6 10A3 LP2 LDA A CNT1+E
0139 BD 1083 JSR OUTA+E
013C 7C 10A3 INC CNT1+E
013F 81 39 CMP A #$39
0141 26 F3 BNE LP2
0143 BD 109D JSR CRLF+E
0146 86 30 LDA A #$30

* PRINT OUT STRING

0148 B7 10A3 STA A CNT1+E
014B CE 2000 LDX #$2000
014E BD 1099 JSR DLTY1+E
0151 7C 10A4 INC CNT2+E

* REINITIALIZE COUNT TO 0

0154 F6 10A4 LDA B CNT2+E
0157 C1 0A CMP B #$A
0159 26 CE BNE LP1
015B 7F 10A4 CLR CNT2+E
015E B6 E002 LDA A PRB

* CHECK IF SWITCH IS STILL ON

-40-
0161 85 01 BIT A #$01 * IF SWITCH ON CHECK AGAIN
0163 26 C4 BNE LP1
0165 86 0D LDA A #$0D
0167 BD 1083 JSR OUTA+E
016A 86 42 LDA A #$42 * SEND BBB AT END OF XMIT
016C BD 1083 JSR OUTA+E
016F BD 1083 JSR OUTA+E
0172 BD 1083 JSR OUTA+E
0175 86 0D LDA A #$0D * CARRIAGE RETURN
0177 BD 1083 JSR OUTA+E
017A CE FFFF LDX #$FFFF * DELAY BEFORE BRANCHING BACK
017D BD 1099 JSR DLY1+E
0180 7E 100D JMP LP+E
END MAIN LOOP

OUTA PUTS OUT ON THE ACIA WHAT IS IN THE A ACCUMULATOR, DESTROYS THE B ACCUMULATOR, RETURNS NOTHING AND PRESERVES THE INDEX REGISTER

0183 F6 E800 OUTA LDA B ACSR
0186 C5 02 BIT B #$02
0188 27 F9 BEQ OUTA
018A B7 E801 STA A ACDR
018D 39 RTS

INA INPUTS A CHARACTER FROM THE ACIA INTO THE A ACCUMULATOR, PRESERVES THE B ACCUMULATOR AND PRESERVES THE INDEX REGISTER

018E F6 E800 INA LDA B ACSR
DLY1 takes what is in the index register and decrements it to zero creating a time delay and destroying what is in the index register.

CRLF puts out carriage return & line feed to ACIA.

Only puts out carriage return because PC understands carriage return as carriage return and line feed.

*VARIABLE DECLARATIONS

CNT1 FCB $30
CNT2 FCB $0
NEXT RMB 1
LENG EQU NEXT-START
ORG $1000
FDB WHERE
FDB LENG
FDB EXEC
FDB START
FDB NEXT
E 0F00
PRA E000
CRA E001
PRB E002
CRB E003
ACDR E801
ACSR E800
WHERE 1000
EXEC 1000
SW2 E002
START 0100
LP 010D
LP1 0129
LP2 0136
OUTA 0183
INA 018E
DLY1 0199
CRLF 019D
CNT1 01A3
CNT2 01A4
NEXT 01A5
LENG 00A5

TOTAL ERRORS 00000

****************************************************************************************

NAM TST13

THIS PROGRAM TRANSFERS ASCII CHARACTERS COMING IN ON THE ACIA OUT TO A PC(AT) VIA ACIA (RS-232-C)
IT ONLY TRANSFERS DATA IF THE PROPER SWITCH IS THROWN
IT DOES NOT TRANSFER NULLS (.00)
IT DOES NOT TRANSFER LINE FEEDS ($0A)
GET ASSEMBLED PROGRAM, RUN <DOWN>
TO TRANSFER THE OBJECT CODE TO THE
MICROPROCESSOR.

OPT O,S,NOP
ORG $0100

* CONSTANT DECLARATIONS

0F00 E EQU $F00 * ASMB AT $0100 TO
       RUN $1000
E000 PRA EQU $E000 * DDR IF CR(2) IS 0
E001 CRA EQU $E001
E002 PRB EQU $E002 * DDR IF CR(2) IS 0
E003 CRB EQU $E003
E801 ACDR EQU $E801
E800 ACSR EQU $E800
1000 WHERE EQU $1000 * WHERE STORED ON
       BOARD
1000 EXEC EQU $1000 * WHERE EXECUTED ON
       BOARD
E002 SW2 EQU $E002 * DIP SWITCH 2 USED
       FOR TRANSMIT
       CONTROL INPUT

PIA INITIALIZATIONS

0100 86 FF START LDA A #$FF
0102 B7 E000 STA A PRA * SET A SIDE FOR
       OUTPUT
0105 86 04 LDA A #$34
0107 B7 E001 STA A CRA * ACCESS PERIPHERAL
       REGISTER A
010A B7 E003 STA A CRB * ACCESS PR-B

MAIN LOOP

010D B6 E002 LP LDA A PRB * CHECK SWITCH
0110 85 01       BIT A #$01
0112 27 F9       BEQ LP

0114 BD 1030 MAIN JSR INA+E * INPUT A CHARACTER
0117 81 00       CMP A #$00 * CHECK IF CHARACTER IS NULL

0119 27 F9       BEQ MAIN
011B 81 0A       CMP A #$A * IS CHARACTER A LINE FEED

011D 27 F5       BEQ MAIN
011F BD 1025 JSR OUTA+E * OUTPUT A CHARACTER
0122 7E 100D JMP LP+E

END MAIN LOOP

OUTA PUTS OUT ON THE ACIA WHAT IS IN THE A ACCUMULATOR, DESTROYS THE B ACCUMULATOR, RETURNS NOTHING, AND PRESERVES THE INDEX REGISTER

0125 F6 E800 OUTA LDA B ACSR
0128 C5 02       BIT B #$02
012A 27 F9       BEQ OUTA
012C B7 E801 STA A ACDR
012F 39         RTS

INA INPUTS A CHARACTER FROM THE ACIA INTO THE A ACCUMULATOR, PRESERVES THE B ACCUMULATOR AND PRESERVES THE INDEX REGISTER

0130 F6 E800 INA LDA B ACSR
0133 C4 01       AND B #$01
0135 27 F9       BEQ INA
0137 B6 E801 LDA A ACDR
013A 39         RTS

-45-
DLY1 TAKES WHAT IS IN THE INDEX REGISTER AND DECREMENTS IT TO ZERO TO CREATE A TIME DELAY AND DESTROYS WHAT IS IN THE INDEX REGISTER

013B 09 DLY1 DEX
013C 26 FD BNE DLY1
013E 39 RTS

CRLF PUTS OUT CARRIAGE RETURN & LINE FEED TO ACIA
ONLY PUTS OUT CARRIAGE RETURN BECAUSE PC UNDERSTANDS CARRIAGE RETURN AS CARRIAGE RETURN AND LINE FEED

013F 86 0D CRLF LDA A #$0D
0141 BD 1025 JSR OUTA+E
*LDAA #$0A NOT USED
*JSR OUTA+E NOT USED
0144 39 RTS

VARIABLE DECLARATIONS

0145 0001 NEXT RMB 1
0045 LENG EQU NEXT-START
1000 ORG $1000
1000 1000 FDB WHERE
1002 0045 FDB LENG
1004 1000 FDB EXEC
1006 0100 FDB START
1008 0145 FDB NEXT
*

END

E 0F00
PRA E000
CRA' E001
PRB E002
CRB E003
ACDR E801
ACSR   E800
WHERE  1000
EXEC   1000
SW2    E002
START  0100
LP     010D
MAIN   0114
OUTA   0125
INA    0130
DLY1   013B
CRLF   013F
NEXT   0145
LENG   0045

TOTAL ERRORS 00000

*******************************************************************************************************************
The single board 6802 based microprocessor system was constructed on a Vector 8804 S-100 circuit card. The S-100 edge connector was not used (the connector was sheared off even with the rest of the board). The Vector board is very desirable because it provides a full conductive grid on both the top and bottom of the board. The top grid (component side) is connected to the regulated plus five volt supply. The bottom grid is ground. The system was constructed with wire wrap sockets (three wrap length), and 30 gauge wire wrap wire (modified wrap).

The wire wrap list for this computer system was generated from the schematic diagram on page 13 (figure number 3). The following method was used to create the wire wrap list and check its accuracy:

STARTING WITH PIN-1 IC-1 A WIRE IS TRACED IN RED ON THE SCHEMATIC DIAGRAM, AND LISTED ON THE WIRE WRAP LIST AFTER IC-1 PIN-1.

IF IT GOES TO A WIRE WRAP PIN, AN "X" IS PLACED AFTER THAT PIN ON THE LIST TO INDICATE THAT A WIRE HAS BEEN RUN THERE (BUT IS LISTED ELSEWHERE).

IF MORE WIRES ARE NEEDED FROM IC-1 PIN-1 THEY ARE MARKED IN RED AND LISTED ON THE LIST ALSO.

WHEN DRAWING THE RED LINES ON THE SCHEMATIC DIAGRAM IT HELPS TO PUT HEAD AND TAIL ARROWS ON THE LINE TO SHOW WHICH WAY IT WAS RUN (WHERE IT WOULD BE LISTED ON THE WIRE WRAP LIST).

DO NOT OVERLAP RED LINES TO THE SAME LOCATION (ON THE SCHEMATIC) SO THAT THE INDIVIDUAL WIRE WRAP WIRES MAY BE DISTINGUISHED.

CHECK THE WIRE WRAP LIST BEFORE ACTUAL CONSTRUCTION BY TAKING A FRESH COPY OF THE SCHEMATIC AND PUT IN RED LINES CORRESPONDING TO THE LIST. ANY BARE BLACK LINES REPRESENT ERRORS IN THE LIST.

Two PC programs written in C are included on page seventy one to assist with the construction of wire wrap lists. The first (WWLIST) creates a blank list which is filled in by hand from the schematic. The second (UPDATEWW) allows easy entry of the hand written data (where to information) into a computer file where it can be duplicated or searched conveniently. It is very convenient to have two persons when creating and updating the wire wrap list.
IC 1 Name 5517 Pins 24

( ) ( ) IC 1 -PIN 1 TO IC2 -PIN 1 & IC5 PIN 1
( ) ( ) IC 1 -PIN 2 TO IC2 -PIN 2 & IC5 PIN 2
( ) ( ) IC 1 -PIN 3 TO IC2 -PIN 3 & IC5 PIN 3
( ) ( ) IC 1 -PIN 4 TO IC2 -PIN 4 & IC5 PIN 4
( ) ( ) IC 1 -PIN 5 TO IC2 -PIN 5 & IC5 PIN 5
( ) ( ) IC 1 -PIN 6 TO IC2 -PIN 6 & IC5 PIN 6
( ) ( ) IC 1 -PIN 7 TO IC2 -PIN 7 & IC5 PIN 7
( ) ( ) IC 1 -PIN 8 TO IC2 -PIN 8 & IC5 PIN 8
( ) ( ) IC 1 -PIN 9 TO IC2 -PIN 9 & IC5 PIN 9
( ) ( ) IC 1 -PIN 10 TO IC2 -PIN 10 & IC5 PIN 10
( ) ( ) IC 1 -PIN 11 TO IC2 -PIN 11 & IC5 PIN 11
( ) ( ) IC 1 -PIN 12 TO ------- GND
( ) ( ) IC 1 -PIN 13 TO IC2 -PIN 13 & IC5 PIN 13
( ) ( ) IC 1 -PIN 14 TO IC2 -PIN 14 & IC5 PIN 14
( ) ( ) IC 1 -PIN 15 TO IC2 -PIN 15 & IC5 PIN 15
( ) ( ) IC 1 -PIN 16 TO IC2 -PIN 16 & IC5 PIN 16
( ) ( ) IC 1 -PIN 17 TO IC2 -PIN 17 & IC5 PIN 17
( ) ( ) IC 1 -PIN 18 TO IC1 -PIN 20
( ) ( ) IC 1 -PIN 19 TO IC2 -PIN 19 & IC5 PIN 19
( ) ( ) IC 1 -PIN 20 TO IC9 -PIN 3 X
( ) ( ) IC 1 -PIN 21 TO IC2 -PIN 21
( ) ( ) IC 1 -PIN 22 TO IC2 -PIN 22 & IC5 PIN 22
( ) ( ) IC 1 -PIN 23 TO IC2 -PIN 23 & IC5 PIN 23
( ) ( ) IC 1 -PIN 24 TO ------- +5V
IC 2 Name 5517 Pins 24

( ) ( ) IC 2 -PIN 1 TO IC3 -PIN 1 X
( ) ( ) IC 2 -PIN 2 TO IC3 -PIN 2 X
( ) ( ) IC 2 -PIN 3 TO IC3 -PIN 3 X
( ) ( ) IC 2 -PIN 4 TO IC3 -PIN 4 X
( ) ( ) IC 2 -PIN 5 TO IC3 -PIN 5 X
( ) ( ) IC 2 -PIN 6 TO IC3 -PIN 6 X
( ) ( ) IC 2 -PIN 7 TO IC3 -PIN 7 X
( ) ( ) IC 2 -PIN 8 TO IC3 -PIN 8 X
( ) ( ) IC 2 -PIN 9 TO IC3 -PIN 9 X
( ) ( ) IC 2 -PIN 10 TO IC3 -PIN 10 X
( ) ( ) IC 2 -PIN 11 TO IC3 -PIN 11 X
( ) ( ) IC 2 -PIN 12 TO -------- GND
( ) ( ) IC 2 -PIN 13 TO IC3 -PIN 13 X
( ) ( ) IC 2 -PIN 14 TO IC3 -PIN 14 X
( ) ( ) IC 2 -PIN 15 TO IC3 -PIN 15 X
( ) ( ) IC 2 -PIN 16 TO IC3 -PIN 16 X
( ) ( ) IC 2 -PIN 17 TO IC3 -PIN 17 X
( ) ( ) IC 2 -PIN 18 TO IC2 -PIN 20
( ) ( ) IC 2 -PIN 19 TO IC3 -PIN 19 X
( ) ( ) IC 2 -PIN 20 TO IC9 -PIN 2 X
( ) ( ) IC 2 -PIN 21 TO IC3 -PIN 21 X
( ) ( ) IC 2 -PIN 22 TO IC3 -PIN 22 X
( ) ( ) IC 2 -PIN 23 TO IC3 -PIN 23 X
( ) ( ) IC 2 -PIN 24 TO-------- +5V

-50-
IC 3  Name 5517  Pins 24

( ) ( )  IC 3 -PIN 1 TO IC7 -PIN 3 X
( ) ( )  IC 3 -PIN 2 TO IC7 -PIN 5 X
( ) ( )  IC 3 -PIN 3 TO IC7 -PIN 7 X
( ) ( )  IC 3 -PIN 4 TO IC7 -PIN 9 X
( ) ( )  IC 3 -PIN 5 TO IC7 -PIN 12 X
( ) ( )  IC 3 -PIN 6 TO IC7 -PIN 14 X
( ) ( )  IC 3 -PIN 7 TO IC7 -PIN 16 X & IC14 PIN 35
( ) ( )  IC 3 -PIN 8 TO IC7 -PIN 18 X & IC18 PIN 11
( ) ( )  IC 3 -PIN 9 TO IC8 -PIN 18 X
( ) ( )  IC 3 -PIN 10 TO IC8 -PIN 17 X
( ) ( )  IC 3 -PIN 11 TO IC8 -PIN 16 X
( ) ( )  IC 3 -PIN 12 TO--------- GND
( ) ( )  IC 3 -PIN 13 TO IC8 -PIN 15 X
( ) ( )  IC 3 -PIN 14 TO IC8 -PIN 14 X
( ) ( )  IC 3 -PIN 15 TO IC8 -PIN 13 X
( ) ( )  IC 3 -PIN 16 TO IC8 -PIN 12 X
( ) ( )  IC 3 -PIN 17 TO IC8 -PIN 11 X
( ) ( )  IC 3 -PIN 18 TO IC3 -PIN 20
( ) ( )  IC 3 -PIN 19 TO IC6 -PIN 14 X
( ) ( )  IC 3 -PIN 20 TO IC9 -PIN 1 X
( ) ( )  IC 3 -PIN 21 TO IC12-PIN 3 X
( ) ( )  IC 3 -PIN 22 TO IC6 -PIN 16 X
( ) ( )  IC 3 -PIN 23 TO IC6 -PIN 18 X
( ) ( )  IC 3 -PIN 24 TO--------- +5V

-51-
IC 4  Name MC6802     Pins 40

( ) ( ) IC 4  -PIN 1 TO ---------  GND
( ) ( ) IC 4  -PIN 2 TO IC4  -PIN 3 & IC4  PIN35
( ) ( ) IC 4  -PIN 3 TO IC4  -PIN 8 X
( ) ( ) IC 4  -PIN 4 TO IC18-PIN 7 & 2.7 OHM PULLUP +5V
( ) ( ) IC 4  -PIN 5 TO IC11-PIN 1
( ) ( ) IC 4  -PIN 6 TO IC4  -PIN 8
( ) ( ) IC 4  -PIN 7 TO --------- NC
( ) ( ) IC 4  -PIN 8 TO --------- X +5V
( ) ( ) IC 4  -PIN 9 TO IC7  -PIN 2
( ) ( ) IC 4  -PIN 10 TO IC7  -PIN 4
( ) ( ) IC 4  -PIN 11 TO IC7  -PIN 6
( ) ( ) IC 4  -PIN 12 TO IC7  -PIN 8
( ) ( ) IC 4  -PIN 13 TO IC7  -PIN 11
( ) ( ) IC 4  -PIN 14 TO IC7  -PIN 13
( ) ( ) IC 4  -PIN 15 TO IC7  -PIN 15
( ) ( ) IC 4  -PIN 16 TO IC7  -PIN 17
( ) ( ) IC 4  -PIN 17 TO IC6  -PIN 2
( ) ( ) IC 4  -PIN 18 TO IC6  -PIN 4
( ) ( ) IC 4  -PIN 19 TO IC6  -PIN 6
( ) ( ) IC 4  -PIN 20 TO IC6  -PIN 8
( ) ( ) IC 4  -PIN 21 TO --------- GND
( ) ( ) IC 4  -PIN 22 TO IC6  -PIN 11
( ) ( ) IC 4  -PIN 23 TO IC6  -PIN 13
( ) ( ) IC 4  -PIN 24 TO IC6  -PIN 15
( ) ( ) IC 4  -PIN 25 TO IC6  -PIN 17
( ) ( ) IC 4  -PIN 26 TO IC8  -PIN 9
( ) ( ) IC 4  -PIN 27 TO IC8  -PIN 8
( ) ( ) IC 4  -PIN 28 TO IC8  -PIN 7
( ) ( ) IC 4  -PIN 29 TO IC8  -PIN 6
( ) ( ) IC 4  -PIN 30 TO IC8  -PIN 5
( ) ( ) IC 4  -PIN 31 TO IC8  -PIN 4
( ) ( ) IC 4  -PIN 32 TO IC8  -PIN 3
( ) ( ) IC 4  -PIN 33 TO IC8  -PIN 2
( ) ( ) IC 4  -PIN 34 TO IC16-PIN 15
( ) ( ) IC 4  -PIN 35 TO --------- X
( ) ( ) IC 4  -PIN 36 TO --------- GND
( ) ( ) IC 4  -PIN 37 TO IC16-PIN 13
( ) ( ) IC 4  -PIN 38 TO --------- 4 MHZ XTAL BYPASS 10PF TO GND
( ) ( ) IC 4  -PIN 39 TO --------- 4 MHZ XTAL BYPASS 10PF TO GND
( ) ( ) IC 4  -PIN 40 TO IC14-PIN 34 & 2.7K OHM PULL UP TO +5V 22 MFD IN PARALLEL W/SPST SW TO GND

-52-
IC 5  Name 2716  Pins 24

( ) ( ) IC 5 -PIN 1 TO IC -PIN X
( ) ( ) IC 5 -PIN 2 TO IC -PIN X
( ) ( ) IC 5 -PIN 3 TO IC -PIN X
( ) ( ) IC 5 -PIN 4 TO IC -PIN X
( ) ( ) IC 5 -PIN 5 TO IC -PIN X
( ) ( ) IC 5 -PIN 6 TO IC -PIN X
( ) ( ) IC 5 -PIN 7 TO IC -PIN X
( ) ( ) IC 5 -PIN 8 TO IC -PIN X
( ) ( ) IC 5 -PIN 9 TO IC18-PIN 22 X
( ) ( ) IC 5 -PIN 10 TO IC18-PIN 21 X
( ) ( ) IC 5 -PIN 11 TO IC18-PIN 20 X
( ) ( ) IC 5 -PIN 12 TO---------- GND
( ) ( ) IC 5 -PIN 13 TO IC18-PIN 19 X
( ) ( ) IC 5 -PIN 14 TO IC18-PIN 18 X
( ) ( ) IC 5 -PIN 15 TO IC18-PIN 17 X
( ) ( ) IC 5 -PIN 16 TO IC18-PIN 16 X
( ) ( ) IC 5 -PIN 17 TO IC18-PIN 15 X
( ) ( ) IC 5 -PIN 18 TO---------- GND
( ) ( ) IC 5 -PIN 19 TO IC -PIN X
( ) ( ) IC 5 -PIN 20 TO IC13-PIN 17
( ) ( ) IC 5 -PIN 21 TO IC5 -PIN 24
( ) ( ) IC 5 -PIN 22 TO IC -PIN X
( ) ( ) IC 5 -PIN 23 TO IC -PIN X
( ) ( ) IC 5 -PIN 24 TO---------- +5V
IC 6   Name 74241   Pins 20

( ) ( ) IC 6 -PIN 1 TO -------- GND
( ) ( ) IC 6 -PIN 2 TO IC -PIN X
( ) ( ) IC 6 -PIN 3 TO IC9 -PIN 18 & IC12 PIN 1
( ) ( ) IC 6 -PIN 4 TO IC -PIN X
( ) ( ) IC 6 -PIN 5 TO IC9 -PIN 20
( ) ( ) IC 6 -PIN 6 TO IC -PIN X
( ) ( ) IC 6 -PIN 7 TO IC9 -PIN 21
( ) ( ) IC 6 -PIN 8 TO IC -PIN X
( ) ( ) IC 6 -PIN 9 TO IC9 -PIN 22
( ) ( ) IC 6 -PIN 10 TO--------GND
( ) ( ) IC 6 -PIN 11 TO IC -PIN X
( ) ( ) IC 6 -PIN 12 TO IC9 -PIN 23
( ) ( ) IC 6 -PIN 13 TO IC -PIN X
( ) ( ) IC 6 -PIN 14 TO IC -PIN X
( ) ( ) IC 6 -PIN 15 TO IC -PIN X
( ) ( ) IC 6 -PIN 16 TO IC -PIN X
( ) ( ) IC 6 -PIN 17 TO IC -PIN X
( ) ( ) IC 6 -PIN 18 TO IC -PIN X
( ) ( ) IC 6 -PIN 19 TO IC6 -PIN 20
( ) ( ) IC 6 -PIN 20 TO-------- +5V X
IC 7    Name 74241    Pins 20

( ) ( ) IC 7 -PIN 1 TO --------- GND
( ) ( ) IC 7 -PIN 2 TO IC -PIN X
( ) ( ) IC 7 -PIN 3 TO IC -PIN X
( ) ( ) IC 7 -PIN 4 TO IC -PIN X
( ) ( ) IC 7 -PIN 5 TO IC -PIN X
( ) ( ) IC 7 -PIN 6 TO IC -PIN X
( ) ( ) IC 7 -PIN 7 TO IC -PIN X
( ) ( ) IC 7 -PIN 8 TO IC -PIN X
( ) ( ) IC 7 -PIN 9 TO IC -PIN X
( ) ( ) IC 7 -PIN 10 TO--------- X
( ) ( ) IC 7 -PIN 11 TO IC -PIN X
( ) ( ) IC 7 -PIN 12 TO IC -PIN X
( ) ( ) IC 7 -PIN 13 TO IC -PIN X
( ) ( ) IC 7 -PIN 14 TO IC -PIN X
( ) ( ) IC 7 -PIN 15 TO IC -PIN X
( ) ( ) IC 7 -PIN 16 TO IC -PIN X
( ) ( ) IC 7 -PIN 17 TO IC -PIN X
( ) ( ) IC 7 -PIN 18 TO IC -PIN X
( ) ( ) IC 7 -PIN 19 TO--------- +5V
( ) ( ) IC 7 -PIN 20 TO--------- +5V

-55-
IC 8  Name 74245  Pins 20
( ) ( ) IC 8 -PIN 1 TO IC12 -PIN 4
( ) ( ) IC 8 -PIN 2 TO IC -PIN X
( ) ( ) IC 8 -PIN 3 TO IC -PIN X
( ) ( ) IC 8 -PIN 4 TO IC -PIN X
( ) ( ) IC 8 -PIN 5 TO IC -PIN X
( ) ( ) IC 8 -PIN 6 TO IC -PIN X
( ) ( ) IC 8 -PIN 7 TO IC -PIN X
( ) ( ) IC 8 -PIN 8 TO IC -PIN X
( ) ( ) IC 8 -PIN 9 TO IC -PIN X
( ) ( ) IC 8 -PIN 10 TO ------- GND
( ) ( ) IC 8 -PIN 11 TO IC -PIN X
( ) ( ) IC 8 -PIN 12 TO IC -PIN X
( ) ( ) IC 8 -PIN 13 TO IC -PIN X
( ) ( ) IC 8 -PIN 14 TO IC -PIN X
( ) ( ) IC 8 -PIN 15 TO IC -PIN X
( ) ( ) IC 8 -PIN 16 TO IC -PIN X
( ) ( ) IC 8 -PIN 17 TO IC -PIN X
( ) ( ) IC 8 -PIN 18 TO IC -PIN X
( ) ( ) IC 8 -PIN 19 TO IC11 -PIN 6
( ) ( ) IC 8 -PIN 20 TO ------- +5V X
<table>
<thead>
<tr>
<th>IC 9</th>
<th>Name 74154</th>
<th>Pins 24</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 1 TO IC -PIN X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 2 TO IC -PIN X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 3 TO IC -PIN X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 4 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 5 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 6 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 7 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 8 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 9 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 10 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 11 TO ---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 12 TO---------- GND</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 13 TO---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 14 TO---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 15 TO---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 16 TO---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 17 TO---------- NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 18 TO IC -PIN X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 19 TO IC11-PIN 3 &amp; IC13 PIN 19</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 20 TO IC13-PIN 20 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 21 TO IC13-PIN 21 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 22 TO IC13-PIN 22 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 23 TO IC13-PIN 23 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 9 -PIN 24 TO--------- +5V</td>
<td></td>
</tr>
</tbody>
</table>

-57-
IC 10  Name 1489  Pins 14

( ) ( ) IC 10 -PIN 1 TO-------- INPUT WIRE
( ) ( ) IC 10 -PIN 2 TO-------- NC
( ) ( ) IC 10 -PIN 3 TO IC18-PIN 2
( ) ( ) IC 10 -PIN 4 TO-------- NC
( ) ( ) IC 10 -PIN 5 TO-------- NC
( ) ( ) IC 10 -PIN 6 TO-------- NC
( ) ( ) IC 10 -PIN 7 TO-------- GND
( ) ( ) IC 10 -PIN 8 TO-------- NC
( ) ( ) IC 10 -PIN 9 TO-------- NC
( ) ( ) IC 10 -PIN 10 TO-------- NC
( ) ( ) IC 10 -PIN 11 TO-------- NC
( ) ( ) IC 10 -PIN 12 TO-------- NC
( ) ( ) IC 10 -PIN 13 TO-------- NC
( ) ( ) IC 10 -PIN 14 TO-------- +5V
IC 11  Name 7400  Pins 14

( ) ( ) IC 11 -PIN 1 TO IC11-PIN 4 X
( ) ( ) IC 11 -PIN 2 TO IC16-PIN 7 & IC18 PIN 14
( ) ( ) IC 11 -PIN 3 TO IC -PIN X
( ) ( ) IC 11 -PIN 4 TO IC -PIN X
( ) ( ) IC 11 -PIN 5 TO IC11-PIN 14
( ) ( ) IC 11 -PIN 6 TO IC -PIN X
( ) ( ) IC 11 -PIN 7 TO---------- GND
( ) ( ) IC 11 -PIN 8 TO---------- NC
( ) ( ) IC 11 -PIN 9 TO---------- NC
( ) ( ) IC 11 -PIN 10 TO---------- NC
( ) ( ) IC 11 -PIN 11 TO---------- NC
( ) ( ) IC 11 -PIN 12 TO---------- NC
( ) ( ) IC 11 -PIN 13 TO---------- NC
( ) ( ) IC 11 -PIN 14 TO 1C -PIN +5V X
IC 12  Name 7404  Pins 14

( ) ( ) IC 12 -PIN 1 TO IC -PIN X
( ) ( ) IC 12 -PIN 2 TO IC13-PIN 18
( ) ( ) IC 12 -PIN 3 TO IC16-PIN 5 X
( ) ( ) IC 12 -PIN 4 TO IC -PIN X
( ) ( ) IC 12 -PIN 5 TO-------- NC
( ) ( ) IC 12 -PIN 6 TO-------- NC
( ) ( ) IC 12 -PIN 7 TO-------- GND
( ) ( ) IC 12 -PIN 8 TO-------- NC
( ) ( ) IC 12 -PIN 9 TO-------- NC
( ) ( ) IC 12 -PIN 10 TO-------- NC
( ) ( ) IC 12 -PIN 11 TO-------- NC
( ) ( ) IC 12 -PIN 12 TO-------- NC
( ) ( ) IC 12 -PIN 13 TO-------- NC
( ) ( ) IC 12 -PIN 14 TO-------- +5V
IC 13  Name 74154  Pins 24

( ) ( ) IC 13 -PIN 1 TO---------- NC
( ) ( ) IC 13 -PIN 2 TO---------- NC
( ) ( ) IC 13 -PIN 3 TO---------- NC
( ) ( ) IC 13 -PIN 4 TO---------- NC
( ) ( ) IC 13 -PIN 5 TO---------- NC
( ) ( ) IC 13 -PIN 6 TO---------- NC
( ) ( ) IC 13 -PIN 7 TO---------- NC
( ) ( ) IC 13 -PIN 8 TO---------- NC
( ) ( ) IC 13 -PIN 9 TO---------- NC
( ) ( ) IC 13 -PIN 10 TO---------- NC
( ) ( ) IC 13 -PIN 11 TO---------- NC
( ) ( ) IC 13 -PIN 12 TO---------- GND
( ) ( ) IC 13 -PIN 13 TO IC15-PIN 23
( ) ( ) IC 13 -PIN 14 TO IC14-PIN 23
( ) ( ) IC 13 -PIN 15 TO IC18-PIN 9
( ) ( ) IC 13 -PIN 16 TO---------- NC
( ) ( ) IC 13 -PIN 17 TO IC -PIN X
( ) ( ) IC 13 -PIN 18 TO IC -PIN X
( ) ( ) IC 13 -PIN 19 TO IC -PIN X
( ) ( ) IC 13 -PIN 20 TO IC -PIN X
( ) ( ) IC 13 -PIN 21 TO IC -PIN X
( ) ( ) IC 13 -PIN 22 TO IC -PIN X
( ) ( ) IC 13 -PIN 23 TO IC -PIN X
( ) ( ) IC 13 -PIN 24 TO---------- +5V
<table>
<thead>
<tr>
<th>IC 14</th>
<th>Name MC6821</th>
<th>Pins 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 1 TO ------------ GND</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 2 TO IC22-PIN 1</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 3 TO IC22-PIN 16</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 4 TO IC22-PIN 2</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 5 TO IC22-PIN 15</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 6 TO IC22-PIN 3</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 7 TO IC22-PIN 14</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 8 TO IC22-PIN 4</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 9 TO IC22-PIN 13</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 10 TO ------------ DIP 2 SWITCH 1</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 11 TO ------------ DIP 2 SWITCH 2</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 12 TO ------------ DIP 2 SWITCH 3</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 13 TO ------------ DIP 2 SWITCH 4</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 14 TO ------------ DIP 2 SWITCH 5</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 15 TO ------------ DIP 2 SWITCH 6</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 16 TO ------------ DIP 2 SWITCH 7</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 17 TO ------------ DIP 2 SWITCH 8</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 18 TO ------------ NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 19 TO ------------ NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 20 TO IC14-PIN 22</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 21 TO IC15-PIN 21 &amp; IC18 PIN 14</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 22 TO IC14-PIN 24 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 23 TO IC -PIN X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 24 TO IC -PIN X +5V</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 25 TO IC15-PIN 25 &amp; IC18 PIN 14</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 26 TO IC15-PIN 26 &amp; IC18 PIN 15</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 27 TO IC15-PIN 27 &amp; IC18 PIN 16</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 28 TO IC15-PIN 28 &amp; IC18 PIN 17</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 29 TO IC15-PIN 29 &amp; IC18 PIN 18</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 30 TO IC15-PIN 30 &amp; IC18 PIN 19</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 31 TO IC15-PIN 31 &amp; IC18 PIN 20</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 32 TO IC15-PIN 32 &amp; IC18 PIN 21</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 33 TO IC15-PIN 33 &amp; IC18 PIN 22</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 34 TO IC15-PIN 34 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 35 TO IC15-PIN 35 X</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 36 TO IC15-PIN 36 &amp; IC18 PIN 11</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 37 TO ------------ NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 38 TO ------------ NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 39 TO ------------ NC</td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 14 -PIN 40 TO ------------ NC</td>
<td></td>
</tr>
</tbody>
</table>
IC 15 Name MC6821 Pins 40

( ) ( ) IC 15 -PIN 1 TO------- GND
( ) ( ) IC 15 -PIN 2 TO IC16-PIN 2
( ) ( ) IC 15 -PIN 3 TO IC16-PIN 4
( ) ( ) IC 15 -PIN 4 TO IC16-PIN 6
( ) ( ) IC 15 -PIN 5 TO IC16-PIN 8
( ) ( ) IC 15 -PIN 6 TO IC16-PIN 11
( ) ( ) IC 15 -PIN 7 TO IC16-PIN 17
( ) ( ) IC 15 -PIN 8 TO-------- NC
( ) ( ) IC 15 -PIN 9 TO-------- NC
( ) ( ) IC 15 -PIN 10 TO IC17-PIN 2
( ) ( ) IC 15 -PIN 11 TO IC17-PIN 4
( ) ( ) IC 15 -PIN 12 TO IC17-PIN 6
( ) ( ) IC 15 -PIN 13 TO IC17-PIN 8
( ) ( ) IC 15 -PIN 14 TO IC17-PIN 11
( ) ( ) IC 15 -PIN 15 TO IC17-PIN 13
( ) ( ) IC 15 -PIN 16 TO IC17-PIN 15
( ) ( ) IC 15 -PIN 17 TO IC?-PIN 17
( ) ( ) IC 15 -PIN 18 TO-------- NC
( ) ( ) IC 15 -PIN 19 TO-------- NC
( ) ( ) IC 15 -PIN 20 TO-------- +5V
( ) ( ) IC 15 -PIN 21 TO IC -PIN X
( ) ( ) IC 15 -PIN 22 TO IC15-PIN 24
( ) ( ) IC 15 -PIN 23 TO IC -PIN X
( ) ( ) IC 15 -PIN 24 TO IC -PIN X +5V
( ) ( ) IC 15 -PIN 25 TO IC -PIN X
( ) ( ) IC 15 -PIN 26 TO IC -PIN X
( ) ( ) IC 15 -PIN 27 TO IC -PIN X
( ) ( ) IC 15 -PIN 28 TO IC -PIN X
( ) ( ) IC 15 -PIN 29 TO IC -PIN X
( ) ( ) IC 15 -PIN 30 TO IC -PIN X
( ) ( ) IC 15 -PIN 31 TO IC -PIN X
( ) ( ) IC 15 -PIN 32 TO IC -PIN X
( ) ( ) IC 15 -PIN 33 TO IC -PIN X
( ) ( ) IC 15 -PIN 34 TO IC -PIN X
( ) ( ) IC 15 -PIN 35 TO IC -PIN X
( ) ( ) IC 15 -PIN 36 TO IC -PIN X
( ) ( ) IC 15 -PIN 37 TO-------- NC
( ) ( ) IC 15 -PIN 38 TO-------- NC
( ) ( ) IC 15 -PIN 39 TO-------- NC
( ) ( ) IC 15 -PIN 40 TO-------- NC
IC 16  Name 74241  Pins 20

( ) ( ) IC 16 -PIN 1 TO---------- GND
( ) ( ) IC 16 -PIN 2 TO IC -PIN
( ) ( ) IC 16 -PIN 3 TO IC21-PIN 10
( ) ( ) IC 16 -PIN 4 TO IC -PIN X
( ) ( ) IC 16 -PIN 5 TO IC18-PIN 13 X
( ) ( ) IC 16 -PIN 6 TO IC -PIN X
( ) ( ) IC 16 -PIN 7 TO IC -PIN X
( ) ( ) IC 16 -PIN 8 TO IC -PIN X
( ) ( ) IC 16 -PIN 9 TO IC21-PIN 11
( ) ( ) IC 16 -PIN 10 TO---------- GND
( ) ( ) IC 16 -PIN 11 TO IC -PIN X
( ) ( ) IC 16 -PIN 12 TO IC21-PIN 11
( ) ( ) IC 16 -PIN 13 TO IC -PIN X
( ) ( ) IC 16 -PIN 14 TO IC21-PIN 6
( ) ( ) IC 16 -PIN 15 TO IC -PIN X
( ) ( ) IC 16 -PIN 16 TO IC21-PIN 12
( ) ( ) IC 16 -PIN 17 TO IC -PIN X
( ) ( ) IC 16 -PIN 18 TO IC21-PIN 5
( ) ( ) IC 16 -PIN 19 TO IC16-PIN 20
( ) ( ) IC 16 -PIN 20 TO---------- +5V
IC 17  Name 74241    Pins 20

( ) ( ) IC 17 -PIN 1 TO-------- GND
( ) ( ) IC 17 -PIN 2 TO IC -PIN X
( ) ( ) IC 17 -PIN 3 TO IC21-PIN 13
( ) ( ) IC 17 -PIN 4 TO IC -PIN X
( ) ( ) IC 17 -PIN 5 TO IC21-PIN 4
( ) ( ) IC 17 -PIN 6 TO IC -PIN X
( ) ( ) IC 17 -PIN 7 TO IC21-PIN 14
( ) ( ) IC 17 -PIN 8 TO IC -PIN X
( ) ( ) IC 17 -PIN 9 TO IC21-PIN 3
( ) ( ) IC 17 -PIN 10 TO-------- GND
( ) ( ) IC 17 -PIN 11 TO IC -PIN X
( ) ( ) IC 17 -PIN 12 TO IC21-PIN 15
( ) ( ) IC 17 -PIN 13 TO IC -PIN X
( ) ( ) IC 17 -PIN 14 TO IC21-PIN 2
( ) ( ) IC 17 -PIN 15 TO IC -PIN X
( ) ( ) IC 17 -PIN 16 TO IC21-PIN 16
( ) ( ) IC 17 -PIN 17 TO IC -PIN X
( ) ( ) IC 17 -PIN 18 TO IC21-PIN 1
( ) ( ) IC 17 -PIN 19 TO IC17-PIN 20
( ) ( ) IC 17 -PIN 20 TO-------- +5V X
<table>
<thead>
<tr>
<th>IC 18</th>
<th>Name MC6850</th>
<th>Pins 24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 1</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 2</td>
<td>TO IC20-PIN 3</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 3</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 4</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 5</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 6</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 7</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 8</td>
<td>TO IC18-PIN 10 &amp; +5V</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 9</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 10</td>
<td>TO IC18-PIN 12</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 11</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 12</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 13</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 14</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 15</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 16</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 17</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 18</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 19</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 20</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 21</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 22</td>
<td>TO IC</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 23</td>
<td>TO-------</td>
</tr>
<tr>
<td>( ) ( )</td>
<td>IC 18 -PIN 24</td>
<td>TO-------</td>
</tr>
</tbody>
</table>

-66-
IC 19  Name MC14411  Pins 24
( ) ( ) IC 19 -PIN 1 TO----------DIP1 SWITCH 8
( ) ( ) IC 19 -PIN 2 TO ---------DIP1 SWITCH 7
( ) ( ) IC 19 -PIN 3 TO ---------DIP1 SWITCH 6
( ) ( ) IC 19 -PIN 4 TO ---------DIP1 SWITCH 5
( ) ( ) IC 19 -PIN 5 TO ---------DIP1 SWITCH 4
( ) ( ) IC 19 -PIN 6 TO ---------NC
( ) ( ) IC 19 -PIN 7 TO ---------DIP1 SWITCH 3
( ) ( ) IC 19 -PIN 8 TO ---------DIP1 SWITCH 2
( ) ( ) IC 19 -PIN 9 TO ---------NC
( ) ( ) IC 19 -PIN 10 TO IC19-PIN 22
( ) ( ) IC 19 -PIN 11 TO---------NC
( ) ( ) IC 19 -PIN 12 TO---------GND
( ) ( ) IC 19 -PIN 13 TO---------DIP1 SWITCH 1
( ) ( ) IC 19 -PIN 14 TO---------NC
( ) ( ) IC 19 -PIN 15 TO---------NC
( ) ( ) IC 19 -PIN 16 TO---------NC
( ) ( ) IC 19 -PIN 17 TO---------NC
( ) ( ) IC 19 -PIN 18 TO---------NC
( ) ( ) IC 19 -PIN 19 TO---------NC
( ) ( ) IC 19 -PIN 20 TO>>
   >>1 M OHM PARALLEL 1.8432 MHz XTAL
( ) ( ) IC 19 -PIN 21 TO>>
( ) ( ) IC 19 -PIN 22 TO IC19-PIN 24 X
( ) ( ) IC 19 -PIN 23 TO---------GND
( ) ( ) IC 19 -PIN 24 TO---------+5V

-67-
IC 20  Name MC1488  Pins 14

( ) ( ) IC 20 -PIN 1 TO -----------MINUS 12 V BYPASS 22 MFD TO GND
( ) ( ) IC 20 -PIN 2 TO -----------X
( ) ( ) IC 20 -PIN 3 TO -----------OUTPUT WIRE RS232C
( ) ( ) IC 20 -PIN 4 TO -----------NC
( ) ( ) IC 20 -PIN 5 TO -----------NC
( ) ( ) IC 20 -PIN 6 TO -----------NC
( ) ( ) IC 20 -PIN 7 TO -----------GND
( ) ( ) IC 20 -PIN 8 TO -----------NC
( ) ( ) IC 20 -PIN 9 TO -----------NC
( ) ( ) IC 20 -PIN 10 TO----------NC
( ) ( ) IC 20 -PIN 11 TO----------NC
( ) ( ) IC 20 -PIN 12 TO----------NC
( ) ( ) IC 20 -PIN 13 TO----------NC
( ) ( ) IC 20 -PIN 14 TO---------- +12 V BYPASS 22 MFD TO GROUND

-68-
<table>
<thead>
<tr>
<th>IC 21</th>
<th>Name I/O</th>
<th>Pins 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 1 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 2 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 3 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 4 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 5 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 6 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 7 TO ---------NC</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 8 TO ---------NC</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 9 TO ---------NC</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 10 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 11 TO ---------X X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 12 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 13 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 14 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 15 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>( ) IC 21 -PIN 16 TO ---------X</td>
<td></td>
</tr>
<tr>
<td>IC 22</td>
<td>Name I/O</td>
<td>Pins 16</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 1 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 2 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 3 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 4 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 5 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 6 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 7 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 8 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 9 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 10 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 11 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 12 TO</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 13 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 14 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 15 TO</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>IC 22 -PIN 16 TO</td>
<td>X</td>
</tr>
</tbody>
</table>
This source program runs with the Zenith C-86 compiler on any Zenith (100, 150, or 240) or fully PC compatible machine. It creates a disk file containing a blank wire wrap list. The program prompts for the IC number, number of pins which that IC has, and the name of the IC. The program accepts output file name on the command line or prompts for one if one is not given on the command line. The program runs in a continuous loop until the escape sequence of "999" is given at the IC number prompt. The program double spaces all ICs with 24 pins or less. It places an ASCII form feed character at the end of each IC in the file. It does not append to the output file. If multiple invocations of the program are required they must be performed with separate output file names and appended externally.

#include <stdio.h> /* NAME: WWLIST.C */
main(argc,argv) /* Program makes a blank wire wrap list*/
    int argc;
    /* Accepts file name on command line */
    char *argv[2];
    /* WWLIST filename */
    /* prompts for IC number, number of pins & IC name*/

    int icnum, pinum, loop;
    char icname[10];
    FILE *fp, *fopen();

    if ( argc == 1 ) {
        printf("Enter Output File ==> ");
        scanf("%s", argv[1]);
    }

    fp = fopen(*++argv, "w");
    putc('>',fp);

    do {
        printf("\n\n\nEnter IC Number (999 to quit) ==> ");
        scanf("%d", &icnum);

        if (icnum != 999) {
            printf("\nEnter IC Name ==> ");
    }

    -71-
scanf("%s",icname);

printf("Enter number of pins ==> ");
scanf("%d", &pinum);

fprintf(fp,". ej; IC %d	Name %s	Pins %d
\n", icnum, icname, pinum);
printf("IC %d	Name %s	Pins %d
\n", icnum, icname, pinum);
printf("... printing to file [%s]\n", *argv);

for(loop=1; loop<=pinum; loop++) {
    if (pinum <= 24) fprintf(fp, "\n");
    fprintf(fp,"( ) ( ) IC %d -PIN %d TO IC -
    PIN\n", icnum, loop);
}

fprintf(fp, "\f");

}

}

while (icnum != 999);

fclose(fp);
return(0);
}

UPDATEWW.C operates on a file originally created by the WWLIST program to fill in the "where to" data. The program expects two arguments on the command line: the first is the name of the input file to be filled in; the second is the name of the output file to be created. "UPDATEWW inputfile outputfile"
The program does not prompt for these names from within the program, however this could easily be added by using the routine in WWLIST.C on the previous page. The program may be called repeatedly in case the job can not be completed at one sitting. It should be noted that at subsequent invocations of the program, the latest outputfile name is used as inputfile
and another name must be used for outputfile.

#include <stdio.h> /* UPDATEWW.C updates the pin file made by WWLIST */

main(argc, argv) /* FILEIN FILEOUT names on command line!!! */
{
    int argc;
    char *argv[];
    
    extern int fgetc();
    extern int fputc();
    extern int fputs();
    int ch,i,j,chl,ch2,flag;

    FILE *fpr, *fpw, *fopen();

    fpw = fopen(argv[2], "w"); /* write over existing file */

    fpr = fopen(argv[1], "r"); /*(*+++argv , "r") is same thing*/
        /* argv is pointer to 0th el */
        /* *argv is the 0th element */
        /* argv[0] is the 0th element */

    printf("\n%s\n", "ENTER IC NUMBER SPACE PIN NUMBER SPACE
COMMENTS CR");

    printf("%s\n", "FOR EXAMPLE>>>12 1 comments here CR<<< WOULD
BE FOR");

    printf("%s\n", "IC-12 PIN-1 comments here");

    printf("%s\n", "FOR JUST IC AND PIN ENTER IC NUMBER PIN
NUMBER CR");

    printf("%s\n", "FOR JUST A COMMENT FIELD ENTER SPACE COMMENTS
CR");

    printf("%s\n", "TO STOP INPUTING TYPE > AT THE END OF A
COMMENT FIELD");

-73-
ch2 = '\0';
ch1 = '\0';
flag = 0;

while ((ch = fgetc(fpr)) != '>') putc(ch, fpw);

while ((ch = fgetc(fpr)) != EOF)
{
    ch2 = ch1;
    ch1 = ch;

    if(ch2 == '\0') continue;

    if(ch2 != '\12' && flag == 1) continue;

    putc(ch2, stdout); putc(ch2, fpw);

    if(ch2 == '\12') { flag = 0; continue; }

    if(ch2 == '0') {
        ch = fgetc(stdin);

        if(ch != ' ') {
            putc(' ', stdout); putc(' ', fpw);
            putc('I', stdout); putc('I', fpw);
            putc('C', stdout); putc('C', fpw);
            putc('-', stdout); putc('-', fpw);
            putc(ch, stdout); putc(ch, fpw);
            ch = fgetc(stdin);
            putc(ch, stdout); putc(ch, fpw);

            if(ch != ' ') { ch = fgetc(stdin); /*this must be a space*/
              putc(ch, stdout); putc(ch, fpw);
            }
            putc('P', stdout); putc('P', fpw);
            putc('I', stdout); putc('I', fpw);
            putc('N', stdout); putc('N', fpw);
            putc('-', stdout); putc('-', fpw);
        }
    }

    if(ch != ' ') {
        ch = fgetc(stdin); /*this must be a space*/
        putc(ch, stdout); putc(ch, fpw);
    }
}

flag = 1;
if (flag == 0) {
    for (i = 1; i < 14; i++) {
       putc('-', stdout);
       putc('-', fpw);
    }

    while ((ch = fgetc(stdin)) != '\12') {
        putc(ch, stdout);
        putc(ch, fpw);
        if (ch == '>') goto done;

        flag = 1;
    }
}

putc(chl, stdout); putc(chl, fpw);

goto dn2;

done:
    while ((ch = fgetc(fpr)) != '\12') continue;
    putc(ch, fpw);
    while ((ch = fgetc(fpr)) != EOF) putc(ch, fpw);

dn2: fclose(fpr);
    fclose(fpw);
}
SS50 MICROPROCESSOR SYSTEM (6800):
A brief description of the mother board will be given before considering the actual central processing unit (CPU) board. Figure 4 is a schematic diagram of the mother board electronics. The mother board provides all the sockets into which the CPU card, memory cards, and all input/output (I/O) cards are placed. The mother board provides 50 single sided contacts at each of the main sockets. Fifty printed circuit lines connect all of the main sockets in parallel on the back side of the mother board. Sixteen of the lines are used as unidirectional address lines (the signal is put out by the microprocessor and received by all other cards). These sixteen lines allow the microprocessor to specify which (any one of 65,536 possible) memory location it wishes to communicate with. Eight of these lines are used as bidirectional data lines. These eight lines allow a number (between 0 and 255) to be transferred either from the microprocessor to a memory location or from a memory location to the microprocessor. A master reset line is provided with a 1K pull up resistor to +5V. When the master reset line is pulled low it performs a hardware reset of the 6800 microprocessor. A reset line is provided with a 470 ohm resistor to +5V. This reset line is driven low any time a master reset occurs. This line provides a clean TTL pulse to reset devices which must be provided with a reset pulse. VMA (valid memory address) is provided to tell the memory devices when the signals on the memory lines are valid requests (the microprocessor places random junk on the address lines during certain internal processes). A line is provided for each of the two clock signals used by the microprocessor for internal timing (phase-1 and phase-2). When the phase-2 signal is high and the VMA signal is high, then and only then is the memory cell selected by the address lines enabled for read or write. A read/write line is provided which allows the microprocessor to tell each memory device which is selected whether it is to memorize the signal on the data lines (write), or place the signal stored by it on the data lines (read). Lines are provided for both nonmaskable interrupt (NMI) and interrupt request (IRQ).
Both of these lines are pulled up to +5V with 6.8K resistors. External devices pull these lines down to interrupt the microprocessor. A halt line is tied to +5V through 470 ohms. If an external device wishes to put its signals on the address lines, it may pull this line low and wait for the bus available line to go high (microprocessor signifying that it has stopped and relinquished the address lines for external use). The remaining five lines which are used on the SS50 bus are five different baud rate clocks. A 16 X clock is provided for the following baud rates: 110, 150, 300, 600, and 1200. In this implementation the 110 baud line has been changed to 9600 baud. In addition to providing interconnect lines between SS50 sockets, the mother board also provides address decoding for eight SS30 I/O slots on the back of the mother board. The decoding circuitry uses address lines A2, A3, A4, A5, A12, A13, A14, and A15 from the CPU card and provides an active low select line at the front of each SS30 I/O socket. The first I/O slot (number zero) is selected for $8000, $8001, $8002, and $8003. The second slot (number one) is decoded for the next four addresses, etc., for all eight I/O slots. CPU A0 and A1 are buffered and supplied to all SS30 slots as RSO and RS1. Thus, four memory locations are available for use at each of the I/O slots (not always used) with a minimum of decode hardware required on the I/O boards. Two 8835s are used to buffer the 8 data lines between the SS50 and SS30 sockets. It should be noted that the I/O slots are not uniquely decoded (address lines A6 through A11 are not used in decoding). This causes each I/O port address to echo through a full 4K of memory (i.e. $8100 is the 3-same as $8000). In addition to the +8V bus, +12 volts and -12 volts are also distributed to both the SS50 and SS30 sockets. It is important to remember that VMA, phase-2, and all of the data lines (D0- D7) are inverted on the SS50 backplane. Inverting drivers correct the sense of these lines before they are used by memory boards, etc. The CPU board (schematic diagram, Figure 5) contains the microprocessor (Motorola 6800), 128 bytes of scratch pad random access memory (6810), 8K bytes of monitor read only memory (2716), and a crystal
controlled baud rate generator (14411). Buffering of data lines to and from the CPU card is done by a pair of four-bit bidirectional inverting line drivers (8835). The direction is controlled by a pair of 7455s, which take into account VMA, R/W, and whether or not the memory address selected is external to the CPU card. Buffering of the sixteen address lines is accomplished with three 74367 line drivers. These buffers are on all the time except when the microprocessor is halted. A halt signal causes the outputs of the 74367s to open (go to a high impedance state) so that an external device may use the address lines. A Motorola 14411 is used to generate baud rate clock signals. These are buffered by sections of a 7404 before going out to the SS50 and SS30 buses. Two 74139 decoder chips, in conjunction with four mechanical switches, are used to select: on-board or external RAM and 2K, 4K, or 8K of on-board monitor ROM. Figures 6 and 7 show the actual top and bottom view of the CPU card. In Figure 6 the four 2716s in the upper right hand corner of the board house the 8K of monitor programs. The option select switches are shown just below these ROMs. The wire shown on the back of the CPU card (Figure 7) is used to supply a 9600 baud sixteen times clock signal, after the 110 baud clock printed circuit line has been cut.

SERIAL INTERFACE BOARD:

Figure 8 is the schematic diagram for a serial interface card which is installed on port one of the mother board. This card is not used under operating conditions of the automatic test system. However, this card is very important when trouble shooting problems or creating a new system. This board allows one to run the monitor program (memory examine and change, etc.) and the disk operating system directly from any RS232-C terminal. As can be seen on Figures 9 and 10, only the components associated with RS232-C operation have been installed on this interface card. In particular all three of the 4N33 optical couplers have been left out, and the input to pin 4 of IC-3 was tied low (simulating no input from the 20 mA current loop input). Also a switch was installed on the top edge of the serial
board. This allows the baud rate (16 X clock) to be selected while the board is installed and the system operating. This is significant when a system problem is at hand, and one does not wish to lose all the memory information while connecting some arbitrary serial terminal to do trouble shooting.

STANDARD PARALLEL INTERFACE BOARD:
Figure 11 is the schematic diagram for the standard interface board. Figures 12 and 13 show the front and back of this board. Two of these are used in the system, however, only one is described because they are identical. These boards mount on any of the SS30 I/O slots on the back of the mother board. Plus 8 volts (unregulated) from the bus is converted to +5 volts (regulated) by a 7805. It is important that the filter capacitors on the input and output of the 7805 be as close to the regulator as possible (closer than 2 inches). The standard interface board uses a Motorola 6821 to interface to the computer, and five 8833 line drivers to interface between the 6821 and the external devices. The 8833 drivers have standard TTL driver outputs. In addition to setting the 6821 for either input or output (direction of signal flow) with the proper computer commands, a jumper on the circuit board must tell the 8833 line drivers to go in the same direction.

OPEN COLLECTOR PARALLEL INTERFACE BOARD:
Figure 14 is the schematic diagram for this board and Figures 15 and 16 show the front and back of the printed circuit card. This board mounts on any of the SS30 I/O slots on the back of the mother board. Plus 8 volts (unregulated) from the bus is converted to +5 volts (regulated) by a 7805. It is important that the filter capacitors on the input and output of the 7805 be as close to the regulator as possible (closer than 2 inches). The open collector interface board uses a Motorola 6821 to interface to the computer, and three 7407 open collector noninverting buffers to interface between the 6821 and the external devices. The 7407 drivers have high voltage open collector outputs. This
Figure 11
board is designed to provide outputs from the CPU only. However, the CPU must set the data direction register in the 6821 for data output.

SS50 SYSTEM MEMORY ALLOCATION:

Sixteen binary address lines allow the CPU to directly select any one memory byte out of 64K (a K is actually 1,024 bytes, the number of different codes possible with ten binary address lines).

The lowest 256 bytes ($0000-$00FF) of memory are reserved for temporary variable storage. This is highly desirable because only eight address lines are involved in selection. This feature allows the CPU to read or write to these locations much faster than possible at higher locations.

The memory from ($0100-$0FFF) is where the system measurement programs reside. These programs control the operation of all measurement hardware.

The memory from ($1000-$1FFF) is used for binary coded decimal data storage. This intermediate data storage is later read to magnetic disk when a test run is complete.

The memory from ($2000-$5000) is not used during normal system operation. This space is used to run DEBUG diagnostic programs when required.

The memory from ($6000-$67FF) is used by a system executive command file program. This EXEC program runs all the disk resident programs which are required to perform the life test.

The memory from ($6800-$6FFF) is used by a subroutine driver which stores and keeps track of memory pointers for all data stored by the system.

The disk operating system resides at ($7000-$7FFF). This system contains all the drivers which are required to read and write files to and from the magnetic disk.
drive.

Input and output boards are addressed at ($8000-$8FFF). Only the first 32 bytes are actually used, however, these are not completely decoded and echo through the entire block.

The CPU uses the memory at ($A000-$A0FF) as program push down stack. This keeps track of return locations for subroutines, etc.

The highest 8K of memory ($E000-$FFFF) is used by CPU monitor read only memory (ROM). Subroutine calls to and from a diagnostic terminal are made through this ROM.

Figures 17 and 18 show the top and bottom views of a representative memory board. The board shown is populated with 2K by 8 CMOS memory chips. It provides contiguous memory from $0000 to $7FFF with the sixteen chips on the right-hand side of the memory board. The other memory chips on this card are not used in the automatic life test system and will not be described here. The jumper shown in Figure 18 is part of a modification of the original memory board. This modification prevents the line drivers on the memory board from being selected at the $8000 block (where I/O is decoded in the measurement system).

POWER SUPPLY:

The system power supply provides the required regulated DC voltages. It should be noted that some of these voltages are further regulated by the circuit card where the power is used. This is done primarily to provide better noise isolation. The supply includes the necessary heat sinking and air cooling fan to enable continuous operation of the measurement system.
(Figure 17)
REAL-TIME CLOCK/CALENDAR SOFTWARE:

This assembly language program reads and sets the real-time clock/calendar in the SS50 microprocessor system. This program is derived from a program which was used initially to test the clock hardware. The hardware contains a ten-year battery, so that it maintains the correct time and date when power is removed from the SS50 system, or the clock board is removed from the system.

Figure 20 on page 107 shows a simple clock/calendar schematic without any system implications. Figure 21 on page 108 shows the clock/calendar chip (ICM7170) with associated buffer and decode hardware needed to run it on the SS50 system. This software was written for the SS50 implementation of Figure 21 on page 108. Figures 23 and 24 show top and bottom views of the actual wire wrap board.

```
8 PAGE 001 CLOCK
9
10
11
12
13 NAM CLOCK
14 OPT 0,S,NOP,NOG
15 ORG $0100
16 *
17 * THIS PROGRAM READS & SETS A REAL-TIME
18 * CLOCK LOCATED AT $C100 THRU $C111
19 *
20 * BECAUSE OF INVERTING BUFFERS
21 * ON THE CPU BOARD THE DATA COMES
22 * BACK INVERTED FROM THE CLOCK
23 *
24 *$C100 IS THE HUNDREDTHS COUNTER
25 ******************* ***
26 ****** $C100 MUST BE READ FIRST TO *****
27 ******************* ***
28 ***UPDATE THE LATCH FOR REST OF COUNTERS
29 **THESE REMAIN UNTIL $C100 IS READ AGAIN.
30 *
31 *$C1XX: 00=1/100 SEC AND LATCH
```

-98-
THE FOLLOWING EQU LINES DEFINE MEMORY LOCATIONS
IN TERMS OF NAMES, SO THE NAMES MAY CONVENIENTLY BE USED
THROUGHOUT THE REST OF THE PROGRAM

45  E1AC  IN  EQU  $E1AC  THIS SUB INPUTS ONE
         CHARACTER FROM THE COMPUTER TERMINAL
46  E07E  PDAT1 EQU  $E07E  THIS SUB PRINTS
         STRING TO THE COMPUTER SCREEN
47  C111  CMQR  EQU  $C111  CLOCK COMMAND
         REGISTER
48  C100  HSEC  EQU  $C100  HUNDREDTHS SECONDS
49  C103  SEC  EQU  $C103  SECONDS
50  C102  MIN  EQU  $C102  MINUTES
51  C101  HR  EQU  $C101  HOURS
52  C105  DAYM  EQU  $C105  DAY OF MONTH
53  C104  MMONT  EQU  $C104  MONTH

-99-
C106 YR EQU $C106 YEAR
C107 DAYW EQU $C107 DAY OF WEEK

*$F3 IN COMMAND REG IS NORMAL RUN
*$FB COMMAND STOPS CLOCK SO TIME
* CAN BE SET BY WRITING (COMP)
* OF DESIRED TIME IN C100 THRU C107

* START THE CLOCK BY WRITING $F3 IN THE
* COMMAND REG.

***THE ABOVE COMANDS ARE ALREADY INVERTED

**********THIS PROGRAM WILL READ THE CLOCK**********
********AND PRINT THE TIME TO THE TERMINAL**********
***************WHEN ANY KEY IS PRESSED***************

*EXCEPT Q WHICH QUILTS THE PROGRAM TO DOS
* OR S WHICH SETS THE TIME OF CLOCK

0100 CE 0202 START LDX #OPENS
0103 BD E07E JSR PDAT1 (PRINT STRING)

0106 A6 00 LDA A 0,X (THIS NOT REQUIRED)
0108 CE 023C LDX #STRING (POINT TO STRING)
010B BD E07E JSR PDAT1 (PRINT STRING)
010E CE C100 LDX #$C100 (CLOCK POINTER)
0111 A6 00 LDA A 0,X (1/100 SEC & LATCH)
0113 43 COM A (BUFFERS INVERTED THE DATA)
0114 B7 0380 STA A OUT+1
0117 A6 03 LDA A 3,X SECONDS
0119 43 COM A
011A B7 0382 STA A OUT+3
011D A6 02 LDA A 2,X MINUTES
011F 43 COM A

-100-
90 0120 B7 0384 STA A OUT+5
91 0123 A6 01 LDA A 1,X HOURS
92 0125 43 COM A
93 0126 B7 0386 STA A OUT+7
94 0129 A6 05 LDA A 5,X DAY OF MONTH
95 012B 43 COM A
96 012C B7 0388 STA A OUT+9
97 012F A6 04 LDA A 4,X MONTH
98 0131 43 COM A
99 0132 B7 038A STA A OUT+11
100 0135 A6 06 LDA A 6,X YEAR
101 0137 43 COM A
102 0138 B7 038C STA A OUT+13
103 013B C6 07 LDA B #7 NUMBER OF DATA ITEMS
104 013D F7 0391 STA B BTMP
105 0140 CE 037F LDX #OUT
106 0143 FF 038F STX XTMP
107 0149 C6 03 LDA B #3
108 014B BD 7133 JSR $7133 OUT-DEC
109 0150 BD E1D1 JSR $E1D1 OUT-CHARACTER
110 0153 FE 038F LDX XTMP
111 0156 08 INX
112 0157 08 INX
113 0158 FF 038F STX XTMP
114 015B 7A 0391 DEC BTMP
115 015E 26 E6 BNE LP1

121 *GET CHARACTER FROM TERMINAL
122 *IF Q THEN EXIT TO DOS
123 *IF S THEN SET TIME
124 0160 BD E1AC JSR $E1AC GET CHARACTER
***THIS SECTION SETS THE CLOCK TO A GIVEN TIME AND STARTS IT WHEN ANY CHARACTER IS INPUT FROM THE KEY BOARD

*IF ITEM IS TO STAY THE SAME
*JUST INPUT A CARRIAGE RETURN (CR)

LDA A #$FB STOPS CLOCK
STA A CMDR

LDX #HSS STRING FOR .01 SEC
JSR PDAT1
JSR INTIM
BCS ++5
STA A HSEC

LDX #SECS STRING FOR SECONDS
JSR PDAT1
JSR INTIM
BCS ++5
STA A SEC

LDX #MINS STRING FOR MINUTES
JSR PDAT1
JSR INTIM
BCS ++5
STA A MIN

LDX #HRS STRING FOR HOURS
JSR PDAT1
JSR INTIM
BCS ++5
STA A HR
162  01AE CE 0322   LDX  #DAYMS STRING FOR DAY
       OF MONTH
163  01B1 BD E07E   JSR  PDAT1
164  01B4 BD 01E3   JSR  INTIM
165  01B7 25 03     BCS  **5
166  01B9 B7 C105   STA A DAYM
167  *               
168  01BC CE 0346   LDX  #MONTH STRING FOR MONTH
169  01BF BD E07E   JSR  PDAT1
170  01C2 BD 01E3   JSR  INTIM
171  01C5 25 03     BCS  **5
172  01C7 B7 C104   STA A MMONT
173  *               
174  01CA CE 0363   LDX  #YRS
175  01CD BD E07E   JSR  PDAT1
176  01D0 BD 01E3   JSR  INTIM
177  01D3 25 03     BCS  **5
178  01D5 B7 C106   STA A YR
179  *               
180  *WAIT FOR ANY CHARACTER
181  01D8 BD E1AC   JSR  $E1AC INPUT CHARACTER
182  01DB 86 F3     LDA A #$F3 STARTS CLOCK
183  01DD B7 C111   STA A CMDR
184  01E0 7E 7103   JMP  $7103 SOFT ENTRY FOR DOS
185  *               
186  * INTIM IS SUB TO INPUT TIME ITEM
187  *SETS CARY AND RETURNS IF CR WAS INPUT
188  *THIS INDICATES NO CHANGE
189  *IT CLEARS CARRY AND RETURNS IF DATA INPUT
190  *
191  01E3 C6 00     INTIM LDA B #0
192  01E5 BD E1AC   JSR  IN
193  01E8 81 0D     CMP A #$D
194  01EA 26 02     BNE  AR22

-103-
195 01EC 0D    SEC
196 01ED 39    RTS
197 01EE 84 CF AR22 AND A #$CF
198 01F0 81 00 CMP A #0
199 01F2 27 05 BEQ AR21
200 01F4 CB 0A LP21 ADD B #10
201 01F6 4A    DEC A
202 01F7 26 FB BNE LP21
203 01F9 BD E1AC AR21 JSR IN
204 01FC 84 CF AND A #$CF
205 01FE 1B    ABA
206 01FF 43    COM A
207 0200 0C    CLC
208 0201 39    RTS

209

* STRINGS TO BE PRINTED TO SCREEN:

210 0202 0D    OPENS FCB $0D,$0A,$0A
211 0205 53    FCC 'S TO SET TIME, Q TO QUIT, ANYTHING ELSE TO READ TIME AGAIN

212 0239 0D    FCB $0D,$0A,4
213 023C 0D    STRING FCB $0D,$0A
214 023E 20    FCC '.01S, SEC, MIN, HRS, DATE,MONTH, YEAR'
215 0269 0D    FCB $0D,$0A,4
216 026C 0D    SECS FCB $0D,$0A
217 026E 49    FCC 'INPUT NUMBER OF SECONDS 2DIGITS NO CR (54),'
218 029A 04    FCB 4
219 029B 0D    HSS FCB $0D,$0A
220 029D 49    FCC 'INPUT NUMBER OF 1/100 SECONDS 2DIGITS NO CR (23),'
221 02CF 04    FCB 4
222 02D0 0D    MINS FCB $0D,$0A
223 02D2 49    FCC 'INPUT NUMBER OF MIN 2DIGITS NO CR (59),'
224 02FA 04    FCB 4

-104-
225 02FB 0D    HRS   FCB   $0D,$0A
226 02FD 49    FCC   'INPUT NUMBER OF HOURS
                 2DIGITS NO CR (23)?
227 0321 04    FCB   4
228 0322 0D    DAYMS FCB   $0D,$0A
229 0324 49    FCC   'INPUT DAY OF MONTH NO CR
                 2DIGITS (21)?'
230 0345 04    FCB   4
231 0346 0D    MONTH FCB   $0D,$0A
232 0348 49    FCC   'INPUT N_MONTH 2DIGITS NO CR
                 (11)? '
233 0362 04    FCB   4
234 0363 0D    YRS   FCB   $0D,$0A
235 0365 49    FCC   'INPUT YEAR 2DIGITS NO CR
                 (41)? '
236 037E 04    FCB   4
237 *           
238 037F 00    OUT   FCB   
                 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
239 038F 0002  XTMP RMB   2
240 0391 0001  BTMP RMB   1
241 END

SYMBOL TABLE
NAME      HEX LOCATION

242 IN      E1AC
243 PDAT1   E07E
244 CMDR    C111
245 HSEC    C100
246 SEC     C103
247 MIN     C102
248 HR      C101
249 DAYM    C105

-105-
TOTAL ERRORS 00000
(FIGURE 20)
SS50 SYSTEM ANALOG TO DIGITAL CONVERTER:

This program was assembled and tested on the SS50 (Motorola 6800) system. However, the code is fully compatible with the single board 6802 system. Figures 21 and 22 on pages 108 and 109 show the exact circuit which this program was developed for.

The CMOS single pole eight throw switch selects channel 0 through 9 with the same write command that starts conversion by the analog to digital converter. Binary data is read from the A/D. This data is a binary number between 0 and 255 representing zero volts and 255 representing five volts (actually in the SS50 system the data is inverted by an inverting buffer on the CPU board) so it must be complemented before use. After inversion the raw data is multiplied by two and a duplicate of the raw data is divided by thirty two. The raw/32 is then subtracted from the rawX2 to give the final binary number which represents hundredths of volts input to the A/D plus or minus three hundredths. This binary number is then converted to decimal and printed to the system monitor. The program reads the A/D again with a space input or exits to DOS if a capital Q is entered.

1 PAGE 001 AD1

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

11 E1AC IN EQU $E1AC INPUTS ONE CHAR
12 E07E PDAT1 EQU $E07E PRINTS STRING TO SCREEN
13 E1D1 OUT EQU $E1D1 PRINT ONE CHAR
14 C200 AD EQU $C200 A/D DECODE ADDRESS
15 7133 OUTD EQU $7133 OUT-DECIMAL X-POINT 2-BYTES
16 * B-FLAG (0=START FIRST NON 0)
17 * (ELSE=PAD WITH SPACE)
* This program reads an analog to digital (A/D) converter which is decoded at $C200 on the SS50 system.
* Eight A/D input lines are connected to a single 0804 A/D converter by a 4051 CMOS switch.
* Figure 21 on page 108 shows the initial decoding for a negative going chip select for the A/D at $C2XX (XX represents doesn't care). This schematic also indicates how not read and not write signals are generated.
* Figure 22 on page 109 is the schematic diagram for the 4051 CMOS switch (IC-10) and 0804 A/D converter (IC-8). This diagram also shows a 74273 (IC-9) which is used to latch the line select data into the 4051 switch.
* Two things occur when a number is written into memory location $C200: first the number written is latched into 74273 to select one of the eight A/D input lines. Second the not write pulse, when $C2XX is selected, starts conversion by the 0804 A/D (IC-8).
* After conversion, location $C200 may be read to obtain the voltage data from the A/D converter. It
* should be noted that a not read
* pulse is required by the 0804
* (IC-8).

0100 CE 019B START LDX #ST2
0103 BD E07E JSR PDAT1
0106 BD E1AC JSR IN
0109 80 30 SUB A #$30
010B 16 TAB
010C 86 FF LDA A #$Fr'
010D 10 SBA
010F B7 C200 STA A $C200 *THIS SELECTS
LINE 0 THRU 9
AND STARTS
CONVERSION
0112 BD 015E JSR DLY1 *WAIT FOR
CONVERSION
0115 B6 C200 LDA A $C200 *THIS GETS
DATA
0118 43 COM A *INVERTED
BUFFERS
0119 48 ASL A
011A 97 01 STA A 1 *PUTS DATA
IN MEM 1
011C B6 C200 LDA A $C200 *THIS GETS
DATA
011F 43 COM A *INVERTED
BUFFERS
0120 44 LSR A
0121 44 LSR A
0122 44 LSR A
0123 44 LSR A
0124 44 LSR A
0125 44 LSR A
0126 44 LSR A
0127 97 00 STA A 0

*
LDA B $C200  *THIS GETS DATA

COM B  *INVERTED BUFFERS

LSR B
LSR B
LSR B
LSR B

*NOW HAVE (A/D)/32 IN ACC B

LDA A 1
SBA
STA A 1

*5V=510-7 (503)

LDA A 0
SBC A #0
STA A 0

*PRINT DECIMAL EQUIVALENT OF
*TWO TIMES A/D VALUE -(1/32 A/D)
*A/D INPUT VOLTAGE IN .01 VOLTS

JSR CRLF
LDX #0 POINT TO BINARY VALUE
LDA B #1 PAD SPACES
JSR OUTD
LDX #ST1
JSR PDAT1
JSR CRLF

-115-
114
115
116
117
118
119
120
121
122
123
124
125
126
127

128
129

130 0151 BD E1AC JSR IN
131 0154 81 51 CMP A #$51
132 0156 27 03 BEQ DN1
133 0158 7E 0100 JMP START
134 015B 7E 7103 DN1 JMP $7103 DOS SOFT ENTRY

135

136 015E CE 0020 DLY1 LDX #$20
137 0161 09 LP1 DEX
138 0162 26 FD BNE LP1
139 0164 39 RTS

140

141 0165 20 ST1 FCC ' .01s VOLT +/-3UNITS'
142 0179 0D FCB $0D,$0A
143 017B 51 FCC 'Q TO QUIT OR SPACE TO RUN AGAIN'
144 019A 04 FCB 4
145

-116-
146 019B 0D ST2  FCB  $D,$A
147 019D 20  FCC ' SELECT CHANNEL 0 THRU 7 NO-CR'
148 01BA 04  FCB  4
149

Symbol Table

150    IN   E1AC
151   PDAT1  E07E
152   OUT   E1D1
153    AD   C200
154  OUTD   7133
155  OUTH   7139
156   CRLF  711E
157   DOS   7103
158  START  0100
159  PRINT  013D
160    DN1  015B
161  DLY1  015E
162    LP1  0161
163   ST1   0165
164   ST2   019B
165
166  TOTAL ERRORS 00000
167   EOF